

mlpack

1.0.12

Generated by Doxygen 1.8.9.1

Fri Sep 25 2015 08:06:54

Contents

1	MLPACK Documentation	1
1.1	Introduction	1
1.2	How To Use This Documentation	1
1.3	Executables	1
1.4	Tutorials	2
1.5	Methods in MLPACK	2
1.6	Final Remarks	3
2	Building MLPACK From Source	5
2.1	Introduction	5
2.2	latest mlpack build	5
2.3	Creating Build Directory	5
2.4	Dependencies of MLPACK	5
2.5	Configuring CMake	6
2.6	Building MLPACK	6
2.7	Installing MLPACK	7
3	MLPACK Input and Output	9
3.1	Introduction	9
3.2	Simple Logging Example	9
3.3	Simple CLI Example	10
4	Matrices in MLPACK	13
4.1	Introduction	13
4.2	Column-wise Matrices	13
4.3	Loading Matrices	13
5	Simple Sample MLPACK Programs	15
5.1	Introduction	15
5.2	Covariance Computation	15

5.3	Nearest Neighbor	15
5.4	Other examples	16
6	MLPACK Timers	17
6.1	Introduction	17
6.2	Timer API	17
6.3	Timer Example	17
7	mlpack version information	19
7.1	mlpack versions in code	19
7.2	mlpack executable versions	19
8	Alternating Matrix Factorization tutorial.	21
8.1	Introduction	21
8.2	Table of Contents	21
8.3	The 'AMF' class	21
8.3.1	Using different termination policies	22
8.3.2	Using different initialization policies	22
8.3.3	Using different update rules	23
8.3.4	Using Non-Negative Matrix Factorization with AMF	23
8.3.5	Using Singular Value Decomposition with AMF	24
8.4	Further documentation	24
9	Density Estimation Tree (DET) tutorial	25
9.1	Introduction	25
9.2	Table of Contents	25
9.3	Command-Line 'det'	26
9.3.1	Plain-vanilla density estimation	27
9.3.2	Estimation on a test set	27
9.3.3	Printing a trained DET	27
9.3.4	Computing the variable importance	28
9.3.5	Leaf Membership	28
9.4	The 'DTree' class	28
9.4.1	Public Functions	28
9.5	'namespace mlpack::det'	29
9.5.1	Utility Functions	29
9.6	Further Documentation	30
10	EMST Tutorial	31

10.1 Introduction	31
10.2 Table of Contents	31
10.3 Command-Line 'EMST'	32
10.4 The 'DualTreeBoruvka' class	33
10.5 Further documentation	33
11 Fast max-kernel search tutorial (fastmks)	35
11.1 Introduction	35
11.2 Table of Contents	35
11.3 Command-line FastMKS (fastmks)	36
11.3.1 FastMKS with a linear kernel on one dataset	37
11.3.2 FastMKS on a reference and query dataset	37
11.3.3 FastMKS with a different kernel	37
11.3.4 Using single-tree search or naive search	38
11.3.5 Paramters for alternate kernels	38
11.4 The 'FastMKS' class	38
11.4.1 FastMKS on one dataset	39
11.4.2 FastMKS with a query and reference dataset	39
11.4.3 FastMKS with an initialized kernel	39
11.4.4 FastMKS with an already-created tree	40
11.5 Writing a custom kernel for FastMKS	41
11.6 Using other tree types for FastMKS	41
11.7 Running FastMKS on objects	41
11.8 Further documentation	42
12 K-Means tutorial (kmeans)	43
12.1 Introduction	43
12.2 Table of Contents	44
12.3 Command-Line 'kmeans'	44
12.3.1 Simple k-means clustering	44
12.3.2 Saving the resulting centroids	45
12.3.3 Allowing empty clusters	45
12.3.4 Limiting the maximum number of iterations	45
12.3.5 Setting the overclustering factor	45
12.3.6 Using Bradley-Fayyad "refined start"	45
12.4 The 'KMeans' class	46
12.4.1 Running k-means and getting cluster assignments	46
12.4.2 Running k-means and getting centroids of clusters	46

12.4.3	Limiting the maximum number of iterations	47
12.4.4	Setting the overclustering factor	47
12.4.5	Setting initial cluster assignments	47
12.4.6	Setting initial cluster centroids	48
12.4.7	Running sparse k-means	49
12.5	Template parameters for the 'KMeans' class	49
12.5.1	Changing the distance metric used for k-means	49
12.5.2	Changing the initial partitioning strategy used for k-means	50
12.5.3	Changing the action taken when an empty cluster is encountered	51
12.6	Further documentation	51
13	Linear/ridge regression tutorial (linear_regression)	53
13.1	Introduction	53
13.2	Table of Contents	53
13.3	Command-Line 'linear_regression'	54
13.3.1	One file, generating the function coefficients	54
13.3.2	Compute model and predict at the same time	55
13.3.3	Prediction using a precomputed model	55
13.3.4	Using ridge regression	56
13.4	The 'LinearRegression' class	57
13.4.1	Generating a model	57
13.4.2	Setting a model	57
13.4.3	Load a model from a file	57
13.4.4	Prediction	57
13.4.5	Setting lambda for ridge regression	58
13.5	Further documentation	58
14	NeighborSearch tutorial (k-nearest-neighbors)	59
14.1	Introduction	59
14.2	Table of Contents	59
14.3	Command-Line 'allknn'	60
14.3.1	One dataset, 5 nearest neighbors	60
14.3.2	Query and reference dataset, 10 nearest neighbors	61
14.3.3	One dataset, 3 nearest neighbors, leaf size of 15 points	61
14.4	The 'AllkNN' class	62
14.4.1	5 nearest neighbors on a single dataset	62
14.4.2	10 nearest neighbors on a query and reference dataset	62
14.4.3	Naive (exhaustive) search for 6 nearest neighbors on one dataset	62

14.5 The extensible 'NeighborSearch' class	63
14.5.1 SortPolicy policy class	63
14.5.2 MetricType policy class	63
14.5.3 TreeType policy class	64
14.6 Further documentation	64
15 RangeSearch tutorial (range_search)	65
15.1 Introduction	65
15.2 Table of Contents	65
15.3 The 'range_search' command-line executable	66
15.3.1 One dataset, points with distance ≤ 0.01	66
15.3.2 Query and reference dataset, range [1.0, 1.5]	67
15.3.3 One dataset, range [4.1 4.2], leaf size of 15 points	67
15.4 The 'RangeSearch' class	68
15.4.1 Distance less than 2.0 on a single dataset	68
15.4.2 Range [3.0, 4.0] on a query and reference dataset	69
15.4.3 Naive (exhaustive) search for distance greater than 5.0 on one dataset	69
15.5 The extensible 'RangeSearch' class	69
15.5.1 MetricType policy class	69
15.5.2 TreeType policy class	70
15.6 Further documentation	70
16 Tutorials	71
16.1 Introductory Tutorials	71
16.2 Method-specific Tutorials	71
17 Bug List	73
18 Namespace Index	75
18.1 Namespace List	75
19 Class Index	77
19.1 Class List	77
20 File Index	85
20.1 File List	85
21 Namespace Documentation	89
21.1 mlpack Namespace Reference	89
21.1.1 Detailed Description	90

21.2	mlpack::amf Namespace Reference	90
21.2.1	Detailed Description	91
21.2.2	Function Documentation	92
21.2.2.1	SVDBatchLearning::HUpdate< arma::sp_mat >	92
21.2.2.2	SVDBatchLearning::WUpdate< arma::sp_mat >	92
21.2.2.3	SVDIncompleteIncrementalLearning::HUpdate< arma::sp_mat >	92
21.2.2.4	SVDIncompleteIncrementalLearning::WUpdate< arma::sp_mat >	92
21.3	mlpack::bound Namespace Reference	92
21.4	mlpack::cf Namespace Reference	92
21.4.1	Detailed Description	92
21.5	mlpack::data Namespace Reference	93
21.5.1	Detailed Description	93
21.5.2	Function Documentation	93
21.5.2.1	Load	93
21.5.2.2	NormalizeLabels	94
21.5.2.3	RevertLabels	94
21.5.2.4	Save	94
21.6	mlpack::decision_stump Namespace Reference	95
21.7	mlpack::det Namespace Reference	95
21.7.1	Detailed Description	96
21.7.2	Function Documentation	96
21.7.2.1	PrintLeafMembership	96
21.7.2.2	PrintVariableImportance	96
21.7.2.3	Trainer	96
21.8	mlpack::distribution Namespace Reference	97
21.8.1	Detailed Description	97
21.9	mlpack::emst Namespace Reference	97
21.9.1	Detailed Description	97
21.10	mlpack::fastmks Namespace Reference	98
21.10.1	Detailed Description	98
21.11	mlpack::gmm Namespace Reference	98
21.11.1	Detailed Description	99
21.11.2	Function Documentation	99
21.11.2.1	phi	99
21.11.2.2	phi	99
21.11.2.3	phi	100
21.11.2.4	phi	100

21.12mlpack::hmm Namespace Reference	100
21.12.1 Detailed Description	101
21.12.2 Function Documentation	101
21.12.2.1 LoadHMM	101
21.12.2.2 SaveHMM	101
21.13mlpack::kernel Namespace Reference	101
21.13.1 Detailed Description	103
21.14mlpack::kmeans Namespace Reference	103
21.14.1 Detailed Description	103
21.15mlpack::kpca Namespace Reference	103
21.16mlpack::lcc Namespace Reference	104
21.17mlpack::math Namespace Reference	104
21.17.1 Detailed Description	105
21.17.2 Function Documentation	105
21.17.2.1 Center	105
21.17.2.2 ClampNonNegative	105
21.17.2.3 ClampNonPositive	106
21.17.2.4 ClampRange	106
21.17.2.5 Orthogonalize	106
21.17.2.6 Orthogonalize	106
21.17.2.7 RandInt	106
21.17.2.8 RandInt	107
21.17.2.9 RandNormal	107
21.17.2.10RandNormal	107
21.17.2.11Random	107
21.17.2.12Random	107
21.17.2.13RandomSeed	107
21.17.2.14RandomVector	108
21.17.2.15RemoveRows	108
21.17.2.16VectorPower	108
21.17.2.17WhitenUsingEig	108
21.17.2.18WhitenUsingSVD	108
21.17.3 Variable Documentation	108
21.17.3.1 randGen	108
21.17.3.2 randNormalDist	108
21.17.3.3 randUniformDist	108
21.18mlpack::metric Namespace Reference	109

21.18.1 Typedef Documentation	109
21.18.1.1 ChebyshevDistance	109
21.18.1.2 EuclideanDistance	109
21.18.1.3 ManhattanDistance	109
21.18.1.4 SquaredEuclideanDistance	109
21.19mlpack::mvu Namespace Reference	109
21.20mlpack::naive_bayes Namespace Reference	109
21.20.1 Detailed Description	110
21.21mlpack::nca Namespace Reference	110
21.21.1 Detailed Description	110
21.22mlpack::neighbor Namespace Reference	110
21.22.1 Detailed Description	111
21.22.2 Typedef Documentation	111
21.22.2.1 AllkFN	111
21.22.2.2 AllkNN	112
21.22.2.3 AllkRAFN	112
21.22.2.4 AllkRANN	112
21.22.3 Function Documentation	112
21.22.3.1 Unmap	112
21.22.3.2 Unmap	113
21.23mlpack::nn Namespace Reference	113
21.24mlpack::optimization Namespace Reference	113
21.25mlpack::optimization::test Namespace Reference	114
21.26mlpack::pca Namespace Reference	114
21.27mlpack::perceptron Namespace Reference	115
21.28mlpack::radical Namespace Reference	115
21.28.1 Function Documentation	115
21.28.1.1 WhitenFeatureMajorMatrix	115
21.29mlpack::range Namespace Reference	115
21.29.1 Detailed Description	115
21.30mlpack::regression Namespace Reference	116
21.30.1 Detailed Description	116
21.31mlpack::sparse_coding Namespace Reference	116
21.32mlpack::svd Namespace Reference	116
21.33mlpack::tree Namespace Reference	116
21.33.1 Detailed Description	117
21.33.2 Typedef Documentation	117

21.33.2.1 CosineNodeQueue	117
21.34mlpack::util Namespace Reference	117
21.34.1 Function Documentation	118
21.34.1.1 GetVersion	118
21.34.1.2 Indent	118
21.34.2 Variable Documentation	118
21.34.2.1 cliDeleter	118
22 Class Documentation	119
22.1 IsVector< VecType > Struct Template Reference	119
22.1.1 Detailed Description	119
22.1.2 Member Data Documentation	119
22.1.2.1 value	119
22.2 IsVector< arma::Col< eT > > Struct Template Reference	120
22.2.1 Detailed Description	120
22.2.2 Member Data Documentation	120
22.2.2.1 value	120
22.3 IsVector< arma::Row< eT > > Struct Template Reference	120
22.3.1 Detailed Description	120
22.3.2 Member Data Documentation	120
22.3.2.1 value	120
22.4 IsVector< arma::SpCol< eT > > Struct Template Reference	120
22.4.1 Detailed Description	121
22.4.2 Member Data Documentation	121
22.4.2.1 value	121
22.5 IsVector< arma::SpRow< eT > > Struct Template Reference	121
22.5.1 Detailed Description	121
22.5.2 Member Data Documentation	121
22.5.2.1 value	121
22.6 IsVector< arma::SpSubview< eT > > Struct Template Reference	121
22.6.1 Detailed Description	122
22.6.2 Member Data Documentation	122
22.6.2.1 value	122
22.7 IsVector< arma::Subview_col< eT > > Struct Template Reference	122
22.7.1 Detailed Description	122
22.7.2 Member Data Documentation	122
22.7.2.1 value	122

22.8	IsVector< arma::subview_row< eT > > Struct Template Reference	122
22.8.1	Detailed Description	122
22.8.2	Member Data Documentation	123
22.8.2.1	value	123
22.9	mlpack::amf::AMF< TerminationPolicyType, InitializationRuleType, UpdateRuleType > Class Template Reference	123
22.9.1	Detailed Description	124
22.9.2	Constructor & Destructor Documentation	124
22.9.2.1	AMF	124
22.9.3	Member Function Documentation	125
22.9.3.1	Apply	125
22.9.3.2	InitializeRule	125
22.9.3.3	InitializeRule	125
22.9.3.4	TerminationPolicy	125
22.9.3.5	TerminationPolicy	126
22.9.3.6	Update	126
22.9.3.7	Update	126
22.9.4	Member Data Documentation	126
22.9.4.1	initializationRule	126
22.9.4.2	terminationPolicy	126
22.9.4.3	update	127
22.10	mlpack::amf::AverageInitialization Class Reference	127
22.10.1	Detailed Description	127
22.10.2	Constructor & Destructor Documentation	127
22.10.2.1	AverageInitialization	127
22.10.3	Member Function Documentation	127
22.10.3.1	Initialize	128
22.11	mlpack::amf::CompleteIncrementalTermination< TerminationPolicy > Class Template Reference	128
22.11.1	Detailed Description	128
22.11.2	Constructor & Destructor Documentation	128
22.11.2.1	CompleteIncrementalTermination	128
22.11.3	Member Function Documentation	129
22.11.3.1	Index	129
22.11.3.2	Initialize	129
22.11.3.3	Initialize	129
22.11.3.4	IsConverged	129
22.11.3.5	Iteration	129

22.11.3.6 MaxIterations	129
22.11.4 Member Data Documentation	130
22.11.4.1 incrementalIndex	130
22.11.4.2 iteration	130
22.11.4.3 t_policy	130
22.12mlpack::amf::IncompleteIncrementalTermination< TerminationPolicy > Class Template Reference	130
22.12.1 Detailed Description	131
22.12.2 Constructor & Destructor Documentation	131
22.12.2.1 IncompleteIncrementalTermination	131
22.12.3 Member Function Documentation	131
22.12.3.1 Index	131
22.12.3.2 Initialize	131
22.12.3.3 IsConverged	131
22.12.3.4 Iteration	132
22.12.3.5 MaxIterations	132
22.12.4 Member Data Documentation	132
22.12.4.1 incrementalIndex	132
22.12.4.2 iteration	132
22.12.4.3 t_policy	132
22.13mlpack::amf::NMFALSUpdate Class Reference	132
22.13.1 Detailed Description	133
22.13.2 Constructor & Destructor Documentation	133
22.13.2.1 NMFALSUpdate	133
22.13.3 Member Function Documentation	133
22.13.3.1 HUpdate	133
22.13.3.2 Initialize	134
22.13.3.3 WUpdate	134
22.14mlpack::amf::NMFMultiplicativeDistanceUpdate Class Reference	134
22.14.1 Detailed Description	135
22.14.2 Constructor & Destructor Documentation	135
22.14.2.1 NMFMultiplicativeDistanceUpdate	135
22.14.3 Member Function Documentation	135
22.14.3.1 HUpdate	135
22.14.3.2 Initialize	135
22.14.3.3 WUpdate	135
22.15mlpack::amf::NMFMultiplicativeDivergenceUpdate Class Reference	136
22.15.1 Detailed Description	136

22.15.2 Constructor & Destructor Documentation	136
22.15.2.1 NFMultiplicativeDivergenceUpdate	136
22.15.3 Member Function Documentation	137
22.15.3.1 HUpdate	137
22.15.3.2 Initialize	137
22.15.3.3 WUpdate	137
22.16mlpack::amf::RandomAcollInitialization< p > Class Template Reference	137
22.16.1 Detailed Description	138
22.16.2 Constructor & Destructor Documentation	138
22.16.2.1 RandomAcollInitialization	138
22.16.3 Member Function Documentation	138
22.16.3.1 Initialize	138
22.17mlpack::amf::RandomInitialization Class Reference	138
22.17.1 Detailed Description	139
22.17.2 Constructor & Destructor Documentation	139
22.17.2.1 RandomInitialization	139
22.17.3 Member Function Documentation	139
22.17.3.1 Initialize	139
22.18mlpack::amf::SimpleResidueTermination Class Reference	139
22.18.1 Detailed Description	140
22.18.2 Constructor & Destructor Documentation	140
22.18.2.1 SimpleResidueTermination	140
22.18.3 Member Function Documentation	140
22.18.3.1 Index	140
22.18.3.2 Initialize	141
22.18.3.3 IsConverged	141
22.18.3.4 Iteration	141
22.18.3.5 MaxIterations	141
22.18.3.6 MaxIterations	141
22.18.3.7 MinResidue	141
22.18.3.8 MinResidue	142
22.18.4 Member Data Documentation	142
22.18.4.1 iteration	142
22.18.4.2 maxIterations	142
22.18.4.3 minResidue	142
22.18.4.4 nm	142
22.18.4.5 normOld	142

22.18.4.6 residue	142
22.19mlpack::amf::SimpleToleranceTermination< MatType > Class Template Reference	143
22.19.1 Detailed Description	144
22.19.2 Constructor & Destructor Documentation	144
22.19.2.1 SimpleToleranceTermination	144
22.19.3 Member Function Documentation	144
22.19.3.1 Index	144
22.19.3.2 Initialize	144
22.19.3.3 IsConverged	145
22.19.3.4 Iteration	145
22.19.3.5 MaxIterations	145
22.19.3.6 MaxIterations	145
22.19.3.7 Tolerance	146
22.19.3.8 Tolerance	146
22.19.4 Member Data Documentation	146
22.19.4.1 c_index	146
22.19.4.2 c_indexOld	146
22.19.4.3 H	146
22.19.4.4 isCopy	146
22.19.4.5 iteration	147
22.19.4.6 maxIterations	147
22.19.4.7 normOld	147
22.19.4.8 residue	147
22.19.4.9 residueOld	147
22.19.4.10reverseStepCount	147
22.19.4.11reverseStepTolerance	148
22.19.4.12tolerance	148
22.19.4.13V	148
22.19.4.14W	148
22.20mlpack::amf::SVDBatchLearning Class Reference	148
22.20.1 Detailed Description	149
22.20.2 Constructor & Destructor Documentation	149
22.20.2.1 SVDBatchLearning	149
22.20.3 Member Function Documentation	149
22.20.3.1 HUpdate	149
22.20.3.2 Initialize	150
22.20.3.3 WUpdate	150

22.20.4 Member Data Documentation	150
22.20.4.1 kh	150
22.20.4.2 kw	150
22.20.4.3 max	150
22.20.4.4 mH	151
22.20.4.5 min	151
22.20.4.6 momentum	151
22.20.4.7 mW	151
22.20.4.8 u	151
22.21 mpack::amf::SVDCompleteIncrementalLearning< MatType > Class Template Reference	151
22.21.1 Detailed Description	152
22.21.2 Constructor & Destructor Documentation	152
22.21.2.1 SVDCompleteIncrementalLearning	152
22.21.3 Member Function Documentation	152
22.21.3.1 HUpdate	152
22.21.3.2 Initialize	152
22.21.3.3 WUpdate	153
22.21.4 Member Data Documentation	153
22.21.4.1 currentItemIndex	153
22.21.4.2 currentUserIndex	153
22.21.4.3 kh	153
22.21.4.4 kw	154
22.21.4.5 m	154
22.21.4.6 n	154
22.21.4.7 u	154
22.22 mpack::amf::SVDCompleteIncrementalLearning< arma::sp_mat > Class Template Reference	154
22.22.1 Detailed Description	155
22.22.2 Constructor & Destructor Documentation	155
22.22.2.1 SVDCompleteIncrementalLearning	155
22.22.2.2 ~SVDCompleteIncrementalLearning	155
22.22.3 Member Function Documentation	155
22.22.3.1 HUpdate	155
22.22.3.2 Initialize	156
22.22.3.3 WUpdate	156
22.22.4 Member Data Documentation	156
22.22.4.1 dummy	156
22.22.4.2 isStart	156

22.22.4.3 it	156
22.22.4.4 kh	156
22.22.4.5 kw	156
22.22.4.6 m	157
22.22.4.7 n	157
22.22.4.8 u	157
22.23mlpack::amf::SVDIncompleteIncrementalLearning Class Reference	157
22.23.1 Detailed Description	157
22.23.2 Constructor & Destructor Documentation	157
22.23.2.1 SVDIncompleteIncrementalLearning	157
22.23.3 Member Function Documentation	158
22.23.3.1 HUpdate	158
22.23.3.2 Initialize	158
22.23.3.3 WUpdate	158
22.23.4 Member Data Documentation	158
22.23.4.1 currentUserIndex	158
22.23.4.2 kh	159
22.23.4.3 kw	159
22.23.4.4 m	159
22.23.4.5 n	159
22.23.4.6 u	159
22.24mlpack::amf::ValidationRMSETermination< MatType > Class Template Reference	159
22.24.1 Detailed Description	160
22.24.2 Constructor & Destructor Documentation	160
22.24.2.1 ValidationRMSETermination	160
22.24.3 Member Function Documentation	160
22.24.3.1 Index	160
22.24.3.2 Initialize	160
22.24.3.3 IsConverged	161
22.24.3.4 Iteration	161
22.24.3.5 MaxIterations	161
22.24.4 Member Data Documentation	161
22.24.4.1 c_index	161
22.24.4.2 c_indexOld	161
22.24.4.3 H	161
22.24.4.4 isCopy	162
22.24.4.5 iteration	162

22.24.4.6 maxIterations	162
22.24.4.7 num_test_points	162
22.24.4.8 reverseStepCount	162
22.24.4.9 reverseStepTolerance	162
22.24.4.10 mse	162
22.24.4.11 rmseOld	163
22.24.4.12 test_points	163
22.24.4.13 tolerance	163
22.24.4.14 W	163
22.25 mlpack::bound::BallBound< VecType, TMetricType > Class Template Reference	163
22.25.1 Detailed Description	165
22.25.2 Member Typedef Documentation	165
22.25.2.1 MetricType	165
22.25.2.2 Vec	165
22.25.3 Constructor & Destructor Documentation	166
22.25.3.1 BallBound	166
22.25.3.2 BallBound	166
22.25.3.3 BallBound	166
22.25.3.4 BallBound	166
22.25.3.5 ~BallBound	166
22.25.4 Member Function Documentation	166
22.25.4.1 Center	166
22.25.4.2 Center	167
22.25.4.3 Centroid	167
22.25.4.4 Contains	167
22.25.4.5 Diameter	167
22.25.4.6 Dim	167
22.25.4.7 MaxDistance	167
22.25.4.8 MaxDistance	167
22.25.4.9 Metric	168
22.25.4.10 MinDistance	168
22.25.4.11 MinDistance	168
22.25.4.12 MinWidth	168
22.25.4.13 operator=	168
22.25.4.14 operator[]	168
22.25.4.15 operator" =	168
22.25.4.16 operator" =	169

22.25.4.17Radius	169
22.25.4.18Radius	169
22.25.4.19RangeDistance	169
22.25.4.20RangeDistance	169
22.25.4.21ToString	169
22.25.5 Member Data Documentation	169
22.25.5.1 center	170
22.25.5.2 metric	170
22.25.5.3 ownsMetric	170
22.25.5.4 radius	170
22.26mlpack::bound::HRectBound< Power, TakeRoot > Class Template Reference	170
22.26.1 Detailed Description	172
22.26.2 Member Typedef Documentation	173
22.26.2.1 MetricType	173
22.26.3 Constructor & Destructor Documentation	173
22.26.3.1 HRectBound	173
22.26.3.2 HRectBound	173
22.26.3.3 HRectBound	173
22.26.3.4 ~HRectBound	173
22.26.4 Member Function Documentation	173
22.26.4.1 Centroid	173
22.26.4.2 Clear	174
22.26.4.3 Contains	174
22.26.4.4 Diameter	174
22.26.4.5 Dim	174
22.26.4.6 MaxDistance	174
22.26.4.7 MaxDistance	174
22.26.4.8 Metric	174
22.26.4.9 MinDistance	175
22.26.4.10MinDistance	175
22.26.4.11MinWidth	175
22.26.4.12MinWidth	175
22.26.4.13operator=	175
22.26.4.14operator[]	175
22.26.4.15operator[]	176
22.26.4.16operator" =	176
22.26.4.17operator" =	176

22.26.4.18	RangeDistance	176
22.26.4.19	RangeDistance	176
22.26.4.20	ToString	177
22.26.5	Member Data Documentation	177
22.26.5.1	bounds	177
22.26.5.2	dim	177
22.26.5.3	minWidth	177
22.27	mlpack::cf::CF< FactorizerType > Class Template Reference	177
22.27.1	Detailed Description	179
22.27.2	Constructor & Destructor Documentation	179
22.27.2.1	CF	179
22.27.3	Member Function Documentation	179
22.27.3.1	CleanData	179
22.27.3.2	CleanedData	180
22.27.3.3	Data	180
22.27.3.4	Factorizer	180
22.27.3.5	GetRecommendations	180
22.27.3.6	GetRecommendations	180
22.27.3.7	H	181
22.27.3.8	InsertNeighbor	181
22.27.3.9	NumUsersForSimilarity	181
22.27.3.10	NumUsersForSimilarity	181
22.27.3.11	Rank	181
22.27.3.12	Rank	182
22.27.3.13	Rating	182
22.27.3.14	ToString	182
22.27.3.15	W	182
22.27.4	Member Data Documentation	182
22.27.4.1	cleanedData	182
22.27.4.2	data	182
22.27.4.3	factorizer	183
22.27.4.4	h	183
22.27.4.5	numUsersForSimilarity	183
22.27.4.6	rank	183
22.27.4.7	rating	183
22.27.4.8	w	183
22.28	mlpack::CLI Class Reference	184

22.28.1 Detailed Description	186
22.28.2 Adding parameters to a program	186
22.28.3 Documenting the program itself	187
22.28.4 Parsing the command line with CLI	187
22.28.5 Getting parameters with CLI	188
22.28.6 Member Typedef Documentation	188
22.28.6.1 amap_t	188
22.28.6.2 gmap_t	188
22.28.7 Constructor & Destructor Documentation	188
22.28.7.1 ~CLI	188
22.28.7.2 CLI	188
22.28.7.3 CLI	189
22.28.7.4 CLI	190
22.28.8 Member Function Documentation	190
22.28.8.1 Add	190
22.28.8.2 Add	190
22.28.8.3 AddAlias	190
22.28.8.4 AddFlag	191
22.28.8.5 AliasReverseLookup	192
22.28.8.6 DefaultMessages	192
22.28.8.7 Destroy	192
22.28.8.8 GetDescription	192
22.28.8.9 GetParam	192
22.28.8.10GetSingleton	193
22.28.8.11HasParam	193
22.28.8.12HyphenateString	193
22.28.8.13ParseCommandLine	193
22.28.8.14ParseStream	193
22.28.8.15Print	194
22.28.8.16PrintHelp	194
22.28.8.17RegisterProgramDoc	194
22.28.8.18RemoveDuplicateFlags	194
22.28.8.19RequiredOptions	194
22.28.8.20SanitizeString	194
22.28.8.21UpdateGmap	194
22.28.9 Friends And Related Function Documentation	195
22.28.9.1 Timer	195

22.28.10 Member Data Documentation	195
22.28.10.1 aliasValues	195
22.28.10.2 desc	195
22.28.10.3 didParse	195
22.28.10.4 doc	195
22.28.10.5 globalValues	195
22.28.10.6 programName	195
22.28.10.7 requiredOptions	195
22.28.10.8 singleton	196
22.28.10.9 timer	196
22.28.10.10 map	196
22.29 mlpack::decision_stump::DecisionStump< MatType > Class Template Reference	196
22.29.1 Detailed Description	197
22.29.2 Constructor & Destructor Documentation	198
22.29.2.1 DecisionStump	198
22.29.2.2 DecisionStump	198
22.29.3 Member Function Documentation	198
22.29.3.1 BinLabels	198
22.29.3.2 BinLabels	198
22.29.3.3 CalculateEntropy	199
22.29.3.4 Classify	199
22.29.3.5 CountMostFreq	199
22.29.3.6 IsDistinct	199
22.29.3.7 MergeRanges	199
22.29.3.8 SetupSplitAttribute	200
22.29.3.9 Split	200
22.29.3.10 Split	200
22.29.3.11 SplitAttribute	200
22.29.3.12 SplitAttribute	200
22.29.3.13 Train	200
22.29.3.14 TrainOnAtt	201
22.29.4 Member Data Documentation	201
22.29.4.1 binLabels	201
22.29.4.2 bucketSize	201
22.29.4.3 numClass	201
22.29.4.4 split	201
22.29.4.5 splitAttribute	202

22.30mlpack::det::DTree Class Reference	202
22.30.1 Detailed Description	205
22.30.2 Constructor & Destructor Documentation	205
22.30.2.1 DTree	205
22.30.2.2 DTree	205
22.30.2.3 DTree	205
22.30.2.4 DTree	205
22.30.2.5 DTree	207
22.30.2.6 ~DTree	207
22.30.3 Member Function Documentation	207
22.30.3.1 AlphaUpper	207
22.30.3.2 ComputeValue	207
22.30.3.3 ComputeVariableImportance	207
22.30.3.4 End	208
22.30.3.5 FindBucket	208
22.30.3.6 FindSplit	208
22.30.3.7 Grow	208
22.30.3.8 Left	208
22.30.3.9 LogNegativeError	208
22.30.3.10LogNegError	209
22.30.3.11LogVolume	209
22.30.3.12MaxVals	209
22.30.3.13MaxVals	209
22.30.3.14MinVals	209
22.30.3.15MinVals	209
22.30.3.16PruneAndUpdate	210
22.30.3.17Ratio	210
22.30.3.18Right	210
22.30.3.19Root	210
22.30.3.20SplitData	210
22.30.3.21SplitDim	210
22.30.3.22SplitValue	211
22.30.3.23Start	211
22.30.3.24SubtreeLeaves	211
22.30.3.25SubtreeLeavesLogNegError	211
22.30.3.26TagTree	211
22.30.3.27ToString	211

22.30.3.28	WithinRange	211
22.30.3.29	WriteTree	211
22.30.4	Member Data Documentation	212
22.30.4.1	alphaUpper	212
22.30.4.2	bucketTag	212
22.30.4.3	end	212
22.30.4.4	left	212
22.30.4.5	logNegError	212
22.30.4.6	logVolume	212
22.30.4.7	maxVals	213
22.30.4.8	minVals	213
22.30.4.9	ratio	213
22.30.4.10	right	213
22.30.4.11	root	213
22.30.4.12	splitDim	213
22.30.4.13	splitValue	213
22.30.4.14	start	214
22.30.4.15	subtreeLeaves	214
22.30.4.16	subtreeLeavesLogNegError	214
22.31	mlpack::distribution::DiscreteDistribution Class Reference	214
22.31.1	Detailed Description	215
22.31.2	Constructor & Destructor Documentation	215
22.31.2.1	DiscreteDistribution	215
22.31.2.2	DiscreteDistribution	215
22.31.2.3	DiscreteDistribution	216
22.31.3	Member Function Documentation	217
22.31.3.1	Dimensionality	217
22.31.3.2	Estimate	217
22.31.3.3	Estimate	217
22.31.3.4	Probabilities	217
22.31.3.5	Probabilities	217
22.31.3.6	Probability	217
22.31.3.7	Random	218
22.31.3.8	ToString	218
22.31.4	Member Data Documentation	218
22.31.4.1	probabilities	218
22.32	mlpack::distribution::GaussianDistribution Class Reference	218

22.32.1 Detailed Description	219
22.32.2 Constructor & Destructor Documentation	219
22.32.2.1 GaussianDistribution	219
22.32.2.2 GaussianDistribution	219
22.32.2.3 GaussianDistribution	220
22.32.3 Member Function Documentation	220
22.32.3.1 Covariance	220
22.32.3.2 Covariance	220
22.32.3.3 Dimensionality	220
22.32.3.4 Estimate	220
22.32.3.5 Estimate	220
22.32.3.6 Mean	220
22.32.3.7 Mean	221
22.32.3.8 Probability	221
22.32.3.9 Random	221
22.32.3.10 ToString	221
22.32.4 Member Data Documentation	221
22.32.4.1 covariance	221
22.32.4.2 mean	221
22.33 mpack::distribution::LaplaceDistribution Class Reference	221
22.33.1 Detailed Description	222
22.33.2 Constructor & Destructor Documentation	223
22.33.2.1 LaplaceDistribution	223
22.33.2.2 LaplaceDistribution	223
22.33.2.3 LaplaceDistribution	223
22.33.3 Member Function Documentation	223
22.33.3.1 Dimensionality	223
22.33.3.2 Estimate	224
22.33.3.3 Estimate	224
22.33.3.4 Mean	224
22.33.3.5 Mean	224
22.33.3.6 Probability	224
22.33.3.7 Random	224
22.33.3.8 Scale	224
22.33.3.9 Scale	225
22.33.3.10 ToString	225
22.33.4 Member Data Documentation	225

22.33.4.1 mean	225
22.33.4.2 scale	225
22.34mlpack::emst::DTBRules< MetricType, TreeType > Class Template Reference	225
22.34.1 Detailed Description	227
22.34.2 Member Typedef Documentation	227
22.34.2.1 TraversalInfoType	227
22.34.3 Constructor & Destructor Documentation	227
22.34.3.1 DTBRules	227
22.34.4 Member Function Documentation	227
22.34.4.1 BaseCase	227
22.34.4.2 BaseCases	227
22.34.4.3 BaseCases	227
22.34.4.4 CalculateBound	228
22.34.4.5 Rescore	228
22.34.4.6 Rescore	228
22.34.4.7 Score	228
22.34.4.8 Score	229
22.34.4.9 Score	229
22.34.4.10Score	229
22.34.4.11Scores	229
22.34.4.12Scores	230
22.34.4.13TraversalInfo	230
22.34.4.14TraversalInfo	230
22.34.5 Member Data Documentation	230
22.34.5.1 baseCases	230
22.34.5.2 connections	230
22.34.5.3 dataSet	230
22.34.5.4 metric	230
22.34.5.5 neighborsDistances	231
22.34.5.6 neighborsInComponent	231
22.34.5.7 neighborsOutComponent	231
22.34.5.8 scores	231
22.34.5.9 traversalInfo	231
22.35mlpack::emst::DTBStat Class Reference	231
22.35.1 Detailed Description	232
22.35.2 Constructor & Destructor Documentation	232
22.35.2.1 DTBStat	232

22.35.2.2 DTBStat	233
22.35.3 Member Function Documentation	234
22.35.3.1 Bound	234
22.35.3.2 Bound	234
22.35.3.3 ComponentMembership	234
22.35.3.4 ComponentMembership	234
22.35.3.5 MaxNeighborDistance	234
22.35.3.6 MaxNeighborDistance	234
22.35.3.7 MinNeighborDistance	235
22.35.3.8 MinNeighborDistance	235
22.35.4 Member Data Documentation	235
22.35.4.1 bound	235
22.35.4.2 componentMembership	235
22.35.4.3 maxNeighborDistance	235
22.35.4.4 minNeighborDistance	235
22.36mlpack::emst::DualTreeBoruvka< MetricType, TreeType > Class Template Reference	236
22.36.1 Detailed Description	237
22.36.2 Constructor & Destructor Documentation	238
22.36.2.1 DualTreeBoruvka	238
22.36.2.2 DualTreeBoruvka	238
22.36.2.3 ~DualTreeBoruvka	239
22.36.3 Member Function Documentation	239
22.36.3.1 AddAllEdges	239
22.36.3.2 AddEdge	239
22.36.3.3 Cleanup	239
22.36.3.4 CleanupHelper	239
22.36.3.5 ComputeMST	239
22.36.3.6 EmitResults	240
22.36.3.7 ToString	240
22.36.4 Member Data Documentation	240
22.36.4.1 connections	240
22.36.4.2 data	240
22.36.4.3 dataCopy	240
22.36.4.4 edges	240
22.36.4.5 metric	241
22.36.4.6 naive	241
22.36.4.7 neighborsDistances	241

22.36.4.8 neighborsInComponent	241
22.36.4.9 neighborsOutComponent	241
22.36.4.10 oldFromNew	241
22.36.4.11 lownTree	241
22.36.4.12 SortFun	242
22.36.4.13 totalDist	242
22.36.4.14 tree	242
22.37 mlpack::emst::DualTreeBoruvka< MetricType, TreeType >::SortEdgesHelper Struct Reference	242
22.37.1 Detailed Description	242
22.37.2 Member Function Documentation	242
22.37.2.1 operator()	243
22.38 mlpack::emst::EdgePair Class Reference	243
22.38.1 Detailed Description	243
22.38.2 Constructor & Destructor Documentation	244
22.38.2.1 EdgePair	244
22.38.3 Member Function Documentation	244
22.38.3.1 Distance	244
22.38.3.2 Distance	244
22.38.3.3 Greater	244
22.38.3.4 Greater	244
22.38.3.5 Lesser	244
22.38.3.6 Lesser	245
22.38.4 Member Data Documentation	245
22.38.4.1 distance	245
22.38.4.2 greater	245
22.38.4.3 lesser	245
22.39 mlpack::emst::UnionFind Class Reference	245
22.39.1 Detailed Description	246
22.39.2 Constructor & Destructor Documentation	246
22.39.2.1 UnionFind	246
22.39.2.2 ~UnionFind	246
22.39.3 Member Function Documentation	246
22.39.3.1 Find	246
22.39.3.2 Union	246
22.39.4 Member Data Documentation	247
22.39.4.1 parent	247
22.39.4.2 rank	247

22.40mlpack::fastmks::FastMKS< KernelType, TreeType > Class Template Reference	247
22.40.1 Detailed Description	248
22.40.2 Constructor & Destructor Documentation	249
22.40.2.1 FastMKS	249
22.40.2.2 FastMKS	249
22.40.2.3 FastMKS	249
22.40.2.4 FastMKS	250
22.40.2.5 FastMKS	250
22.40.2.6 FastMKS	250
22.40.2.7 ~FastMKS	251
22.40.3 Member Function Documentation	251
22.40.3.1 InsertNeighbor	251
22.40.3.2 Metric	251
22.40.3.3 Metric	251
22.40.3.4 Search	251
22.40.3.5 ToString	253
22.40.4 Member Data Documentation	253
22.40.4.1 metric	253
22.40.4.2 naive	253
22.40.4.3 querySet	253
22.40.4.4 queryTree	253
22.40.4.5 referenceSet	254
22.40.4.6 referenceTree	254
22.40.4.7 single	254
22.40.4.8 treeOwner	254
22.41mlpack::fastmks::FastMKSRules< KernelType, TreeType > Class Template Reference	254
22.41.1 Detailed Description	256
22.41.2 Member Typedef Documentation	256
22.41.2.1 TraversalInfoType	256
22.41.3 Constructor & Destructor Documentation	256
22.41.3.1 FastMKSRules	256
22.41.4 Member Function Documentation	256
22.41.4.1 BaseCase	256
22.41.4.2 BaseCases	256
22.41.4.3 BaseCases	256
22.41.4.4 CalculateBound	257
22.41.4.5 InsertNeighbor	257

22.41.4.6 Rescore	257
22.41.4.7 Rescore	257
22.41.4.8 Score	257
22.41.4.9 Score	258
22.41.4.10 Scores	258
22.41.4.11 Scores	258
22.41.4.12 TraversalInfo	258
22.41.4.13 TraversalInfo	258
22.41.5 Member Data Documentation	258
22.41.5.1 baseCases	259
22.41.5.2 indices	259
22.41.5.3 kernel	259
22.41.5.4 lastKernel	259
22.41.5.5 lastQueryIndex	259
22.41.5.6 lastReferenceIndex	259
22.41.5.7 products	259
22.41.5.8 queryKernels	260
22.41.5.9 querySet	260
22.41.5.10 referenceKernels	260
22.41.5.11 referenceSet	260
22.41.5.12 scores	260
22.41.5.13 traversalInfo	260
22.42 mlpack::fastmks::FastMKStat Class Reference	260
22.42.1 Detailed Description	261
22.42.2 Constructor & Destructor Documentation	261
22.42.2.1 FastMKStat	261
22.42.2.2 FastMKStat	262
22.42.3 Member Function Documentation	262
22.42.3.1 Bound	262
22.42.3.2 Bound	262
22.42.3.3 LastKernel	262
22.42.3.4 LastKernel	262
22.42.3.5 LastKernelNode	262
22.42.3.6 LastKernelNode	263
22.42.3.7 SelfKernel	263
22.42.3.8 SelfKernel	263
22.42.4 Member Data Documentation	263

22.42.4.1 bound	263
22.42.4.2 lastKernel	263
22.42.4.3 lastKernelNode	263
22.42.4.4 selfKernel	263
22.43mlpack::gmm::DiagonalConstraint Class Reference	264
22.43.1 Detailed Description	264
22.43.2 Member Function Documentation	264
22.43.2.1 ApplyConstraint	264
22.44mlpack::gmm::EigenvalueRatioConstraint Class Reference	264
22.44.1 Detailed Description	265
22.44.2 Constructor & Destructor Documentation	265
22.44.2.1 EigenvalueRatioConstraint	265
22.44.3 Member Function Documentation	265
22.44.3.1 ApplyConstraint	265
22.44.4 Member Data Documentation	265
22.44.4.1 ratios	265
22.45mlpack::gmm::EMFit< InitialClusteringType, CovarianceConstraintPolicy > Class Template Reference	266
22.45.1 Detailed Description	267
22.45.2 Constructor & Destructor Documentation	267
22.45.2.1 EMFit	267
22.45.3 Member Function Documentation	268
22.45.3.1 Clusterer	268
22.45.3.2 Clusterer	268
22.45.3.3 Constraint	268
22.45.3.4 Constraint	268
22.45.3.5 Estimate	268
22.45.3.6 Estimate	269
22.45.3.7 InitialClustering	269
22.45.3.8 LogLikelihood	269
22.45.3.9 MaxIterations	270
22.45.3.10MaxIterations	270
22.45.3.11Tolerance	270
22.45.3.12Tolerance	270
22.45.4 Member Data Documentation	270
22.45.4.1 clusterer	271
22.45.4.2 constraint	271
22.45.4.3 maxIterations	271

22.45.4.4 tolerance	271
22.46mlpack::gmm::GMM< FittingType > Class Template Reference	271
22.46.1 Detailed Description	274
22.46.2 Constructor & Destructor Documentation	274
22.46.2.1 GMM	274
22.46.2.2 GMM	275
22.46.2.3 GMM	275
22.46.2.4 GMM	275
22.46.2.5 GMM	275
22.46.2.6 GMM	276
22.46.2.7 GMM	276
22.46.3 Member Function Documentation	276
22.46.3.1 Classify	276
22.46.3.2 Covariances	276
22.46.3.3 Covariances	276
22.46.3.4 Dimensionality	277
22.46.3.5 Dimensionality	277
22.46.3.6 Estimate	277
22.46.3.7 Estimate	277
22.46.3.8 Fitter	279
22.46.3.9 Fitter	279
22.46.3.10Gaussians	279
22.46.3.11Gaussians	279
22.46.3.12Load	279
22.46.3.13LogLikelihood	280
22.46.3.14Means	280
22.46.3.15Means	280
22.46.3.16operator=	280
22.46.3.17operator=	280
22.46.3.18Probability	280
22.46.3.19Probability	281
22.46.3.20Random	281
22.46.3.21Save	281
22.46.3.22ToString	281
22.46.3.23Weights	281
22.46.3.24Weights	281
22.46.4 Member Data Documentation	282

22.46.4.1 covariances	282
22.46.4.2 dimensionality	282
22.46.4.3 fitter	282
22.46.4.4 gaussians	282
22.46.4.5 localFitter	282
22.46.4.6 means	282
22.46.4.7 weights	283
22.47mlpack::gmm::NoConstraint Class Reference	283
22.47.1 Detailed Description	283
22.47.2 Member Function Documentation	283
22.47.2.1 ApplyConstraint	283
22.48mlpack::gmm::PositiveDefiniteConstraint Class Reference	283
22.48.1 Detailed Description	284
22.48.2 Member Function Documentation	284
22.48.2.1 ApplyConstraint	284
22.49mlpack::hmm::HMM< Distribution > Class Template Reference	284
22.49.1 Detailed Description	286
22.49.2 Constructor & Destructor Documentation	286
22.49.2.1 HMM	286
22.49.2.2 HMM	287
22.49.3 Member Function Documentation	287
22.49.3.1 Backward	287
22.49.3.2 Dimensionality	288
22.49.3.3 Dimensionality	288
22.49.3.4 Emission	288
22.49.3.5 Emission	288
22.49.3.6 Estimate	288
22.49.3.7 Estimate	289
22.49.3.8 Forward	289
22.49.3.9 Generate	289
22.49.3.10Initial	290
22.49.3.11Initial	290
22.49.3.12LogLikelihood	290
22.49.3.13Predict	290
22.49.3.14Tolerance	291
22.49.3.15Tolerance	291
22.49.3.16ToString	291

22.49.3.17Train	291
22.49.3.18Train	291
22.49.3.19Transition	292
22.49.3.20Transition	292
22.49.4 Member Data Documentation	292
22.49.4.1 dimensionality	292
22.49.4.2 emission	292
22.49.4.3 initial	293
22.49.4.4 tolerance	293
22.49.4.5 transition	293
22.50mlpack::kernel::CosineDistance Class Reference	293
22.50.1 Detailed Description	293
22.50.2 Member Function Documentation	294
22.50.2.1 Evaluate	294
22.50.2.2 ToString	294
22.51mlpack::kernel::EpanechnikovKernel Class Reference	294
22.51.1 Detailed Description	295
22.51.2 Constructor & Destructor Documentation	295
22.51.2.1 EpanechnikovKernel	295
22.51.3 Member Function Documentation	295
22.51.3.1 ConvolutionIntegral	295
22.51.3.2 Evaluate	296
22.51.3.3 Evaluate	296
22.51.3.4 Normalizer	296
22.51.3.5 ToString	296
22.51.4 Member Data Documentation	296
22.51.4.1 bandwidth	296
22.51.4.2 inverseBandwidthSquared	296
22.52mlpack::kernel::ExampleKernel Class Reference	296
22.52.1 Detailed Description	297
22.52.2 Constructor & Destructor Documentation	297
22.52.2.1 ExampleKernel	297
22.52.3 Member Function Documentation	298
22.52.3.1 ConvolutionIntegral	298
22.52.3.2 Evaluate	298
22.52.3.3 Normalizer	299
22.52.3.4 ToString	299

22.53mlpack::kernel::GaussianKernel Class Reference	299
22.53.1 Detailed Description	300
22.53.2 Constructor & Destructor Documentation	300
22.53.2.1 GaussianKernel	300
22.53.2.2 GaussianKernel	300
22.53.3 Member Function Documentation	300
22.53.3.1 Bandwidth	300
22.53.3.2 Bandwidth	301
22.53.3.3 ConvolutionIntegral	301
22.53.3.4 Evaluate	301
22.53.3.5 Evaluate	302
22.53.3.6 Gamma	303
22.53.3.7 Normalizer	303
22.53.3.8 ToString	303
22.53.4 Member Data Documentation	303
22.53.4.1 bandwidth	303
22.53.4.2 gamma	304
22.54mlpack::kernel::HyperbolicTangentKernel Class Reference	304
22.54.1 Detailed Description	304
22.54.2 Constructor & Destructor Documentation	305
22.54.2.1 HyperbolicTangentKernel	305
22.54.2.2 HyperbolicTangentKernel	305
22.54.3 Member Function Documentation	305
22.54.3.1 Evaluate	305
22.54.3.2 Offset	305
22.54.3.3 Offset	305
22.54.3.4 Scale	306
22.54.3.5 Scale	306
22.54.3.6 ToString	306
22.54.4 Member Data Documentation	306
22.54.4.1 offset	306
22.54.4.2 scale	306
22.55mlpack::kernel::KernelTraits< KernelType > Class Template Reference	306
22.55.1 Detailed Description	307
22.55.2 Member Data Documentation	307
22.55.2.1 IsNormalized	307
22.56mlpack::kernel::KernelTraits< CosineDistance > Class Template Reference	307

22.56.1 Detailed Description	307
22.56.2 Member Data Documentation	307
22.56.2.1 IsNormalized	307
22.57mlpack::kernel::KernelTraits< EpanechnikovKernel > Class Template Reference	308
22.57.1 Detailed Description	308
22.57.2 Member Data Documentation	308
22.57.2.1 IsNormalized	308
22.58mlpack::kernel::KernelTraits< GaussianKernel > Class Template Reference	308
22.58.1 Detailed Description	309
22.58.2 Member Data Documentation	309
22.58.2.1 IsNormalized	309
22.59mlpack::kernel::KernelTraits< LaplacianKernel > Class Template Reference	309
22.59.1 Detailed Description	309
22.59.2 Member Data Documentation	309
22.59.2.1 IsNormalized	309
22.60mlpack::kernel::KernelTraits< SphericalKernel > Class Template Reference	310
22.60.1 Detailed Description	310
22.60.2 Member Data Documentation	310
22.60.2.1 IsNormalized	310
22.61mlpack::kernel::KernelTraits< TriangularKernel > Class Template Reference	310
22.61.1 Detailed Description	310
22.61.2 Member Data Documentation	311
22.61.2.1 IsNormalized	311
22.62mlpack::kernel::KMeansSelection< ClusteringType > Class Template Reference	311
22.62.1 Detailed Description	311
22.62.2 Member Function Documentation	311
22.62.2.1 Select	311
22.63mlpack::kernel::LaplacianKernel Class Reference	312
22.63.1 Detailed Description	312
22.63.2 Constructor & Destructor Documentation	312
22.63.2.1 LaplacianKernel	312
22.63.2.2 LaplacianKernel	313
22.63.3 Member Function Documentation	314
22.63.3.1 Bandwidth	314
22.63.3.2 Bandwidth	314
22.63.3.3 Evaluate	314
22.63.3.4 Evaluate	314

22.63.3.5 ToString	315
22.63.4 Member Data Documentation	315
22.63.4.1 bandwidth	315
22.64mlpack::kernel::LinearKernel Class Reference	315
22.64.1 Detailed Description	316
22.64.2 Constructor & Destructor Documentation	316
22.64.2.1 LinearKernel	316
22.64.3 Member Function Documentation	316
22.64.3.1 Evaluate	316
22.64.3.2 ToString	316
22.65mlpack::kernel::NystroemMethod< KernelType, PointSelectionPolicy > Class Template Reference	317
22.65.1 Detailed Description	317
22.65.2 Constructor & Destructor Documentation	317
22.65.2.1 NystroemMethod	317
22.65.3 Member Function Documentation	318
22.65.3.1 Apply	318
22.65.3.2 GetKernelMatrix	318
22.65.3.3 GetKernelMatrix	318
22.65.4 Member Data Documentation	318
22.65.4.1 data	318
22.65.4.2 kernel	318
22.65.4.3 rank	319
22.66mlpack::kernel::OrderedSelection Class Reference	319
22.66.1 Detailed Description	319
22.66.2 Member Function Documentation	319
22.66.2.1 Select	319
22.67mlpack::kernel::PolynomialKernel Class Reference	319
22.67.1 Detailed Description	320
22.67.2 Constructor & Destructor Documentation	320
22.67.2.1 PolynomialKernel	320
22.67.3 Member Function Documentation	321
22.67.3.1 Degree	321
22.67.3.2 Degree	321
22.67.3.3 Evaluate	321
22.67.3.4 Offset	321
22.67.3.5 Offset	322
22.67.3.6 ToString	322

22.67.4 Member Data Documentation	322
22.67.4.1 degree	322
22.67.4.2 offset	322
22.68mlpack::kernel::PSpectrumStringKernel Class Reference	322
22.68.1 Detailed Description	323
22.68.2 Constructor & Destructor Documentation	323
22.68.2.1 PSpectrumStringKernel	323
22.68.3 Member Function Documentation	324
22.68.3.1 Counts	324
22.68.3.2 Counts	324
22.68.3.3 Evaluate	324
22.68.3.4 P	324
22.68.3.5 P	324
22.68.3.6 ToString	325
22.68.4 Member Data Documentation	325
22.68.4.1 counts	325
22.68.4.2 datasets	325
22.68.4.3 p	325
22.69mlpack::kernel::RandomSelection Class Reference	325
22.69.1 Detailed Description	325
22.69.2 Member Function Documentation	326
22.69.2.1 Select	326
22.70mlpack::kernel::SphericalKernel Class Reference	327
22.70.1 Detailed Description	327
22.70.2 Constructor & Destructor Documentation	327
22.70.2.1 SphericalKernel	327
22.70.2.2 SphericalKernel	328
22.70.3 Member Function Documentation	328
22.70.3.1 ConvolutionIntegral	328
22.70.3.2 Evaluate	328
22.70.3.3 Evaluate	328
22.70.3.4 Normalizer	328
22.70.3.5 ToString	329
22.70.4 Member Data Documentation	329
22.70.4.1 bandwidth	329
22.70.4.2 bandwidthSquared	329
22.71mlpack::kernel::TriangularKernel Class Reference	329

22.71.1 Detailed Description	330
22.71.2 Constructor & Destructor Documentation	330
22.71.2.1 TriangularKernel	330
22.71.3 Member Function Documentation	330
22.71.3.1 Bandwidth	330
22.71.3.2 Bandwidth	330
22.71.3.3 Evaluate	330
22.71.3.4 Evaluate	331
22.71.3.5 ToString	332
22.71.4 Member Data Documentation	332
22.71.4.1 bandwidth	332
22.72mlpack::kmeans::AllowEmptyClusters Class Reference	332
22.72.1 Detailed Description	332
22.72.2 Constructor & Destructor Documentation	333
22.72.2.1 AllowEmptyClusters	333
22.72.3 Member Function Documentation	333
22.72.3.1 EmptyCluster	333
22.73mlpack::kmeans::KMeans< MetricType, InitialPartitionPolicy, EmptyClusterPolicy > Class Template Reference	333
22.73.1 Detailed Description	334
22.73.2 Constructor & Destructor Documentation	335
22.73.2.1 KMeans	335
22.73.3 Member Function Documentation	336
22.73.3.1 Cluster	336
22.73.3.2 Cluster	336
22.73.3.3 EmptyClusterAction	337
22.73.3.4 EmptyClusterAction	337
22.73.3.5 MaxIterations	337
22.73.3.6 MaxIterations	337
22.73.3.7 Metric	338
22.73.3.8 Metric	338
22.73.3.9 OverclusteringFactor	338
22.73.3.10OverclusteringFactor	338
22.73.3.11Partitioner	338
22.73.3.12Partitioner	338
22.73.3.13ToString	339
22.73.4 Member Data Documentation	339

22.73.4.1 emptyClusterAction	339
22.73.4.2 maxIterations	339
22.73.4.3 metric	339
22.73.4.4 overclusteringFactor	339
22.73.4.5 partitioner	340
22.74mlpack::kmeans::MaxVarianceNewCluster Class Reference	340
22.74.1 Detailed Description	340
22.74.2 Constructor & Destructor Documentation	340
22.74.2.1 MaxVarianceNewCluster	340
22.74.3 Member Function Documentation	340
22.74.3.1 EmptyCluster	341
22.75mlpack::kmeans::RandomPartition Class Reference	342
22.75.1 Detailed Description	342
22.75.2 Constructor & Destructor Documentation	342
22.75.2.1 RandomPartition	342
22.75.3 Member Function Documentation	343
22.75.3.1 Cluster	343
22.76mlpack::kmeans::RefinedStart Class Reference	343
22.76.1 Detailed Description	344
22.76.2 Constructor & Destructor Documentation	344
22.76.2.1 RefinedStart	344
22.76.3 Member Function Documentation	344
22.76.3.1 Cluster	344
22.76.3.2 Percentage	344
22.76.3.3 Percentage	345
22.76.3.4 Samplings	345
22.76.3.5 Samplings	345
22.76.4 Member Data Documentation	345
22.76.4.1 percentage	345
22.76.4.2 samplings	345
22.77mlpack::kpca::KernelPCA< KernelType, KernelRule > Class Template Reference	345
22.77.1 Detailed Description	346
22.77.2 Constructor & Destructor Documentation	346
22.77.2.1 KernelPCA	346
22.77.3 Member Function Documentation	348
22.77.3.1 Apply	348
22.77.3.2 Apply	348

22.77.3.3 Apply	348
22.77.3.4 Apply	348
22.77.3.5 CenterTransformedData	349
22.77.3.6 CenterTransformedData	349
22.77.3.7 Kernel	349
22.77.3.8 Kernel	349
22.77.3.9 ToString	349
22.77.4 Member Data Documentation	349
22.77.4.1 centerTransformedData	350
22.77.4.2 kernel	350
22.78mlpack::kpca::NaiveKernelRule< KernelType > Class Template Reference	350
22.78.1 Detailed Description	350
22.78.2 Member Function Documentation	350
22.78.2.1 ApplyKernelMatrix	350
22.79mlpack::kpca::NystroemKernelRule< KernelType, PointSelectionPolicy > Class Template Reference	351
22.79.1 Detailed Description	351
22.79.2 Member Function Documentation	351
22.79.2.1 ApplyKernelMatrix	351
22.80mlpack::lcc::LocalCoordinateCoding< DictionaryInitializer > Class Template Reference	352
22.80.1 Detailed Description	353
22.80.2 Constructor & Destructor Documentation	353
22.80.2.1 LocalCoordinateCoding	353
22.80.3 Member Function Documentation	354
22.80.3.1 Codes	354
22.80.3.2 Codes	354
22.80.3.3 Data	354
22.80.3.4 Dictionary	354
22.80.3.5 Dictionary	354
22.80.3.6 Encode	355
22.80.3.7 Objective	356
22.80.3.8 OptimizeCode	356
22.80.3.9 OptimizeDictionary	356
22.80.3.10ToString	356
22.80.4 Member Data Documentation	356
22.80.4.1 atoms	356
22.80.4.2 codes	356
22.80.4.3 data	356

22.80.4.4 dictionary	357
22.80.4.5 lambda	357
22.81 mpack::Log Class Reference	357
22.81.1 Detailed Description	358
22.81.2 Member Function Documentation	358
22.81.2.1 Assert	358
22.81.3 Member Data Documentation	359
22.81.3.1 cout	359
22.81.3.2 Debug	359
22.81.3.3 Fatal	359
22.81.3.4 Info	359
22.81.3.5 Warn	359
22.82 mpack::math::Range Class Reference	359
22.82.1 Detailed Description	361
22.82.2 Constructor & Destructor Documentation	361
22.82.2.1 Range	361
22.82.2.2 Range	361
22.82.2.3 Range	361
22.82.3 Member Function Documentation	361
22.82.3.1 Contains	361
22.82.3.2 Contains	361
22.82.3.3 Hi	362
22.82.3.4 Hi	362
22.82.3.5 Lo	362
22.82.3.6 Lo	362
22.82.3.7 Mid	362
22.82.3.8 operator"!="	362
22.82.3.9 operator&	362
22.82.3.10 operator&=	363
22.82.3.11 operator*	363
22.82.3.12 operator*=	363
22.82.3.13 operator<	363
22.82.3.14 operator==	363
22.82.3.15 operator>	363
22.82.3.16 operator" 	364
22.82.3.17 operator" =	364
22.82.3.18 ToString	364

22.82.3.19Width	364
22.82.4 Friends And Related Function Documentation	364
22.82.4.1 operator*	364
22.82.5 Member Data Documentation	364
22.82.5.1 hi	364
22.82.5.2 lo	365
22.83mlpack::metric::IPMetric< KernelType > Class Template Reference	365
22.83.1 Detailed Description	365
22.83.2 Constructor & Destructor Documentation	365
22.83.2.1 IPMetric	365
22.83.2.2 IPMetric	366
22.83.2.3 ~IPMetric	366
22.83.3 Member Function Documentation	366
22.83.3.1 Evaluate	366
22.83.3.2 Kernel	366
22.83.3.3 Kernel	366
22.83.3.4 ToString	366
22.83.4 Member Data Documentation	366
22.83.4.1 kernel	366
22.83.4.2 localKernel	366
22.84mlpack::metric::LMetric< Power, TakeRoot > Class Template Reference	367
22.84.1 Detailed Description	367
22.84.2 Constructor & Destructor Documentation	368
22.84.2.1 LMetric	368
22.84.3 Member Function Documentation	368
22.84.3.1 Evaluate	368
22.84.3.2 ToString	368
22.85mlpack::metric::MahalanobisDistance< TakeRoot > Class Template Reference	368
22.85.1 Detailed Description	369
22.85.2 Constructor & Destructor Documentation	369
22.85.2.1 MahalanobisDistance	369
22.85.2.2 MahalanobisDistance	370
22.85.2.3 MahalanobisDistance	371
22.85.3 Member Function Documentation	371
22.85.3.1 Covariance	371
22.85.3.2 Covariance	371
22.85.3.3 Evaluate	371

22.85.3.4 ToString	371
22.85.4 Member Data Documentation	372
22.85.4.1 covariance	372
22.86mlpack::mvu::MVU Class Reference	372
22.86.1 Detailed Description	372
22.86.2 Constructor & Destructor Documentation	372
22.86.2.1 MVU	372
22.86.3 Member Function Documentation	373
22.86.3.1 Unfold	373
22.86.4 Member Data Documentation	373
22.86.4.1 data	373
22.87mlpack::naive_bayes::NaiveBayesClassifier< MatType > Class Template Reference	373
22.87.1 Detailed Description	374
22.87.2 Constructor & Destructor Documentation	374
22.87.2.1 NaiveBayesClassifier	374
22.87.3 Member Function Documentation	374
22.87.3.1 Classify	374
22.87.3.2 Means	375
22.87.3.3 Means	375
22.87.3.4 Probabilities	375
22.87.3.5 Probabilities	375
22.87.3.6 Variances	375
22.87.3.7 Variances	376
22.87.4 Member Data Documentation	376
22.87.4.1 means	376
22.87.4.2 probabilities	376
22.87.4.3 variances	376
22.88mlpack::nca::NCA< MetricType, OptimizerType > Class Template Reference	376
22.88.1 Detailed Description	377
22.88.2 Constructor & Destructor Documentation	377
22.88.2.1 NCA	378
22.88.3 Member Function Documentation	378
22.88.3.1 Dataset	378
22.88.3.2 Labels	378
22.88.3.3 LearnDistance	378
22.88.3.4 Optimizer	379
22.88.3.5 Optimizer	379

22.88.3.6 ToString	379
22.88.4 Member Data Documentation	379
22.88.4.1 dataset	379
22.88.4.2 errorFunction	379
22.88.4.3 labels	379
22.88.4.4 metric	380
22.88.4.5 optimizer	380
22.89mlpack::nca::SoftmaxErrorFunction< MetricType > Class Template Reference	380
22.89.1 Detailed Description	381
22.89.2 Constructor & Destructor Documentation	381
22.89.2.1 SoftmaxErrorFunction	381
22.89.3 Member Function Documentation	382
22.89.3.1 Evaluate	382
22.89.3.2 Evaluate	382
22.89.3.3 GetInitialPoint	382
22.89.3.4 Gradient	382
22.89.3.5 Gradient	383
22.89.3.6 NumFunctions	383
22.89.3.7 Precalculate	383
22.89.3.8 ToString	383
22.89.4 Member Data Documentation	383
22.89.4.1 dataset	383
22.89.4.2 denominators	384
22.89.4.3 labels	384
22.89.4.4 lastCoordinates	384
22.89.4.5 metric	384
22.89.4.6 p	384
22.89.4.7 precalculated	384
22.89.4.8 stretchedDataset	384
22.90mlpack::neighbor::FurthestNeighborSort Class Reference	385
22.90.1 Detailed Description	385
22.90.2 Member Function Documentation	386
22.90.2.1 BestDistance	386
22.90.2.2 BestNodeToNodeDistance	386
22.90.2.3 BestNodeToNodeDistance	386
22.90.2.4 BestNodeToNodeDistance	386
22.90.2.5 BestPointToNodeDistance	386

22.90.2.6 BestPointToNodeDistance	387
22.90.2.7 CombineBest	387
22.90.2.8 CombineWorst	387
22.90.2.9 IsBetter	387
22.90.2.10 SortDistance	387
22.90.2.11 WorstDistance	388
22.91 mlpack::neighbor::LSHSearch< SortPolicy > Class Template Reference	388
22.91.1 Detailed Description	389
22.91.2 Constructor & Destructor Documentation	390
22.91.2.1 LSHSearch	390
22.91.2.2 LSHSearch	390
22.91.3 Member Function Documentation	390
22.91.3.1 BaseCase	391
22.91.3.2 BuildHash	392
22.91.3.3 InsertNeighbor	392
22.91.3.4 ReturnIndicesFromTable	392
22.91.3.5 Search	392
22.91.3.6 ToString	393
22.91.4 Member Data Documentation	393
22.91.4.1 bucketContentSize	393
22.91.4.2 bucketRowInHashTable	393
22.91.4.3 bucketSize	393
22.91.4.4 distancePtr	393
22.91.4.5 hashWidth	393
22.91.4.6 metric	394
22.91.4.7 neighborPtr	394
22.91.4.8 numProj	394
22.91.4.9 numTables	394
22.91.4.10 offsets	394
22.91.4.11 projections	394
22.91.4.12 querySet	394
22.91.4.13 referenceSet	395
22.91.4.14 secondHashSize	395
22.91.4.15 secondHashTable	395
22.91.4.16 secondHashWeights	395
22.92 mlpack::neighbor::NearestNeighborSort Class Reference	395
22.92.1 Detailed Description	396

22.92.2 Member Function Documentation	396
22.92.2.1 BestDistance	396
22.92.2.2 BestNodeToNodeDistance	397
22.92.2.3 BestNodeToNodeDistance	397
22.92.2.4 BestNodeToNodeDistance	397
22.92.2.5 BestPointToNodeDistance	397
22.92.2.6 BestPointToNodeDistance	397
22.92.2.7 CombineBest	398
22.92.2.8 CombineWorst	398
22.92.2.9 IsBetter	398
22.92.2.10 SortDistance	398
22.92.2.11 WorstDistance	398
22.93 mlpack::neighbor::NeighborSearch< SortPolicy, MetricType, TreeType > Class Template Reference	399
22.93.1 Detailed Description	400
22.93.2 Constructor & Destructor Documentation	400
22.93.2.1 NeighborSearch	400
22.93.2.2 NeighborSearch	401
22.93.2.3 NeighborSearch	401
22.93.2.4 NeighborSearch	402
22.93.2.5 ~NeighborSearch	402
22.93.3 Member Function Documentation	403
22.93.3.1 Search	403
22.93.3.2 ToString	403
22.93.4 Member Data Documentation	403
22.93.4.1 hasQuerySet	403
22.93.4.2 metric	403
22.93.4.3 naive	403
22.93.4.4 oldFromNewQueries	404
22.93.4.5 oldFromNewReferences	404
22.93.4.6 queryCopy	404
22.93.4.7 querySet	404
22.93.4.8 queryTree	404
22.93.4.9 referenceCopy	404
22.93.4.10 referenceSet	405
22.93.4.11 referenceTree	405
22.93.4.12 singleMode	405
22.93.4.13 freeOwner	405

22.94	mlpack::neighbor::NeighborSearchRules< SortPolicy, MetricType, TreeType > Class Template Reference	405
22.94.1	Detailed Description	407
22.94.2	Member Typedef Documentation	407
22.94.2.1	TraversallInfoType	407
22.94.3	Constructor & Destructor Documentation	407
22.94.3.1	NeighborSearchRules	407
22.94.4	Member Function Documentation	407
22.94.4.1	BaseCase	407
22.94.4.2	BaseCases	408
22.94.4.3	BaseCases	408
22.94.4.4	CalculateBound	408
22.94.4.5	InsertNeighbor	408
22.94.4.6	Rescore	408
22.94.4.7	Rescore	409
22.94.4.8	Score	409
22.94.4.9	Score	409
22.94.4.10	Scores	410
22.94.4.11	Scores	410
22.94.4.12	TraversallInfo	410
22.94.4.13	TraversallInfo	410
22.94.5	Member Data Documentation	410
22.94.5.1	baseCases	410
22.94.5.2	distances	410
22.94.5.3	lastBaseCase	411
22.94.5.4	lastQueryIndex	411
22.94.5.5	lastReferenceIndex	411
22.94.5.6	metric	411
22.94.5.7	neighbors	411
22.94.5.8	querySet	411
22.94.5.9	referenceSet	411
22.94.5.10	scores	412
22.94.5.11	traversallInfo	412
22.95	mlpack::neighbor::NeighborSearchStat< SortPolicy > Class Template Reference	412
22.95.1	Detailed Description	413
22.95.2	Constructor & Destructor Documentation	413
22.95.2.1	NeighborSearchStat	413
22.95.2.2	NeighborSearchStat	413

22.95.3 Member Function Documentation	413
22.95.3.1 Bound	413
22.95.3.2 Bound	414
22.95.3.3 FirstBound	414
22.95.3.4 FirstBound	414
22.95.3.5 LastDistance	414
22.95.3.6 LastDistance	414
22.95.3.7 LastDistanceNode	414
22.95.3.8 LastDistanceNode	415
22.95.3.9 SecondBound	415
22.95.3.10 SecondBound	415
22.95.4 Member Data Documentation	415
22.95.4.1 bound	415
22.95.4.2 firstBound	415
22.95.4.3 lastDistance	415
22.95.4.4 lastDistanceNode	416
22.95.4.5 secondBound	416
22.96mlpack::neighbor::NeighborSearchTraversallInfo< TreeType > Class Template Reference	416
22.96.1 Detailed Description	417
22.96.2 Constructor & Destructor Documentation	417
22.96.2.1 NeighborSearchTraversallInfo	417
22.96.3 Member Function Documentation	417
22.96.3.1 LastBaseCase	417
22.96.3.2 LastBaseCase	418
22.96.3.3 LastQueryNode	418
22.96.3.4 LastQueryNode	418
22.96.3.5 LastReferenceNode	418
22.96.3.6 LastReferenceNode	418
22.96.3.7 LastScore	418
22.96.3.8 LastScore	419
22.96.4 Member Data Documentation	419
22.96.4.1 lastBaseCase	419
22.96.4.2 lastQueryNode	419
22.96.4.3 lastReferenceNode	419
22.96.4.4 lastScore	419
22.97mlpack::neighbor::RASearchRules< SortPolicy, MetricType, TreeType > Class Template Reference	419
22.97.1 Detailed Description	421

22.97.2 Member Typedef Documentation	421
22.97.2.1 TraversalInfoType	421
22.97.3 Constructor & Destructor Documentation	422
22.97.3.1 RASearchRules	422
22.97.4 Member Function Documentation	422
22.97.4.1 BaseCase	422
22.97.4.2 InsertNeighbor	422
22.97.4.3 MinimumSamplesReqd	422
22.97.4.4 NumDistComputations	422
22.97.4.5 NumEffectiveSamples	423
22.97.4.6 ObtainDistinctSamples	423
22.97.4.7 Rescore	423
22.97.4.8 Rescore	423
22.97.4.9 Score	425
22.97.4.10Score	425
22.97.4.11Score	426
22.97.4.12Score	426
22.97.4.13Score	426
22.97.4.14Score	426
22.97.4.15SuccessProbability	427
22.97.4.16TraversalInfo	428
22.97.4.17TraversalInfo	428
22.97.5 Friends And Related Function Documentation	428
22.97.5.1 RASearch< SortPolicy, MetricType, TreeType >	428
22.97.6 Member Data Documentation	428
22.97.6.1 distances	428
22.97.6.2 firstLeafExact	428
22.97.6.3 metric	428
22.97.6.4 neighbors	429
22.97.6.5 numDistComputations	429
22.97.6.6 numSamplesMade	429
22.97.6.7 numSamplesReqd	429
22.97.6.8 querySet	429
22.97.6.9 referenceSet	429
22.97.6.10sampleAtLeaves	429
22.97.6.11samplingRatio	430
22.97.6.12singleSampleLimit	430

22.97.6.13 traversalInfo	430
22.98mlpack::nn::SparseAutoencoder< OptimizerType > Class Template Reference	430
22.98.1 Detailed Description	431
22.98.2 Constructor & Destructor Documentation	432
22.98.2.1 SparseAutoencoder	432
22.98.2.2 SparseAutoencoder	432
22.98.3 Member Function Documentation	432
22.98.3.1 Beta	433
22.98.3.2 Beta	433
22.98.3.3 GetNewFeatures	433
22.98.3.4 HiddenSize	433
22.98.3.5 HiddenSize	433
22.98.3.6 Lambda	433
22.98.3.7 Lambda	434
22.98.3.8 Rho	434
22.98.3.9 Rho	434
22.98.3.10 Sigmoid	434
22.98.3.11 VisibleSize	434
22.98.3.12 VisibleSize	434
22.98.4 Member Data Documentation	435
22.98.4.1 beta	435
22.98.4.2 hiddenSize	435
22.98.4.3 lambda	435
22.98.4.4 parameters	435
22.98.4.5 rho	435
22.98.4.6 visibleSize	435
22.99mlpack::nn::SparseAutoencoderFunction Class Reference	436
22.99.1 Detailed Description	437
22.99.2 Constructor & Destructor Documentation	437
22.99.2.1 SparseAutoencoderFunction	437
22.99.3 Member Function Documentation	437
22.99.3.1 Beta	437
22.99.3.2 Beta	437
22.99.3.3 Evaluate	438
22.99.3.4 GetInitialPoint	438
22.99.3.5 Gradient	438
22.99.3.6 HiddenSize	438

22.99.3.7 HiddenSize	438
22.99.3.8 InitializeWeights	438
22.99.3.9 Lambda	438
22.99.3.10 Lambda	439
22.99.3.11 Rho	439
22.99.3.12 Rho	439
22.99.3.13 Sigmoid	439
22.99.3.14 VisibleSize	439
22.99.3.15 VisibleSize	439
22.99.4 Member Data Documentation	440
22.99.4.1 beta	440
22.99.4.2 data	440
22.99.4.3 hiddenSize	440
22.99.4.4 initialPoint	440
22.99.4.5 lambda	440
22.99.4.6 rho	440
22.99.4.7 visibleSize	440
22.100.1 mlpack::optimization::AugLagrangian< LagrangianFunction > Class Template Reference	441
22.100.2 Detailed Description	442
22.100.3 Member Typedef Documentation	443
22.100.3.1 L_BFGSType	443
22.100.4 Constructor & Destructor Documentation	443
22.100.4.1 AugLagrangian	443
22.100.4.2 AugLagrangian	443
22.100.5 Member Function Documentation	443
22.100.5.1 Function	443
22.100.5.2 Function	443
22.100.5.3 lambda	444
22.100.5.4 lambda	444
22.100.5.5 LBFGS	444
22.100.5.6 LBFGS	444
22.100.5.7 Optimize	444
22.100.5.8 Optimize	444
22.100.5.9 Sigma	445
22.100.5.10 Sigma	445
22.100.5.11 ToString	445
22.100.6 Member Data Documentation	445

22.100.5.1augfunc	445
22.100.5.2function	445
22.100.5.3bfgs	445
22.100.5.4bfgsInternal	446
22.101.mpack::optimization::AugLagrangianFunction< LagrangianFunction > Class Template Reference	446
22.101.1Detailed Description	447
22.101.2Constructor & Destructor Documentation	447
22.101.2.1AugLagrangianFunction	447
22.101.2.2AugLagrangianFunction	448
22.101.3Member Function Documentation	448
22.101.3.1Evaluate	448
22.101.3.2Evaluate	448
22.101.3.3Function	448
22.101.3.4Function	448
22.101.3.5GetInitialPoint	449
22.101.3.6Gradient	449
22.101.3.7Gradient	449
22.101.3.8Lambda	449
22.101.3.9Lambda	449
22.101.3.10Sigma	449
22.101.3.11Sigma	449
22.101.3.12String	450
22.101.4Member Data Documentation	450
22.101.4.1function	450
22.101.4.2lambda	450
22.101.4.3sigma	450
22.102.mpack::optimization::AugLagrangianTestFunction Class Reference	450
22.102.1Detailed Description	451
22.102.2Constructor & Destructor Documentation	451
22.102.2.1AugLagrangianTestFunction	451
22.102.2.2AugLagrangianTestFunction	451
22.102.3Member Function Documentation	451
22.102.3.1Evaluate	451
22.102.3.2EvaluateConstraint	451
22.102.3.3GetInitialPoint	451
22.102.3.4Gradient	451
22.102.3.5GradientConstraint	451

22.102.3.6	NumConstraints	451
22.102.3.7	ToString	451
22.102.4	Member Data Documentation	451
22.102.4.1	InitialPoint	451
22.103	mlpack::optimization::ExponentialSchedule Class Reference	452
22.103.1	Detailed Description	452
22.103.2	Constructor & Destructor Documentation	452
22.103.2.1	ExponentialSchedule	452
22.103.3	Member Function Documentation	453
22.103.3.1	Lambda	453
22.103.3.2	lambda	453
22.103.3.3	NextTemperature	453
22.103.4	Member Data Documentation	453
22.103.4.1	lambda	453
22.104	mlpack::optimization::GockenbachFunction Class Reference	453
22.104.1	Detailed Description	454
22.104.2	Constructor & Destructor Documentation	454
22.104.2.1	GockenbachFunction	454
22.104.2.2	GockenbachFunction	454
22.104.3	Member Function Documentation	454
22.104.3.1	Evaluate	454
22.104.3.2	EvaluateConstraint	454
22.104.3.3	GetInitialPoint	454
22.104.3.4	Gradient	454
22.104.3.5	GradientConstraint	455
22.104.3.6	NumConstraints	455
22.104.4	Member Data Documentation	455
22.104.4.1	InitialPoint	455
22.105	mlpack::optimization::L_BFGS< FunctionType > Class Template Reference	455
22.105.1	Detailed Description	458
22.105.2	Constructor & Destructor Documentation	458
22.105.2.1	L_BFGS	458
22.105.3	Member Function Documentation	458
22.105.3.1	ArmijoConstant	458
22.105.3.2	ArmijoConstant	459
22.105.3.3	ChooseScalingFactor	459
22.105.3.4	Evaluate	459

22.105.3.5Function	459
22.105.3.6Function	459
22.105.3.7GradientNormTooSmall	459
22.105.3.8LineSearch	460
22.105.3.9MaxIterations	460
22.105.3.10MaxIterations	460
22.105.3.11MaxLineSearchTrials	460
22.105.3.12MaxLineSearchTrials	460
22.105.3.13MaxStep	460
22.105.3.14MaxStep	461
22.105.3.15minGradientNorm	461
22.105.3.16minGradientNorm	461
22.105.3.17minPointIterate	461
22.105.3.18minStep	461
22.105.3.19minStep	461
22.105.3.20NumBasis	461
22.105.3.21NumBasis	462
22.105.3.22Optimize	462
22.105.3.23Optimize	462
22.105.3.24SearchDirection	462
22.105.3.25String	463
22.105.3.26UpdateBasisSet	463
22.105.3.27Wolfe	463
22.105.3.28Wolfe	463
22.105.4Member Data Documentation	463
22.105.4.1armijoConstant	463
22.105.4.2Function	464
22.105.4.3maxIterations	464
22.105.4.4maxLineSearchTrials	464
22.105.4.5maxStep	464
22.105.4.6minGradientNorm	464
22.105.4.7minPointIterate	464
22.105.4.8minStep	465
22.105.4.9newIterateTmp	465
22.105.4.10NumBasis	465
22.105.4.11	465
22.105.4.12Wolfe	465

22.105.4.18	465
22.106.1mlpack::optimization::LovaszThetaSDP Class Reference	466
22.106.1Detailed Description	466
22.106.2Constructor & Destructor Documentation	466
22.106.2.1LovaszThetaSDP	466
22.106.2.2LovaszThetaSDP	466
22.106.3Member Function Documentation	467
22.106.3.1Edges	467
22.106.3.2Edges	467
22.106.3.3Evaluate	467
22.106.3.4EvaluateConstraint	467
22.106.3.5GetInitialPoint	467
22.106.3.6Gradient	467
22.106.3.7GradientConstraint	467
22.106.3.8NumConstraints	467
22.106.4Member Data Documentation	467
22.106.4.1edges	467
22.106.4.2initialPoint	467
22.106.4.3vertices	467
22.107.1mlpack::optimization::LRSDP Class Reference	468
22.107.1Detailed Description	469
22.107.2Constructor & Destructor Documentation	469
22.107.2.1LRSDP	469
22.107.2.2LRSDP	469
22.107.3Member Function Documentation	469
22.107.3.1A	469
22.107.3.2A	470
22.107.3.3AModes	470
22.107.3.4AModes	470
22.107.3.5AugLag	470
22.107.3.6AugLag	470
22.107.3.7B	470
22.107.3.8B	470
22.107.3.9C	470
22.107.3.10	471
22.107.3.11Function	471
22.107.3.12Function	471

22.107.3.10Optimize	471
22.107.3.11bString	471
22.107.4Member Data Documentation	471
22.107.4.1augLag	471
22.107.4.2function	471
22.108mlpack::optimization::LRSDPFunction Class Reference	472
22.108.1Detailed Description	473
22.108.2Constructor & Destructor Documentation	473
22.108.2.1LRSDPFunction	473
22.108.3Member Function Documentation	473
22.108.3.1A	473
22.108.3.2A	473
22.108.3.3AModes	473
22.108.3.4AModes	473
22.108.3.5B	474
22.108.3.6B	474
22.108.3.7C	474
22.108.3.8C	474
22.108.3.9Evaluate	474
22.108.3.10EvaluateConstraint	474
22.108.3.11GetInitialPoint	474
22.108.3.12Gradient	474
22.108.3.13GradientConstraint	475
22.108.3.14NumConstraints	475
22.108.3.15bString	475
22.108.4Member Data Documentation	475
22.108.4.1a	475
22.108.4.2aModes	475
22.108.4.3b	475
22.108.4.4c	475
22.108.4.5InitialPoint	475
22.109mlpack::optimization::SA< FunctionType, CoolingScheduleType > Class Template Reference	476
22.109.1Detailed Description	478
22.109.2Constructor & Destructor Documentation	478
22.109.2.1SA	478
22.109.3Member Function Documentation	479
22.109.3.1Function	479

22.109.3.2Function	479
22.109.3.3Gain	479
22.109.3.4Gain	479
22.109.3.5GenerateMove	480
22.109.3.6InitMoves	480
22.109.3.7InitMoves	480
22.109.3.8MaxIterations	480
22.109.3.9MaxIterations	480
22.109.3.10MaxMove	481
22.109.3.11MaxMove	481
22.109.3.12MaxToleranceSweep	481
22.109.3.13MaxToleranceSweep	481
22.109.3.14MoveControl	481
22.109.3.15MoveCtrlSweep	482
22.109.3.16MoveCtrlSweep	482
22.109.3.17MoveSize	482
22.109.3.18MoveSize	482
22.109.3.19Optimize	482
22.109.3.20Temperature	483
22.109.3.21Temperature	483
22.109.3.22Tolerance	483
22.109.3.23Tolerance	483
22.109.3.24To String	483
22.109.4Member Data Documentation	483
22.109.4.1coolingSchedule	483
22.109.4.2function	484
22.109.4.3gain	484
22.109.4.4initMoves	484
22.109.4.5maxIterations	484
22.109.4.6maxMove	484
22.109.4.7maxToleranceSweep	484
22.109.4.8moveCtrlSweep	485
22.109.4.9moveSize	485
22.109.4.10mperature	485
22.109.4.11tolerance	485
22.110mlpack::optimization::SGD< DecomposableFunctionType > Class Template Reference	485
22.110.1Detailed Description	487

22.110.2	Constructor & Destructor Documentation	488
22.110.2.1	ISGD	488
22.110.3	Member Function Documentation	488
22.110.3.1	Function	488
22.110.3.2	Function	488
22.110.3.3	MaxIterations	488
22.110.3.4	MaxIterations	488
22.110.3.5	Optimize	489
22.110.3.6	Optimize	489
22.110.3.7	Shuffle	489
22.110.3.8	Shuffle	489
22.110.3.9	StepSize	489
22.110.3.10	StepSize	489
22.110.3.11	Tolerance	490
22.110.3.12	Tolerance	490
22.110.3.13	ToString	490
22.110.4	Member Data Documentation	490
22.110.4.1	function	490
22.110.4.2	maxIterations	490
22.110.4.3	shuffle	490
22.110.4.4	stepSize	490
22.110.4.5	tolerance	491
22.111	mlpack::optimization::test::GeneralizedRosenbrockFunction Class Reference	491
22.111.1	Detailed Description	491
22.111.2	Constructor & Destructor Documentation	492
22.111.2.1	GeneralizedRosenbrockFunction	492
22.111.3	Member Function Documentation	492
22.111.3.1	Evaluate	492
22.111.3.2	Evaluate	492
22.111.3.3	GetInitialPoint	492
22.111.3.4	Gradient	492
22.111.3.5	Gradient	492
22.111.3.6	NumFunctions	492
22.111.4	Member Data Documentation	492
22.111.4.1	initialPoint	492
22.111.4.2	n	492
22.112	mlpack::optimization::test::RosenbrockFunction Class Reference	492

22.112.1	Detailed Description	493
22.112.2	Constructor & Destructor Documentation	493
22.112.2.1	RosenbrockFunction	493
22.112.3	Member Function Documentation	493
22.112.3.1	Evaluate	493
22.112.3.2	GetInitialPoint	493
22.112.3.3	Gradient	493
22.112.4	Member Data Documentation	493
22.112.4.1	InitialPoint	493
22.113	mlpack::optimization::test::RosenbrockWoodFunction Class Reference	493
22.113.1	Detailed Description	494
22.113.2	Constructor & Destructor Documentation	494
22.113.2.1	RosenbrockWoodFunction	494
22.113.3	Member Function Documentation	494
22.113.3.1	Evaluate	494
22.113.3.2	GetInitialPoint	494
22.113.3.3	Gradient	494
22.113.4	Member Data Documentation	494
22.113.4.1	InitialPoint	494
22.113.4.2	f	494
22.113.4.3	wf	494
22.114	mlpack::optimization::test::SGDTestFunction Class Reference	495
22.114.1	Detailed Description	495
22.114.2	Constructor & Destructor Documentation	495
22.114.2.1	SGDTestFunction	495
22.114.3	Member Function Documentation	495
22.114.3.1	Evaluate	495
22.114.3.2	GetInitialPoint	495
22.114.3.3	Gradient	495
22.114.3.4	NumFunctions	496
22.115	mlpack::optimization::test::WoodFunction Class Reference	496
22.115.1	Detailed Description	496
22.115.2	Constructor & Destructor Documentation	496
22.115.2.1	WoodFunction	496
22.115.3	Member Function Documentation	496
22.115.3.1	Evaluate	496
22.115.3.2	GetInitialPoint	496

22.115.3.3	Gradient	496
22.115.4	Member Data Documentation	496
22.115.4.1	InitialPoint	497
22.116	mlpack::ParamData Struct Reference	497
22.116.1	Detailed Description	497
22.116.2	Member Data Documentation	497
22.116.2.1	desc	497
22.116.2.2	isFlag	497
22.116.2.3	name	498
22.116.2.4	name	498
22.116.2.5	value	498
22.116.2.6	wasPassed	498
22.117	mlpack::pca::PCA Class Reference	498
22.117.1	Detailed Description	499
22.117.2	Constructor & Destructor Documentation	499
22.117.2.1	PCA	499
22.117.3	Member Function Documentation	499
22.117.3.1	Apply	499
22.117.3.2	Apply	499
22.117.3.3	Apply	500
22.117.3.4	Apply	500
22.117.3.5	Apply	500
22.117.3.6	ScaleData	500
22.117.3.7	ScaleData	501
22.117.3.8	ToString	501
22.117.4	Member Data Documentation	501
22.117.4.1	scaleData	501
22.118	mlpack::perceptron::Perceptron< LearnPolicy, WeightInitializationPolicy, MatType > Class Template Reference	501
22.118.1	Detailed Description	502
22.118.2	Constructor & Destructor Documentation	502
22.118.2.1	Perceptron	502
22.118.2.2	Perceptron	502
22.118.3	Member Function Documentation	503
22.118.3.1	Classify	503
22.118.3.2	Train	503
22.118.4	Member Data Documentation	503

22.118.4.1	classLabels	503
22.118.4.2	iter	503
22.118.4.3	trainData	504
22.118.4.4	weightVectors	504
22.119	mlpack::perceptron::RandomInitialization Class Reference	504
22.119.1	Detailed Description	504
22.119.2	Constructor & Destructor Documentation	504
22.119.2.1	RandomInitialization	504
22.119.3	Member Function Documentation	504
22.119.3.1	Initialize	504
22.120	mlpack::perceptron::SimpleWeightUpdate Class Reference	505
22.120.1	Detailed Description	505
22.120.2	Member Function Documentation	505
22.120.2.1	UpdateWeights	505
22.121	mlpack::perceptron::ZeroInitialization Class Reference	505
22.121.1	Detailed Description	506
22.121.2	Constructor & Destructor Documentation	506
22.121.2.1	ZeroInitialization	506
22.121.3	Member Function Documentation	506
22.121.3.1	Initialize	506
22.122	mlpack::radical::Radical Class Reference	506
22.122.1	Detailed Description	507
22.122.2	Constructor & Destructor Documentation	508
22.122.2.1	Radical	508
22.122.3	Member Function Documentation	509
22.122.3.1	Angles	509
22.122.3.2	Angles	509
22.122.3.3	CopyAndPerturb	509
22.122.3.4	DoRadical	509
22.122.3.5	DoRadical2D	509
22.122.3.6	NoiseStdDev	509
22.122.3.7	NoiseStdDev	510
22.122.3.8	Replicates	510
22.122.3.9	Replicates	510
22.122.3.10	Sweeps	510
22.122.3.11	Sweeps	510
22.122.3.12	String	510

22.122.3.1	Basic	510
22.122.4	Member Data Documentation	511
22.122.4.1	angles	511
22.122.4.2	candidate	511
22.122.4.3	m	511
22.122.4.4	noiseStdDev	511
22.122.4.5	perturbed	511
22.122.4.6	replicates	511
22.122.4.7	sweeps	511
22.123	mlpack::range::RangeSearch< MetricType, TreeType > Class Template Reference	512
22.123.1	Detailed Description	513
22.123.2	Constructor & Destructor Documentation	513
22.123.2.1	RangeSearch	513
22.123.2.2	RangeSearch	513
22.123.2.3	RangeSearch	514
22.123.2.4	RangeSearch	514
22.123.2.5	~RangeSearch	515
22.123.3	Member Function Documentation	515
22.123.3.1	Search	515
22.123.3.2	ToString	516
22.123.4	Member Data Documentation	516
22.123.4.1	hasQuerySet	516
22.123.4.2	metric	516
22.123.4.3	naive	516
22.123.4.4	numPrunes	516
22.123.4.5	oldFromNewQueries	516
22.123.4.6	oldFromNewReferences	517
22.123.4.7	queryCopy	517
22.123.4.8	querySet	517
22.123.4.9	queryTree	517
22.123.4.10	referenceCopy	517
22.123.4.11	referenceSet	517
22.123.4.12	referenceTree	518
22.123.4.13	singleMode	518
22.123.4.14	treeOwner	518
22.124	mlpack::range::RangeSearchRules< MetricType, TreeType > Class Template Reference	518
22.124.1	Detailed Description	519

22.124.1	Member Typedef Documentation	520
22.124.2.1	TraversalInfoType	520
22.124.3	Constructor & Destructor Documentation	520
22.124.3.1	RangeSearchRules	520
22.124.4	Member Function Documentation	520
22.124.4.1	AddResult	520
22.124.4.2	BaseCase	520
22.124.4.3	Rescore	520
22.124.4.4	Rescore	521
22.124.4.5	Score	521
22.124.4.6	Score	521
22.124.4.7	TraversalInfo	522
22.124.4.8	TraversalInfo	522
22.124.5	Member Data Documentation	522
22.124.5.1	distances	522
22.124.5.2	lastQueryIndex	522
22.124.5.3	lastReferenceIndex	522
22.124.5.4	metric	522
22.124.5.5	neighbors	522
22.124.5.6	querySet	523
22.124.5.7	range	523
22.124.5.8	referenceSet	523
22.124.5.9	traversalInfo	523
22.125	mlpack::range::RangeSearchStat Class Reference	523
22.125.1	Detailed Description	524
22.125.2	Constructor & Destructor Documentation	524
22.125.2.1	RangeSearchStat	524
22.125.2.2	RangeSearchStat	524
22.125.3	Member Function Documentation	524
22.125.3.1	LastDistance	524
22.125.3.2	LastDistance	524
22.125.3.3	LastDistanceNode	525
22.125.3.4	LastDistanceNode	525
22.125.4	Member Data Documentation	525
22.125.4.1	lastDistance	525
22.125.4.2	lastDistanceNode	525
22.126	mlpack::regression::LARS Class Reference	525

22.126.1 Detailed Description	527
22.126.2 Constructor & Destructor Documentation	528
22.126.2.1 LARS	528
22.126.2.2 LARS	528
22.126.3 Member Function Documentation	528
22.126.3.1 Activate	528
22.126.3.2 ActiveSet	528
22.126.3.3 BetaPath	528
22.126.3.4 CholeskyDelete	529
22.126.3.5 CholeskyInsert	529
22.126.3.6 CholeskyInsert	529
22.126.3.7 ComputeYHatDirection	529
22.126.3.8 Deactivate	529
22.126.3.9 GivensRotate	529
22.126.3.10 Ignore	529
22.126.3.11 InterpolateBeta	529
22.126.3.12 LambdaPath	529
22.126.3.13 MatUtriCholFactor	529
22.126.3.14 Regress	530
22.126.3.15 ToString	530
22.126.4 Member Data Documentation	530
22.126.4.1 activeSet	530
22.126.4.2 betaPath	530
22.126.4.3 elasticNet	530
22.126.4.4 ignoreSet	530
22.126.4.5 sActive	530
22.126.4.6 sIgnored	531
22.126.4.7 lambda1	531
22.126.4.8 lambda2	531
22.126.4.9 lambdaPath	531
22.126.4.10 Lasso	531
22.126.4.11 MatGram	531
22.126.4.12 MatGramInternal	531
22.126.4.13 MatUtriCholFactor	531
22.126.4.14 tolerance	532
22.126.4.15 UseCholesky	532
22.127 mlpack::regression::LinearRegression Class Reference	532

22.127.1	Detailed Description	533
22.127.2	Constructor & Destructor Documentation	533
22.127.2.1	LinearRegression	533
22.127.2.2	LinearRegression	533
22.127.2.3	LinearRegression	533
22.127.2.4	LinearRegression	533
22.127.3	Member Function Documentation	534
22.127.3.1	ComputeError	534
22.127.3.2	lambda	534
22.127.3.3	lambda	534
22.127.3.4	Parameters	534
22.127.3.5	Parameters	534
22.127.3.6	Predict	535
22.127.3.7	ToString	536
22.127.4	Member Data Documentation	536
22.127.4.1	intercept	536
22.127.4.2	lambda	536
22.127.4.3	parameters	536
22.128	mlpack::regression::LogisticRegression< OptimizerType > Class Template Reference	536
22.128.1	Detailed Description	537
22.128.2	Constructor & Destructor Documentation	537
22.128.2.1	LogisticRegression	537
22.128.2.2	LogisticRegression	538
22.128.2.3	LogisticRegression	538
22.128.2.4	LogisticRegression	538
22.128.3	Member Function Documentation	538
22.128.3.1	ComputeAccuracy	539
22.128.3.2	ComputeError	539
22.128.3.3	lambda	539
22.128.3.4	lambda	539
22.128.3.5	Parameters	540
22.128.3.6	Parameters	540
22.128.3.7	Predict	540
22.128.3.8	ToString	540
22.128.4	Member Data Documentation	540
22.128.4.1	lambda	540
22.128.4.2	parameters	540

22.129.0	mlpack::regression::LogisticRegressionFunction Class Reference	541
22.129.1	Detailed Description	542
22.129.2	Constructor & Destructor Documentation	542
22.129.2.1	LogisticRegressionFunction	542
22.129.2.2	LogisticRegressionFunction	542
22.129.3	Member Function Documentation	542
22.129.3.1	Evaluate	542
22.129.3.2	Evaluate	542
22.129.3.3	GetInitialPoint	543
22.129.3.4	Gradient	543
22.129.3.5	Gradient	543
22.129.3.6	InitialPoint	543
22.129.3.7	InitialPoint	543
22.129.3.8	Lambda	544
22.129.3.9	Lambda	544
22.129.3.10	NumFunctions	544
22.129.3.11	Predictors	544
22.129.3.12	Responses	544
22.129.4	Member Data Documentation	544
22.129.4.1	InitialPoint	544
22.129.4.2	lambda	544
22.129.4.3	predictors	545
22.129.4.4	responses	545
22.130.0	mlpack::sparse_coding::DataDependentRandomInitializer Class Reference	545
22.130.1	Detailed Description	545
22.130.2	Member Function Documentation	545
22.130.2.1	Initialize	545
22.131.0	mlpack::sparse_coding::NothingInitializer Class Reference	546
22.131.1	Detailed Description	546
22.131.2	Member Function Documentation	546
22.131.2.1	Initialize	546
22.132.0	mlpack::sparse_coding::RandomInitializer Class Reference	546
22.132.1	Detailed Description	547
22.132.2	Member Function Documentation	547
22.132.2.1	Initialize	547
22.133.0	mlpack::sparse_coding::SparseCoding< DictionaryInitializer > Class Template Reference	547
22.133.1	Detailed Description	548

22.133.2	Constructor & Destructor Documentation	549
22.133.2.1	SparseCoding	549
22.133.3	Member Function Documentation	550
22.133.3.1	Codes	550
22.133.3.2	Codes	550
22.133.3.3	Data	550
22.133.3.4	Dictionary	550
22.133.3.5	Dictionary	550
22.133.3.6	Encode	551
22.133.3.7	Objective	552
22.133.3.8	OptimizeCode	552
22.133.3.9	OptimizeDictionary	552
22.133.3.10	ProjectDictionary	552
22.133.3.11	ToString	552
22.133.4	Member Data Documentation	552
22.133.4.1	atoms	552
22.133.4.2	codes	553
22.133.4.3	data	553
22.133.4.4	dictionary	553
22.133.4.5	lambda1	553
22.133.4.6	lambda2	553
22.134	mlpack::svd::QUIC_SVD Class Reference	553
22.134.1	Detailed Description	554
22.134.2	Constructor & Destructor Documentation	554
22.134.2.1	QUIC_SVD	554
22.134.3	Member Function Documentation	554
22.134.3.1	ExtractSVD	554
22.134.4	Member Data Documentation	555
22.134.4.1	basis	555
22.134.4.2	dataset	555
22.134.4.3	delta	555
22.134.4.4	epsilon	555
22.135	mlpack::svd::RegularizedSVD< OptimizerType > Class Template Reference	555
22.135.1	Detailed Description	556
22.135.2	Constructor & Destructor Documentation	556
22.135.2.1	RegularizedSVD	556
22.135.3	Member Data Documentation	556

22.135.3.1alpha	556
22.135.3.2data	557
22.135.3.3iterations	557
22.135.3.4lambda	557
22.135.3.5optimizer	557
22.135.3.6rank	557
22.135.3.7SVDFunc	557
22.136mlpack::svd::RegularizedSVDFunction Class Reference	557
22.136.1Detailed Description	558
22.136.2Constructor & Destructor Documentation	559
22.136.2.1RegularizedSVDFunction	559
22.136.3Member Function Documentation	559
22.136.3.1Dataset	559
22.136.3.2Evaluate	559
22.136.3.3Evaluate	559
22.136.3.4GetInitialPoint	559
22.136.3.5Gradient	560
22.136.3.6Lambda	561
22.136.3.7NumFunctions	561
22.136.3.8NumItems	561
22.136.3.9NumUsers	561
22.136.3.10Rank	561
22.136.4Member Data Documentation	561
22.136.4.1data	561
22.136.4.2initialPoint	562
22.136.4.3lambda	562
22.136.4.4numItems	562
22.136.4.5numUsers	562
22.136.4.6rank	562
22.137mlpack::Timer Class Reference	562
22.137.1Detailed Description	563
22.137.2Member Function Documentation	563
22.137.2.1Get	563
22.137.2.2Start	563
22.137.2.3Stop	563
22.138mlpack::Timers Class Reference	564
22.138.1Detailed Description	564

22.138.2	Constructor & Destructor Documentation	564
22.138.2.1	Timers	564
22.138.3	Member Function Documentation	564
22.138.3.1	FileTimeToTimeVal	564
22.138.3.2	GetAllTimers	564
22.138.3.3	GetTime	565
22.138.3.4	GetTimer	565
22.138.3.5	PrintTimer	566
22.138.3.6	StartTimer	566
22.138.3.7	StopTimer	566
22.138.4	Member Data Documentation	566
22.138.4.1	timers	566
22.139	mlpack::tree::BinarySpaceTree< BoundType, StatisticType, MatType, SplitType > Class Template Reference	566
22.139.1	Detailed Description	571
22.139.2	Member Typedef Documentation	571
22.139.2.1	Mat	571
22.139.3	Constructor & Destructor Documentation	571
22.139.3.1	BinarySpaceTree	571
22.139.3.2	BinarySpaceTree	572
22.139.3.3	BinarySpaceTree	572
22.139.3.4	BinarySpaceTree	572
22.139.3.5	BinarySpaceTree	573
22.139.3.6	BinarySpaceTree	573
22.139.3.7	BinarySpaceTree	573
22.139.3.8	~BinarySpaceTree	574
22.139.3.9	BinarySpaceTree	574
22.139.4	Member Function Documentation	574
22.139.4.1	Begin	574
22.139.4.2	Begin	574
22.139.4.3	Bound	574
22.139.4.4	Bound	575
22.139.4.5	Centroid	575
22.139.4.6	Child	575
22.139.4.7	CopyMe	575
22.139.4.8	Count	575
22.139.4.9	Count	576

22.139.4.10Dataset	576
22.139.4.11Dataset	576
22.139.4.12Descendant	576
22.139.4.13End	576
22.139.4.14ExtendTree	576
22.139.4.15FindByBeginCount	577
22.139.4.16FindByBeginCount	577
22.139.4.17FurthestDescendantDistance	577
22.139.4.18FurthestPointDistance	577
22.139.4.19GetSplitDimension	578
22.139.4.20HasSelfChildren	578
22.139.4.21Leaf	578
22.139.4.22Left	578
22.139.4.23Left	578
22.139.4.24MaxDistance	578
22.139.4.25MaxDistance	579
22.139.4.26MaxLeafSize	579
22.139.4.27MaxLeafSize	579
22.139.4.28Metric	579
22.139.4.29MinDistance	579
22.139.4.30MinDistance	579
22.139.4.31MinimumBoundDistance	580
22.139.4.32NumChildren	580
22.139.4.33NumDescendants	580
22.139.4.34NumPoints	580
22.139.4.35Parent	580
22.139.4.36Parent	580
22.139.4.37ParentDistance	581
22.139.4.38ParentDistance	581
22.139.4.39Point	581
22.139.4.40RangeDistance	581
22.139.4.41RangeDistance	581
22.139.4.42Right	582
22.139.4.43Right	582
22.139.4.44SplitDimension	582
22.139.4.45SplitDimension	582
22.139.4.46SplitNode	582

22.139.4.4	SplitNode	583
22.139.4.4	Stat	583
22.139.4.4	Stat	583
22.139.4.5	ToString	583
22.139.4.5	TreeDepth	583
22.139.4.5	TreeSize	583
22.139.5	Member Data Documentation	584
22.139.5.1	begin	584
22.139.5.2	bound	584
22.139.5.3	count	584
22.139.5.4	dataset	584
22.139.5.5	furthestDescendantDistance	584
22.139.5.6	left	584
22.139.5.7	maxLeafSize	585
22.139.5.8	minimumBoundDistance	585
22.139.5.9	parent	585
22.139.5.10	parentDistance	585
22.139.5.11	right	585
22.139.5.12	splitDimension	586
22.139.5.13	stat	586
22.140	mlpack::tree::BinarySpaceTree< BoundType, StatisticType, MatType, SplitType >::DualTreeTraverser< RuleType > Class Template Reference	586
22.140.1	Detailed Description	587
22.140.2	Constructor & Destructor Documentation	587
22.140.2.1	DualTreeTraverser	587
22.140.3	Member Function Documentation	587
22.140.3.1	NumBaseCases	587
22.140.3.2	NumBaseCases	587
22.140.3.3	NumPrunes	588
22.140.3.4	NumPrunes	588
22.140.3.5	NumScores	588
22.140.3.6	NumScores	588
22.140.3.7	NumVisited	588
22.140.3.8	NumVisited	589
22.140.3.9	Traverse	589
22.140.4	Member Data Documentation	589
22.140.4.1	numBaseCases	589

22.140.4.2numPrunes	589
22.140.4.3numScores	589
22.140.4.4numVisited	590
22.140.4.5rule	590
22.140.4.6traversalInfo	590
22.141mlpack::tree::BinarySpaceTree< BoundType, StatisticType, MatType, SplitType >::SingleTree← Traverser< RuleType > Class Template Reference	590
22.141.1Detailed Description	591
22.141.2Constructor & Destructor Documentation	591
22.141.2.1SingleTreeTraverser	591
22.141.3Member Function Documentation	591
22.141.3.1NumPrunes	591
22.141.3.2NumPrunes	591
22.141.3.3Traverse	592
22.141.4Member Data Documentation	593
22.141.4.1numPrunes	593
22.141.4.2rule	593
22.142mlpack::tree::CompareCosineNode Class Reference	593
22.142.1Detailed Description	593
22.142.2Member Function Documentation	593
22.142.2.1operator()	593
22.143mlpack::tree::CosineTree Class Reference	594
22.143.1Detailed Description	596
22.143.2Constructor & Destructor Documentation	596
22.143.2.1CosineTree	596
22.143.2.2CosineTree	596
22.143.2.3CosineTree	596
22.143.2.4~CosineTree	597
22.143.3Member Function Documentation	597
22.143.3.1BasisVector	597
22.143.3.2BasisVector	597
22.143.3.3BinarySearch	597
22.143.3.4CalculateCentroid	597
22.143.3.5CalculateCosines	597
22.143.3.6Centroid	598
22.143.3.7ColumnSampleLS	598
22.143.3.8ColumnSamplesLS	598

22.143.3.9ConstructBasis	598
22.143.3.10CosineNodeSplit	598
22.143.3.11FrobNormSquared	598
22.143.3.12GetDataset	598
22.143.3.13GetFinalBasis	599
22.143.3.14Error	599
22.143.3.15Error	599
22.143.3.16ft	599
22.143.3.17ModifiedGramSchmidt	599
22.143.3.18MonteCarloError	599
22.143.3.19NumColumns	600
22.143.3.20ight	600
22.143.3.21SplitPointIndex	600
22.143.3.22ectorIndices	600
22.143.4Member Data Documentation	600
22.143.4.1basis	600
22.143.4.2basisVector	601
22.143.4.3centroid	601
22.143.4.4dataset	601
22.143.4.5delta	601
22.143.4.6epsilon	601
22.143.4.7frobNormSquared	601
22.143.4.8ndices	601
22.143.4.9Error	601
22.143.4.10NormsSquared	602
22.143.4.11ft	602
22.143.4.12umColumns	602
22.143.4.13arent	602
22.143.4.14ight	602
22.143.4.15plitPointIndex	602
22.144mlpack::tree::CoverTree< MetricType, RootPointPolicy, StatisticType > Class Template Reference	603
22.144.1Detailed Description	607
22.144.2Member Typedef Documentation	608
22.144.2.1Mat	608
22.144.3Constructor & Destructor Documentation	608
22.144.3.1CoverTree	608
22.144.3.2CoverTree	608

22.144.3.3CoverTree	609
22.144.3.4CoverTree	609
22.144.3.5CoverTree	610
22.144.3.6CoverTree	610
22.144.4Member Function Documentation	610
22.144.4.1Base	610
22.144.4.2Base	610
22.144.4.3Centroid	610
22.144.4.4Child	611
22.144.4.5Child	611
22.144.4.6Children	611
22.144.4.7Children	611
22.144.4.8ComputeDistances	611
22.144.4.9CreateChildren	612
22.144.4.10Dataset	612
22.144.4.11Descendant	612
22.144.4.12DistanceComps	612
22.144.4.13DistanceComps	612
22.144.4.14FurthestDescendantDistance	612
22.144.4.15FurthestDescendantDistance	613
22.144.4.16FurthestPointDistance	613
22.144.4.17HasSelfChildren	613
22.144.4.18Leaf	613
22.144.4.19MaxDistance	613
22.144.4.20MaxDistance	613
22.144.4.21MaxDistance	614
22.144.4.22MaxDistance	614
22.144.4.23Metric	614
22.144.4.24MinDistance	614
22.144.4.25MinDistance	614
22.144.4.26MinDistance	614
22.144.4.27MinDistance	614
22.144.4.28MinimumBoundDistance	615
22.144.4.29MoveToUsedSet	615
22.144.4.30NumChildren	615
22.144.4.31NumDescendants	615
22.144.4.32NumPoints	615

22.144.4.33	Parent	615
22.144.4.34	Parent	615
22.144.4.35	ParentDistance	616
22.144.4.36	ParentDistance	616
22.144.4.37	Point	616
22.144.4.38	Point	616
22.144.4.39	PruneFarSet	616
22.144.4.40	RangeDistance	616
22.144.4.41	RangeDistance	617
22.144.4.42	RangeDistance	617
22.144.4.43	RangeDistance	617
22.144.4.44	RemoveNewImplicitNodes	617
22.144.4.45	Scale	617
22.144.4.46	Scale	617
22.144.4.47	SortPointSet	618
22.144.4.48	SplitNearFar	618
22.144.4.49	Stat	618
22.144.4.50	Stat	618
22.144.4.51	ToString	619
22.144.5	Member Data Documentation	619
22.144.5.1	base	619
22.144.5.2	children	619
22.144.5.3	dataset	619
22.144.5.4	distanceComps	619
22.144.5.5	furthestDescendantDistance	620
22.144.5.6	localMetric	620
22.144.5.7	metric	620
22.144.5.8	numDescendants	620
22.144.5.9	parent	620
22.144.5.10	parentDistance	620
22.144.5.11	point	621
22.144.5.12	scale	621
22.144.5.13	stat	621
22.145	mlpack::tree::CoverTree< MetricType, RootPointPolicy, StatisticType >::DualTreeTraverser< RuleType > Class Template Reference	621
22.145.1	Detailed Description	622
22.145.2	Constructor & Destructor Documentation	622

22.145.2.1DualTreeTraverser	622
22.145.3Member Function Documentation	623
22.145.3.1NumBaseCases	623
22.145.3.2NumPrunes	623
22.145.3.3NumPrunes	623
22.145.3.4NumScores	623
22.145.3.5NumVisited	623
22.145.3.6PruneMap	623
22.145.3.7ReferenceRecursion	623
22.145.3.8Traverse	624
22.145.3.9Traverse	625
22.145.4Member Data Documentation	625
22.145.4.1numPrunes	625
22.145.4.2rule	625
22.146mlpack::tree::CoverTree< MetricType, RootPointPolicy, StatisticType >::DualTreeTraverser< RuleType >::DualCoverTreeMapEntry Struct Reference	625
22.146.Detailed Description	626
22.146.1Member Function Documentation	626
22.146.2.1operator<	626
22.146.3Member Data Documentation	626
22.146.3.1baseCase	626
22.146.3.2referenceNode	626
22.146.3.3score	627
22.146.3.4traversalInfo	627
22.147mlpack::tree::CoverTree< MetricType, RootPointPolicy, StatisticType >::SingleTreeTraverser< RuleType > Class Template Reference	627
22.147.Detailed Description	628
22.147.1Constructor & Destructor Documentation	628
22.147.2.1SingleTreeTraverser	628
22.147.3Member Function Documentation	628
22.147.3.1NumPrunes	628
22.147.3.2NumPrunes	628
22.147.3.3Traverse	628
22.147.4Member Data Documentation	628
22.147.4.1numPrunes	629
22.147.4.2rule	629
22.148mlpack::tree::EmptyStatistic Class Reference	629

22.148.1	Detailed Description	629
22.148.2	Constructor & Destructor Documentation	629
22.148.2.1	EmptyStatistic	629
22.148.2.2	EmptyStatistic	630
22.148.2.3	EmptyStatistic	630
22.148.3	Member Function Documentation	630
22.148.3.1	ToString	630
22.149	mlpack::tree::ExampleTree< MetricType, StatisticType, MatType > Class Template Reference	630
22.149.1	Detailed Description	631
22.149.2	Constructor & Destructor Documentation	632
22.149.2.1	ExampleTree	632
22.149.3	Member Function Documentation	632
22.149.3.1	Centroid	633
22.149.3.2	Child	634
22.149.3.3	Child	634
22.149.3.4	Descendant	634
22.149.3.5	FurthestDescendantDistance	634
22.149.3.6	MaxDistance	634
22.149.3.7	MaxDistance	635
22.149.3.8	Metric	635
22.149.3.9	Metric	635
22.149.3.10	MinDistance	635
22.149.3.11	MinDistance	635
22.149.3.12	NumChildren	636
22.149.3.13	NumDescendants	636
22.149.3.14	NumPoints	636
22.149.3.15	Parent	636
22.149.3.16	ParentDistance	636
22.149.3.17	Point	636
22.149.3.18	RangeDistance	636
22.149.3.19	RangeDistance	637
22.149.3.20	Stat	637
22.149.3.21	Stat	637
22.149.4	Member Data Documentation	637
22.149.4.1	metric	637
22.149.4.2	stat	638
22.150	mlpack::tree::FirstPointIsRoot Class Reference	638

22.150. Detailed Description	638
22.150. Member Function Documentation	638
22.150.2.1 ChooseRoot	638
22.151. mlpack::tree::MeanSplit< BoundType, MatType > Class Template Reference	638
22.151. Detailed Description	639
22.151. Member Function Documentation	639
22.151.2.1 PerformSplit	639
22.151.2.2 PerformSplit	639
22.151.2.3 SplitNode	640
22.151.2.4 SplitNode	640
22.152. mlpack::tree::MRKDStatistic Class Reference	640
22.152. Detailed Description	642
22.152. Constructor & Destructor Documentation	642
22.152.2.1 MRKDStatistic	642
22.152.2.2 MRKDStatistic	642
22.152. Member Function Documentation	642
22.152.3.1 Begin	642
22.152.3.2 Begin	643
22.152.3.3 CenterOfMass	643
22.152.3.4 CenterOfMass	643
22.152.3.5 Count	643
22.152.3.6 Count	643
22.152.3.7 DominatingCentroid	643
22.152.3.8 DominatingCentroid	643
22.152.3.9 ToString	644
22.152.3.10 Whitelist	644
22.152.3.11 Whitelist	644
22.152. Member Data Documentation	644
22.152.4.1 begin	644
22.152.4.2 centerOfMass	644
22.152.4.3 count	644
22.152.4.4 dataset	644
22.152.4.5 dominatingCentroid	645
22.152.4.6 sWhitelistValid	645
22.152.4.7 leftStat	645
22.152.4.8 parentStat	645
22.152.4.9 rightStat	645

22.152.4.1	SumOfSquaredNorms	645
22.152.4.1	Whitelist	645
22.153	mlpack::tree::TreeTraits< TreeType > Class Template Reference	645
22.153.1	Detailed Description	646
22.153.2	Member Data Documentation	647
22.153.2.1	FirstPointIsCentroid	647
22.153.2.2	HasOverlappingChildren	647
22.153.2.3	HasSelfChildren	647
22.153.2.4	RearrangesDataset	647
22.154	mlpack::tree::TreeTraits< BinarySpaceTree< BoundType, StatisticType, MatType > > Class Template Reference	647
22.154.1	Detailed Description	648
22.154.2	Member Data Documentation	648
22.154.2.1	FirstPointIsCentroid	648
22.154.2.2	HasOverlappingChildren	648
22.154.2.3	HasSelfChildren	648
22.154.2.4	RearrangesDataset	648
22.155	mlpack::tree::TreeTraits< CoverTree< MetricType, RootPointPolicy, StatisticType > > Class Template Reference	649
22.155.1	Detailed Description	649
22.155.2	Member Data Documentation	649
22.155.2.1	FirstPointIsCentroid	649
22.155.2.2	HasOverlappingChildren	649
22.155.2.3	HasSelfChildren	650
22.155.2.4	RearrangesDataset	650
22.156	mlpack::util::CLIDeleter Class Reference	650
22.156.1	Detailed Description	650
22.156.2	Constructor & Destructor Documentation	650
22.156.2.1	CLIDeleter	650
22.156.2.2	~CLIDeleter	650
22.157	mlpack::util::NullOutputStream Class Reference	651
22.157.1	Detailed Description	652
22.157.2	Constructor & Destructor Documentation	652
22.157.2.1	NullOutputStream	652
22.157.2.2	NullOutputStream	652
22.157.3	Member Function Documentation	652
22.157.3.1	operator<<	652

22.157.3.2	operator<<	652
22.157.3.3	operator<<	652
22.157.3.4	operator<<	652
22.157.3.5	operator<<	652
22.157.3.6	operator<<	653
22.157.3.7	operator<<	653
22.157.3.8	operator<<	653
22.157.3.9	operator<<	653
22.157.3.10	operator<<	653
22.157.3.11	operator<<	653
22.157.3.12	operator<<	653
22.157.3.13	operator<<	653
22.157.3.14	operator<<	653
22.157.3.15	operator<<	654
22.157.3.16	operator<<	654
22.157.3.17	operator<<	654
22.157.3.18	operator<<	654
22.158	mpack::util::Option< N > Class Template Reference	654
22.158.1	Detailed Description	654
22.158.2	Constructor & Destructor Documentation	655
22.158.2.1	Option	655
22.158.2.2	Option	655
22.159	mpack::util::PrefixedOutputStream Class Reference	655
22.159.1	Detailed Description	657
22.159.2	Constructor & Destructor Documentation	658
22.159.2.1	PrefixedOutputStream	658
22.159.3	Member Function Documentation	659
22.159.3.1	BaseLogic	659
22.159.3.2	CallBaseLogic	659
22.159.3.3	CallBaseLogic	659
22.159.3.4	CallBaseLogic	659
22.159.3.5	operator<<	659
22.159.3.6	operator<<	659
22.159.3.7	operator<<	659
22.159.3.8	operator<<	660
22.159.3.9	operator<<	660
22.159.3.10	operator<<	660

22.159.3.1operator<<	660
22.159.3.1operator<<	660
22.159.3.1operator<<	660
22.159.3.1operator<<	660
22.159.3.1operator<<	660
22.159.3.1operator<<	660
22.159.3.1operator<<	660
22.159.3.1operator<<	660
22.159.3.1operator<<	661
22.159.3.2operator<<	661
22.159.3.2operator<<	661
22.159.3.2operator<<	661
22.159.3.2PrefixIfNeeded	661
22.159.4Member Data Documentation	661
22.159.4.1carriageReturned	661
22.159.4.2destination	661
22.159.4.3fatal	661
22.159.4.4ignoreInput	661
22.159.4.5prefix	662
22.160mlpack::util::ProgramDoc Class Reference	662
22.160.1Detailed Description	662
22.160.2Constructor & Destructor Documentation	662
22.160.2.1ProgramDoc	662
22.160.3Member Data Documentation	663
22.160.3.1documentation	663
22.160.3.2programName	663
22.161mlpack::util::SaveRestoreUtility Class Reference	663
22.161.1Detailed Description	664
22.161.2Constructor & Destructor Documentation	664
22.161.2.1SaveRestoreUtility	664
22.161.2.2~SaveRestoreUtility	664
22.161.3Member Function Documentation	664
22.161.3.1LoadParameter	664
22.161.3.2LoadParameter	664
22.161.3.3LoadParameter	665
22.161.3.4LoadParameter	665
22.161.3.5LoadParameter	665

22.161.3.6	LoadParameter	665
22.161.3.7	ReadFile	665
22.161.3.8	RecurseOnNodes	665
22.161.3.9	SaveParameter	665
22.161.3.10	SaveParameter	665
22.161.3.11	SaveParameter	665
22.161.3.12	SaveParameter	665
22.161.3.13	SaveParameter	665
22.161.3.14	WriteFile	666
22.161.4	Member Data Documentation	666
22.161.4.1	parameters	666
22.162	RASearch< SortPolicy, MetricType, TreeType > Class Template Reference	666
22.162.1	Detailed Description	667
22.162.2	Constructor & Destructor Documentation	668
22.162.2.1	RASearch	668
22.162.2.2	RASearch	668
22.162.2.3	RASearch	669
22.162.2.4	RASearch	669
22.162.2.5	~RASearch	670
22.162.3	Member Function Documentation	670
22.162.3.1	ResetQueryTree	670
22.162.3.2	ResetRAQueryStat	670
22.162.3.3	Search	670
22.162.3.4	ToString	671
22.162.4	Member Data Documentation	671
22.162.4.1	hasQuerySet	671
22.162.4.2	metric	671
22.162.4.3	naive	671
22.162.4.4	numberOfPrunes	672
22.162.4.5	oldFromNewQueries	672
22.162.4.6	oldFromNewReferences	672
22.162.4.7	queryCopy	672
22.162.4.8	querySet	672
22.162.4.9	queryTree	672
22.162.4.10	referenceCopy	672
22.162.4.11	referenceSet	673
22.162.4.12	referenceTree	673

22.162.4.1	SingleMode	673
22.162.4.14	TreeOwner	673
22.163	TraversalInfo< TreeType > Class Template Reference	673
22.163.1	Detailed Description	674
22.163.2	Constructor & Destructor Documentation	675
22.163.2.1	TraversalInfo	675
22.163.3	Member Function Documentation	675
22.163.3.1	LastBaseCase	675
22.163.3.2	LastBaseCase	675
22.163.3.3	LastQueryNode	675
22.163.3.4	LastQueryNode	675
22.163.3.5	LastReferenceNode	675
22.163.3.6	LastReferenceNode	675
22.163.3.7	LastScore	676
22.163.3.8	LastScore	676
22.163.4	Member Data Documentation	676
22.163.4.1	LastBaseCase	676
22.163.4.2	LastQueryNode	676
22.163.4.3	LastReferenceNode	676
22.163.4.4	LastScore	676
23	File Documentation	677
23.1	doc/guide/build.hpp File Reference	677
23.2	doc/guide/iodoc.hpp File Reference	677
23.3	doc/guide/matrices.hpp File Reference	677
23.4	doc/guide/sample.hpp File Reference	677
23.5	doc/guide/timer.hpp File Reference	677
23.6	doc/tutorials/amf/amf.txt File Reference	677
23.6.1	Detailed Description	677
23.7	doc/tutorials/det/det.txt File Reference	677
23.7.1	Detailed Description	678
23.7.2	Function Documentation	678
23.7.2.1	\$V	678
23.7.2.2	alpha	678
23.7.3	Variable Documentation	678
23.7.3.1	estimation	678
23.7.3.2	now	678

23.7.3.3 regularization	678
23.7.3.4 Thus	679
23.8 doc/tutorials/emst/emst.txt File Reference	679
23.8.1 Detailed Description	679
23.9 doc/tutorials/fastmks/fastmks.txt File Reference	679
23.9.1 Detailed Description	679
23.10 doc/tutorials/kmeans/kmeans.txt File Reference	679
23.10.1 Detailed Description	679
23.11 doc/tutorials/linear_regression/linear_regression.txt File Reference	679
23.11.1 Detailed Description	680
23.12 doc/tutorials/neighbor_search/neighbor_search.txt File Reference	680
23.12.1 Detailed Description	680
23.13 doc/tutorials/range_search/range_search.txt File Reference	680
23.13.1 Detailed Description	680
23.14 doc/tutorials/tutorials.txt File Reference	680
23.14.1 Detailed Description	680
23.15 src/mlpack/CMakeLists.txt File Reference	681
23.15.1 Function Documentation	681
23.15.1.1 include_directories	681
23.16 src/mlpack/core/CMakeLists.txt File Reference	681
23.16.1 Function Documentation	681
23.16.1.1 add_subdirectory	681
23.16.1.2 set	681
23.17 src/mlpack/core/data/CMakeLists.txt File Reference	681
23.17.1 Function Documentation	681
23.17.1.1 set	681
23.17.1.2 set	682
23.18 src/mlpack/core/dists/CMakeLists.txt File Reference	682
23.18.1 Function Documentation	682
23.18.1.1 set	682
23.18.1.2 set	682
23.19 src/mlpack/core/kernels/CMakeLists.txt File Reference	682
23.19.1 Function Documentation	682
23.19.1.1 set	682
23.19.1.2 set	683
23.20 src/mlpack/core/math/CMakeLists.txt File Reference	683
23.20.1 Function Documentation	683

23.20.1.1 set	683
23.20.1.2 set	683
23.21src/mlpack/core/metrics/CMakeLists.txt File Reference	683
23.21.1 Function Documentation	683
23.21.1.1 set	683
23.21.1.2 set	683
23.22src/mlpack/core/optimizers/aug_lagrangian/CMakeLists.txt File Reference	684
23.22.1 Function Documentation	684
23.22.1.1 set	684
23.22.1.2 set	684
23.23src/mlpack/core/optimizers/CMakeLists.txt File Reference	684
23.23.1 Function Documentation	684
23.23.1.1 add_subdirectory	684
23.23.1.2 set	684
23.24src/mlpack/core/optimizers/lbfgs/CMakeLists.txt File Reference	684
23.24.1 Function Documentation	685
23.24.1.1 set	685
23.24.1.2 set	685
23.25src/mlpack/core/optimizers/lrsdp/CMakeLists.txt File Reference	685
23.25.1 Function Documentation	685
23.25.1.1 set	685
23.25.1.2 set	685
23.26src/mlpack/core/optimizers/sa/CMakeLists.txt File Reference	685
23.26.1 Function Documentation	685
23.26.1.1 set	685
23.26.1.2 set	685
23.27src/mlpack/core/optimizers/sgd/CMakeLists.txt File Reference	686
23.27.1 Function Documentation	686
23.27.1.1 set	686
23.27.1.2 set	686
23.28src/mlpack/core/tree/CMakeLists.txt File Reference	686
23.28.1 Function Documentation	686
23.28.1.1 set	686
23.28.1.2 set	687
23.29src/mlpack/core/util/CMakeLists.txt File Reference	687
23.29.1 Function Documentation	687
23.29.1.1 set	687

23.29.1.2 set	687
23.30src/mlpack/methods/amf/CMakeLists.txt File Reference	687
23.30.1 Function Documentation	687
23.30.1.1 set	687
23.30.1.2 set	687
23.31src/mlpack/methods/amf/init_rules/CMakeLists.txt File Reference	688
23.31.1 Function Documentation	688
23.31.1.1 set	688
23.31.1.2 set	688
23.32src/mlpack/methods/amf/termination_policies/CMakeLists.txt File Reference	688
23.32.1 Function Documentation	688
23.32.1.1 set	688
23.32.1.2 set	688
23.33src/mlpack/methods/amf/update_rules/CMakeLists.txt File Reference	688
23.33.1 Function Documentation	689
23.33.1.1 set	689
23.33.1.2 set	689
23.34src/mlpack/methods/cf/CMakeLists.txt File Reference	689
23.34.1 Function Documentation	689
23.34.1.1 set	689
23.34.1.2 set	689
23.35src/mlpack/methods/CMakeLists.txt File Reference	689
23.35.1 Function Documentation	689
23.35.1.1 add_subdirectory	689
23.35.1.2 set	690
23.36src/mlpack/methods/decision_stump/CMakeLists.txt File Reference	690
23.36.1 Function Documentation	690
23.36.1.1 cmake_minimum_required	690
23.36.1.2 set	690
23.37src/mlpack/methods/det/CMakeLists.txt File Reference	690
23.37.1 Function Documentation	690
23.37.1.1 set	690
23.37.1.2 set	690
23.38src/mlpack/methods/emst/CMakeLists.txt File Reference	691
23.38.1 Function Documentation	691
23.38.1.1 set	691
23.38.1.2 set	691

23.39src/mlpack/methods/fastmks/CMakeLists.txt File Reference	691
23.39.1 Function Documentation	691
23.39.1.1 set	691
23.39.1.2 set	691
23.40src/mlpack/methods/gmm/CMakeLists.txt File Reference	691
23.40.1 Function Documentation	692
23.40.1.1 set	692
23.40.1.2 set	692
23.41src/mlpack/methods/hmm/CMakeLists.txt File Reference	692
23.41.1 Function Documentation	692
23.41.1.1 set	692
23.41.1.2 set	692
23.42src/mlpack/methods/kernel_pca/CMakeLists.txt File Reference	692
23.42.1 Function Documentation	692
23.42.1.1 set	692
23.42.1.2 set	692
23.43src/mlpack/methods/kernel_pca/kernel_rules/CMakeLists.txt File Reference	693
23.43.1 Function Documentation	693
23.43.1.1 set	693
23.43.1.2 set	693
23.44src/mlpack/methods/kmeans/CMakeLists.txt File Reference	693
23.44.1 Function Documentation	693
23.44.1.1 set	693
23.44.1.2 set	693
23.45src/mlpack/methods/lars/CMakeLists.txt File Reference	693
23.45.1 Function Documentation	694
23.45.1.1 set	694
23.45.1.2 set	694
23.46src/mlpack/methods/linear_regression/CMakeLists.txt File Reference	694
23.46.1 Function Documentation	694
23.46.1.1 set	694
23.46.1.2 set	694
23.47src/mlpack/methods/local_coordinate_coding/CMakeLists.txt File Reference	694
23.47.1 Function Documentation	694
23.47.1.1 set	694
23.47.1.2 set	694
23.48src/mlpack/methods/logistic_regression/CMakeLists.txt File Reference	695

23.48.1 Function Documentation	695
23.48.1.1 set	695
23.48.1.2 set	695
23.49src/mlpack/methods/lsh/CMakeLists.txt File Reference	695
23.49.1 Function Documentation	695
23.49.1.1 set	695
23.49.1.2 set	695
23.50src/mlpack/methods/mvu/CMakeLists.txt File Reference	695
23.50.1 Function Documentation	696
23.50.1.1 set	696
23.50.1.2 set	696
23.51src/mlpack/methods/naive_bayes/CMakeLists.txt File Reference	696
23.51.1 Function Documentation	696
23.51.1.1 set	696
23.51.1.2 set	696
23.52src/mlpack/methods/nca/CMakeLists.txt File Reference	696
23.52.1 Function Documentation	696
23.52.1.1 set	696
23.52.1.2 set	696
23.53src/mlpack/methods/neighbor_search/CMakeLists.txt File Reference	697
23.53.1 Function Documentation	697
23.53.1.1 set	697
23.53.1.2 set	697
23.54src/mlpack/methods/nmf/CMakeLists.txt File Reference	697
23.55src/mlpack/methods/nystroem_method/CMakeLists.txt File Reference	697
23.55.1 Function Documentation	697
23.55.1.1 set	697
23.55.1.2 set	697
23.56src/mlpack/methods/pca/CMakeLists.txt File Reference	698
23.56.1 Function Documentation	698
23.56.1.1 set	698
23.56.1.2 set	698
23.57src/mlpack/methods/perceptron/CMakeLists.txt File Reference	698
23.57.1 Function Documentation	698
23.57.1.1 cmake_minimum_required	698
23.57.1.2 set	698
23.58src/mlpack/methods/perceptron/initialization_methods/CMakeLists.txt File Reference	698

23.58.1 Function Documentation	699
23.58.1.1 set	699
23.58.1.2 set	699
23.59src/mlpack/methods/perceptron/learning_policies/CMakeLists.txt File Reference	699
23.59.1 Function Documentation	699
23.59.1.1 set	699
23.59.1.2 set	699
23.60src/mlpack/methods/quic_svd/CMakeLists.txt File Reference	699
23.60.1 Function Documentation	699
23.60.1.1 set	699
23.60.1.2 set	699
23.61src/mlpack/methods/radical/CMakeLists.txt File Reference	700
23.61.1 Function Documentation	700
23.61.1.1 set	700
23.61.1.2 set	700
23.62src/mlpack/methods/range_search/CMakeLists.txt File Reference	700
23.62.1 Function Documentation	700
23.62.1.1 set	700
23.62.1.2 set	700
23.63src/mlpack/methods/rann/CMakeLists.txt File Reference	700
23.63.1 Function Documentation	701
23.63.1.1 set	701
23.63.1.2 set	701
23.64src/mlpack/methods/regularized_svd/CMakeLists.txt File Reference	701
23.64.1 Function Documentation	701
23.64.1.1 set	701
23.64.1.2 set	701
23.65src/mlpack/methods/sparse_autoencoder/CMakeLists.txt File Reference	701
23.65.1 Function Documentation	701
23.65.1.1 set	701
23.65.1.2 set	702
23.66src/mlpack/methods/sparse_coding/CMakeLists.txt File Reference	702
23.66.1 Function Documentation	702
23.66.1.1 set	702
23.66.1.2 set	702
23.67src/mlpack/tests/CMakeLists.txt File Reference	702
23.67.1 Function Documentation	702

23.67.1.1 add_custom_command	702
23.67.1.2 add_executable	703
23.68src/mlpack/core.hpp File Reference	703
23.69src/mlpack/core/data/load.hpp File Reference	703
23.69.1 Detailed Description	704
23.70src/mlpack/core/data/normalize_labels.hpp File Reference	704
23.70.1 Detailed Description	705
23.71src/mlpack/core/data/save.hpp File Reference	705
23.71.1 Detailed Description	706
23.72src/mlpack/core/dists/discrete_distribution.hpp File Reference	706
23.72.1 Detailed Description	707
23.73src/mlpack/core/dists/gaussian_distribution.hpp File Reference	707
23.73.1 Detailed Description	707
23.74src/mlpack/core/dists/laplace_distribution.hpp File Reference	708
23.75src/mlpack/core/kernels/cosine_distance.hpp File Reference	708
23.75.1 Detailed Description	709
23.76src/mlpack/core/kernels/epanechnikov_kernel.hpp File Reference	709
23.76.1 Detailed Description	710
23.77src/mlpack/core/kernels/example_kernel.hpp File Reference	710
23.77.1 Detailed Description	710
23.78src/mlpack/core/kernels/gaussian_kernel.hpp File Reference	711
23.78.1 Detailed Description	711
23.79src/mlpack/core/kernels/hyperbolic_tangent_kernel.hpp File Reference	712
23.79.1 Detailed Description	712
23.80src/mlpack/core/kernels/kernel_traits.hpp File Reference	712
23.80.1 Detailed Description	713
23.81src/mlpack/core/kernels/laplacian_kernel.hpp File Reference	713
23.81.1 Detailed Description	714
23.82src/mlpack/core/kernels/linear_kernel.hpp File Reference	714
23.82.1 Detailed Description	715
23.83src/mlpack/core/kernels/polynomial_kernel.hpp File Reference	715
23.83.1 Detailed Description	715
23.84src/mlpack/core/kernels/pspectrum_string_kernel.hpp File Reference	716
23.84.1 Detailed Description	716
23.85src/mlpack/core/kernels/spherical_kernel.hpp File Reference	717
23.85.1 Detailed Description	717
23.86src/mlpack/core/kernels/triangular_kernel.hpp File Reference	718

23.86.1 Detailed Description	718
23.87src/mlpack/core/math/clamp.hpp File Reference	718
23.87.1 Detailed Description	719
23.88src/mlpack/core/math/lin_alg.hpp File Reference	720
23.88.1 Detailed Description	721
23.89src/mlpack/core/math/random.hpp File Reference	721
23.89.1 Detailed Description	722
23.90src/mlpack/core/math/range.hpp File Reference	722
23.90.1 Detailed Description	723
23.91src/mlpack/core/math/round.hpp File Reference	723
23.91.1 Detailed Description	723
23.92src/mlpack/core/metrics/ip_metric.hpp File Reference	724
23.92.1 Detailed Description	724
23.93src/mlpack/core/metrics/lmetric.hpp File Reference	725
23.93.1 Detailed Description	725
23.94src/mlpack/core/metrics/mahalanobis_distance.hpp File Reference	726
23.95src/mlpack/core/optimizers/aug_lagrangian/aug_lagrangian.hpp File Reference	726
23.95.1 Detailed Description	727
23.96src/mlpack/core/optimizers/aug_lagrangian/aug_lagrangian_function.hpp File Reference	728
23.96.1 Detailed Description	729
23.97src/mlpack/core/optimizers/aug_lagrangian/aug_lagrangian_test_functions.hpp File Reference	729
23.97.1 Detailed Description	729
23.98src/mlpack/core/optimizers/lbfgs/lbfgs.hpp File Reference	730
23.98.1 Detailed Description	730
23.99src/mlpack/core/optimizers/lbfgs/test_functions.hpp File Reference	731
23.99.1 Detailed Description	731
23.100src/mlpack/core/optimizers/lrsdp/lrsdp.hpp File Reference	732
23.100.1 Detailed Description	732
23.101src/mlpack/core/optimizers/lrsdp/lrsdp_function.hpp File Reference	732
23.101.1 Detailed Description	733
23.102src/mlpack/core/optimizers/sa/exponential_schedule.hpp File Reference	734
23.102.1 Detailed Description	734
23.103src/mlpack/core/optimizers/sa/sa.hpp File Reference	735
23.103.1 Detailed Description	735
23.104src/mlpack/core/optimizers/sgd/sgd.hpp File Reference	735
23.104.1 Detailed Description	736
23.105src/mlpack/core/optimizers/sgd/test_function.hpp File Reference	737

23.105. Detailed Description	737
23.106rc/mlpack/core/tree/ballbound.hpp File Reference	737
23.106. Detailed Description	738
23.107rc/mlpack/core/tree/binary_space_tree/binary_space_tree.hpp File Reference	738
23.108rc/mlpack/core/tree/binary_space_tree.hpp File Reference	739
23.109rc/mlpack/core/tree/binary_space_tree/dual_tree_traverser.hpp File Reference	740
23.110rc/mlpack/core/tree/cover_tree/dual_tree_traverser.hpp File Reference	741
23.111rc/mlpack/core/tree/binary_space_tree/mean_split.hpp File Reference	742
23.111. Detailed Description	743
23.112rc/mlpack/core/tree/binary_space_tree/single_tree_traverser.hpp File Reference	743
23.113rc/mlpack/core/tree/cover_tree/single_tree_traverser.hpp File Reference	744
23.114rc/mlpack/core/tree/binary_space_tree/traits.hpp File Reference	745
23.115rc/mlpack/core/tree/cover_tree/traits.hpp File Reference	746
23.116rc/mlpack/core/tree/bounds.hpp File Reference	747
23.116. Detailed Description	747
23.117rc/mlpack/core/tree/cosine_tree/cosine_tree.hpp File Reference	748
23.117. Detailed Description	748
23.118rc/mlpack/core/tree/cover_tree/cover_tree.hpp File Reference	749
23.119rc/mlpack/core/tree/cover_tree.hpp File Reference	750
23.120rc/mlpack/core/tree/cover_tree/first_point_is_root.hpp File Reference	750
23.120. Detailed Description	751
23.121rc/mlpack/core/tree/example_tree.hpp File Reference	752
23.121. Detailed Description	752
23.122rc/mlpack/core/tree/hrectbound.hpp File Reference	752
23.122. Detailed Description	753
23.123rc/mlpack/core/tree/mrkd_statistic.hpp File Reference	753
23.123. Detailed Description	754
23.124rc/mlpack/core/tree/rectangle_tree.hpp File Reference	754
23.124. Detailed Description	754
23.125rc/mlpack/core/tree/statistic.hpp File Reference	755
23.125. Detailed Description	755
23.126rc/mlpack/core/tree/traversal_info.hpp File Reference	755
23.126. Detailed Description	756
23.127rc/mlpack/core/tree/TREE_EXPLANATION.txt File Reference	756
23.127. Variable Documentation	756
23.127.1.1bottom	756
23.127.1.2separately	756

23.127.1.3supported	757
23.127.1.4top	757
23.127.1.5tree	757
23.128rc/mlpack/core/tree/tree_traits.hpp File Reference	757
23.128.1Detailed Description	758
23.129rc/mlpack/core/util/arma_traits.hpp File Reference	758
23.129.1Detailed Description	758
23.130rc/mlpack/core/util/cli.hpp File Reference	759
23.130.1Detailed Description	760
23.130.2Macro Definition Documentation	760
23.130.2.1PARAM	760
23.130.2.2PARAM_DOUBLE	761
23.130.2.3PARAM_DOUBLE_REQ	761
23.130.2.4PARAM_FLAG	762
23.130.2.5PARAM_FLOAT	762
23.130.2.6PARAM_FLOAT_REQ	762
23.130.2.7PARAM_INT	763
23.130.2.8PARAM_INT_REQ	763
23.130.2.9PARAM_STRING	764
23.130.2.10PARAM_STRING_REQ	764
23.130.2.11PARAM_VECTOR	765
23.130.2.12PARAM_VECTOR_REQ	765
23.130.2.13PROGRAM_INFO	766
23.130.2.14TYPE_NAME	766
23.131rc/mlpack/core/util/cli_deleter.hpp File Reference	767
23.131.1Detailed Description	767
23.132rc/mlpack/core/util/log.hpp File Reference	767
23.132.1Detailed Description	768
23.133rc/mlpack/core/util/nullostream.hpp File Reference	768
23.133.1Detailed Description	769
23.134rc/mlpack/core/util/option.hpp File Reference	769
23.134.1Detailed Description	770
23.135rc/mlpack/core/util/prefixedostream.hpp File Reference	770
23.135.1Detailed Description	770
23.136rc/mlpack/core/util/save_restore_utility.hpp File Reference	771
23.136.1Detailed Description	771
23.137rc/mlpack/core/util/sfinae_utility.hpp File Reference	772

23.137. Detailed Description	772
23.137.2. Macro Definition Documentation	772
23.137.2.1 HAS_MEM_FUNC	772
23.138rc/mlpack/core/util/string_util.hpp File Reference	773
23.138. Detailed Description	773
23.139rc/mlpack/core/util/timers.hpp File Reference	774
23.139. Detailed Description	774
23.140rc/mlpack/core/util/version.hpp File Reference	775
23.140. Macro Definition Documentation	775
23.140.1.1 __MLPACK_VERSION_MAJOR	775
23.140.1.2 __MLPACK_VERSION_MINOR	776
23.140.1.3 __MLPACK_VERSION_PATCH	776
23.141doc/guide/version.hpp File Reference	776
23.142rc/mlpack/methods/amf/amf.hpp File Reference	776
23.142. Detailed Description	777
23.143rc/mlpack/methods/amf/init_rules/average_init.hpp File Reference	777
23.144rc/mlpack/methods/amf/init_rules/random_acol_init.hpp File Reference	777
23.144. Detailed Description	778
23.145rc/mlpack/methods/amf/init_rules/random_init.hpp File Reference	778
23.146rc/mlpack/methods/perceptron/initialization_methods/random_init.hpp File Reference	779
23.147rc/mlpack/methods/amf/termination_policies/complete_incremental_termination.hpp File Reference	780
23.148rc/mlpack/methods/amf/termination_policies/incomplete_incremental_termination.hpp File Reference	781
23.148. Detailed Description	781
23.149rc/mlpack/methods/amf/termination_policies/simple_residue_termination.hpp File Reference	781
23.149. Detailed Description	782
23.150rc/mlpack/methods/amf/termination_policies/simple_tolerance_termination.hpp File Reference	783
23.150. Detailed Description	783
23.151rc/mlpack/methods/amf/termination_policies/validation_RMSE_termination.hpp File Reference	784
23.151. Detailed Description	784
23.152rc/mlpack/methods/amf/update_rules/nmf_als.hpp File Reference	784
23.152. Detailed Description	785
23.153rc/mlpack/methods/amf/update_rules/nmf_mult_dist.hpp File Reference	785
23.153. Detailed Description	786
23.154rc/mlpack/methods/amf/update_rules/nmf_mult_div.hpp File Reference	787
23.155rc/mlpack/methods/amf/update_rules/svd_batch_learning.hpp File Reference	787
23.156rc/mlpack/methods/amf/update_rules/svd_complete_incremental_learning.hpp File Reference	788
23.157rc/mlpack/methods/amf/update_rules/svd_incomplete_incremental_learning.hpp File Reference	788

23.158rc/mlpack/methods/cf/cf.hpp File Reference	789
23.158. Detailed Description	789
23.159rc/mlpack/methods/decision_stump/decision_stump.hpp File Reference	790
23.159. Detailed Description	790
23.160rc/mlpack/methods/det/dt_utils.hpp File Reference	790
23.160. Detailed Description	791
23.161rc/mlpack/methods/det/dtree.hpp File Reference	791
23.161. Detailed Description	792
23.162rc/mlpack/methods/emst/dtb.hpp File Reference	792
23.162. Detailed Description	793
23.163rc/mlpack/methods/emst/dtb_rules.hpp File Reference	793
23.164rc/mlpack/methods/emst/dtb_stat.hpp File Reference	794
23.165rc/mlpack/methods/emst/edge_pair.hpp File Reference	795
23.165. Detailed Description	795
23.166rc/mlpack/methods/emst/union_find.hpp File Reference	796
23.166. Detailed Description	796
23.167rc/mlpack/methods/fastmks/fastmks.hpp File Reference	797
23.167. Detailed Description	797
23.168rc/mlpack/methods/fastmks/fastmks_rules.hpp File Reference	798
23.168. Detailed Description	798
23.169rc/mlpack/methods/fastmks/fastmks_stat.hpp File Reference	798
23.169. Detailed Description	799
23.170rc/mlpack/methods/gmm/diagonal_constraint.hpp File Reference	799
23.170. Detailed Description	800
23.171rc/mlpack/methods/gmm/eigenvalue_ratio_constraint.hpp File Reference	800
23.171. Detailed Description	801
23.172rc/mlpack/methods/gmm/em_fit.hpp File Reference	801
23.172. Detailed Description	802
23.173rc/mlpack/methods/gmm/gmm.hpp File Reference	802
23.173. Detailed Description	802
23.174rc/mlpack/methods/gmm/no_constraint.hpp File Reference	803
23.174. Detailed Description	803
23.175rc/mlpack/methods/gmm/phi.hpp File Reference	803
23.175. Detailed Description	804
23.176rc/mlpack/methods/gmm/positive_definite_constraint.hpp File Reference	804
23.176. Detailed Description	805
23.177rc/mlpack/methods/hmm/hmm.hpp File Reference	806

23.177. Detailed Description	806
23.178rc/mlpack/methods/hmm/hmm_util.hpp File Reference	807
23.178. Detailed Description	807
23.179rc/mlpack/methods/kernel_pca/kernel_pca.hpp File Reference	807
23.179. Detailed Description	808
23.180rc/mlpack/methods/kernel_pca/kernel_rules/naive_method.hpp File Reference	808
23.180. Detailed Description	809
23.181rc/mlpack/methods/kernel_pca/kernel_rules/nystroem_method.hpp File Reference	809
23.182rc/mlpack/methods/nystroem_method/nystroem_method.hpp File Reference	810
23.183rc/mlpack/methods/kmeans/allow_empty_clusters.hpp File Reference	810
23.183. Detailed Description	811
23.184rc/mlpack/methods/kmeans/kmeans.hpp File Reference	811
23.184. Detailed Description	812
23.185rc/mlpack/methods/kmeans/max_variance_new_cluster.hpp File Reference	813
23.185. Detailed Description	814
23.186rc/mlpack/methods/kmeans/random_partition.hpp File Reference	814
23.186. Detailed Description	815
23.187rc/mlpack/methods/kmeans/refined_start.hpp File Reference	816
23.187. Detailed Description	816
23.188rc/mlpack/methods/lars/lars.hpp File Reference	817
23.188. Detailed Description	817
23.189rc/mlpack/methods/linear_regression/linear_regression.hpp File Reference	818
23.189. Detailed Description	818
23.190rc/mlpack/methods/local_coordinate_coding/lcc.hpp File Reference	819
23.190. Detailed Description	819
23.191rc/mlpack/methods/logistic_regression/logistic_regression.hpp File Reference	819
23.191. Detailed Description	820
23.192rc/mlpack/methods/logistic_regression/logistic_regression_function.hpp File Reference	820
23.192. Detailed Description	821
23.193rc/mlpack/methods/lsh/lsh_search.hpp File Reference	821
23.193. Detailed Description	822
23.194rc/mlpack/methods/mvu/mvu.hpp File Reference	822
23.194. Detailed Description	822
23.195rc/mlpack/methods/naive_bayes/naive_bayes_classifier.hpp File Reference	823
23.195. Detailed Description	823
23.196rc/mlpack/methods/nca/nca.hpp File Reference	823
23.196. Detailed Description	824

23.197rc/mlpack/methods/nca/nca_softmax_error_function.hpp File Reference	824
23.197. Detailed Description	825
23.198rc/mlpack/methods/neighbor_search/neighbor_search.hpp File Reference	825
23.198. Detailed Description	826
23.199rc/mlpack/methods/neighbor_search/neighbor_search_rules.hpp File Reference	826
23.199. Detailed Description	827
23.200rc/mlpack/methods/neighbor_search/neighbor_search_stat.hpp File Reference	827
23.201rc/mlpack/methods/neighbor_search/ns_traversal_info.hpp File Reference	828
23.201. Detailed Description	828
23.202rc/mlpack/methods/neighbor_search/sort_policies/furthest_neighbor_sort.hpp File Reference	829
23.202. Detailed Description	830
23.203rc/mlpack/methods/neighbor_search/sort_policies/nearest_neighbor_sort.hpp File Reference	830
23.203. Detailed Description	831
23.204rc/mlpack/methods/neighbor_search/typedef.hpp File Reference	831
23.204. Detailed Description	832
23.205rc/mlpack/methods/neighbor_search/unmap.hpp File Reference	833
23.205. Detailed Description	833
23.206rc/mlpack/methods/nystroem_method/kmeans_selection.hpp File Reference	834
23.206. Detailed Description	834
23.207rc/mlpack/methods/nystroem_method/ordered_selection.hpp File Reference	835
23.207. Detailed Description	835
23.208rc/mlpack/methods/nystroem_method/random_selection.hpp File Reference	836
23.208. Detailed Description	836
23.209rc/mlpack/methods/pca/pca.hpp File Reference	836
23.209. Detailed Description	837
23.210rc/mlpack/methods/perceptron/initialization_methods/zero_init.hpp File Reference	837
23.210. Detailed Description	838
23.211rc/mlpack/methods/perceptron/learning_policies/simple_weight_update.hpp File Reference	838
23.211. Detailed Description	839
23.212rc/mlpack/methods/perceptron/perceptron.hpp File Reference	839
23.212. Detailed Description	839
23.213rc/mlpack/methods/quic_svd/quic_svd.hpp File Reference	840
23.213. Detailed Description	840
23.214rc/mlpack/methods/radical/radical.hpp File Reference	841
23.214. Detailed Description	841
23.215rc/mlpack/methods/range_search/range_search.hpp File Reference	841
23.215. Detailed Description	842

23.216rc/mlpack/methods/range_search/range_search_rules.hpp File Reference	842
23.216. Detailed Description	843
23.217rc/mlpack/methods/range_search/range_search_stat.hpp File Reference	843
23.217. Detailed Description	844
23.218rc/mlpack/methods/rann/ra_query_stat.hpp File Reference	844
23.218. Detailed Description	845
23.219rc/mlpack/methods/rann/ra_search.hpp File Reference	845
23.219. Detailed Description	846
23.220rc/mlpack/methods/rann/ra_search_rules.hpp File Reference	846
23.220. Detailed Description	847
23.221rc/mlpack/methods/rann/ra_typedef.hpp File Reference	847
23.221. Detailed Description	848
23.222rc/mlpack/methods/regularized_svd/regularized_svd.hpp File Reference	849
23.222. Detailed Description	849
23.223rc/mlpack/methods/regularized_svd/regularized_svd_function.hpp File Reference	849
23.223. Detailed Description	850
23.224rc/mlpack/methods/sparse_autoencoder/sparse_autoencoder.hpp File Reference	850
23.224. Detailed Description	851
23.225rc/mlpack/methods/sparse_autoencoder/sparse_autoencoder_function.hpp File Reference	851
23.225. Detailed Description	852
23.226rc/mlpack/methods/sparse_coding/data_dependent_random_initializer.hpp File Reference	852
23.226. Detailed Description	853
23.227rc/mlpack/methods/sparse_coding/nothing_initializer.hpp File Reference	853
23.227. Detailed Description	854
23.228rc/mlpack/methods/sparse_coding/random_initializer.hpp File Reference	854
23.228. Detailed Description	855
23.229rc/mlpack/methods/sparse_coding/sparse_coding.hpp File Reference	855
23.229. Detailed Description	856
23.230rc/mlpack/prereqs.hpp File Reference	856
23.230. Detailed Description	856
23.230. Macro Definition Documentation	856
23.230.2.1 USE_MATH_DEFINES	857
23.230.2.2 force_inline	857
23.230.2.3 M_PI	857
23.231rc/mlpack/tests/data/data_3d_ind.txt File Reference	857
23.232rc/mlpack/tests/data/data_3d_mixed.txt File Reference	857
23.233rc/mlpack/tests/data/iris.txt File Reference	857

23.234rc/mlpack/tests/data/iris_labels.txt File Reference	857
23.235rc/mlpack/tests/old_boost_test_definitions.hpp File Reference	857
23.235.1 Detailed Description	858
23.235.2 Macro Definition Documentation	858
23.235.2.1 BOOST_REQUIRE_GE	858
23.235.2.2 BOOST_REQUIRE_GT	858
23.235.2.3 BOOST_REQUIRE_LE	858
23.235.2.4 BOOST_REQUIRE_LT	858
23.235.2.5 BOOST_REQUIRE_NE	858

Index	859
--------------	------------

Chapter 1

MLPACK Documentation

1.1 Introduction

MLPACK is an intuitive, fast, scalable C++ machine learning library, meant to be a machine learning analog to LAPACK. It aims to implement a wide array of machine learning methods and function as a "swiss army knife" for machine learning researchers. The MLPACK development website can be found at <http://mlpack.org>.

MLPACK uses the Armadillo C++ matrix library (<http://arma.sourceforge.net>) for general matrix, vector, and linear algebra support. MLPACK also uses the `program_options`, `math_c99`, and `unit_test_framework` components of the Boost library; in addition, LibXml2 is used.

1.2 How To Use This Documentation

This documentation is API documentation similar to Javadoc. It isn't necessarily a tutorial, but it does provide detailed documentation on every namespace, method, and class.

Each MLPACK namespace generally refers to one machine learning method, so browsing the list of namespaces provides some insight as to the breadth of the methods contained in the library.

To generate this documentation in your own local copy of MLPACK, you can simply use Doxygen, from the root directory (`/mlpack/trunk/`):

```
$ doxygen
```

1.3 Executables

MLPACK provides several executables so that MLPACK methods can be used without any need for knowledge of C++. These executables are all self-documented, and that documentation can be accessed by running the executables with the `-h` or `--help` flag.

A full list of executables is given below:

`allkfn`, `allknn`, `emst`, `gmm`, `hmm_train`, `hmm_loglik`, `hmm_viterbi`, `hmm_generate`, `kernel_pca`, `kmeans`, `lars`, `linear_↔`
`regression`, `local_coordinate_coding`, `mvu`, `nbc`, `nca`, `pca`, `radical`, `sparse_coding`

1.4 Tutorials

A few short tutorials on how to use MLPACK are given below.

- **Building MLPACK** (p. 6)
- **Matrices in MLPACK** (p. 13)
- **MLPACK Input and Output** (p. 9)
- **MLPACK Timers** (p. 17)
- **Simple Sample MLPACK Programs** (p. 15)
- **mlpack version information** (p. 19)

Tutorials on specific methods are also available.

- **NeighborSearch tutorial (k-nearest-neighbors)** (p. 59)
- **Linear/ridge regression tutorial (linear_regression)** (p. 53)
- **RangeSearch tutorial (range_search)** (p. 65)
- **Density Estimation Tree (DET) tutorial** (p. 25)
- **EMST Tutorial** (p. 31)
- **K-Means tutorial (kmeans)** (p. 43)
- **Fast max-kernel search tutorial (fastmks)** (p. 35)

1.5 Methods in MLPACK

The following methods are included in MLPACK:

- Decision Stump - **mlpack::decision_stump::DecisionStump** (p. 196)
- Density Estimation Trees - **mlpack::det::DTree** (p. 202)
- Euclidean Minimum Spanning Trees - **mlpack::emst::DualTreeBoruvka** (p. 236)
- Gaussian Mixture Models (GMMs) - **mlpack::gmm::GMM** (p. 271)
- Hidden Markov Models (HMMs) - **mlpack::hmm::HMM** (p. 284)
- Kernel PCA - **mlpack::kpca::KernelPCA** (p. 345)
- K-Means Clustering - **mlpack::kmeans::KMeans** (p. 333)
- Least-Angle Regression (LARS/LASSO) - **mlpack::regression::LARS** (p. 525)
- Local Coordinate Coding - **mlpack::lcc::LocalCoordinateCoding** (p. 352)
- Locality-Sensitive Hashing - **mlpack::neighbor::LSHSearch** (p. 388)
- Naive Bayes Classifier - **mlpack::naive_bayes::NaiveBayesClassifier** (p. 373)
- Neighborhood Components Analysis (NCA) - **mlpack::nca::NCA** (p. 376)

- Nonnegative Matrix Factorization (NMF) - `mlpack::amf::AMF<>`
- Nystroem Method - `mlpack::kernel::NystroemMethod` (p. 317)
- Perceptron - `mlpack::perceptron::Perceptron` (p. 501)
- Principal Components Analysis (PCA) - `mlpack::pca::PCA` (p. 498)
- QUIC-SVD - `mlpack::svd::QUIC_SVD` (p. 553)
- RADICAL (ICA) - `mlpack::radical::Radical` (p. 506)
- Regularized SVD - `mlpack::svd::RegularizedSVD` (p. 555)
- Simple Least-Squares Linear Regression - `mlpack::regression::LinearRegression` (p. 532)
- Sparse Autoencoding - `mlpack::nn::SparseAutoencoder` (p. 430)
- Sparse Coding - `mlpack::sparse_coding::SparseCoding` (p. 547)
- Tree-based neighbor search (AllkNN, AllkFN) - `mlpack::neighbor::NeighborSearch` (p. 399)
- Tree-based range search - `mlpack::range::RangeSearch` (p. 512)

1.6 Final Remarks

MLPACK contributors include:

- Ryan Curtin `gth671b@mail.gatech.edu`
- James Cline `james.cline@gatech.edu`
- Neil Slagle `nslagle3@gatech.edu`
- Matthew Amidon `mamidon@gatech.edu`
- Vlad Grantcharov `vlad321@gatech.edu`
- Ajinkya Kale `kaleajinkya@gmail.com`
- Bill March `march@gatech.edu`
- Dongryeol Lee `dongryel@cc.gatech.edu`
- Nishant Mehta `niche@cc.gatech.edu`
- Parikshit Ram `p.ram@gatech.edu`
- Rajendran Mohan `rmohan88@gatech.edu`
- Trironk Kiatkungwanglai `trironk@gmail.com`
- Patrick Mason `patrick.s.mason@gmail.com`
- Chip Mappus `cmappus@gatech.edu`
- Hua Ouyang `houyang@gatech.edu`
- Long Quoc Tran `tqlong@gmail.com`
- Noah Kauffman `notoriousnoah@gmail.com`

- Guillermo Colon gcolon7@mail.gatech.edu
- Wei Guan wguan@cc.gatech.edu
- Ryan Riegel rriegel@cc.gatech.edu
- Nikolaos Vasiloglou nvasil@ieee.org
- Garry Boyer garryb@gmail.com
- Andreas Löf andreas.lof@cs.waikato.ac.nz
- Marcus Edel marcus.edel@fu-berlin.de
- Mudit Raj Gupta mudit.raaj.gupta@gmail.com
- Sumedh Ghaisas sumedhghaisas@gmail.com
- Michael Fox michaelfox99@gmail.com
- Ryan Birmingham birm@gatech.edu
- Siddharth Agrawal siddharth.950@gmail.com
- Saheb Motiani saheb210692@gmail.com
- Yash Vadalia yashdv@gmail.com
- Abhishek Laddha laddhaabhishek11@gmail.com
- Vahab Akbarzadeh v.akbarzadeh@gmail.com
- Andrew Wells andrewmw94@gmail.com
- Zhihao Lou lzh1984@gmail.com

Chapter 2

Building MLPACK From Source

2.1 Introduction

MLPACK uses CMake as a build system and allows several flexible build configuration options. One can consult any of numerous CMake tutorials for further documentation, but this tutorial should be enough to get MLPACK built and installed.

2.2 latest mlpack build

Download latest mlpack build from here : `mlpack-1.0.12`

2.3 Creating Build Directory

Once the MLPACK source is unpacked, you should create a build directory.

```
$ cd mlpack-1.0.12
$ mkdir build
```

The directory can have any name, not just 'build', but 'build' is sufficient enough.

2.4 Dependencies of MLPACK

MLPACK depends on the following libraries, which need to be installed on the system and have headers present:

- Armadillo $\geq 3.6.0$ (with LAPACK support)
- LibXML2 $\geq 2.6.0$
- Boost (math_c99, program_options, unit_test_framework, random, heap) ≥ 1.49

In Ubuntu and Debian, you can get all of these dependencies through apt:

```
# apt-get install libboost-math-dev libboost-program-options-dev
libboost-random-dev libboost-test-dev libxml2-dev libarmadillo-dev
```

If you are using an Ubuntu version older than 13.10 ("Saucy Salamander") or Debian older than Jessie, you will have to compile Armadillo from source. See the README.txt distributed with Armadillo for more information.

On Fedora, Red Hat, or CentOS, these same dependencies can be obtained via yum:

```
# yum install boost-devel boost-random boost-test boost-program-options
boost-math libxml2-devel armadillo-devel
```

On Red Hat Enterprise Linux 5 and older (as well as CentOS 5), the Armadillo version available is too old and must be compiled by hand. The same applies for Fedora 16 and older.

2.5 Configuring CMake

Running CMake is the equivalent to running `./configure` with autotools. If you run CMake with no options, it will configure the project to build with debugging symbols and profiling information:

```
$ cd build
$ cmake ../
```

You can specify options to compile without debugging information and profiling information (i.e. as fast as possible):

```
$ cd build
$ cmake -D DEBUG=OFF -D PROFILE=OFF ../
```

The full list of options MLPACK allows:

- `DEBUG=(ON/OFF)`: compile with debugging symbols (default ON)
- `PROFILE=(ON/OFF)`: compile with profiling symbols (default ON)
- `ARMA_EXTRA_DEBUG=(ON/OFF)`: compile with extra Armadillo debugging symbols (default OFF)

Each option can be specified to CMake with the '-D' flag. Other tools can also be used to configure CMake, but those are not documented here.

2.6 Building MLPACK

Once CMake is configured, building the library is as simple as typing 'make'. This will build all library components as well as 'mlpack_test'.

```
$ make
Scanning dependencies of target mlpack
[ 1%] Building CXX object
src/mlpack/CMakeFiles/mlpack.dir/core/optimizers/aug_lagrangian/aug_lagrangian_test_functions.cpp.o
<...>
```

You can specify individual components which you want to build, if you do not want to build everything in the library:

```
$ make pca allknn allkfn
```

If the build fails and you cannot figure out why, register an account on Trac and submit a ticket and the MLPACK developers will quickly help you figure it out:

<http://mlpack.org/>

Alternately, MLPACK help can be found in IRC at #mlpack on irc.freenode.net.

2.7 Installing MLPACK

If you wish to install MLPACK to `/usr/include/mlpack/` and `/usr/lib/` and `/usr/bin/`, once it has built, make sure you have root privileges (or write permissions to those two directories), and simply type

```
# make install
```

You can now run the executables by name; you can link against MLPACK with `-lmlpack`, and the MLPACK headers are found in `/usr/include/mlpack/`.

Chapter 3

MLPACK Input and Output

3.1 Introduction

MLPACK provides the following:

- **mlpack::Log** (p. 357), for debugging / informational / warning / fatal output
- **mlpack::CLI** (p. 184), for parsing command line options

Each of those classes are well-documented, and that documentation should be consulted for further reference.

3.2 Simple Logging Example

MLPACK has four logging levels:

- Log::Debug
- Log::Info
- Log::Warn
- Log::Fatal

Output to Log::Debug does not show (and has no performance penalty) when MLPACK is compiled without debugging symbols. Output to Log::Info is only shown when the program is run with the `--verbose` (or `-v`) flag. Log::Warn is always shown, and Log::Fatal will halt the program, when a newline is sent to it.

Here is a simple example, and its output:

```
#include <mlpack/core.hpp>

using namespace mlpack;

int main(int argc, char** argv)
{
    CLI::ParseCommandLine(argc, argv);

    Log::Debug << "Compiled with debugging symbols." << std::endl;

    Log::Info << "Some test informational output." << std::endl;
```

```

Log::Warn << "A warning!" << std::endl;

Log::Fatal << "Program has crashed." << std::endl;

Log::Warn << "Made it!" << std::endl;
}

```

With debugging output and `--verbose`, the following is shown:

```

$ ./main --verbose
[DEBUG] Compiled with debugging symbols.
[INFO ] Some test informational output.
[WARN ] A warning!
[FATAL] Program has crashed.

```

The last warning is not reached, because `Log::Fatal` terminates the program.

Without debugging symbols and without `--verbose`, the following is shown:

```

$ ./main
[WARN ] A warning!
[FATAL] Program has crashed.

```

These four outputs can be very useful for both providing informational output and debugging output for your MLPACK program.

3.3 Simple CLI Example

Through the `mlpack::CLI` (p. 184) object, command-line parameters can be easily added with the `PROGRAM_INFO`, `PARAM_INT`, `PARAM_DOUBLE`, `PARAM_STRING`, and `PARAM_FLAG` macros.

Here is a sample use of those macros, extracted from `methods/pca/pca_main.cpp`.

```

#include <mlpack/core.hpp>

// Document program.
PROGRAM_INFO("Principal Components Analysis", "This program performs principal "
    "components analysis on the given dataset. It will transform the data "
    "onto its principal components, optionally performing dimensionality "
    "reduction by ignoring the principal components with the smallest "
    "eigenvalues.");

// Parameters for program.
PARAM_STRING_REQ("input_file", "Input dataset to perform PCA on.", "");
PARAM_STRING_REQ("output_file", "Output dataset to perform PCA on.", "");
PARAM_INT("new_dimensionality", "Desired dimensionality of output dataset.",
    "", 0);

using namespace mlpack;

int main(int argc, char** argv)
{
    // Parse commandline.
    CLI::ParseCommandLine(argc, argv);

    ...
}

```

Documentation is automatically generated using those macros, and when the program is run with `--help` the following is displayed:

```

$ pca --help
Principal Components Analysis

This program performs principal components analysis on the given dataset. It
will transform the data onto its principal components, optionally performing

```

dimensionality reduction by ignoring the principal components with the smallest eigenvalues.

Required options:

<code>--input_file [string]</code>	Input dataset to perform PCA on.
<code>--output_file [string]</code>	Output dataset to perform PCA on.

Options:

<code>--help (-h)</code>	Default help info.
<code>--info [string]</code>	Get help on a specific module or option. Default value ''.
<code>--new_dimensionality [int]</code>	Desired dimensionality of output dataset. Default value 0.
<code>--verbose (-v)</code>	Display informational messages and the full list of parameters and timers at the end of execution.

The **mlpack::CLI** (p. 184) documentation can be consulted for further and complete documentation.

Chapter 4

Matrices in MLPACK

4.1 Introduction

MLPACK uses Armadillo matrices for matrix support. Armadillo is a fast C++ matrix library which makes use of advanced template techniques to provide the fastest possible matrix operations.

Documentation on Armadillo can be found on their website:

`http://arma.sourceforge.net/docs.html`

Nonetheless, there are a few further caveats for MLPACK Armadillo usage.

4.2 Column-wise Matrices

Armadillo matrices are stored in a column-major format; this means that on disk, each column is located in contiguous memory.

This means that, for the vast majority of machine learning methods, it is faster to store observations as columns and dimensions as rows. This is counter to most standard machine learning texts!

Major implications of this are for linear algebra. For instance, the covariance of a matrix is typically

$$C = X^T X$$

but for a column-wise matrix, it is

$$C = X X^T$$

and this is very important to keep in mind! If your MLPACK code is not working, this may be a factor in why.

4.3 Loading Matrices

MLPACK provides a **data::Load()** (p. 93) and **data::Save()** (p. 94) function, which should be used instead of Armadillo's loading and saving functions.

Most machine learning data is stored in row-major format; a CSV, for example, will generally have one observation per line and each column will correspond to a dimension.

The **data::Load()** (p. 93) and **data::Save()** (p. 94) functions transpose the matrix upon loading, meaning that the following CSV:

```
$ cat data.csv
3,3,3,3,0
3,4,4,3,0
3,4,4,3,0
3,3,4,3,0
3,6,4,3,0
2,4,4,3,0
2,4,4,1,0
3,3,3,2,0
3,4,4,2,0
3,4,4,2,0
3,3,4,2,0
3,6,4,2,0
2,4,4,2,0
```

is actually loaded with 5 rows and 13 columns, not 13 rows and 5 columns like the CSV is written.

This is important to remember!

Chapter 5

Simple Sample MLPACK Programs

5.1 Introduction

On this page, several simple MLPACK examples are contained, in increasing order of complexity.

5.2 Covariance Computation

A simple program to compute the covariance of a data matrix ("data.csv"), assuming that the data is already centered, and save it to file.

```
// Includes all relevant components of MLPACK.
#include <mlpack/core.hpp>

// Convenience.
using namespace mlpack;

int main()
{
    // First, load the data.
    arma::mat data;
    // Use data::Load() which transposes the matrix.
    data::Load("data.csv", data, true);

    // Now compute the covariance. We assume that the data is already centered.
    // Remember, because the matrix is column-major, the covariance operation is
    // transposed.
    arma::mat cov = data * trans(data) / data.n_cols;

    // Save the output.
    data::Save("cov.csv", cov, true);
}
```

5.3 Nearest Neighbor

This simple program uses the `mlpack::neighbor::NeighborSearch` (p. 399) object to find the nearest neighbor of each point in a dataset using the L1 metric, and then print the index of the neighbor and the distance of it to stdout.

```
#include <mlpack/core.hpp>
#include <mlpack/methods/neighbor_search/neighbor_search.hpp>

using namespace mlpack;
using namespace mlpack::neighbor; // NeighborSearch and NearestNeighborSort
using namespace mlpack::metric; // ManhattanDistance
```

```
int main()
{
    // Load the data from data.csv (hard-coded). Use CLI for simple command-line
    // parameter handling.
    arma::mat data;
    data::Load("data.csv", data, true);

    // Use templates to specify that we want a NeighborSearch object which uses
    // the Manhattan distance.
    NeighborSearch<NearestNeighborSort, ManhattanDistance> nn(data);

    // Create the object we will store the nearest neighbors in.
    arma::Col<size_t> neighbors;
    arma::vec distances; // We need to store the distance too.

    // Compute the neighbors.
    nn.Search(1, neighbors, distances);

    // Write each neighbor and distance using Log.
    for (size_t i = 0; i < neighbors.n_elem; ++i)
    {
        Log::Info << "Nearest neighbor of point " << i << " is point "
            << neighbors[i] << " and the distance is " << distances[i] << ".\n";
    }
}
```

5.4 Other examples

For more complex examples, it is useful to refer to the main executables:

- methods/neighbor_search/allknn_main.cpp
- methods/neighbor_search/allkfn_main.cpp
- methods/emst/emst_main.cpp
- methods/radical/radical_main.cpp
- methods/nca/nca_main.cpp
- methods/naive_bayes/nbc_main.cpp
- methods/pca/pca_main.cpp
- methods/lars/lars_main.cpp
- methods/linear_regression/linear_regression_main.cpp
- methods/gmm/gmm_main.cpp
- methods/kmeans/kmeans_main.cpp

Chapter 6

MLPACK Timers

6.1 Introduction

MLPACK provides a simple timer interface for the timing of machine learning methods. The results of any timers used during the program are displayed at output by the **mlpack::CLI** (p. 184) object, when `-verbose` is given:

```
$ allknn -r dataset.csv -n neighbors_out.csv -d distances_out.csv -k 5 -v
<...>
[INFO ] Program timers:
[INFO ]   computing_neighbors: 0.010650s
[INFO ]   loading_data: 0.002567s
[INFO ]   saving_data: 0.001115s
[INFO ]   total_time: 0.149816s
[INFO ]   tree_building: 0.000534s
```

6.2 Timer API

The **mlpack::Timer** (p. 562) class provides three simple methods:

```
void Timer::Start(const char* name);
void Timer::Stop(const char* name);
timeval Timer::Get(const char* name);
```

Each timer is given a name, and is referenced by that name. You can call `Timer::Start()` and `Timer::Stop()` multiple times for a particular timer name, and the result will be the sum of the runs of the timer. Note that `Timer::Stop()` must be called before `Timer::Start()` is called again.

A "total_time" timer is run by default for each MLPACK program.

6.3 Timer Example

Below is a very simple example of timer usage in code.

```
#include <mlpack/core.hpp>

using namespace mlpack;

int main(int argc, char** argv)
{
    CLI::ParseCommandLine(argc, argv);
```

```
// Start a timer.  
Timer::Start("some_timer");  
  
// Do some things.  
DoSomeStuff();  
  
// Stop the timer.  
Timer::Stop("some_timer");  
}
```

If the `-verbose` flag was given to this executable, the resultant time that "some_timer" ran for would be shown.

Chapter 7

mlpack version information

7.1 mlpack versions in code

mlpack provides a couple of convenience macros and functions to get the version of mlpack. More information (and straightforward code) can be found in **src/mlpack/core/util/version.hpp** (p. 775).

The following three macros provide major, minor, and patch versions of mlpack (i.e. for mlpack-x.y.z, 'x' is the major version, 'y' is the minor version, and 'z' is the patch version):

```
__MLPACK_VERSION_MAJOR  
__MLPACK_VERSION_MINOR  
__MLPACK_VERSION_PATCH
```

In addition, the function **mlpack::util::GetVersion()** (p. 118) returns the mlpack version as a string (for instance, "mlpack 1.0.12").

7.2 mlpack executable versions

Each mlpack executable supports the `-version` (or `-V`) option, which will print the version of mlpack used. If the version is not an official release but instead from svn trunk, the version will be "mlpack trunk" (and may have a revision number appended to "trunk").

Chapter 8

Alternating Matrix Factorization tutorial.

8.1 Introduction

Alternating Matrix Factorization

Alternating matrix factorization decomposes matrix V in the form $V \approx WH$ where W is called the basis matrix and H is called the encoding matrix. V is taken to be of size $n \times m$ and the obtained W is $n \times r$ and H is $r \times m$. The size r is called the rank of the factorization. Factorization is done by alternately calculating W and H respectively while holding the other matrix constant.

mlpack provides:

- a **simple C++ interface** (p. 21) to perform Alternating Matrix Factorization

8.2 Table of Contents

A list of all the sections this tutorial contains.

- **Introduction** (p. 21)
- **Table of Contents** (p. 21)
- **The 'AMF' class** (p. 21)
 - **Using different termination policies** (p. 22)
 - **Using different initialization policies** (p. 22)
 - **Using different update rules** (p. 23)
 - **Using Non-Negative Matrix Factorization with AMF** (p. 23)
 - **Using Singular Value Decomposition with AMF** (p. 24)
- **Further documentation** (p. 24)

8.3 The 'AMF' class

The AMF class is templated with 3 parameters; the first contains the policy used to determine when the algorithm has converged; the second contains the initialization rule for the W and H matrix; the last contains the update rule to be used

during each iteration. This templization allows the user to try various update rules, initialization rules, and termination policies (including ones not supplied with MLPACK) for factorization.

The class provides the following method that performs factorization

```
template<typename MatType> double Apply(const MatType& V,
                                       const size_t r,
                                       arma::mat& W,
                                       arma::mat& H);
```

8.3.1 Using different termination policies

The AMF implementation comes with different termination policies to support many implemented algorithms. Every termination policy implements the following method which returns the status of convergence.

```
bool IsConverged(arma::mat& W, arma::mat& H)
```

list of all the termination policies

- **mlpack::amf::SimpleResidueTermination** (p. 139)
- **mlpack::amf::SimpleToleranceTermination** (p. 143)
- **mlpack::amf::ValidationRMSETermination** (p. 159)

In **SimpleResidueTermination**, termination decision depends on two factors, value of residue and number of iteration. If the current value of residue drops below the threshold or the number of iterations goes beyond the threshold, positive termination signal is passed to AMF.

In **SimpleToleranceTermination**, termination criterion is met when increase in residue value drops below the given tolerance. To accommodate spikes, certain number of successive residue drops are accepted. Secondary termination criterion terminates algorithm when iteration count goes beyond the threshold.

ValidationRMSETermination divides the data into 2 sets, training set and validation set. Entries of validation set are nullified in the input matrix. Termination criterion is met when increase in validation set RMSE value drops below the given tolerance. To accommodate spikes certain number of successive validation RMSE drops are accepted. This upper limit on successive drops can be adjusted with **reverseStepCount**. Secondary termination criterion terminates algorithm when iteration count goes above the threshold. Though this termination policy is better measure of convergence than the above 2 termination policies, it may cause a overhead in performance.

On the other hand **CompleteIncrementalTermination** (p.128) and **mlpack::amf::IncompleteIncrementalTermination** (p. 130) are just wrapper classes for other termination policies. These policies are used when AMF is applied with **SVDCompleteIncrementalLearning** (p.151) and **SVDIncompleteIncrementalLearning** (p.157) respectively.

8.3.2 Using different initialization policies

The AMF class comes with 2 initialization policies

- **RandomInitialization** (p. 138)
- **RandomAcolInitialization** (p. 137)

RandomInitialization initializes matrices W and H with random uniform distribution while **RandomAcolInitialization** initializes the W matrix by averaging p randomly chosen columns of V. In case of **RandomAcolInitialization**, p is a template parameter.

To implement their own initialization policy, users need to define the following function in their class.

```
template<typename MatType>
inline static void Initialize(const MatType& V,
                             const size_t r,
                             arma::mat& W,
                             arma::mat& H)
```

8.3.3 Using different update rules

AMF supports following update rules

- **AMFALSUpdate** (p. 132)
- **NMFMultiplicativeDistanceUpdate** (p. 134)
- **NMFMultiplicativeDivergenceUpdate** (p. 136)
- **SVDBatchLearning** (p. 148)
- **SVDIncompleteIncrementalLearning** (p. 157)
- **SVDCompleteIncrementalLearning** (p. 151)

Non-Negative Matrix factorization can be achieved with NMFALSUpdate, NMFMultiplicativeDivergenceUpdate or NMF↔FMultiplicativeDivergenceUpdate. NMFALSUpdate implements simple Alternating Least Square optimization while the other rules implement algorithms given in paper 'Algorithms for Non-negative Matrix Factorization'.

The remaining update rules perform Singular Value Decomposition of matrix V. This SVD factorization is optimized for the use by Collaborative Filtering. This use of SVD factorizers for Collaborative Filtering is described in the paper 'A Guide to singular Value Decomposition' by Chih-Chao Ma. For further details about the algorithms refer to the respective class documentation.

8.3.4 Using Non-Negative Matrix Factorization with AMF

The use of AMF for Non-Negative Matrix factorization is simple. The AMF module defines NMFALSFactorizer which can be used directly without knowing the internal structure of AMF. For example -

```
#include <iostream>
#include <mlpack/core.hpp>
#include <mlpack/methods/amf/amf.hpp>

using namespace std;
using namespace arma;
using namespace mlpack::amf;

int main()
{
    NMFALSFactorizer nmf;
    mat W, H;
    mat V = randu<mat>(100, 100);
    double residue = Apply(V, W, H);
    return 1;
}
```

NMFALSFactorizer uses SimpleResidueTermination which is most preferred with Non-Negative Matrix factorizers. Initialization of W and H in NMFALSFactorizer is random. The Apply function returns the residue obtained by comparing the constructed matrix $W * H$ with the original matrix V.

8.3.5 Using Singular Value Decomposition with AMF

AMF implementation supports following SVD factorizers

- SVDBatchFactorizer
- SparseSVDBatchFactorizer
- SVDIncompleteIncrementalFactorizer
- SparseSVDIncompleteIncrementalFactorizer
- SVDCompleteIncrementalFactorizer
- SparseSVDCompleteIncrementalFactorizer

The sparse version of factorizers can be used with Armadillo's sparse matrix support. These specialized implementations boost runtime performance when the matrix to be factorized is relatively sparse.

```
#include <mlpack/core.hpp>
#include <mlpack/methods/amf/amf.hpp>

using namespace std;
using namespace arma;
using namespace mlpack::amf;

int main()
{
  sp_mat V = randu<sp_mat>(100,100);
  mat W, H;

  SparseSVDBatchFactorizer svd;
  double residue = svd.Apply(V, W, H);
}
```

8.4 Further documentation

For further documentation on the AMF class, consult the **complete API documentation** (p. 123).

Chapter 9

Density Estimation Tree (DET) tutorial

9.1 Introduction

DETs perform the unsupervised task of density estimation using decision trees. Using a trained density estimation tree (DET), the density at any particular point can be estimated very quickly ($O(\log n)$ time, where n is the number of points the tree is built on).

The details of this work is presented in the following paper:

```
@inproceedings{ram2011density,
  title={Density estimation trees},
  author={Ram, P. and Gray, A.G.},
  booktitle={Proceedings of the 17th ACM SIGKDD International Conference on
    Knowledge Discovery and Data Mining},
  pages={627--635},
  year={2011},
  organization={ACM}
}
```

mlpack provides:

- a **simple command-line executable** (p. 26) to perform density estimation and related analyses using DETs
- a **generic C++ class (DTree)** (p. 28) which provides various functionality for the DETs
- a set of functions in the namespace **mlpack::det** (p. 29) to perform cross-validation for the task of density estimation with DETs

9.2 Table of Contents

A list of all the sections this tutorial contains.

- **Introduction** (p. 25)
- **Table of Contents** (p. 25)
- **Command-Line 'det'** (p. 26)
 - **Plain-vanilla density estimation** (p. 27)
 - **Estimation on a test set** (p. 27)

- **Printing a trained DET** (p.27)
- **Computing the variable importance** (p.28)
- **Leaf Membership** (p.28)
- **The 'DTree' class** (p.28)
 - **Public Functions** (p.28)
- **'namespace mlpack::det'** (p.29)
 - **Utility Functions** (p.29)
- **Further Documentation** (p.30)

9.3 Command-Line 'det'

The command line arguments of this program can be viewed using the '-h' option:

```
$ det -h
Density Estimation With Density Estimation Trees

This program performs a number of functions related to Density Estimation
Trees. The optimal Density Estimation Tree (DET) can be trained on a set of
data (specified by --train_file) using cross-validation (with number of folds
specified by --folds). In addition, the density of a set of test points
(specified by --test_file) can be estimated, and the importance of each
dimension can be computed. If class labels are given for the training points
(with --labels_file), the class memberships of each leaf in the DET can be
calculated.

The created DET can be saved to a file, along with the density estimates for
the test set and the variable importances.

Required options:

--train_file (-t) [string]    The data set on which to build a density
                              estimation tree.

Options:

--folds (-f) [int]           The number of folds of cross-validation to
                              perform for the estimation (0 is LOOCV) Default
                              value 10.
--help (-h)                  Default help info.
--info [string]              Get help on a specific module or option.
                              Default value ''.
--labels_file (-l) [string]  The labels for the given training data to
                              generate the class membership of each leaf (as
                              an extra statistic) Default value ''.
--leaf_class_table_file (-L) [string]
                              The file in which to output the leaf class
                              membership table. Default value
                              'leaf_class_membership.txt'.
--max_leaf_size (-M) [int]   The maximum size of a leaf in the unpruned,
                              fully grown DET. Default value 10.
--min_leaf_size (-N) [int]   The minimum size of a leaf in the unpruned,
                              fully grown DET. Default value 5.
--print_tree (-p)            Print the tree out on the command line (or in
                              the file specified with --tree_file).
--print_vi (-I)              Print the variable importance of each feature
                              out on the command line (or in the file
                              specified with --vi_file).
--test_file (-T) [string]    A set of test points to estimate the density of.
                              Default value ''.
--test_set_estimates_file (-E) [string]
                              The file in which to output the estimates on the
                              test set from the final optimally pruned tree.
                              Default value ''.
--training_set_estimates_file (-e) [string]
                              The file in which to output the density
                              estimates on the training set from the final
```

```

--tree_file (-r) [string]    optimally pruned tree. Default value ''.
                             The file in which to print the final optimally
                             pruned tree. Default value ''.
--unpruned_tree_estimates_file (-u) [string]
                             The file in which to output the density
                             estimates on the training set from the large
                             unpruned tree. Default value ''.
--verbose (-v)               Display informational messages and the full list
                             of parameters and timers at the end of
                             execution.
--vi_file (-i) [string]      The file to output the variable importance
                             values for each feature. Default value ''.

```

For further information, including relevant papers, citations, and theory, consult the documentation found at <http://www.mlpack.org> or included with your distribution of MLPACK.

9.3.1 Plain-vanilla density estimation

We can just train a DET on the provided data set *S*. Like all datasets **mlpack** uses, the data should be row-major (**mlpack** transposes data when it is loaded; internally, the data is column-major – see [this page](#) (p. 13) for more information).

```
$ det -t dataset.csv -v
```

By default, **det** performs 10-fold cross-validation (using the α -pruning regularization for decision trees). To perform LO↵OCV (leave-one-out cross-validation), which can provide better results but will take longer, use the following command:

```
$ det -t dataset.csv -f 0 -v
```

To perform *k*-fold crossvalidation, use `-f k` (or `-folds k`). There are certain other options available for training. For example, in the construction of the initial tree, you can specify the maximum and minimum leaf sizes. By default, they are 10 and 5 respectively; you can set them using the `-M` (`-max_leaf_size`) and the `-N` (`-min_leaf_size`) options.

```
$ det -t dataset.csv -M 20 -N 10
```

In case you want to output the density estimates at the points in the training set, use the `-e` (`-training_set_↵estimates_file`) option to specify the output file to which the estimates will be saved. The first line in `density_↵_estimates.txt` will correspond to the density at the first point in the training set. Note that the logarithm of the density estimates are given, which allows smaller estimates to be saved.

```
$ det -t dataset.csv -e density_estimates.txt -v
```

9.3.2 Estimation on a test set

Often, it is useful to train a density estimation tree on a training set and then obtain density estimates from the learned estimator for a separate set of test points. The `-T` (`-test_file`) option allows specification of a set of test points, and the `-E` (`-test_set_estimates_file`) option allows specification of the file into which the test set estimates are saved. Note that the logarithm of the density estimates are saved; this allows smaller values to be saved.

```
$ det -t dataset.csv -T test_points.csv -E test_density_estimates.txt -v
```

9.3.3 Printing a trained DET

A depth-first visualization of the DET can be obtained by using the `-p` (`-print_tree`) flag.

```
$ det -t dataset.csv -p -v
```

To print this tree in a file, use the `-r` (`-tree_file`) option to specify the output file along with the `-P` (`-print_tree`) flag.

```
$ det -t dataset.csv -p -r tree.txt -v
```

9.3.4 Computing the variable importance

The variable importance (with respect to density estimation) of the different features in the data set can be obtained by using the `-I` (`-print_vi`) option. This outputs the absolute (as opposed to relative) variable importance of the all the features.

```
$ det -t dataset.csv -I -v
```

To print this in a file, use the `-i` (`-vi_file`) option.

```
$ det -t dataset.csv -I -i variable_importance.txt -v
```

9.3.5 Leaf Membership

In case the dataset is labeled and you want to find the class membership of the leaves of the tree, there is an option to print the class membership into a file. The training data has to still be input in an unlabeled format, but an additional label file containing the corresponding labels of each point has to be input using the `-l` (`-labels_file`) option. The file to output the class memberships into can be specified with `-L` (`-leaf_class_table_file`). If `-L` is left unspecified, `leaf_class_membership.txt` is used by default.

```
$ det -t dataset.csv -l labels.csv -v
$ det -t dataset.csv -l labels.csv -L leaf_class_membership_file.txt -v
```

9.4 The 'DTree' class

This class implements density estimation trees. Below is a simple example which initializes a density estimation tree.

```
#include <mlpack/methods/det/dtree.hpp>

using namespace mlpack::det;

// The dataset matrix, on which to learn the density estimation tree.
extern arma::Mat<float> data;

// Initialize the tree. This function also creates and saves the bounding box
// of the data. Note that it does not actually build the tree.
DTree<> det(data);
```

9.4.1 Public Functions

The function `Grow()` greedily grows the tree, adding new points to the tree. Note that the points in the dataset will be reordered. This should only be run on a tree which has not already been built. In general, it is more useful to use the `Trainer()` (p. 96) function found in `'namespace mlpack::det'` (p. 29).


```
// This keeps track of the data during the shuffle that occurs while growing the
// tree.
arma::Col<size_t> oldFromNew(data.n_cols);
for (size_t i = 0; i < data.n_cols; i++)
  oldFromNew[i] = i;

// This function grows the tree down to the leaves. It returns the current
// minimum value of the regularization parameter alpha.
size_t maxLeafSize = 10;
size_t minLeafSize = 5;

double alpha = det.Grow(data, oldFromNew, false, maxLeafSize, minLeafSize);
```

Note that the alternate volume regularization should not be used (see ticket #238).

To estimate the density at a given query point, use the following code. Note that the logarithm of the density is returned.

```
// For a given query, you can obtain the density estimate.
extern arma::Col<float> query;
extern DTree* det;
double estimate = det->ComputeValue(&query);
```

Computing the **variable importance** of each feature for the given DET.

```
// The data matrix and density estimation tree.
extern arma::mat data;
extern DTree* det;

// The variable importances will be saved into this vector.
arma::Col<double> varImps;

// You can obtain the variable importance from the current tree.
det->ComputeVariableImportance(varImps);
```

9.5 'namespace mlpack::det'

The functions in this namespace allows the user to perform tasks with the 'DTree' class. Most importantly, the **Trainer()** (p. 96) method allows the full training of a density estimation tree with cross-validation. There are also utility functions which allow printing of leaf membership and variable importance.

9.5.1 Utility Functions

The code below details how to train a density estimation tree with cross-validation.

```
#include <mlpack/methods/det/dt_utils.hpp>

using namespace mlpack::det;

// The dataset matrix, on which to learn the density estimation tree.
extern arma::Mat<float> data;

// The number of folds for cross-validation.
const size_t folds = 10; // Set folds = 0 for LOOCV.

const size_t maxLeafSize = 10;
const size_t minLeafSize = 5;

// Train the density estimation tree with cross-validation.
DTree<>* dtree_opt = Trainer(data, folds, false, maxLeafSize, minLeafSize);
```

Note that the alternate volume regularization should be set to false because it has known bugs (see #238).

To print the class membership of leaves in the tree into a file, see the following code.

```
extern arma::Mat<size_t> labels;
extern DTree* det;
const size_t numClasses = 3; // The number of classes must be known.

extern string leafClassMembershipFile;

PrintLeafMembership(det, data, labels, numClasses, leafClassMembershipFile);
```

Note that you can find the number of classes with `max(labels) + 1`. The variable importance can also be printed to a file in a similar manner.

```
extern DTree* det;

extern string variableImportanceFile;
const size_t numFeatures = data.n_rows;

PrintVariableImportance(det, numFeatures, variableImportanceFile);
```

9.6 Further Documentation

For further documentation on the DTree class, consult the **complete API documentation** (p. 202).

Chapter 10

EMST Tutorial

10.1 Introduction

The Euclidean Minimum Spanning Tree problem is widely used in machine learning and data mining applications. Given a set S of points in \mathbf{R}^d , our task is to compute lowest weight spanning tree in the complete graph on S with edge weights given by the Euclidean distance between points.

Among other applications, the EMST can be used to compute hierarchical clusterings of data. A *single-linkage clustering* can be obtained from the EMST by deleting all edges longer than a given cluster length. This technique is also referred to as a *Friends-of-Friends* clustering in the astronomy literature.

MLPACK includes an implementation of **Dual-Tree Boruvka** which uses *kd*-trees by default; this is the empirically and theoretically fastest EMST algorithm. In addition, the implementation supports the use of different trees via templates. For more details, see the following paper:

```
@inproceedings{march2010fast,
  title={Fast {E}uclidean minimum spanning tree: algorithm, analysis, and applications},
  author={March, William B. and Ram, Parikshit and Gray, Alexander G.},
  booktitle={Proceedings of the 16th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (KDD '10)},
  pages={603--612},
  year={2010},
  organization={ACM}
}
```

mlpack provides:

- a **simple command-line executable** (p. 32) to compute the EMST of a given data set
- a **simple C++ interface** (p. 33) to compute the EMST

10.2 Table of Contents

A list of all the sections this tutorial contains.

- **Introduction** (p. 31)
- **Table of Contents** (p. 31)
- **Command-Line 'EMST'** (p. 32)

- The 'DualTreeBoruvka' class (p. 33)
- Further documentation (p. 33)

10.3 Command-Line 'EMST'

The `emst` executable in `mlpack` will compute the EMST of a given set of points and store the resulting edge list to a file.

The output file contains an edge list representation of the MST in an $n-1 \times 3$ matrix, where the first and second columns are labels of points and the third column is the edge weight. The edges are sorted in order of increasing weight.

Below are several examples of simple usage (and the resultant output). The '-v' option is used so that verbose output is given. Further documentation on each individual option can be found by typing

```
$ emst --help

$ emst --input_file=dataset.csv --output_file=edge_list.csv -v
[INFO ] Reading in data.
[INFO ] Loading 'dataset.csv' as CSV data.
[INFO ] Data read, building tree.
[INFO ] Tree built, running algorithm.
[INFO ] 4 edges found so far.
[INFO ] 5 edges found so far.
[INFO ] Total spanning tree length: 1002.45
[INFO ] Saving CSV data to 'edge_list.csv'.
[INFO ]
[INFO ] Execution parameters:
[INFO ]   help: false
[INFO ]   info: ""
[INFO ]   input_file: dataset.csv
[INFO ]   leaf_size: 1
[INFO ]   naive: false
[INFO ]   output_file: edge_list.csv
[INFO ]   verbose: true
[INFO ]
[INFO ] Program timers:
[INFO ]   emst/mst_computation: 0.000179s
[INFO ]   emst/tree_building: 0.000061s
[INFO ]   total_time: 0.052641s
```

The code performs at most $\log N$ iterations for N data points. It will print an update on the number of MST edges found after each iteration. Convenient program timers are given for different parts of the calculation at the bottom of the output, as well as the parameters the simulation was run with.

```
$ cat dataset.csv
0, 0
1, 1
3, 3
0.5, 0
1000, 0
1001, 0

$ cat edge_list.csv
0.0000000000e+00,3.0000000000e+00,5.0000000000e-01
4.0000000000e+00,5.0000000000e+00,1.0000000000e+00
1.0000000000e+00,3.0000000000e+00,1.1180339887e+00
1.0000000000e+00,2.0000000000e+00,2.8284271247e+00
2.0000000000e+00,4.0000000000e+00,9.9700451353e+02
```

The input points are labeled 0-5. The output tells us that the MST connects point 0 to point 3, point 4 to point 5, point 1 to point 3, point 1 to point 2, and point 2 to point 4, with the corresponding edge weights given in the third column. The total length of the MST is also given in the verbose output.

Note that it is also possible to compute the EMST using a naive ($O(N^2)$) algorithm for timing and comparison purposes.

10.4 The 'DualTreeBoruvka' class

The 'DualTreeBoruvka' class contains our implementation of the Dual-Tree Boruvka algorithm.

The class has two constructors: the first takes the data set, constructs the tree (where the type of tree constructed is the `TreeType` template parameter), and computes the MST. The second takes data set and an already constructed tree.

The class provides one method that performs the MST computation:

```
void ComputeMST(const arma::mat& results);
```

This method stores the computed MST in the matrix results in the format given above.

10.5 Further documentation

For further documentation on the DualTreeBoruvka class, consult the **complete API documentation** (p. 236).

Chapter 11

Fast max-kernel search tutorial (fastmks)

11.1 Introduction

The FastMKS algorithm (fast exact max-kernel search) is a recent algorithm proposed in the following paper:

```
@inproceedings{curtin2013fast,
  title={Fast Exact Max-Kernel Search},
  author={Curtin, Ryan R. and Ram, Parikshit and Gray, Alexander G.},
  booktitle={Proceedings of the 2013 SIAM International Conference on Data
    Mining (SDM '13)},
  year={2013},
  pages={1--9}
}
```

Given a set of query points Q and a set of reference points R , the FastMKS algorithm is a fast dual-tree (or single-tree) algorithm which finds

$$\arg \max_{p_r \in R} K(p_q, p_r)$$

for all points $p_q \in Q$ and for some Mercer kernel $K(\cdot, \cdot)$. A Mercer kernel is a kernel that is positive semidefinite; these are the classes of kernels that can be used with the kernel trick. In short, the positive semidefiniteness of a Mercer kernel means that any kernel matrix (or Gram matrix) created on a dataset must be positive semidefinite.

The FastMKS algorithm builds trees on the datasets Q and R in such a way that explicit representation of the points in the kernel space is unnecessary, by using cover trees (**mlpack::tree::CoverTree** (p. 603)). This allows the algorithm to be run, for instance, on string kernels, where there is no sensible explicit representation. The **mlpack** implementation allows any type of tree that does not require an explicit representation to be used. For more details, see the paper.

At the time of this writing there is no other fast algorithm for exact max-kernel search. Also, **mlpack** implements dual-tree FastMKS, while the paper referenced above only explains single-tree search.

mlpack provides:

- a **simple command-line executable** (p. 36) to run FastMKS
- a **C++ interface** (p. 38) to run FastMKS

11.2 Table of Contents

A list of all the sections this tutorial contains.

- **Introduction** (p. 35)
- **Table of Contents** (p. 35)
- **Command-line FastMKS (fastmks)** (p. 36)
 - **FastMKS with a linear kernel on one dataset** (p. 37)
 - **FastMKS on a reference and query dataset** (p. 37)
 - **FastMKS with a different kernel** (p. 37)
 - **Using single-tree search or naive search** (p. 38)
 - **Parameters for alternate kernels** (p. 38)
- **The 'FastMKS' class** (p. 38)
 - **FastMKS on one dataset** (p. 39)
 - **FastMKS with a query and reference dataset** (p. 39)
 - **FastMKS with an initialized kernel** (p. 39)
 - **FastMKS with an already-created tree** (p. 40)
- **Writing a custom kernel for FastMKS** (p. 41)
- **Using other tree types for FastMKS** (p. 41)
- **Running FastMKS on objects** (p. 41)
- **Further documentation** (p. 42)

11.3 Command-line FastMKS (fastmks)

mlpack provides a command-line program, `fastmks`, which is used to perform FastMKS on a given query and reference dataset. It supports numerous different types of kernels:

- **linear kernel** (p. 315)
- **polynomial kernel** (p. 319)
- **cosine distance** (p. 293)
- **Gaussian kernel** (p. 299)
- **Epanechnikov kernel** (p. 294)
- **triangular kernel** (p. 329)
- **hyperbolic tangent kernel** (p. 304)
- **Laplacian kernel** (p. 312)

Note that when a shift-invariant kernel is used, the results will be the same as nearest neighbor search, so **allknn** (p. 59) may be a better option. A shift-invariant kernel is a kernel that depends only on the distance between the two input points. The **Gaussian kernel** (p. 299), **Epanechnikov kernel** (p. 294), **triangular kernel** (p. 329), and **Laplacian kernel** (p. 312) are instances of shift-invariant kernels. The paper contains more details on this situation. The `fastmks` executable still provides these kernels as options, though.

The following examples detail usage of the `fastmks` program. Note that you can get documentation on all the possible parameters by typing:

```
$ fastmks --help
```


11.3.1 FastMKS with a linear kernel on one dataset

If only one dataset is specified (with `-r` or `-reference_file`), the reference dataset is taken to be both the query and reference datasets. The example below finds the 4 maximum kernels of each point in `dataset.csv`, using the default linear kernel.

```
$ fastmks -r dataset.csv -k 4 -v -p products.csv -i indices.csv
```

When the operation completes, the values of the kernels are saved in `products.csv` and the indices of the points which give the maximum kernels are saved in `indices.csv`.

```
$ head indices.csv
```

```
762,910,863,890
762,910,426,568
910,762,863,426
762,910,863,426
863,910,614,762
762,863,910,614
762,910,488,568
762,910,863,426
910,762,863,426
863,762,910,614
```

```
$ head products.csv
```

```
1.6221652894e+00,1.5998743443e+00,1.5898890769e+00,1.5406789753e+00
1.3387953449e+00,1.3317349486e+00,1.2966613184e+00,1.2774493620e+00
1.6386110476e+00,1.6332029753e+00,1.5952629124e+00,1.5887195330e+00
1.0917545803e+00,1.0820878726e+00,1.0668992636e+00,1.0419838050e+00
1.2272441028e+00,1.2169643942e+00,1.2104597963e+00,1.2067780154e+00
1.5720962456e+00,1.5618504956e+00,1.5609069923e+00,1.5235605095e+00
1.3655478674e+00,1.3548593212e+00,1.3311547298e+00,1.3250728881e+00
2.0119149744e+00,2.0043668067e+00,1.9847289214e+00,1.9298280046e+00
1.1586923205e+00,1.1494586097e+00,1.1274872962e+00,1.1248172766e+00
4.4789820372e-01,4.4618539778e-01,4.4200024852e-01,4.3989721792e-01
```

We can see in this example that for point 0, the point with maximum kernel value is point 762, with a kernel value of 1.622165. For point 3, the point with third largest kernel value is point 863, with a kernel value of 1.0669.

11.3.2 FastMKS on a reference and query dataset

The query points may be different than the reference points. To specify a different query set, the `-q` (or `-query_file`) option is used, as in the example below.

```
$ fastmks -q query_set.csv -r reference_set.csv -k 5 -i indices.csv -p products.csv
```

11.3.3 FastMKS with a different kernel

The `fastmks` program offers more than just the linear kernel. Valid options are `'linear'`, `'polynomial'`, `'cosine'`, `'gaussian'`, `'epanechnikov'`, `'triangular'`, `'laplacian'`, and `'hyptan'` (the hyperbolic tangent kernel). Note that the hyperbolic tangent kernel is provably not a Mercer kernel but is positive semidefinite on most datasets and is commonly used as a kernel. Note also that the Gaussian kernel and other shift-invariant kernels give the same results as nearest neighbor search (see **NeighborSearch tutorial (k-nearest-neighbors)** (p. 59)).

The kernel to use is specified with the `-K` (or `-kernel`) option. The example below uses the cosine similarity as a kernel.

```
$ fastmks -r dataset.csv -k 5 -K cosine -i indices.csv -p products.csv -v
```

11.3.4 Using single-tree search or naive search

In some cases, it may be useful to not use the dual-tree FastMKS algorithm. Instead you can specify the `-single` option, indicating that a tree should be built only on the reference set, and then the queries should be processed in a linear scan (instead of in a tree). Alternately, the `-N` (or `-naive`) option makes the program not build trees at all and instead use brute-force search to find the solutions.

The example below uses single-tree search on two datasets.

```
$ fastmks -q query_set.csv -r reference_set.csv --single -k 5 -p products.csv \
> -i indices.csv
```

The example below uses naive search on one dataset.

```
$ fastmks -r reference_set.csv -k 5 -N -p products.csv -i indices.csv
```

11.3.5 Parameters for alternate kernels

Many of the alternate kernel choices have parameters which can be chosen; these are detailed in this section.

- **-w** (`-bandwidth`): this sets the bandwidth of the kernel, and is applicable to the 'gaussian', 'epanechnikov', and 'triangular' kernels. This is the "spread" of the kernel.
- **-d** (`-degree`): this sets the degree of the polynomial kernel (the power to which the result is raised). It is only applicable to the 'polynomial' kernel.
- **-o** (`-offset`): this sets the offset of the kernel, for the 'polynomial' and 'hyptan' kernel. See [the polynomial kernel documentation](#) (p.319) and [the hyperbolic tangent kernel documentation](#) (p.304) for more information.
- **-s** (`-scale`): this sets the scale of the kernel, and is only applicable to the 'hyptan' kernel. See [the hyperbolic tangent kernel documentation](#) (p.304) for more information.

11.4 The 'FastMKS' class

The `FastMKS<>` class offers a simple API for use within C++ applications, and allows further flexibility in kernel choice and tree type choice. However, `FastMKS<>` has no default template parameter for the kernel type – that must be manually specified. Choices that **mlpack** provides include:

- `mlpack::kernel::LinearKernel` (p. 315)
- `mlpack::kernel::PolynomialKernel` (p. 319)
- `mlpack::kernel::CosineDistance` (p. 293)
- `mlpack::kernel::GaussianKernel` (p. 299)
- `mlpack::kernel::EpanechnikovKernel` (p. 294)
- `mlpack::kernel::TriangularKernel` (p. 329)
- `mlpack::kernel::HyperbolicTangentKernel` (p. 304)
- `mlpack::kernel::LaplacianKernel` (p. 312)
- `mlpack::kernel::PSpectrumStringKernel` (p. 322)

The following examples use kernels from that list. Writing your own kernel is detailed in **the next section** (p.41). Remember that when you are using the C++ interface, the data matrices must be column-major. See **Matrices in MLPACK** (p. 13) for more information.

11.4.1 FastMKS on one dataset

Given only a reference dataset, the following code will run FastMKS with k set to 5.

```
#include <mlpack/methods/fastmks/fastmks.hpp>
#include <mlpack/core/kernels/linear_kernel.hpp>

using namespace mlpack::fastmks;

// The reference dataset, which is column-major.
extern arma::mat data;

// This will initialize the FastMKS object with the linear kernel with default
// options:  $K(x, y) = x^T y$ . The tree is built in the constructor.
FastMKS<LinearKernel> f(data);

// The results will be stored in these matrices.
arma::Mat<size_t> indices;
arma::mat products;

// Run FastMKS.
f.Search(5, indices, products);
```

11.4.2 FastMKS with a query and reference dataset

In this setting we have both a query and reference dataset. We search for 10 maximum kernels.

```
#include <mlpack/methods/fastmks/fastmks.hpp>
#include <mlpack/core/kernels/triangular_kernel.hpp>

using namespace mlpack::fastmks;
using namespace mlpack::kernel;

// The reference and query datasets, which are column-major.
extern arma::mat referenceData;
extern arma::mat queryData;

// This will initialize the FastMKS object with the triangular kernel with
// default options (bandwidth of 1). The trees are built in the constructor.
FastMKS<TriangularKernel> f(queryData, referenceData);

// The results will be stored in these matrices.
arma::Mat<size_t> indices;
arma::mat products;

// Run FastMKS.
f.Search(10, indices, products);
```

11.4.3 FastMKS with an initialized kernel

Often, kernels have parameters which need to be specified. FastMKS<> has constructors which take initialized kernels. Note that temporary kernels cannot be passed as an argument. The example below initializes a Polynomial< Kernel object and then runs FastMKS with a query and reference dataset.

```
#include <mlpack/methods/fastmks/fastmks.hpp>
#include <mlpack/core/kernels/polynomial_kernel.hpp>

using namespace mlpack::fastmks;
using namespace mlpack::kernel;

// The reference and query datasets, which are column-major.
extern arma::mat referenceData;
```

```
extern arma::mat queryData;

// Initialize the polynomial kernel with degree of 3 and offset of 2.5.
PolynomialKernel pk(3.0, 2.5);

// Create the FastMKS object with the initialized kernel.
FastMKS<PolynomialKernel> f(referenceData, queryData, pk);

// The results will be stored in these matrices.
arma::Mat<size_t> indices;
arma::mat products;

// Run FastMKS.
f.Search(10, indices, products);
```

The syntax for running FastMKS with one dataset and an initialized kernel is very similar:

```
FastMKS<PolynomialKernel> f(referenceData, pk);
```

11.4.4 FastMKS with an already-created tree

By default, `FastMKS<>` uses the cover tree datastructure (see `mlpack::tree::CoverTree` (p.603)). Sometimes, it is useful to modify the parameters of the cover tree. In this scenario, a tree must be built outside of the constructor, and then passed to the appropriate `FastMKS<>` constructor. An example on just a reference dataset is shown below, where the base of the cover tree is modified.

We also use an instantiated kernel, but because we are building our own tree, we must use `IPMetric` (p.365) so that our tree is built on the metric induced by our kernel function.

```
#include <mlpack/methods/fastmks/fastmks.hpp>
#include <mlpack/core/kernels/polynomial_kernel.hpp>

// The reference dataset, which is column-major.
extern arma::mat data;

// Initialize the polynomial kernel with a degree of 4 and offset of 2.0.
PolynomialKernel pk(4.0, 2.0);

// Create the metric induced by this kernel (because a kernel is not a metric
// and we can't build a tree on a kernel alone).
IPMetric<PolynomialKernel> metric(pk);

// Now build a tree on the reference dataset using the instantiated metric and
// the custom base of 1.5 (default is 1.3). We have to be sure to use the right
// type here -- FastMKS needs the FastMKSStat object as the tree's
// StatisticType.
typedef tree::CoverTree<IPMetric<PolynomialKernel>, tree::FirstPointIsRoot,
    FastMKSStat> TreeType; // Convenience typedef.
TreeType* tree = new TreeType(data, metric, 1.5);

// Now initialize FastMKS with that statistic. We don't need to specify the
// TreeType template parameter since we are still using the default. We don't
// need to pass the kernel because that is contained in the tree.
FastMKS<PolynomialKernel> f(data, tree);

// The results will be stored in these matrices.
arma::Mat<size_t> indices;
arma::mat products;

// Run FastMKS.
f.Search(10, indices, products);
```

The syntax is similar for the case where different query and reference datasets are given; but trees for both need to be built in the manner specified above. Be sure to build both trees using the same metric (or at least a metric with the exact same parameters).

```
FastMKS<PolynomialKernel> f(referenceData, referenceTree, queryData, queryTree);
```

11.5 Writing a custom kernel for FastMKS

While **mlpack** provides some number of kernels in the **mlpack::kernel** (p. 101) namespace, it is easy to create a custom kernel. To satisfy the KernelType policy, a class must implement the following methods:

```
// Empty constructor is required.
KernelType();

// Evaluate the kernel between two points.
template<typename VecType>
double Evaluate(const VecType& a, const VecType& b);
```

The template parameter `VecType` is helpful (but not necessary) so that the kernel can be used with both sparse and dense matrices (`arma::sp_mat` and `arma::mat`).

11.6 Using other tree types for FastMKS

The use of the cover tree (see **CoverTree** (p. 603)) is not necessary for FastMKS, although it is the default tree type. A different type of tree can be specified with the `TreeType` template parameter. However, the tree type is required to have **FastMKSStat** (p. 260) as the `StatisticType`, and for FastMKS to work, the tree must be built only on kernel evaluations (or distance evaluations in the kernel space via **IPMetric::Evaluate()** (p. 365)).

Below is an example where a custom tree class, `CustomTree`, is used as the tree type for FastMKS. In this example FastMKS is only run on one dataset.

```
#include <mlpack/methods/fastmks/fastmks.hpp>
#include "custom_tree.hpp"

using namespace mlpack::fastmks;
using namespace mlpack::tree;

// The dataset that FastMKS will be run on.
extern arma::mat data;

// The custom tree type. We'll assume that the first template parameter is the
// statistic type.
typedef CustomTree<FastMKSStat> TreeType;

// The FastMKS constructor will create the tree.
FastMKS<LinearKernel, TreeType> f(data);

// These will hold the results.
arma::Mat<size_t> indices;
arma::mat products;

// Run FastMKS.
f.Search(5, indices, products);
```

11.7 Running FastMKS on objects

FastMKS has a lot of utility on objects which are not representable in some sort of metric space. These objects might be strings, graphs, models, or other objects. For these types of objects, questions based on distance don't really make sense. One good example is with strings. The question "how far is 'dog' from 'Taki Inoue'?" simply doesn't make sense. We can't have a centroid of the terms 'Fritz', 'E28', and 'popsicle'.

However, what we can do is define some sort of kernel on these objects. These kernels generally correspond to some similarity measure, with one example being the p-spectrum string kernel (see **mlpack::kernel::PSpectrumStringKernel** (p. 322)). Using that, we can say "how similar is 'dog' to 'Taki Inoue'?" and get an actual numerical result by evaluating $K(\text{'dog'}, \text{'Taki Inoue'})$ (where K is our p-spectrum string kernel).

The only requirement on these kernels is that they are positive definite kernels (or Mercer kernels). For more information on those details, refer to the FastMKS paper.

Remember that FastMKS is a tree-based method. But trees like the binary space tree require centroids – and as we said earlier, centroids often don't make sense with these types of objects. Therefore, we need a type of tree which is built **exclusively** on points in the dataset – those are points which we can evaluate our kernel function on. The cover tree is one example of a type of tree satisfying this condition; its construction will only call the kernel function on two points that are in the dataset.

But, we have one more problem. The `CoverTree` class is built on `arma::mat` objects (dense matrices). Our objects, however, are not necessarily representable in a column of a matrix. To use the example we have been using, strings cannot be represented easily in a matrix because they may all have different lengths.

The way to work around this problem is to create a "fake" data matrix which simply holds indices to objects. A good example of how to do this is detailed in the documentation for the **PSpectrumStringKernel** (p. 322).

In short, the trick is to make each data matrix one-dimensional and containing linear indices:

```
arma::mat data = "0 1 2 3 4 5 6 7 8";
```

Then, when `Evaluate()` is called on the kernel function, the parameters will be two one-dimensional vectors that simply contain indices to objects. The example below details the process a little better:

```
// This function evaluates the kernel on two Objects (in this example, its
// implementation is not important; the only important thing is that the
// function exists).
double ObjectKernel::Evaluate(const Object& a, const Object& b) const;

template<typename VecType>
double ObjectKernel::Evaluate(const VecType& a, const VecType& b) const
{
    // Extract the indices from the vectors.
    const size_t indexA = size_t(a[0]);
    const size_t indexB = size_t(b[0]);

    // Assume that 'objects' is an array (or std::vector or other container)
    // holding Objects.
    const Object& objectA = objects[indexA];
    const Object& objectB = objects[indexB];

    // Now call the function that does the actual evaluation on the objects and
    // return its result.
    return Evaluate(objectA, objectB);
}
```

As written earlier, the documentation for **PSpectrumStringKernel** (p. 322) is a good place to consult for further reference on this. That kernel uses two dimensional indices; one dimension represents the index of the string, and the other represents whether it is referring to the query set or the reference set. If your kernel is meant to work on separate query and reference sets, that strategy should be considered.

11.8 Further documentation

For further documentation on the FastMKS class, consult the **complete API documentation** (p. 247).

Chapter 12

K-Means tutorial (kmeans)

12.1 Introduction

The popular k-means algorithm for clustering has been around since the late 1950s, and the standard algorithm was proposed by Stuart Lloyd in 1957. Given a set of points X , k-means clustering aims to partition each point x_i into a cluster c_j (where $j \leq k$ and k , the number of clusters, is a parameter). The partitioning is done to minimize the objective function

$$\sum_{j=1}^k \sum_{x_i \in c_j} \|x_i - \mu_j\|^2$$

where μ_j is the centroid of cluster c_j . The standard algorithm is a two-step algorithm:

- **Assignment step.** Each point x_i in X is assigned to the cluster whose centroid it is closest to.
- **Update step.** Using the new cluster assignments, the centroids of each cluster are recalculated.

The algorithm has converged when no more assignment changes are happening with each iteration. However, this algorithm can get stuck in local minima of the objective function and is particularly sensitive to the initial cluster assignments. Also, situations can arise where the algorithm will never converge but reaches steady state – for instance, one point may be changing between two cluster assignments.

There is vast literature on the k-means algorithm and its uses, as well as strategies for choosing initial points effectively and keeping the algorithm from converging in local minima. **mlpack** does implement some of these, notably the Bradley-Fayyad algorithm (see the reference below) for choosing refined initial points. Importantly, the C++ `KMeans` class makes it very easy to improve the k-means algorithm in a modular way.

```
@inproceedings{bradley1998refining,
  title={Refining initial points for k-means clustering},
  author={Bradley, Paul S. and Fayyad, Usama M.},
  booktitle={Proceedings of the Fifteenth International Conference on Machine
    Learning (ICML 1998)},
  volume={66},
  year={1998}
}
```

mlpack provides:

- a **simple command-line executable** (p. 44) to run k-means
- a **simple C++ interface** (p. 46) to run k-means
- a **generic, extensible, and powerful C++ class** (p. 49) for complex usage

12.2 Table of Contents

A list of all the sections this tutorial contains.

- **Introduction** (p. 43)
- **Table of Contents** (p. 44)
- **Command-Line 'kmeans'** (p. 44)
 - **Simple k-means clustering** (p. 44)
 - **Saving the resulting centroids** (p. 45)
 - **Allowing empty clusters** (p. 45)
 - **Limiting the maximum number of iterations** (p. 45)
 - **Setting the overclustering factor** (p. 45)
 - **Using Bradley-Fayyad "refined start"** (p. 45)
- **The 'KMeans' class** (p. 46)
 - **Running k-means and getting cluster assignments** (p. 46)
 - **Running k-means and getting centroids of clusters** (p. 46)
 - **Limiting the maximum number of iterations** (p. 47)
 - **Setting the overclustering factor** (p. 47)
 - **Setting initial cluster assignments** (p. 47)
 - **Setting initial cluster centroids** (p. 48)
 - **Running sparse k-means** (p. 49)
- **Template parameters for the 'KMeans' class** (p. 49)
 - **Changing the distance metric used for k-means** (p. 49)
 - **Changing the initial partitioning strategy used for k-means** (p. 50)
 - **Changing the action taken when an empty cluster is encountered** (p. 51)
- **Further documentation** (p. 51)

12.3 Command-Line 'kmeans'

mlpack provides a command-line executable, `kmeans`, to allow easy execution of the k-means algorithm on data. Complete documentation of the executable can be found by typing

```
$ kmeans --help
```

Below are several examples demonstrating simple use of the `kmeans` executable.

12.3.1 Simple k-means clustering

We want to find 5 clusters using the points in the file `dataset.csv`. By default, if any of the clusters end up empty, that cluster will be reinitialized to contain the point furthest from the cluster with maximum variance. The cluster assignments of each point will be stored in `assignments.csv`. Each row in `assignments.csv` will correspond to the row in `dataset.csv`.

```
$ kmeans -c 5 -i dataset.csv -v -o assignments.csv
```


12.3.2 Saving the resulting centroids

Sometimes it is useful to save the centroids of the clusters found by k-means; one example might be for plotting the points. The `-C` (`-centroid_file`) option allows specification of a file into which the centroids will be saved (one centroid per line, if it is a CSV or other text format).

```
$ kmeans -c 5 -i dataset.csv -v -o assignments.csv -C centroids.csv
```

12.3.3 Allowing empty clusters

If you would like to allow empty clusters to exist, instead of reinitializing them, simply specify the `-e` (`-allow_empty_clusters`) option. Note that when you save your clusters, some of the clusters may be filled with NaNs. This is expected behavior – if a cluster has no points, the concept of a centroid makes no sense.

```
$ kmeans -c 5 -i dataset.csv -v -o assignments.csv -C centroids.csv
```

12.3.4 Limiting the maximum number of iterations

As mentioned earlier, the k-means algorithm can often fail to converge. In such a situation, it may be useful to stop the algorithm by way of limiting the maximum number of iterations. This can be done with the `-m` (`-max_iterations`) parameter, which is set to 1000 by default. If the maximum number of iterations is 0, the algorithm will run until convergence – or potentially forever. The example below sets a maximum of 250 iterations.

```
$ kmeans -c 5 -i dataset.csv -v -o assignments.csv -m 250
```

12.3.5 Setting the overclustering factor

The **mlpack** k-means implementation allows "overclustering", which is when the k-means algorithm is run with more than the requested number of clusters. Upon convergence, the clusters with the nearest centroids are merged until only the requested number of centroids remain. This can provide better clustering results. The overclustering factor, specified with `-O` or `-overclustering`, determines how many more clusters are found than were requested. For instance, with `k` set to 5 and an overclustering factor of 2, 10 clusters will be found. Note that the overclustering factor does not need to be an integer.

The following code snippet finds 5 clusters, but with an overclustering factor of 2.4 (so 12 clusters are found and then merged together to produce 5 final clusters).

```
$ kmeans -c 5 -O 2.4 -i dataset.csv -v -o assignments.csv
```

12.3.6 Using Bradley-Fayyad "refined start"

The method proposed by Bradley and Fayyad in their paper "Refining initial points for k-means clustering" is implemented in **mlpack**. This strategy samples points from the dataset and runs k-means clustering on those points multiple times, saving the resulting clusters. Then, k-means clustering is run on those clusters, yielding the original number of clusters. The centroids of those resulting clusters are used as initial centroids for k-means clustering on the entire dataset.

This technique generally gives better initial points than the default random partitioning, but depending on the parameters, it can take much longer. This initialization technique is enabled with the `-r` (`-refined_start`) option. The `-S` (`-samplings`) parameter controls how many samplings of the dataset are performed, and the `-p` (`-percentage`) parameter controls how much of the dataset is randomly sampled for each sampling (it must be between 0.0 and 1.0). For more information on the refined start technique, see the paper referenced in the introduction of this tutorial.

The example below performs k-means clustering, giving 5 clusters, using the refined start technique, sampling 10% of the dataset 25 times to produce the initial centroids.

```
$ kmeans -c 5 -i dataset.csv -v -o assignments.csv -r -S 25 -p 0.2
```

12.4 The 'KMeans' class

The `KMeans<>` class (with default template parameters) provides a simple way to run k-means clustering using **mlpack** in C++. The default template parameters for `KMeans<>` will initialize cluster assignments randomly and disallow empty clusters. When an empty cluster is encountered, the point furthest from the cluster with maximum variance is set to the centroid of the empty cluster.

12.4.1 Running k-means and getting cluster assignments

The simplest way to use the `KMeans<>` class is to pass in a dataset and a number of clusters, and receive the cluster assignments in return. Note that the dataset must be column-major – that is, one column corresponds to one point. See [the matrices guide](#) (p. 13) for more information.

```
#include <mlpack/methods/kmeans/kmeans.hpp>

using namespace mlpack::kmeans;

// The dataset we are clustering.
extern arma::mat data;
// The number of clusters we are getting.
extern size_t clusters;

// The assignments will be stored in this vector.
arma::Col<size_t> assignments;

// Initialize with the default arguments.
KMeans<> k;
k.Cluster(data, clusters, assignments);
```

Now, the vector `assignments` holds the cluster assignments of each point in the dataset.

12.4.2 Running k-means and getting centroids of clusters

Often it is useful to not only have the cluster assignments, but the centroids of each cluster. Another overload of `Cluster()` makes this easily possible:

```
#include <mlpack/methods/kmeans/kmeans.hpp>

using namespace mlpack::kmeans;

// The dataset we are clustering.
extern arma::mat data;
// The number of clusters we are getting.
extern size_t clusters;

// The assignments will be stored in this vector.
arma::Col<size_t> assignments;
// The centroids will be stored in this matrix.
arma::mat centroids;

// Initialize with the default arguments.
KMeans<> k;
k.Cluster(data, clusters, assignments, centroids);
```

Note that the centroids matrix has columns equal to the number of clusters and rows equal to the dimensionality of the dataset. Each column represents the centroid of the according cluster – `centroids.col(0)` represents the centroid of the first cluster.

12.4.3 Limiting the maximum number of iterations

The first argument to the constructor allows specification of the maximum number of iterations. This is useful because often, the k-means algorithm does not converge, and is terminated after a number of iterations. Setting this parameter to 0 indicates that the algorithm will run until convergence – note that in some cases, convergence may never happen. The default maximum number of iterations is 1000.

```
// The first argument is the maximum number of iterations. Here we set it to
// 500 iterations.
KMeans<> k(500);
```

Then you can run `Cluster()` as normal.

12.4.4 Setting the overclustering factor

For a description of what overclustering is, see [the command-line interface tutorial about overclustering](#) (p. 45).

The overclustering factor, which by default is 1.0 (this indicates that no overclustering is happening), is specified in the second argument to the constructor.

```
// We will keep the default maximum iterations of 1000, but set the
// overclustering factor to 2.5.
KMeans<> k(1000, 2.5);
```

Then you can run `Cluster()` as normal.

12.4.5 Setting initial cluster assignments

If you have an initial guess for the cluster assignments for each point, you can fill the assignments vector with the guess and then pass an extra boolean (`initialAssignmentGuess`) as true to the `Cluster()` method. Below are examples for either overload of `Cluster()`.

```
#include <mlpack/methods/kmeans/kmeans.hpp>

using namespace mlpack::kmeans;

// The dataset we are clustering on.
extern arma::mat dataset;
// The number of clusters we are obtaining.
extern size_t clusters;

// A vector pre-filled with initial assignment guesses.
extern arma::Col<size_t> assignments;

KMeans<> k;

// The boolean set to true indicates that our assignments vector is filled with
// initial guesses.
k.Cluster(dataset, clusters, assignments, true);

#include <mlpack/methods/kmeans/kmeans.hpp>

using namespace mlpack::kmeans;

// The dataset we are clustering on.
extern arma::mat dataset;
// The number of clusters we are obtaining.
extern size_t clusters;

// A vector pre-filled with initial assignment guesses.
extern arma::Col<size_t> assignments;

// This will hold the centroids of the finished clusters.
arma::mat centroids;
```

```
KMeans<> k;

// The boolean set to true indicates that our assignments vector is filled with
// initial guesses.
k.Cluster(dataset, clusters, assignments, centroids, true);
```

Note

If you have a heuristic or algorithm which makes initial guesses, a more elegant solution is to create a new class fulfilling the `InitialPartitionPolicy` template policy. See **the section about changing the initial partitioning strategy** (p. 50) for more details.

Note

If you set the `InitialPartitionPolicy` parameter to something other than the default but give an initial cluster assignment guess, the `InitialPartitionPolicy` will not be used to initialize the algorithm. See **the section about changing the initial partitioning strategy** (p. 50) for more details.

12.4.6 Setting initial cluster centroids

An equally important option to being able to make initial cluster assignment guesses is to make initial cluster centroid guesses without having to assign each point in the dataset to an initial cluster. This is similar to the previous section, but now you must pass two extra booleans – the first (`initialAssignmentGuess`) as false, indicating that there are not initial cluster assignment guesses, and the second (`initialCentroidGuess`) as true, indicating that the centroids matrix is filled with initial centroid guesses.

This, of course, only works with the overload of `Cluster()` that takes a matrix to put the resulting centroids in. Below is an example.

```
#include <mlpack/methods/kmeans/kmeans.hpp>

using namespace mlpack::kmeans;

// The dataset we are clustering on.
extern arma::mat dataset;
// The number of clusters we are obtaining.
extern size_t clusters;

// A matrix pre-filled with guesses for the initial cluster centroids.
extern arma::mat centroids;

// This will be filled with the final cluster assignments for each point.
arma::Col<size_t> assignments;

KMeans<> k;

// Remember, the first boolean indicates that we are not giving initial
// assignment guesses, and the second boolean indicates that we are giving
// initial centroid guesses.
k.Cluster(dataset, clusters, assignments, centroids, false, true);
```

Note

If you have a heuristic or algorithm which makes initial guesses, a more elegant solution is to create a new class fulfilling the `InitialPartitionPolicy` template policy. See **the section about changing the initial partitioning strategy** (p. 50) for more details.

Note

If you set the `InitialPartitionPolicy` parameter to something other than the default but give an initial cluster centroid guess, the `InitialPartitionPolicy` will not be used to initialize the algorithm. See **the section about changing the initial partitioning strategy** (p. 50) for more details.

12.4.7 Running sparse k-means

The `Cluster()` function can work on both sparse and dense matrices, so all of the above examples can be used with sparse matrices instead. Below is a simple example. Note that the centroids are returned as a sparse matrix also.

```
// The sparse dataset.
extern arma::sp_mat sparseDataset;
// The number of clusters.
extern size_t clusters;

// The assignments will be stored in this vector.
arma::Col<size_t> assignments;
// The centroids of each cluster will be stored in this sparse matrix.
arma::sp_mat sparseCentroids;

// No template parameter modification is necessary.
KMeans<> k;
k.Cluster(sparseDataset, clusters, assignments, sparseCentroids);
```

12.5 Template parameters for the 'KMeans' class

The `KMeans<>` class also takes three template parameters, which can be modified to change the behavior of the k-means algorithm. There are three template parameters:

- `MetricType`: controls the distance metric used for clustering (by default, the squared Euclidean distance is used)
- `InitialPartitionPolicy`: the method by which initial clusters are set; by default, **RandomPartition** (p. 342) is used
- `EmptyClusterPolicy`: the action taken when an empty cluster is encountered; by default, **MaxVarianceNewCluster** (p. 340) is used

The class is defined like below:

```
template<
    typename DistanceMetric = mlpack::metric::SquaredEuclideanDistance,
    typename InitialPartitionPolicy = RandomPartition,
    typename EmptyClusterPolicy = MaxVarianceNewCluster
>
class KMeans;
```

In the following sections, each policy is described further, with examples of how to modify them.

12.5.1 Changing the distance metric used for k-means

Most machine learning algorithms in **mlpack** support modifying the distance metric, and `KMeans<>` is no exception. Similar to **NeighborSearch** (p. 399) (see **the section in the NeighborSearch tutorial** (p. 63)), any class in **mlpack** `::metric` (p. 109) can be given as an argument. The **mlpack::metric::LMetric** (p. 367) class is a good example implementation.

A class fulfilling the `MetricType` policy must provide the following two functions:

```
// Empty constructor is required.
MetricType();

// Computer the distance between two points.
template<typename VecType>
double Evaluate(const VecType& a, const VecType& b);
```

Most of the standard metrics that could be used are stateless and therefore the `Evaluate()` method is implemented statically. However, there are metrics, such as the Mahalanobis distance (**mlpack::metric::MahalanobisDistance** (p.368)), that store state. To this end, an instantiated `MetricType` object is stored within the `KMeans` class. The example below shows how to pass an instantiated `MahalanobisDistance` in the constructor.

```
// The initialized Mahalanobis distance.
extern mlpack::metric::MahalanobisDistance distance;

// We keep the default arguments for the maximum number of iterations and
// overclustering factor, but pass our instantiated metric.
KMeans<mlpack::metric::MahalanobisDistance> k(1000, 1.0, distance);
```

Note

While the `MetricType` policy only requires two methods, one of which is an empty constructor, more can always be added. **mlpack::metric::MahalanobisDistance** (p. 368) also has constructors with parameters, because it is a stateful metric.

12.5.2 Changing the initial partitioning strategy used for k-means

There have been many initial cluster strategies for k-means proposed in the literature. Fortunately, the `KMeans<>` class makes it very easy to implement one of these methods and plug it in without needing to modify the existing algorithm code at all.

By default, the `KMeans<>` class uses **mlpack::kmeans::RandomPartition** (p. 342), which randomly partitions points into clusters. However, writing a new policy is simple; it needs to only implement the following functions:

```
// Empty constructor is required.
InitialPartitionPolicy();

// This function is called to initialize the clusters.
template<typename MatType>
void Cluster(MatType& data,
             const size_t clusters,
             arma::Col<size_t> assignments);
```

The templatzation of the `Cluster()` function allows both dense and sparse matrices to be passed in. If the desired policy does not work with sparse (or dense) matrices, then the method can be written specifically for one type of matrix – however, be warned that if you try to use `KMeans` with that policy and the wrong type of matrix, you will get many ugly compilation errors!

```
// The Cluster() function specialized for dense matrices.
void Cluster(arma::mat& data,
             const size_t clusters,
             arma::Col<size_t> assignments);
```

One alternate to the default `RandomPartition` policy is the `RefinedStart` policy, which is an implementation of the Bradley and Fayyad approach for finding initial points detailed in "Refined initial points for k-means clustering" and other places in this document. Also see the documentation for **mlpack::kmeans::RefinedStart** (p. 343) for more information.

The `Cluster()` method must return valid initial assignments for every point in the dataset.

As with the `MetricType` template parameter, an initialized `InitialPartitionPolicy` can be passed to the constructor of `KMeans` as a fourth argument.

12.5.3 Changing the action taken when an empty cluster is encountered

Sometimes, during clustering, a situation will arise where a cluster has no points in it. The `KMeans` class allows easy customization of the action to be taken when this occurs. By default, the point furthest from the centroid of the cluster with maximum variance is taken as the centroid of the empty cluster; this is implemented in the `mlpack::kmeans::MaxVarianceNewCluster` (p. 340) class. Another alternate choice is the `mlpack::kmeans::AllowEmptyClusters` (p. 332) class, which simply allows empty clusters to persist.

A custom policy can be written and it must implement the following methods:

```
// Empty constructor is required.
EmptyClusterPolicy();

// This function is called when an empty cluster is encountered. emptyCluster
// indicates the cluster which is empty, and then the clusterCounts and
// assignments are meant to be modified by the function. The function should
// return the number of modified points.
template<typename MatType>
size_t EmptyCluster(const MatType& data,
                    const size_t emptyCluster,
                    const MatType& centroids,
                    arma::Col<size_t>& clusterCounts,
                    arma::Col<size_t>& assignments);
```

The `EmptyCluster()` function is called for each cluster that is empty at each iteration of the algorithm. As with `InitialPartitionPolicy`, the `EmptyCluster()` function does not need to be generalized to support both dense and sparse matrices – but usage with the wrong type of matrix will cause compilation errors.

Like the other template parameters to `KMeans`, `EmptyClusterPolicy` implementations that have state can be passed to the constructor of `KMeans` as a fifth argument. See the `kmeans::KMeans` documentation for further details.

12.6 Further documentation

For further documentation on the `KMeans` class, consult the **complete API documentation** (p. 333).

Chapter 13

Linear/ridge regression tutorial (linear_regression)

13.1 Introduction

Linear regression and ridge regression are simple machine learning techniques that aim to estimate the parameters of a linear model. Assuming we have n **predictor** points \mathbf{x}_i , $0 \leq i < n$ of dimensionality d and n responses y_i , $0 \leq i < n$, we are trying to estimate the best fit for β_i , $0 \leq i \leq d$ in the linear model

$$y_i = \beta_0 + \sum_{j=1}^d \beta_j x_{ij}$$

for each predictor \mathbf{x}_i and response y_i . If we take each predictor \mathbf{x}_i as a row in the matrix \mathbf{X} and each response y_i as an entry of the vector \mathbf{y} , we can represent the model in vector form:

$$\mathbf{y} = \mathbf{X}\boldsymbol{\beta} + \beta_0$$

The result of this method is the vector $\boldsymbol{\beta}$, including the offset term (or intercept term) β_0 .

mlpack provides:

- a **simple command-line executable** (p. 54) to perform linear regression or ridge regression
- a **simple C++ interface** (p. 57) to perform linear regression or ridge regression

13.2 Table of Contents

A list of all the sections this tutorial contains.

- **Introduction** (p. 53)
- **Table of Contents** (p. 53)
- **Command-Line 'linear_regression'** (p. 54)
 - **One file, generating the function coefficients** (p. 54)
 - **Compute model and predict at the same time** (p. 55)
 - **Prediction using a precomputed model** (p. 55)

- Using ridge regression (p. 56)
- The 'LinearRegression' class (p. 57)
 - Generating a model (p. 57)
 - Setting a model (p. 57)
 - Load a model from a file (p. 57)
 - Prediction (p. 57)
 - Setting lambda for ridge regression (p. 58)
- Further documentation (p. 58)

13.3 Command-Line 'linear_regression'

The simplest way to perform linear regression or ridge regression in **mlpack** is to use the `linear_regression` executable. This program will perform linear regression and place the resultant coefficients into one file.

The output file holds a vector of coefficients in increasing order of dimension; that is, the offset term (β_0), the coefficient for dimension 1 (β_1), then dimension 2 (β_2) and so forth, as well as the intercept. This executable can also predict the y values of a second dataset based on the computed coefficients.

Below are several examples of simple usage (and the resultant output). The '-v' option is used so that verbose output is given. Further documentation on each individual option can be found by typing

```
$ linear_regression --help
```

13.3.1 One file, generating the function coefficients

```
$ linear_regression --input_file dataset.csv -v
[INFO ] Loading 'dataset.csv' as CSV data.
[INFO ] Saving CSV data to 'parameters.csv'.
[INFO ]
[INFO ] Execution parameters:
[INFO ]   help: false
[INFO ]   info: ""
[INFO ]   input_file: dataset.csv
[INFO ]   input_responses: ""
[INFO ]   lambda: 0
[INFO ]   output_file: parameters.csv
[INFO ]   output_predictions: predictions.csv
[INFO ]   test_file: ""
[INFO ]   verbose: true
[INFO ]
[INFO ] Program timers:
[INFO ]   load_regressors: 0.006461s
[INFO ]   regression: 0.000347s
[INFO ]   total_time: 0.026589s
```

Convenient program timers are given for different parts of the calculation at the bottom of the output, as well as the parameters the simulation was run with. Now, if we look at the output file, which, unless specified, is `parameters.csv`:

```
$ cat dataset.csv
0,0
1,1
2,2
3,3
4,4

$ cat parameters.csv
-0.0000000000e+00,1.0000000000e+00
```

As you can see, the function for this input is $f(y) = 0 + 1x_1$. Keep in mind that in this example, the regressors for the dataset are the second column. That is, the dataset is one dimensional, and the last column has the y values, or responses, for each row. You can specify these responses in a separate file if you want, using the `-input_responses`, or `-r`, option.

13.3.2 Compute model and predict at the same time

```
$ linear_regression --input_file dataset.csv --test_file predict.csv -v
[INFO ] Loading 'dataset.csv' as CSV data.
[INFO ] Saving CSV data to 'parameters.csv'.
[INFO ] Loading 'predict.csv' as CSV data.
[INFO ] Saving CSV data to 'predictions.csv'.
[INFO ]
[INFO ] Execution parameters:
[INFO ]   help: false
[INFO ]   info: ""
[INFO ]   input_file: dataset.csv
[INFO ]   input_responses: ""
[INFO ]   lambda: 0
[INFO ]   model_file: ""
[INFO ]   output_file: parameters.csv
[INFO ]   output_predictions: predictions.csv
[INFO ]   test_file: predict.csv
[INFO ]   verbose: true
[INFO ]
[INFO ] Program timers:
[INFO ]   load_regressors: 0.000360s
[INFO ]   load_test_points: 0.000090s
[INFO ]   prediction: 0.000006s
[INFO ]   regression: 0.000335s
[INFO ]   total_time: 0.001522s

$ cat dataset.csv
0,0
1,1
2,2
3,3
4,4

$ cat parameters.csv
-0.00000000000e+00,1.0000000000e+00

$ cat predict.csv
2
3
4

$ cat predictions.csv
2.0000000000e+00
3.0000000000e+00
4.0000000000e+00
```

We used the same dataset, so we got the same parameters. The key thing to note about the predict.csv dataset is that it has the same dimensionality as the dataset used to create the model, one. Generally, if the model generating dataset has d dimensions, so must the dataset we want to predict for.

13.3.3 Prediction using a precomputed model

```
$ linear_regression --model_file parameters.csv --test_file predict.csv -v
[INFO ] Loading 'parameters.csv' as CSV data.
[INFO ] Loading 'predict.csv' as CSV data.
[INFO ] Saving CSV data to 'predictions.csv'.
[INFO ]
[INFO ] Execution parameters:
[INFO ]   help: false
[INFO ]   info: ""
[INFO ]   input_file: ""
[INFO ]   input_responses: ""
[INFO ]   lambda: 0
[INFO ]   model_file: parameters.csv
[INFO ]   output_file: parameters.csv
```

```
[INFO ] output_predictions: predictions.csv
[INFO ] test_file: predict.csv
[INFO ] verbose: true
[INFO ]
[INFO ] Program timers:
[INFO ]   load_model: 0.009519s
[INFO ]   load_test_points: 0.000067s
[INFO ]   prediction: 0.000007s
[INFO ]   total_time: 0.010081s

$ cat parameters.csv
-0.0000000000e+00,1.0000000000e+00

$ cat predict.csv
2
3
4

$ cat predictions.csv
2.0000000000e+00
3.0000000000e+00
4.0000000000e+00
```

13.3.4 Using ridge regression

Sometimes, the input matrix of predictors has a covariance matrix that is not invertible, or the system is overdetermined. In this case, ridge regression is useful: it adds a normalization term to the covariance matrix to make it invertible. Ridge regression is a standard technique and documentation for the mathematics behind it can be found anywhere on the Internet. In short, the covariance matrix

$$\mathbf{X}'\mathbf{X}$$

is replaced with

$$\mathbf{X}'\mathbf{X} + \lambda \mathbf{I}$$

where \mathbf{I} is the identity matrix. So, a λ parameter greater than zero should be specified to perform ridge regression, using the `-lambda` (or `-l`) option. An example is given below.

```
$ linear_regression --input_file dataset.csv -v --lambda 0.5
[INFO ] Loading 'dataset.csv' as CSV data. Size is 3 x 1000.
[INFO ] Saving CSV data to 'parameters.csv'.
[INFO ]
[INFO ] Execution parameters:
[INFO ]   help: false
[INFO ]   info: ""
[INFO ]   input_file: test_data_3_1000.csv
[INFO ]   input_responses: ""
[INFO ]   lambda: 0.5
[INFO ]   model_file: ""
[INFO ]   output_file: parameters.csv
[INFO ]   output_predictions: predictions.csv
[INFO ]   test_file: ""
[INFO ]   verbose: true
[INFO ]
[INFO ] Program timers:
[INFO ]   load_regressors: 0.005236s
[INFO ]   loading_data: 0.005208s
[INFO ]   regression: 0.013206s
[INFO ]   saving_data: 0.000276s
[INFO ]   total_time: 0.020019s
```

Further documentation on options should be found by using the `-help` option.

13.4 The 'LinearRegression' class

The 'LinearRegression' class is a simple implementation of linear regression.

Using the LinearRegression class is very simple. It has two available constructors; one for generating a model from a matrix of predictors and a vector of responses, and one for loading an already computed model from a given file.

The class provides one method that performs computation:

```
void Predict(const arma::mat& points, arma::vec& predictions);
```

Once you have generated or loaded a model, you can call this method and pass it a matrix of data points to predict values for using the model. The second parameter, predictions, will be modified to contain the predicted values corresponding to each row of the points matrix.

13.4.1 Generating a model

```
#include <mlpack/methods/linear_regression/linear_regression.hpp>

using namespace mlpack::regression;

arma::mat data; // The dataset itself.
arma::vec responses; // The responses, one row for each row in data.

// Regress.
LinearRegression lr(data, responses);

// Get the parameters, or coefficients.
arma::vec parameters = lr.Parameters();
```

13.4.2 Setting a model

Assuming you already have a model and do not need to create one, this is how you would set the parameters for a LinearRegression instance.

```
arma::vec parameters; // Your model.

LinearRegression lr(); // Create a new LinearRegression instance or reuse one.
lr.Parameters() = parameters; // Set the model.
```

13.4.3 Load a model from a file

If you have a generated model in a file somewhere you would like to load and use, you can simply pass it to the LinearRegression initializer like so.

```
std::string filename; // The path and name of your file.

LinearRegression lr(filename); // Will load the model internally.
```

13.4.4 Prediction

Once you have generated or loaded a model using one of the aforementioned methods, you can predict values for a dataset.

```
LinearRegression lr();
// Load or generate your model.

// The dataset we want to predict on; each row is a data point.
```

```
arma::mat points;
// This will store the predictions; one row for each point.
arma::vec predictions;

lr.Predict(points, predictions); // Predict.

// Now, the vector 'predictions' will contain the predicted values.
```

13.4.5 Setting lambda for ridge regression

As discussed in **Using ridge regression** (p. 56), ridge regression is useful when the covariance of the predictors is not invertible. The standard constructor can be used to set a value of lambda:

```
#include <mlpack/methods/linear_regression/linear_regression.hpp>

using namespace mlpack::regression;

arma::mat data; // The dataset itself.
arma::vec responses; // The responses, one row for each row in data.

// Regress, with a lambda of 0.5.
LinearRegression lr(data, responses, 0.5);

// Get the parameters, or coefficients.
arma::vec parameters = lr.Parameters();
```

In addition, the `Lambda()` function can be used to get or modify the lambda value:

```
LinearRegression lr;
lr.Lambda() = 0.5;
Log::Info << "Lambda is " << lr.Lambda() << "." << std::endl;
```

13.5 Further documentation

For further documentation on the `LinearRegression` class, consult the **complete API documentation** (p. 532).

Chapter 14

NeighborSearch tutorial (k-nearest-neighbors)

14.1 Introduction

Nearest-neighbors search is a common machine learning task. In this setting, we have a **query** and a **reference** dataset. For each point in the **query** dataset, we wish to know the k points in the **reference** dataset which are closest to the given query point.

Alternately, if the query and reference datasets are the same, the problem can be stated more simply: for each point in the dataset, we wish to know the k nearest points to that point.

mlpack provides:

- a **simple command-line executable** (p. 60) to run nearest-neighbors search (and furthest-neighbors search)
- a **simple C++ interface** (p. 62) to perform nearest-neighbors search (and furthest-neighbors search)
- a **generic, extensible, and powerful C++ class (NeighborSearch)** (p. 63) for complex usage

14.2 Table of Contents

A list of all the sections this tutorial contains.

- **Introduction** (p. 59)
- **Table of Contents** (p. 59)
- **Command-Line 'allknn'** (p. 60)
 - **One dataset, 5 nearest neighbors** (p. 60)
 - **Query and reference dataset, 10 nearest neighbors** (p. 61)
 - **One dataset, 3 nearest neighbors, leaf size of 15 points** (p. 61)
- **The 'AllkNN' class** (p. 62)
 - **5 nearest neighbors on a single dataset** (p. 62)
 - **10 nearest neighbors on a query and reference dataset** (p. 62)
 - **Naive (exhaustive) search for 6 nearest neighbors on one dataset** (p. 62)
- **The extensible 'NeighborSearch' class** (p. 63)

- **SortPolicy** policy class (p. 63)
- **MetricType** policy class (p. 63)
- **TreeType** policy class (p. 64)
- **Further documentation** (p. 64)

14.3 Command-Line 'allknn'

The simplest way to perform nearest-neighbors search in **mlpack** is to use the **allknn** executable. This program will perform nearest-neighbors search and place the resultant neighbors into one file and the resultant distances into another. The output files are organized such that the first row corresponds to the nearest neighbors of the first query point, with the first column corresponding to the nearest neighbor, and so forth.

Below are several examples of simple usage (and the resultant output). The '-v' option is used so that output is given. Further documentation on each individual option can be found by typing

```
$ allknn --help
```

14.3.1 One dataset, 5 nearest neighbors

```
$ allknn -r dataset.csv -n neighbors_out.csv -d distances_out.csv -k 5 -v
[INFO ] Loading 'dataset.csv' as CSV data.
[INFO ] Loaded reference data from 'dataset.csv'.
[INFO ] Building reference tree...
[INFO ] Trees built.
[INFO ] Computing 5 nearest neighbors...
[INFO ] Neighbors computed.
[INFO ] Re-mapping indices...
[INFO ] Saving CSV data to 'distances_out.csv'.
[INFO ] Saving CSV data to 'neighbors_out.csv'.
[INFO ]
[INFO ] Execution parameters:
[INFO ]   distances_file: distances_out.csv
[INFO ]   help: false
[INFO ]   info: ""
[INFO ]   k: 5
[INFO ]   leaf_size: 20
[INFO ]   naive: false
[INFO ]   neighbors_file: neighbors_out.csv
[INFO ]   query_file: ""
[INFO ]   reference_file: dataset.csv
[INFO ]   single_mode: false
[INFO ]   verbose: true
[INFO ]
[INFO ] Program timers:
[INFO ]   computing_neighbors: 0.152495s
[INFO ]   total_time: 0.201274s
[INFO ]   tree_building: 0.005050s
```

Convenient program timers are given for different parts of the calculation at the bottom of the output, as well as the parameters the simulation was run with. Now, if we look at the output files:

```
$ head neighbors_out.csv
14,5,13,16,27
90,79,80,15,10
39,84,10,123,1
81,43,109,12,37
15,1,79,90,10
0,14,16,13,27
90,79,11,1,15
41,45,12,37,49
11,81,13,6,15
41,7,45,49,47

$ head distances_out.csv
```



```

7.09614421e-04,2.05940173e-03,4.05346068e-03,4.66175278e-03,1.09757665e-02
8.92190948e-04,1.69442242e-03,2.82750475e-03,4.06590850e-03,7.54169243e-03
5.91539406e-03,6.83482612e-03,8.02877800e-03,9.04907425e-03,1.61458442e-02
7.15652913e-03,9.18228524e-03,1.00540941e-02,1.07541171e-02,1.28892864e-02
5.37535983e-03,9.05721409e-03,9.89017184e-03,1.01457735e-02,1.14021593e-02
2.05940173e-03,5.14437192e-03,9.97483954e-03,1.02463627e-02,1.44355783e-02
4.27355419e-03,6.36750547e-03,6.72478577e-03,8.77323532e-03,1.04530549e-02
1.99935847e-03,3.88240331e-03,4.19118273e-03,9.30693568e-03,1.21237481e-02
2.15454276e-03,8.18895210e-03,1.18360450e-02,1.25135454e-02,1.27783327e-02
8.43087996e-03,1.22946325e-02,1.60472209e-02,1.88661413e-02,1.89727686e-02

```

So, the nearest neighbor to point 0 is point 14, with a distance of 7.096144e-4. The second nearest neighbor to point 0 is point 5, with a distance of 2.059402e-3. The third nearest neighbor to point 5 is point 16, with a distance of 9.9748395e-3.

14.3.2 Query and reference dataset, 10 nearest neighbors

```

$ allknn -q query_dataset.csv -r reference_dataset.csv -n neighbors_out.csv \
> -d distances_out.csv -k 10 -v
[INFO ] Loading 'reference_dataset.csv' as CSV data.
[INFO ] Loaded reference data from 'reference_dataset.csv'.
[INFO ] Building reference tree...
[INFO ] Loading 'query_dataset.csv' as CSV data.
[INFO ] Query data loaded from 'query_dataset.csv'.
[INFO ] Building query tree...
[INFO ] Tree built.
[INFO ] Computing 10 nearest neighbors...
[INFO ] Neighbors computed.
[INFO ] Re-mapping indices...
[INFO ] Saving CSV data to 'distances_out.csv'.
[INFO ] Saving CSV data to 'neighbors_out.csv'.
[INFO ]
[INFO ] Execution parameters:
[INFO ]   distances_file: distances_out.csv
[INFO ]   help: false
[INFO ]   info: ""
[INFO ]   k: 10
[INFO ]   leaf_size: 20
[INFO ]   naive: false
[INFO ]   neighbors_file: neighbors_out.csv
[INFO ]   query_file: query_dataset.csv
[INFO ]   reference_file: reference_dataset.csv
[INFO ]   single_mode: false
[INFO ]   verbose: true
[INFO ]
[INFO ] Program timers:
[INFO ]   computing_neighbors: 0.000081s
[INFO ]   total_time: 0.062828s
[INFO ]   tree_building: 0.004949s

```

14.3.3 One dataset, 3 nearest neighbors, leaf size of 15 points

```

$ allknn -r dataset.csv -n neighbors_out.csv -d distances_out.csv -k 3 -l 15 -v
[INFO ] Loading 'dataset.csv' as CSV data.
[INFO ] Loaded reference data from 'dataset.csv'.
[INFO ] Building reference tree...
[INFO ] Trees built.
[INFO ] Computing 3 nearest neighbors...
[INFO ] Neighbors computed.
[INFO ] Re-mapping indices...
[INFO ] Saving CSV data to 'distances_out.csv'.
[INFO ] Saving CSV data to 'neighbors_out.csv'.
[INFO ]
[INFO ] Execution parameters:
[INFO ]   distances_file: distances_out.csv
[INFO ]   help: false
[INFO ]   info: ""
[INFO ]   k: 3
[INFO ]   leaf_size: 15
[INFO ]   naive: false
[INFO ]   neighbors_file: neighbors_out.csv
[INFO ]   query_file: ""
[INFO ]   reference_file: dataset.csv

```

```
[INFO ] single_mode: false
[INFO ] verbose: true
[INFO ]
[INFO ] Program timers:
[INFO ]   computing_neighbors: 0.105119s
[INFO ]   total_time: 0.145321s
[INFO ]   tree_building: 0.005690s
```

Further documentation on options should be found by using the `--help` option.

14.4 The 'AllkNN' class

The 'AllkNN' class is, specifically, a typedef of the more extensible NeighborSearch class, querying for nearest neighbors using the Euclidean distance.

```
typedef NeighborSearch<NearestNeighborSort, metric::EuclideanDistance>
    AllkNN;
```

Using the AllkNN class is particularly simple; first, the object must be constructed and given a dataset. Then, the method is run, and two matrices are returned: one which holds the indices of the nearest neighbors, and one which holds the distances of the nearest neighbors. These are of the same structure as the output `--neighbors_file` and `--distances_file` for the CLI interface (see above). A handful of examples of simple usage of the AllkNN class are given below.

14.4.1 5 nearest neighbors on a single dataset

```
#include <mlpack/methods/neighbor_search/neighbor_search.hpp>

using namespace mlpack::neighbor;

// Our dataset matrix, which is column-major.
extern arma::mat data;

AllkNN a(data);

// The matrices we will store output in.
arma::Mat<size_t> resultingNeighbors;
arma::mat resultingDistances;

a.Search(5, resultingNeighbors, resultingDistances);
```

The output of the search is stored in `resultingNeighbors` and `resultingDistances`.

14.4.2 10 nearest neighbors on a query and reference dataset

```
#include <mlpack/methods/neighbor_search/neighbor_search.hpp>

using namespace mlpack::neighbor;

// Our dataset matrices, which are column-major.
extern arma::mat queryData, referenceData;

AllkNN a(referenceData, queryData);

// The matrices we will store output in.
arma::Mat<size_t> resultingNeighbors;
arma::mat resultingDistances;

a.Search(10, resultingNeighbors, resultingDistances);
```

14.4.3 Naive (exhaustive) search for 6 nearest neighbors on one dataset

This example uses the $O(n^2)$ naive search (not the tree-based search).

```
#include <mlpack/methods/neighbor_search/neighbor_search.hpp>

using namespace mlpack::neighbor;

// Our dataset matrix, which is column-major.
extern arma::mat dataset;

AllkNN a(dataset, true);

// The matrices we will store output in.
arma::Mat<size_t> resultingNeighbors;
arma::mat resultingDistances;

a.Search(6, resultingNeighbors, resultingDistances);
```

Needless to say, naive search can be very slow...

14.5 The extensible 'NeighborSearch' class

The NeighborSearch class is very extensible, having the following template arguments:

```
template<
  typename SortPolicy = NearestNeighborSort,
  typename MetricType = mlpack::metric::EuclideanDistance,
  typename TreeType = mlpack::tree::BinarySpaceTree<bound::HRectBound<2>,
    QueryStat<SortPolicy> >
>
class NeighborSearch;
```

By choosing different components for each of these template classes, a very arbitrary neighbor searching object can be constructed.

14.5.1 SortPolicy policy class

The SortPolicy template parameter allows specification of how the NeighborSearch object will decide which points are to be searched for. The **mlpack::neighbor::NearestNeighborSort** (p. 395) class is a well-documented example. A custom SortPolicy class must implement the same methods which NearestNeighborSort does:

```
static size_t SortDistance(const arma::vec& list, double newDistance);

static bool IsBetter(const double value, const double ref);

template<typename TreeType>
static double BestNodeToNodeDistance(const TreeType* queryNode,
    const TreeType* referenceNode);

template<typename TreeType>
static double BestPointToNodeDistance(const arma::vec& queryPoint,
    const TreeType* referenceNode);

static const double WorstDistance();

static const double BestDistance();
```

The **mlpack::neighbor::FurthestNeighborSort** (p. 385) class is another implementation, which is used to create the 'AllkFN' typedef class, which finds the furthest neighbors, as opposed to the nearest neighbors.

14.5.2 MetricType policy class

The MetricType policy class allows the neighbor search to take place in any arbitrary metric space. The **mlpack::metric::LMetric** (p. 367) class is a good example implementation. A MetricType class must provide the following functions:

```
// Empty constructor is required.
MetricType();

// Compute the distance between two points.
template<typename VecType>
double Evaluate(const VecType& a, const VecType& b);
```

Internally, the NeighborSearch class keeps an instantiated MetricType class (which can be given in the constructor). This is useful for a metric like the Mahalanobis distance (**mlpack::metric::MahalanobisDistance** (p. 368)), which must store state (the covariance matrix). Therefore, you can write a non-static MetricType class and use it seamlessly with NeighborSearch.

14.5.3 TreeType policy class

The NeighborSearch class also allows a custom tree to be used. The standard MLPACK tree, **mlpack::tree::BinarySpaceTree** (p. 566), is also highly extensible in its own right, and its documentation should be consulted for more information. Currently, the NeighborSearch tree requires a tree which only has left and right children, and no points in nodes (only in leaves), but this support is planned to be extended.

A simple usage of the TreeType policy could be to use a different type of bound with the tree. For instance, you could use a ball bound instead of a rectangular bound:

```
// Construct a NeighborSearch object with ball bounds.
NeighborSearch<
    NearestNeighborSort,
    metric::EuclideanDistance,
    tree::BinarySpaceTree<bound::BallBound<2>,
                        QueryStat<SortPolicy> >
> neighborSearch(dataset);
```

It is important to note that the NeighborSearch class requires use of the QueryStat tree statistic to function properly. Therefore, if you write a custom tree, be sure it can accept the QueryStat type. See the **mlpack::tree::BinarySpaceTree** (p. 566) documentation for more information on tree statistics.

14.6 Further documentation

For further documentation on the NeighborSearch class, consult the **complete API documentation** (p. 399).

Chapter 15

RangeSearch tutorial (range_search)

15.1 Introduction

Range search is a simple machine learning task which aims to find all the neighbors of a point that fall into a certain range of distances. In this setting, we have a **query** and a **reference** dataset. Given a certain range, for each point in the **query** dataset, we wish to know all points in the **reference** dataset which have distances within that given range to the given query point.

Alternately, if the query and reference datasets are the same, the problem can be stated more simply: for each point in the dataset, we wish to know all points which have distance in the given range to that point.

mlpack provides:

- a **simple command-line executable** (p. 66) to run range search
- a **simple C++ interface** (p. 68) to perform range search
- a **generic, extensible, and powerful C++ class (RangeSearch)** (p. 69) for complex usage

15.2 Table of Contents

A list of all the sections this tutorial contains.

- **Introduction** (p. 65)
- **Table of Contents** (p. 65)
- **The 'range_search' command-line executable** (p. 66)
 - **One dataset, points with distance ≤ 0.01** (p. 66)
 - **Query and reference dataset, range [1.0, 1.5]** (p. 67)
 - **One dataset, range [4.1 4.2], leaf size of 15 points** (p. 67)
- **The 'RangeSearch' class** (p. 68)
 - **Distance less than 2.0 on a single dataset** (p. 68)
 - **Range [3.0, 4.0] on a query and reference dataset** (p. 69)
 - **Naive (exhaustive) search for distance greater than 5.0 on one dataset** (p. 69)

- The extensible 'RangeSearch' class (p. 69)
 - MetricType policy class (p. 69)
 - TreeType policy class (p. 70)
- Further documentation (p. 70)

15.3 The 'range_search' command-line executable

mlpack provides an executable, `range_search`, which can be used to perform range searches quickly and simply from the command-line. This program will perform the range search and place the resulting neighbor index list into one file and their corresponding distances into another file. These files are organized such that the first row corresponds to the neighbors (or distances) of the first query point, and the second row corresponds to the neighbors (or distances) of the second query point, and so forth. The neighbors of a specific point are not arranged in any specific order.

Because a range search may return different numbers of points (including zero), the output file is technically not a valid CSV and may not be loadable by other programs. Therefore, if you need the results in a certain format, it may be better to use the **C++ interface** (p. 68) to manually export the data in the preferred format.

Below are several examples of simple usage (and the resultant output). The '-v' option is used so that output is given. Further documentation on each individual option can be found by typing

```
$ range_search --help
```

15.3.1 One dataset, points with distance ≤ 0.01

```
$ range_search -r dataset.csv -n neighbors_out.csv -d distances_out.csv -M 0.01 -v
[INFO ] Loading 'dataset.csv' as CSV data.
[INFO ] Loaded reference data from 'dataset.csv'.
[INFO ] Building reference tree...
[INFO ] Trees built.
[INFO ] Computing neighbors within range [0, 0.01].
[INFO ] Number of pruned nodes during computation: 0.
[INFO ] Neighbors computed.
[INFO ] Re-mapping indices...
[INFO ]
[INFO ] Execution parameters:
[INFO ]   distances_file: distances_out.csv
[INFO ]   help: false
[INFO ]   info: ""
[INFO ]   leaf_size: 20
[INFO ]   max: 2.5
[INFO ]   min: 0
[INFO ]   naive: false
[INFO ]   neighbors_file: neighbors_out.csv
[INFO ]   query_file: ""
[INFO ]   reference_file: dataset.csv
[INFO ]   single_mode: false
[INFO ]   verbose: true
[INFO ]
[INFO ] Program timers:
[INFO ]   range_search/computing_neighbors: 1.564744s
[INFO ]   total_time: 3.841249s
[INFO ]   tree_building: 0.005112s
```

Convenient program timers are given for different parts of the calculation at the bottom of the output, as well as the parameters the simulation was run with. Now, if we look at the output files:

```
$ head neighbors_out.csv
344, 862
703

397, 277, 319, 443
840, 827
```

```

876, 732
569, 222, 563
437, 361, 97, 928
961, 419, 547, 695
113, 843, 634, 982, 689

$ head distances_out.csv
0.0058751, 0.00358331
0.00567406

0.000432393, 0.00577239, 0.00221909, 0.00841252
0.00501577, 0.00810424
0.00898339, 0.0032354
0.00945658, 0.00893871, 0.006213
0.00979697, 0.00490745, 0.00833828, 0.00902167
0.00957553, 0.00657434, 0.0028044, 0.00303588
0.00199936, 0.00843088, 0.00968861, 0.00159429, 0.00539645

```

We can see that points 344 and 862 are within distance 0.01 of point 0. We can also see that point 2 has no points within a distance of 0.01 – that line is empty.

15.3.2 Query and reference dataset, range [1.0, 1.5]

```

$ range_search -q query_dataset.csv -r reference_dataset.csv -n \
> neighbors_out.csv -d distances_out.csv -m 1.0 -M 1.5 -v
[INFO ] Loading 'dataset.csv' as CSV data.
[INFO ] Loaded reference data from 'dataset.csv'.
[INFO ] Building reference tree...
[INFO ] Loading 'dataset.csv' as CSV data.
[INFO ] Loaded query data from 'dataset.csv'.
[INFO ] Building query tree...
[INFO ] Tree built.
[INFO ] Computing neighbors within range [1, 1.5].
[INFO ] Number of pruned nodes during computation: 1110.
[INFO ] Neighbors computed.
[INFO ] Re-mapping indices...
[INFO ]
[INFO ] Execution parameters:
[INFO ]   distances_file: distances_out.csv
[INFO ]   help: false
[INFO ]   info: ""
[INFO ]   leaf_size: 20
[INFO ]   max: 1.5
[INFO ]   min: 1
[INFO ]   naive: false
[INFO ]   neighbors_file: neighbors_out.csv
[INFO ]   query_file: dataset.csv
[INFO ]   reference_file: dataset.csv
[INFO ]   single_mode: false
[INFO ]   verbose: true
[INFO ]
[INFO ] Program timers:
[INFO ]   range_search/computing_neighbors: 0.466848s
[INFO ]   total_time: 0.725183s
[INFO ]   tree_building: 0.004769s

```

15.3.3 One dataset, range [4.1 4.2], leaf size of 15 points

The **mlpack** implementation of range search is a dual-tree algorithm; when *k_d*-trees are used, the leaf size of the tree can be changed. Depending on the characteristics of the dataset, a larger or smaller leaf size can provide faster computation. The leaf size is modifiable through the command-line interface, as shown below.

```

$ range_search -r dataset.csv -n neighbors_out.csv -d distances_out.csv -m 4.1 \
> -M 4.2 -l 15 -v
[INFO ] Loading 'dataset.csv' as CSV data.
[INFO ] Loaded reference data from 'dataset.csv'.
[INFO ] Building reference tree...
[INFO ] Trees built.
[INFO ] Computing neighbors within range [4.1, 4.2].
[INFO ] Number of pruned nodes during computation: 1.
[INFO ] Neighbors computed.

```

```
[INFO ] Re-mapping indices...
[INFO ]
[INFO ] Execution parameters:
[INFO ]   distances_file: distances_out.csv
[INFO ]   help: false
[INFO ]   info: ""
[INFO ]   leaf_size: 20
[INFO ]   max: 4.2
[INFO ]   min: 4.1
[INFO ]   naive: false
[INFO ]   neighbors_file: neighbors_out.csv
[INFO ]   query_file: ""
[INFO ]   reference_file: dataset.csv
[INFO ]   single_mode: false
[INFO ]   verbose: true
[INFO ]
[INFO ] Program timers:
[INFO ]   range_search/computing_neighbors: 0.003857s
[INFO ]   total_time: 0.056154s
[INFO ]   tree_building: 0.004831s
```

Further documentation on options should be found by using the `--help` option.

15.4 The 'RangeSearch' class

The 'RangeSearch' class is an extensible template class which allows a high level of flexibility. However, all of the template arguments have default parameters, allowing a user to simply use 'RangeSearch<>' for simple usage without worrying about the exact necessary template parameters.

The class bears many similarities to the **NeighborSearch** (p. 59) class; usage generally consists of calling the constructor with one or two datasets, and then calling the 'Search()' method to perform the actual range search.

The 'Search()' method stores the results in two vector-of-vector objects. This is necessary because each query point may have a different number of neighbors in the specified distance range. The structure of those two objects is very similar to the output files `--neighbors_file` and `--distances_file` for the CLI interface (see above). A handful of examples of simple usage of the RangeSearch class are given below.

Using the AllkNN class is particularly simple; first, the object must be constructed and given a dataset. Then, the method is run, and two matrices are returned: one which holds the indices of the nearest neighbors, and one which holds the distances of the nearest neighbors. These are of the same structure as the output `--neighbors_file` and `--reference_file` for the CLI interface (see above). A handful of examples of simple usage of the AllkNN class are given below.

15.4.1 Distance less than 2.0 on a single dataset

```
#include <mlpack/methods/range_search/range_search.hpp>

using namespace mlpack::range;

// Our dataset matrix, which is column-major.
extern arma::mat data;

RangeSearch<> a(data);

// The vector-of-vector objects we will store output in.
std::vector<std::vector<size_t>> > resultingNeighbors;
std::vector<std::vector<double>> > resultingDistances;

// The range we will use.
math::Range r(0.0, 2.0); // [0.0, 2.0].

a.Search(r, resultingNeighbors, resultingDistances);
```

The output of the search is stored in `resultingNeighbors` and `resultingDistances`.

15.4.2 Range [3.0, 4.0] on a query and reference dataset

```
#include <mlpack/methods/range_search/range_search.hpp>

using namespace mlpack::range;

// Our dataset matrices, which are column-major.
extern arma::mat queryData, referenceData;

RangeSearch<> a(referenceData, queryData);

// The vector-of-vector objects we will store output in.
std::vector<std::vector<size_t> > resultingNeighbors;
std::vector<std::vector<double> > resultingDistances;

// The range we will use.
math::Range r(3.0, 4.0); // [3.0, 4.0].

a.Search(r, resultingNeighbors, resultingDistances);
```

15.4.3 Naive (exhaustive) search for distance greater than 5.0 on one dataset

This example uses the $O(n^2)$ naive search (not the tree-based search).

```
#include <mlpack/methods/range_search/range_search.hpp>

using namespace mlpack::range;

// Our dataset matrix, which is column-major.
extern arma::mat dataset;

// The 'true' option indicates that we will use naive calculation.
RangeSearch<> a(dataset, true);

// The vector-of-vector objects we will store output in.
std::vector<std::vector<size_t> > resultingNeighbors;
std::vector<std::vector<double> > resultingDistances;

// The range we will use. The upper bound is DBL_MAX.
math::Range r(5.0, DBL_MAX); // [5.0, inf).

a.Search(r, resultingNeighbors, resultingDistances);
```

Needless to say, naive search can be very slow...

15.5 The extensible 'RangeSearch' class

Similar to the **NeighborSearch** class (p. 59), the **RangeSearch** class is very extensible, having the following template arguments:

```
template<
  typename MetricType = mlpack::metric::EuclideanDistance,
  typename TreeType = mlpack::tree::BinarySpaceTree<bound::HRectBound<2>,
                                                    tree::EmptyStatistic>
>
class RangeSearch;
```

By choosing different components for each of these template classes, a very arbitrary range searching object can be constructed.

15.5.1 MetricType policy class

The **MetricType** policy class allows the range search to take place in any arbitrary metric space. The **mlpack::metric::LMetric** (p. 367) class is a good example implementation. A **MetricType** class must provide the following functions:

```
// Empty constructor is required.
MetricType();

// Compute the distance between two points.
template<typename VecType>
double Evaluate(const VecType& a, const VecType& b);
```

Internally, the `RangeSearch` class keeps an instantiated `MetricType` class (which can be given in the constructor). This is useful for a metric like the Mahalanobis distance (`mlpack::metric::MahalanobisDistance` (p. 368)), which must store state (the covariance matrix). Therefore, you can write a non-static `MetricType` class and use it seamlessly with `RangeSearch`.

15.5.2 TreeType policy class

The `RangeSearch` class also allows a custom tree to be used. The standard `mlpack` tree, `mlpack::tree::BinarySpaceTree` (p. 566), is also highly extensible in its own right, and its documentation should be consulted for more information.

A simple usage of the `TreeType` policy could be to use a different type of bound with the existing `mlpack::tree::BinarySpaceTree` (p. 566) class. For instance, you could use a ball bound instead of a rectangular bound:

```
// Construct a RangeSearch object with ball bounds.
RangeSearch<
    metric::EuclideanDistance,
    tree::BinarySpaceTree<bound::BallBound<2>,
                        EmptyStatistic>
> rangeSearch(dataset);
```

Unlike the `NeighborSearch` class (p. 59), the `RangeSearch` class does not make use of tree statistics; therefore, the `EmptyStatistic` class should be used for the `StatisticType` parameter of the `BinarySpaceTree` (but this is not technically necessary – `RangeSearch` simply makes no use of the tree statistic).

It is also possible to use a completely different type of tree. The example below shows the use of the `RangeSearch` class with the `mlpack::tree::CoverTree` (p. 603) class (which has the `EmptyStatistic` statistic type as a default, so we do not need to specify that).

```
// Construct a RangeSearch object that uses cover trees.
RangeSearch<tree::CoverTree<> > rangeSearch(dataset);
```

15.6 Further documentation

For further documentation on the `RangeSearch` class, consult the **complete API documentation** (p. 512).

Chapter 16

Tutorials

16.1 Introductory Tutorials

These tutorials introduce the basic concepts of working with MLPACK, aimed at developers who want to use and contribute to MLPACK but are not sure where to start.

- **Building MLPACK** (p. 6)
- **Matrices in MLPACK** (p. 13)
- **MLPACK Input and Output** (p. 9)
- **MLPACK Timers** (p. 17)
- **Simple Sample MLPACK Programs** (p. 15)

16.2 Method-specific Tutorials

These tutorials introduce the various methods MLPACK offers, aimed at users who simply want to use the methods MLPACK offers. These tutorials start with simple examples and progress to complex, extensible uses.

- **NeighborSearch tutorial (k-nearest-neighbors)** (p. 59)
- **Linear/ridge regression tutorial (linear_regression)** (p. 53)
- **RangeSearch tutorial (range_search)** (p. 65)
- **Density Estimation Tree (DET) tutorial** (p. 25)
- **K-Means tutorial (kmeans)** (p. 43)
- **Fast max-kernel search tutorial (fastmks)** (p. 35)
- **EMST Tutorial** (p. 31)
- **Alternating Matrix Factorization tutorial.** (p. 21)

Chapter 17

Bug List

Class `mlpack::CLI` (p. 184)

The **COUNTER** variable is used in most cases to guarantee a unique global identifier for options declared using the `PARAM_*`() macros. However, not all compilers have this support—most notably, `gcc < 4.3`. In that case, the **LINE** macro is used as an attempt to get a unique global identifier, but collisions are still possible, and they produce bizarre error messages. See <http://mlpack.org/trac/ticket/74> for more information.

`globalScope>` Member `PARAM_DOUBLE` (p. 761) (ID, DESC, ALIAS, DEF)

The **COUNTER** variable is used in most cases to guarantee a unique global identifier for options declared using the `PARAM_*`() macros. However, not all compilers have this support—most notably, `gcc < 4.3`. In that case, the **LINE** macro is used as an attempt to get a unique global identifier, but collisions are still possible, and they produce bizarre error messages. See <http://mlpack.org/trac/ticket/74> for more information.

`globalScope>` Member `PARAM_DOUBLE_REQ` (p. 761) (ID, DESC, ALIAS)

The **COUNTER** variable is used in most cases to guarantee a unique global identifier for options declared using the `PARAM_*`() macros. However, not all compilers have this support—most notably, `gcc < 4.3`. In that case, the **LINE** macro is used as an attempt to get a unique global identifier, but collisions are still possible, and they produce bizarre error messages. See <http://mlpack.org/trac/ticket/74> for more information.

`globalScope>` Member `PARAM_FLAG` (p. 762) (ID, DESC, ALIAS)

The **COUNTER** variable is used in most cases to guarantee a unique global identifier for options declared using the `PARAM_*`() macros. However, not all compilers have this support—most notably, `gcc < 4.3`. In that case, the **LINE** macro is used as an attempt to get a unique global identifier, but collisions are still possible, and they produce bizarre error messages. See <http://mlpack.org/trac/ticket/74> for more information.

`globalScope>` Member `PARAM_FLOAT` (p. 762) (ID, DESC, ALIAS, DEF)

The **COUNTER** variable is used in most cases to guarantee a unique global identifier for options declared using the `PARAM_*`() macros. However, not all compilers have this support—most notably, `gcc < 4.3`. In that case, the **LINE** macro is used as an attempt to get a unique global identifier, but collisions are still possible, and they produce bizarre error messages. See <http://mlpack.org/trac/ticket/74> for more information.

`globalScope>` Member `PARAM_FLOAT_REQ` (p. 762) (ID, DESC, ALIAS)

The **COUNTER** variable is used in most cases to guarantee a unique global identifier for options declared using the `PARAM_*`() macros. However, not all compilers have this support—most notably, `gcc < 4.3`. In that case, the **LINE** macro is used as an attempt to get a unique global identifier, but collisions are still possible, and they produce bizarre error messages. See <http://mlpack.org/trac/ticket/74> for more information.

`globalScope>` Member `PARAM_INT` (p. 763) (ID, DESC, ALIAS, DEF)

The **COUNTER** variable is used in most cases to guarantee a unique global identifier for options declared using the `PARAM_*`() macros. However, not all compilers have this support—most notably, `gcc < 4.3`. In that case, the **LINE** macro is used as an attempt to get a unique global identifier, but collisions are still possible, and they produce bizarre error messages. See <http://mlpack.org/trac/ticket/74> for more information.

globalScope> Member PARAM_INT_REQ (p. 763) (ID, DESC, ALIAS)

The **COUNTER** variable is used in most cases to guarantee a unique global identifier for options declared using the **PARAM_***() macros. However, not all compilers have this support—most notably, gcc < 4.3. In that case, the **LINE** macro is used as an attempt to get a unique global identifier, but collisions are still possible, and they produce bizarre error messages. See <http://mlpack.org/trac/ticket/74> for more information.

globalScope> Member PARAM_STRING (p. 764) (ID, DESC, ALIAS, DEF)

The **COUNTER** variable is used in most cases to guarantee a unique global identifier for options declared using the **PARAM_***() macros. However, not all compilers have this support—most notably, gcc < 4.3. In that case, the **LINE** macro is used as an attempt to get a unique global identifier, but collisions are still possible, and they produce bizarre error messages. See <http://mlpack.org/trac/ticket/74> for more information.

globalScope> Member PARAM_STRING_REQ (p. 764) (ID, DESC, ALIAS)

The **COUNTER** variable is used in most cases to guarantee a unique global identifier for options declared using the **PARAM_***() macros. However, not all compilers have this support—most notably, gcc < 4.3. In that case, the **LINE** macro is used as an attempt to get a unique global identifier, but collisions are still possible, and they produce bizarre error messages. See <http://mlpack.org/trac/ticket/74> for more information.

globalScope> Member PARAM_VECTOR (p. 765) (T, ID, DESC, ALIAS)

The **COUNTER** variable is used in most cases to guarantee a unique global identifier for options declared using the **PARAM_***() macros. However, not all compilers have this support—most notably, gcc < 4.3. In that case, the **LINE** macro is used as an attempt to get a unique global identifier, but collisions are still possible, and they produce bizarre error messages. See <http://mlpack.org/trac/ticket/74> for more information.

globalScope> Member PARAM_VECTOR_REQ (p. 765) (T, ID, DESC, ALIAS)

The **COUNTER** variable is used in most cases to guarantee a unique global identifier for options declared using the **PARAM_***() macros. However, not all compilers have this support—most notably, gcc < 4.3. In that case, the **LINE** macro is used as an attempt to get a unique global identifier, but collisions are still possible, and they produce bizarre error messages. See <http://mlpack.org/trac/ticket/74> for more information.

Chapter 18

Namespace Index

18.1 Namespace List

Here is a list of all namespaces with brief descriptions:

mlpack	Linear algebra utility functions, generally performed on matrices or vectors	89
mlpack::amf	Alternating Matrix Factorization	90
mlpack::bound	92
mlpack::cf	Collaborative filtering	92
mlpack::data	Functions to load and save matrices	93
mlpack::decision_stump	95
mlpack::det	Density Estimation Trees	95
mlpack::distribution	Probability distributions	97
mlpack::emst	Euclidean Minimum Spanning Trees	97
mlpack::fastmks	Fast max-kernel search	98
mlpack::gmm	Gaussian Mixture Models	98
mlpack::hmm	Hidden Markov Models	100
mlpack::kernel	Kernel functions	101
mlpack::kmeans	K-Means clustering	103
mlpack::kpca	103
mlpack::lcc	104
mlpack::math	Miscellaneous math routines	104
mlpack::metric	109
mlpack::mvu	109
mlpack::naive_bayes	The Naive Bayes Classifier	109

mlpack::nca	
Neighborhood Components Analysis	110
mlpack::neighbor	
Neighbor-search routines	110
mlpack::nn	113
mlpack::optimization	113
mlpack::optimization::test	114
mlpack::pca	114
mlpack::perceptron	115
mlpack::radical	115
mlpack::range	
Range-search routines	115
mlpack::regression	
Regression methods	116
mlpack::sparse_coding	116
mlpack::svd	116
mlpack::tree	
Trees and tree-building procedures	116
mlpack::util	117

Chapter 19

Class Index

19.1 Class List

Here are the classes, structs, unions and interfaces with brief descriptions:

IsVector< VecType >	
If value == true, then VecType is some sort of Armadillo vector or subview	119
IsVector< arma::Col< eT > >	120
IsVector< arma::Row< eT > >	120
IsVector< arma::SpCol< eT > >	120
IsVector< arma::SpRow< eT > >	121
IsVector< arma::SpSubview< eT > >	121
IsVector< arma::subview_col< eT > >	122
IsVector< arma::subview_row< eT > >	122
mlpack::amf::AMF< TerminationPolicyType, InitializationRuleType, UpdateRuleType >	
This class implements AMF (p. 123) (alternating matrix factorization) on the given matrix V	123
mlpack::amf::AverageInitialization	
This initialization rule initializes matrix W and H to root of average of V with uniform noise	127
mlpack::amf::CompleteIncrementalTermination< TerminationPolicy >	128
mlpack::amf::IncompleteIncrementalTermination< TerminationPolicy >	130
mlpack::amf::NMFALSUpdate	
This class implements a method titled 'Alternating Least Squares' described in the paper 'Positive Matrix Factorization: A Non-negative Factor Model with Optimal Utilization of Error Estimates of Data Values' by P Paatero and U Tapper	132
mlpack::amf::NMFMultiplicativeDistanceUpdate	
The multiplicative distance update rules for matrices W and H	134
mlpack::amf::NMFMultiplicativeDivergenceUpdate	
This follows a method described in the paper 'Algorithms for Non-negative Matrix Factorization' by D	136
mlpack::amf::RandomAcolInitialization< p >	
This class initializes the W matrix of the AMF (p. 123) algorithm by averaging p randomly chosen columns of V	137
mlpack::amf::RandomInitialization	138
mlpack::amf::SimpleResidueTermination	
This class implements a simple residue-based termination policy	139
mlpack::amf::SimpleToleranceTermination< MatType >	
This class implements residue tolerance termination policy	143
mlpack::amf::SVDBatchLearning	
This class implements SVD batch learning with momentum	148
mlpack::amf::SVDCompleteIncrementalLearning< MatType >	151

mlpack::amf::SVDCompleteIncrementalLearning < arma::sp_mat >	154
mlpack::amf::SVDIncompleteIncrementalLearning	157
mlpack::amf::ValidationRMSETermination < MatType >	159
mlpack::bound::BallBound < VecType, TMetricType >	
Ball bound encloses a set of points at a specific distance (radius) from a specific point (center)	163
mlpack::bound::HRectBound < Power, TakeRoot >	
Hyper-rectangle bound for an L-metric	170
mlpack::cf::CF < FactorizerType >	
This class implements Collaborative Filtering (CF (p. 177))	177
mlpack::CLI	
Parses the command line for parameters and holds user-specified parameters	184
mlpack::decision_stump::DecisionStump < MatType >	
This class implements a decision stump	196
mlpack::det::DTree	
A density estimation tree is similar to both a decision tree and a space partitioning tree (like a kd-tree)	202
mlpack::distribution::DiscreteDistribution	
A discrete distribution where the only observations are discrete observations	214
mlpack::distribution::GaussianDistribution	
A single multivariate Gaussian distribution	218
mlpack::distribution::LaplaceDistribution	
The multivariate Laplace distribution centered at 0 has pdf	221
mlpack::emst::DTBRules < MetricType, TreeType >	225
mlpack::emst::DTBStat	
A statistic for use with MLPACK trees, which stores the upper bound on distance to nearest neighbors and the component which this node belongs to	231
mlpack::emst::DualTreeBoruvka < MetricType, TreeType >	
Performs the MST calculation using the Dual-Tree Boruvka algorithm, using any type of tree	236
mlpack::emst::DualTreeBoruvka < MetricType, TreeType >::SortEdgesHelper	
For sorting the edge list after the computation	242
mlpack::emst::EdgePair	
An edge pair is simply two indices and a distance	243
mlpack::emst::UnionFind	
A Union-Find data structure	245
mlpack::fastmks::FastMKS < KernelType, TreeType >	
An implementation of fast exact max-kernel search	247
mlpack::fastmks::FastMKSRules < KernelType, TreeType >	
The base case and pruning rules for FastMKS (p. 247) (fast max-kernel search)	254
mlpack::fastmks::FastMKSSStat	
The statistic used in trees with FastMKS (p. 247)	260
mlpack::gmm::DiagonalConstraint	
Force a covariance matrix to be diagonal	264
mlpack::gmm::EigenvalueRatioConstraint	
Given a vector of eigenvalue ratios, ensure that the covariance matrix always has those eigenvalue ratios	264
mlpack::gmm::EMFit < InitialClusteringType, CovarianceConstraintPolicy >	
This class contains methods which can fit a GMM (p. 271) to observations using the EM algorithm	266
mlpack::gmm::GMM < FittingType >	
A Gaussian Mixture Model (GMM (p. 271))	271
mlpack::gmm::NoConstraint	
This class enforces no constraint on the covariance matrix	283
mlpack::gmm::PositiveDefiniteConstraint	
Given a covariance matrix, force the matrix to be positive definite	283
mlpack::hmm::HMM < Distribution >	
A class that represents a Hidden Markov Model with an arbitrary type of emission distribution	284

mlpack::kernel::CosineDistance	
The cosine distance (or cosine similarity)	293
mlpack::kernel::EpanechnikovKernel	
The Epanechnikov kernel, defined as	294
mlpack::kernel::ExampleKernel	
An example kernel function	296
mlpack::kernel::GaussianKernel	
The standard Gaussian kernel	299
mlpack::kernel::HyperbolicTangentKernel	
Hyperbolic tangent kernel	304
mlpack::kernel::KernelTraits< KernelType >	
This is a template class that can provide information about various kernels	306
mlpack::kernel::KernelTraits< CosineDistance >	
Kernel traits for the cosine distance	307
mlpack::kernel::KernelTraits< EpanechnikovKernel >	
Kernel traits for the Epanechnikov kernel	308
mlpack::kernel::KernelTraits< GaussianKernel >	
Kernel traits for the Gaussian kernel	308
mlpack::kernel::KernelTraits< LaplacianKernel >	
Kernel traits of the Laplacian kernel	309
mlpack::kernel::KernelTraits< SphericalKernel >	
Kernel traits for the spherical kernel	310
mlpack::kernel::KernelTraits< TriangularKernel >	
Kernel traits for the triangular kernel	310
mlpack::kernel::KMeansSelection< ClusteringType >	311
mlpack::kernel::LaplacianKernel	
The standard Laplacian kernel	312
mlpack::kernel::LinearKernel	
The simple linear kernel (dot product)	315
mlpack::kernel::NystroemMethod< KernelType, PointSelectionPolicy >	317
mlpack::kernel::OrderedSelection	319
mlpack::kernel::PolynomialKernel	
The simple polynomial kernel	319
mlpack::kernel::PSpectrumStringKernel	
The p-spectrum string kernel	322
mlpack::kernel::RandomSelection	325
mlpack::kernel::SphericalKernel	327
mlpack::kernel::TriangularKernel	
The trivially simple triangular kernel, defined by	329
mlpack::kmeans::AllowEmptyClusters	
Policy which allows K-Means to create empty clusters without any error being reported	332
mlpack::kmeans::KMeans< MetricType, InitialPartitionPolicy, EmptyClusterPolicy >	
This class implements K-Means clustering	333
mlpack::kmeans::MaxVarianceNewCluster	
When an empty cluster is detected, this class takes the point furthest from the centroid of the cluster with maximum variance as a new cluster	340
mlpack::kmeans::RandomPartition	
A very simple partitioner which partitions the data randomly into the number of desired clusters	342
mlpack::kmeans::RefinedStart	
A refined approach for choosing initial points for k-means clustering	343
mlpack::kpca::KernelPCA< KernelType, KernelRule >	
This class performs kernel principal components analysis (Kernel PCA), for a given kernel	345
mlpack::kpca::NaiveKernelRule< KernelType >	350
mlpack::kpca::NystroemKernelRule< KernelType, PointSelectionPolicy >	351

mlpack::lcc::LocalCoordinateCoding < DictionaryInitializer >	
An implementation of Local Coordinate Coding (LCC) that codes data which approximately lives on a manifold using a variation of l1-norm regularized sparse coding; in LCC, the penalty on the absolute value of each point's coefficient for each atom is weighted by the squared distance of that point to that atom	352
mlpack::Log	
Provides a convenient way to give formatted output	357
mlpack::math::Range	
Simple real-valued range	359
mlpack::metric::IPMetric < KernelType >	365
mlpack::metric::LMetric < Power , TakeRoot >	
The L _p metric for arbitrary integer p, with an option to take the root	367
mlpack::metric::MahalanobisDistance < TakeRoot >	
The Mahalanobis distance, which is essentially a stretched Euclidean distance	368
mlpack::mvu::MVU	
Meant to provide a good abstraction for users	372
mlpack::naive_bayes::NaiveBayesClassifier < MatType >	
The simple Naive Bayes classifier	373
mlpack::nca::NCA < MetricType , OptimizerType >	
An implementation of Neighborhood Components Analysis, both a linear dimensionality reduction technique and a distance learning technique	376
mlpack::nca::SoftmaxErrorFunction < MetricType >	
The "softmax" stochastic neighbor assignment probability function	380
mlpack::neighbor::FurthestNeighborSort	
This class implements the necessary methods for the SortPolicy template parameter of the NeighborSearch (p. 399) class	385
mlpack::neighbor::LSHSearch < SortPolicy >	
The LSHSearch (p. 388) class – This class builds a hash on the reference set and uses this hash to compute the distance-approximate nearest-neighbors of the given queries	388
mlpack::neighbor::NearestNeighborSort	
This class implements the necessary methods for the SortPolicy template parameter of the NeighborSearch (p. 399) class	395
mlpack::neighbor::NeighborSearch < SortPolicy , MetricType , TreeType >	
The NeighborSearch (p. 399) class is a template class for performing distance-based neighbor searches	399
mlpack::neighbor::NeighborSearchRules < SortPolicy , MetricType , TreeType >	405
mlpack::neighbor::NeighborSearchStat < SortPolicy >	
Extra data for each node in the tree	412
mlpack::neighbor::NeighborSearchTraversalInfo < TreeType >	
Traversal information for NeighborSearch (p. 399)	416
mlpack::neighbor::RASearchRules < SortPolicy , MetricType , TreeType >	419
mlpack::nn::SparseAutoencoder < OptimizerType >	
A sparse autoencoder is a neural network whose aim to learn compressed representations of the data, typically for dimensionality reduction, with a constraint on the activity of the neurons in the network	430
mlpack::nn::SparseAutoencoderFunction	
This is a class for the sparse autoencoder objective function	436
mlpack::optimization::AugLagrangian < LagrangianFunction >	
The AugLagrangian (p. 441) class implements the Augmented Lagrangian method of optimization	441
mlpack::optimization::AugLagrangianFunction < LagrangianFunction >	
This is a utility class used by AugLagrangian (p. 441), meant to wrap a LagrangianFunction into a function usable by a simple optimizer like L-BFGS	446

mlpack::optimization::AugLagrangianTestFunction	
This function is taken from "Practical Mathematical Optimization" (Snyman), section 5.3.8 ("Application of the Augmented Lagrangian Method")	450
mlpack::optimization::ExponentialSchedule	
The exponential cooling schedule cools the temperature T at every step according to the equation	452
mlpack::optimization::GockenbachFunction	
This function is taken from M	453
mlpack::optimization::L_BFGS< FunctionType >	
The generic L-BFGS optimizer, which uses a back-tracking line search algorithm to minimize a function	455
mlpack::optimization::LovaszThetaSDP	
This function is the Lovasz-Theta semidefinite program, as implemented in the following paper:	466
mlpack::optimization::LRSDP	
LRSDP (p. 468) is the implementation of Monteiro and Burer's formulation of low-rank semidefinite programs (LR-SDP)	468
mlpack::optimization::LRSDPFunction	
The objective function that LRSDP (p. 468) is trying to optimize	472
mlpack::optimization::SA< FunctionType, CoolingScheduleType >	
Simulated Annealing is an stochastic optimization algorithm which is able to deliver near-optimal results quickly without knowing the gradient of the function being optimized	476
mlpack::optimization::SGD< DecomposableFunctionType >	
Stochastic Gradient Descent is a technique for minimizing a function which can be expressed as a sum of other functions	485
mlpack::optimization::test::GeneralizedRosenbrockFunction	
The Generalized Rosenbrock function in n dimensions, defined by $f(x) = \sum_{i=1}^{n-1} (f(i)(x)) f_{i+1}(x)$ $= 100 * (x_{i+1}^2 - x_{i+1})^2 + (1 - x_{i+1})^2$ $x_0 = [-1.2, 1, -1.2, 1, \dots]$	491
mlpack::optimization::test::RosenbrockFunction	
The Rosenbrock function, defined by $f(x) = f_1(x) + f_2(x)$ $f_1(x) = 100 (x_2 - x_1^2)^2$ $f_2(x) = (1 - x_1)^2$ $x_0 = [-1.2, 1]$	492
mlpack::optimization::test::RosenbrockWoodFunction	
The Generalized Rosenbrock function in 4 dimensions with the Wood Function in four dimensions	493
mlpack::optimization::test::SGDTestFunction	
Very, very simple test function which is the composite of three other functions	495
mlpack::optimization::test::WoodFunction	
The Wood function, defined by $f(x) = f_1(x) + f_2(x) + f_3(x) + f_4(x) + f_5(x) + f_6(x)$ $f_1(x) = 100 (x_2 - x_1^2)^2$ $f_2(x) = (1 - x_1)^2$ $f_3(x) = 90 (x_4 - x_3^2)^2$ $f_4(x) = (1 - x_3)^2$ $f_5(x) = 10 (x_2 + x_4 - 2)^2$ $f_6(x) = (1 / 10) (x_2 - x_4)^2$ $x_0 = [-3, -1, -3, -1]$	496
mlpack::ParamData	
Aids in the extensibility of CLI (p. 184) by focusing potential changes into one structure	497
mlpack::pca::PCA	
This class implements principal components analysis (PCA (p. 498))	498
mlpack::perceptron::Perceptron< LearnPolicy, WeightInitializationPolicy, MatType >	
This class implements a simple perceptron (i.e., a single layer neural network)	501
mlpack::perceptron::RandomInitialization	
This class is used to initialize weights for the weightVectors matrix in a random manner	504
mlpack::perceptron::SimpleWeightUpdate	505
mlpack::perceptron::ZeroInitialization	
This class is used to initialize the matrix weightVectors to zero	505
mlpack::radical::Radical	
An implementation of RADICAL, an algorithm for independent component analysis (ICA)	506
mlpack::range::RangeSearch< MetricType, TreeType >	
The RangeSearch (p. 512) class is a template class for performing range searches	512
mlpack::range::RangeSearchRules< MetricType, TreeType >	518

mlpack::range::RangeSearchStat	
Statistic class for RangeSearch (p. 512), to be set to the StatisticType of the tree type that range search is being performed with	523
mlpack::regression::LARS	
An implementation of LARS (p. 525), a stage-wise homotopy-based algorithm for l1-regularized linear regression (LASSO) and l1+l2 regularized linear regression (Elastic Net)	525
mlpack::regression::LinearRegression	
A simple linear regression algorithm using ordinary least squares	532
mlpack::regression::LogisticRegression< OptimizerType >	536
mlpack::regression::LogisticRegressionFunction	
The log-likelihood function for the logistic regression objective function	541
mlpack::sparse_coding::DataDependentRandomInitializer	
A data-dependent random dictionary initializer for SparseCoding (p. 547)	545
mlpack::sparse_coding::NothingInitializer	
A DictionaryInitializer for SparseCoding (p. 547) which does not initialize anything; it is useful for when the dictionary is already known and will be set with SparseCoding::Dictionary() (p. 550)	546
mlpack::sparse_coding::RandomInitializer	
A DictionaryInitializer for use with the SparseCoding (p. 547) class	546
mlpack::sparse_coding::SparseCoding< DictionaryInitializer >	
An implementation of Sparse Coding with Dictionary Learning that achieves sparsity via an l1-norm regularizer on the codes (LASSO) or an (l1+l2)-norm regularizer on the codes (the Elastic Net)	547
mlpack::svd::QUIC_SVD	553
mlpack::svd::RegularizedSVD< OptimizerType >	555
mlpack::svd::RegularizedSVDFunction	557
mlpack::Timer	
The timer class provides a way for MLPACK methods to be timed	562
mlpack::Timers	564
mlpack::tree::BinarySpaceTree< BoundType, StatisticType, MatType, SplitType >	
A binary space partitioning tree, such as a KD-tree or a ball tree	566
mlpack::tree::BinarySpaceTree< BoundType, StatisticType, MatType, SplitType >::DualTreeTraverser< RuleType >	
A dual-tree traverser for binary space trees; see <code>dual_tree_traverser.hpp</code>	586
mlpack::tree::BinarySpaceTree< BoundType, StatisticType, MatType, SplitType >::SingleTreeTraverser< RuleType >	
A single-tree traverser for binary space trees; see <code>single_tree_traverser.hpp</code> for implementation	590
mlpack::tree::CompareCosineNode	593
mlpack::tree::CosineTree	594
mlpack::tree::CoverTree< MetricType, RootPointPolicy, StatisticType >	
A cover tree is a tree specifically designed to speed up nearest-neighbor computation in high-dimensional spaces	603
mlpack::tree::CoverTree< MetricType, RootPointPolicy, StatisticType >::DualTreeTraverser< RuleType >	
A dual-tree cover tree traverser; see <code>dual_tree_traverser.hpp</code>	621
mlpack::tree::CoverTree< MetricType, RootPointPolicy, StatisticType >::DualTreeTraverser< RuleType >::DualCoverTreeMapEntry	
Struct used for traversal	625
mlpack::tree::CoverTree< MetricType, RootPointPolicy, StatisticType >::SingleTreeTraverser< RuleType >	
A single-tree cover tree traverser; see <code>single_tree_traverser.hpp</code> for implementation	627
mlpack::tree::EmptyStatistic	
Empty statistic if you are not interested in storing statistics in your tree	629
mlpack::tree::ExampleTree< MetricType, StatisticType, MatType >	
This is not an actual space tree but instead an example tree that exists to show and document all the functions that mlpack trees must implement	630

mlpack::tree::FirstPointIsRoot	
This class is meant to be used as a choice for the policy class RootPointPolicy of the CoverTree (p. 603) class	638
mlpack::tree::MeanSplit< BoundType, MatType >	
A binary space partitioning tree node is split into its left and right child	638
mlpack::tree::MRKDStatistic	
Statistic for multi-resolution kd-trees	640
mlpack::tree::TreeTraits< TreeType >	
The TreeTraits (p. 645) class provides compile-time information on the characteristics of a given tree type	645
mlpack::tree::TreeTraits< BinarySpaceTree< BoundType, StatisticType, MatType > >	
This is a specialization of the TreeType class to the BinarySpaceTree (p. 566) tree type	647
mlpack::tree::TreeTraits< CoverTree< MetricType, RootPointPolicy, StatisticType > >	
The specialization of the TreeTraits (p. 645) class for the CoverTree (p. 603) tree type	649
mlpack::util::CLIDeleter	
Extremely simple class whose only job is to delete the existing CLI (p. 184) object at the end of execution	650
mlpack::util::NullOutputStream	
Used for Log::Debug (p. 359) when not compiled with debugging symbols	651
mlpack::util::Option< N >	
A static object whose constructor registers a parameter with the CLI (p. 184) class	654
mlpack::util::PrefixedOutputStream	
Allows us to output to an ostream with a prefix at the beginning of each line, in the same way we would output to cout or cerr	655
mlpack::util::ProgramDoc	
A static object whose constructor registers program documentation with the CLI (p. 184) class	662
mlpack::util::SaveRestoreUtility	663
RASearch< SortPolicy, MetricType, TreeType >	
The RASearch (p. 666) class: This class provides a generic manner to perform rank-approximate search via random-sampling	666
TraversallInfo< TreeType >	
The TraversallInfo (p. 673) class holds traversal information which is used in dual-tree (and single-tree) traversals	673

Chapter 20

File Index

20.1 File List

Here is a list of all files with brief descriptions:

doc/guide/ build.hpp	677
doc/guide/ iodoc.hpp	677
doc/guide/ matrices.hpp	677
doc/guide/ sample.hpp	677
doc/guide/ timer.hpp	677
doc/guide/ version.hpp	776
src/mlpack/ core.hpp	703
src/mlpack/ prereqs.hpp	
The core includes that mlpack expects; standard C++ includes and Armadillo	856
src/mlpack/core/data/ load.hpp	703
src/mlpack/core/data/ normalize_labels.hpp	704
src/mlpack/core/data/ save.hpp	705
src/mlpack/core/dists/ discrete_distribution.hpp	706
src/mlpack/core/dists/ gaussian_distribution.hpp	707
src/mlpack/core/dists/ laplace_distribution.hpp	708
src/mlpack/core/kernels/ cosine_distance.hpp	708
src/mlpack/core/kernels/ epanechnikov_kernel.hpp	709
src/mlpack/core/kernels/ example_kernel.hpp	710
src/mlpack/core/kernels/ gaussian_kernel.hpp	711
src/mlpack/core/kernels/ hyperbolic_tangent_kernel.hpp	712
src/mlpack/core/kernels/ kernel_traits.hpp	712
src/mlpack/core/kernels/ laplacian_kernel.hpp	713
src/mlpack/core/kernels/ linear_kernel.hpp	714
src/mlpack/core/kernels/ polynomial_kernel.hpp	715
src/mlpack/core/kernels/ pspectrum_string_kernel.hpp	716
src/mlpack/core/kernels/ spherical_kernel.hpp	717
src/mlpack/core/kernels/ triangular_kernel.hpp	718
src/mlpack/core/math/ clamp.hpp	
Miscellaneous math clamping routines	718
src/mlpack/core/math/ lin_alg.hpp	720
src/mlpack/core/math/ random.hpp	
Miscellaneous math random-related routines	721
src/mlpack/core/math/ range.hpp	
Definition of the Range class, which represents a simple range with a lower and upper bound	722

src/mlpack/core/math/ round.hpp	723
src/mlpack/core/metrics/ ip_metric.hpp	724
src/mlpack/core/metrics/ lmetric.hpp	725
src/mlpack/core/metrics/ mahalanobis_distance.hpp	726
src/mlpack/core/optimizers/aug_lagrangian/ aug_lagrangian.hpp	726
src/mlpack/core/optimizers/aug_lagrangian/ aug_lagrangian_function.hpp	728
src/mlpack/core/optimizers/aug_lagrangian/ aug_lagrangian_test_functions.hpp	729
src/mlpack/core/optimizers/lbfgs/ lbfgs.hpp	730
src/mlpack/core/optimizers/lbfgs/ test_functions.hpp	731
src/mlpack/core/optimizers/lrsdp/ lrsdp.hpp	732
src/mlpack/core/optimizers/lrsdp/ lrsdp_function.hpp	732
src/mlpack/core/optimizers/sa/ exponential_schedule.hpp	734
src/mlpack/core/optimizers/sa/ sa.hpp	735
src/mlpack/core/optimizers/sgd/ sgd.hpp	735
src/mlpack/core/optimizers/sgd/ test_function.hpp	737
src/mlpack/core/tree/ ballbound.hpp	
Bounds that are useful for binary space partitioning trees	737
src/mlpack/core/tree/ binary_space_tree.hpp	739
src/mlpack/core/tree/ bounds.hpp	
Bounds that are useful for binary space partitioning trees	747
src/mlpack/core/tree/ cover_tree.hpp	750
src/mlpack/core/tree/ example_tree.hpp	752
src/mlpack/core/tree/ hrectbound.hpp	
Bounds that are useful for binary space partitioning trees	752
src/mlpack/core/tree/ mrkd_statistic.hpp	753
src/mlpack/core/tree/ rectangle_tree.hpp	754
src/mlpack/core/tree/ statistic.hpp	
Definition of the policy type for the statistic class	755
src/mlpack/core/tree/ traversal_info.hpp	755
src/mlpack/core/tree/ tree_traits.hpp	757
src/mlpack/core/tree/binary_space_tree/ binary_space_tree.hpp	738
src/mlpack/core/tree/binary_space_tree/ dual_tree_traverser.hpp	740
src/mlpack/core/tree/binary_space_tree/ mean_split.hpp	742
src/mlpack/core/tree/binary_space_tree/ single_tree_traverser.hpp	743
src/mlpack/core/tree/binary_space_tree/ traits.hpp	745
src/mlpack/core/tree/cosine_tree/ cosine_tree.hpp	748
src/mlpack/core/tree/cover_tree/ cover_tree.hpp	749
src/mlpack/core/tree/cover_tree/ dual_tree_traverser.hpp	741
src/mlpack/core/tree/cover_tree/ first_point_is_root.hpp	750
src/mlpack/core/tree/cover_tree/ single_tree_traverser.hpp	744
src/mlpack/core/tree/cover_tree/ traits.hpp	746
src/mlpack/core/util/ arma_traits.hpp	758
src/mlpack/core/util/ cli.hpp	759
src/mlpack/core/util/ cli_deleter.hpp	767
src/mlpack/core/util/ log.hpp	767
src/mlpack/core/util/ nulloutstream.hpp	768
src/mlpack/core/util/ option.hpp	769
src/mlpack/core/util/ prefixedoutstream.hpp	770
src/mlpack/core/util/ save_restore_utility.hpp	771
src/mlpack/core/util/ sfinae_utility.hpp	772
src/mlpack/core/util/ string_util.hpp	773
src/mlpack/core/util/ timers.hpp	774
src/mlpack/core/util/ version.hpp	775
src/mlpack/methods/amf/ amf.hpp	776

src/mlpack/methods/amf/init_rules/average_init.hpp	777
src/mlpack/methods/amf/init_rules/random_acol_init.hpp	777
src/mlpack/methods/amf/init_rules/random_init.hpp	778
src/mlpack/methods/amf/termination_policies/complete_incremental_termination.hpp	780
src/mlpack/methods/amf/termination_policies/incomplete_incremental_termination.hpp	781
src/mlpack/methods/amf/termination_policies/simple_residue_termination.hpp	781
src/mlpack/methods/amf/termination_policies/simple_tolerance_termination.hpp	783
src/mlpack/methods/amf/termination_policies/validation_RMSE_termination.hpp	784
src/mlpack/methods/amf/update_rules/nmf_als.hpp	784
src/mlpack/methods/amf/update_rules/nmf_mult_dist.hpp	785
src/mlpack/methods/amf/update_rules/nmf_mult_div.hpp	787
src/mlpack/methods/amf/update_rules/svd_batch_learning.hpp	787
src/mlpack/methods/amf/update_rules/svd_complete_incremental_learning.hpp	788
src/mlpack/methods/amf/update_rules/svd_incomplete_incremental_learning.hpp	788
src/mlpack/methods/cf/cf.hpp	789
src/mlpack/methods/decision_stump/decision_stump.hpp	790
src/mlpack/methods/det/dt_utils.hpp	790
src/mlpack/methods/det/dtree.hpp	791
src/mlpack/methods/emst/dtb.hpp	792
src/mlpack/methods/emst/dtb_rules.hpp	793
src/mlpack/methods/emst/dtb_stat.hpp	794
src/mlpack/methods/emst/edge_pair.hpp	795
src/mlpack/methods/emst/union_find.hpp	796
src/mlpack/methods/fastmks/fastmks.hpp	797
src/mlpack/methods/fastmks/fastmks_rules.hpp	798
src/mlpack/methods/fastmks/fastmks_stat.hpp	798
src/mlpack/methods/gmm/diagonal_constraint.hpp	799
src/mlpack/methods/gmm/eigenvalue_ratio_constraint.hpp	800
src/mlpack/methods/gmm/em_fit.hpp	801
src/mlpack/methods/gmm/gmm.hpp	802
src/mlpack/methods/gmm/no_constraint.hpp	803
src/mlpack/methods/gmm/phi.hpp	803
src/mlpack/methods/gmm/positive_definite_constraint.hpp	804
src/mlpack/methods/hmm/hmm.hpp	806
src/mlpack/methods/hmm/hmm_util.hpp	807
src/mlpack/methods/kernel_pca/kernel_pca.hpp	807
src/mlpack/methods/kernel_pca/kernel_rules/naive_method.hpp	808
src/mlpack/methods/kernel_pca/kernel_rules/nystroem_method.hpp	809
src/mlpack/methods/kmeans/allow_empty_clusters.hpp	810
src/mlpack/methods/kmeans/kmeans.hpp	811
src/mlpack/methods/kmeans/max_variance_new_cluster.hpp	813
src/mlpack/methods/kmeans/random_partition.hpp	814
src/mlpack/methods/kmeans/refined_start.hpp	816
src/mlpack/methods/lars/lars.hpp	817
src/mlpack/methods/linear_regression/linear_regression.hpp	818
src/mlpack/methods/local_coordinate_coding/lcc.hpp	819
src/mlpack/methods/logistic_regression/logistic_regression.hpp	819
src/mlpack/methods/logistic_regression/logistic_regression_function.hpp	820
src/mlpack/methods/lsh/lsh_search.hpp	821
src/mlpack/methods/mvu/mvu.hpp	822
src/mlpack/methods/naive_bayes/naive_bayes_classifier.hpp	823
src/mlpack/methods/nca/nca.hpp	823
src/mlpack/methods/nca/nca_softmax_error_function.hpp	824
src/mlpack/methods/neighbor_search/neighbor_search.hpp	825

src/mlpack/methods/neighbor_search/ neighbor_search_rules.hpp	826
src/mlpack/methods/neighbor_search/ neighbor_search_stat.hpp	827
src/mlpack/methods/neighbor_search/ ns_traversal_info.hpp	828
src/mlpack/methods/neighbor_search/ typedef.hpp	831
src/mlpack/methods/neighbor_search/ unmap.hpp	833
src/mlpack/methods/neighbor_search/sort_policies/ furthest_neighbor_sort.hpp	829
src/mlpack/methods/neighbor_search/sort_policies/ nearest_neighbor_sort.hpp	830
src/mlpack/methods/nystroem_method/ kmeans_selection.hpp	834
src/mlpack/methods/nystroem_method/ nystroem_method.hpp	810
src/mlpack/methods/nystroem_method/ ordered_selection.hpp	835
src/mlpack/methods/nystroem_method/ random_selection.hpp	836
src/mlpack/methods/pca/ pca.hpp	836
src/mlpack/methods/perceptron/ perceptron.hpp	839
src/mlpack/methods/perceptron/initialization_methods/ random_init.hpp	779
src/mlpack/methods/perceptron/initialization_methods/ zero_init.hpp	837
src/mlpack/methods/perceptron/learning_policies/ simple_weight_update.hpp	838
src/mlpack/methods/quic_svd/ quic_svd.hpp	840
src/mlpack/methods/radical/ radical.hpp	841
src/mlpack/methods/range_search/ range_search.hpp	841
src/mlpack/methods/range_search/ range_search_rules.hpp	842
src/mlpack/methods/range_search/ range_search_stat.hpp	843
src/mlpack/methods/rann/ ra_query_stat.hpp	844
src/mlpack/methods/rann/ ra_search.hpp	845
src/mlpack/methods/rann/ ra_search_rules.hpp	846
src/mlpack/methods/rann/ ra_typedef.hpp	847
src/mlpack/methods/regularized_svd/ regularized_svd.hpp	849
src/mlpack/methods/regularized_svd/ regularized_svd_function.hpp	849
src/mlpack/methods/sparse_autoencoder/ sparse_autoencoder.hpp	850
src/mlpack/methods/sparse_autoencoder/ sparse_autoencoder_function.hpp	851
src/mlpack/methods/sparse_coding/ data_dependent_random_initializer.hpp	852
src/mlpack/methods/sparse_coding/ nothing_initializer.hpp	853
src/mlpack/methods/sparse_coding/ random_initializer.hpp	854
src/mlpack/methods/sparse_coding/ sparse_coding.hpp	855
src/mlpack/tests/ old_boost_test_definitions.hpp	857

Chapter 21

Namespace Documentation

21.1 mlpack Namespace Reference

Linear algebra utility functions, generally performed on matrices or vectors.

Namespaces

- **amf**
Alternating Matrix Factorization.
- **bound**
- **cf**
Collaborative filtering.
- **data**
Functions to load and save matrices.
- **decision_stump**
- **det**
Density Estimation Trees.
- **distribution**
Probability distributions.
- **emst**
Euclidean Minimum Spanning Trees.
- **fastmks**
Fast max-kernel search.
- **gmm**
Gaussian Mixture Models.
- **hmm**
Hidden Markov Models.
- **kernel**
Kernel functions.
- **kmeans**
K-Means clustering.
- **kpca**
- **lcc**
- **math**

Miscellaneous math routines.

- **metric**
- **mvu**
- **naive_bayes**

The Naive Bayes Classifier.

- **nca**

Neighborhood Components Analysis.

- **neighbor**

Neighbor-search routines.

- **nn**
- **optimization**
- **pca**
- **perceptron**
- **radical**
- **range**

Range-search routines.

- **regression**

Regression methods.

- **sparse_coding**
- **svd**
- **tree**

Trees and tree-building procedures.

- **util**

Classes

- class **CLI**

Parses the command line for parameters and holds user-specified parameters.

- class **Log**

Provides a convenient way to give formatted output.

- struct **ParamData**

*Aids in the extensibility of **CLI** (p. 184) by focusing potential changes into one structure.*

- class **Timer**

The timer class provides a way for MLPACK methods to be timed.

- class **Timers**

21.1.1 Detailed Description

Linear algebra utility functions, generally performed on matrices or vectors.

This class is used to update the weightVectors matrix according to the simple update rule as discussed by Rosenblatt:

if a vector x has been incorrectly classified by a weight w , then $w = w - x$ and $w' = w' + x$

where w' is the weight vector which correctly classifies x .

21.2 mlpack::amf Namespace Reference

Alternating Matrix Factorization.

Classes

- class **AMF**

*This class implements **AMF** (p. 123) (alternating matrix factorization) on the given matrix V .*

- class **AverageInitialization**

This initialization rule initializes matrix W and H to root of average of V with uniform noise.

- class **CompleteIncrementalTermination**
- class **IncompleteIncrementalTermination**
- class **NMFALSUpdate**

This class implements a method titled 'Alternating Least Squares' described in the paper 'Positive Matrix Factorization: A Non-negative Factor Model with Optimal Utilization of Error Estimates of Data Values' by P Paatero and U Tapper.

- class **NMFMultiplicativeDistanceUpdate**

The multiplicative distance update rules for matrices W and H .

- class **NMFMultiplicativeDivergenceUpdate**

This follows a method described in the paper 'Algorithms for Non-negative Matrix Factorization' by D.

- class **RandomAcolInitialization**

*This class initializes the W matrix of the **AMF** (p. 123) algorithm by averaging p randomly chosen columns of V .*

- class **RandomInitialization**
- class **SimpleResidueTermination**

This class implements a simple residue-based termination policy.

- class **SimpleToleranceTermination**

This class implements residue tolerance termination policy.

- class **SVDBatchLearning**

This class implements SVD batch learning with momentum.

- class **SVDCompleteIncrementalLearning**
- class **SVDCompleteIncrementalLearning**< arma::sp_mat >
- class **SVDIncompleteIncrementalLearning**
- class **ValidationRMSETermination**

Functions

- template<>
void **SVDBatchLearning::HUpdate**< arma::sp_mat > (const arma::sp_mat & V , const arma::mat & W , arma::mat & H)
- template<>
void **SVDBatchLearning::WUpdate**< arma::sp_mat > (const arma::sp_mat & V , arma::mat & W , const arma::mat & H)

TODO : Merge this template specialized function for sparse matrix using common row_col_iterator.
- template<>
void **SVDIncompleteIncrementalLearning::HUpdate**< arma::sp_mat > (const arma::sp_mat & V , const arma::mat & W , arma::mat & H)
- template<>
void **SVDIncompleteIncrementalLearning::WUpdate**< arma::sp_mat > (const arma::sp_mat & V , arma::mat & W , const arma::mat & H)

21.2.1 Detailed Description

Alternating Matrix Factorization.

21.2.2 Function Documentation

21.2.2.1 `template<> void mlpack::amf::SVDBatchLearning::HUpdate< arma::sp_mat > (const arma::sp_mat & V, const arma::mat & W, arma::mat & H) [inline]`

Definition at line 189 of file `svd_batch_learning.hpp`.

21.2.2.2 `template<> void mlpack::amf::SVDBatchLearning::WUpdate< arma::sp_mat > (const arma::sp_mat & V, arma::mat & W, const arma::mat & H) [inline]`

TODO : Merge this template specialized function for sparse matrix using common `row_col_iterator`.

WUpdate function specialization for sparse matrix

Definition at line 158 of file `svd_batch_learning.hpp`.

21.2.2.3 `template<> void mlpack::amf::SVDIncompleteIncrementalLearning::HUpdate< arma::sp_mat > (const arma::sp_mat & V, const arma::mat & W, arma::mat & H) [inline]`

Definition at line 127 of file `svd_incomplete_incremental_learning.hpp`.

21.2.2.4 `template<> void mlpack::amf::SVDIncompleteIncrementalLearning::WUpdate< arma::sp_mat > (const arma::sp_mat & V, arma::mat & W, const arma::mat & H) [inline]`

Definition at line 106 of file `svd_incomplete_incremental_learning.hpp`.

21.3 mlpack::bound Namespace Reference

Classes

- class **BallBound**
Ball bound encloses a set of points at a specific distance (radius) from a specific point (center).
- class **HRectBound**
Hyper-rectangle bound for an L-metric.

21.4 mlpack::cf Namespace Reference

Collaborative filtering.

Classes

- class **CF**
*This class implements Collaborative Filtering (**CF** (p. 177)).*

21.4.1 Detailed Description

Collaborative filtering.

21.5 mlpack::data Namespace Reference

Functions to load and save matrices.

Functions

- `template<typename eT >`
`bool Load (const std::string &filename, arma::Mat< eT > &matrix, bool fatal=false, bool transpose=true)`
Loads a matrix from file, guessing the filetype from the extension.
- `template<typename eT >`
`void NormalizeLabels (const arma::Col< eT > &labelsIn, arma::Col< size_t > &labels, arma::Col< eT > &mapping)`
Given a set of labels of a particular datatype, convert them to unsigned labels in the range [0, n) where n is the number of different labels.
- `template<typename eT >`
`void RevertLabels (const arma::Col< size_t > &labels, const arma::Col< eT > &mapping, arma::Col< eT > &labelsOut)`
Given a set of labels that have been mapped to the range [0, n), map them back to the original labels given by the 'mapping' vector.
- `template<typename eT >`
`bool Save (const std::string &filename, const arma::Mat< eT > &matrix, bool fatal=false, bool transpose=true)`
Saves a matrix to file, guessing the filetype from the extension.

21.5.1 Detailed Description

Functions to load and save matrices.

21.5.2 Function Documentation

21.5.2.1 `template<typename eT > bool mlpack::data::Load (const std::string & filename, arma::Mat< eT > & matrix, bool fatal = false, bool transpose = true)`

Loads a matrix from file, guessing the filetype from the extension.

This will transpose the matrix at load time. If the filetype cannot be determined, an error will be given.

The supported types of files are the same as found in Armadillo:

- CSV (csv_ascii), denoted by .csv, or optionally .txt
- ASCII (raw_ascii), denoted by .txt
- Armadillo ASCII (arma_ascii), also denoted by .txt
- PGM (pgm_binary), denoted by .pgm
- PPM (ppm_binary), denoted by .ppm
- Raw binary (raw_binary), denoted by .bin
- Armadillo binary (arma_binary), denoted by .bin
- HDF5, denoted by .hdf, .hdf5, .h5, or .he5

If the file extension is not one of those types, an error will be given. This is preferable to Armadillo's default behavior of loading an unknown filetype as `raw_binary`, which can have very confusing effects.

If the parameter 'fatal' is set to true, the program will exit with an error if the matrix does not load successfully. The parameter 'transpose' controls whether or not the matrix is transposed after loading. In most cases, because data is generally stored in a row-major format and MLPACK requires column-major matrices, this should be left at its default value of 'true'.

Parameters

<i>filename</i>	Name of file to load.
<i>matrix</i>	Matrix to load contents of file into.
<i>fatal</i>	If an error should be reported as fatal (default false).
<i>transpose</i>	If true, transpose the matrix after loading.

Returns

Boolean value indicating success or failure of load.

21.5.2.2 `template<typename eT> void mlpack::data::NormalizeLabels (const arma::Col< eT > & labelsIn, arma::Col< size_t > & labels, arma::Col< eT > & mapping)`

Given a set of labels of a particular datatype, convert them to unsigned labels in the range [0, n) where n is the number of different labels.

Also, a reverse mapping from the new label to the old value is stored in the 'mapping' vector.

Parameters

<i>labelsIn</i>	Input labels of arbitrary datatype.
<i>labels</i>	Vector that unsigned labels will be stored in.
<i>mapping</i>	Reverse mapping to convert new labels back to old labels.

21.5.2.3 `template<typename eT> void mlpack::data::RevertLabels (const arma::Col< size_t > & labels, const arma::Col< eT > & mapping, arma::Col< eT > & labelsOut)`

Given a set of labels that have been mapped to the range [0, n), map them back to the original labels given by the 'mapping' vector.

Parameters

<i>labels</i>	Set of normalized labels to convert.
<i>mapping</i>	Mapping to use to convert labels.
<i>labelsOut</i>	Vector to store new labels in.

21.5.2.4 `template<typename eT> bool mlpack::data::Save (const std::string & filename, const arma::Mat< eT > & matrix, bool fatal = false, bool transpose = true)`

Saves a matrix to file, guessing the filetype from the extension.

This will transpose the matrix at save time. If the filetype cannot be determined, an error will be given.

The supported types of files are the same as found in Armadillo:

- CSV (csv_ascii), denoted by .csv, or optionally .txt

- ASCII (raw_ascii), denoted by .txt
- Armadillo ASCII (arma_ascii), also denoted by .txt
- PGM (pgm_binary), denoted by .pgm
- PPM (ppm_binary), denoted by .ppm
- Raw binary (raw_binary), denoted by .bin
- Armadillo binary (arma_binary), denoted by .bin
- HDF5 (hdf5_binary), denoted by .hdf5, .hdf, .h5, or .he5

If the file extension is not one of those types, an error will be given. If the 'fatal' parameter is set to true, an error will cause the program to exit. If the 'transpose' parameter is set to true, the matrix will be transposed before saving. Generally, because MLPACK stores matrices in a column-major format and most datasets are stored on disk as row-major, this parameter should be left at its default value of 'true'.

Parameters

<i>filename</i>	Name of file to save to.
<i>matrix</i>	Matrix to save into file.
<i>fatal</i>	If an error should be reported as fatal (default false).
<i>transpose</i>	If true, transpose the matrix before saving.

Returns

Boolean value indicating success or failure of save.

21.6 mlpack::decision_stump Namespace Reference

Classes

- class **DecisionStump**
This class implements a decision stump.

21.7 mlpack::det Namespace Reference

Density Estimation Trees.

Classes

- class **DTree**
A density estimation tree is similar to both a decision tree and a space partitioning tree (like a kd-tree).

Functions

- void **PrintLeafMembership** (**DTree** *dtree, const arma::mat &data, const arma::Mat< size_t > &labels, const size_t numClasses, const std::string leafClassMembershipFile="")
Print the membership of leaves of a density estimation tree given the labels and number of classes.

- void **PrintVariableImportance** (const **DTree** *dtree, const std::string viFile="")
Print the variable importance of each dimension of a density estimation tree.
- **DTree** * **Trainer** (arma::mat &dataset, const size_t folds, const bool useVolumeReg=false, const size_t maxLeafSize=10, const size_t minLeafSize=5, const std::string unprunedTreeOutput="")
Train the optimal decision tree using cross-validation with the given number of folds.

21.7.1 Detailed Description

Density Estimation Trees.

21.7.2 Function Documentation

21.7.2.1 void mlpack::det::PrintLeafMembership (**DTree** * dtree, const arma::mat & data, const arma::Mat< size_t > & labels, const size_t numClasses, const std::string leafClassMembershipFile = " ")

Print the membership of leaves of a density estimation tree given the labels and number of classes.

Optionally, pass the name of a file to print this information to (otherwise stdout is used).

Parameters

<i>dtree</i>	Tree to print membership of.
<i>data</i>	Dataset tree is built upon.
<i>labels</i>	Class labels of dataset.
<i>numClasses</i>	Number of classes in dataset.
<i>leafClassMembershipFile</i>	Name of file to print to (optional).

21.7.2.2 void mlpack::det::PrintVariableImportance (const **DTree** * dtree, const std::string viFile = " ")

Print the variable importance of each dimension of a density estimation tree.

Optionally, pass the name of a file to print this information to (otherwise stdout is used).

Parameters

<i>dtree</i>	Density tree to use.
<i>viFile</i>	Name of file to print to (optional).

21.7.2.3 **DTree*** mlpack::det::Trainer (arma::mat & dataset, const size_t folds, const bool useVolumeReg = false, const size_t maxLeafSize = 10, const size_t minLeafSize = 5, const std::string unprunedTreeOutput = " ")

Train the optimal decision tree using cross-validation with the given number of folds.

Optionally, give a filename to print the unpruned tree to. This initializes a tree on the heap, so you are responsible for deleting it.

Parameters

<i>dataset</i>	Dataset for the tree to use.
<i>folds</i>	Number of folds to use for cross-validation.
<i>useVolumeReg</i>	If true, use volume regularization.
<i>maxLeafSize</i>	Maximum number of points allowed in a leaf.
<i>minLeafSize</i>	Minimum number of points allowed in a leaf.
<i>unprunedTree↔ Output</i>	Filename to print unpruned tree to (optional).

21.8 mlpack::distribution Namespace Reference

Probability distributions.

Classes

- class **DiscreteDistribution**
A discrete distribution where the only observations are discrete observations.
- class **GaussianDistribution**
A single multivariate Gaussian distribution.
- class **LaplaceDistribution**
The multivariate Laplace distribution centered at 0 has pdf.

21.8.1 Detailed Description

Probability distributions.

21.9 mlpack::emst Namespace Reference

Euclidean Minimum Spanning Trees.

Classes

- class **DTBRules**
- class **DTBStat**
A statistic for use with MLPACK trees, which stores the upper bound on distance to nearest neighbors and the component which this node belongs to.
- class **DualTreeBoruvka**
Performs the MST calculation using the Dual-Tree Boruvka algorithm, using any type of tree.
- class **EdgePair**
An edge pair is simply two indices and a distance.
- class **UnionFind**
A Union-Find data structure.

21.9.1 Detailed Description

Euclidean Minimum Spanning Trees.

21.10 mlpack::fastmks Namespace Reference

Fast max-kernel search.

Classes

- class **FastMKS**
An implementation of fast exact max-kernel search.
- class **FastMKSRules**
*The base case and pruning rules for **FastMKS** (p. 247) (fast max-kernel search).*
- class **FastMKSStat**
*The statistic used in trees with **FastMKS** (p. 247).*

21.10.1 Detailed Description

Fast max-kernel search.

21.11 mlpack::gmm Namespace Reference

Gaussian Mixture Models.

Classes

- class **DiagonalConstraint**
Force a covariance matrix to be diagonal.
- class **EigenvalueRatioConstraint**
Given a vector of eigenvalue ratios, ensure that the covariance matrix always has those eigenvalue ratios.
- class **EMFit**
*This class contains methods which can fit a **GMM** (p. 271) to observations using the EM algorithm.*
- class **GMM**
*A Gaussian Mixture Model (**GMM** (p. 271)).*
- class **NoConstraint**
This class enforces no constraint on the covariance matrix.
- class **PositiveDefiniteConstraint**
Given a covariance matrix, force the matrix to be positive definite.

Functions

- double **phi** (const double x, const double mean, const double var)
Calculates the univariate Gaussian probability density function.
- double **phi** (const arma::vec &x, const arma::vec &mean, const arma::mat &cov)
Calculates the multivariate Gaussian probability density function.
- double **phi** (const arma::vec &x, const arma::vec &mean, const arma::mat &cov, const std::vector< arma::mat > &d_cov, arma::vec &g_mean, arma::vec &g_cov)

Calculates the multivariate Gaussian probability density function and also the gradients with respect to the mean and the variance.

- void **phi** (const arma::mat &x, const arma::vec &mean, const arma::mat &cov, arma::vec &probabilities)

Calculates the multivariate Gaussian probability density function for each data point (column) in the given matrix, with respect to the given mean and variance.

21.11.1 Detailed Description

Gaussian Mixture Models.

21.11.2 Function Documentation

21.11.2.1 double mlpack::gmm::phi (const double x, const double mean, const double var) [inline]

Calculates the univariate Gaussian probability density function.

Example use:

```
double x, mean, var;
....
double f = phi(x, mean, var);
```

Parameters

<i>x</i>	Observation.
<i>mean</i>	Mean of univariate Gaussian.
<i>var</i>	Variance of univariate Gaussian.

Returns

Probability of x being observed from the given univariate Gaussian.

Definition at line 38 of file phi.hpp.

References M_PI.

Referenced by mlpack::distribution::GaussianDistribution::Probability().

21.11.2.2 double mlpack::gmm::phi (const arma::vec &x, const arma::vec &mean, const arma::mat &cov) [inline]

Calculates the multivariate Gaussian probability density function.

Example use:

```
extern arma::vec x, mean;
extern arma::mat cov;
....
double f = phi(x, mean, cov);
```

Parameters

<i>x</i>	Observation.
<i>mean</i>	Mean of multivariate Gaussian.
<i>cov</i>	Covariance of multivariate Gaussian.

Returns

Probability of *x* being observed from the given multivariate Gaussian.

Definition at line 60 of file phi.hpp.

References M_PI.

```
21.11.2.3 double mlpack::gmm::phi ( const arma::vec & x, const arma::vec & mean, const arma::mat & cov, const std::vector<
      arma::mat > & d_cov, arma::vec & g_mean, arma::vec & g_cov ) [inline]
```

Calculates the multivariate Gaussian probability density function and also the gradients with respect to the mean and the variance.

Example use:

```
extern arma::vec x, mean, g_mean, g_cov;
std::vector<arma::mat> d_cov; // the dSigma
....
double f = phi(x, mean, cov, d_cov, &g_mean, &g_cov);
```

Definition at line 86 of file phi.hpp.

References M_PI.

```
21.11.2.4 void mlpack::gmm::phi ( const arma::mat & x, const arma::vec & mean, const arma::mat & cov, arma::vec & probabilities
      ) [inline]
```

Calculates the multivariate Gaussian probability density function for each data point (column) in the given matrix, with respect to the given mean and variance.

Parameters

<i>x</i>	List of observations.
<i>mean</i>	Mean of multivariate Gaussian.
<i>cov</i>	Covariance of multivariate Gaussian.
<i>probabilities</i>	Output probabilities for each input observation.

Definition at line 130 of file phi.hpp.

References M_PI.

21.12 mlpack::hmm Namespace Reference

Hidden Markov Models.

Classes

- class **HMM**

A class that represents a Hidden Markov Model with an arbitrary type of emission distribution.

Functions

- `template<typename Distribution >`
`void LoadHMM (HMM< Distribution > &hmm, util::SaveRestoreUtility &sr)`
*Load an **HMM** (p. 284) from file.*
- `template<typename Distribution >`
`void SaveHMM (const HMM< Distribution > &hmm, util::SaveRestoreUtility &sr)`
*Save an **HMM** (p. 284) to file.*

21.12.1 Detailed Description

Hidden Markov Models.

21.12.2 Function Documentation

21.12.2.1 `template<typename Distribution > void mlpack::hmm::LoadHMM (HMM< Distribution > & hmm,`
`util::SaveRestoreUtility & sr)`

Load an **HMM** (p. 284) from file.

This only works for GMMs, DiscreteDistributions, and GaussianDistributions.

Template Parameters

<i>Distribution</i>	Distribution type of HMM (p. 284).
---------------------	---

Parameters

<i>sr</i>	SaveRestoreUtility to use.
-----------	----------------------------

21.12.2.2 `template<typename Distribution > void mlpack::hmm::SaveHMM (const HMM< Distribution > & hmm,`
`util::SaveRestoreUtility & sr)`

Save an **HMM** (p. 284) to file.

This only works for GMMs, DiscreteDistributions, and GaussianDistributions.

Template Parameters

<i>Distribution</i>	Distribution type of HMM (p. 284).
---------------------	---

Parameters

<i>sr</i>	SaveRestoreUtility to use.
-----------	----------------------------

21.13 mlpack::kernel Namespace Reference

Kernel functions.

Classes

- class **CosineDistance**
The cosine distance (or cosine similarity).
- class **EpanechnikovKernel**
The Epanechnikov kernel, defined as.
- class **ExampleKernel**
An example kernel function.
- class **GaussianKernel**
The standard Gaussian kernel.
- class **HyperbolicTangentKernel**
Hyperbolic tangent kernel.
- class **KernelTraits**
This is a template class that can provide information about various kernels.
- class **KernelTraits**< **CosineDistance** >
Kernel traits for the cosine distance.
- class **KernelTraits**< **EpanechnikovKernel** >
Kernel traits for the Epanechnikov kernel.
- class **KernelTraits**< **GaussianKernel** >
Kernel traits for the Gaussian kernel.
- class **KernelTraits**< **LaplacianKernel** >
Kernel traits of the Laplacian kernel.
- class **KernelTraits**< **SphericalKernel** >
Kernel traits for the spherical kernel.
- class **KernelTraits**< **TriangularKernel** >
Kernel traits for the triangular kernel.
- class **KMeansSelection**
- class **LaplacianKernel**
The standard Laplacian kernel.
- class **LinearKernel**
The simple linear kernel (dot product).
- class **NystroemMethod**
- class **OrderedSelection**
- class **PolynomialKernel**
The simple polynomial kernel.
- class **PSpectrumStringKernel**
The p-spectrum string kernel.
- class **RandomSelection**
- class **SphericalKernel**
- class **TriangularKernel**
The trivially simple triangular kernel, defined by.

21.13.1 Detailed Description

Kernel functions.

This namespace contains kernel functions, which evaluate some kernel function $K(x, y)$ for some arbitrary vectors x and y of the same dimension. The single restriction on the function $K(x, y)$ is that it must satisfy Mercer's condition:

$$\int \int K(x, y) g(x) g(y) dx dy \geq 0$$

for all square integrable functions $g(x)$.

The kernels in this namespace all implement the same methods as the **ExampleKernel** (p. 296) class. Any additional custom kernels should implement all the methods that class implements; in addition, any method using a kernel should rely on any arbitrary kernel function class having a default constructor and a function

```
double Evaluate(arma::vec&, arma::vec&);
```

21.14 mlpack::kmeans Namespace Reference

K-Means clustering.

Classes

- class **AllowEmptyClusters**
Policy which allows K-Means to create empty clusters without any error being reported.
- class **KMeans**
This class implements K-Means clustering.
- class **MaxVarianceNewCluster**
When an empty cluster is detected, this class takes the point furthest from the centroid of the cluster with maximum variance as a new cluster.
- class **RandomPartition**
A very simple partitioner which partitions the data randomly into the number of desired clusters.
- class **RefinedStart**
A refined approach for choosing initial points for k-means clustering.

21.14.1 Detailed Description

K-Means clustering.

21.15 mlpack::kpca Namespace Reference

Classes

- class **KernelPCA**
This class performs kernel principal components analysis (Kernel PCA), for a given kernel.
- class **NaiveKernelRule**
- class **NystroemKernelRule**

21.16 mlpack::lcc Namespace Reference

Classes

- class **LocalCoordinateCoding**

An implementation of Local Coordinate Coding (LCC) that codes data which approximately lives on a manifold using a variation of l1-norm regularized sparse coding; in LCC, the penalty on the absolute value of each point's coefficient for each atom is weighted by the squared distance of that point to that atom.

21.17 mlpack::math Namespace Reference

Miscellaneous math routines.

Classes

- class **Range**

Simple real-valued range.

Functions

- void **Center** (const arma::mat &x, arma::mat &xCentered)
Creates a centered matrix, where centering is done by subtracting the sum over the columns (a column vector) from each column of the matrix.
- double **ClampNonNegative** (const double d)
Forces a number to be non-negative, turning negative numbers into zero.
- double **ClampNonPositive** (const double d)
Forces a number to be non-positive, turning positive numbers into zero.
- double **ClampRange** (double value, const double rangeMin, const double rangeMax)
Clamp a number between a particular range.
- void **Orthogonalize** (const arma::mat &x, arma::mat &W)
Orthogonalize x and return the result in W, using eigendecomposition.
- void **Orthogonalize** (arma::mat &x)
Orthogonalize x in-place.
- int **RandInt** (const int hiExclusive)
Generates a uniform random integer.
- int **RandInt** (const int lo, const int hiExclusive)
Generates a uniform random integer.
- double **RandNormal** ()
Generates a normally distributed random number with mean 0 and variance 1.
- double **RandNormal** (const double mean, const double variance)
Generates a normally distributed random number with specified mean and variance.
- double **Random** ()
Generates a uniform random number between 0 and 1.
- double **Random** (const double lo, const double hi)
Generates a uniform random number in the specified range.
- void **RandomSeed** (const size_t seed)

Set the random seed used by the random functions (**Random()** (p. 107) and **RandInt()** (p. 106)).

- void **RandVector** (arma::vec &v)

Overwrites a dimension- N vector to a random vector on the unit sphere in R^N .

- void **RemoveRows** (const arma::mat &input, const std::vector< size_t > &rowsToRemove, arma::mat &output)

Remove a certain set of rows in a matrix while copying to a second matrix.

- void **VectorPower** (arma::vec &vec, const double power)

Auxiliary function to raise vector elements to a specific power.

- void **WhitenUsingEig** (const arma::mat &x, arma::mat &xWhitened, arma::mat &whiteningMatrix)

Whitens a matrix using the eigendecomposition of the covariance matrix.

- void **WhitenUsingSVD** (const arma::mat &x, arma::mat &xWhitened, arma::mat &whiteningMatrix)

Whitens a matrix using the singular value decomposition of the covariance matrix.

Variables

- boost::mt19937 **randGen**
- boost::normal_distribution **randNormalDist**
- boost::uniform_01< boost::mt19937, double > **randUniformDist**

21.17.1 Detailed Description

Miscellaneous math routines.

21.17.2 Function Documentation

21.17.2.1 void mlpack::math::Center (const arma::mat & x, arma::mat & xCentered)

Creates a centered matrix, where centering is done by subtracting the sum over the columns (a column vector) from each column of the matrix.

Parameters

x	Input matrix
$xCentered$	Matrix to write centered output into

21.17.2.2 double mlpack::math::ClampNonNegative (const double d) [inline]

Forces a number to be non-negative, turning negative numbers into zero.

Avoids branching costs (this is a measurable improvement).

Parameters

d	Double to clamp.
-----	------------------

Returns

0 if $d < 0$, d otherwise.

Definition at line 30 of file clamp.hpp.

Referenced by ClampRange().

21.17.2.3 double mlpack::math::ClampNonPositive (const double *d*) [inline]

Forces a number to be non-positive, turning positive numbers into zero.

Avoids branching costs (this is a measurable improvement).

Parameters

<i>d</i>	Double to clamp.
0	if $d > 0$, d otherwise.

Definition at line 42 of file clamp.hpp.

Referenced by ClampRange().

21.17.2.4 double mlpack::math::ClampRange (double *value*, const double *rangeMin*, const double *rangeMax*) [inline]

Clamp a number between a particular range.

Parameters

<i>value</i>	The number to clamp.
<i>rangeMin</i>	The first of the range.
<i>rangeMax</i>	The last of the range.

Returns

$\max(\text{rangeMin}, \min(\text{rangeMax}, d))$.

Definition at line 55 of file clamp.hpp.

References ClampNonNegative(), and ClampNonPositive().

21.17.2.5 void mlpack::math::Orthogonalize (const arma::mat & *x*, arma::mat & *W*)

Orthogonalize *x* and return the result in *W*, using eigendecomposition.

We will be using the formula $W = x(x^T x)^{-0.5}$.

21.17.2.6 void mlpack::math::Orthogonalize (arma::mat & *x*)

Orthogonalize *x* in-place.

This could be sped up by a custom implementation.

21.17.2.7 int mlpack::math::RandInt (const int *hiExclusive*) [inline]

Generates a uniform random integer.

Definition at line 98 of file random.hpp.

References randUniformDist.

Referenced by mlpack::sparse_coding::DataDependentRandomInitializer::Initialize(), mlpack::amf::RandomAcolInitialization< p >::Initialize(), and mlpack::kernel::RandomSelection::Select().

21.17.2.8 `int mpack::math::RandInt (const int lo, const int hiExclusive) [inline]`

Generates a uniform random integer.

Definition at line 112 of file random.hpp.

References randUniformDist.

21.17.2.9 `double mpack::math::RandNormal () [inline]`

Generates a normally distributed random number with mean 0 and variance 1.

Definition at line 129 of file random.hpp.

References randNormalDist.

21.17.2.10 `double mpack::math::RandNormal (const double mean, const double variance) [inline]`

Generates a normally distributed random number with specified mean and variance.

Parameters

<i>mean</i>	Mean of distribution.
<i>variance</i>	Variance of distribution.

Definition at line 141 of file random.hpp.

References randNormalDist.

21.17.2.11 `double mpack::math::Random () [inline]`

Generates a uniform random number between 0 and 1.

Definition at line 70 of file random.hpp.

References randUniformDist.

21.17.2.12 `double mpack::math::Random (const double lo, const double hi) [inline]`

Generates a uniform random number in the specified range.

Definition at line 84 of file random.hpp.

References randUniformDist.

21.17.2.13 `void mpack::math::RandomSeed (const size_t seed) [inline]`

Set the random seed used by the random functions (**Random()** (p. 107) and **RandInt()** (p. 106)).

The seed is casted to a 32-bit integer before being given to the random number generator, but a `size_t` is taken as a parameter for API consistency.

Parameters

<i>seed</i>	Seed for the random number generator.
-------------	---------------------------------------

Definition at line 55 of file random.hpp.

21.17.2.14 `void mlpack::math::RandVector (arma::vec & v)`

Overwrites a dimension-N vector to a random vector on the unit sphere in R^N .

21.17.2.15 `void mlpack::math::RemoveRows (const arma::mat & input, const std::vector< size_t > & rowsToRemove, arma::mat & output)`

Remove a certain set of rows in a matrix while copying to a second matrix.

Parameters

<i>input</i>	Input matrix to copy.
<i>rowsToRemove</i>	Vector containing indices of rows to be removed.
<i>output</i>	Matrix to copy non-removed rows into.

21.17.2.16 `void mlpack::math::VectorPower (arma::vec & vec, const double power)`

Auxiliary function to raise vector elements to a specific power.

The sign is ignored in the power operation and then re-added. Useful for eigenvalues.

21.17.2.17 `void mlpack::math::WhitenUsingEig (const arma::mat & x, arma::mat & xWhitened, arma::mat & whiteningMatrix)`

Whitens a matrix using the eigendecomposition of the covariance matrix.

Whitening means the covariance matrix of the result is the identity matrix.

21.17.2.18 `void mlpack::math::WhitenUsingSVD (const arma::mat & x, arma::mat & xWhitened, arma::mat & whiteningMatrix)`

Whitens a matrix using the singular value decomposition of the covariance matrix.

Whitening means the covariance matrix of the result is the identity matrix.

21.17.3 Variable Documentation

21.17.3.1 `boost::mt19937 mlpack::math::randGen`

21.17.3.2 `boost::normal_distribution mlpack::math::randNormalDist`

Referenced by `RandNormal()`.

21.17.3.3 `boost::uniform_01<boost::mt19937, double> mlpack::math::randUniformDist`

Referenced by `RandInt()`, and `Random()`.

21.18 mlpack::metric Namespace Reference

Classes

- class **IPMetric**
- class **LMetric**
The L_p metric for arbitrary integer p , with an option to take the root.
- class **MahalanobisDistance**
The Mahalanobis distance, which is essentially a stretched Euclidean distance.

Typedefs

- typedef **LMetric**< INT_MAX, false > **ChebyshevDistance**
- typedef **LMetric**< 2, true > **EuclideanDistance**
- typedef **LMetric**< 1, false > **ManhattanDistance**
- typedef **LMetric**< 2, false > **SquaredEuclideanDistance**

21.18.1 Typedef Documentation

21.18.1.1 typedef **LMetric**<INT_MAX, false> mlpack::metric::ChebyshevDistance

Definition at line 102 of file lmetric.hpp.

21.18.1.2 typedef **LMetric**<2, true> mlpack::metric::EuclideanDistance

Definition at line 97 of file lmetric.hpp.

21.18.1.3 typedef **LMetric**<1, false> mlpack::metric::ManhattanDistance

Definition at line 87 of file lmetric.hpp.

21.18.1.4 typedef **LMetric**<2, false> mlpack::metric::SquaredEuclideanDistance

Definition at line 92 of file lmetric.hpp.

21.19 mlpack::mvu Namespace Reference

Classes

- class **MVU**
*The **MVU** (p. 372) class is meant to provide a good abstraction for users.*

21.20 mlpack::naive_bayes Namespace Reference

The Naive Bayes Classifier.

Classes

- class **NaiveBayesClassifier**
The simple Naive Bayes classifier.

21.20.1 Detailed Description

The Naive Bayes Classifier.

21.21 mlpack::nca Namespace Reference

Neighborhood Components Analysis.

Classes

- class **NCA**
An implementation of Neighborhood Components Analysis, both a linear dimensionality reduction technique and a distance learning technique.
- class **SoftmaxErrorFunction**
The "softmax" stochastic neighbor assignment probability function.

21.21.1 Detailed Description

Neighborhood Components Analysis.

21.22 mlpack::neighbor Namespace Reference

Neighbor-search routines.

Classes

- class **FurthestNeighborSort**
*This class implements the necessary methods for the SortPolicy template parameter of the **NeighborSearch** (p. 399) class.*
- class **LSHSearch**
*The **LSHSearch** (p. 388) class – This class builds a hash on the reference set and uses this hash to compute the distance-approximate nearest-neighbors of the given queries.*
- class **NearestNeighborSort**
*This class implements the necessary methods for the SortPolicy template parameter of the **NeighborSearch** (p. 399) class.*
- class **NeighborSearch**
*The **NeighborSearch** (p. 399) class is a template class for performing distance-based neighbor searches.*
- class **NeighborSearchRules**
- class **NeighborSearchStat**
Extra data for each node in the tree.

- class **NeighborSearchTraversalInfo**
*Traversal information for **NeighborSearch** (p. 399).*
- class **RASearchRules**

Typedefs

- typedef **NeighborSearch**< **FurthestNeighborSort**, **metric::EuclideanDistance** > **AllkFN**
The AllkFN class is the all-k-furthest-neighbors method.
- typedef **NeighborSearch**< **NearestNeighborSort**, **metric::EuclideanDistance** > **AllkNN**
The AllkNN class is the all-k-nearest-neighbors method.
- typedef **RASearch**< **FurthestNeighborSort** > **AllkRAFN**
The AllkRAFN class is the all-k-rank-approximate-farthest-neighbors method.
- typedef **RASearch** **AllkRANN**
The AllkRANN class is the all-k-rank-approximate-nearest-neighbors method.

Functions

- void **Unmap** (const arma::Mat< size_t > &neighbors, const arma::mat &distances, const std::vector< size_t > &referenceMap, const std::vector< size_t > &queryMap, arma::Mat< size_t > &neighborsOut, arma::mat &distancesOut, const bool squareRoot=false)
Assuming that the datasets have been mapped using the referenceMap and the queryMap (such as during kd-tree construction), unmap the columns of the distances and neighbors matrices into neighborsOut and distancesOut, and also unmap the entries in each row of neighbors.
- void **Unmap** (const arma::Mat< size_t > &neighbors, const arma::mat &distances, const std::vector< size_t > &referenceMap, arma::Mat< size_t > &neighborsOut, arma::mat &distancesOut, const bool squareRoot=false)
Assuming that the datasets have been mapped using referenceMap (such as during kd-tree construction), unmap the columns of the distances and neighbors matrices into neighborsOut and distancesOut, and also unmap the entries in each row of neighbors.

21.22.1 Detailed Description

Neighbor-search routines.

These include all-nearest-neighbors and all-furthest-neighbors searches.

21.22.2 Typedef Documentation

21.22.2.1 typedef NeighborSearch<FurthestNeighborSort, metric::EuclideanDistance> mlpack::neighbor::AllkFN

The AllkFN class is the all-k-furthest-neighbors method.

It returns L2 distances (Euclidean distances) for each of the k furthest neighbors.

Definition at line 40 of file typedef.hpp.

21.22.2.2 `typedef NeighborSearch<NearestNeighborSort, metric::EuclideanDistance> mlpack::neighbor::AllkNN`

The AllkNN class is the all-k-nearest-neighbors method.

It returns L2 distances (Euclidean distances) for each of the k nearest neighbors.

Definition at line 34 of file `typedef.hpp`.

21.22.2.3 `typedef RASearch<FurthestNeighborSort> mlpack::neighbor::AllkRAFN`

The AllkRAFN class is the all-k-rank-approximate-farthest-neighbors method.

It returns squared L2 distances (squared Euclidean distances) for each of the k rank-approximate farthest-neighbors. Squared distances are used because they are slightly faster than non-squared distances (they have one fewer call to `sqrt()`).

The approximation is controlled with two parameters (see `allkrann_main.cpp`) which can be specified at search time. So the tree building is done only once while the search can be performed multiple times with different approximation levels.

Definition at line 55 of file `ra_typedef.hpp`.

21.22.2.4 `typedef RASearch mlpack::neighbor::AllkRANN`

The AllkRANN class is the all-k-rank-approximate-nearest-neighbors method.

It returns squared L2 distances (squared Euclidean distances) for each of the k rank-approximate nearest-neighbors. Squared distances are used because they are slightly faster than non-squared distances (they have one fewer call to `sqrt()`).

The approximation is controlled with two parameters (see `allkrann_main.cpp`) which can be specified at search time. So the tree building is done only once while the search can be performed multiple times with different approximation levels.

Definition at line 41 of file `ra_typedef.hpp`.

21.22.3 Function Documentation

21.22.3.1 `void mlpack::neighbor::Unmap (const arma::Mat< size_t > & neighbors, const arma::mat & distances, const std::vector< size_t > & referenceMap, const std::vector< size_t > & queryMap, arma::Mat< size_t > & neighborsOut, arma::mat & distancesOut, const bool squareRoot = false)`

Assuming that the datasets have been mapped using the `referenceMap` and the `queryMap` (such as during kd-tree construction), `unmap` the columns of the distances and neighbors matrices into `neighborsOut` and `distancesOut`, and also `unmap` the entries in each row of `neighbors`.

This is useful for the dual-tree case.

Parameters

<i>neighbors</i>	Matrix of neighbors resulting from neighbor search.
<i>distances</i>	Matrix of distances resulting from neighbor search.
<i>referenceMap</i>	Mapping of reference set to old points.

<i>queryMap</i>	Mapping of query set to old points.
<i>neighborsOut</i>	Matrix to store unmapped neighbors into.
<i>distancesOut</i>	Matrix to store unmapped distances into.
<i>squareRoot</i>	If true, take the square root of the distances.

21.22.3.2 `void mlpack::neighbor::Unmap (const arma::Mat< size_t > & neighbors, const arma::mat & distances, const std::vector< size_t > & referenceMap, arma::Mat< size_t > & neighborsOut, arma::mat & distancesOut, const bool squareRoot = false)`

Assuming that the datasets have been mapped using `referenceMap` (such as during kd-tree construction), unmap the columns of the distances and neighbors matrices into `neighborsOut` and `distancesOut`, and also unmap the entries in each row of neighbors.

This is useful for the single-tree case.

Parameters

<i>neighbors</i>	Matrix of neighbors resulting from neighbor search.
<i>distances</i>	Matrix of distances resulting from neighbor search.
<i>referenceMap</i>	Mapping of reference set to old points.
<i>neighborsOut</i>	Matrix to store unmapped neighbors into.
<i>distancesOut</i>	Matrix to store unmapped distances into.
<i>squareRoot</i>	If true, take the square root of the distances.

21.23 mlpack::nn Namespace Reference

Classes

- class **SparseAutoencoder**

A sparse autoencoder is a neural network whose aim to learn compressed representations of the data, typically for dimensionality reduction, with a constraint on the activity of the neurons in the network.

- class **SparseAutoencoderFunction**

This is a class for the sparse autoencoder objective function.

21.24 mlpack::optimization Namespace Reference

Namespaces

- **test**

Classes

- class **AugLagrangian**

*The **AugLagrangian** (p. 441) class implements the Augmented Lagrangian method of optimization.*

- class **AugLagrangianFunction**

*This is a utility class used by **AugLagrangian** (p. 441), meant to wrap a **LagrangianFunction** into a function usable by a simple optimizer like L-BFGS.*

- class **AugLagrangianTestFunction**

This function is taken from "Practical Mathematical Optimization" (Snyman), section 5.3.8 ("Application of the Augmented Lagrangian Method").

- class **ExponentialSchedule**

The exponential cooling schedule cools the temperature T at every step according to the equation.

- class **GockenbachFunction**

This function is taken from M.

- class **L_BFGS**

The generic L-BFGS optimizer, which uses a back-tracking line search algorithm to minimize a function.

- class **LovaszThetaSDP**

This function is the Lovasz-Theta semidefinite program, as implemented in the following paper:

- class **LRSDP**

***LRSDP** (p. 468) is the implementation of Monteiro and Burer's formulation of low-rank semidefinite programs (LR-SDP).*

- class **LRSDPFunction**

*The objective function that **LRSDP** (p. 468) is trying to optimize.*

- class **SA**

Simulated Annealing is an stochastic optimization algorithm which is able to deliver near-optimal results quickly without knowing the gradient of the function being optimized.

- class **SGD**

Stochastic Gradient Descent is a technique for minimizing a function which can be expressed as a sum of other functions.

21.25 mpack::optimization::test Namespace Reference

Classes

- class **GeneralizedRosenbrockFunction**

*The Generalized Rosenbrock function in n dimensions, defined by $f(x) = \sum_{i=1}^{n-1} (f(i)(x))$ $f_i(x) = 100 * (x_i^2 - x_{i+1})^2 + (1 - x_i)^2$ $x_0 = [-1.2, 1, -1.2, 1, \dots]$.*

- class **RosenbrockFunction**

The Rosenbrock function, defined by $f(x) = f_1(x) + f_2(x)$ $f_1(x) = 100 (x_2 - x_1^2)^2$ $f_2(x) = (1 - x_1)^2$ $x_0 = [-1.2, 1]$.

- class **RosenbrockWoodFunction**

The Generalized Rosenbrock function in 4 dimensions with the Wood Function in four dimensions.

- class **SGDTestFunction**

Very, very simple test function which is the composite of three other functions.

- class **WoodFunction**

The Wood function, defined by $f(x) = f_1(x) + f_2(x) + f_3(x) + f_4(x) + f_5(x) + f_6(x)$ $f_1(x) = 100 (x_2 - x_1^2)^2$ $f_2(x) = (1 - x_1)^2$ $f_3(x) = 90 (x_4 - x_3^2)^2$ $f_4(x) = (1 - x_3)^2$ $f_5(x) = 10 (x_2 + x_4 - 2)^2$ $f_6(x) = (1 / 10) (x_2 - x_4)^2$ $x_0 = [-3, -1, -3, -1]$.

21.26 mpack::pca Namespace Reference

Classes

- class **PCA**

*This class implements principal components analysis (**PCA** (p. 498)).*

21.27 mlpack::perceptron Namespace Reference

Classes

- class **Perceptron**
This class implements a simple perceptron (i.e., a single layer neural network).
- class **RandomInitialization**
This class is used to initialize weights for the weightVectors matrix in a random manner.
- class **SimpleWeightUpdate**
- class **ZeroInitialization**
This class is used to initialize the matrix weightVectors to zero.

21.28 mlpack::radical Namespace Reference

Classes

- class **Radical**
An implementation of RADICAL, an algorithm for independent component analysis (ICA).

Functions

- void **WhitenFeatureMajorMatrix** (const arma::mat &matX, arma::mat &matXWhitened, arma::mat &mat←
Whitening)

21.28.1 Function Documentation

- 21.28.1.1 void mlpack::radical::WhitenFeatureMajorMatrix (const arma::mat & matX, arma::mat & matXWhitened, arma::mat & matWhitening)

21.29 mlpack::range Namespace Reference

Range-search routines.

Classes

- class **RangeSearch**
*The **RangeSearch** (p. 512) class is a template class for performing range searches.*
- class **RangeSearchRules**
- class **RangeSearchStat**
*Statistic class for **RangeSearch** (p. 512), to be set to the StatisticType of the tree type that range search is being performed with.*

21.29.1 Detailed Description

Range-search routines.

21.30 mlpack::regression Namespace Reference

Regression methods.

Classes

- class **LARS**
*An implementation of **LARS** (p. 525), a stage-wise homotopy-based algorithm for l_1 -regularized linear regression (LASSO) and l_1+l_2 regularized linear regression (Elastic Net).*
- class **LinearRegression**
A simple linear regression algorithm using ordinary least squares.
- class **LogisticRegression**
- class **LogisticRegressionFunction**
The log-likelihood function for the logistic regression objective function.

21.30.1 Detailed Description

Regression methods.

21.31 mlpack::sparse_coding Namespace Reference

Classes

- class **DataDependentRandomInitializer**
*A data-dependent random dictionary initializer for **SparseCoding** (p. 547).*
- class **NothingInitializer**
*A DictionaryInitializer for **SparseCoding** (p. 547) which does not initialize anything; it is useful for when the dictionary is already known and will be set with **SparseCoding::Dictionary()** (p. 550).*
- class **RandomInitializer**
*A DictionaryInitializer for use with the **SparseCoding** (p. 547) class.*
- class **SparseCoding**
An implementation of Sparse Coding with Dictionary Learning that achieves sparsity via an l_1 -norm regularizer on the codes (LASSO) or an (l_1+l_2) -norm regularizer on the codes (the Elastic Net).

21.32 mlpack::svd Namespace Reference

Classes

- class **QUIC_SVD**
- class **RegularizedSVD**
- class **RegularizedSVDFunction**

21.33 mlpack::tree Namespace Reference

Trees and tree-building procedures.

Classes

- class **BinarySpaceTree**
A binary space partitioning tree, such as a KD-tree or a ball tree.
- class **CompareCosineNode**
- class **CosineTree**
- class **CoverTree**
A cover tree is a tree specifically designed to speed up nearest-neighbor computation in high-dimensional spaces.
- class **EmptyStatistic**
Empty statistic if you are not interested in storing statistics in your tree.
- class **ExampleTree**
This is not an actual space tree but instead an example tree that exists to show and document all the functions that mlpack trees must implement.
- class **FirstPointIsRoot**
*This class is meant to be used as a choice for the policy class RootPointPolicy of the **CoverTree** (p. 603) class.*
- class **MeanSplit**
A binary space partitioning tree node is split into its left and right child.
- class **MRKDStatistic**
Statistic for multi-resolution kd-trees.
- class **TreeTraits**
*The **TreeTraits** (p. 645) class provides compile-time information on the characteristics of a given tree type.*
- class **TreeTraits**< **BinarySpaceTree**< **BoundType**, **StatisticType**, **MatType** > >
*This is a specialization of the **TreeType** class to the **BinarySpaceTree** (p. 566) tree type.*
- class **TreeTraits**< **CoverTree**< **MetricType**, **RootPointPolicy**, **StatisticType** > >
*The specialization of the **TreeTraits** (p. 645) class for the **CoverTree** (p. 603) tree type.*

Typedefs

- typedef boost::heap::priority_queue< **CosineTree** *, boost::heap::compare< **CompareCosineNode** > > **CosineNodeQueue**

21.33.1 Detailed Description

Trees and tree-building procedures.

21.33.2 Typedef Documentation

- 21.33.2.1 typedef boost::heap::priority_queue<**CosineTree***, boost::heap::compare<**CompareCosineNode**> > mlpack::tree::CosineNodeQueue

Definition at line 26 of file cosine_tree.hpp.

21.34 mlpack::util Namespace Reference

Classes

- class **CLIDeleter**

*Extremely simple class whose only job is to delete the existing **CLI** (p. 184) object at the end of execution.*

- class **NullOutputStream**

*Used for **Log::Debug** (p. 359) when not compiled with debugging symbols.*

- class **Option**

*A static object whose constructor registers a parameter with the **CLI** (p. 184) class.*

- class **PrefixedOutputStream**

Allows us to output to an ostream with a prefix at the beginning of each line, in the same way we would output to cout or cerr.

- class **ProgramDoc**

*A static object whose constructor registers program documentation with the **CLI** (p. 184) class.*

- class **SaveRestoreUtility**

Functions

- std::string **GetVersion** ()

This will return either "mlpack x.y.z" or "mlpack trunk-rXXXXX" depending on whether or not this is a stable version of mlpack or an svn revision.

- std::string **Indent** (std::string input, const size_t howManyTabs=1)

A utility function that replaces all all newlines with a number of spaces depending on the indentation level.

Variables

- static **CLIDeleter cliDeleter**

Declare the deleter.

21.34.1 Function Documentation

21.34.1.1 std::string mlpack::util::GetVersion ()

This will return either "mlpack x.y.z" or "mlpack trunk-rXXXXX" depending on whether or not this is a stable version of mlpack or an svn revision.

21.34.1.2 std::string mlpack::util::Indent (std::string input, const size_t howManyTabs = 1)

A utility function that replaces all all newlines with a number of spaces depending on the indentation level.

Referenced by mlpack::kernel::PSpectrumStringKernel::ToString().

21.34.2 Variable Documentation

21.34.2.1 CLIDeleter mlpack::util::cliDeleter [static]

Declare the deleter.

Definition at line 35 of file cli_deleter.hpp.

Chapter 22

Class Documentation

22.1 `IsVector< VecType >` Struct Template Reference

If `value == true`, then `VecType` is some sort of Armadillo vector or subview.

Static Public Attributes

- static const bool **value** = false

22.1.1 Detailed Description

```
template<typename VecType> struct IsVector< VecType >
```

If `value == true`, then `VecType` is some sort of Armadillo vector or subview.

You might use this struct like this:

```
// Only accepts VecTypes that are actually Armadillo vector types.
template<typename VecType>
void Function(const VecType& argumentA,
              typename boost::enable_if<IsVector<VecType> >*> = 0);
```

The use of the `enable_if` object allows the compiler to instantiate `Function()` only if `VecType` is one of the Armadillo vector types. It has a default argument because it isn't meant to be used in either the function call or the function body.

Definition at line 37 of file `arma_traits.hpp`.

22.1.2 Member Data Documentation

22.1.2.1 `template<typename VecType > const bool IsVector< VecType >::value = false` [static]

Definition at line 39 of file `arma_traits.hpp`.

The documentation for this struct was generated from the following file:

- `src/mlpack/core/util/arma_traits.hpp`

22.2 `IsVector< arma::Col< eT > >` Struct Template Reference

Static Public Attributes

- static const bool **value** = true

22.2.1 Detailed Description

```
template<typename eT>struct IsVector< arma::Col< eT > >
```

Definition at line 46 of file arma_traits.hpp.

22.2.2 Member Data Documentation

22.2.2.1 `template<typename eT > const bool IsVector< arma::Col< eT > >::value = true` `[static]`

Definition at line 48 of file arma_traits.hpp.

The documentation for this struct was generated from the following file:

- src/mlpack/core/util/**arma_traits.hpp**

22.3 `IsVector< arma::Row< eT > >` Struct Template Reference

Static Public Attributes

- static const bool **value** = true

22.3.1 Detailed Description

```
template<typename eT>struct IsVector< arma::Row< eT > >
```

Definition at line 60 of file arma_traits.hpp.

22.3.2 Member Data Documentation

22.3.2.1 `template<typename eT > const bool IsVector< arma::Row< eT > >::value = true` `[static]`

Definition at line 62 of file arma_traits.hpp.

The documentation for this struct was generated from the following file:

- src/mlpack/core/util/**arma_traits.hpp**

22.4 `IsVector< arma::SpCol< eT > >` Struct Template Reference

Static Public Attributes

- static const bool **value** = true

22.4.1 Detailed Description

```
template<typename eT>struct IsVector< arma::SpCol< eT > >
```

Definition at line 53 of file `arma_traits.hpp`.

22.4.2 Member Data Documentation

22.4.2.1 `template<typename eT > const bool IsVector< arma::SpCol< eT > >::value = true` `[static]`

Definition at line 55 of file `arma_traits.hpp`.

The documentation for this struct was generated from the following file:

- `src/mlpack/core/util/arma_traits.hpp`

22.5 `IsVector< arma::SpRow< eT > >` Struct Template Reference

Static Public Attributes

- static const bool **value** = true

22.5.1 Detailed Description

```
template<typename eT>struct IsVector< arma::SpRow< eT > >
```

Definition at line 67 of file `arma_traits.hpp`.

22.5.2 Member Data Documentation

22.5.2.1 `template<typename eT > const bool IsVector< arma::SpRow< eT > >::value = true` `[static]`

Definition at line 69 of file `arma_traits.hpp`.

The documentation for this struct was generated from the following file:

- `src/mlpack/core/util/arma_traits.hpp`

22.6 `IsVector< arma::SpSubview< eT > >` Struct Template Reference

Static Public Attributes

- static const bool **value** = true

22.6.1 Detailed Description

```
template<typename eT>struct IsVector< arma::SpSubview< eT > >
```

Definition at line 91 of file arma_traits.hpp.

22.6.2 Member Data Documentation

22.6.2.1 `template<typename eT > const bool IsVector< arma::SpSubview< eT > >::value = true` `[static]`

Definition at line 93 of file arma_traits.hpp.

The documentation for this struct was generated from the following file:

- src/mlpack/core/util/**arma_traits.hpp**

22.7 IsVector< arma::subview_col< eT > > Struct Template Reference

Static Public Attributes

- static const bool **value** = true

22.7.1 Detailed Description

```
template<typename eT>struct IsVector< arma::subview_col< eT > >
```

Definition at line 74 of file arma_traits.hpp.

22.7.2 Member Data Documentation

22.7.2.1 `template<typename eT > const bool IsVector< arma::subview_col< eT > >::value = true` `[static]`

Definition at line 76 of file arma_traits.hpp.

The documentation for this struct was generated from the following file:

- src/mlpack/core/util/**arma_traits.hpp**

22.8 IsVector< arma::subview_row< eT > > Struct Template Reference

Static Public Attributes

- static const bool **value** = true

22.8.1 Detailed Description

```
template<typename eT>struct IsVector< arma::subview_row< eT > >
```

Definition at line 81 of file arma_traits.hpp.

22.8.2 Member Data Documentation

```
22.8.2.1 template<typename eT > const bool IsVector< arma::subview_row< eT > >::value = true [static]
```

Definition at line 83 of file arma_traits.hpp.

The documentation for this struct was generated from the following file:

- src/mlpack/core/util/arma_traits.hpp

22.9 mlpack::amf::AMF< TerminationPolicyType, InitializationRuleType, UpdateRuleType > Class Template Reference

This class implements **AMF** (p. 123) (alternating matrix factorization) on the given matrix V.

Public Member Functions

- **AMF** (const TerminationPolicyType &terminationPolicy=TerminationPolicyType(), const InitializationRuleType &initializeRule=InitializationRuleType(), const UpdateRuleType &update=UpdateRuleType())
*Create the **AMF** (p. 123) object and (optionally) set the parameters which **AMF** (p. 123) will run with.*
- template<typename MatType >
double **Apply** (const MatType &V, const size_t r, arma::mat &W, arma::mat &H)
Apply Alternating Matrix Factorization to the provided matrix.
- const InitializationRuleType & **InitializeRule** () const
Access the initialization rule.
- InitializationRuleType & **InitializeRule** ()
Modify the initialization rule.
- const TerminationPolicyType & **TerminationPolicy** () const
Access the termination policy.
- TerminationPolicyType & **TerminationPolicy** ()
Modify the termination policy.
- const UpdateRuleType & **Update** () const
Access the update rule.
- UpdateRuleType & **Update** ()
Modify the update rule.

Private Attributes

- InitializationRuleType **initializationRule**
Instantiated initialization Rule.
- TerminationPolicyType **terminationPolicy**
Termination policy.
- UpdateRuleType **update**
Instantiated update rule.

22.9.1 Detailed Description

```
template<typename TerminationPolicyType = SimpleResidueTermination, typename InitializationRuleType = RandomInitialization,
typename UpdateRuleType = NMFMultiplicativeDistanceUpdate>class mlpack::amf::AMF< TerminationPolicyType, InitializationRuleType, UpdateRuleType >
```

This class implements **AMF** (p. 123) (alternating matrix factorization) on the given matrix V .

Alternating matrix factorization decomposes V in the form $V \approx WH$ where W is called the basis matrix and H is called the encoding matrix. V is taken to be of size $n \times m$ and the obtained W is $n \times r$ and H is $r \times m$. The size r is called the rank of the factorization.

The implementation requires three template types; the first contains the policy used to determine when the algorithm has converged; the second contains the initialization rule for the W and H matrix; the last contains the update rule to be used during each iteration. This templization allows the user to try various update rules, initialization rules, and termination policies (including ones not supplied with MLPACK) for factorization. By default, the template parameters to **AMF** (p. 123) implement non-negative matrix factorization with the multiplicative distance update.

A simple example of how to run **AMF** (p. 123) (or NMF) is shown below.

```
extern arma::mat V; // Matrix that we want to perform LMF on.
size_t r = 10; // Rank of decomposition
arma::mat W; // Basis matrix
arma::mat H; // Encoding matrix

AMF<> amf; // Default options: NMF with multiplicative distance update rules.
amf.Apply(V, W, H, r);
```

Template Parameters

<i>TerminationPolicy</i>	The policy to use for determining when the factorization has converged.
<i>InitializationRule</i>	The initialization rule for initializing W and H matrix.
<i>UpdateRule</i>	The update rule for calculating W and H matrix at each iteration.

See also

NMFMultiplicativeDistanceUpdate (p. 134), **SimpleResidueTermination** (p. 139)

Definition at line 71 of file amf.hpp.

22.9.2 Constructor & Destructor Documentation

```
22.9.2.1 template<typename TerminationPolicyType = SimpleResidueTermination, typename InitializationRuleType =
RandomInitialization, typename UpdateRuleType = NMFMultiplicativeDistanceUpdate> mlpack::amf::AMF<
TerminationPolicyType, InitializationRuleType, UpdateRuleType >::AMF ( const TerminationPolicyType &
terminationPolicy = TerminationPolicyType(), const InitializationRuleType & initializeRule =
InitializationRuleType(), const UpdateRuleType & update = UpdateRuleType() )
```

Create the **AMF** (p. 123) object and (optionally) set the parameters which **AMF** (p. 123) will run with.

The minimum residue refers to the root mean square of the difference between two subsequent iterations of the product $W * H$. A low residue indicates that subsequent iterations are not producing much change in W and H . Once the residue goes below the specified minimum residue, the algorithm terminates.

Parameters

<i>initializationRule</i>	Optional instantiated InitializationRule object for initializing the W and H matrices.
<i>updateRule</i>	Optional instantiated UpdateRule object; this parameter is useful when the update rule for the W and H vector has state that it needs to store (i.e. HUpdate() and WUpdate() are not static functions).
<i>terminationPolicy</i>	Optional instantiated TerminationPolicy object.

22.9.3 Member Function Documentation

22.9.3.1 `template<typename TerminationPolicyType = SimpleResidueTermination, typename InitializationRuleType = RandomInitialization, typename UpdateRuleType = NMFMultiplicativeDistanceUpdate> template<typename MatType > double mlpack::amf::AMF< TerminationPolicyType, InitializationRuleType, UpdateRuleType >::Apply (const MatType & V, const size_t r, arma::mat & W, arma::mat & H)`

Apply Alternating Matrix Factorization to the provided matrix.

Parameters

<i>V</i>	Input matrix to be factorized.
<i>W</i>	Basis matrix to be output.
<i>H</i>	Encoding matrix to output.
<i>r</i>	Rank r of the factorization.

22.9.3.2 `template<typename TerminationPolicyType = SimpleResidueTermination, typename InitializationRuleType = RandomInitialization, typename UpdateRuleType = NMFMultiplicativeDistanceUpdate> const InitializationRuleType& mlpack::amf::AMF< TerminationPolicyType, InitializationRuleType, UpdateRuleType >::InitializeRule () const [inline]`

Access the initialization rule.

Definition at line 115 of file amf.hpp.

References mlpack::amf::AMF< TerminationPolicyType, InitializationRuleType, UpdateRuleType >::initializationRule.

22.9.3.3 `template<typename TerminationPolicyType = SimpleResidueTermination, typename InitializationRuleType = RandomInitialization, typename UpdateRuleType = NMFMultiplicativeDistanceUpdate> InitializationRuleType& mlpack::amf::AMF< TerminationPolicyType, InitializationRuleType, UpdateRuleType >::InitializeRule () [inline]`

Modify the initialization rule.

Definition at line 118 of file amf.hpp.

References mlpack::amf::AMF< TerminationPolicyType, InitializationRuleType, UpdateRuleType >::initializationRule.

22.9.3.4 `template<typename TerminationPolicyType = SimpleResidueTermination, typename InitializationRuleType = RandomInitialization, typename UpdateRuleType = NMFMultiplicativeDistanceUpdate> const TerminationPolicyType& mlpack::amf::AMF< TerminationPolicyType, InitializationRuleType, UpdateRuleType >::TerminationPolicy () const [inline]`

Access the termination policy.

Definition at line 109 of file amf.hpp.

References `mlpack::amf::AMF< TerminationPolicyType, InitializationRuleType, UpdateRuleType >::terminationPolicy`.

```
22.9.3.5  template<typename TerminationPolicyType = SimpleResidueTermination, typename InitializationRuleType =
          RandomInitialization, typename UpdateRuleType = NMFMultiplicativeDistanceUpdate> TerminationPolicyType&
          mlpack::amf::AMF< TerminationPolicyType, InitializationRuleType, UpdateRuleType >::TerminationPolicy ( )
          [inline]
```

Modify the termination policy.

Definition at line 112 of file `amf.hpp`.

References `mlpack::amf::AMF< TerminationPolicyType, InitializationRuleType, UpdateRuleType >::terminationPolicy`.

```
22.9.3.6  template<typename TerminationPolicyType = SimpleResidueTermination, typename InitializationRuleType =
          RandomInitialization, typename UpdateRuleType = NMFMultiplicativeDistanceUpdate> const UpdateRuleType&
          mlpack::amf::AMF< TerminationPolicyType, InitializationRuleType, UpdateRuleType >::Update ( ) const
          [inline]
```

Access the update rule.

Definition at line 121 of file `amf.hpp`.

References `mlpack::amf::AMF< TerminationPolicyType, InitializationRuleType, UpdateRuleType >::update`.

```
22.9.3.7  template<typename TerminationPolicyType = SimpleResidueTermination, typename InitializationRuleType
          = RandomInitialization, typename UpdateRuleType = NMFMultiplicativeDistanceUpdate> UpdateRuleType&
          mlpack::amf::AMF< TerminationPolicyType, InitializationRuleType, UpdateRuleType >::Update ( ) [inline]
```

Modify the update rule.

Definition at line 123 of file `amf.hpp`.

References `mlpack::amf::AMF< TerminationPolicyType, InitializationRuleType, UpdateRuleType >::update`.

22.9.4 Member Data Documentation

```
22.9.4.1  template<typename TerminationPolicyType = SimpleResidueTermination, typename InitializationRuleType =
          RandomInitialization, typename UpdateRuleType = NMFMultiplicativeDistanceUpdate> InitializationRuleType
          mlpack::amf::AMF< TerminationPolicyType, InitializationRuleType, UpdateRuleType >::initializationRule
          [private]
```

Instantiated initialization Rule.

Definition at line 129 of file `amf.hpp`.

Referenced by `mlpack::amf::AMF< TerminationPolicyType, InitializationRuleType, UpdateRuleType >::InitializeRule()`.

```
22.9.4.2  template<typename TerminationPolicyType = SimpleResidueTermination, typename InitializationRuleType =
          RandomInitialization, typename UpdateRuleType = NMFMultiplicativeDistanceUpdate> TerminationPolicyType
          mlpack::amf::AMF< TerminationPolicyType, InitializationRuleType, UpdateRuleType >::terminationPolicy
          [private]
```

Termination policy.

Definition at line 127 of file `amf.hpp`.

Referenced by mlpack::amf::AMF< TerminationPolicyType, InitializationRuleType, UpdateRuleType >::TerminationPolicy().

```
22.9.4.3 template<typename TerminationPolicyType = SimpleResidueTermination, typename InitializationRuleType
        = RandomInitialization, typename UpdateRuleType = NMFMultiplicativeDistanceUpdate> UpdateRuleType
        mlpack::amf::AMF< TerminationPolicyType, InitializationRuleType, UpdateRuleType >::update [private]
```

Instantiated update rule.

Definition at line 131 of file amf.hpp.

Referenced by mlpack::amf::AMF< TerminationPolicyType, InitializationRuleType, UpdateRuleType >::Update().

The documentation for this class was generated from the following file:

- src/mlpack/methods/amf/**amf.hpp**

22.10 mlpack::amf::AverageInitialization Class Reference

This initialization rule initializes matrix W and H to root of average of V with uniform noise.

Public Member Functions

- **AverageInitialization** ()

Static Public Member Functions

- template<typename MatType >
static void **Initialize** (const MatType &V, const size_t r, arma::mat &W, arma::mat &H)

22.10.1 Detailed Description

This initialization rule initializes matrix W and H to root of average of V with uniform noise.

Uniform noise is generated by Armadillo's 'randu' function. To have a better effect lower bound of the matrix is subtracted from average before dividing it by the factorization rank. This computed value is added with the random noise.

Definition at line 29 of file average_init.hpp.

22.10.2 Constructor & Destructor Documentation

22.10.2.1 mlpack::amf::AverageInitialization::AverageInitialization () [inline]

Definition at line 33 of file average_init.hpp.

22.10.3 Member Function Documentation

22.10.3.1 `template<typename MatType > static void mlpack::amf::AverageInitialization::Initialize (const MatType & V, const size_t r, arma::mat & W, arma::mat & H) [inline],[static]`

Definition at line 36 of file `average_init.hpp`.

The documentation for this class was generated from the following file:

- `src/mlpack/methods/amf/init_rules/average_init.hpp`

22.11 `mlpack::amf::CompleteIncrementalTermination< TerminationPolicy >` Class Template Reference

Public Member Functions

- **CompleteIncrementalTermination** (TerminationPolicy **t_policy**=TerminationPolicy())
Empty constructor.
- const double & **Index** ()
- `template<class MatType >`
void **Initialize** (const MatType &V)
- void **Initialize** (const arma::sp_mat &V)
- bool **IsConverged** (arma::mat &W, arma::mat &H)
- const size_t & **Iteration** ()
- const size_t & **MaxIterations** ()

Private Attributes

- size_t **incrementalIndex**
- size_t **iteration**
- TerminationPolicy **t_policy**

22.11.1 Detailed Description

`template<class TerminationPolicy> class mlpack::amf::CompleteIncrementalTermination< TerminationPolicy >`

Definition at line 23 of file `complete_incremental_termination.hpp`.

22.11.2 Constructor & Destructor Documentation

22.11.2.1 `template<class TerminationPolicy > mlpack::amf::CompleteIncrementalTermination< TerminationPolicy >::CompleteIncrementalTermination (TerminationPolicy t_policy = TerminationPolicy()) [inline]`

Empty constructor.

Parameters

<code>t_policy</code>	object of wrapped class.
-----------------------	--------------------------

Definition at line 31 of file `complete_incremental_termination.hpp`.

22.11.3 Member Function Documentation

22.11.3.1 `template<class TerminationPolicy > const double& mpack::amf::CompleteIncrementalTermination< TerminationPolicy >::Index () [inline]`

Definition at line 59 of file `complete_incremental_termination.hpp`.

References `mpack::amf::CompleteIncrementalTermination< TerminationPolicy >::t_policy`.

22.11.3.2 `template<class TerminationPolicy > template<class MatType > void mpack::amf::CompleteIncrementalTermination< TerminationPolicy >::Initialize (const MatType & V) [inline]`

Definition at line 35 of file `complete_incremental_termination.hpp`.

References `mpack::amf::CompleteIncrementalTermination< TerminationPolicy >::incrementalIndex`, `mpack::amf::CompleteIncrementalTermination< TerminationPolicy >::iteration`, and `mpack::amf::CompleteIncrementalTermination< TerminationPolicy >::t_policy`.

22.11.3.3 `template<class TerminationPolicy > void mpack::amf::CompleteIncrementalTermination< TerminationPolicy >::Initialize (const arma::sp_mat & V) [inline]`

Definition at line 43 of file `complete_incremental_termination.hpp`.

References `mpack::amf::CompleteIncrementalTermination< TerminationPolicy >::incrementalIndex`, `mpack::amf::CompleteIncrementalTermination< TerminationPolicy >::iteration`, and `mpack::amf::CompleteIncrementalTermination< TerminationPolicy >::t_policy`.

22.11.3.4 `template<class TerminationPolicy > bool mpack::amf::CompleteIncrementalTermination< TerminationPolicy >::IsConverged (arma::mat & W, arma::mat & H) [inline]`

Definition at line 51 of file `complete_incremental_termination.hpp`.

References `mpack::amf::CompleteIncrementalTermination< TerminationPolicy >::incrementalIndex`, `mpack::amf::CompleteIncrementalTermination< TerminationPolicy >::iteration`, and `mpack::amf::CompleteIncrementalTermination< TerminationPolicy >::t_policy`.

22.11.3.5 `template<class TerminationPolicy > const size_t& mpack::amf::CompleteIncrementalTermination< TerminationPolicy >::Iteration () [inline]`

Definition at line 63 of file `complete_incremental_termination.hpp`.

References `mpack::amf::CompleteIncrementalTermination< TerminationPolicy >::iteration`.

22.11.3.6 `template<class TerminationPolicy > const size_t& mpack::amf::CompleteIncrementalTermination< TerminationPolicy >::MaxIterations () [inline]`

Definition at line 68 of file `complete_incremental_termination.hpp`.

References `mlpack::amf::CompleteIncrementalTermination< TerminationPolicy >::t_policy`.

22.11.4 Member Data Documentation

22.11.4.1 `template<class TerminationPolicy > size_t mlpack::amf::CompleteIncrementalTermination< TerminationPolicy >::incrementalIndex [private]`

Definition at line 76 of file `complete_incremental_termination.hpp`.

Referenced by `mlpack::amf::CompleteIncrementalTermination< TerminationPolicy >::Initialize()`, and `mlpack::amf::CompleteIncrementalTermination< TerminationPolicy >::IsConverged()`.

22.11.4.2 `template<class TerminationPolicy > size_t mlpack::amf::CompleteIncrementalTermination< TerminationPolicy >::iteration [private]`

Definition at line 77 of file `complete_incremental_termination.hpp`.

Referenced by `mlpack::amf::CompleteIncrementalTermination< TerminationPolicy >::Initialize()`, `mlpack::amf::CompleteIncrementalTermination< TerminationPolicy >::IsConverged()`, and `mlpack::amf::CompleteIncrementalTermination< TerminationPolicy >::Iteration()`.

22.11.4.3 `template<class TerminationPolicy > TerminationPolicy mlpack::amf::CompleteIncrementalTermination< TerminationPolicy >::t_policy [private]`

Definition at line 74 of file `complete_incremental_termination.hpp`.

Referenced by `mlpack::amf::CompleteIncrementalTermination< TerminationPolicy >::Index()`, `mlpack::amf::CompleteIncrementalTermination< TerminationPolicy >::Initialize()`, `mlpack::amf::CompleteIncrementalTermination< TerminationPolicy >::IsConverged()`, and `mlpack::amf::CompleteIncrementalTermination< TerminationPolicy >::MaxIterations()`.

The documentation for this class was generated from the following file:

- `src/mlpack/methods/amf/termination_policies/complete_incremental_termination.hpp`

22.12 mlpack::amf::IncompleteIncrementalTermination< TerminationPolicy > Class Template Reference

Public Member Functions

- **IncompleteIncrementalTermination** (TerminationPolicy **t_policy**=TerminationPolicy())
Empty constructor.
- const double & **Index** ()
- template<class MatType >
void **Initialize** (const MatType &V)
- bool **IsConverged** (arma::mat &W, arma::mat &H)
- const size_t & **Iteration** ()
- const size_t & **MaxIterations** ()

Private Attributes

- size_t **incrementalIndex**
- size_t **iteration**
- TerminationPolicy **t_policy**

22.12.1 Detailed Description

template<class TerminationPolicy>class mlpack::amf::IncompleteIncrementalTermination< TerminationPolicy >

Definition at line 21 of file incomplete_incremental_termination.hpp.

22.12.2 Constructor & Destructor Documentation

22.12.2.1 template<class TerminationPolicy > mlpack::amf::IncompleteIncrementalTermination< TerminationPolicy >::IncompleteIncrementalTermination (TerminationPolicy *t_policy* = TerminationPolicy())
[inline]

Empty constructor.

Parameters

<i>t_policy</i>	object of wrapped class.
-----------------	--------------------------

Definition at line 29 of file incomplete_incremental_termination.hpp.

22.12.3 Member Function Documentation

22.12.3.1 template<class TerminationPolicy > const double& mlpack::amf::IncompleteIncrementalTermination< TerminationPolicy >::Index () [inline]

Definition at line 49 of file incomplete_incremental_termination.hpp.

References mlpack::amf::IncompleteIncrementalTermination< TerminationPolicy >::t_policy.

22.12.3.2 template<class TerminationPolicy > template<class MatType > void mlpack::amf::IncompleteIncrementalTermination< TerminationPolicy >::Initialize (const MatType & *V*)
[inline]

Definition at line 33 of file incomplete_incremental_termination.hpp.

References mlpack::amf::IncompleteIncrementalTermination< TerminationPolicy >::incrementalIndex, mlpack::amf::IncompleteIncrementalTermination< TerminationPolicy >::iteration, and mlpack::amf::IncompleteIncrementalTermination< TerminationPolicy >::t_policy.

22.12.3.3 template<class TerminationPolicy > bool mlpack::amf::IncompleteIncrementalTermination< TerminationPolicy >::IsConverged (arma::mat & *W*, arma::mat & *H*) [inline]

Definition at line 41 of file incomplete_incremental_termination.hpp.

References mlpack::amf::IncompleteIncrementalTermination< TerminationPolicy >::incrementalIndex, mlpack::amf::IncompleteIncrementalTermination< TerminationPolicy >::iteration, and mlpack::amf::IncompleteIncrementalTermination< TerminationPolicy >::t_policy.

22.12.3.4 `template<class TerminationPolicy > const size_t& mlpack::amf::IncompleteIncrementalTermination< TerminationPolicy >::iteration () [inline]`

Definition at line 53 of file `incomplete_incremental_termination.hpp`.

References `mlpack::amf::IncompleteIncrementalTermination< TerminationPolicy >::iteration`.

22.12.3.5 `template<class TerminationPolicy > const size_t& mlpack::amf::IncompleteIncrementalTermination< TerminationPolicy >::MaxIterations () [inline]`

Definition at line 57 of file `incomplete_incremental_termination.hpp`.

References `mlpack::amf::IncompleteIncrementalTermination< TerminationPolicy >::t_policy`.

22.12.4 Member Data Documentation

22.12.4.1 `template<class TerminationPolicy > size_t mlpack::amf::IncompleteIncrementalTermination< TerminationPolicy >::incrementalIndex [private]`

Definition at line 65 of file `incomplete_incremental_termination.hpp`.

Referenced by `mlpack::amf::IncompleteIncrementalTermination< TerminationPolicy >::Initialize()`, and `mlpack::amf::IncompleteIncrementalTermination< TerminationPolicy >::IsConverged()`.

22.12.4.2 `template<class TerminationPolicy > size_t mlpack::amf::IncompleteIncrementalTermination< TerminationPolicy >::iteration [private]`

Definition at line 66 of file `incomplete_incremental_termination.hpp`.

Referenced by `mlpack::amf::IncompleteIncrementalTermination< TerminationPolicy >::Initialize()`, `mlpack::amf::IncompleteIncrementalTermination< TerminationPolicy >::IsConverged()`, and `mlpack::amf::IncompleteIncrementalTermination< TerminationPolicy >::iteration()`.

22.12.4.3 `template<class TerminationPolicy > TerminationPolicy mlpack::amf::IncompleteIncrementalTermination< TerminationPolicy >::t_policy [private]`

Definition at line 63 of file `incomplete_incremental_termination.hpp`.

Referenced by `mlpack::amf::IncompleteIncrementalTermination< TerminationPolicy >::Index()`, `mlpack::amf::IncompleteIncrementalTermination< TerminationPolicy >::Initialize()`, `mlpack::amf::IncompleteIncrementalTermination< TerminationPolicy >::IsConverged()`, and `mlpack::amf::IncompleteIncrementalTermination< TerminationPolicy >::MaxIterations()`.

The documentation for this class was generated from the following file:

- `src/mlpack/methods/amf/termination_policies/incomplete_incremental_termination.hpp`

22.13 mlpack::amf::NMFALSUpdate Class Reference

This class implements a method titled 'Alternating Least Squares' described in the paper 'Positive Matrix Factorization: A Non-negative Factor Model with Optimal Utilization of Error Estimates of Data Values' by P Paatero and U Tapper.

Public Member Functions

- **NMFALSUpdate** ()
Empty constructor required for the UpdateRule template.
- template<typename MatType >
void **Initialize** (const MatType &dataset, const size_t rank)

Static Public Member Functions

- template<typename MatType >
static void **HUpdate** (const MatType &V, const arma::mat &W, arma::mat &H)
The update rule for the encoding matrix H.
- template<typename MatType >
static void **WUpdate** (const MatType &V, arma::mat &W, const arma::mat &H)
The update rule for the basis matrix W.

22.13.1 Detailed Description

This class implements a method titled 'Alternating Least Squares' described in the paper 'Positive Matrix Factorization: A Non-negative Factor Model with Optimal Utilization of Error Estimates of Data Values' by P Paatero and U Tapper.

It uses least squares projection formula to reduce the error value of $\sqrt{\sum_i \sum_j (V - WH)^2}$ by alternately calculating W and H respectively while holding the other matrix constant.

Definition at line 30 of file nmf_als.hpp.

22.13.2 Constructor & Destructor Documentation

22.13.2.1 mlpack::amf::NMFALSUpdate::NMFALSUpdate () [inline]

Empty constructor required for the UpdateRule template.

Definition at line 34 of file nmf_als.hpp.

22.13.3 Member Function Documentation

22.13.3.1 template<typename MatType > static void mlpack::amf::NMFALSUpdate::HUpdate (const MatType & V, const arma::mat & W, arma::mat & H) [inline], [static]

The update rule for the encoding matrix H.

The formula used is

$$H = \frac{W^T V}{W^T W}$$

The function takes in all the matrices and only changes the value of the H matrix.

Parameters

V	Input matrix to be factorized.
W	Basis matrix.
H	Encoding matrix to be updated.

Definition at line 87 of file nmf_als.hpp.

22.13.3.2 `template<typename MatType > void mlpack::amf::NMFALSUpdate::Initialize (const MatType & dataset, const size_t rank) [inline]`

Definition at line 37 of file nmf_als.hpp.

22.13.3.3 `template<typename MatType > static void mlpack::amf::NMFALSUpdate::WUpdate (const MatType & V, arma::mat & W, const arma::mat & H) [inline],[static]`

The update rule for the basis matrix W .

The formula used is

$$W^T = \frac{HV^T}{HH^T}$$

The function takes in all the matrices and only changes the value of the W matrix.

Parameters

V	Input matrix to be factorized.
W	Basis matrix to be updated.
H	Encoding matrix.

Definition at line 56 of file nmf_als.hpp.

The documentation for this class was generated from the following file:

- src/mlpack/methods/amf/update_rules/nmf_als.hpp

22.14 mlpack::amf::NFMultiplicativeDistanceUpdate Class Reference

The multiplicative distance update rules for matrices W and H .

Public Member Functions

- **NFMultiplicativeDistanceUpdate** ()
- `template<typename MatType > void Initialize (const MatType &dataset, const size_t rank)`

Static Public Member Functions

- `template<typename MatType > static void HUpdate (const MatType &V, const arma::mat &W, arma::mat &H)`
The update rule for the encoding matrix H .
- `template<typename MatType > static void WUpdate (const MatType &V, arma::mat &W, const arma::mat &H)`
The update rule for the basis matrix W .

22.14.1 Detailed Description

The multiplicative distance update rules for matrices W and H .

This follows a method described in the paper 'Algorithms for Non-negative Matrix Factorization' by D. D. Lee and H. S. Seung. This is a multiplicative rule that ensures that the Frobenius norm $\sqrt{\sum_i \sum_j (V - WH)^2}$ is non-increasing between subsequent iterations. Both of the update rules for W and H are defined in this file.

Definition at line 30 of file nmf_mult_dist.hpp.

22.14.2 Constructor & Destructor Documentation

22.14.2.1 `mlpack::amf::NMFMultiplicativeDistanceUpdate::NMFMultiplicativeDistanceUpdate () [inline]`

Definition at line 34 of file nmf_mult_dist.hpp.

22.14.3 Member Function Documentation

22.14.3.1 `template<typename MatType > static void mlpack::amf::NMFMultiplicativeDistanceUpdate::HUpdate (const MatType & V, const arma::mat & W, arma::mat & H) [inline],[static]`

The update rule for the encoding matrix H .

The formula used is

$$H_{a\mu} \leftarrow H_{a\mu} \frac{(W^T V)_{a\mu}}{(W^T W H)_{a\mu}}$$

The function takes in all the matrices and only changes the value of the H matrix.

Parameters

V	Input matrix to be factorized.
W	Basis matrix.
H	Encoding matrix to be updated.

Definition at line 76 of file nmf_mult_dist.hpp.

22.14.3.2 `template<typename MatType > void mlpack::amf::NMFMultiplicativeDistanceUpdate::Initialize (const MatType & dataset, const size_t rank) [inline]`

Definition at line 37 of file nmf_mult_dist.hpp.

22.14.3.3 `template<typename MatType > static void mlpack::amf::NMFMultiplicativeDistanceUpdate::WUpdate (const MatType & V, arma::mat & W, const arma::mat & H) [inline],[static]`

The update rule for the basis matrix W .

The formula used is

$$W_{ia} \leftarrow W_{ia} \frac{(V H^T)_{ia}}{(W H H^T)_{ia}}$$

The function takes in all the matrices and only changes the value of the W matrix.

Parameters

V	Input matrix to be factorized.
W	Basis matrix to be updated.
H	Encoding matrix.

Definition at line 56 of file nmf_mult_dist.hpp.

The documentation for this class was generated from the following file:

- src/mlpack/methods/amf/update_rules/nmf_mult_dist.hpp

22.15 mlpack::amf::NMFMultiplicativeDivergenceUpdate Class Reference

This follows a method described in the paper 'Algorithms for Non-negative Matrix Factorization' by D.

Public Member Functions

- **NMFMultiplicativeDivergenceUpdate** ()
- template<typename MatType >
void **Initialize** (const MatType &dataset, const size_t rank)

Static Public Member Functions

- template<typename MatType >
static void **HUpdate** (const MatType &V, const arma::mat &W, arma::mat &H)
The update rule for the encoding matrix H.
- template<typename MatType >
static void **WUpdate** (const MatType &V, arma::mat &W, const arma::mat &H)
The update rule for the basis matrix W.

22.15.1 Detailed Description

This follows a method described in the paper 'Algorithms for Non-negative Matrix Factorization' by D.

D. Lee and H. S. Seung. This is a multiplicative rule that ensures that the Kullback–Leibler divergence $\sum_i \sum_j (V_{ij} \log \frac{V_{ij}}{(WH)_{ij}} - V_{ij} + (WH)_{ij})$ is non-increasing between subsequent iterations. Both of the update rules for W and H are defined in this file.

This set of update rules is not meant to work with sparse matrices. Using sparse matrices often causes NaNs in the output, so other choices of update rules are better in that situation.

Definition at line 34 of file nmf_mult_div.hpp.

22.15.2 Constructor & Destructor Documentation

22.15.2.1 mlpack::amf::NMFMultiplicativeDivergenceUpdate::NMFMultiplicativeDivergenceUpdate () [inline]

Definition at line 38 of file nmf_mult_div.hpp.

22.15.3 Member Function Documentation

22.15.3.1 `template<typename MatType > static void mlpack::amf::NMFMultiplicativeDivergenceUpdate::HUpdate (const MatType & V, const arma::mat & W, arma::mat & H) [inline],[static]`

The update rule for the encoding matrix H.

The formula used is

$$H_{a\mu} \leftarrow H_{a\mu} \frac{\sum_i W_{ia} V_{i\mu} / (WH)_{i\mu}}{\sum_k H_{ka}}$$

The function takes in all the matrices and only changes the value of the H matrix.

Parameters

<i>V</i>	Input matrix to be factorized.
<i>W</i>	Basis matrix.
<i>H</i>	Encoding matrix to updated.

Definition at line 103 of file nmf_mult_div.hpp.

22.15.3.2 `template<typename MatType > void mlpack::amf::NMFMultiplicativeDivergenceUpdate::Initialize (const MatType & dataset, const size_t rank) [inline]`

Definition at line 41 of file nmf_mult_div.hpp.

22.15.3.3 `template<typename MatType > static void mlpack::amf::NMFMultiplicativeDivergenceUpdate::WUpdate (const MatType & V, arma::mat & W, const arma::mat & H) [inline],[static]`

The update rule for the basis matrix W.

The formula used is

$$W_{ia} \leftarrow W_{ia} \frac{\sum_{\mu} H_{a\mu} V_{i\mu} / (WH)_{i\mu}}{\sum_{\nu} H_{a\nu}}$$

The function takes in all the matrices and only changes the value of the W matrix.

Parameters

<i>V</i>	Input matrix to be factorized.
<i>W</i>	Basis matrix to be updated.
<i>H</i>	Encoding matrix.

Definition at line 61 of file nmf_mult_div.hpp.

The documentation for this class was generated from the following file:

- src/mlpack/methods/amf/update_rules/nmf_mult_div.hpp

22.16 mlpack::amf::RandomAcollInitialization< p > Class Template Reference

This class initializes the W matrix of the **AMF** (p. 123) algorithm by averaging p randomly chosen columns of V.

Public Member Functions

- **RandomAcollInitialization** ()

Static Public Member Functions

- `template<typename MatType >`
static void **Initialize** (const MatType &V, const size_t r, arma::mat &W, arma::mat &H)

22.16.1 Detailed Description

`template<int p = 5> class mlpack::amf::RandomAcolInitialization< p >`

This class initializes the W matrix of the **AMF** (p. 123) algorithm by averaging p randomly chosen columns of V .

In this case, p is a template parameter. H is then set randomly. This simple initialization is performed by the random Acol initialization introduced in the paper 'Algorithms, Initializations and Convergence' by Langville et al.

Template Parameters

<i>The</i>	number of random columns to average for each column of W .
------------	--

Definition at line 32 of file `random_acol_init.hpp`.

22.16.2 Constructor & Destructor Documentation

22.16.2.1 `template<int p = 5> mlpack::amf::RandomAcolInitialization< p >::RandomAcolInitialization ()`
[inline]

Definition at line 36 of file `random_acol_init.hpp`.

22.16.3 Member Function Documentation

22.16.3.1 `template<int p = 5> template<typename MatType > static void mlpack::amf::RandomAcolInitialization< p >::Initialize (const MatType & V, const size_t r, arma::mat & W, arma::mat & H)` [inline], [static]

Definition at line 40 of file `random_acol_init.hpp`.

References `mlpack::math::RandInt()`, and `mlpack::Log::Warn`.

The documentation for this class was generated from the following file:

- `src/mlpack/methods/amf/init_rules/random_acol_init.hpp`

22.17 mlpack::amf::RandomInitialization Class Reference

Public Member Functions

- **RandomInitialization** ()

Static Public Member Functions

- `template<typename MatType >`
static void **Initialize** (const MatType &V, const size_t r, arma::mat &W, arma::mat &H)

22.17.1 Detailed Description

Definition at line 23 of file random_init.hpp.

22.17.2 Constructor & Destructor Documentation

22.17.2.1 mpack::amf::RandomInitialization::RandomInitialization () [inline]

Definition at line 27 of file random_init.hpp.

22.17.3 Member Function Documentation

22.17.3.1 template<typename MatType > static void mpack::amf::RandomInitialization::Initialize (const MatType & V, const size_t r, arma::mat & W, arma::mat & H) [inline],[static]

Definition at line 30 of file random_init.hpp.

The documentation for this class was generated from the following file:

- src/mpack/methods/amf/init_rules/**random_init.hpp**

22.18 mpack::amf::SimpleResidueTermination Class Reference

This class implements a simple residue-based termination policy.

Public Member Functions

- **SimpleResidueTermination** (const double **minResidue**=1e-10, const size_t **maxIterations**=10000)
*Construct the **SimpleResidueTermination** (p. 139) object with the given minimum residue (or the default) and the given maximum number of iterations (or the default).*
- const double & **Index** () const
Get current value of residue.
- template<typename MatType >
void **Initialize** (const MatType &V)
Initializes the termination policy before stating the factorization.
- bool **IsConverged** (arma::mat &W, arma::mat &H)
Check if termination criterion is met.
- const size_t & **Iteration** () const
Get current iteration count.
- const size_t & **MaxIterations** () const
Access max iteration count.
- size_t & **MaxIterations** ()
- const double & **MinResidue** () const
Access minimum residue value.
- double & **MinResidue** ()

Public Attributes

- **size_t iteration**
current iteration count
- **size_t maxIterations**
iteration threshold
- **double minResidue**
residue threshold
- **size_t nm**
- **double normOld**
norm of previous iteration
- **double residue**
current value of residue

22.18.1 Detailed Description

This class implements a simple residue-based termination policy.

The termination decision depends on two factors: the value of the residue (the difference between the norm of WH this iteration and the previous iteration), and the number of iterations. If the current value of residue drops below the threshold or the number of iterations goes above the iteration limit, **IsConverged()** (p. 141) will return true. This class is meant for use with the **AMF** (p. 123) (alternating matrix factorization) class.

See also

AMF (p. 123)

Definition at line 33 of file `simple_residue_termination.hpp`.

22.18.2 Constructor & Destructor Documentation

22.18.2.1 `mlpack::amf::SimpleResidueTermination::SimpleResidueTermination (const double minResidue = 1e-10, const size_t maxIterations = 10000) [inline]`

Construct the **SimpleResidueTermination** (p. 139) object with the given minimum residue (or the default) and the given maximum number of iterations (or the default).

0 indicates no iteration limit.

Parameters

<i>minResidue</i>	Minimum residue for termination.
<i>maxIterations</i>	Maximum number of iterations.

Definition at line 44 of file `simple_residue_termination.hpp`.

22.18.3 Member Function Documentation

22.18.3.1 `const double& mlpack::amf::SimpleResidueTermination::Index () const [inline]`

Get current value of residue.

Definition at line 87 of file `simple_residue_termination.hpp`.

References `residue`.

22.18.3.2 `template<typename MatType > void mpack::amf::SimpleResidueTermination::Initialize (const MatType & V)`
`[inline]`

Initializes the termination policy before stating the factorization.

Parameters

V	Input matrix being factorized.
-----	--------------------------------

Definition at line 54 of file simple_residue_termination.hpp.

References iteration, nm, normOld, and residue.

22.18.3.3 `bool mpack::amf::SimpleResidueTermination::IsConverged (arma::mat & W, arma::mat & H)` `[inline]`

Check if termination criterion is met.

Parameters

W	Basis matrix of output.
H	Encoding matrix of output.

Definition at line 70 of file simple_residue_termination.hpp.

References iteration, maxIterations, normOld, and residue.

22.18.3.4 `const size_t& mpack::amf::SimpleResidueTermination::Iteration () const` `[inline]`

Get current iteration count.

Definition at line 90 of file simple_residue_termination.hpp.

References iteration.

22.18.3.5 `const size_t& mpack::amf::SimpleResidueTermination::MaxIterations () const` `[inline]`

Access max iteration count.

Definition at line 93 of file simple_residue_termination.hpp.

References maxIterations.

22.18.3.6 `size_t& mpack::amf::SimpleResidueTermination::MaxIterations ()` `[inline]`

Definition at line 94 of file simple_residue_termination.hpp.

References maxIterations.

22.18.3.7 `const double& mpack::amf::SimpleResidueTermination::MinResidue () const` `[inline]`

Access minimum residue value.

Definition at line 97 of file simple_residue_termination.hpp.

References minResidue.

22.18.3.8 `double& mlpack::amf::SimpleResidueTermination::MinResidue () [inline]`

Definition at line 98 of file `simple_residue_termination.hpp`.

References `minResidue`.

22.18.4 Member Data Documentation

22.18.4.1 `size_t mlpack::amf::SimpleResidueTermination::iteration`

current iteration count

Definition at line 109 of file `simple_residue_termination.hpp`.

Referenced by `Initialize()`, `IsConverged()`, and `Iteration()`.

22.18.4.2 `size_t mlpack::amf::SimpleResidueTermination::maxIterations`

iteration threshold

Definition at line 104 of file `simple_residue_termination.hpp`.

Referenced by `IsConverged()`, and `MaxIterations()`.

22.18.4.3 `double mlpack::amf::SimpleResidueTermination::minResidue`

residue threshold

Definition at line 102 of file `simple_residue_termination.hpp`.

Referenced by `MinResidue()`.

22.18.4.4 `size_t mlpack::amf::SimpleResidueTermination::nm`

Definition at line 113 of file `simple_residue_termination.hpp`.

Referenced by `Initialize()`.

22.18.4.5 `double mlpack::amf::SimpleResidueTermination::normOld`

norm of previous iteration

Definition at line 111 of file `simple_residue_termination.hpp`.

Referenced by `Initialize()`, and `IsConverged()`.

22.18.4.6 `double mlpack::amf::SimpleResidueTermination::residue`

current value of residue

Definition at line 107 of file `simple_residue_termination.hpp`.

Referenced by `Index()`, `Initialize()`, and `IsConverged()`.

The documentation for this class was generated from the following file:

- src/mlpack/methods/amf/termination_policies/simple_residue_termination.hpp

22.19 mlpack::amf::SimpleToleranceTermination< MatType > Class Template Reference

This class implements residue tolerance termination policy.

Public Member Functions

- **SimpleToleranceTermination** (const double **tolerance**=1e-5, const size_t maxIterations=10000, const size_t reverseStepTolerance=3)
empty constructor
- const double & **Index** () const
Get current value of residue.
- void **Initialize** (const MatType &V)
Initializes the termination policy before stating the factorization.
- bool **IsConverged** (arma::mat &W, arma::mat &H)
Check if termination criterio is met.
- const size_t & **Iteration** () const
Get current iteration count.
- const size_t & **MaxIterations** () const
Access upper limit of iteration count.
- size_t & **MaxIterations** ()
- const double & **Tolerance** () const
Access tolerance value.
- double & **Tolerance** ()

Private Attributes

- double **c_index**
- double **c_indexOld**
- arma::mat **H**
- bool **isCopy**
indicates whether a copy of information is available which corresponds to minimum residue point
- size_t **iteration**
current iteration count
- size_t **maxIterations**
iteration threshold
- double **normOld**
- double **residue**
- double **residueOld**
residue values
- size_t **reverseStepCount**
successive residue drops
- size_t **reverseStepTolerance**
tolerance on successive residue drops
- double **tolerance**

- tolerance*
- `const MatType * V`
pointer to matrix being factorized
- `arma::mat W`
variables to store information of minimum residue poi

22.19.1 Detailed Description

```
template<class MatType>class mlpack::amf::SimpleToleranceTermination< MatType >
```

This class implements residue tolerance termination policy.

Termination criterion is met when increase in residue value drops below the given tolerance. To accommodate spikes certain number of successive residue drops are accepted. This upper limit on successive drops can be adjusted with `reverseStepCount`. Secondary termination criterion terminates algorithm when iteration count goes above the threshold.

See also

AMF (p. 123)

Definition at line 33 of file `simple_tolerance_termination.hpp`.

22.19.2 Constructor & Destructor Documentation

22.19.2.1 `template<class MatType > mlpack::amf::SimpleToleranceTermination< MatType >::SimpleToleranceTermination (const double tolerance = 1e-5, const size_t maxIterations = 10000, const size_t reverseStepTolerance = 3) [inline]`

empty constructor

Definition at line 37 of file `simple_tolerance_termination.hpp`.

22.19.3 Member Function Documentation

22.19.3.1 `template<class MatType > const double& mlpack::amf::SimpleToleranceTermination< MatType >::Index () const [inline]`

Get current value of residue.

Definition at line 149 of file `simple_tolerance_termination.hpp`.

References `mlpack::amf::SimpleToleranceTermination< MatType >::residue`.

22.19.3.2 `template<class MatType > void mlpack::amf::SimpleToleranceTermination< MatType >::Initialize (const MatType & V) [inline]`

Initializes the termination policy before stating the factorization.

Parameters

V	Input matrix to be factorized.
-----	--------------------------------

Definition at line 49 of file simple_tolerance_termination.hpp.

References mlpack::amf::SimpleToleranceTermination< MatType >::c_index, mlpack::amf::SimpleToleranceTermination< MatType >::c_indexOld, mlpack::amf::SimpleToleranceTermination< MatType >::isCopy, mlpack::amf::SimpleToleranceTermination< MatType >::iteration, mlpack::amf::SimpleToleranceTermination< MatType >::residue, mlpack::amf::SimpleToleranceTermination< MatType >::residueOld, mlpack::amf::SimpleToleranceTermination< MatType >::reverseStepCount, and mlpack::amf::SimpleToleranceTermination< MatType >::V.

22.19.3.3 `template<class MatType> bool mlpack::amf::SimpleToleranceTermination< MatType >::IsConverged (arma::mat & W, arma::mat & H) [inline]`

Check if termination criterio is met.

Parameters

W	Basis matrix of output.
H	Encoding matrix of output.

Definition at line 71 of file simple_tolerance_termination.hpp.

References mlpack::amf::SimpleToleranceTermination< MatType >::c_index, mlpack::amf::SimpleToleranceTermination< MatType >::c_indexOld, mlpack::amf::SimpleToleranceTermination< MatType >::H, mlpack::amf::SimpleToleranceTermination< MatType >::isCopy, mlpack::amf::SimpleToleranceTermination< MatType >::iteration, mlpack::amf::SimpleToleranceTermination< MatType >::maxIterations, mlpack::amf::SimpleToleranceTermination< MatType >::residue, mlpack::amf::SimpleToleranceTermination< MatType >::residueOld, mlpack::amf::SimpleToleranceTermination< MatType >::reverseStepCount, mlpack::amf::SimpleToleranceTermination< MatType >::reverseStepTolerance, mlpack::amf::SimpleToleranceTermination< MatType >::V, and mlpack::amf::SimpleToleranceTermination< MatType >::W.

22.19.3.4 `template<class MatType> const size_t& mlpack::amf::SimpleToleranceTermination< MatType >::Iteration () const [inline]`

Get current iteration count.

Definition at line 152 of file simple_tolerance_termination.hpp.

References mlpack::amf::SimpleToleranceTermination< MatType >::iteration.

22.19.3.5 `template<class MatType> const size_t& mlpack::amf::SimpleToleranceTermination< MatType >::MaxIterations () const [inline]`

Access upper limit of iteration count.

Definition at line 155 of file simple_tolerance_termination.hpp.

References mlpack::amf::SimpleToleranceTermination< MatType >::maxIterations.

22.19.3.6 `template<class MatType> size_t& mlpack::amf::SimpleToleranceTermination< MatType >::MaxIterations () [inline]`

Definition at line 156 of file simple_tolerance_termination.hpp.

References mlpack::amf::SimpleToleranceTermination< MatType >::maxIterations.

22.19.3.7 `template<class MatType > const double& mlpack::amf::SimpleToleranceTermination< MatType >::Tolerance () const [inline]`

Access tolerance value.

Definition at line 159 of file `simple_tolerance_termination.hpp`.

References `mlpack::amf::SimpleToleranceTermination< MatType >::tolerance`.

22.19.3.8 `template<class MatType > double& mlpack::amf::SimpleToleranceTermination< MatType >::Tolerance () [inline]`

Definition at line 160 of file `simple_tolerance_termination.hpp`.

References `mlpack::amf::SimpleToleranceTermination< MatType >::tolerance`.

22.19.4 Member Data Documentation

22.19.4.1 `template<class MatType > double mlpack::amf::SimpleToleranceTermination< MatType >::c_index [private]`

Definition at line 192 of file `simple_tolerance_termination.hpp`.

Referenced by `mlpack::amf::SimpleToleranceTermination< MatType >::Initialize()`, and `mlpack::amf::SimpleToleranceTermination< MatType >::IsConverged()`.

22.19.4.2 `template<class MatType > double mlpack::amf::SimpleToleranceTermination< MatType >::c_indexOld [private]`

Definition at line 191 of file `simple_tolerance_termination.hpp`.

Referenced by `mlpack::amf::SimpleToleranceTermination< MatType >::Initialize()`, and `mlpack::amf::SimpleToleranceTermination< MatType >::IsConverged()`.

22.19.4.3 `template<class MatType > arma::mat mlpack::amf::SimpleToleranceTermination< MatType >::H [private]`

Definition at line 190 of file `simple_tolerance_termination.hpp`.

Referenced by `mlpack::amf::SimpleToleranceTermination< MatType >::IsConverged()`.

22.19.4.4 `template<class MatType > bool mlpack::amf::SimpleToleranceTermination< MatType >::isCopy [private]`

indicates whether a copy of information is available which corresponds to minimum residue point

Definition at line 186 of file `simple_tolerance_termination.hpp`.

Referenced by `mlpack::amf::SimpleToleranceTermination< MatType >::Initialize()`, and `mlpack::amf::SimpleToleranceTermination< MatType >::IsConverged()`.

22.19.4.5 `template<class MatType > size_t mlpac::amf::SimpleToleranceTermination< MatType >::iteration`
`[private]`

current iteration count

Definition at line 172 of file `simple_tolerance_termination.hpp`.

Referenced by `mlpac::amf::SimpleToleranceTermination< MatType >::Initialize()`, `mlpac::amf::SimpleToleranceTermination< MatType >::IsConverged()`, and `mlpac::amf::SimpleToleranceTermination< MatType >::Iteration()`.

22.19.4.6 `template<class MatType > size_t mlpac::amf::SimpleToleranceTermination< MatType >::maxIterations`
`[private]`

iteration threshold

Definition at line 166 of file `simple_tolerance_termination.hpp`.

Referenced by `mlpac::amf::SimpleToleranceTermination< MatType >::IsConverged()`, and `mlpac::amf::SimpleToleranceTermination< MatType >::MaxIterations()`.

22.19.4.7 `template<class MatType > double mlpac::amf::SimpleToleranceTermination< MatType >::normOld`
`[private]`

Definition at line 177 of file `simple_tolerance_termination.hpp`.

22.19.4.8 `template<class MatType > double mlpac::amf::SimpleToleranceTermination< MatType >::residue`
`[private]`

Definition at line 176 of file `simple_tolerance_termination.hpp`.

Referenced by `mlpac::amf::SimpleToleranceTermination< MatType >::Index()`, `mlpac::amf::SimpleToleranceTermination< MatType >::Initialize()`, and `mlpac::amf::SimpleToleranceTermination< MatType >::IsConverged()`.

22.19.4.9 `template<class MatType > double mlpac::amf::SimpleToleranceTermination< MatType >::residueOld`
`[private]`

residue values

Definition at line 175 of file `simple_tolerance_termination.hpp`.

Referenced by `mlpac::amf::SimpleToleranceTermination< MatType >::Initialize()`, and `mlpac::amf::SimpleToleranceTermination< MatType >::IsConverged()`.

22.19.4.10 `template<class MatType > size_t mlpac::amf::SimpleToleranceTermination< MatType >::reverseStepCount`
`[private]`

successive residue drops

Definition at line 182 of file `simple_tolerance_termination.hpp`.

Referenced by `mlpac::amf::SimpleToleranceTermination< MatType >::Initialize()`, and `mlpac::amf::SimpleToleranceTermination< MatType >::IsConverged()`.

22.19.4.11 `template<class MatType > size_t mlpack::amf::SimpleToleranceTermination< MatType >::reverseStepTolerance [private]`

tolerance on successive residue drops

Definition at line 180 of file `simple_tolerance_termination.hpp`.

Referenced by `mlpack::amf::SimpleToleranceTermination< MatType >::IsConverged()`.

22.19.4.12 `template<class MatType > double mlpack::amf::SimpleToleranceTermination< MatType >::tolerance [private]`

tolerance

Definition at line 164 of file `simple_tolerance_termination.hpp`.

Referenced by `mlpack::amf::SimpleToleranceTermination< MatType >::Tolerance()`.

22.19.4.13 `template<class MatType > const MatType* mlpack::amf::SimpleToleranceTermination< MatType >::V [private]`

pointer to matrix being factorized

Definition at line 169 of file `simple_tolerance_termination.hpp`.

Referenced by `mlpack::amf::SimpleToleranceTermination< MatType >::Initialize()`, and `mlpack::amf::SimpleToleranceTermination< MatType >::IsConverged()`.

22.19.4.14 `template<class MatType > arma::mat mlpack::amf::SimpleToleranceTermination< MatType >::W [private]`

variables to store information of minimum residue poi

Definition at line 189 of file `simple_tolerance_termination.hpp`.

Referenced by `mlpack::amf::SimpleToleranceTermination< MatType >::IsConverged()`.

The documentation for this class was generated from the following file:

- `src/mlpack/methods/amf/termination_policies/simple_tolerance_termination.hpp`

22.20 mlpack::amf::SVDBatchLearning Class Reference

This class implements SVD batch learning with momentum.

Public Member Functions

- **SVDBatchLearning** (double **u**=0.0002, double **kw**=0, double **kh**=0, double **momentum**=0.9, double **min**=-DBL_MIN, double **max**=DBL_MAX)
SVD Batch learning constructor.
- `template<typename MatType >`
void **HUpdate** (const MatType &V, const arma::mat &W, arma::mat &H)
The update rule for the encoding matrix H.

- `template<typename MatType >`
void **Initialize** (const MatType &dataset, const size_t rank)
- `template<typename MatType >`
void **WUpdate** (const MatType &V, arma::mat &W, const arma::mat &H)

The update rule for the basis matrix W.

Private Attributes

- double **kh**
- double **kw**
- double **max**
- arma::mat **mH**
- double **min**
- double **momentum**
- arma::mat **mW**
- double **u**

22.20.1 Detailed Description

This class implements SVD batch learning with momentum.

This procedure is described in the paper 'A Guide to singular Value Decomposition' by Chih-Chao Ma. Class implements 'Algorithm 4' given in the paper. This factorizer decomposes the matrix V into two matrices W and H such that sum of sum of squared error between V and W*H is minimum. This optimization is performed with gradient descent. To make gradient descent faster momentum is added.

Definition at line 29 of file `svd_batch_learning.hpp`.

22.20.2 Constructor & Destructor Documentation

22.20.2.1 `mlpack::amf::SVDBatchLearning::SVDBatchLearning (double u = 0.0002, double kw = 0, double kh = 0, double momentum = 0.9, double min = -DBL_MIN, double max = DBL_MAX) [inline]`

SVD Batch learning constructor.

Parameters

<i>u</i>	step value used in batch learning
<i>kw</i>	regularization constant for W matrix
<i>kh</i>	regularization constant for H matrix
<i>momentum</i>	momentum applied to batch learning process

Definition at line 40 of file `svd_batch_learning.hpp`.

22.20.3 Member Function Documentation

22.20.3.1 `template<typename MatType > void mlpack::amf::SVDBatchLearning::HUpdate (const MatType & V, const arma::mat & W, arma::mat & H) [inline]`

The update rule for the encoding matrix H.

The function takes in all the matrices and only changes the value of the H matrix.

Parameters

V	Input matrix to be factorized.
W	Basis matrix.
H	Encoding matrix to be updated.

Definition at line 109 of file `svd_batch_learning.hpp`.

References `kh`, `mH`, `momentum`, and `u`.

22.20.3.2 `template<typename MatType > void mlpack::amf::SVDBatchLearning::Initialize (const MatType & dataset, const size_t rank) [inline]`

Definition at line 50 of file `svd_batch_learning.hpp`.

References `mH`, and `mW`.

22.20.3.3 `template<typename MatType > void mlpack::amf::SVDBatchLearning::WUpdate (const MatType & V, arma::mat & W, const arma::mat & H) [inline]`

The update rule for the basis matrix W .

The function takes in all the matrices and only changes the value of the W matrix.

Parameters

V	Input matrix to be factorized.
W	Basis matrix to be updated.
H	Encoding matrix.

Definition at line 69 of file `svd_batch_learning.hpp`.

References `kw`, `momentum`, `mW`, and `u`.

22.20.4 Member Data Documentation

22.20.4.1 `double mlpack::amf::SVDBatchLearning::kh [private]`

Definition at line 142 of file `svd_batch_learning.hpp`.

Referenced by `HUpdate()`.

22.20.4.2 `double mlpack::amf::SVDBatchLearning::kw [private]`

Definition at line 141 of file `svd_batch_learning.hpp`.

Referenced by `WUpdate()`.

22.20.4.3 `double mlpack::amf::SVDBatchLearning::max [private]`

Definition at line 144 of file `svd_batch_learning.hpp`.

22.20.4.4 arma::mat mlpack::amf::SVDBatchLearning::mH [private]

Definition at line 148 of file svd_batch_learning.hpp.

Referenced by HUpdate(), and Initialize().

22.20.4.5 double mlpack::amf::SVDBatchLearning::min [private]

Definition at line 143 of file svd_batch_learning.hpp.

22.20.4.6 double mlpack::amf::SVDBatchLearning::momentum [private]

Definition at line 145 of file svd_batch_learning.hpp.

Referenced by HUpdate(), and WUpdate().

22.20.4.7 arma::mat mlpack::amf::SVDBatchLearning::mW [private]

Definition at line 147 of file svd_batch_learning.hpp.

Referenced by Initialize(), and WUpdate().

22.20.4.8 double mlpack::amf::SVDBatchLearning::u [private]

Definition at line 140 of file svd_batch_learning.hpp.

Referenced by HUpdate(), and WUpdate().

The documentation for this class was generated from the following file:

- src/mlpack/methods/amf/update_rules/svd_batch_learning.hpp

22.21 mlpack::amf::SVDCompleteIncrementalLearning< MatType > Class Template Reference

Public Member Functions

- **SVDCompleteIncrementalLearning** (double **u**=0.0001, double **kw**=0, double **kh**=0)
- void **HUpdate** (const MatType &V, const arma::mat &W, arma::mat &H)
The update rule for the encoding matrix H.
- void **Initialize** (const MatType &dataset, const size_t rank)
- void **WUpdate** (const MatType &V, arma::mat &W, const arma::mat &H)
The update rule for the basis matrix W.

Private Attributes

- size_t **currentItemIndex**
- size_t **currentUserIndex**
- double **kh**

- double **kw**
- size_t **m**
- size_t **n**
- double **u**

22.21.1 Detailed Description

template<class MatType>class mlpack::amf::SVDCompleteIncrementalLearning< MatType >

Definition at line 12 of file svd_complete_incremental_learning.hpp.

22.21.2 Constructor & Destructor Documentation

22.21.2.1 template<class MatType > mlpack::amf::SVDCompleteIncrementalLearning< MatType >::SVDCompleteIncrementalLearning(double *u* = 0.0001, double *kw* = 0, double *kh* = 0) [inline]

Definition at line 15 of file svd_complete_incremental_learning.hpp.

22.21.3 Member Function Documentation

22.21.3.1 template<class MatType > void mlpack::amf::SVDCompleteIncrementalLearning< MatType >::HUpdate (const MatType & *V*, const arma::mat & *W*, arma::mat & *H*) [inline]

The update rule for the encoding matrix *H*.

The function takes in all the matrices and only changes the value of the *H* matrix.

Parameters

<i>V</i>	Input matrix to be factorized.
<i>W</i>	Basis matrix.
<i>H</i>	Encoding matrix to be updated.

Definition at line 83 of file svd_complete_incremental_learning.hpp.

References mlpack::amf::SVDCompleteIncrementalLearning< MatType >::currentItemIndex, mlpack::amf::SVDCompleteIncrementalLearning< MatType >::currentUserIndex, mlpack::amf::SVDCompleteIncrementalLearning< MatType >::kh, mlpack::amf::SVDCompleteIncrementalLearning< MatType >::m, mlpack::amf::SVDCompleteIncrementalLearning< MatType >::n, and mlpack::amf::SVDCompleteIncrementalLearning< MatType >::u.

22.21.3.2 template<class MatType > void mlpack::amf::SVDCompleteIncrementalLearning< MatType >::Initialize (const MatType & *dataset*, const size_t *rank*) [inline]

Definition at line 21 of file svd_complete_incremental_learning.hpp.

References mlpack::amf::SVDCompleteIncrementalLearning< MatType >::currentItemIndex, mlpack::amf::SVDCompleteIncrementalLearning< MatType >::currentUserIndex, mlpack::amf::SVDCompleteIncrementalLearning< MatType >::m, and mlpack::amf::SVDCompleteIncrementalLearning< MatType >::n.

22.21.3.3 `template<class MatType > void mlpack::amf::SVDCompleteIncrementalLearning< MatType >::WUpdate (const MatType & V, arma::mat & W, const arma::mat & H) [inline]`

The update rule for the basis matrix W.

The function takes in all the matrices and only changes the value of the W matrix.

Parameters

<i>V</i>	Input matrix to be factorized.
<i>W</i>	Basis matrix to be updated.
<i>H</i>	Encoding matrix.

This file is part of mlpack 1.0.12.

mlpack is free software; you may redistribute it and/or modify it under the terms of the 3-clause BSD license. You should have received a copy of the 3-clause BSD license along with mlpack. If not, see <http://www.opensource.org/licenses/BSD-3-Clause> for more information.

Definition at line 47 of file `svd_complete_incremental_learning.hpp`.

References `mlpack::amf::SVDCompleteIncrementalLearning< MatType >::currentItemIndex`, `mlpack::amf::SVDCompleteIncrementalLearning< MatType >::currentUserIndex`, `mlpack::amf::SVDCompleteIncrementalLearning< MatType >::kw`, `mlpack::amf::SVDCompleteIncrementalLearning< MatType >::m`, `mlpack::amf::SVDCompleteIncrementalLearning< MatType >::n`, and `mlpack::amf::SVDCompleteIncrementalLearning< MatType >::u`.

22.21.4 Member Data Documentation

22.21.4.1 `template<class MatType > size_t mlpack::amf::SVDCompleteIncrementalLearning< MatType >::currentItemIndex [private]`

Definition at line 118 of file `svd_complete_incremental_learning.hpp`.

Referenced by `mlpack::amf::SVDCompleteIncrementalLearning< MatType >::HUpdate()`, `mlpack::amf::SVDCompleteIncrementalLearning< arma::sp_mat >::HUpdate()`, `mlpack::amf::SVDCompleteIncrementalLearning< MatType >::Initialize()`, `mlpack::amf::SVDCompleteIncrementalLearning< MatType >::WUpdate()`, and `mlpack::amf::SVDCompleteIncrementalLearning< arma::sp_mat >::WUpdate()`.

22.21.4.2 `template<class MatType > size_t mlpack::amf::SVDCompleteIncrementalLearning< MatType >::currentUserIndex [private]`

Definition at line 117 of file `svd_complete_incremental_learning.hpp`.

Referenced by `mlpack::amf::SVDCompleteIncrementalLearning< MatType >::HUpdate()`, `mlpack::amf::SVDCompleteIncrementalLearning< arma::sp_mat >::HUpdate()`, `mlpack::amf::SVDCompleteIncrementalLearning< MatType >::Initialize()`, `mlpack::amf::SVDCompleteIncrementalLearning< MatType >::WUpdate()`, and `mlpack::amf::SVDCompleteIncrementalLearning< arma::sp_mat >::WUpdate()`.

22.21.4.3 `template<class MatType > double mlpack::amf::SVDCompleteIncrementalLearning< MatType >::kh [private]`

Definition at line 112 of file `svd_complete_incremental_learning.hpp`.

Referenced by `mlpack::amf::SVDCompleteIncrementalLearning< MatType >::HUpdate()`, and `mlpack::amf::SVDCompleteIncrementalLearning< arma::sp_mat >::HUpdate()`.

22.21.4.4 `template<class MatType > double mlpack::amf::SVDCompleteIncrementalLearning< MatType >::kw`
`[private]`

Definition at line 111 of file `svd_complete_incremental_learning.hpp`.

Referenced by `mlpack::amf::SVDCompleteIncrementalLearning< MatType >::WUpdate()`, and `mlpack::amf::SVDCompleteIncrementalLearning< arma::sp_mat >::WUpdate()`.

22.21.4.5 `template<class MatType > size_t mlpack::amf::SVDCompleteIncrementalLearning< MatType >::m`
`[private]`

Definition at line 115 of file `svd_complete_incremental_learning.hpp`.

Referenced by `mlpack::amf::SVDCompleteIncrementalLearning< MatType >::HUpdate()`, `mlpack::amf::SVDCompleteIncrementalLearning< MatType >::Initialize()`, `mlpack::amf::SVDCompleteIncrementalLearning< arma::sp_mat >::Initialize()`, and `mlpack::amf::SVDCompleteIncrementalLearning< MatType >::WUpdate()`.

22.21.4.6 `template<class MatType > size_t mlpack::amf::SVDCompleteIncrementalLearning< MatType >::n`
`[private]`

Definition at line 114 of file `svd_complete_incremental_learning.hpp`.

Referenced by `mlpack::amf::SVDCompleteIncrementalLearning< MatType >::HUpdate()`, `mlpack::amf::SVDCompleteIncrementalLearning< MatType >::Initialize()`, `mlpack::amf::SVDCompleteIncrementalLearning< arma::sp_mat >::Initialize()`, and `mlpack::amf::SVDCompleteIncrementalLearning< MatType >::WUpdate()`.

22.21.4.7 `template<class MatType > double mlpack::amf::SVDCompleteIncrementalLearning< MatType >::u`
`[private]`

Definition at line 110 of file `svd_complete_incremental_learning.hpp`.

Referenced by `mlpack::amf::SVDCompleteIncrementalLearning< MatType >::HUpdate()`, `mlpack::amf::SVDCompleteIncrementalLearning< arma::sp_mat >::HUpdate()`, `mlpack::amf::SVDCompleteIncrementalLearning< MatType >::WUpdate()`, and `mlpack::amf::SVDCompleteIncrementalLearning< arma::sp_mat >::WUpdate()`.

The documentation for this class was generated from the following file:

- `src/mlpack/methods/amf/update_rules/svd_complete_incremental_learning.hpp`

22.22 `mlpack::amf::SVDCompleteIncrementalLearning< arma::sp_mat >` Class Template Reference

Public Member Functions

- **SVDCompleteIncrementalLearning** (double `u`=0.01, double `kw`=0, double `kh`=0)
- **~SVDCompleteIncrementalLearning** ()
- void **HUpdate** (const arma::sp_mat &V, const arma::mat &W, arma::mat &H)
The update rule for the encoding matrix H.
- void **Initialize** (const arma::sp_mat &dataset, const size_t rank)
- void **WUpdate** (const arma::sp_mat &V, arma::mat &W, const arma::mat &H)
The update rule for the basis matrix W.

Private Attributes

- arma::sp_mat **dummy**
- bool **isStart**
- arma::sp_mat::const_iterator * **it**
- double **kh**
- double **kw**
- size_t **m**
- size_t **n**
- double **u**

22.22.1 Detailed Description

template<> class mlpack::amf::SVDCompleteIncrementalLearning< arma::sp_mat >

Definition at line 122 of file svd_complete_incremental_learning.hpp.

22.22.2 Constructor & Destructor Documentation

22.22.2.1 mlpack::amf::SVDCompleteIncrementalLearning< arma::sp_mat >::SVDCompleteIncrementalLearning(double *u* = 0.01, double *kw* = 0, double *kh* = 0) [inline]

Definition at line 125 of file svd_complete_incremental_learning.hpp.

22.22.2.2 mlpack::amf::SVDCompleteIncrementalLearning< arma::sp_mat >::~~SVDCompleteIncrementalLearning() [inline]

Definition at line 131 of file svd_complete_incremental_learning.hpp.

22.22.3 Member Function Documentation

22.22.3.1 void mlpack::amf::SVDCompleteIncrementalLearning< arma::sp_mat >::HUpdate(const arma::sp_mat & *V*, const arma::mat & *W*, arma::mat & *H*) [inline]

The update rule for the encoding matrix *H*.

The function takes in all the matrices and only changes the value of the *H* matrix.

Parameters

<i>V</i>	Input matrix to be factorized.
<i>W</i>	Basis matrix.
<i>H</i>	Encoding matrix to be updated.

Definition at line 190 of file svd_complete_incremental_learning.hpp.

References mlpack::amf::SVDCompleteIncrementalLearning< MatType >::currentItemIndex, mlpack::amf::SVDCompleteIncrementalLearning< MatType >::currentUserIndex, mlpack::amf::SVDCompleteIncrementalLearning< MatType >::kh, and mlpack::amf::SVDCompleteIncrementalLearning< MatType >::u.

22.22.3.2 `void mlpack::amf::SVDCompleteIncrementalLearning< arma::sp_mat >::Initialize (const arma::sp_mat & dataset, const size_t rank) [inline]`

Definition at line 136 of file `svd_complete_incremental_learning.hpp`.

References `mlpack::amf::SVDCompleteIncrementalLearning< MatType >::m`, and `mlpack::amf::SVDCompleteIncrementalLearning< MatType >::n`.

22.22.3.3 `void mlpack::amf::SVDCompleteIncrementalLearning< arma::sp_mat >::WUpdate (const arma::sp_mat & V, arma::mat & W, const arma::mat & H) [inline]`

The update rule for the basis matrix W .

The function takes in all the matrices and only changes the value of the W matrix.

Parameters

V	Input matrix to be factorized.
W	Basis matrix to be updated.
H	Encoding matrix.

Definition at line 155 of file `svd_complete_incremental_learning.hpp`.

References `mlpack::amf::SVDCompleteIncrementalLearning< MatType >::currentItemIndex`, `mlpack::amf::SVDCompleteIncrementalLearning< MatType >::currentUserIndex`, `mlpack::amf::SVDCompleteIncrementalLearning< MatType >::kw`, and `mlpack::amf::SVDCompleteIncrementalLearning< MatType >::u`.

22.22.4 Member Data Documentation

22.22.4.1 `arma::sp_mat mlpack::amf::SVDCompleteIncrementalLearning< arma::sp_mat >::dummy [private]`

Definition at line 217 of file `svd_complete_incremental_learning.hpp`.

22.22.4.2 `bool mlpack::amf::SVDCompleteIncrementalLearning< arma::sp_mat >::isStart [private]`

Definition at line 220 of file `svd_complete_incremental_learning.hpp`.

22.22.4.3 `arma::sp_mat::const_iterator* mlpack::amf::SVDCompleteIncrementalLearning< arma::sp_mat >::it [private]`

Definition at line 218 of file `svd_complete_incremental_learning.hpp`.

22.22.4.4 `double mlpack::amf::SVDCompleteIncrementalLearning< arma::sp_mat >::kh [private]`

Definition at line 212 of file `svd_complete_incremental_learning.hpp`.

22.22.4.5 `double mlpack::amf::SVDCompleteIncrementalLearning< arma::sp_mat >::kw [private]`

Definition at line 211 of file `svd_complete_incremental_learning.hpp`.

22.22.4.6 `size_t mlpack::amf::SVDCompleteIncrementalLearning< arma::sp_mat >::m` [private]

Definition at line 215 of file `svd_complete_incremental_learning.hpp`.

22.22.4.7 `size_t mlpack::amf::SVDCompleteIncrementalLearning< arma::sp_mat >::n` [private]

Definition at line 214 of file `svd_complete_incremental_learning.hpp`.

22.22.4.8 `double mlpack::amf::SVDCompleteIncrementalLearning< arma::sp_mat >::u` [private]

Definition at line 210 of file `svd_complete_incremental_learning.hpp`.

The documentation for this class was generated from the following file:

- `src/mlpack/methods/amf/update_rules/svd_complete_incremental_learning.hpp`

22.23 mlpack::amf::SVDIncompleteIncrementalLearning Class Reference

Public Member Functions

- **SVDIncompleteIncrementalLearning** (double **u**=0.001, double **kw**=0, double **kh**=0)
- `template<typename MatType >`
void **HUpdate** (const MatType &V, const arma::mat &W, arma::mat &H)
The update rule for the encoding matrix H.
- `template<typename MatType >`
void **Initialize** (const MatType &dataset, const size_t rank)
- `template<typename MatType >`
void **WUpdate** (const MatType &V, arma::mat &W, const arma::mat &H)
The update rule for the basis matrix W.

Private Attributes

- `size_t` **currentUserIndex**
- `double` **kh**
- `double` **kw**
- `size_t` **m**
- `size_t` **n**
- `double` **u**

22.23.1 Detailed Description

Definition at line 8 of file `svd_incomplete_incremental_learning.hpp`.

22.23.2 Constructor & Destructor Documentation

22.23.2.1 `mlpack::amf::SVDIncompleteIncrementalLearning::SVDIncompleteIncrementalLearning (double u = 0.001, double kw = 0, double kh = 0)` [inline]

Definition at line 11 of file `svd_incomplete_incremental_learning.hpp`.

22.23.3 Member Function Documentation

22.23.3.1 `template<typename MatType > void mlpack::amf::SVDIncompleteIncrementalLearning::HUpdate (const MatType & V, const arma::mat & W, arma::mat & H) [inline]`

The update rule for the encoding matrix *H*.

The function takes in all the matrices and only changes the value of the *H* matrix.

Parameters

<i>V</i>	Input matrix to be factorized.
<i>W</i>	Basis matrix.
<i>H</i>	Encoding matrix to be updated.

Definition at line 73 of file `svd_incomplete_incremental_learning.hpp`.

References `currentUserIndex`, `kh`, `m`, `n`, and `u`.

22.23.3.2 `template<typename MatType > void mlpack::amf::SVDIncompleteIncrementalLearning::Initialize (const MatType & dataset, const size_t rank) [inline]`

Definition at line 18 of file `svd_incomplete_incremental_learning.hpp`.

References `currentUserIndex`, `m`, and `n`.

22.23.3.3 `template<typename MatType > void mlpack::amf::SVDIncompleteIncrementalLearning::WUpdate (const MatType & V, arma::mat & W, const arma::mat & H) [inline]`

The update rule for the basis matrix *W*.

The function takes in all the matrices and only changes the value of the *W* matrix.

Parameters

<i>V</i>	Input matrix to be factorized.
<i>W</i>	Basis matrix to be updated.
<i>H</i>	Encoding matrix.

This file is part of mlpack 1.0.12.

mlpack is free software; you may redistribute it and/or modify it under the terms of the 3-clause BSD license. You should have received a copy of the 3-clause BSD license along with mlpack. If not, see <http://www.opensource.org/licenses/BSD-3-Clause> for more information.

Definition at line 45 of file `svd_incomplete_incremental_learning.hpp`.

References `currentUserIndex`, `kw`, `n`, and `u`.

22.23.4 Member Data Documentation

22.23.4.1 `size_t mlpack::amf::SVDIncompleteIncrementalLearning::currentUserIndex [private]`

Definition at line 101 of file `svd_incomplete_incremental_learning.hpp`.

Referenced by `HUpdate()`, `Initialize()`, and `WUpdate()`.

22.23.4.2 `double mlpack::amf::SVDIncompleteIncrementalLearning::kh` `[private]`

Definition at line 96 of file `svd_incomplete_incremental_learning.hpp`.

Referenced by `HUpdate()`.

22.23.4.3 `double mlpack::amf::SVDIncompleteIncrementalLearning::kw` `[private]`

Definition at line 95 of file `svd_incomplete_incremental_learning.hpp`.

Referenced by `WUpdate()`.

22.23.4.4 `size_t mlpack::amf::SVDIncompleteIncrementalLearning::m` `[private]`

Definition at line 99 of file `svd_incomplete_incremental_learning.hpp`.

Referenced by `HUpdate()`, and `Initialize()`.

22.23.4.5 `size_t mlpack::amf::SVDIncompleteIncrementalLearning::n` `[private]`

Definition at line 98 of file `svd_incomplete_incremental_learning.hpp`.

Referenced by `HUpdate()`, `Initialize()`, and `WUpdate()`.

22.23.4.6 `double mlpack::amf::SVDIncompleteIncrementalLearning::u` `[private]`

Definition at line 94 of file `svd_incomplete_incremental_learning.hpp`.

Referenced by `HUpdate()`, and `WUpdate()`.

The documentation for this class was generated from the following file:

- `src/mlpack/methods/amf/update_rules/svd_incomplete_incremental_learning.hpp`

22.24 mlpack::amf::ValidationRMSETermination< MatType > Class Template Reference

Public Member Functions

- **ValidationRMSETermination** (MatType &V, size_t **num_test_points**, double **tolerance**=1e-5, size_t max←Iterations=10000, size_t reverseStepTolerance=3)
- const double & **Index** ()
- void **Initialize** (const MatType &)
- bool **IsConverged** (arma::mat &W, arma::mat &H)
- const size_t & **Iteration** ()
- const size_t & **MaxIterations** ()

Private Attributes

- double **c_index**
- double **c_indexOld**
- arma::mat **H**

- bool **isCopy**
- size_t **iteration**
- size_t **maxIterations**
- size_t **num_test_points**
- size_t **reverseStepCount**
- size_t **reverseStepTolerance**
- double **rmse**
- double **rmseOld**
- arma::Mat< double > **test_points**
- double **tolerance**
- arma::mat **W**

22.24.1 Detailed Description

template<class MatType>class mlpack::amf::ValidationRMSETermination< MatType >

Definition at line 24 of file validation_RMSE_termination.hpp.

22.24.2 Constructor & Destructor Documentation

22.24.2.1 template<class MatType > mlpack::amf::ValidationRMSETermination< MatType >::ValidationRMSETermination (MatType & V, size_t num_test_points, double tolerance = 1e-5, size_t maxIterations = 10000, size_t reverseStepTolerance = 3) [inline]

Definition at line 27 of file validation_RMSE_termination.hpp.

References mlpack::amf::ValidationRMSETermination< MatType >::num_test_points, and mlpack::amf::ValidationRMSETermination< MatType >::test_points.

22.24.3 Member Function Documentation

22.24.3.1 template<class MatType > const double& mlpack::amf::ValidationRMSETermination< MatType >::Index () [inline]

Definition at line 134 of file validation_RMSE_termination.hpp.

References mlpack::amf::ValidationRMSETermination< MatType >::rmse.

22.24.3.2 template<class MatType > void mlpack::amf::ValidationRMSETermination< MatType >::Initialize (const MatType &) [inline]

Definition at line 60 of file validation_RMSE_termination.hpp.

References mlpack::amf::ValidationRMSETermination< MatType >::c_index, mlpack::amf::ValidationRMSETermination< MatType >::c_indexOld, mlpack::amf::ValidationRMSETermination< MatType >::isCopy, mlpack::amf::ValidationRMSETermination< MatType >::iteration, mlpack::amf::ValidationRMSETermination< MatType >::reverseStepCount, mlpack::amf::ValidationRMSETermination< MatType >::rmse, and mlpack::amf::ValidationRMSETermination< MatType >::rmseOld.

22.24.3.3 `template<class MatType > bool mlpack::amf::ValidationRMSETermination< MatType >::IsConverged (arma::mat & W, arma::mat & H) [inline]`

Definition at line 74 of file validation_RMSE_termination.hpp.

References mlpack::amf::ValidationRMSETermination< MatType >::c_index, mlpack::amf::ValidationRMSETermination< MatType >::c_indexOld, mlpack::amf::ValidationRMSETermination< MatType >::H, mlpack::amf::ValidationRMSETermination< MatType >::isCopy, mlpack::amf::ValidationRMSETermination< MatType >::iteration, mlpack::amf::ValidationRMSETermination< MatType >::maxIterations, mlpack::amf::ValidationRMSETermination< MatType >::num_test_points, mlpack::amf::ValidationRMSETermination< MatType >::reverseStepCount, mlpack::amf::ValidationRMSETermination< MatType >::reverseStepTolerance, mlpack::amf::ValidationRMSETermination< MatType >::rmse, mlpack::amf::ValidationRMSETermination< MatType >::rmseOld, mlpack::amf::ValidationRMSETermination< MatType >::test_points, and mlpack::amf::ValidationRMSETermination< MatType >::W.

22.24.3.4 `template<class MatType > const size_t& mlpack::amf::ValidationRMSETermination< MatType >::Iteration () [inline]`

Definition at line 136 of file validation_RMSE_termination.hpp.

References mlpack::amf::ValidationRMSETermination< MatType >::iteration.

22.24.3.5 `template<class MatType > const size_t& mlpack::amf::ValidationRMSETermination< MatType >::MaxIterations () [inline]`

Definition at line 138 of file validation_RMSE_termination.hpp.

References mlpack::amf::ValidationRMSETermination< MatType >::maxIterations.

22.24.4 Member Data Documentation

22.24.4.1 `template<class MatType > double mlpack::amf::ValidationRMSETermination< MatType >::c_index [private]`

Definition at line 158 of file validation_RMSE_termination.hpp.

Referenced by mlpack::amf::ValidationRMSETermination< MatType >::Initialize(), and mlpack::amf::ValidationRMSETermination< MatType >::IsConverged().

22.24.4.2 `template<class MatType > double mlpack::amf::ValidationRMSETermination< MatType >::c_indexOld [private]`

Definition at line 157 of file validation_RMSE_termination.hpp.

Referenced by mlpack::amf::ValidationRMSETermination< MatType >::Initialize(), and mlpack::amf::ValidationRMSETermination< MatType >::IsConverged().

22.24.4.3 `template<class MatType > arma::mat mlpack::amf::ValidationRMSETermination< MatType >::H [private]`

Definition at line 156 of file validation_RMSE_termination.hpp.

Referenced by mlpack::amf::ValidationRMSETermination< MatType >::IsConverged().

22.24.4.4 `template<class MatType > bool mlpack::amf::ValidationRMSETermination< MatType >::isCopy`
`[private]`

Definition at line 154 of file validation_RMSE_termination.hpp.

Referenced by `mlpack::amf::ValidationRMSETermination< MatType >::Initialize()`, and `mlpack::amf::ValidationRMSETermination< MatType >::IsConverged()`.

22.24.4.5 `template<class MatType > size_t mlpack::amf::ValidationRMSETermination< MatType >::iteration`
`[private]`

Definition at line 144 of file validation_RMSE_termination.hpp.

Referenced by `mlpack::amf::ValidationRMSETermination< MatType >::Initialize()`, `mlpack::amf::ValidationRMSETermination< MatType >::IsConverged()`, and `mlpack::amf::ValidationRMSETermination< MatType >::Iteration()`.

22.24.4.6 `template<class MatType > size_t mlpack::amf::ValidationRMSETermination< MatType >::maxIterations`
`[private]`

Definition at line 142 of file validation_RMSE_termination.hpp.

Referenced by `mlpack::amf::ValidationRMSETermination< MatType >::IsConverged()`, and `mlpack::amf::ValidationRMSETermination< MatType >::MaxIterations()`.

22.24.4.7 `template<class MatType > size_t mlpack::amf::ValidationRMSETermination< MatType >::num_test_points`
`[private]`

Definition at line 143 of file validation_RMSE_termination.hpp.

Referenced by `mlpack::amf::ValidationRMSETermination< MatType >::IsConverged()`, and `mlpack::amf::ValidationRMSETermination< MatType >::ValidationRMSETermination()`.

22.24.4.8 `template<class MatType > size_t mlpack::amf::ValidationRMSETermination< MatType >::reverseStepCount`
`[private]`

Definition at line 152 of file validation_RMSE_termination.hpp.

Referenced by `mlpack::amf::ValidationRMSETermination< MatType >::Initialize()`, and `mlpack::amf::ValidationRMSETermination< MatType >::IsConverged()`.

22.24.4.9 `template<class MatType > size_t mlpack::amf::ValidationRMSETermination< MatType >::reverseStepTolerance`
`[private]`

Definition at line 151 of file validation_RMSE_termination.hpp.

Referenced by `mlpack::amf::ValidationRMSETermination< MatType >::IsConverged()`.

22.24.4.10 `template<class MatType > double mlpack::amf::ValidationRMSETermination< MatType >::rmse`
`[private]`

Definition at line 149 of file validation_RMSE_termination.hpp.

Referenced by mlpack::amf::ValidationRMSETermination< MatType >::Index(), mlpack::amf::ValidationRMSETermination< MatType >::Initialize(), and mlpack::amf::ValidationRMSETermination< MatType >::IsConverged().

22.24.4.11 `template<class MatType> double mlpack::amf::ValidationRMSETermination< MatType >::rmseOld`
[private]

Definition at line 148 of file validation_RMSE_termination.hpp.

Referenced by mlpack::amf::ValidationRMSETermination< MatType >::Initialize(), and mlpack::amf::ValidationRMSETermination< MatType >::IsConverged().

22.24.4.12 `template<class MatType> arma::Mat<double> mlpack::amf::ValidationRMSETermination< MatType >::test_points` [private]

Definition at line 146 of file validation_RMSE_termination.hpp.

Referenced by mlpack::amf::ValidationRMSETermination< MatType >::IsConverged(), and mlpack::amf::ValidationRMSETermination< MatType >::ValidationRMSETermination().

22.24.4.13 `template<class MatType> double mlpack::amf::ValidationRMSETermination< MatType >::tolerance`
[private]

Definition at line 141 of file validation_RMSE_termination.hpp.

22.24.4.14 `template<class MatType> arma::mat mlpack::amf::ValidationRMSETermination< MatType >::W`
[private]

Definition at line 155 of file validation_RMSE_termination.hpp.

Referenced by mlpack::amf::ValidationRMSETermination< MatType >::IsConverged().

The documentation for this class was generated from the following file:

- src/mlpack/methods/amf/termination_policies/**validation_RMSE_termination.hpp**

22.25 mlpack::bound::BallBound< VecType, TMetricType > Class Template Reference

Ball bound encloses a set of points at a specific distance (radius) from a specific point (center).

Public Types

- typedef TMetricType **MetricType**
Need this for Binary Space Partition Tree.
- typedef VecType **Vec**

Public Member Functions

- **BallBound** ()
Empty Constructor.

- **BallBound** (const size_t dimension)
Create the ball bound with the specified dimensionality.
- **BallBound** (const double radius, const VecType ¢er)
Create the ball bound with the specified radius and center.
- **BallBound** (const **BallBound** &other)
Copy constructor. To prevent memory leaks.
- **~BallBound** ()
Destructor to release allocated memory.
- const VecType & **Center** () const
Get the center point of the ball.
- VecType & **Center** ()
Modify the center point of the ball.
- void **Centroid** (VecType ¢roid) const
*Place the centroid of **BallBound** (p. 163) into the given vector.*
- bool **Contains** (const VecType &point) const
Determines if a point is within this bound.
- double **Diameter** () const
Returns the diameter of the ballbound.
- double **Dim** () const
Get the dimensionality of the ball.
- template<typename OtherVecType >
double **MaxDistance** (const OtherVecType &point, typename boost::enable_if< **IsVector**< OtherVecType > >
*=0) const
Computes maximum distance.
- double **MaxDistance** (const **BallBound** &other) const
Computes maximum distance.
- TMetricType **Metric** () const
Returns the distance metric used in this bound.
- template<typename OtherVecType >
double **MinDistance** (const OtherVecType &point, typename boost::enable_if< **IsVector**< OtherVecType > >
*=0) const
Calculates minimum bound-to-point squared distance.
- double **MinDistance** (const **BallBound** &other) const
Calculates minimum bound-to-bound squared distance.
- double **MinWidth** () const
Get the minimum width of the bound (this is same as the diameter).
- **BallBound** & **operator=** (const **BallBound** &other)
For the same reason as the Copy Constructor. To prevent memory leaks.
- math::Range **operator[]** (const size_t i) const
Get the range in a certain dimension.
- const **BallBound** & **operator|**= (const **BallBound** &other)
Expand the bound to include the given node.
- template<typename MatType >
const **BallBound** & **operator|**= (const MatType &data)
Expand the bound to include the given point.
- double **Radius** () const
Get the radius of the ball.
- double & **Radius** ()

Modify the radius of the ball.

- `template<typename OtherVecType > math::Range RangeDistance (const OtherVecType &other, typename boost::enable_if< IsVector< OtherVecType > > * = 0) const`

Calculates minimum and maximum bound-to-point distance.

- `math::Range RangeDistance (const BallBound &other) const`

Calculates minimum and maximum bound-to-bound distance.

- `std::string ToString () const`

Returns a string representation of this object.

Private Attributes

- `VecType center`

The center of the ball bound.

- `TMetricType * metric`

The metric used in this bound.

- `bool ownsMetric`

To know whether this object allocated memory to the metric member variable.

- `double radius`

The radius of the ball bound.

22.25.1 Detailed Description

```
template<typename VecType = arma::vec, typename TMetricType = metric::LMetric<2, true>>
class mlpack::bound::BallBound<
VecType, TMetricType >
```

Ball bound encloses a set of points at a specific distance (radius) from a specific point (center).

TMetricType is the custom metric type that defaults to the Euclidean (L2) distance.

Template Parameters

<i>VecType</i>	Type of vector (arma::vec or arma::sp_vec).
<i>TMetricType</i>	metric type used in the distance measure.

Definition at line 34 of file ballbound.hpp.

22.25.2 Member Typedef Documentation

22.25.2.1 `template<typename VecType = arma::vec, typename TMetricType = metric::LMetric<2, true>> typedef TMetricType mlpack::bound::BallBound< VecType, TMetricType >::MetricType`

Need this for Binary Space Partition Tree.

Definition at line 39 of file ballbound.hpp.

22.25.2.2 `template<typename VecType = arma::vec, typename TMetricType = metric::LMetric<2, true>> typedef VecType mlpack::bound::BallBound< VecType, TMetricType >::Vec`

Definition at line 37 of file ballbound.hpp.

22.25.3 Constructor & Destructor Documentation

22.25.3.1 `template<typename VecType = arma::vec, typename TMetricType = metric::LMetric<2, true>>
mlpack::bound::BallBound< VecType, TMetricType >::BallBound ()`

Empty Constructor.

22.25.3.2 `template<typename VecType = arma::vec, typename TMetricType = metric::LMetric<2, true>>
mlpack::bound::BallBound< VecType, TMetricType >::BallBound (const size_t dimension)`

Create the ball bound with the specified dimensionality.

Parameters

<i>dimension</i>	Dimensionality of ball bound.
------------------	-------------------------------

22.25.3.3 `template<typename VecType = arma::vec, typename TMetricType = metric::LMetric<2, true>>
mlpack::bound::BallBound< VecType, TMetricType >::BallBound (const double radius, const VecType & center
)`

Create the ball bound with the specified radius and center.

Parameters

<i>radius</i>	Radius of ball bound.
<i>center</i>	Center of ball bound.

22.25.3.4 `template<typename VecType = arma::vec, typename TMetricType = metric::LMetric<2, true>>
mlpack::bound::BallBound< VecType, TMetricType >::BallBound (const BallBound< VecType, TMetricType
> & other)`

Copy constructor. To prevent memory leaks.

22.25.3.5 `template<typename VecType = arma::vec, typename TMetricType = metric::LMetric<2, true>>
mlpack::bound::BallBound< VecType, TMetricType >::~~BallBound ()`

Destructor to release allocated memory.

22.25.4 Member Function Documentation

22.25.4.1 `template<typename VecType = arma::vec, typename TMetricType = metric::LMetric<2, true>> const VecType&
mlpack::bound::BallBound< VecType, TMetricType >::Center () const [inline]`

Get the center point of the ball.

Definition at line 95 of file ballbound.hpp.

References `mlpack::bound::BallBound< VecType, TMetricType >::center`.

22.25.4.2 `template<typename VecType = arma::vec, typename TMetricType = metric::LMetric<2, true>> VecType& mlpack::bound::BallBound< VecType, TMetricType >::Center () [inline]`

Modify the center point of the ball.

Definition at line 97 of file ballbound.hpp.

References mlpack::bound::BallBound< VecType, TMetricType >::center.

22.25.4.3 `template<typename VecType = arma::vec, typename TMetricType = metric::LMetric<2, true>> void mlpack::bound::BallBound< VecType, TMetricType >::Centroid (VecType & centroid) const [inline]`

Place the centroid of **BallBound** (p. 163) into the given vector.

Parameters

<i>centroid</i>	Vector which the centroid will be written to.
-----------------	---

Definition at line 121 of file ballbound.hpp.

References mlpack::bound::BallBound< VecType, TMetricType >::center.

22.25.4.4 `template<typename VecType = arma::vec, typename TMetricType = metric::LMetric<2, true>> bool mlpack::bound::BallBound< VecType, TMetricType >::Contains (const VecType & point) const`

Determines if a point is within this bound.

22.25.4.5 `template<typename VecType = arma::vec, typename TMetricType = metric::LMetric<2, true>> double mlpack::bound::BallBound< VecType, TMetricType >::Diameter () const [inline]`

Returns the diameter of the ballbound.

Definition at line 183 of file ballbound.hpp.

References mlpack::bound::BallBound< VecType, TMetricType >::radius.

22.25.4.6 `template<typename VecType = arma::vec, typename TMetricType = metric::LMetric<2, true>> double mlpack::bound::BallBound< VecType, TMetricType >::Dim () const [inline]`

Get the dimensionality of the ball.

Definition at line 100 of file ballbound.hpp.

22.25.4.7 `template<typename VecType = arma::vec, typename TMetricType = metric::LMetric<2, true>> template<typename OtherVecType > double mlpack::bound::BallBound< VecType, TMetricType >::MaxDistance (const OtherVecType & point, typename boost::enable_if< IsVector< OtherVecType > > * = 0) const`

Computes maximum distance.

22.25.4.8 `template<typename VecType = arma::vec, typename TMetricType = metric::LMetric<2, true>> double mlpack::bound::BallBound< VecType, TMetricType >::MaxDistance (const BallBound< VecType, TMetricType > & other) const`

Computes maximum distance.

22.25.4.9 `template<typename VecType = arma::vec, typename TMetricType = metric::LMetric<2, true>> TMetricType
mlpack::bound::BallBound< VecType, TMetricType >::Metric () const [inline]`

Returns the distance metric used in this bound.

Definition at line 188 of file ballbound.hpp.

References `mlpack::bound::BallBound< VecType, TMetricType >::metric`.

22.25.4.10 `template<typename VecType = arma::vec, typename TMetricType = metric::LMetric<2, true>> template<typename
OtherVecType > double mlpack::bound::BallBound< VecType, TMetricType >::MinDistance (const OtherVecType
& point, typename boost::enable_if< IsVector< OtherVecType > > * = 0) const`

Calculates minimum bound-to-point squared distance.

22.25.4.11 `template<typename VecType = arma::vec, typename TMetricType = metric::LMetric<2, true>> double
mlpack::bound::BallBound< VecType, TMetricType >::MinDistance (const BallBound< VecType, TMetricType
> & other) const`

Calculates minimum bound-to-bound squared distance.

22.25.4.12 `template<typename VecType = arma::vec, typename TMetricType = metric::LMetric<2, true>> double
mlpack::bound::BallBound< VecType, TMetricType >::MinWidth () const [inline]`

Get the minimum width of the bound (this is same as the diameter).

For ball bounds, width along all dimensions remain same.

Definition at line 106 of file ballbound.hpp.

22.25.4.13 `template<typename VecType = arma::vec, typename TMetricType = metric::LMetric<2, true>> BallBound&
mlpack::bound::BallBound< VecType, TMetricType >::operator= (const BallBound< VecType, TMetricType >
& other)`

For the same reason as the Copy Constructor. To prevent memory leaks.

22.25.4.14 `template<typename VecType = arma::vec, typename TMetricType = metric::LMetric<2, true>> math::Range
mlpack::bound::BallBound< VecType, TMetricType >::operator[] (const size_t i) const`

Get the range in a certain dimension.

22.25.4.15 `template<typename VecType = arma::vec, typename TMetricType = metric::LMetric<2, true>> const BallBound&
mlpack::bound::BallBound< VecType, TMetricType >::operator|= (const BallBound< VecType, TMetricType >
& other)`

Expand the bound to include the given node.

22.25.4.16 `template<typename VecType = arma::vec, typename TMetricType = metric::LMetric<2, true>> template<typename MatType > const BallBound& mlpack::bound::BallBound< VecType, TMetricType >::operator|=(const MatType & data)`

Expand the bound to include the given point.

The centroid is recalculated to be the center of all of the given points.

Template Parameters

<i>MatType</i>	Type of matrix; could be arma::mat, arma::spmat, or a vector.
<i>data</i>	Data points to add.

22.25.4.17 `template<typename VecType = arma::vec, typename TMetricType = metric::LMetric<2, true>> double mlpack::bound::BallBound< VecType, TMetricType >::Radius () const [inline]`

Get the radius of the ball.

Definition at line 90 of file ballbound.hpp.

References mlpack::bound::BallBound< VecType, TMetricType >::radius.

22.25.4.18 `template<typename VecType = arma::vec, typename TMetricType = metric::LMetric<2, true>> double& mlpack::bound::BallBound< VecType, TMetricType >::Radius () [inline]`

Modify the radius of the ball.

Definition at line 92 of file ballbound.hpp.

References mlpack::bound::BallBound< VecType, TMetricType >::radius.

22.25.4.19 `template<typename VecType = arma::vec, typename TMetricType = metric::LMetric<2, true>> template<typename OtherVecType > math::Range mlpack::bound::BallBound< VecType, TMetricType >::RangeDistance (const OtherVecType & other, typename boost::enable_if< IsVector< OtherVecType > > * = 0) const`

Calculates minimum and maximum bound-to-point distance.

22.25.4.20 `template<typename VecType = arma::vec, typename TMetricType = metric::LMetric<2, true>> math::Range mlpack::bound::BallBound< VecType, TMetricType >::RangeDistance (const BallBound< VecType, TMetricType > & other) const`

Calculates minimum and maximum bound-to-bound distance.

Example: bound1.MinDistanceSq(other) for minimum distance.

22.25.4.21 `template<typename VecType = arma::vec, typename TMetricType = metric::LMetric<2, true>> std::string mlpack::bound::BallBound< VecType, TMetricType >::ToString () const`

Returns a string representation of this object.

22.25.5 Member Data Documentation

22.25.5.1 `template<typename VecType = arma::vec, typename TMetricType = metric::LMetric<2, true>> VecType
mlpack::bound::BallBound< VecType, TMetricType >::center [private]`

The center of the ball bound.

Definition at line 47 of file ballbound.hpp.

Referenced by `mlpack::bound::BallBound< VecType, TMetricType >::Center()`, and `mlpack::bound::BallBound< VecType, TMetricType >::Centroid()`.

22.25.5.2 `template<typename VecType = arma::vec, typename TMetricType = metric::LMetric<2, true>> TMetricType*
mlpack::bound::BallBound< VecType, TMetricType >::metric [private]`

The metric used in this bound.

Definition at line 50 of file ballbound.hpp.

Referenced by `mlpack::bound::BallBound< VecType, TMetricType >::Metric()`.

22.25.5.3 `template<typename VecType = arma::vec, typename TMetricType = metric::LMetric<2, true>> bool
mlpack::bound::BallBound< VecType, TMetricType >::ownsMetric [private]`

To know whether this object allocated memory to the metric member variable.

This will be true except in the copy constructor and the overloaded assignment operator. We need this to know whether we should delete the metric member variable in the destructor.

Definition at line 58 of file ballbound.hpp.

22.25.5.4 `template<typename VecType = arma::vec, typename TMetricType = metric::LMetric<2, true>> double
mlpack::bound::BallBound< VecType, TMetricType >::radius [private]`

The radius of the ball bound.

Definition at line 44 of file ballbound.hpp.

Referenced by `mlpack::bound::BallBound< VecType, TMetricType >::Diameter()`, and `mlpack::bound::BallBound< VecType, TMetricType >::Radius()`.

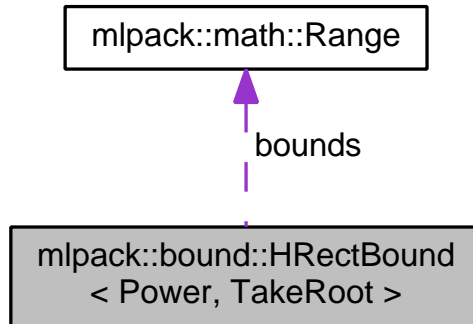
The documentation for this class was generated from the following file:

- `src/mlpack/core/tree/ballbound.hpp`

22.26 `mlpack::bound::HRectBound< Power, TakeRoot >` Class Template Reference

Hyper-rectangle bound for an L-metric.

Collaboration diagram for mpack::bound::HRectBound< Power, TakeRoot >:



Public Types

- typedef **metric::LMetric**< Power, TakeRoot > **MetricType**
This is the metric type that this bound is using.

Public Member Functions

- **HRectBound** ()
Empty constructor; creates a bound of dimensionality 0.
- **HRectBound** (const size_t dimension)
Initializes to specified dimensionality with each dimension the empty set.
- **HRectBound** (const **HRectBound** &other)
Copy constructor; necessary to prevent memory leaks.
- **~HRectBound** ()
Destructor: clean up memory.
- void **Centroid** (arma::vec ¢roid) const
Calculates the centroid of the range, placing it into the given vector.
- void **Clear** ()
Resets all dimensions to the empty set (so that this bound contains nothing).
- template<typename VecType >
bool **Contains** (const VecType &point) const
Determines if a point is within this bound.
- double **Diameter** () const
Returns the diameter of the hyperrectangle (that is, the longest diagonal).
- size_t **Dim** () const
Gets the dimensionality.
- template<typename VecType >
double **MaxDistance** (const VecType &point, typename boost::enable_if< **IsVector**< VecType > > *=0) const
Calculates maximum bound-to-point squared distance.
- double **MaxDistance** (const **HRectBound** &other) const
Computes maximum distance.
- template<typename VecType >
double **MinDistance** (const VecType &point, typename boost::enable_if< **IsVector**< VecType > > *=0) const
Calculates minimum bound-to-point distance.

- double **MinDistance** (const **HRectBound** &other) const
Calculates minimum bound-to-bound distance.
- double **MinWidth** () const
Get the minimum width of the bound.
- double & **MinWidth** ()
Modify the minimum width of the bound.
- **HRectBound** & **operator=** (const **HRectBound** &other)
Same as copy constructor; necessary to prevent memory leaks.
- **math::Range** & **operator[]** (const size_t i)
Get the range for a particular dimension.
- const **math::Range** & **operator[]** (const size_t i) const
Modify the range for a particular dimension. No bounds checking.
- template<typename MatType >
HRectBound & **operator|=** (const MatType &data)
Expands this region to include new points.
- **HRectBound** & **operator|=** (const **HRectBound** &other)
Expands this region to encompass another bound.
- **math::Range RangeDistance** (const **HRectBound** &other) const
Calculates minimum and maximum bound-to-bound distance.
- template<typename VecType >
math::Range RangeDistance (const VecType &point, typename boost::enable_if< **IsVector**< VecType > > *=0) const
Calculates minimum and maximum bound-to-point distance.
- std::string **ToString** () const
Returns a string representation of this object.

Static Public Member Functions

- static **MetricType Metric** ()
Return the metric associated with this bound.

Private Attributes

- **math::Range * bounds**
The bounds for each dimension.
- size_t **dim**
The dimensionality of the bound.
- double **minWidth**
Cached minimum width of bound.

22.26.1 Detailed Description

```
template<int Power = 2, bool TakeRoot = true>class mlpack::bound::HRectBound< Power, TakeRoot >
```

Hyper-rectangle bound for an L-metric.

This should be used in conjunction with the **LMetric** class. Be sure to use the same template parameters for **LMetric** as you do for **HRectBound** (p. 170) – otherwise odd results may occur.

Template Parameters

<i>Power</i>	The metric to use; use 2 for Euclidean (L2).
<i>TakeRoot</i>	Whether or not the root should be taken (see LMetric documentation).

Definition at line 36 of file hrectbound.hpp.

22.26.2 Member Typedef Documentation

22.26.2.1 `template<int Power = 2, bool TakeRoot = true> typedef metric::LMetric<Power, TakeRoot>
mlpack::bound::HRectBound< Power, TakeRoot >::MetricType`

This is the metric type that this bound is using.

Definition at line 40 of file hrectbound.hpp.

22.26.3 Constructor & Destructor Documentation

22.26.3.1 `template<int Power = 2, bool TakeRoot = true> mlpack::bound::HRectBound< Power, TakeRoot >::HRectBound
()`

Empty constructor; creates a bound of dimensionality 0.

22.26.3.2 `template<int Power = 2, bool TakeRoot = true> mlpack::bound::HRectBound< Power, TakeRoot >::HRectBound
(const size_t dimension)`

Initializes to specified dimensionality with each dimension the empty set.

22.26.3.3 `template<int Power = 2, bool TakeRoot = true> mlpack::bound::HRectBound< Power, TakeRoot >::HRectBound
(const HRectBound< Power, TakeRoot > & other)`

Copy constructor; necessary to prevent memory leaks.

22.26.3.4 `template<int Power = 2, bool TakeRoot = true> mlpack::bound::HRectBound< Power, TakeRoot
>::~~HRectBound()`

Destructor: clean up memory.

22.26.4 Member Function Documentation

22.26.4.1 `template<int Power = 2, bool TakeRoot = true> void mlpack::bound::HRectBound< Power, TakeRoot >::Centroid (arma::vec & centroid) const`

Calculates the centroid of the range, placing it into the given vector.

Parameters

<i>centroid</i>	Vector which the centroid will be written to.
-----------------	---

22.26.4.2 `template<int Power = 2, bool TakeRoot = true> void mpack::bound::HRectBound< Power, TakeRoot >::Clear ()`

Resets all dimensions to the empty set (so that this bound contains nothing).

22.26.4.3 `template<int Power = 2, bool TakeRoot = true> template<typename VecType > bool mpack::bound::HRectBound< Power, TakeRoot >::Contains (const VecType & point) const`

Determines if a point is within this bound.

22.26.4.4 `template<int Power = 2, bool TakeRoot = true> double mpack::bound::HRectBound< Power, TakeRoot >::Diameter () const`

Returns the diameter of the hyperrectangle (that is, the longest diagonal).

22.26.4.5 `template<int Power = 2, bool TakeRoot = true> size_t mpack::bound::HRectBound< Power, TakeRoot >::Dim () const [inline]`

Gets the dimensionality.

Definition at line 68 of file hrectbound.hpp.

References `mpack::bound::HRectBound< Power, TakeRoot >::dim`.

22.26.4.6 `template<int Power = 2, bool TakeRoot = true> template<typename VecType > double mpack::bound::HRectBound< Power, TakeRoot >::MaxDistance (const VecType & point, typename boost::enable_if< IsVector< VecType > > * = 0) const`

Calculates maximum bound-to-point squared distance.

Parameters

<i>point</i>	Point to which the maximum distance is requested.
--------------	---

22.26.4.7 `template<int Power = 2, bool TakeRoot = true> double mpack::bound::HRectBound< Power, TakeRoot >::MaxDistance (const HRectBound< Power, TakeRoot > & other) const`

Computes maximum distance.

Parameters

<i>other</i>	Bound to which the maximum distance is requested.
--------------	---

22.26.4.8 `template<int Power = 2, bool TakeRoot = true> static MetricType mpack::bound::HRectBound< Power, TakeRoot >::Metric () [inline],[static]`

Return the metric associated with this bound.

Because it is an LMetric, it cannot store state, so we can make it on the fly. It is also static because the metric is only dependent on the template arguments.

Definition at line 175 of file hrectbound.hpp.

```
22.26.4.9  template<int Power = 2, bool TakeRoot = true> template<typename VecType > double mpack::bound::HRectBound< Power, TakeRoot >::MinDistance ( const VecType & point, typename boost::enable_if< IsVector< VecType > > * = 0 ) const
```

Calculates minimum bound-to-point distance.

Parameters

<i>point</i>	Point to which the minimum distance is requested.
--------------	---

```
22.26.4.10  template<int Power = 2, bool TakeRoot = true> double mpack::bound::HRectBound< Power, TakeRoot >::MinDistance ( const HRectBound< Power, TakeRoot > & other ) const
```

Calculates minimum bound-to-bound distance.

Parameters

<i>other</i>	Bound to which the minimum distance is requested.
--------------	---

```
22.26.4.11  template<int Power = 2, bool TakeRoot = true> double mpack::bound::HRectBound< Power, TakeRoot >::MinWidth ( ) const [inline]
```

Get the minimum width of the bound.

Definition at line 77 of file hrectbound.hpp.

References mpack::bound::HRectBound< Power, TakeRoot >::minWidth.

```
22.26.4.12  template<int Power = 2, bool TakeRoot = true> double& mpack::bound::HRectBound< Power, TakeRoot >::MinWidth ( ) [inline]
```

Modify the minimum width of the bound.

Definition at line 79 of file hrectbound.hpp.

References mpack::bound::HRectBound< Power, TakeRoot >::minWidth.

```
22.26.4.13  template<int Power = 2, bool TakeRoot = true> HRectBound& mpack::bound::HRectBound< Power, TakeRoot >::operator= ( const HRectBound< Power, TakeRoot > & other )
```

Same as copy constructor; necessary to prevent memory leaks.

```
22.26.4.14  template<int Power = 2, bool TakeRoot = true> math::Range& mpack::bound::HRectBound< Power, TakeRoot >::operator[] ( const size_t i ) [inline]
```

Get the range for a particular dimension.

No bounds checking. Be careful: this may make **MinWidth()** (p. 175) invalid.

Definition at line 72 of file hrectbound.hpp.

References `mlpack::bound::HRectBound< Power, TakeRoot >::bounds`.

22.26.4.15 `template<int Power = 2, bool TakeRoot = true> const math::Range& mlpack::bound::HRectBound< Power, TakeRoot >::operator[](const size_t i) const [inline]`

Modify the range for a particular dimension. No bounds checking.

Definition at line 74 of file hrectbound.hpp.

References `mlpack::bound::HRectBound< Power, TakeRoot >::bounds`.

22.26.4.16 `template<int Power = 2, bool TakeRoot = true> template<typename MatType > HRectBound& mlpack::bound::HRectBound< Power, TakeRoot >::operator|= (const MatType & data)`

Expands this region to include new points.

Template Parameters

<i>MatType</i>	Type of matrix; could be <code>Mat</code> , <code>SpMat</code> , a subview, or just a vector.
----------------	---

Parameters

<i>data</i>	Data points to expand this region to include.
-------------	---

22.26.4.17 `template<int Power = 2, bool TakeRoot = true> HRectBound& mlpack::bound::HRectBound< Power, TakeRoot >::operator|= (const HRectBound< Power, TakeRoot > & other)`

Expands this region to encompass another bound.

22.26.4.18 `template<int Power = 2, bool TakeRoot = true> math::Range mlpack::bound::HRectBound< Power, TakeRoot >::RangeDistance (const HRectBound< Power, TakeRoot > & other) const`

Calculates minimum and maximum bound-to-bound distance.

Parameters

<i>other</i>	Bound to which the minimum and maximum distances are requested.
--------------	---

22.26.4.19 `template<int Power = 2, bool TakeRoot = true> template<typename VecType > math::Range mlpack::bound::HRectBound< Power, TakeRoot >::RangeDistance (const VecType & point, typename boost::enable_if< IsVector< VecType > > * = 0) const`

Calculates minimum and maximum bound-to-point distance.

Parameters

<i>point</i>	Point to which the minimum and maximum distances are requested.
--------------	---

22.26.4.20 `template<int Power = 2, bool TakeRoot = true> std::string mlpack::bound::HRectBound< Power, TakeRoot >::ToString () const`

Returns a string representation of this object.

22.26.5 Member Data Documentation

22.26.5.1 `template<int Power = 2, bool TakeRoot = true> math::Range* mlpack::bound::HRectBound< Power, TakeRoot >::bounds [private]`

The bounds for each dimension.

Definition at line 181 of file `hrectbound.hpp`.

Referenced by `mlpack::bound::HRectBound< Power, TakeRoot >::operator[]()`.

22.26.5.2 `template<int Power = 2, bool TakeRoot = true> size_t mlpack::bound::HRectBound< Power, TakeRoot >::dim [private]`

The dimensionality of the bound.

Definition at line 179 of file `hrectbound.hpp`.

Referenced by `mlpack::bound::HRectBound< Power, TakeRoot >::Dim()`.

22.26.5.3 `template<int Power = 2, bool TakeRoot = true> double mlpack::bound::HRectBound< Power, TakeRoot >::minWidth [private]`

Cached minimum width of bound.

Definition at line 183 of file `hrectbound.hpp`.

Referenced by `mlpack::bound::HRectBound< Power, TakeRoot >::MinWidth()`.

The documentation for this class was generated from the following file:

- `src/mlpack/core/tree/hrectbound.hpp`

22.27 mlpack::cf::CF< FactorizerType > Class Template Reference

This class implements Collaborative Filtering (**CF** (p. 177)).

Public Member Functions

- **CF** (`arma::mat &data`, `const size_t numUsersForSimilarity=5`, `const size_t rank=0`)
*Initialize the **CF** (p. 177) object.*
- `const arma::sp_mat & CleanedData () const`
Get the cleaned data matrix.
- `const arma::mat & Data () const`
Get the data matrix.
- `void Factorizer (const FactorizerType &f)`

- Sets factorizer for NMF.*
 - void **GetRecommendations** (const size_t numRecs, arma::Mat< size_t > &recommendations)
 - Generates the given number of recommendations for all users.*
 - void **GetRecommendations** (const size_t numRecs, arma::Mat< size_t > &recommendations, arma::Col< size_t > &users)
 - Generates the given number of recommendations for the specified users.*
 - const arma::mat & **H** () const
 - Get the Item Matrix.*
 - void **NumUsersForSimilarity** (const size_t num)
 - Sets number of users for calculating similarity.*
 - size_t **NumUsersForSimilarity** () const
 - Gets number of users for calculating similarity.*
 - void **Rank** (const size_t rankValue)
 - Sets rank parameter for matrix factorization.*
 - size_t **Rank** () const
 - Gets rank parameter for matrix factorization.*
 - const arma::mat & **Rating** () const
 - Get the Rating Matrix.*
 - std::string **ToString** () const
 - Returns a string representation of this object.*
 - const arma::mat & **W** () const
 - Get the User Matrix.*

Private Member Functions

- void **CleanData** ()
 - Converts the User, Item, Value Matrix to User-Item Table.*
- void **InsertNeighbor** (const size_t queryIndex, const size_t pos, const size_t neighbor, const double value, arma::Mat< size_t > &recommendations, arma::mat &values) const
 - Helper function to insert a point into the recommendation matrices.*

Private Attributes

- arma::sp_mat **cleanedData**
 - Cleaned data matrix.*
- arma::mat **data**
 - Initial data matrix.*
- FactorizerType **factorizer**
 - Instantiated factorizer object.*
- arma::mat **h**
 - Item matrix.*
- size_t **numUsersForSimilarity**
 - Number of users for similarity.*
- size_t **rank**
 - Rank used for matrix factorization.*
- arma::mat **rating**
 - Rating matrix.*
- arma::mat **w**
 - User matrix.*

22.27.1 Detailed Description

```
template<typename FactorizerType = amf::AMF<amf::SimpleResidueTermination, amf::RandomInitialization, amf::NMFALSUpdate>> class mlpack::cf::CF< FactorizerType >
```

This class implements Collaborative Filtering (**CF** (p. 177)).

This implementation presently supports Alternating Least Squares (ALS) for collaborative filtering.

A simple example of how to run Collaborative Filtering is shown below.

```
extern arma::mat data; // (user, item, rating) table
extern arma::Col<size_t> users; // users seeking recommendations
arma::Mat<size_t> recommendations; // Recommendations

CF<> cf(data); // Default options.

// Generate 10 recommendations for all users.
cf.GetRecommendations(10, recommendations);

// Generate 10 recommendations for specified users.
cf.GetRecommendations(10, recommendations, users);
```

The data matrix is a (user, item, rating) table. Each column in the matrix should have three rows. The first represents the user; the second represents the item; and the third represents the rating. The user and item, while they are in a matrix that holds doubles, should hold integer (or size_t) values. The user and item indices are assumed to start at 0.

Template Parameters

<i>FactorizerType</i>	The type of matrix factorization to use to decompose the rating matrix (a W and H matrix). This must implement the method Apply(arma::sp_mat& data, size_t rank, arma::mat& W, arma::mat& H).
-----------------------	---

Definition at line 68 of file cf.hpp.

22.27.2 Constructor & Destructor Documentation

```
22.27.2.1 template<typename FactorizerType = amf::AMF<amf::SimpleResidueTermination, amf::RandomInitialization, amf::NMFALSUpdate>> mlpack::cf::CF< FactorizerType >::CF ( arma::mat & data, const size_t numUsersForSimilarity = 5, const size_t rank = 0 )
```

Initialize the **CF** (p. 177) object.

Store a reference to the data that we will be using. There are parameters that can be set; default values are provided for each of them. If the rank is left unset (or is set to 0), a simple density-based heuristic will be used to choose a rank.

Parameters

<i>data</i>	Initial (user, item, rating) matrix.
<i>numUsersForSimilarity</i>	Size of the neighborhood.
<i>rank</i>	Rank parameter for matrix factorization.

22.27.3 Member Function Documentation

```
22.27.3.1 template<typename FactorizerType = amf::AMF<amf::SimpleResidueTermination, amf::RandomInitialization, amf::NMFALSUpdate>> void mlpack::cf::CF< FactorizerType >::CleanData ( ) [private]
```

Converts the User, Item, Value Matrix to User-Item Table.

22.27.3.2 `template<typename FactorizerType = amf::AMF<amf::SimpleResidueTermination, amf::RandomInitialization, amf::NMFALSUpdate>> const arma::sp_mat& mlpack::cf::CF<FactorizerType>::CleanedData () const [inline]`

Get the cleaned data matrix.

Definition at line 130 of file cf.hpp.

References `mlpack::cf::CF<FactorizerType>::cleanedData`.

22.27.3.3 `template<typename FactorizerType = amf::AMF<amf::SimpleResidueTermination, amf::RandomInitialization, amf::NMFALSUpdate>> const arma::mat& mlpack::cf::CF<FactorizerType>::Data () const [inline]`

Get the data matrix.

Definition at line 128 of file cf.hpp.

References `mlpack::cf::CF<FactorizerType>::data`.

22.27.3.4 `template<typename FactorizerType = amf::AMF<amf::SimpleResidueTermination, amf::RandomInitialization, amf::NMFALSUpdate>> void mlpack::cf::CF<FactorizerType>::Factorizer (const FactorizerType & f) [inline]`

Sets factorizer for NMF.

Definition at line 116 of file cf.hpp.

References `mlpack::cf::CF<FactorizerType>::factorizer`.

22.27.3.5 `template<typename FactorizerType = amf::AMF<amf::SimpleResidueTermination, amf::RandomInitialization, amf::NMFALSUpdate>> void mlpack::cf::CF<FactorizerType>::GetRecommendations (const size_t numRecs, arma::Mat<size_t> & recommendations)`

Generates the given number of recommendations for all users.

Parameters

<i>numRecs</i>	Number of Recommendations
<i>recommendations</i>	Matrix to save recommendations into.

22.27.3.6 `template<typename FactorizerType = amf::AMF<amf::SimpleResidueTermination, amf::RandomInitialization, amf::NMFALSUpdate>> void mlpack::cf::CF<FactorizerType>::GetRecommendations (const size_t numRecs, arma::Mat<size_t> & recommendations, arma::Col<size_t> & users)`

Generates the given number of recommendations for the specified users.

Parameters

<i>numRecs</i>	Number of Recommendations
<i>recommendations</i>	Matrix to save recommendations

<i>users</i>	Users for which recommendations are to be generated
--------------	---

22.27.3.7 `template<typename FactorizerType = amf::AMF<amf::SimpleResidueTermination, amf::RandomInitialization, amf::NMFALSUpdate>> const arma::mat& mlpack::cf::CF< FactorizerType >::H () const [inline]`

Get the Item Matrix.

Definition at line 124 of file cf.hpp.

References mlpack::cf::CF< FactorizerType >::h.

22.27.3.8 `template<typename FactorizerType = amf::AMF<amf::SimpleResidueTermination, amf::RandomInitialization, amf::NMFALSUpdate>> void mlpack::cf::CF< FactorizerType >::InsertNeighbor (const size_t queryIndex, const size_t pos, const size_t neighbor, const double value, arma::Mat< size_t > & recommendations, arma::mat & values) const [private]`

Helper function to insert a point into the recommendation matrices.

Parameters

<i>queryIndex</i>	Index of point whose recommendations we are inserting into.
<i>pos</i>	Position in list to insert into.
<i>neighbor</i>	Index of item being inserted as a recommendation.
<i>value</i>	Value of recommendation.

22.27.3.9 `template<typename FactorizerType = amf::AMF<amf::SimpleResidueTermination, amf::RandomInitialization, amf::NMFALSUpdate>> void mlpack::cf::CF< FactorizerType >::NumUsersForSimilarity (const size_t num) [inline]`

Sets number of users for calculating similarity.

Definition at line 86 of file cf.hpp.

References mlpack::cf::CF< FactorizerType >::numUsersForSimilarity, and mlpack::Log::Warn.

22.27.3.10 `template<typename FactorizerType = amf::AMF<amf::SimpleResidueTermination, amf::RandomInitialization, amf::NMFALSUpdate>> size_t mlpack::cf::CF< FactorizerType >::NumUsersForSimilarity () const [inline]`

Gets number of users for calculating similarity.

Definition at line 98 of file cf.hpp.

References mlpack::cf::CF< FactorizerType >::numUsersForSimilarity.

22.27.3.11 `template<typename FactorizerType = amf::AMF<amf::SimpleResidueTermination, amf::RandomInitialization, amf::NMFALSUpdate>> void mlpack::cf::CF< FactorizerType >::Rank (const size_t rankValue) [inline]`

Sets rank parameter for matrix factorization.

Definition at line 104 of file cf.hpp.

References mlpack::cf::CF< FactorizerType >::rank.

22.27.3.12 `template<typename FactorizerType = amf::AMF<amf::SimpleResidueTermination, amf::RandomInitialization, amf::NMFALSUpdate>> size_t mlpack::cf::CF<FactorizerType>::Rank () const [inline]`

Gets rank parameter for matrix factorization.

Definition at line 110 of file cf.hpp.

References mlpack::cf::CF<FactorizerType>::rank.

22.27.3.13 `template<typename FactorizerType = amf::AMF<amf::SimpleResidueTermination, amf::RandomInitialization, amf::NMFALSUpdate>> const arma::mat& mlpack::cf::CF<FactorizerType>::Rating () const [inline]`

Get the Rating Matrix.

Definition at line 126 of file cf.hpp.

References mlpack::cf::CF<FactorizerType>::rating.

22.27.3.14 `template<typename FactorizerType = amf::AMF<amf::SimpleResidueTermination, amf::RandomInitialization, amf::NMFALSUpdate>> std::string mlpack::cf::CF<FactorizerType>::ToString () const`

Returns a string representation of this object.

22.27.3.15 `template<typename FactorizerType = amf::AMF<amf::SimpleResidueTermination, amf::RandomInitialization, amf::NMFALSUpdate>> const arma::mat& mlpack::cf::CF<FactorizerType>::W () const [inline]`

Get the User Matrix.

Definition at line 122 of file cf.hpp.

References mlpack::cf::CF<FactorizerType>::w.

22.27.4 Member Data Documentation

22.27.4.1 `template<typename FactorizerType = amf::AMF<amf::SimpleResidueTermination, amf::RandomInitialization, amf::NMFALSUpdate>> arma::sp_mat mlpack::cf::CF<FactorizerType>::cleanedData [private]`

Cleaned data matrix.

Definition at line 173 of file cf.hpp.

Referenced by mlpack::cf::CF<FactorizerType>::CleanedData().

22.27.4.2 `template<typename FactorizerType = amf::AMF<amf::SimpleResidueTermination, amf::RandomInitialization, amf::NMFALSUpdate>> arma::mat mlpack::cf::CF<FactorizerType>::data [private]`

Initial data matrix.

Definition at line 159 of file cf.hpp.

Referenced by mlpack::cf::CF<FactorizerType>::Data().

22.27.4.3 `template<typename FactorizerType = amf::AMF<amf::SimpleResidueTermination, amf::RandomInitialization, amf::NMFALSUpdate>> FactorizerType mlpack::cf::CF< FactorizerType >::factorizer [private]`

Instantiated factorizer object.

Definition at line 165 of file cf.hpp.

Referenced by `mlpack::cf::CF< FactorizerType >::Factorizer()`.

22.27.4.4 `template<typename FactorizerType = amf::AMF<amf::SimpleResidueTermination, amf::RandomInitialization, amf::NMFALSUpdate>> arma::mat mlpack::cf::CF< FactorizerType >::h [private]`

Item matrix.

Definition at line 169 of file cf.hpp.

Referenced by `mlpack::cf::CF< FactorizerType >::H()`.

22.27.4.5 `template<typename FactorizerType = amf::AMF<amf::SimpleResidueTermination, amf::RandomInitialization, amf::NMFALSUpdate>> size_t mlpack::cf::CF< FactorizerType >::numUsersForSimilarity [private]`

Number of users for similarity.

Definition at line 161 of file cf.hpp.

Referenced by `mlpack::cf::CF< FactorizerType >::NumUsersForSimilarity()`.

22.27.4.6 `template<typename FactorizerType = amf::AMF<amf::SimpleResidueTermination, amf::RandomInitialization, amf::NMFALSUpdate>> size_t mlpack::cf::CF< FactorizerType >::rank [private]`

Rank used for matrix factorization.

Definition at line 163 of file cf.hpp.

Referenced by `mlpack::cf::CF< FactorizerType >::Rank()`.

22.27.4.7 `template<typename FactorizerType = amf::AMF<amf::SimpleResidueTermination, amf::RandomInitialization, amf::NMFALSUpdate>> arma::mat mlpack::cf::CF< FactorizerType >::rating [private]`

Rating matrix.

Definition at line 171 of file cf.hpp.

Referenced by `mlpack::cf::CF< FactorizerType >::Rating()`.

22.27.4.8 `template<typename FactorizerType = amf::AMF<amf::SimpleResidueTermination, amf::RandomInitialization, amf::NMFALSUpdate>> arma::mat mlpack::cf::CF< FactorizerType >::w [private]`

User matrix.

Definition at line 167 of file cf.hpp.

Referenced by `mlpack::cf::CF< FactorizerType >::W()`.

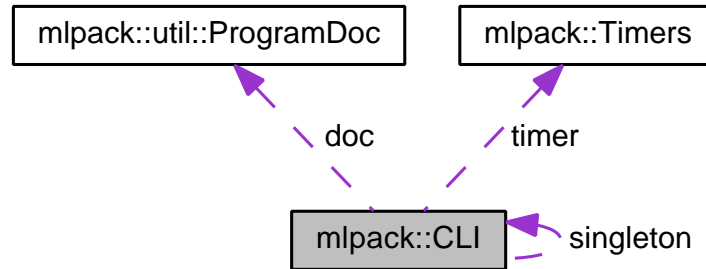
The documentation for this class was generated from the following file:

- `src/mlpack/methods/cf/cf.hpp`

22.28 mlpack::CLI Class Reference

Parses the command line for parameters and holds user-specified parameters.

Collaboration diagram for mlpack::CLI:



Public Member Functions

- `~CLI()`
Destructor.

Static Public Member Functions

- static void **Add** (const std::string &path, const std::string &description, const std::string &alias="", bool required=false)
Adds a parameter to the hierarchy; use the `PARAM_`() macros instead of this (i.e.*
- template<class T >
static void **Add** (const std::string &identifier, const std::string &description, const std::string &alias="", bool required=false)
Adds a parameter to the hierarchy; use the `PARAM_`() macros instead of this (i.e.*
- static void **AddFlag** (const std::string &identifier, const std::string &description, const std::string &alias="")
Adds a flag parameter to the hierarchy; use `PARAM_FLAG()` (p. 762) instead of this.
- static void **DefaultMessages** ()
Parses the parameters for 'help' and 'info'.
- static void **Destroy** ()
*Destroy the **CLI** (p. 184) object.*
- static std::string **GetDescription** (const std::string &identifier)
Get the description of the specified node.
- template<typename T >
static T & **GetParam** (const std::string &identifier)
Grab the value of type T found while parsing.
- static **CLI** & **GetSingleton** ()
Retrieve the singleton.
- static bool **HasParam** (const std::string &identifier)
See if the specified flag was found while parsing.
- static std::string **HyphenateString** (const std::string &str, int padding)
Hyphenate a string or split it onto multiple 80-character lines, with some amount of padding on each line.
- static void **ParseCommandLine** (int argc, char **argv)

- Parses the commandline for arguments.*
- static void **ParseStream** (std::istream &stream)
Parses a stream for arguments.
- static void **Print** ()
Print out the current hierarchy.
- static void **PrintHelp** (const std::string ¶m="")
Print out the help info of the hierarchy.
- static void **RegisterProgramDoc** (util::ProgramDoc *doc)
Registers a ProgramDoc object, which contains documentation about the program.
- static void **RemoveDuplicateFlags** (po::basic_parsed_options< char > &bpo)
Removes duplicate flags.

Public Attributes

- util::ProgramDoc * **doc**
Pointer to the ProgramDoc object.

Private Types

- typedef std::map< std::string, std::string > **amap_t**
Map for aliases, from alias to actual name.
- typedef std::map< std::string, ParamData > **gmap_t**
Map of global values.

Private Member Functions

- **CLI** ()
Make the constructor private, to preclude unauthorized instances.
- **CLI** (const std::string &optionsName)
Initialize desc with a particular name.
- **CLI** (const CLI &other)
Private copy constructor; we don't want copies floating around.

Static Private Member Functions

- static void **AddAlias** (const std::string &alias, const std::string &original)
Maps a given alias to a given parameter.
- static std::string **AliasReverseLookup** (const std::string &value)
Returns an alias, if given the name of the original.
- static void **RequiredOptions** ()
Checks that all required parameters have been specified on the command line.
- static std::string **SanitizeString** (const std::string &str)
Cleans up input pathnames, rendering strings such as /foo/bar and foo/bar/ equivalent inputs.
- static void **UpdateGmap** ()
Parses the values given on the command line, overriding any default values.

Private Attributes

- **amap_t aliasValues**
- po::options_description **desc**
The documentation and names of options.
- bool **didParse**
*True, if **CLI** (p. 184) was used to parse command line options.*
- **gmap_t globalValues**
- std::string **programName**
Hold the name of the program for `--version`.
- std::list< std::string > **requiredOptions**
Pathnames of required options.
- **Timers timer**
Holds the timer objects.
- po::variables_map **vmap**
Values of the options given by user.

Static Private Attributes

- static **CLI * singleton**
The singleton itself.

Friends

- class **Timer**
*So that **Timer::Start()** (p. 563) and **Timer::Stop()** (p. 563) can access the timer variable.*

22.28.1 Detailed Description

Parses the command line for parameters and holds user-specified parameters.

The **CLI** (p. 184) class is a subsystem by which parameters for machine learning methods can be specified and accessed. In conjunction with the macros `PARAM_DOUBLE`, `PARAM_INT`, `PARAM_STRING`, `PARAM_FLAG`, and others, this class aims to make user configurability of MLPACK methods very easy. There are only three methods in **CLI** (p. 184) that a user should need: **CLI::ParseCommandLine()** (p. 193), **CLI::GetParam()** (p. 192), and **CLI::HasParam()** (p. 193) (in addition to the `PARAM_*` macros).

22.28.2 Adding parameters to a program

```
$ ./executable --bar=5
```

Note

The `=` is optional; a space can also be used.

A parameter is specified by using one of the following macros (this is not a complete list; see `core/io/cli.hpp`):

- **PARAM_FLAG(ID, DESC, ALIAS)** (p. 762)

- **PARAM_DOUBLE**(ID, DESC, ALIAS, DEF) (p. 761)
- **PARAM_INT**(ID, DESC, ALIAS, DEF) (p. 763)
- **PARAM_STRING**(ID, DESC, ALIAS, DEF) (p. 764)

Parameters

<i>ID</i>	Name of the parameter.
<i>DESC</i>	Short description of the parameter (one/two sentences).
<i>ALIAS</i>	An alias for the parameter.
<i>DEF</i>	Default value of the parameter.

The flag (boolean) type automatically defaults to false; it is specified merely as a flag on the command line (no '=true' is required).

Here is an example of a few parameters being defined; this is for the AllkNN executable (methods/neighbor_search/allknn_main.cpp):

```
PARAM_STRING_REQ("reference_file", "File containing the reference dataset.",
    "r");
PARAM_STRING_REQ("distances_file", "File to output distances into.", "d");
PARAM_STRING_REQ("neighbors_file", "File to output neighbors into.", "n");
PARAM_INT_REQ("k", "Number of furthest neighbors to find.", "k");
PARAM_STRING("query_file", "File containing query points (optional).", "q",
    "");
PARAM_INT("leaf_size", "Leaf size for tree building.", "l", 20);
PARAM_FLAG("naive", "If true, O(n^2) naive mode is used for computation.",
    "N");
PARAM_FLAG("single_mode", "If true, single-tree search is used (as opposed "
    "to dual-tree search.", "s");
```

More documentation is available on the **PARAM_***() macros in the documentation for core/io/cli.hpp.

22.28.3 Documenting the program itself

In addition to allowing documentation for each individual parameter and module, the **PROGRAM_INFO**() (p. 766) macro provides support for documenting the program itself. There should only be one instance of the **PROGRAM_INFO**() (p. 766) macro. Below is an example:

```
PROGRAM_INFO("Maximum Variance Unfolding", "This program performs maximum "
    "variance unfolding on the given dataset, writing a lower-dimensional "
    "unfolded dataset to the given output file.");
```

This description should be verbose, and explain to a non-expert user what the program does and how to use it. If relevant, paper citations should be included.

22.28.4 Parsing the command line with CLI

To have **CLI** (p. 184) parse the command line at the beginning of code execution, only a call to **ParseCommandLine**() (p. 193) is necessary:

```
int main(int argc, char** argv)
{
    CLI::ParseCommandLine(argc, argv);

    ...
}
```

CLI (p. 184) provides **-help** and **-info** options which give nicely formatted documentation of each option; the documentation is generated from the **DESC** arguments in the **PARAM_***() macros.

22.28.5 Getting parameters with CLI

When the parameters have been defined, the next important thing is how to access them. For this, the **HasParam()** (p. 193) and **GetParam()** (p. 192) methods are used. For instance, to see if the user passed the flag (boolean) "naive":

```
if (CLI::HasParam("naive"))
{
    Log::Info << "Naive has been passed!" << std::endl;
}
```

To get the value of a parameter, such as a string, use **GetParam()**:

```
const std::string filename = CLI::GetParam<std::string>("filename");
```

Note

Options should only be defined in files which define `main()` (that is, main executables). If options are defined elsewhere, they may be spuriously included into other executables and confuse users. Similarly, if your executable has options which you did not define, it is probably because the option is defined somewhere else and included in your executable.

Bug The **COUNTER** variable is used in most cases to guarantee a unique global identifier for options declared using the `PARAM_*`() macros. However, not all compilers have this support—most notably, `gcc < 4.3`. In that case, the **LINE** macro is used as an attempt to get a unique global identifier, but collisions are still possible, and they produce bizarre error messages. See <http://mlpack.org/trac/ticket/74> for more information.

Definition at line 523 of file `cli.hpp`.

22.28.6 Member Typedef Documentation

22.28.6.1 `typedef std::map<std::string, std::string> mlpack::CLI::amap_t` [private]

Map for aliases, from alias to actual name.

Definition at line 691 of file `cli.hpp`.

22.28.6.2 `typedef std::map<std::string, ParamData> mlpack::CLI::gmap_t` [private]

Map of global values.

Definition at line 687 of file `cli.hpp`.

22.28.7 Constructor & Destructor Documentation

22.28.7.1 `mlpack::CLI::~CLI ()`

Destructor.

22.28.7.2 `mlpack::CLI::CLI ()` [private]

Make the constructor private, to preclude unauthorized instances.

22.28.7.3 mpack::CLI::CLI (const std::string & *optionsName*) [private]

Initialize desc with a particular name.

Parameters

<i>optionsName</i>	Name of the module, as far as boost is concerned.
--------------------	---

22.28.7.4 `mlpack::CLI::CLI (const CLI & other) [private]`

Private copy constructor; we don't want copies floating around.

22.28.8 Member Function Documentation

22.28.8.1 `static void mlpack::CLI::Add (const std::string & path, const std::string & description, const std::string & alias = " ", bool required = false) [static]`

Adds a parameter to the hierarchy; use the `PARAM_*`() macros instead of this (i.e.

PARAM_INT() (p. 763)). Uses `char*` and not `std::string` since the vast majority of use cases will be literal strings.

Parameters

<i>identifier</i>	The name of the parameter.
<i>description</i>	Short string description of the parameter.
<i>alias</i>	An alias for the parameter, defaults to "" which is no alias. ("").
<i>required</i>	Indicates if parameter must be set on command line.

22.28.8.2 `template<class T> static void mlpack::CLI::Add (const std::string & identifier, const std::string & description, const std::string & alias = " ", bool required = false) [static]`

Adds a parameter to the hierarchy; use the `PARAM_*`() macros instead of this (i.e.

PARAM_INT() (p. 763)). Uses `char*` and not `std::string` since the vast majority of use cases will be literal strings. If the argument requires a parameter, you must specify a type.

Parameters

<i>identifier</i>	The name of the parameter.
<i>description</i>	Short string description of the parameter.
<i>alias</i>	An alias for the parameter, defaults to "" which is no alias.
<i>required</i>	Indicates if parameter must be set on command line.

22.28.8.3 `static void mlpack::CLI::AddAlias (const std::string & alias, const std::string & original) [static], [private]`

Maps a given alias to a given parameter.

Parameters

<i>alias</i>	The name of the alias to be mapped.
<i>original</i>	The name of the parameter to be mapped.

```
22.28.8.4 static void mpack::CLI::AddFlag ( const std::string & identifier, const std::string & description, const std::string & alias =  
      "" ) [static]
```

Adds a flag parameter to the hierarchy; use **PARAM_FLAG()** (p. 762) instead of this.

Parameters

<i>identifier</i>	The name of the paramater.
<i>description</i>	Short string description of the parameter.
<i>alias</i>	An alias for the parameter, defaults to "" which is no alias.

22.28.8.5 `static std::string mlpack::CLI::AliasReverseLookup (const std::string & value) [static],[private]`

Returns an alias, if given the name of the original.

Parameters

<i>value</i>	The value in a key:value pair where the key is an alias.
--------------	--

Returns

The alias associated with value.

22.28.8.6 `static void mlpack::CLI::DefaultMessages () [static]`

Parses the parameters for 'help' and 'info'.

If found, will print out the appropriate information and kill the program.

22.28.8.7 `static void mlpack::CLI::Destroy () [static]`

Destroy the **CLI** (p. 184) object.

This resets the pointer to the singleton, so in case someone tries to access it after destruction, a new one will be made (the program will not fail).

22.28.8.8 `static std::string mlpack::CLI::GetDescription (const std::string & identifier) [static]`

Get the description of the specified node.

Parameters

<i>identifier</i>	Name of the node in question.
-------------------	-------------------------------

Returns

Description of the node in question.

22.28.8.9 `template<typename T> static T& mlpack::CLI::GetParam (const std::string & identifier) [static]`

Grab the value of type T found while parsing.

You can set the value using this reference safely.

Parameters

<i>identifier</i>	The name of the parameter in question.
-------------------	--

22.28.8.10 static CLI& mpack::CLI::GetSingleton () [static]

Retrieve the singleton.

Not exposed to the outside, so as to spare users some ungainly x.GetSingleton().foo() syntax.

In this case, the singleton is used to store data for the static methods, as there is no point in defining static methods only to have users call private instance methods.

Returns

The singleton instance for use in the static methods.

22.28.8.11 static bool mpack::CLI::HasParam (const std::string & *identifier*) [static]

See if the specified flag was found while parsing.

Parameters

<i>identifier</i>	The name of the parameter in question.
-------------------	--

22.28.8.12 static std::string mpack::CLI::HyphenateString (const std::string & *str*, int *padding*) [static]

Hyphenate a string or split it onto multiple 80-character lines, with some amount of padding on each line.

This is used for option output.

Parameters

<i>str</i>	String to hyphenate (splits are on ' ').
<i>padding</i>	Amount of padding on the left for each new line.

22.28.8.13 static void mpack::CLI::ParseCommandLine (int *argc*, char ** *argv*) [static]

Parses the commandline for arguments.

Parameters

<i>argc</i>	The number of arguments on the commandline.
<i>argv</i>	The array of arguments as strings.

22.28.8.14 static void mpack::CLI::ParseStream (std::istream & *stream*) [static]

Parses a stream for arguments.

Parameters

<i>stream</i>	The stream to be parsed.
---------------	--------------------------

22.28.8.15 `static void mlpack::CLI::Print () [static]`

Print out the current hierarchy.

22.28.8.16 `static void mlpack::CLI::PrintHelp (const std::string & param = " ") [static]`

Print out the help info of the hierarchy.

22.28.8.17 `static void mlpack::CLI::RegisterProgramDoc (util::ProgramDoc * doc) [static]`

Registers a ProgramDoc object, which contains documentation about the program.

If this method has been called before (that is, if two ProgramDocs are instantiated in the program), a fatal error will occur.

Parameters

<i>doc</i>	Pointer to the ProgramDoc object.
------------	-----------------------------------

22.28.8.18 `static void mlpack::CLI::RemoveDuplicateFlags (po::basic_parsed_options< char > & bpo) [static]`

Removes duplicate flags.

Parameters

<i>bpo</i>	The basic_program_options to remove duplicate flags from.
------------	---

22.28.8.19 `static void mlpack::CLI::RequiredOptions () [static], [private]`

Checks that all required parameters have been specified on the command line.

If any have not been specified, an error message is printed and the program is terminated.

22.28.8.20 `static std::string mlpack::CLI::SanitizeString (const std::string & str) [static], [private]`

Cleans up input pathnames, rendering strings such as /foo/bar and foo/bar/ equivalent inputs.

Parameters

<i>str</i>	Input string.
------------	---------------

Returns

Sanitized string.

22.28.8.21 `static void mlpack::CLI::UpdateGmap () [static], [private]`

Parses the values given on the command line, overriding any default values.

22.28.9 Friends And Related Function Documentation

22.28.9.1 friend class Timer [friend]

So that **Timer::Start()** (p. 563) and **Timer::Stop()** (p. 563) can access the timer variable.

Definition at line 707 of file cli.hpp.

22.28.10 Member Data Documentation

22.28.10.1 amap_t mpack::CLI::aliasValues [private]

Definition at line 692 of file cli.hpp.

22.28.10.2 po::options_description mpack::CLI::desc [private]

The documentation and names of options.

Definition at line 678 of file cli.hpp.

22.28.10.3 bool mpack::CLI::didParse [private]

True, if **CLI** (p. 184) was used to parse command line options.

Definition at line 698 of file cli.hpp.

22.28.10.4 util::ProgramDoc* mpack::CLI::doc

Pointer to the ProgramDoc object.

Definition at line 711 of file cli.hpp.

22.28.10.5 gmap_t mpack::CLI::globalValues [private]

Definition at line 688 of file cli.hpp.

22.28.10.6 std::string mpack::CLI::programName [private]

Hold the name of the program for `-version`.

Definition at line 701 of file cli.hpp.

22.28.10.7 std::list<std::string> mpack::CLI::requiredOptions [private]

Pathnames of required options.

Definition at line 684 of file cli.hpp.

22.28.10.8 CLI* mlpack::CLI::singleton [static], [private]

The singleton itself.

Definition at line 695 of file cli.hpp.

22.28.10.9 Timers mlpack::CLI::timer [private]

Holds the timer objects.

Definition at line 704 of file cli.hpp.

22.28.10.10 po::variables_map mlpack::CLI::vmap [private]

Values of the options given by user.

Definition at line 681 of file cli.hpp.

The documentation for this class was generated from the following file:

- src/mlpack/core/util/**cli.hpp**

22.29 mlpack::decision_stump::DecisionStump< MatType > Class Template Reference

This class implements a decision stump.

Public Member Functions

- **DecisionStump** (const MatType &data, const arma::Row< size_t > &labels, const size_t classes, size_t input← BucketSize)
Constructor.
- **DecisionStump** (const **DecisionStump**<> &other, const MatType &data, const arma::rowvec &weights, const arma::Row< size_t > &labels)
Alternate constructor which copies parameters bucketSize and numClass from an already initiated decision stump, other.
- const arma::Col< size_t > **BinLabels** () const
Access the labels for each split bin.
- arma::Col< size_t > & **BinLabels** ()
Modify the labels for each split bin (be careful!).
- void **Classify** (const MatType &test, arma::Row< size_t > &predictedLabels)
Classification function.
- const arma::vec & **Split** () const
Access the splitting values.
- arma::vec & **Split** ()
Modify the splitting values (be careful!).
- int **SplitAttribute** () const
Access the splitting attribute.
- int & **SplitAttribute** ()
Modify the splitting attribute (be careful!).

Private Member Functions

- template<typename LabelType, bool isWeight>
double **CalculateEntropy** (arma::subview_row< LabelType > labels, int begin, const arma::rowvec &tempD)
Calculate the entropy of the given attribute.
- template<typename rType >
rType **CountMostFreq** (const arma::Row< rType > &subCols)
Count the most frequently occurring element in subCols.
- template<typename rType >
int **IsDistinct** (const arma::Row< rType > &featureRow)
Returns 1 if all the values of featureRow are not same.
- void **MergeRanges** ()
After the "split" matrix has been set up, merge ranges with identical class labels.
- template<bool isWeight>
double **SetupSplitAttribute** (const arma::rowvec &attribute, const arma::Row< size_t > &labels, const arma::rowvec &weightD)
Sets up attribute as if it were splitting on it and finds entropy when splitting on attribute.
- template<bool isWeight>
void **Train** (const MatType &data, const arma::Row< size_t > &labels, const arma::rowvec &weightD)
Train the decision stump on the given data and labels.
- template<typename rType >
void **TrainOnAtt** (const arma::rowvec &attribute, const arma::Row< size_t > &labels)
After having decided the attribute on which to split, train on that attribute.

Private Attributes

- arma::Col< size_t > **binLabels**
Stores the labels for each splitting bin.
- size_t **bucketSize**
Size of bucket while determining splitting criterion.
- size_t **numClass**
Stores the number of classes.
- arma::vec **split**
Stores the splitting values after training.
- int **splitAttribute**
Stores the value of the attribute on which to split.

22.29.1 Detailed Description

```
template<typename MatType = arma::mat> class mlpack::decision_stump::DecisionStump< MatType >
```

This class implements a decision stump.

It constructs a single level decision tree, i.e., a decision stump. It uses entropy to decide splitting ranges.

The stump is parameterized by a splitting attribute (the dimension on which points are split), a vector of bin split values, and a vector of labels for each bin. Bin i is specified by the range $[\text{split}[i], \text{split}[i + 1])$. The last bin has range up to $(\text{split}[i + 1])$ does not exist in that case). Points that are below the first bin will take the label of the first bin.

Template Parameters

<i>MatType</i>	Type of matrix that is being used (sparse or dense).
----------------	--

Definition at line 36 of file decision_stump.hpp.

22.29.2 Constructor & Destructor Documentation

22.29.2.1 `template<typename MatType = arma::mat> mlpack::decision_stump::DecisionStump< MatType >::DecisionStump (const MatType & data, const arma::Row< size_t > & labels, const size_t classes, size_t inpBucketSize)`

Constructor.

Train on the provided data. Generate a decision stump from data.

Parameters

<i>data</i>	Input, training data.
<i>labels</i>	Labels of training data.
<i>classes</i>	Number of distinct classes in labels.
<i>inpBucketSize</i>	Minimum size of bucket when splitting.

22.29.2.2 `template<typename MatType = arma::mat> mlpack::decision_stump::DecisionStump< MatType >::DecisionStump (const DecisionStump<> & other, const MatType & data, const arma::rowvec & weights, const arma::Row< size_t > & labels)`

Alternate constructor which copies parameters bucketSize and numClass from an already initiated decision stump, other.

It appropriately sets the weight vector.

Parameters

<i>other</i>	The other initiated Decision Stump object from which we copy the values.
<i>data</i>	The data on which to train this object on.
<i>D</i>	Weight vector to use while training. For boosting purposes.
<i>labels</i>	The labels of data.
<i>isWeight</i>	Whether we need to run a weighted Decision Stump.

22.29.3 Member Function Documentation

22.29.3.1 `template<typename MatType = arma::mat> const arma::Col<size_t> mlpack::decision_stump::DecisionStump< MatType >::BinLabels () const [inline]`

Access the labels for each split bin.

Definition at line 91 of file decision_stump.hpp.

References `mlpack::decision_stump::DecisionStump< MatType >::binLabels`.

22.29.3.2 `template<typename MatType = arma::mat> arma::Col<size_t> & mlpack::decision_stump::DecisionStump< MatType >::BinLabels () [inline]`

Modify the labels for each split bin (be careful!).

Definition at line 93 of file decision_stump.hpp.

References mlpack::decision_stump::DecisionStump< MatType >::binLabels.

```
22.29.3.3  template<typename MatType = arma::mat> template<typename LabelType , bool isWeight> double
           mlpack::decision_stump::DecisionStump< MatType >::CalculateEntropy ( arma::subview_row< LabelType >
           labels, int begin, const arma::rowvec & tempD ) [private]
```

Calculate the entropy of the given attribute.

Parameters

<i>attribute</i>	The attribute of which we calculate the entropy.
<i>labels</i>	Corresponding labels of the attribute.
<i>isWeight</i>	Whether we need to run a weighted Decision Stump.

```
22.29.3.4  template<typename MatType = arma::mat> void mlpack::decision_stump::DecisionStump< MatType
           >::Classify ( const MatType & test, arma::Row< size_t > & predictedLabels )
```

Classification function.

After training, classify test, and put the predicted classes in predictedLabels.

Parameters

<i>test</i>	Testing data or data to classify.
<i>predictedLabels</i>	Vector to store the predicted classes after classifying test data.

```
22.29.3.5  template<typename MatType = arma::mat> template<typename rType > rType mlpack::decision_
           _stump::DecisionStump< MatType >::CountMostFreq ( const arma::Row< rType > & subCols )
           [private]
```

Count the most frequently occurring element in subCols.

Parameters

<i>subCols</i>	The vector in which to find the most frequently occurring element.
----------------	--

```
22.29.3.6  template<typename MatType = arma::mat> template<typename rType > int mlpack::decision_
           _stump::DecisionStump< MatType >::IsDistinct ( const arma::Row< rType > & featureRow )
           [private]
```

Returns 1 if all the values of featureRow are not same.

Parameters

<i>featureRow</i>	The attribute which is checked for identical values.
-------------------	--

```
22.29.3.7  template<typename MatType = arma::mat> void mlpack::decision_stump::DecisionStump< MatType
           >::MergeRanges ( ) [private]
```

After the "split" matrix has been set up, merge ranges with identical class labels.

```
22.29.3.8  template<typename MatType = arma::mat> template<bool isWeight> double mlpack::decision_stump::DecisionStump< MatType >::SetupSplitAttribute ( const arma::rowvec & attribute, const arma::Row< size_t > & labels, const arma::rowvec & weightD ) [private]
```

Sets up attribute as if it were splitting on it and finds entropy when splitting on attribute.

Parameters

<i>attribute</i>	A row from the training data, which might be a candidate for the splitting attribute.
<i>isWeight</i>	Whether we need to run a weighted Decision Stump.

```
22.29.3.9  template<typename MatType = arma::mat> const arma::vec& mlpack::decision_stump::DecisionStump< MatType >::Split ( ) const [inline]
```

Access the splitting values.

Definition at line 86 of file decision_stump.hpp.

References mlpack::decision_stump::DecisionStump< MatType >::split.

```
22.29.3.10 template<typename MatType = arma::mat> arma::vec& mlpack::decision_stump::DecisionStump< MatType >::Split ( ) [inline]
```

Modify the splitting values (be careful!).

Definition at line 88 of file decision_stump.hpp.

References mlpack::decision_stump::DecisionStump< MatType >::split.

```
22.29.3.11 template<typename MatType = arma::mat> int mlpack::decision_stump::DecisionStump< MatType >::SplitAttribute ( ) const [inline]
```

Access the splitting attribute.

Definition at line 81 of file decision_stump.hpp.

References mlpack::decision_stump::DecisionStump< MatType >::splitAttribute.

```
22.29.3.12 template<typename MatType = arma::mat> int& mlpack::decision_stump::DecisionStump< MatType >::SplitAttribute ( ) [inline]
```

Modify the splitting attribute (be careful!).

Definition at line 83 of file decision_stump.hpp.

References mlpack::decision_stump::DecisionStump< MatType >::splitAttribute.

```
22.29.3.13 template<typename MatType = arma::mat> template<bool isWeight> void mlpack::decision_stump::DecisionStump< MatType >::Train ( const MatType & data, const arma::Row< size_t > & labels, const arma::rowvec & weightD ) [private]
```

Train the decision stump on the given data and labels.

Parameters

<i>data</i>	Dataset to train on.
<i>labels</i>	Labels for dataset.
<i>isWeight</i>	Whether we need to run a weighted Decision Stump.

```
22.29.3.14 template<typename MatType = arma::mat> template<typename rType > void mlpack::decision_stump::
DecisionStump< MatType >::TrainOnAtt ( const arma::rowvec & attribute, const arma::Row< size_t > & labels )
[private]
```

After having decided the attribute on which to split, train on that attribute.

Parameters

<i>attribute</i>	attribute is the attribute decided by the constructor on which we now train the decision stump.
------------------	---

22.29.4 Member Data Documentation

```
22.29.4.1 template<typename MatType = arma::mat> arma::Col<size_t> mlpack::decision_stump::DecisionStump<
MatType >::binLabels [private]
```

Stores the labels for each splitting bin.

Definition at line 109 of file decision_stump.hpp.

Referenced by mlpack::decision_stump::DecisionStump< MatType >::BinLabels().

```
22.29.4.2 template<typename MatType = arma::mat> size_t mlpack::decision_stump::DecisionStump< MatType
>::bucketSize [private]
```

Size of bucket while determining splitting criterion.

Definition at line 103 of file decision_stump.hpp.

```
22.29.4.3 template<typename MatType = arma::mat> size_t mlpack::decision_stump::DecisionStump< MatType
>::numClass [private]
```

Stores the number of classes.

Definition at line 97 of file decision_stump.hpp.

```
22.29.4.4 template<typename MatType = arma::mat> arma::vec mlpack::decision_stump::DecisionStump< MatType
>::split [private]
```

Stores the splitting values after training.

Definition at line 106 of file decision_stump.hpp.

Referenced by mlpack::decision_stump::DecisionStump< MatType >::Split().

22.29.4.5 `template<typename MatType = arma::mat> int mlpack::decision_stump::DecisionStump< MatType >::splitAttribute [private]`

Stores the value of the attribute on which to split.

Definition at line 100 of file `decision_stump.hpp`.

Referenced by `mlpack::decision_stump::DecisionStump< MatType >::SplitAttribute()`.

The documentation for this class was generated from the following file:

- `src/mlpack/methods/decision_stump/decision_stump.hpp`

22.30 mlpack::det::DTree Class Reference

A density estimation tree is similar to both a decision tree and a space partitioning tree (like a kd-tree).

Collaboration diagram for `mlpack::det::DTree`:



Public Member Functions

- **DTree** ()
Create an empty density estimation tree.
- **DTree** (const arma::vec &**maxVals**, const arma::vec &**minVals**, const size_t totalPoints)
Create a density estimation tree with the given bounds and the given number of total points.
- **DTree** (arma::mat &data)
Create a density estimation tree on the given data.
- **DTree** (const arma::vec &**maxVals**, const arma::vec &**minVals**, const size_t **start**, const size_t **end**, const double **logNegError**)
Create a child node of a density estimation tree given the bounding box specified by maxVals and minVals, using the size given in start and end and the specified error.
- **DTree** (const arma::vec &**maxVals**, const arma::vec &**minVals**, const size_t totalPoints, const size_t **start**, const size_t **end**)
Create a child node of a density estimation tree given the bounding box specified by maxVals and minVals, using the size given in start and end, and calculating the error with the total number of points given.
- **~DTree** ()
Clean up memory allocated by the tree.
- double **AlphaUpper** () const
Return the upper part of the alpha sum.
- double **ComputeValue** (const arma::vec &query) const
Compute the logarithm of the density estimate of a given query point.
- void **ComputeVariableImportance** (arma::vec &importances) const
Compute the variable importance of each dimension in the learned tree.
- size_t **End** () const
Return the first index of a point not contained in this node.

- int **FindBucket** (const arma::vec &query) const
Return the tag of the leaf containing the query.
- double **Grow** (arma::mat &data, arma::Col< size_t > &oldFromNew, const bool useVolReg=false, const size_t maxLeafSize=10, const size_t minLeafSize=5)
Greedily expand the tree.
- **DTree * Left** () const
Return the left child.
- double **LogNegativeError** (const size_t totalPoints) const
Compute the log-negative-error for this point, given the total number of points in the dataset.
- double **LogNegError** () const
Return the log negative error of this node.
- double **LogVolume** () const
Return the inverse of the volume of this node.
- const arma::vec & **MaxVals** () const
Return the maximum values.
- arma::vec & **MaxVals** ()
Modify the maximum values.
- const arma::vec & **MinVals** () const
Return the minimum values.
- arma::vec & **MinVals** ()
Modify the minimum values.
- double **PruneAndUpdate** (const double oldAlpha, const size_t points, const bool useVolReg=false)
Perform alpha pruning on a tree.
- double **Ratio** () const
Return the ratio of points in this node to the points in the whole dataset.
- **DTree * Right** () const
Return the right child.
- bool **Root** () const
Return whether or not this is the root of the tree.
- size_t **SplitDim** () const
Return the split dimension of this node.
- double **SplitValue** () const
Return the split value of this node.
- size_t **Start** () const
Return the starting index of points contained in this node.
- size_t **SubtreeLeaves** () const
Return the number of leaves which are descendants of this node.
- double **SubtreeLeavesLogNegError** () const
Return the log negative error of all descendants of this node.
- int **TagTree** (const int tag=0)
Index the buckets for possible usage later; this results in every leaf in the tree having a specific tag (accessible with BucketTag()).
- std::string **ToString** () const
Returns a string representation of this object.
- bool **WithinRange** (const arma::vec &query) const
Return whether a query point is within the range of this node.
- void **WriteTree** (FILE *fp, const size_t level=0) const
Print the tree in a depth-first manner (this function is called recursively).

Private Member Functions

- bool **FindSplit** (const arma::mat &data, size_t &**splitDim**, double &**splitValue**, double &leftError, double &rightError, const size_t minLeafSize=5) const
Find the dimension to split on.
- size_t **SplitData** (arma::mat &data, const size_t **splitDim**, const double **splitValue**, arma::Col< size_t > &oldFromNew) const
Split the data, returning the number of points left of the split.

Private Attributes

- double **alphaUpper**
Upper part of alpha sum; used for pruning.
- int **bucketTag**
The tag for the leaf, used for hashing points.
- size_t **end**
The index of the last point in the dataset contained in this node (and its children).
- **DTree * left**
The left child.
- double **logNegError**
log-negative-L2-error of the node.
- double **logVolume**
The logarithm of the volume of the node.
- arma::vec **maxVals**
Upper half of bounding box for this node.
- arma::vec **minVals**
Lower half of bounding box for this node.
- double **ratio**
Ratio of the number of points in the node to the total number of points.
- **DTree * right**
The right child.
- bool **root**
If true, this node is the root of the tree.
- size_t **splitDim**
The splitting dimension for this node.
- double **splitValue**
The split value on the splitting dimension for this node.
- size_t **start**
The index of the first point in the dataset contained in this node (and its children).
- size_t **subtreeLeaves**
Number of leaves of the subtree.
- double **subtreeLeavesLogNegError**
Sum of the error of the leaves of the subtree.

22.30.1 Detailed Description

A density estimation tree is similar to both a decision tree and a space partitioning tree (like a kd-tree).

Each leaf represents a constant-density hyper-rectangle. The tree is constructed in such a way as to minimize the integrated square error between the probability distribution of the tree and the observed probability distribution of the data. Because the tree is similar to a decision tree, the density estimation tree can provide very fast density estimates for a given point.

For more information, see the following paper:

```
@incollection{ram2011,
  author = {Ram, Parikshit and Gray, Alexander G.},
  title = {Density estimation trees},
  booktitle = {{Proceedings of the 17th ACM SIGKDD International Conference
    on Knowledge Discovery and Data Mining}},
  series = {KDD '11},
  year = {2011},
  pages = {627--635}
}
```

Definition at line 46 of file dtree.hpp.

22.30.2 Constructor & Destructor Documentation

22.30.2.1 mlpack::det::DTree::DTree ()

Create an empty density estimation tree.

22.30.2.2 mlpack::det::DTree::DTree (const arma::vec & maxVals, const arma::vec & minVals, const size_t totalPoints)

Create a density estimation tree with the given bounds and the given number of total points.

Children will not be created.

Parameters

<i>maxVals</i>	Maximum values of the bounding box.
<i>minVals</i>	Minimum values of the bounding box.
<i>totalPoints</i>	Total number of points in the dataset.

22.30.2.3 mlpack::det::DTree::DTree (arma::mat & data)

Create a density estimation tree on the given data.

Children will be created following the procedure outlined in the paper. The data will be modified; it will be reordered similar to the way BinarySpaceTree modifies datasets.

Parameters

<i>data</i>	Dataset to build tree on.
-------------	---------------------------

22.30.2.4 mlpack::det::DTree::DTree (const arma::vec & maxVals, const arma::vec & minVals, const size_t start, const size_t end, const double logNegError)

Create a child node of a density estimation tree given the bounding box specified by maxVals and minVals, using the size given in start and end and the specified error.

Children of this node will not be created recursively.

Parameters

<i>maxVals</i>	Upper bound of bounding box.
<i>minVals</i>	Lower bound of bounding box.
<i>start</i>	Start of points represented by this node in the data matrix.
<i>end</i>	End of points represented by this node in the data matrix.
<i>error</i>	log-negative error of this node.

22.30.2.5 `mlpack::det::DTree::DTree (const arma::vec & maxVals, const arma::vec & minVals, const size_t totalPoints, const size_t start, const size_t end)`

Create a child node of a density estimation tree given the bounding box specified by *maxVals* and *minVals*, using the size given in *start* and *end*, and calculating the error with the total number of points given.

Children of this node will not be created recursively.

Parameters

<i>maxVals</i>	Upper bound of bounding box.
<i>minVals</i>	Lower bound of bounding box.
<i>start</i>	Start of points represented by this node in the data matrix.
<i>end</i>	End of points represented by this node in the data matrix.

22.30.2.6 `mlpack::det::DTree::~~DTree ()`

Clean up memory allocated by the tree.

22.30.3 Member Function Documentation

22.30.3.1 `double mlpack::det::DTree::AlphaUpper () const [inline]`

Return the upper part of the alpha sum.

Definition at line 276 of file `dtree.hpp`.

References `alphaUpper`.

22.30.3.2 `double mlpack::det::DTree::ComputeValue (const arma::vec & query) const`

Compute the logarithm of the density estimate of a given query point.

Parameters

<i>query</i>	Point to estimate density of.
--------------	-------------------------------

22.30.3.3 `void mlpack::det::DTree::ComputeVariableImportance (arma::vec & importances) const`

Compute the variable importance of each dimension in the learned tree.

Parameters

<i>importances</i>	Vector to store the calculated importances in.
--------------------	--

22.30.3.4 `size_t mlpack::det::DTree::End () const [inline]`

Return the first index of a point not contained in this node.

Definition at line 253 of file dtree.hpp.

References end.

22.30.3.5 `int mlpack::det::DTree::FindBucket (const arma::vec & query) const`

Return the tag of the leaf containing the query.

This is useful for generating class memberships.

Parameters

<i>query</i>	Query to search for.
--------------	----------------------

22.30.3.6 `bool mlpack::det::DTree::FindSplit (const arma::mat & data, size_t & splitDim, double & splitValue, double & leftError, double & rightError, const size_t minLeafSize = 5) const [private]`

Find the dimension to split on.

22.30.3.7 `double mlpack::det::DTree::Grow (arma::mat & data, arma::Col< size_t > & oldFromNew, const bool useVolReg = false, const size_t maxLeafSize = 10, const size_t minLeafSize = 5)`

Greedily expand the tree.

The points in the dataset will be reordered during tree growth.

Parameters

<i>data</i>	Dataset to build tree on.
<i>oldFromNew</i>	Mappings from old points to new points.
<i>useVolReg</i>	If true, volume regularization is used.
<i>maxLeafSize</i>	Maximum size of a leaf.
<i>minLeafSize</i>	Minimum size of a leaf.

22.30.3.8 `DTree* mlpack::det::DTree::Left () const [inline]`

Return the left child.

Definition at line 270 of file dtree.hpp.

References left.

22.30.3.9 `double mlpack::det::DTree::LogNegativeError (const size_t totalPoints) const`

Compute the log-negative-error for this point, given the total number of points in the dataset.

Parameters

<i>totalPoints</i>	Total number of points in the dataset.
--------------------	--

22.30.3.10 `double mlpack::det::DTree::LogNegError () const` `[inline]`

Return the log negative error of this node.

Definition at line 259 of file dtree.hpp.

References `logNegError`.

22.30.3.11 `double mlpack::det::DTree::LogVolume () const` `[inline]`

Return the inverse of the volume of this node.

Definition at line 268 of file dtree.hpp.

References `logVolume`.

22.30.3.12 `const arma::vec& mlpack::det::DTree::MaxVals () const` `[inline]`

Return the maximum values.

Definition at line 279 of file dtree.hpp.

References `maxVals`.

22.30.3.13 `arma::vec& mlpack::det::DTree::MaxVals ()` `[inline]`

Modify the maximum values.

Definition at line 281 of file dtree.hpp.

References `maxVals`.

22.30.3.14 `const arma::vec& mlpack::det::DTree::MinVals () const` `[inline]`

Return the minimum values.

Definition at line 284 of file dtree.hpp.

References `minVals`.

22.30.3.15 `arma::vec& mlpack::det::DTree::MinVals ()` `[inline]`

Modify the minimum values.

Definition at line 286 of file dtree.hpp.

References `minVals`.

22.30.3.16 `double mlpack::det::DTree::PruneAndUpdate (const double oldAlpha, const size_t points, const bool useVolReg = false)`

Perform alpha pruning on a tree.

Returns the new value of alpha.

Parameters

<i>oldAlpha</i>	Old value of alpha.
<i>points</i>	Total number of points in dataset.
<i>useVolReg</i>	If true, volume regularization is used.

Returns

New value of alpha.

22.30.3.17 `double mlpack::det::DTree::Ratio () const [inline]`

Return the ratio of points in this node to the points in the whole dataset.

Definition at line 266 of file dtree.hpp.

References ratio.

22.30.3.18 `DTree* mlpack::det::DTree::Right () const [inline]`

Return the right child.

Definition at line 272 of file dtree.hpp.

References right.

22.30.3.19 `bool mlpack::det::DTree::Root () const [inline]`

Return whether or not this is the root of the tree.

Definition at line 274 of file dtree.hpp.

References root.

22.30.3.20 `size_t mlpack::det::DTree::SplitData (arma::mat & data, const size_t splitDim, const double splitValue, arma::Col< size_t > & oldFromNew) const [private]`

Split the data, returning the number of points left of the split.

22.30.3.21 `size_t mlpack::det::DTree::SplitDim () const [inline]`

Return the split dimension of this node.

Definition at line 255 of file dtree.hpp.

References splitDim.

22.30.3.22 `double mlpack::det::DTree::SplitValue () const [inline]`

Return the split value of this node.

Definition at line 257 of file dtree.hpp.

References `splitValue`.

22.30.3.23 `size_t mlpack::det::DTree::Start () const [inline]`

Return the starting index of points contained in this node.

Definition at line 251 of file dtree.hpp.

References `start`.

22.30.3.24 `size_t mlpack::det::DTree::SubtreeLeaves () const [inline]`

Return the number of leaves which are descendants of this node.

Definition at line 263 of file dtree.hpp.

References `subtreeLeaves`.

22.30.3.25 `double mlpack::det::DTree::SubtreeLeavesLogNegError () const [inline]`

Return the log negative error of all descendants of this node.

Definition at line 261 of file dtree.hpp.

References `subtreeLeavesLogNegError`.

22.30.3.26 `int mlpack::det::DTree::TagTree (const int tag = 0)`

Index the buckets for possible usage later; this results in every leaf in the tree having a specific tag (accessible with `BucketTag()`).

This function calls itself recursively.

Parameters

<i>tag</i>	Tag for the next leaf; leave at 0 for the initial call.
------------	---

22.30.3.27 `std::string mlpack::det::DTree::ToString () const`

Returns a string representation of this object.

22.30.3.28 `bool mlpack::det::DTree::WithinRange (const arma::vec & query) const`

Return whether a query point is within the range of this node.

22.30.3.29 `void mlpack::det::DTree::WriteTree (FILE * fp, const size_t level = 0) const`

Print the tree in a depth-first manner (this function is called recursively).

Parameters

<i>fp</i>	File to write the tree to.
<i>level</i>	Level of the tree (should start at 0).

22.30.4 Member Data Documentation

22.30.4.1 `double mlpack::det::DTree::alphaUpper` [private]

Upper part of alpha sum; used for pruning.

Definition at line 242 of file dtree.hpp.

Referenced by AlphaUpper().

22.30.4.2 `int mlpack::det::DTree::bucketTag` [private]

The tag for the leaf, used for hashing points.

Definition at line 239 of file dtree.hpp.

22.30.4.3 `size_t mlpack::det::DTree::end` [private]

The index of the last point in the dataset contained in this node (and its children).

Definition at line 207 of file dtree.hpp.

Referenced by End().

22.30.4.4 `DTree* mlpack::det::DTree::left` [private]

The left child.

Definition at line 245 of file dtree.hpp.

Referenced by Left().

22.30.4.5 `double mlpack::det::DTree::logNegError` [private]

log-negative-L2-error of the node.

Definition at line 221 of file dtree.hpp.

Referenced by LogNegError().

22.30.4.6 `double mlpack::det::DTree::logVolume` [private]

The logarithm of the volume of the node.

Definition at line 236 of file dtree.hpp.

Referenced by LogVolume().

22.30.4.7 `arma::vec mlpack::det::DTree::maxVals` `[private]`

Upper half of bounding box for this node.

Definition at line 210 of file `dtree.hpp`.

Referenced by `MaxVals()`.

22.30.4.8 `arma::vec mlpack::det::DTree::minVals` `[private]`

Lower half of bounding box for this node.

Definition at line 212 of file `dtree.hpp`.

Referenced by `MinVals()`.

22.30.4.9 `double mlpack::det::DTree::ratio` `[private]`

Ratio of the number of points in the node to the total number of points.

Definition at line 233 of file `dtree.hpp`.

Referenced by `Ratio()`.

22.30.4.10 `DTree* mlpack::det::DTree::right` `[private]`

The right child.

Definition at line 247 of file `dtree.hpp`.

Referenced by `Right()`.

22.30.4.11 `bool mlpack::det::DTree::root` `[private]`

If true, this node is the root of the tree.

Definition at line 230 of file `dtree.hpp`.

Referenced by `Root()`.

22.30.4.12 `size_t mlpack::det::DTree::splitDim` `[private]`

The splitting dimension for this node.

Definition at line 215 of file `dtree.hpp`.

Referenced by `SplitDim()`.

22.30.4.13 `double mlpack::det::DTree::splitValue` `[private]`

The split value on the splitting dimension for this node.

Definition at line 218 of file `dtree.hpp`.

Referenced by `SplitValue()`.

22.30.4.14 `size_t mlpack::det::DTree::start` [private]

The index of the first point in the dataset contained in this node (and its children).

Definition at line 204 of file `dtree.hpp`.

Referenced by `Start()`.

22.30.4.15 `size_t mlpack::det::DTree::subtreeLeaves` [private]

Number of leaves of the subtree.

Definition at line 227 of file `dtree.hpp`.

Referenced by `SubtreeLeaves()`.

22.30.4.16 `double mlpack::det::DTree::subtreeLeavesLogNegError` [private]

Sum of the error of the leaves of the subtree.

Definition at line 224 of file `dtree.hpp`.

Referenced by `SubtreeLeavesLogNegError()`.

The documentation for this class was generated from the following file:

- `src/mlpack/methods/det/dtree.hpp`

22.31 `mlpack::distribution::DiscreteDistribution` Class Reference

A discrete distribution where the only observations are discrete observations.

Public Member Functions

- **DiscreteDistribution ()**
Default constructor, which creates a distribution that has no observations.
- **DiscreteDistribution (const size_t numObservations)**
Define the discrete distribution as having numObservations possible observations.
- **DiscreteDistribution (const arma::vec &probabilities)**
Define the discrete distribution as having the given probabilities for each observation.
- `size_t Dimensionality () const`
Get the dimensionality of the distribution.
- `void Estimate (const arma::mat &observations)`
Estimate the probability distribution directly from the given observations.
- `void Estimate (const arma::mat &observations, const arma::vec &probabilities)`
Estimate the probability distribution from the given observations, taking into account the probability of each observation actually being from this distribution.
- `const arma::vec & Probabilities () const`
Return the vector of probabilities.
- `arma::vec & Probabilities ()`
Modify the vector of probabilities.

- double **Probability** (const arma::vec &observation) const
Return the probability of the given observation.
- arma::vec **Random** () const
Return a randomly generated observation (one-dimensional vector; one observation) according to the probability distribution defined by this object.
- std::string **ToString** () const

Private Attributes

- arma::vec **probabilities**

22.31.1 Detailed Description

A discrete distribution where the only observations are discrete observations.

This is useful (for example) with discrete Hidden Markov Models, where observations are non-negative integers representing specific emissions.

No bounds checking is performed for observations, so if an invalid observation is passed (i.e. `observation > numObservations`), a crash will probably occur.

This distribution only supports one-dimensional observations, so when passing an `arma::vec` as an observation, it should only have one dimension (`vec.n_rows == 1`). Any additional dimensions will simply be ignored.

Note

This class, like every other class in MLPACK, uses `arma::vec` to represent observations. While a discrete distribution only has positive integers (`size_t`) as observations, these can be converted to doubles (which is what `arma::vec` holds). This distribution internally converts those doubles back into `size_t` before comparisons.

Definition at line 45 of file `discrete_distribution.hpp`.

22.31.2 Constructor & Destructor Documentation

22.31.2.1 mlpack::distribution::DiscreteDistribution::DiscreteDistribution () [inline]

Default constructor, which creates a distribution that has no observations.

Definition at line 51 of file `discrete_distribution.hpp`.

22.31.2.2 mlpack::distribution::DiscreteDistribution::DiscreteDistribution (const size_t numObservations) [inline]

Define the discrete distribution as having `numObservations` possible observations.

The probability in each state will be set to $(1 / \text{numObservations})$.

Parameters

<i>numObservations</i>	Number of possible observations this distribution can have.
------------------------	---

Definition at line 61 of file `discrete_distribution.hpp`.

22.31.2.3 `mlpack::distribution::DiscreteDistribution::DiscreteDistribution (const arma::vec & probabilities)` `[inline]`

Define the discrete distribution as having the given probabilities for each observation.

Parameters

<i>probabilities</i>	Probabilities of each possible observation.
----------------------	---

Definition at line 71 of file discrete_distribution.hpp.

22.31.3 Member Function Documentation

22.31.3.1 `size_t mpack::distribution::DiscreteDistribution::Dimensionality () const` `[inline]`

Get the dimensionality of the distribution.

Definition at line 87 of file discrete_distribution.hpp.

22.31.3.2 `void mpack::distribution::DiscreteDistribution::Estimate (const arma::mat & observations)`

Estimate the probability distribution directly from the given observations.

If any of the observations is greater than numObservations, a crash is likely to occur.

Parameters

<i>observations</i>	List of observations.
---------------------	-----------------------

22.31.3.3 `void mpack::distribution::DiscreteDistribution::Estimate (const arma::mat & observations, const arma::vec & probabilities)`

Estimate the probability distribution from the given observations, taking into account the probability of each observation actually being from this distribution.

Parameters

<i>observations</i>	List of observations.
<i>probabilities</i>	List of probabilities that each observation is actually from this distribution.

22.31.3.4 `const arma::vec& mpack::distribution::DiscreteDistribution::Probabilities () const` `[inline]`

Return the vector of probabilities.

Definition at line 145 of file discrete_distribution.hpp.

References probabilities.

22.31.3.5 `arma::vec& mpack::distribution::DiscreteDistribution::Probabilities ()` `[inline]`

Modify the vector of probabilities.

Definition at line 147 of file discrete_distribution.hpp.

References probabilities.

22.31.3.6 `double mpack::distribution::DiscreteDistribution::Probability (const arma::vec & observation) const` `[inline]`

Return the probability of the given observation.

If the observation is greater than the number of possible observations, then a crash will probably occur – bounds checking is not performed.

Parameters

<i>observation</i>	Observation to return the probability of.
--------------------	---

Returns

Probability of the given observation.

Definition at line 97 of file `discrete_distribution.hpp`.

References `mlpack::Log::Debug`, and `probabilities`.

22.31.3.7 `arma::vec mlpack::distribution::DiscreteDistribution::Random () const`

Return a randomly generated observation (one-dimensional vector; one observation) according to the probability distribution defined by this object.

Returns

Random observation.

22.31.3.8 `std::string mlpack::distribution::DiscreteDistribution::ToString () const`

22.31.4 Member Data Documentation

22.31.4.1 `arma::vec mlpack::distribution::DiscreteDistribution::probabilities [private]`

Definition at line 155 of file `discrete_distribution.hpp`.

Referenced by `Probabilities()`, and `Probability()`.

The documentation for this class was generated from the following file:

- `src/mlpack/core/dists/discrete_distribution.hpp`

22.32 `mlpack::distribution::GaussianDistribution` Class Reference

A single multivariate Gaussian distribution.

Public Member Functions

- **`GaussianDistribution ()`**
Default constructor, which creates a Gaussian with zero dimension.
- **`GaussianDistribution (const size_t dimension)`**
Create a Gaussian distribution with zero mean and identity covariance with the given dimensionality.
- **`GaussianDistribution (const arma::vec &mean, const arma::mat &covariance)`**
Create a Gaussian distribution with the given mean and covariance.
- `const arma::mat & Covariance () const`

- Return the covariance matrix.*

 - arma::mat & **Covariance** ()

Return a modifiable copy of the covariance.
- size_t **Dimensionality** () const

Return the dimensionality of this distribution.
- void **Estimate** (const arma::mat &observations)

Estimate the Gaussian distribution directly from the given observations.
- void **Estimate** (const arma::mat &observations, const arma::vec &probabilities)

Estimate the Gaussian distribution from the given observations, taking into account the probability of each observation actually being from this distribution.
- const arma::vec & **Mean** () const

Return the mean.
- arma::vec & **Mean** ()

Return a modifiable copy of the mean.
- double **Probability** (const arma::vec &observation) const

Return the probability of the given observation.
- arma::vec **Random** () const

Return a randomly generated observation according to the probability distribution defined by this object.
- std::string **ToString** () const

Returns a string representation of this object.

Private Attributes

- arma::mat **covariance**

Covariance of the distribution.
- arma::vec **mean**

Mean of the distribution.

22.32.1 Detailed Description

A single multivariate Gaussian distribution.

Definition at line 27 of file gaussian_distribution.hpp.

22.32.2 Constructor & Destructor Documentation

22.32.2.1 mpack::distribution::GaussianDistribution::GaussianDistribution () [inline]

Default constructor, which creates a Gaussian with zero dimension.

Definition at line 39 of file gaussian_distribution.hpp.

22.32.2.2 mpack::distribution::GaussianDistribution::GaussianDistribution (const size_t *dimension*) [inline]

Create a Gaussian distribution with zero mean and identity covariance with the given dimensionality.

Definition at line 45 of file gaussian_distribution.hpp.

22.32.2.3 `mlpack::distribution::GaussianDistribution::GaussianDistribution (const arma::vec & mean, const arma::mat & covariance) [inline]`

Create a Gaussian distribution with the given mean and covariance.

Definition at line 53 of file gaussian_distribution.hpp.

22.32.3 Member Function Documentation

22.32.3.1 `const arma::mat& mlpack::distribution::GaussianDistribution::Covariance () const [inline]`

Return the covariance matrix.

Definition at line 96 of file gaussian_distribution.hpp.

References covariance.

22.32.3.2 `arma::mat& mlpack::distribution::GaussianDistribution::Covariance () [inline]`

Return a modifiable copy of the covariance.

Definition at line 98 of file gaussian_distribution.hpp.

References covariance.

22.32.3.3 `size_t mlpack::distribution::GaussianDistribution::Dimensionality () const [inline]`

Return the dimensionality of this distribution.

Definition at line 57 of file gaussian_distribution.hpp.

22.32.3.4 `void mlpack::distribution::GaussianDistribution::Estimate (const arma::mat & observations)`

Estimate the Gaussian distribution directly from the given observations.

Parameters

<i>observations</i>	List of observations.
---------------------	-----------------------

22.32.3.5 `void mlpack::distribution::GaussianDistribution::Estimate (const arma::mat & observations, const arma::vec & probabilities)`

Estimate the Gaussian distribution from the given observations, taking into account the probability of each observation actually being from this distribution.

22.32.3.6 `const arma::vec& mlpack::distribution::GaussianDistribution::Mean () const [inline]`

Return the mean.

Definition at line 91 of file gaussian_distribution.hpp.

References mean.

22.32.3.7 `arma::vec& mpack::distribution::GaussianDistribution::Mean () [inline]`

Return a modifiable copy of the mean.

Definition at line 93 of file `gaussian_distribution.hpp`.

References `mean`.

22.32.3.8 `double mpack::distribution::GaussianDistribution::Probability (const arma::vec & observation) const [inline]`

Return the probability of the given observation.

Definition at line 62 of file `gaussian_distribution.hpp`.

References `mpack::gmm::phi()`.

22.32.3.9 `arma::vec mpack::distribution::GaussianDistribution::Random () const`

Return a randomly generated observation according to the probability distribution defined by this object.

Returns

Random observation from this Gaussian distribution.

22.32.3.10 `std::string mpack::distribution::GaussianDistribution::ToString () const`

Returns a string representation of this object.

22.32.4 Member Data Documentation

22.32.4.1 `arma::mat mpack::distribution::GaussianDistribution::covariance [private]`

Covariance of the distribution.

Definition at line 33 of file `gaussian_distribution.hpp`.

Referenced by `Covariance()`.

22.32.4.2 `arma::vec mpack::distribution::GaussianDistribution::mean [private]`

Mean of the distribution.

Definition at line 31 of file `gaussian_distribution.hpp`.

Referenced by `Mean()`.

The documentation for this class was generated from the following file:

- `src/mpack/core/dists/gaussian_distribution.hpp`

22.33 mpack::distribution::LaplaceDistribution Class Reference

The multivariate Laplace distribution centered at 0 has pdf.

Public Member Functions

- **LaplaceDistribution** ()
Default constructor, which creates a Laplace distribution with zero dimension and zero scale parameter.
- **LaplaceDistribution** (const size_t dimensionality, const double **scale**)
Construct the Laplace distribution with the given scale and dimensionality.
- **LaplaceDistribution** (const arma::vec &**mean**, const double **scale**)
Construct the Laplace distribution with the given mean and scale parameter.
- size_t **Dimensionality** () const
Return the dimensionality of this distribution.
- void **Estimate** (const arma::mat &observations)
Estimate the Laplace distribution directly from the given observations.
- void **Estimate** (const arma::mat &observations, const arma::vec &probabilities)
Estimate the Laplace distribution from the given observations, taking into account the probability of each observation actually being from this distribution.
- const arma::vec & **Mean** () const
Return the mean.
- arma::vec & **Mean** ()
Modify the mean.
- double **Probability** (const arma::vec &observation) const
Return the probability of the given observation.
- arma::vec **Random** () const
Return a randomly generated observation according to the probability distribution defined by this object.
- double **Scale** () const
Return the scale parameter.
- double & **Scale** ()
Modify the scale parameter.
- std::string **ToString** () const
Return a string representation of the object.

Private Attributes

- arma::vec **mean**
Mean of the distribution.
- double **scale**
Scale parameter of the distribution.

22.33.1 Detailed Description

The multivariate Laplace distribution centered at 0 has pdf.

$$f(x|\theta) = \frac{1}{2\theta} \exp\left(-\frac{\|x - \mu\|}{\theta}\right)$$

given scale parameter θ and mean μ . This implementation assumes a diagonal covariance, but a rewrite to support arbitrary covariances is possible.

See the following paper for more information on the non-diagonal-covariance Laplace distribution and estimation techniques:

```
@article{eltoft2006multivariate,
  title={On the Multivariate Laplace Distribution},
  author={Eltoft, Torbjørn and Kim, Taesu and Lee, Te-Won},
  journal={IEEE Signal Processing Letters},
  volume={13},
  number={5},
  pages={300--304},
  year={2006}
}
```

Note that because of the diagonal covariance restriction, much of the algebra in the paper above becomes simplified, and the PDF takes roughly the same form as the univariate case.

Definition at line 51 of file laplace_distribution.hpp.

22.33.2 Constructor & Destructor Documentation

22.33.2.1 mpack::distribution::LaplaceDistribution::LaplaceDistribution () `[inline]`

Default constructor, which creates a Laplace distribution with zero dimension and zero scale parameter.

Definition at line 58 of file laplace_distribution.hpp.

22.33.2.2 mpack::distribution::LaplaceDistribution::LaplaceDistribution (const size_t *dimensionality*, const double *scale*) `[inline]`

Construct the Laplace distribution with the given scale and dimensionality.

The mean is initialized to zero.

Parameters

<i>dimensionality</i>	Dimensionality of distribution.
<i>scale</i>	Scale of distribution.

Definition at line 67 of file laplace_distribution.hpp.

22.33.2.3 mpack::distribution::LaplaceDistribution::LaplaceDistribution (const arma::vec & *mean*, const double *scale*) `[inline]`

Construct the Laplace distribution with the given mean and scale parameter.

Parameters

<i>mean</i>	Mean of distribution.
<i>scale</i>	Scale of distribution.

Definition at line 76 of file laplace_distribution.hpp.

22.33.3 Member Function Documentation

22.33.3.1 size_t mpack::distribution::LaplaceDistribution::Dimensionality () const `[inline]`

Return the dimensionality of this distribution.

Definition at line 80 of file laplace_distribution.hpp.

References [mean](#).

22.33.3.2 `void mlpack::distribution::LaplaceDistribution::Estimate (const arma::mat & observations)`

Estimate the Laplace distribution directly from the given observations.

Parameters

<i>observations</i>	List of observations.
---------------------	-----------------------

22.33.3.3 `void mlpack::distribution::LaplaceDistribution::Estimate (const arma::mat & observations, const arma::vec & probabilities)`

Estimate the Laplace distribution from the given observations, taking into account the probability of each observation actually being from this distribution.

22.33.3.4 `const arma::vec& mlpack::distribution::LaplaceDistribution::Mean () const` `[inline]`

Return the mean.

Definition at line 130 of file `laplace_distribution.hpp`.

References `mean`.

22.33.3.5 `arma::vec& mlpack::distribution::LaplaceDistribution::Mean ()` `[inline]`

Modify the mean.

Definition at line 132 of file `laplace_distribution.hpp`.

References `mean`.

22.33.3.6 `double mlpack::distribution::LaplaceDistribution::Probability (const arma::vec & observation) const`

Return the probability of the given observation.

22.33.3.7 `arma::vec mlpack::distribution::LaplaceDistribution::Random () const` `[inline]`

Return a randomly generated observation according to the probability distribution defined by this object.

This is inlined for speed.

Returns

Random observation from this Laplace distribution.

Definition at line 93 of file `laplace_distribution.hpp`.

References `mean`, and `scale`.

22.33.3.8 `double mlpack::distribution::LaplaceDistribution::Scale () const` `[inline]`

Return the scale parameter.

Definition at line 135 of file `laplace_distribution.hpp`.

References `scale`.

22.33.3.9 `double& mlpack::distribution::LaplaceDistribution::Scale () [inline]`

Modify the scale parameter.

Definition at line 137 of file `laplace_distribution.hpp`.

References `scale`.

22.33.3.10 `std::string mlpack::distribution::LaplaceDistribution::ToString () const`

Return a string representation of the object.

22.33.4 Member Data Documentation

22.33.4.1 `arma::vec mlpack::distribution::LaplaceDistribution::mean [private]`

Mean of the distribution.

Definition at line 144 of file `laplace_distribution.hpp`.

Referenced by `Dimensionality()`, `Mean()`, and `Random()`.

22.33.4.2 `double mlpack::distribution::LaplaceDistribution::scale [private]`

Scale parameter of the distribution.

Definition at line 146 of file `laplace_distribution.hpp`.

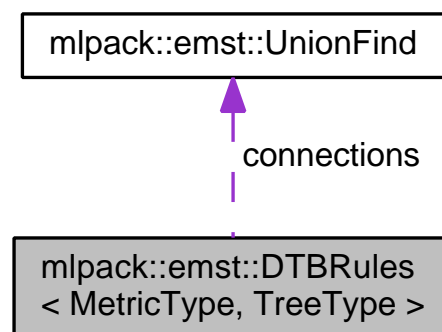
Referenced by `Random()`, and `Scale()`.

The documentation for this class was generated from the following file:

- `src/mlpack/core/dists/laplace_distribution.hpp`

22.34 mlpack::emst::DTBRules< MetricType, TreeType > Class Template Reference

Collaboration diagram for `mlpack::emst::DTBRules< MetricType, TreeType >`:



Public Types

- typedef **neighbor::NeighborSearchTraversallInfo** < TreeType > **TraversallInfoType**

Public Member Functions

- **DTBRules** (const arma::mat &**dataSet**, **UnionFind** &**connections**, arma::vec &**neighborsDistances**, arma::Col< size_t > &**neighborsInComponent**, arma::Col< size_t > &**neighborsOutComponent**, MetricType &**metric**)
- double **BaseCase** (const size_t queryIndex, const size_t referenceIndex)
- size_t **BaseCases** () const
Get the number of base cases performed.
- size_t & **BaseCases** ()
Modify the number of base cases performed.
- double **Rescore** (const size_t queryIndex, TreeType &referenceNode, const double oldScore)
Re-evaluate the score for recursion order.
- double **Rescore** (TreeType &queryNode, TreeType &referenceNode, const double oldScore) const
Re-evaluate the score for recursion order.
- double **Score** (const size_t queryIndex, TreeType &referenceNode)
Get the score for recursion order.
- double **Score** (const size_t queryIndex, TreeType &referenceNode, const double baseCaseResult)
Get the score for recursion order, passing the base case result (in the situation where it may be needed to calculate the recursion order).
- double **Score** (TreeType &queryNode, TreeType &referenceNode)
Get the score for recursion order.
- double **Score** (TreeType &queryNode, TreeType &referenceNode, const double baseCaseResult)
Get the score for recursion order, passing the base case result (in the situation where it may be needed to calculate the recursion order).
- size_t **Scores** () const
Get the number of node combinations that have been scored.
- size_t & **Scores** ()
Modify the number of node combinations that have been scored.
- const **TraversallInfoType** & **TraversallInfo** () const
- **TraversallInfoType** & **TraversallInfo** ()

Private Member Functions

- double **CalculateBound** (TreeType &queryNode) const
Update the bound for the given query node.

Private Attributes

- size_t **baseCases**
The number of base cases calculated.
- **UnionFind** & **connections**
Stores the tree structure so far.
- const arma::mat & **dataSet**
The data points.

- **MetricType & metric**
The instantiated metric.
- **arma::vec & neighborsDistances**
The distance to the candidate nearest neighbor for each component.
- **arma::Col< size_t > & neighborsInComponent**
The index of the point in the component that is an endpoint of the candidate edge.
- **arma::Col< size_t > & neighborsOutComponent**
The index of the point outside of the component that is an endpoint of the candidate edge.
- **size_t scores**
The number of node combinations that have been scored.
- **TraversallInfoType traversallInfo**

22.34.1 Detailed Description

template<typename MetricType, typename TreeType>class mlpack::emst::DTBRules< MetricType, TreeType >

Definition at line 25 of file dtb_rules.hpp.

22.34.2 Member Typedef Documentation

22.34.2.1 template<typename MetricType , typename TreeType > typedef neighbor::Neighbor↔
SearchTraversallInfo<TreeType> mlpack::emst::DTBRules< MetricType, TreeType
>::TraversallInfoType

Definition at line 115 of file dtb_rules.hpp.

22.34.3 Constructor & Destructor Documentation

22.34.3.1 template<typename MetricType , typename TreeType > mlpack::emst::DTBRules< MetricType, TreeType
>::DTBRules (const arma::mat & dataSet, UnionFind & connections, arma::vec & neighborsDistances, arma::Col<
size_t > & neighborsInComponent, arma::Col< size_t > & neighborsOutComponent, MetricType & metric)

22.34.4 Member Function Documentation

22.34.4.1 template<typename MetricType , typename TreeType > double mlpack::emst::DTBRules< MetricType, TreeType
>::BaseCase (const size_t queryIndex, const size_t referenceIndex)

22.34.4.2 template<typename MetricType , typename TreeType > size_t mlpack::emst::DTBRules< MetricType, TreeType
>::BaseCases () const [inline]

Get the number of base cases performed.

Definition at line 121 of file dtb_rules.hpp.

References mlpack::emst::DTBRules< MetricType, TreeType >::baseCases.

22.34.4.3 template<typename MetricType , typename TreeType > size_t& mlpack::emst::DTBRules< MetricType, TreeType
>::BaseCases () [inline]

Modify the number of base cases performed.

Definition at line 123 of file dtb_rules.hpp.

References `mlpack::emst::DTBRules< MetricType, TreeType >::baseCases`.

22.34.4.4 `template<typename MetricType , typename TreeType > double mlpack::emst::DTBRules< MetricType, TreeType >::CalculateBound (TreeType & queryNode) const` `[inline],[private]`

Update the bound for the given query node.

22.34.4.5 `template<typename MetricType , typename TreeType > double mlpack::emst::DTBRules< MetricType, TreeType >::Rescore (const size_t queryIndex, TreeType & referenceNode, const double oldScore)`

Re-evaluate the score for recursion order.

A low score indicates priority for recursion, while `DBL_MAX` indicates that the node should not be recursed into at all (it should be pruned). This is used when the score has already been calculated, but another recursion may have modified the bounds for pruning. So the old score is checked against the new pruning bound.

Parameters

<i>queryIndex</i>	Index of query point.
<i>referenceNode</i>	Candidate node to be recursed into.
<i>oldScore</i>	Old score produced by Score() (p. 228) (or Rescore() (p. 228)).

22.34.4.6 `template<typename MetricType , typename TreeType > double mlpack::emst::DTBRules< MetricType, TreeType >::Rescore (TreeType & queryNode, TreeType & referenceNode, const double oldScore) const`

Re-evaluate the score for recursion order.

A low score indicates priority for recursion, while `DBL_MAX` indicates that the node should not be recursed into at all (it should be pruned). This is used when the score has already been calculated, but another recursion may have modified the bounds for pruning. So the old score is checked against the new pruning bound.

Parameters

<i>queryNode</i>	Candidate query node to recurse into.
<i>referenceNode</i>	Candidate reference node to recurse into.
<i>oldScore</i>	Old score produced by Socre() (or Rescore() (p. 228)).

22.34.4.7 `template<typename MetricType , typename TreeType > double mlpack::emst::DTBRules< MetricType, TreeType >::Score (const size_t queryIndex, TreeType & referenceNode)`

Get the score for recursion order.

A low score indicates priority for recursion, while `DBL_MAX` indicates that the node should not be recursed into at all (it should be pruned).

Parameters

<i>queryIndex</i>	Index of query point.
-------------------	-----------------------

<i>referenceNode</i>	Candidate node to be recursed into.
----------------------	-------------------------------------

22.34.4.8 `template<typename MetricType , typename TreeType > double mlpack::emst::DTBRules< MetricType, TreeType >::Score (const size_t queryIndex, TreeType & referenceNode, const double baseCaseResult)`

Get the score for recursion order, passing the base case result (in the situation where it may be needed to calculate the recursion order).

A low score indicates priority for recursion, while DBL_MAX indicates that the node should not be recursed into at all (it should be pruned).

Parameters

<i>queryIndex</i>	Index of query point.
<i>referenceNode</i>	Candidate node to be recursed into.
<i>baseCaseResult</i>	Result of BaseCase(queryIndex, referenceNode).

22.34.4.9 `template<typename MetricType , typename TreeType > double mlpack::emst::DTBRules< MetricType, TreeType >::Score (TreeType & queryNode, TreeType & referenceNode)`

Get the score for recursion order.

A low score indicates priority for recursion while DBL_MAX indicates that the node should not be recursed into at all (it should be pruned).

Parameters

<i>queryNode</i>	Candidate query node to recurse into.
<i>referenceNode</i>	Candidate reference node to recurse into.

22.34.4.10 `template<typename MetricType , typename TreeType > double mlpack::emst::DTBRules< MetricType, TreeType >::Score (TreeType & queryNode, TreeType & referenceNode, const double baseCaseResult)`

Get the score for recursion order, passing the base case result (in the situation where it may be needed to calculate the recursion order).

A low score indicates priority for recursion, while DBL_MAX indicates that the node should not be recursed into at all (it should be pruned).

Parameters

<i>queryNode</i>	Candidate query node to recurse into.
<i>referenceNode</i>	Candidate reference node to recurse into.
<i>baseCaseResult</i>	Result of BaseCase(queryIndex, referenceNode).

22.34.4.11 `template<typename MetricType , typename TreeType > size_t mlpack::emst::DTBRules< MetricType, TreeType >::Scores () const [inline]`

Get the number of node combinations that have been scored.

Definition at line 126 of file dtb_rules.hpp.

References mlpack::emst::DTBRules< MetricType, TreeType >::scores.

22.34.4.12 `template<typename MetricType , typename TreeType > size_t mlpack::emst::DTBRules< MetricType, TreeType >::Scores () [inline]`

Modify the number of node combinations that have been scored.

Definition at line 128 of file dtb_rules.hpp.

References mlpack::emst::DTBRules< MetricType, TreeType >::scores.

22.34.4.13 `template<typename MetricType , typename TreeType > const TraversalInfoType& mlpack::emst::DTBRules< MetricType, TreeType >::TraversalInfo () const [inline]`

Definition at line 117 of file dtb_rules.hpp.

References mlpack::emst::DTBRules< MetricType, TreeType >::traversalInfo.

22.34.4.14 `template<typename MetricType , typename TreeType > TraversalInfoType& mlpack::emst::DTBRules< MetricType, TreeType >::TraversalInfo () [inline]`

Definition at line 118 of file dtb_rules.hpp.

References mlpack::emst::DTBRules< MetricType, TreeType >::traversalInfo.

22.34.5 Member Data Documentation

22.34.5.1 `template<typename MetricType , typename TreeType > size_t mlpack::emst::DTBRules< MetricType, TreeType >::baseCases [private]`

The number of base cases calculated.

Definition at line 159 of file dtb_rules.hpp.

Referenced by mlpack::emst::DTBRules< MetricType, TreeType >::BaseCases().

22.34.5.2 `template<typename MetricType , typename TreeType > UnionFind& mlpack::emst::DTBRules< MetricType, TreeType >::connections [private]`

Stores the tree structure so far.

Definition at line 135 of file dtb_rules.hpp.

22.34.5.3 `template<typename MetricType , typename TreeType > const arma::mat& mlpack::emst::DTBRules< MetricType, TreeType >::dataSet [private]`

The data points.

Definition at line 132 of file dtb_rules.hpp.

22.34.5.4 `template<typename MetricType , typename TreeType > MetricType& mlpack::emst::DTBRules< MetricType, TreeType >::metric [private]`

The instantiated metric.

Definition at line 149 of file dtb_rules.hpp.

22.34.5.5 `template<typename MetricType , typename TreeType > arma::vec& mlpack::emst::DTBRules< MetricType, TreeType >::neighborsDistances [private]`

The distance to the candidate nearest neighbor for each component.

Definition at line 138 of file dtb_rules.hpp.

22.34.5.6 `template<typename MetricType , typename TreeType > arma::Col<size_t>& mlpack::emst::DTBRules< MetricType, TreeType >::neighborsInComponent [private]`

The index of the point in the component that is an endpoint of the candidate edge.

Definition at line 142 of file dtb_rules.hpp.

22.34.5.7 `template<typename MetricType , typename TreeType > arma::Col<size_t>& mlpack::emst::DTBRules< MetricType, TreeType >::neighborsOutComponent [private]`

The index of the point outside of the component that is an endpoint of the candidate edge.

Definition at line 146 of file dtb_rules.hpp.

22.34.5.8 `template<typename MetricType , typename TreeType > size_t mlpack::emst::DTBRules< MetricType, TreeType >::scores [private]`

The number of node combinations that have been scored.

Definition at line 161 of file dtb_rules.hpp.

Referenced by `mlpack::emst::DTBRules< MetricType, TreeType >::Scores()`.

22.34.5.9 `template<typename MetricType , typename TreeType > TraversalInfoType mlpack::emst::DTBRules< MetricType, TreeType >::traversalInfo [private]`

Definition at line 156 of file dtb_rules.hpp.

Referenced by `mlpack::emst::DTBRules< MetricType, TreeType >::TraversalInfo()`.

The documentation for this class was generated from the following file:

- `src/mlpack/methods/emst/dtb_rules.hpp`

22.35 mlpack::emst::DTBStat Class Reference

A statistic for use with MLPACK trees, which stores the upper bound on distance to nearest neighbors and the component which this node belongs to.

Public Member Functions

- **DTBStat ()**
A generic initializer.
- `template<typename TreeType > DTBStat (const TreeType &node)`

This is called when a node is finished initializing.

- double **Bound** () const
Get the total bound for pruning.
- double & **Bound** ()
Modify the total bound for pruning.
- int **ComponentMembership** () const
Get the component membership of this node.
- int & **ComponentMembership** ()
Modify the component membership of this node.
- double **MaxNeighborDistance** () const
Get the maximum neighbor distance.
- double & **MaxNeighborDistance** ()
Modify the maximum neighbor distance.
- double **MinNeighborDistance** () const
Get the minimum neighbor distance.
- double & **MinNeighborDistance** ()
Modify the minimum neighbor distance.

Private Attributes

- double **bound**
Total bound for pruning.
- int **componentMembership**
The index of the component that all points in this node belong to.
- double **maxNeighborDistance**
Upper bound on the distance to the nearest neighbor of any point in this node.
- double **minNeighborDistance**
Lower bound on the distance to the nearest neighbor of any point in this node.

22.35.1 Detailed Description

A statistic for use with MLPACK trees, which stores the upper bound on distance to nearest neighbors and the component which this node belongs to.

Definition at line 26 of file dtb_stat.hpp.

22.35.2 Constructor & Destructor Documentation

22.35.2.1 mlpack::emst::DTBStat::DTBStat () [inline]

A generic initializer.

Sets the maximum neighbor distance to its default, and the component membership to -1 (no component).

Definition at line 51 of file dtb_stat.hpp.

22.35.2.2 `template<typename TreeType > mlpack::emst::DTBStat::DTBStat (const TreeType & node) [inline]`

This is called when a node is finished initializing.

We set the maximum neighbor distance to its default, and if possible, we set the component membership of the node (if it has only one point and no children).

Parameters

<i>node</i>	Node that has been finished.
-------------	------------------------------

Definition at line 65 of file dtb_stat.hpp.

22.35.3 Member Function Documentation

22.35.3.1 `double mlpack::emst::DTBStat::Bound () const [inline]`

Get the total bound for pruning.

Definition at line 84 of file dtb_stat.hpp.

References bound.

22.35.3.2 `double& mlpack::emst::DTBStat::Bound () [inline]`

Modify the total bound for pruning.

Definition at line 86 of file dtb_stat.hpp.

References bound.

22.35.3.3 `int mlpack::emst::DTBStat::ComponentMembership () const [inline]`

Get the component membership of this node.

Definition at line 89 of file dtb_stat.hpp.

References componentMembership.

22.35.3.4 `int& mlpack::emst::DTBStat::ComponentMembership () [inline]`

Modify the component membership of this node.

Definition at line 91 of file dtb_stat.hpp.

References componentMembership.

22.35.3.5 `double mlpack::emst::DTBStat::MaxNeighborDistance () const [inline]`

Get the maximum neighbor distance.

Definition at line 74 of file dtb_stat.hpp.

References maxNeighborDistance.

22.35.3.6 `double& mlpack::emst::DTBStat::MaxNeighborDistance () [inline]`

Modify the maximum neighbor distance.

Definition at line 76 of file dtb_stat.hpp.

References maxNeighborDistance.

22.35.3.7 `double mlpack::emst::DTBStat::MinNeighborDistance () const [inline]`

Get the minimum neighbor distance.

Definition at line 79 of file dtb_stat.hpp.

References minNeighborDistance.

22.35.3.8 `double& mlpack::emst::DTBStat::MinNeighborDistance () [inline]`

Modify the minimum neighbor distance.

Definition at line 81 of file dtb_stat.hpp.

References minNeighborDistance.

22.35.4 Member Data Documentation

22.35.4.1 `double mlpack::emst::DTBStat::bound [private]`

Total bound for pruning.

Definition at line 38 of file dtb_stat.hpp.

Referenced by Bound().

22.35.4.2 `int mlpack::emst::DTBStat::componentMembership [private]`

The index of the component that all points in this node belong to.

This is the same index returned by **UnionFind** (p. 245) for all points in this node. If points in this node are in different components, this value will be negative.

Definition at line 44 of file dtb_stat.hpp.

Referenced by ComponentMembership().

22.35.4.3 `double mlpack::emst::DTBStat::maxNeighborDistance [private]`

Upper bound on the distance to the nearest neighbor of any point in this node.

Definition at line 31 of file dtb_stat.hpp.

Referenced by MaxNeighborDistance().

22.35.4.4 `double mlpack::emst::DTBStat::minNeighborDistance [private]`

Lower bound on the distance to the nearest neighbor of any point in this node.

Definition at line 35 of file dtb_stat.hpp.

Referenced by MinNeighborDistance().

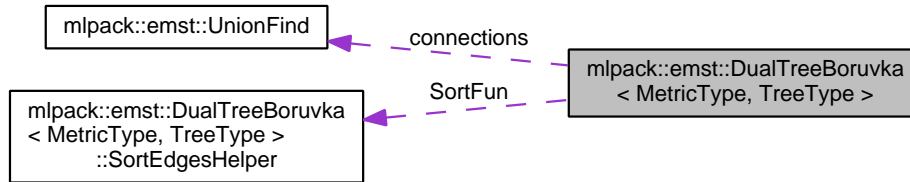
The documentation for this class was generated from the following file:

- src/mlpack/methods/emst/dtb_stat.hpp

22.36 mlpack::emst::DualTreeBoruvka< MetricType, TreeType > Class Template Reference

Performs the MST calculation using the Dual-Tree Boruvka algorithm, using any type of tree.

Collaboration diagram for mlpack::emst::DualTreeBoruvka< MetricType, TreeType >:



Classes

- struct **SortEdgesHelper**

For sorting the edge list after the computation.

Public Member Functions

- **DualTreeBoruvka** (const typename TreeType::Mat &dataset, const bool **naive**=false, const MetricType **metric**=MetricType())
Create the tree from the given dataset.
- **DualTreeBoruvka** (TreeType ***tree**, const typename TreeType::Mat &dataset, const MetricType **metric**=MetricType())
*Create the **DualTreeBoruvka** (p. 236) object with an already initialized tree.*
- **~DualTreeBoruvka** ()
Delete the tree, if it was created inside the object.
- void **ComputeMST** (arma::mat &results)
Iteratively find the nearest neighbor of each component until the MST is complete.
- std::string **Tostring** () const
Returns a string representation of this object.

Private Member Functions

- void **AddAllEdges** ()
Adds all the edges found in one iteration to the list of neighbors.
- void **AddEdge** (const size_t e1, const size_t e2, const double distance)
Adds a single edge to the edge list.
- void **Cleanup** ()
The values stored in the tree must be reset on each iteration.
- void **CleanupHelper** (TreeType ***tree**)
This function resets the values in the nodes of the tree nearest neighbor distance, and checks for fully connected nodes.
- void **EmitResults** (arma::mat &results)
Unpermute the edge list and output it to results.

Private Attributes

- **UnionFind connections**
Connections.
- **const TreeType::Mat & data**
Reference to the data (this is what should be used for accessing data).
- **TreeType::Mat dataCopy**
Copy of the data (if necessary).
- **std::vector< EdgePair > edges**
Edges.
- **MetricType metric**
The instantiated metric.
- **bool naive**
Indicates whether or not $O(n^2)$ naive mode will be used.
- **arma::vec neighborsDistances**
List of edge distances.
- **arma::Col< size_t > neighborsInComponent**
List of edge nodes.
- **arma::Col< size_t > neighborsOutComponent**
List of edge nodes.
- **std::vector< size_t > oldFromNew**
Permutations of points during tree building.
- **bool ownTree**
Indicates whether or not we "own" the tree.
- **struct mlpack::emst::DualTreeBoruvka::SortEdgesHelper SortFun**
- **double totalDist**
Total distance of the tree.
- **TreeType * tree**
Pointer to the root of the tree.

22.36.1 Detailed Description

```
template<typename MetricType = metric::EuclideanDistance, typename TreeType = tree::BinarySpaceTree<bound::HRectBound<2>,
DTBStat>>class mlpack::emst::DualTreeBoruvka< MetricType, TreeType >
```

Performs the MST calculation using the Dual-Tree Boruvka algorithm, using any type of tree.

For more information on the algorithm, see the following citation:

```
@inproceedings{
  author = {March, W.B., Ram, P., and Gray, A.G.},
  title = {{Fast Euclidean Minimum Spanning Tree: Algorithm, Analysis,
    Applications.}},
  booktitle = {Proceedings of the 16th ACM SIGKDD International Conference
    on Knowledge Discovery and Data Mining}
  series = {KDD 2010},
  year = {2010}
}
```

General usage of this class might be like this:

```
extern arma::mat data; // We want to find the MST of this dataset.
DualTreeBoruvka<> dtb(data); // Create the tree with default options.

// Find the MST.
arma::mat mstResults;
dtb.ComputeMST(mstResults);
```

More advanced usage of the class can use different types of trees, pass in an already-built tree, or compute the MST using the $O(n^2)$ naive algorithm.

Template Parameters

<i>MetricType</i>	The metric to use. IMPORTANT: this hasn't really been tested with anything other than the L2 metric, so user beware. Note that the tree type needs to compute bounds using the same metric as the type specified here.
<i>TreeType</i>	Type of tree to use. Should use DTBStat (p. 231) as a statistic.

Definition at line 83 of file dtb.hpp.

22.36.2 Constructor & Destructor Documentation

22.36.2.1 `template<typename MetricType = metric::EuclideanDistance, typename TreeType = tree::BinarySpaceTree<bound::H↔RectBound<2>, DTBStat>> mlpack::emst::DualTreeBoruvka< MetricType, TreeType >::DualTreeBoruvka (const typename TreeType::Mat & dataset, const bool naive = false, const MetricType metric = MetricType())`

Create the tree from the given dataset.

This copies the dataset to an internal copy, because tree-building modifies the dataset.

Parameters

<i>data</i>	Dataset to build a tree for.
<i>naive</i>	Whether the computation should be done in $O(n^2)$ naive mode.
<i>leafSize</i>	The leaf size to be used during tree construction.

22.36.2.2 `template<typename MetricType = metric::EuclideanDistance, typename TreeType = tree::BinarySpaceTree<bound::H↔RectBound<2>, DTBStat>> mlpack::emst::DualTreeBoruvka< MetricType, TreeType >::DualTreeBoruvka (TreeType * tree, const typename TreeType::Mat & dataset, const MetricType metric = MetricType())`

Create the **DualTreeBoruvka** (p. 236) object with an already initialized tree.

This will not copy the dataset, and can save a little processing power. Naive mode is not available as an option for this constructor; instead, to run naive computation, construct a tree with all the points in one leaf (i.e. leafSize = number of points).

Note

Because tree-building (at least with BinarySpaceTree) modifies the ordering of a matrix, be sure you pass the modified matrix to this object! In addition, mapping the points of the matrix back to their original indices is not done when this constructor is used.

Parameters

<i>tree</i>	Pre-built tree.
<i>dataset</i>	Dataset corresponding to the pre-built tree.

22.36.2.3 `template<typename MetricType = metric::EuclideanDistance, typename TreeType = tree::BinarySpaceTree<bound::H↔RectBound<2>, DTBStat>> mlpack::emst::DualTreeBoruvka< MetricType, TreeType >::~~DualTreeBoruvka ()`

Delete the tree, if it was created inside the object.

22.36.3 Member Function Documentation

22.36.3.1 `template<typename MetricType = metric::EuclideanDistance, typename TreeType = tree::BinarySpaceTree<bound::H↔RectBound<2>, DTBStat>> void mlpack::emst::DualTreeBoruvka< MetricType, TreeType >::AddAllEdges ()`
[private]

Adds all the edges found in one iteration to the list of neighbors.

22.36.3.2 `template<typename MetricType = metric::EuclideanDistance, typename TreeType = tree::BinarySpaceTree<bound::H↔RectBound<2>, DTBStat>> void mlpack::emst::DualTreeBoruvka< MetricType, TreeType >::AddEdge (const size_t e1, const size_t e2, const double distance)` [private]

Adds a single edge to the edge list.

22.36.3.3 `template<typename MetricType = metric::EuclideanDistance, typename TreeType = tree::BinarySpaceTree<bound::H↔RectBound<2>, DTBStat>> void mlpack::emst::DualTreeBoruvka< MetricType, TreeType >::Cleanup ()`
[private]

The values stored in the tree must be reset on each iteration.

22.36.3.4 `template<typename MetricType = metric::EuclideanDistance, typename TreeType = tree::BinarySpaceTree<bound::H↔RectBound<2>, DTBStat>> void mlpack::emst::DualTreeBoruvka< MetricType, TreeType >::CleanupHelper (TreeType * tree)` [private]

This function resets the values in the nodes of the tree nearest neighbor distance, and checks for fully connected nodes.

22.36.3.5 `template<typename MetricType = metric::EuclideanDistance, typename TreeType = tree::BinarySpaceTree<bound::H↔RectBound<2>, DTBStat>> void mlpack::emst::DualTreeBoruvka< MetricType, TreeType >::ComputeMST (arma::mat & results)`

Iteratively find the nearest neighbor of each component until the MST is complete.

The results will be a 3xN matrix (with N equal to the number of edges in the minimum spanning tree). The first row will contain the lesser index of the edge; the second row will contain the greater index of the edge; and the third row will contain the distance between the two edges.

Parameters

<i>results</i>	Matrix which results will be stored in.
----------------	---

22.36.3.6 `template<typename MetricType = metric::EuclideanDistance, typename TreeType = tree::BinarySpaceTree<bound::H↔
RectBound<2>, DTBStat>> void mlpack::emst::DualTreeBoruvka< MetricType, TreeType >::EmitResults (arma::mat & results) [private]`

Unpermute the edge list and output it to results.

22.36.3.7 `template<typename MetricType = metric::EuclideanDistance, typename TreeType = tree::BinarySpaceTree<bound::H↔
RectBound<2>, DTBStat>> std::string mlpack::emst::DualTreeBoruvka< MetricType, TreeType >::ToString () const`

Returns a string representation of this object.

22.36.4 Member Data Documentation

22.36.4.1 `template<typename MetricType = metric::EuclideanDistance, typename TreeType = tree::BinarySpaceTree<bound::H↔
HRectBound<2>, DTBStat>> UnionFind mlpack::emst::DualTreeBoruvka< MetricType, TreeType >::connections [private]`

Connections.

Definition at line 103 of file dtb.hpp.

22.36.4.2 `template<typename MetricType = metric::EuclideanDistance, typename TreeType = tree::BinarySpaceTree<bound::H↔
RectBound<2>, DTBStat>> const TreeType::Mat& mlpack::emst::DualTreeBoruvka< MetricType, TreeType >::data [private]`

Reference to the data (this is what should be used for accessing data).

Definition at line 89 of file dtb.hpp.

22.36.4.3 `template<typename MetricType = metric::EuclideanDistance, typename TreeType = tree::BinarySpaceTree<bound::H↔
RectBound<2>, DTBStat>> TreeType::Mat mlpack::emst::DualTreeBoruvka< MetricType, TreeType >::dataCopy [private]`

Copy of the data (if necessary).

Definition at line 87 of file dtb.hpp.

22.36.4.4 `template<typename MetricType = metric::EuclideanDistance, typename TreeType = tree::BinarySpaceTree<bound::H↔
RectBound<2>, DTBStat>> std::vector<EdgePair> mlpack::emst::DualTreeBoruvka< MetricType, TreeType >::edges [private]`

Edges.

Definition at line 100 of file dtb.hpp.

22.36.4.5 `template<typename MetricType = metric::EuclideanDistance, typename TreeType = tree::BinarySpaceTree<bound::H↵
RectBound<2>, DTBStat>> MetricType mlpack::emst::DualTreeBoruvka< MetricType, TreeType >::metric
[private]`

The instantiated metric.

Definition at line 118 of file dtb.hpp.

22.36.4.6 `template<typename MetricType = metric::EuclideanDistance, typename TreeType = tree::BinarySpaceTree<bound::H↵
HRectBound<2>, DTBStat>> bool mlpack::emst::DualTreeBoruvka< MetricType, TreeType >::naive
[private]`

Indicates whether or not $O(n^2)$ naive mode will be used.

Definition at line 97 of file dtb.hpp.

22.36.4.7 `template<typename MetricType = metric::EuclideanDistance, typename TreeType = tree::BinarySpaceTree<bound↵
::HRectBound<2>, DTBStat>> arma::vec mlpack::emst::DualTreeBoruvka< MetricType, TreeType
>::neighborsDistances [private]`

List of edge distances.

Definition at line 112 of file dtb.hpp.

22.36.4.8 `template<typename MetricType = metric::EuclideanDistance, typename TreeType = tree::BinarySpaceTree<bound::H↵
HRectBound<2>, DTBStat>> arma::Col<size_t> mlpack::emst::DualTreeBoruvka< MetricType, TreeType
>::neighborsInComponent [private]`

List of edge nodes.

Definition at line 108 of file dtb.hpp.

22.36.4.9 `template<typename MetricType = metric::EuclideanDistance, typename TreeType = tree::BinarySpaceTree<bound::H↵
HRectBound<2>, DTBStat>> arma::Col<size_t> mlpack::emst::DualTreeBoruvka< MetricType, TreeType
>::neighborsOutComponent [private]`

List of edge nodes.

Definition at line 110 of file dtb.hpp.

22.36.4.10 `template<typename MetricType = metric::EuclideanDistance, typename TreeType = tree::BinarySpaceTree<bound::H↵
RectBound<2>, DTBStat>> std::vector<size_t> mlpack::emst::DualTreeBoruvka< MetricType, TreeType
>::oldFromNew [private]`

Permutations of points during tree building.

Definition at line 106 of file dtb.hpp.

22.36.4.11 `template<typename MetricType = metric::EuclideanDistance, typename TreeType = tree::BinarySpaceTree<bound↵
::HRectBound<2>, DTBStat>> bool mlpack::emst::DualTreeBoruvka< MetricType, TreeType >::ownTree
[private]`

Indicates whether or not we "own" the tree.

Definition at line 94 of file dtb.hpp.

```
22.36.4.12 template<typename MetricType = metric::EuclideanDistance, typename TreeType = tree::BinarySpaceTree<bound::HRectBound<2>, DTBStat>> struct mlpack::emst::DualTreeBoruvka::SortEdgesHelper
mlpack::emst::DualTreeBoruvka< MetricType, TreeType >::SortFun [private]
```

```
22.36.4.13 template<typename MetricType = metric::EuclideanDistance, typename TreeType = tree::BinarySpaceTree<bound::HRectBound<2>, DTBStat>> double mlpack::emst::DualTreeBoruvka< MetricType, TreeType >::totalDist
[private]
```

Total distance of the tree.

Definition at line 115 of file dtb.hpp.

```
22.36.4.14 template<typename MetricType = metric::EuclideanDistance, typename TreeType = tree::BinarySpaceTree<bound::HRectBound<2>, DTBStat>> TreeType* mlpack::emst::DualTreeBoruvka< MetricType, TreeType >::tree
[private]
```

Pointer to the root of the tree.

Definition at line 92 of file dtb.hpp.

The documentation for this class was generated from the following file:

- src/mlpack/methods/emst/dtb.hpp

22.37 mlpack::emst::DualTreeBoruvka< MetricType, TreeType >::SortEdgesHelper Struct Reference

For sorting the edge list after the computation.

Public Member Functions

- bool **operator()** (const **EdgePair** &pairA, const **EdgePair** &pairB)

22.37.1 Detailed Description

```
template<typename MetricType = metric::EuclideanDistance, typename TreeType = tree::BinarySpaceTree<bound::HRectBound<2>, DTBStat>> struct mlpack::emst::DualTreeBoruvka< MetricType, TreeType >::SortEdgesHelper
```

For sorting the edge list after the computation.

Definition at line 121 of file dtb.hpp.

22.37.2 Member Function Documentation

```
22.37.2.1 template<typename MetricType = metric::EuclideanDistance, typename TreeType = tree::BinarySpace↵
    Tree<bound::HRectBound<2>, DTBStat>> bool mlpack::emst::DualTreeBoruvka< MetricType, TreeType
    >::SortEdgesHelper::operator() ( const EdgePair & pairA, const EdgePair & pairB ) [inline]
```

Definition at line 123 of file dtb.hpp.

References mlpack::emst::EdgePair::Distance().

The documentation for this struct was generated from the following file:

- src/mlpack/methods/emst/dtb.hpp

22.38 mlpack::emst::EdgePair Class Reference

An edge pair is simply two indices and a distance.

Public Member Functions

- **EdgePair** (const size_t **lesser**, const size_t **greater**, const double dist)
*Initialize an **EdgePair** (p. 243) with two indices and a distance.*
- double **Distance** () const
Get the distance.
- double & **Distance** ()
Modify the distance.
- size_t **Greater** () const
Get the greater index.
- size_t & **Greater** ()
Modify the greater index.
- size_t **Lesser** () const
Get the lesser index.
- size_t & **Lesser** ()
Modify the lesser index.

Private Attributes

- double **distance**
Distance between two indices.
- size_t **greater**
Greater index.
- size_t **lesser**
Lesser index.

22.38.1 Detailed Description

An edge pair is simply two indices and a distance.

It is used as the basic element of an edge list when computing a minimum spanning tree.

Definition at line 30 of file edge_pair.hpp.

22.38.2 Constructor & Destructor Documentation

22.38.2.1 `mlpack::emst::EdgePair::EdgePair (const size_t lesser, const size_t greater, const double dist) [inline]`

Initialize an **EdgePair** (p. 243) with two indices and a distance.

The indices are called lesser and greater, implying that they be sorted before calling Init. However, this is not necessary for functionality; it is just a way to keep the edge list organized in other code.

Definition at line 47 of file `edge_pair.hpp`.

References `mlpack::Log::Assert()`.

22.38.3 Member Function Documentation

22.38.3.1 `double mlpack::emst::EdgePair::Distance () const [inline]`

Get the distance.

Definition at line 65 of file `edge_pair.hpp`.

References `distance`.

Referenced by `mlpack::emst::DualTreeBoruvka< MetricType, TreeType >::SortEdgesHelper::operator()()`.

22.38.3.2 `double& mlpack::emst::EdgePair::Distance () [inline]`

Modify the distance.

Definition at line 67 of file `edge_pair.hpp`.

References `distance`.

22.38.3.3 `size_t mlpack::emst::EdgePair::Greater () const [inline]`

Get the greater index.

Definition at line 60 of file `edge_pair.hpp`.

References `greater`.

22.38.3.4 `size_t& mlpack::emst::EdgePair::Greater () [inline]`

Modify the greater index.

Definition at line 62 of file `edge_pair.hpp`.

References `greater`.

22.38.3.5 `size_t mlpack::emst::EdgePair::Lesser () const [inline]`

Get the lesser index.

Definition at line 55 of file `edge_pair.hpp`.

References `lesser`.

22.38.3.6 `size_t& mlpack::emst::EdgePair::Lesser ()` [inline]

Modify the lesser index.

Definition at line 57 of file `edge_pair.hpp`.

References `lesser`.

22.38.4 Member Data Documentation

22.38.4.1 `double mlpack::emst::EdgePair::distance` [private]

Distance between two indices.

Definition at line 38 of file `edge_pair.hpp`.

Referenced by `Distance()`.

22.38.4.2 `size_t mlpack::emst::EdgePair::greater` [private]

Greater index.

Definition at line 36 of file `edge_pair.hpp`.

Referenced by `Greater()`.

22.38.4.3 `size_t mlpack::emst::EdgePair::lesser` [private]

Lesser index.

Definition at line 34 of file `edge_pair.hpp`.

Referenced by `Lesser()`.

The documentation for this class was generated from the following file:

- `src/mlpack/methods/emst/edge_pair.hpp`

22.39 mlpack::emst::UnionFind Class Reference

A Union-Find data structure.

Public Member Functions

- **UnionFind** (const `size_t` size)
Construct the object with the given size.
- **~UnionFind** ()
Destroy the object (nothing to do).
- `size_t` **Find** (const `size_t` x)
Returns the component containing an element.
- void **Union** (const `size_t` x, const `size_t` y)
Union the components containing x and y.

Private Attributes

- arma::Col< size_t > **parent**
- arma::ivec **rank**

22.39.1 Detailed Description

A Union-Find data structure.

See Cormen, Rivest, & Stein for details. The structure tracks the components of a graph. Each point in the graph is initially in its own component. Calling Union(x, y) unites the components indexed by x and y. Find(x) returns the index of the component containing point x.

Definition at line 32 of file union_find.hpp.

22.39.2 Constructor & Destructor Documentation

22.39.2.1 `mlpack::emst::UnionFind (const size_t size) [inline]`

Construct the object with the given size.

Definition at line 40 of file union_find.hpp.

22.39.2.2 `mlpack::emst::UnionFind::~~UnionFind () [inline]`

Destroy the object (nothing to do).

Definition at line 50 of file union_find.hpp.

22.39.3 Member Function Documentation

22.39.3.1 `size_t mlpack::emst::UnionFind::Find (const size_t x) [inline]`

Returns the component containing an element.

Parameters

<code>x</code>	the component to be found
----------------	---------------------------

Returns

The index of the component containing x

Definition at line 58 of file union_find.hpp.

Referenced by Union().

22.39.3.2 `void mlpack::emst::UnionFind::Union (const size_t x, const size_t y) [inline]`

Union the components containing x and y.

Parameters

x	one component
y	the other component

Definition at line 78 of file union_find.hpp.

References Find().

22.39.4 Member Data Documentation

22.39.4.1 arma::Col<size_t> mlpack::emst::UnionFind::parent [private]

Definition at line 35 of file union_find.hpp.

22.39.4.2 arma::ivec mlpack::emst::UnionFind::rank [private]

Definition at line 36 of file union_find.hpp.

The documentation for this class was generated from the following file:

- src/mlpack/methods/emst/union_find.hpp

22.40 mlpack::fastmks::FastMKS< KernelType, TreeType > Class Template Reference

An implementation of fast exact max-kernel search.

Public Member Functions

- **FastMKS** (const arma::mat &referenceSet, const bool **single**=false, const bool **naive**=false)
Create the **FastMKS** (p. 247) object using the reference set as the query set.
- **FastMKS** (const arma::mat &referenceSet, const arma::mat &querySet, const bool **single**=false, const bool **naive**=false)
Create the **FastMKS** (p. 247) object using separate reference and query sets.
- **FastMKS** (const arma::mat &referenceSet, KernelType &kernel, const bool **single**=false, const bool **naive**=false)
Create the **FastMKS** (p. 247) object using the reference set as the query set, and with an initialized kernel.
- **FastMKS** (const arma::mat &referenceSet, const arma::mat &querySet, KernelType &kernel, const bool **single**=false, const bool **naive**=false)
Create the **FastMKS** (p. 247) object using separate reference and query sets, and with an initialized kernel.
- **FastMKS** (const arma::mat &referenceSet, TreeType *referenceTree, const bool **single**=false, const bool **naive**=false)
Create the **FastMKS** (p. 247) object with an already-initialized tree built on the reference points.
- **FastMKS** (const arma::mat &referenceSet, TreeType *referenceTree, const arma::mat &querySet, TreeType *queryTree, const bool **single**=false, const bool **naive**=false)
Create the **FastMKS** (p. 247) object with already-initialized trees built on the reference and query points.
- **~FastMKS** ()
Destructor for the **FastMKS** (p. 247) object.
- const **metric::IPMetric**< KernelType > & **Metric** () const
Get the inner-product metric induced by the given kernel.

- **metric::IPMetric**< KernelType > & **Metric** ()
Modify the inner-product metric induced by the given kernel.
- void **Search** (const size_t k, arma::Mat< size_t > &indices, arma::mat &products)
Search for the maximum inner products of the query set (or if no query set was passed, the reference set is used).
- std::string **ToString** () const
Returns a string representation of this object.

Private Member Functions

- void **InsertNeighbor** (arma::Mat< size_t > &indices, arma::mat &products, const size_t queryIndex, const size_t pos, const size_t neighbor, const double distance)
Utility function. Copied too many times from too many places.

Private Attributes

- **metric::IPMetric**< KernelType > **metric**
The instantiated inner-product metric induced by the given kernel.
- bool **naive**
If true, naive (brute-force) search is used.
- const arma::mat & **querySet**
The query dataset.
- TreeType * **queryTree**
The tree built on the query dataset.
- const arma::mat & **referenceSet**
The reference dataset.
- TreeType * **referenceTree**
The tree built on the reference dataset.
- bool **single**
If true, single-tree search is used.
- bool **treeOwner**
If true, this object created the trees and is responsible for them.

22.40.1 Detailed Description

```
template<typename KernelType, typename TreeType = tree::CoverTree<metric::IPMetric<KernelType>, tree::FirstPointsRoot, FastMKStat>> class mlpack::fastmks::FastMKS< KernelType, TreeType >
```

An implementation of fast exact max-kernel search.

Given a query dataset and a reference dataset (or optionally just a reference dataset which is also used as the query dataset), fast exact max-kernel search finds, for each point in the query dataset, the k points in the reference set with maximum kernel value $K(p_q, p_r)$, where k is a specified parameter and $K()$ is a Mercer kernel.

For more information, see the following paper.

```
@inproceedings{curtin2013fast,
  title={Fast Exact Max-Kernel Search},
  author={Curtin, Ryan R. and Ram, Parikshit and Gray, Alexander G.},
  booktitle={Proceedings of the 2013 SIAM International Conference on Data Mining (SDM 13)},
  year={2013}
}
```

This class allows specification of the type of kernel and also of the type of tree. **FastMKS** (p. 247) can be run on kernels that work on arbitrary objects – however, this only works with cover trees and other trees that are built only on points in the dataset (and not centroids of regions or anything like that).

Template Parameters

<i>KernelType</i>	Type of kernel to run FastMKS (p. 247) with.
<i>TreeType</i>	Type of tree to run FastMKS (p. 247) with; it must have metric IPMetric<KernelType>.

Definition at line 61 of file fastmks.hpp.

22.40.2 Constructor & Destructor Documentation

22.40.2.1 `template<typename KernelType, typename TreeType = tree::CoverTree<metric::IPMetric<KernelType>, tree::FirstPointIsRoot, FastMKSSStat>> mlpack::fastmks::FastMKS< KernelType, TreeType >::FastMKS (const arma::mat & referenceSet, const bool single = false, const bool naive = false)`

Create the **FastMKS** (p. 247) object using the reference set as the query set.

Optionally, specify whether or not single-tree search or naive (brute-force) search should be used.

Parameters

<i>referenceSet</i>	Set of data to run FastMKS (p. 247) on.
<i>single</i>	Whether or not to run single-tree search.
<i>naive</i>	Whether or not to run brute-force (naive) search.

22.40.2.2 `template<typename KernelType, typename TreeType = tree::CoverTree<metric::IPMetric<KernelType>, tree::FirstPointIsRoot, FastMKSSStat>> mlpack::fastmks::FastMKS< KernelType, TreeType >::FastMKS (const arma::mat & referenceSet, const arma::mat & querySet, const bool single = false, const bool naive = false)`

Create the **FastMKS** (p. 247) object using separate reference and query sets.

Optionally, specify whether or not single-tree search or naive (brute-force) search should be used.

Parameters

<i>referenceSet</i>	Reference set of data for FastMKS (p. 247).
<i>querySet</i>	Set of query points for FastMKS (p. 247).
<i>single</i>	Whether or not to run single-tree search.
<i>naive</i>	Whether or not to run brute-force (naive) search.

22.40.2.3 `template<typename KernelType, typename TreeType = tree::CoverTree<metric::IPMetric<KernelType>, tree::FirstPointIsRoot, FastMKSSStat>> mlpack::fastmks::FastMKS< KernelType, TreeType >::FastMKS (const arma::mat & referenceSet, KernelType & kernel, const bool single = false, const bool naive = false)`

Create the **FastMKS** (p. 247) object using the reference set as the query set, and with an initialized kernel.

This is useful for when the kernel stores state. Optionally, specify whether or not single-tree search or naive (brute-force) search should be used.

Parameters

<i>referenceSet</i>	Reference set of data for FastMKS (p. 247).
<i>kernel</i>	Initialized kernel.
<i>single</i>	Whether or not to run single-tree search.
<i>naive</i>	Whether or not to run brute-force (naive) search.

```
22.40.2.4  template<typename KernelType, typename TreeType = tree::CoverTree<metric::IPMetric<KernelType>,
tree::FirstPointsRoot, FastMKSSStat>> mlpack::fastmks::FastMKS< KernelType, TreeType >::FastMKS ( const
arma::mat & referenceSet, const arma::mat & querySet, KernelType & kernel, const bool single = false, const bool
naive = false )
```

Create the **FastMKS** (p. 247) object using separate reference and query sets, and with an initialized kernel.

This is useful for when the kernel stores state. Optionally, specify whether or not single-tree search or naive (brute-force) search should be used.

Parameters

<i>referenceSet</i>	Reference set of data for FastMKS (p. 247).
<i>querySet</i>	Set of query points for FastMKS (p. 247).
<i>kernel</i>	Initialized kernel.
<i>single</i>	Whether or not to run single-tree search.
<i>naive</i>	Whether or not to run brute-force (naive) search.

```
22.40.2.5  template<typename KernelType, typename TreeType = tree::CoverTree<metric::IPMetric<KernelType>,
tree::FirstPointsRoot, FastMKSSStat>> mlpack::fastmks::FastMKS< KernelType, TreeType >::FastMKS ( const
arma::mat & referenceSet, TreeType * referenceTree, const bool single = false, const bool naive = false )
```

Create the **FastMKS** (p. 247) object with an already-initialized tree built on the reference points.

Be sure that the tree is built with the metric type `IPMetric<KernelType>`. For this constructor, the reference set and the query set are the same points. Optionally, whether or not to run single-tree search or brute-force (naive) search can be specified.

Parameters

<i>referenceSet</i>	Reference set of data for FastMKS (p. 247).
<i>referenceTree</i>	Tree built on reference data.
<i>single</i>	Whether or not to run single-tree search.
<i>naive</i>	Whether or not to run brute-force (naive) search.

```
22.40.2.6  template<typename KernelType, typename TreeType = tree::CoverTree<metric::IPMetric<KernelType>,
tree::FirstPointsRoot, FastMKSSStat>> mlpack::fastmks::FastMKS< KernelType, TreeType >::FastMKS ( const
arma::mat & referenceSet, TreeType * referenceTree, const arma::mat & querySet, TreeType * queryTree, const bool
single = false, const bool naive = false )
```

Create the **FastMKS** (p. 247) object with already-initialized trees built on the reference and query points.

Be sure that the trees are built with the metric type `IPMetric<KernelType>`. Optionally, whether or not to run single-tree search or naive (brute-force) search can be specified.

Parameters

<i>referenceSet</i>	Reference set of data for FastMKS (p. 247).
<i>referenceTree</i>	Tree built on reference data.
<i>querySet</i>	Set of query points for FastMKS (p. 247).
<i>queryTree</i>	Tree built on query data.
<i>single</i>	Whether or not to use single-tree search.
<i>naive</i>	Whether or not to use naive (brute-force) search.

22.40.2.7 `template<typename KernelType, typename TreeType = tree::CoverTree<metric::IPMetric<KernelType>, tree::FirstPointsRoot, FastMKSSStat>> mlpack::fastmks::FastMKS< KernelType, TreeType >::~FastMKS ()`

Destructor for the **FastMKS** (p. 247) object.

22.40.3 Member Function Documentation

22.40.3.1 `template<typename KernelType, typename TreeType = tree::CoverTree<metric::IPMetric<KernelType>, tree::FirstPointsRoot, FastMKSSStat>> void mlpack::fastmks::FastMKS< KernelType, TreeType >::InsertNeighbor (arma::Mat< size_t > & indices, arma::mat & products, const size_t queryIndex, const size_t pos, const size_t neighbor, const double distance) [private]`

Utility function. Copied too many times from too many places.

22.40.3.2 `template<typename KernelType, typename TreeType = tree::CoverTree<metric::IPMetric<KernelType>, tree::FirstPointsRoot, FastMKSSStat>> const metric::IPMetric<KernelType> & mlpack::fastmks::FastMKS< KernelType, TreeType >::Metric () const [inline]`

Get the inner-product metric induced by the given kernel.

Definition at line 185 of file fastmks.hpp.

References mlpack::fastmks::FastMKS< KernelType, TreeType >::metric.

22.40.3.3 `template<typename KernelType, typename TreeType = tree::CoverTree<metric::IPMetric<KernelType>, tree::FirstPointsRoot, FastMKSSStat>> metric::IPMetric<KernelType> & mlpack::fastmks::FastMKS< KernelType, TreeType >::Metric () [inline]`

Modify the inner-product metric induced by the given kernel.

Definition at line 187 of file fastmks.hpp.

References mlpack::fastmks::FastMKS< KernelType, TreeType >::metric.

22.40.3.4 `template<typename KernelType, typename TreeType = tree::CoverTree<metric::IPMetric<KernelType>, tree::FirstPointsRoot, FastMKSSStat>> void mlpack::fastmks::FastMKS< KernelType, TreeType >::Search (const size_t k, arma::Mat< size_t > & indices, arma::mat & products)`

Search for the maximum inner products of the query set (or if no query set was passed, the reference set is used).

The resulting maximum inner products are stored in the products matrix and the corresponding point indices are stores in the indices matrix. The results for each point in the query set are stored in the corresponding column of the indices

and products matrices; for instance, the index of the point with maximum inner product to point 4 in the query set will be stored in row 0 and column 4 of the indices matrix.

Parameters

<i>k</i>	The number of maximum kernels to find.
<i>indices</i>	Matrix to store resulting indices of max-kernel search in.
<i>products</i>	Matrix to store resulting max-kernel values in.

22.40.3.5 `template<typename KernelType, typename TreeType = tree::CoverTree<metric::IPMetric<KernelType>, tree::FirstPointIsRoot, FastMKSSStat>> std::string mlpack::fastmks::FastMKS< KernelType, TreeType >::ToString () const`

Returns a string representation of this object.

22.40.4 Member Data Documentation

22.40.4.1 `template<typename KernelType, typename TreeType = tree::CoverTree<metric::IPMetric<KernelType>, tree::FirstPointIsRoot, FastMKSSStat>> metric::IPMetric<KernelType> mlpack::fastmks::FastMKS< KernelType, TreeType >::metric [private]`

The instantiated inner-product metric induced by the given kernel.

Definition at line 215 of file fastmks.hpp.

Referenced by `mlpack::fastmks::FastMKS< KernelType, TreeType >::Metric()`.

22.40.4.2 `template<typename KernelType, typename TreeType = tree::CoverTree<metric::IPMetric<KernelType>, tree::FirstPointIsRoot, FastMKSSStat>> bool mlpack::fastmks::FastMKS< KernelType, TreeType >::naive [private]`

If true, naive (brute-force) search is used.

Definition at line 212 of file fastmks.hpp.

22.40.4.3 `template<typename KernelType, typename TreeType = tree::CoverTree<metric::IPMetric<KernelType>, tree::FirstPointIsRoot, FastMKSSStat>> const arma::mat& mlpack::fastmks::FastMKS< KernelType, TreeType >::querySet [private]`

The query dataset.

Definition at line 198 of file fastmks.hpp.

22.40.4.4 `template<typename KernelType, typename TreeType = tree::CoverTree<metric::IPMetric<KernelType>, tree::FirstPointIsRoot, FastMKSSStat>> TreeType* mlpack::fastmks::FastMKS< KernelType, TreeType >::queryTree [private]`

The tree built on the query dataset.

This is NULL if there is no query set.

Definition at line 204 of file fastmks.hpp.

```
22.40.4.5  template<typename KernelType, typename TreeType = tree::CoverTree<metric::IPMetric<KernelType>,
            tree::FirstPointIsRoot, FastMKSSStat>> const arma::mat& mlpack::fastmks::FastMKS< KernelType, TreeType
            >::referenceSet  [private]
```

The reference dataset.

Definition at line 196 of file fastmks.hpp.

```
22.40.4.6  template<typename KernelType, typename TreeType = tree::CoverTree<metric::IPMetric<KernelType>,
            tree::FirstPointIsRoot, FastMKSSStat>> TreeType* mlpack::fastmks::FastMKS< KernelType, TreeType
            >::referenceTree  [private]
```

The tree built on the reference dataset.

Definition at line 201 of file fastmks.hpp.

```
22.40.4.7  template<typename KernelType, typename TreeType = tree::CoverTree<metric::IPMetric<KernelType>,
            tree::FirstPointIsRoot, FastMKSSStat>> bool mlpack::fastmks::FastMKS< KernelType, TreeType >::single
            [private]
```

If true, single-tree search is used.

Definition at line 210 of file fastmks.hpp.

```
22.40.4.8  template<typename KernelType, typename TreeType = tree::CoverTree<metric::IPMetric<KernelType>,
            tree::FirstPointIsRoot, FastMKSSStat>> bool mlpack::fastmks::FastMKS< KernelType, TreeType >::treeOwner
            [private]
```

If true, this object created the trees and is responsible for them.

Definition at line 207 of file fastmks.hpp.

The documentation for this class was generated from the following file:

- src/mlpack/methods/fastmks/**fastmks.hpp**

22.41 mlpack::fastmks::FastMKSRules< KernelType, TreeType > Class Template Reference

The base case and pruning rules for **FastMKS** (p. 247) (fast max-kernel search).

Public Types

- typedef **neighbor::NeighborSearchTraversalInfo**< TreeType > **TraversalInfoType**

Public Member Functions

- **FastMKSRules** (const arma::mat &**referenceSet**, const arma::mat &**querySet**, arma::Mat< size_t > &**indices**, arma::mat &**products**, KernelType &**kernel**)
- double **BaseCase** (const size_t queryIndex, const size_t referenceIndex)
Compute the base case (kernel value) between two points.
- size_t **BaseCases** () const

- Get the number of times **BaseCase()** (p. 256) was called.
- **size_t & BaseCases ()**
 - Modify the number of times **BaseCase()** (p. 256) was called.
- **double Rescore** (const **size_t** queryIndex, **TreeType** &referenceNode, const double oldScore) const
 - Re-evaluate the score for recursion order.
- **double Rescore** (**TreeType** &queryNode, **TreeType** &referenceNode, const double oldScore) const
 - Re-evaluate the score for recursion order.
- **double Score** (const **size_t** queryIndex, **TreeType** &referenceNode)
 - Get the score for recursion order.
- **double Score** (**TreeType** &queryNode, **TreeType** &referenceNode)
 - Get the score for recursion order.
- **size_t Scores ()** const
 - Get the number of times **Score()** (p. 257) was called.
- **size_t & Scores ()**
 - Modify the number of times **Score()** (p. 257) was called.
- const **TraversallInfoType & TraversallInfo ()** const
- **TraversallInfoType & TraversallInfo ()**

Private Member Functions

- **double CalculateBound** (**TreeType** &queryNode) const
 - Calculate the bound for a given query node.
- **void InsertNeighbor** (const **size_t** queryIndex, const **size_t** pos, const **size_t** neighbor, const double distance)
 - Utility function to insert neighbor into list of results.

Private Attributes

- **size_t baseCases**
 - For benchmarking.
- **arma::Mat< size_t > & indices**
 - The indices of the maximum kernel results.
- **KernelType & kernel**
 - The instantiated kernel.
- **double lastKernel**
 - The last kernel evaluation resulting from **BaseCase()** (p. 256).
- **size_t lastQueryIndex**
 - The last query index **BaseCase()** (p. 256) was called with.
- **size_t lastReferenceIndex**
 - The last reference index **BaseCase()** (p. 256) was called with.
- **arma::mat & products**
 - The maximum kernels.
- **arma::vec queryKernels**
 - Cached query set self-kernels ($\| q \parallel$ for each q).
- const **arma::mat & querySet**
 - The query dataset.
- **arma::vec referenceKernels**
 - Cached reference set self-kernels ($\| r \parallel$ for each r).

- `const arma::mat & referenceSet`
The reference dataset.
- `size_t scores`
For benchmarking.
- **TraversallInfoType** `traversalInfo`

22.41.1 Detailed Description

`template<typename KernelType, typename TreeType> class mlpack::fastmks::FastMKSRules< KernelType, TreeType >`

The base case and pruning rules for **FastMKS** (p. 247) (fast max-kernel search).

Definition at line 29 of file `fastmks_rules.hpp`.

22.41.2 Member Typedef Documentation

22.41.2.1 `template<typename KernelType, typename TreeType > typedef neighbor::NeighborSearch↔
TraversallInfo<TreeType> mlpack::fastmks::FastMKSRules< KernelType, TreeType
>::TraversallInfoType`

Definition at line 101 of file `fastmks_rules.hpp`.

22.41.3 Constructor & Destructor Documentation

22.41.3.1 `template<typename KernelType, typename TreeType > mlpack::fastmks::FastMKSRules< KernelType, TreeType
>::FastMKSRules (const arma::mat & referenceSet, const arma::mat & querySet, arma::Mat< size_t > & indices,
arma::mat & products, KernelType & kernel)`

22.41.4 Member Function Documentation

22.41.4.1 `template<typename KernelType, typename TreeType > double mlpack::fastmks::FastMKSRules< KernelType,
TreeType >::BaseCase (const size_t queryIndex, const size_t referenceIndex)`

Compute the base case (kernel value) between two points.

22.41.4.2 `template<typename KernelType, typename TreeType > size_t mlpack::fastmks::FastMKSRules< KernelType,
TreeType >::BaseCases () const [inline]`

Get the number of times **BaseCase()** (p. 256) was called.

Definition at line 92 of file `fastmks_rules.hpp`.

References `mlpack::fastmks::FastMKSRules< KernelType, TreeType >::baseCases`.

22.41.4.3 `template<typename KernelType, typename TreeType > size_t & mlpack::fastmks::FastMKSRules< KernelType,
TreeType >::BaseCases () [inline]`

Modify the number of times **BaseCase()** (p. 256) was called.

Definition at line 94 of file `fastmks_rules.hpp`.

References `mlpack::fastmks::FastMKSRules< KernelType, TreeType >::baseCases`.

22.41.4.4 `template<typename KernelType , typename TreeType > double mlpack::fastmks::FastMKSRules< KernelType, TreeType >::CalculateBound (TreeType & queryNode) const [private]`

Calculate the bound for a given query node.

22.41.4.5 `template<typename KernelType , typename TreeType > void mlpack::fastmks::FastMKSRules< KernelType, TreeType >::InsertNeighbor (const size_t queryIndex, const size_t pos, const size_t neighbor, const double distance) [private]`

Utility function to insert neighbor into list of results.

22.41.4.6 `template<typename KernelType , typename TreeType > double mlpack::fastmks::FastMKSRules< KernelType, TreeType >::Rescore (const size_t queryIndex, TreeType & referenceNode, const double oldScore) const`

Re-evaluate the score for recursion order.

A low score indicates priority for recursion, while DBL_MAX indicates that a node should not be recursed into at all (it should be pruned). This is used when the score has already been calculated, but another recursion may have modified the bounds for pruning. So the old score is checked against the new pruning bound.

Parameters

<i>queryIndex</i>	Index of query point.
<i>referenceNode</i>	Candidate node to be recursed into.
<i>oldScore</i>	Old score produced by Score() (p. 257) (or Rescore() (p. 257)).

22.41.4.7 `template<typename KernelType , typename TreeType > double mlpack::fastmks::FastMKSRules< KernelType, TreeType >::Rescore (TreeType & queryNode, TreeType & referenceNode, const double oldScore) const`

Re-evaluate the score for recursion order.

A low score indicates priority for recursion, while DBL_MAX indicates that a node should not be recursed into at all (it should be pruned). This is used when the score has already been calculated, but another recursion may have modified the bounds for pruning. So the old score is checked against the new pruning bound.

Parameters

<i>queryNode</i>	Candidate query node to be recursed into.
<i>referenceNode</i>	Candidate reference node to be recursed into.
<i>oldScore</i>	Old score produced by Score() (p. 257) (or Rescore() (p. 257)).

22.41.4.8 `template<typename KernelType , typename TreeType > double mlpack::fastmks::FastMKSRules< KernelType, TreeType >::Score (const size_t queryIndex, TreeType & referenceNode)`

Get the score for recursion order.

A low score indicates priority for recursion, while DBL_MAX indicates that the node should not be recursed into at all (it should be pruned).

Parameters

<i>queryIndex</i>	Index of query point.
<i>referenceNode</i>	Candidate to be recursed into.

22.41.4.9 `template<typename KernelType , typename TreeType > double mlpack::fastmks::FastMKSRules< KernelType, TreeType >::Score (TreeType & queryNode, TreeType & referenceNode)`

Get the score for recursion order.

A low score indicates priority for recursion, while DBL_MAX indicates that the node should not be recursed into at all (it should be pruned).

Parameters

<i>queryNode</i>	Candidate query node to be recursed into.
<i>referenceNode</i>	Candidate reference node to be recursed into.

22.41.4.10 `template<typename KernelType , typename TreeType > size_t mlpack::fastmks::FastMKSRules< KernelType, TreeType >::Scores () const [inline]`

Get the number of times **Score()** (p. 257) was called.

Definition at line 97 of file fastmks_rules.hpp.

References mlpack::fastmks::FastMKSRules< KernelType, TreeType >::scores.

22.41.4.11 `template<typename KernelType , typename TreeType > size_t& mlpack::fastmks::FastMKSRules< KernelType, TreeType >::Scores () [inline]`

Modify the number of times **Score()** (p. 257) was called.

Definition at line 99 of file fastmks_rules.hpp.

References mlpack::fastmks::FastMKSRules< KernelType, TreeType >::scores.

22.41.4.12 `template<typename KernelType , typename TreeType > const TraversalInfoType& mlpack::fastmks::FastMKSRules< KernelType, TreeType >::TraversalInfo () const [inline]`

Definition at line 103 of file fastmks_rules.hpp.

References mlpack::fastmks::FastMKSRules< KernelType, TreeType >::traversalInfo.

22.41.4.13 `template<typename KernelType , typename TreeType > TraversalInfoType& mlpack::fastmks::FastMKSRules< KernelType, TreeType >::TraversalInfo () [inline]`

Definition at line 104 of file fastmks_rules.hpp.

References mlpack::fastmks::FastMKSRules< KernelType, TreeType >::traversalInfo.

22.41.5 Member Data Documentation

22.41.5.1 `template<typename KernelType , typename TreeType > size_t mlpack::fastmks::FastMKSRules< KernelType, TreeType >::baseCases [private]`

For benchmarking.

Definition at line 142 of file fastmks_rules.hpp.

Referenced by mlpack::fastmks::FastMKSRules< KernelType, TreeType >::BaseCases().

22.41.5.2 `template<typename KernelType , typename TreeType > arma::Mat<size_t>& mlpack::fastmks::FastMKSRules< KernelType, TreeType >::indices [private]`

The indices of the maximum kernel results.

Definition at line 113 of file fastmks_rules.hpp.

22.41.5.3 `template<typename KernelType , typename TreeType > KernelType& mlpack::fastmks::FastMKSRules< KernelType, TreeType >::kernel [private]`

The instantiated kernel.

Definition at line 123 of file fastmks_rules.hpp.

22.41.5.4 `template<typename KernelType , typename TreeType > double mlpack::fastmks::FastMKSRules< KernelType, TreeType >::lastKernel [private]`

The last kernel evaluation resulting from **BaseCase()** (p. 256).

Definition at line 130 of file fastmks_rules.hpp.

22.41.5.5 `template<typename KernelType , typename TreeType > size_t mlpack::fastmks::FastMKSRules< KernelType, TreeType >::lastQueryIndex [private]`

The last query index **BaseCase()** (p. 256) was called with.

Definition at line 126 of file fastmks_rules.hpp.

22.41.5.6 `template<typename KernelType , typename TreeType > size_t mlpack::fastmks::FastMKSRules< KernelType, TreeType >::lastReferenceIndex [private]`

The last reference index **BaseCase()** (p. 256) was called with.

Definition at line 128 of file fastmks_rules.hpp.

22.41.5.7 `template<typename KernelType , typename TreeType > arma::mat& mlpack::fastmks::FastMKSRules< KernelType, TreeType >::products [private]`

The maximum kernels.

Definition at line 115 of file fastmks_rules.hpp.

22.41.5.8 `template<typename KernelType , typename TreeType > arma::vec mlpack::fastmks::FastMKSRules< KernelType, TreeType >::queryKernels [private]`

Cached query set self-kernels (`|| q ||` for each `q`).

Definition at line 118 of file `fastmks_rules.hpp`.

22.41.5.9 `template<typename KernelType , typename TreeType > const arma::mat& mlpack::fastmks::FastMKSRules< KernelType, TreeType >::querySet [private]`

The query dataset.

Definition at line 110 of file `fastmks_rules.hpp`.

22.41.5.10 `template<typename KernelType , typename TreeType > arma::vec mlpack::fastmks::FastMKSRules< KernelType, TreeType >::referenceKernels [private]`

Cached reference set self-kernels (`|| r ||` for each `r`).

Definition at line 120 of file `fastmks_rules.hpp`.

22.41.5.11 `template<typename KernelType , typename TreeType > const arma::mat& mlpack::fastmks::FastMKSRules< KernelType, TreeType >::referenceSet [private]`

The reference dataset.

Definition at line 108 of file `fastmks_rules.hpp`.

22.41.5.12 `template<typename KernelType , typename TreeType > size_t mlpack::fastmks::FastMKSRules< KernelType, TreeType >::scores [private]`

For benchmarking.

Definition at line 144 of file `fastmks_rules.hpp`.

Referenced by `mlpack::fastmks::FastMKSRules< KernelType, TreeType >::Scores()`.

22.41.5.13 `template<typename KernelType , typename TreeType > TraversalInfoType mlpack::fastmks::FastMKSRules< KernelType, TreeType >::traversalInfo [private]`

Definition at line 146 of file `fastmks_rules.hpp`.

Referenced by `mlpack::fastmks::FastMKSRules< KernelType, TreeType >::TraversalInfo()`.

The documentation for this class was generated from the following file:

- `src/mlpack/methods/fastmks/fastmks_rules.hpp`

22.42 mlpack::fastmks::FastMKStat Class Reference

The statistic used in trees with **FastMKS** (p. 247).

Public Member Functions

- **FastMKStat** ()
Default initialization.
- template<typename TreeType >
FastMKStat (const TreeType &node)
Initialize this statistic for the given tree node.
- double **Bound** () const
Get the bound.
- double & **Bound** ()
Modify the bound.
- double **LastKernel** () const
Get the last kernel evaluation.
- double & **LastKernel** ()
Modify the last kernel evaluation.
- void * **LastKernelNode** () const
Get the address of the node corresponding to the last distance evaluation.
- void *& **LastKernelNode** ()
Modify the address of the node corresponding to the last distance evaluation.
- double **SelfKernel** () const
Get the self-kernel.
- double & **SelfKernel** ()
Modify the self-kernel.

Private Attributes

- double **bound**
The bound for pruning.
- double **lastKernel**
The last kernel evaluation.
- void * **lastKernelNode**
The node corresponding to the last kernel evaluation.
- double **selfKernel**
The self-kernel evaluation: $\sqrt{K(\text{centroid}, \text{centroid})}$.

22.42.1 Detailed Description

The statistic used in trees with **FastMKS** (p. 247).

This stores both the bound and the self-kernels for each node in the tree.

Definition at line 27 of file fastmks_stat.hpp.

22.42.2 Constructor & Destructor Documentation

22.42.2.1 mlpack::fastmks::FastMKStat::FastMKStat () [inline]

Default initialization.

Definition at line 33 of file fastmks_stat.hpp.

22.42.2.2 `template<typename TreeType > mlpack::fastmks::FastMKSSStat::FastMKSSStat (const TreeType & node) [inline]`

Initialize this statistic for the given tree node.

The TreeType's metric better be IPMetric with some kernel type (that is, Metric().Kernel() must exist).

Parameters

<i>node</i>	Node that this statistic is built for.
-------------	--

Definition at line 48 of file fastmks_stat.hpp.

References selfKernel.

22.42.3 Member Function Documentation

22.42.3.1 `double mlpack::fastmks::FastMKSSStat::Bound () const [inline]`

Get the bound.

Definition at line 88 of file fastmks_stat.hpp.

References bound.

22.42.3.2 `double& mlpack::fastmks::FastMKSSStat::Bound () [inline]`

Modify the bound.

Definition at line 90 of file fastmks_stat.hpp.

References bound.

22.42.3.3 `double mlpack::fastmks::FastMKSSStat::LastKernel () const [inline]`

Get the last kernel evaluation.

Definition at line 93 of file fastmks_stat.hpp.

References lastKernel.

22.42.3.4 `double& mlpack::fastmks::FastMKSSStat::LastKernel () [inline]`

Modify the last kernel evaluation.

Definition at line 95 of file fastmks_stat.hpp.

References lastKernel.

22.42.3.5 `void* mlpack::fastmks::FastMKSSStat::LastKernelNode () const [inline]`

Get the address of the node corresponding to the last distance evaluation.

Definition at line 98 of file fastmks_stat.hpp.

References lastKernelNode.

22.42.3.6 `void*& mpack::fastmks::FastMKStat::LastKernelNode () [inline]`

Modify the address of the node corresponding to the last distance evaluation.

Definition at line 101 of file fastmks_stat.hpp.

References lastKernelNode.

22.42.3.7 `double mpack::fastmks::FastMKStat::SelfKernel () const [inline]`

Get the self-kernel.

Definition at line 83 of file fastmks_stat.hpp.

References selfKernel.

22.42.3.8 `double& mpack::fastmks::FastMKStat::SelfKernel () [inline]`

Modify the self-kernel.

Definition at line 85 of file fastmks_stat.hpp.

References selfKernel.

22.42.4 Member Data Documentation

22.42.4.1 `double mpack::fastmks::FastMKStat::bound [private]`

The bound for pruning.

Definition at line 105 of file fastmks_stat.hpp.

Referenced by Bound().

22.42.4.2 `double mpack::fastmks::FastMKStat::lastKernel [private]`

The last kernel evaluation.

Definition at line 111 of file fastmks_stat.hpp.

Referenced by LastKernel().

22.42.4.3 `void* mpack::fastmks::FastMKStat::lastKernelNode [private]`

The node corresponding to the last kernel evaluation.

This has to be void otherwise we get recursive template arguments.

Definition at line 115 of file fastmks_stat.hpp.

Referenced by LastKernelNode().

22.42.4.4 `double mpack::fastmks::FastMKStat::selfKernel [private]`

The self-kernel evaluation: $\sqrt{K(\text{centroid}, \text{centroid})}$.

Definition at line 108 of file `fastmks_stat.hpp`.

Referenced by `FastMKSSStat()`, and `SelfKernel()`.

The documentation for this class was generated from the following file:

- `src/mlpack/methods/fastmks/fastmks_stat.hpp`

22.43 mlpack::gmm::DiagonalConstraint Class Reference

Force a covariance matrix to be diagonal.

Static Public Member Functions

- static void **ApplyConstraint** (arma::mat &covariance)
Force a covariance matrix to be diagonal.

22.43.1 Detailed Description

Force a covariance matrix to be diagonal.

Definition at line 25 of file `diagonal_constraint.hpp`.

22.43.2 Member Function Documentation

22.43.2.1 static void mlpack::gmm::DiagonalConstraint::ApplyConstraint (arma::mat & covariance) [inline],[static]

Force a covariance matrix to be diagonal.

Definition at line 29 of file `diagonal_constraint.hpp`.

The documentation for this class was generated from the following file:

- `src/mlpack/methods/gmm/diagonal_constraint.hpp`

22.44 mlpack::gmm::EigenvalueRatioConstraint Class Reference

Given a vector of eigenvalue ratios, ensure that the covariance matrix always has those eigenvalue ratios.

Public Member Functions

- **EigenvalueRatioConstraint** (const arma::vec &ratios)
*Create the **EigenvalueRatioConstraint** (p. 264) object with the given vector of eigenvalue ratios.*
- void **ApplyConstraint** (arma::mat &covariance) const
Apply the eigenvalue ratio constraint to the given covariance matrix.

Private Attributes

- `const arma::vec & ratios`

Ratios for eigenvalues.

22.44.1 Detailed Description

Given a vector of eigenvalue ratios, ensure that the covariance matrix always has those eigenvalue ratios.

When you create this object, make sure that the vector of ratios that you pass does not go out of scope, because this object holds a reference to that vector instead of copying it.

Definition at line 28 of file `eigenvalue_ratio_constraint.hpp`.

22.44.2 Constructor & Destructor Documentation

22.44.2.1 `mlpack::gmm::EigenvalueRatioConstraint::EigenvalueRatioConstraint (const arma::vec & ratios)` `[inline]`

Create the **EigenvalueRatioConstraint** (p. 264) object with the given vector of eigenvalue ratios.

These ratios are with respect to the first eigenvalue, which is the largest eigenvalue, so the first element of the vector should be 1. In addition, all other elements should be less than or equal to 1.

Definition at line 37 of file `eigenvalue_ratio_constraint.hpp`.

References `mlpack::Log::Fatal`, and `mlpack::Log::Warn`.

22.44.3 Member Function Documentation

22.44.3.1 `void mlpack::gmm::EigenvalueRatioConstraint::ApplyConstraint (arma::mat & covariance) const` `[inline]`

Apply the eigenvalue ratio constraint to the given covariance matrix.

Definition at line 62 of file `eigenvalue_ratio_constraint.hpp`.

References `ratios`.

22.44.4 Member Data Documentation

22.44.4.1 `const arma::vec& mlpack::gmm::EigenvalueRatioConstraint::ratios` `[private]`

Ratios for eigenvalues.

Definition at line 81 of file `eigenvalue_ratio_constraint.hpp`.

Referenced by `ApplyConstraint()`.

The documentation for this class was generated from the following file:

- `src/mlpack/methods/gmm/eigenvalue_ratio_constraint.hpp`

22.45 `mlpack::gmm::EMFit`< `InitialClusteringType`, `CovarianceConstraintPolicy` > Class Template Reference

This class contains methods which can fit a **GMM** (p. 271) to observations using the EM algorithm.

Public Member Functions

- **EMFit** (const size_t **maxiterations**=300, const double **tolerance**=1e-10, InitialClusteringType clusterer=InitialClusteringType(), CovarianceConstraintPolicy constraint=CovarianceConstraintPolicy())
*Construct the **EMFit** (p. 266) object, optionally passing an InitialClusteringType object (just in case it needs to store state).*
- const InitialClusteringType & **Clusterer** () const
Get the clusterer.
- InitialClusteringType & **Clusterer** ()
Modify the clusterer.
- const CovarianceConstraintPolicy & **Constraint** () const
Get the covariance constraint policy class.
- CovarianceConstraintPolicy & **Constraint** ()
Modify the covariance constraint policy class.
- void **Estimate** (const arma::mat &observations, std::vector< arma::vec > &means, std::vector< arma::mat > &covariances, arma::vec &weights, const bool useInitialModel=false)
*Fit the observations to a Gaussian mixture model (**GMM** (p. 271)) using the EM algorithm.*
- void **Estimate** (const arma::mat &observations, const arma::vec &probabilities, std::vector< arma::vec > &means, std::vector< arma::mat > &covariances, arma::vec &weights, const bool useInitialModel=false)
*Fit the observations to a Gaussian mixture model (**GMM** (p. 271)) using the EM algorithm, taking into account the probabilities of each point being from this mixture.*
- size_t **MaxIterations** () const
Get the maximum number of iterations of the EM algorithm.
- size_t & **MaxIterations** ()
Modify the maximum number of iterations of the EM algorithm.
- double **Tolerance** () const
Get the tolerance for the convergence of the EM algorithm.
- double & **Tolerance** ()
Modify the tolerance for the convergence of the EM algorithm.

Private Member Functions

- void **InitialClustering** (const arma::mat &observations, std::vector< arma::vec > &means, std::vector< arma::mat > &covariances, arma::vec &weights)
Run the clusterer, and then turn the cluster assignments into Gaussians.
- double **LogLikelihood** (const arma::mat &data, const std::vector< arma::vec > &means, const std::vector< arma::mat > &covariances, const arma::vec &weights) const
Calculate the log-likelihood of a model.

Private Attributes

- InitialClusteringType **clusterer**
Object which will perform the clustering.
- CovarianceConstraintPolicy **constraint**
Object which applies constraints to the covariance matrix.
- size_t **maxIterations**
Maximum iterations of EM algorithm.
- double **tolerance**
Tolerance for convergence of EM.

22.45.1 Detailed Description

```
template<typename InitialClusteringType = kmeans::KMeans<>, typename CovarianceConstraintPolicy = PositiveDefiniteConstraint>
class mlpack::gmm::EMFit< InitialClusteringType, CovarianceConstraintPolicy >
```

This class contains methods which can fit a **GMM** (p. 271) to observations using the EM algorithm.

It requires an initial clustering mechanism, which is by default the KMeans algorithm. The clustering mechanism must implement the following method:

- void Cluster(const arma::mat& observations, const size_t clusters, arma::Col<size_t>& assignments);

This method should create 'clusters' clusters, and return the assignment of each point to a cluster.

Definition at line 43 of file em_fit.hpp.

22.45.2 Constructor & Destructor Documentation

```
22.45.2.1 template<typename InitialClusteringType = kmeans::KMeans<>, typename CovarianceConstraintPolicy =
PositiveDefiniteConstraint> mlpack::gmm::EMFit< InitialClusteringType, CovarianceConstraintPolicy
>::EMFit ( const size_t maxIterations = 300, const double tolerance = 1e-10, InitialClusteringType
clusterer = InitialClusteringType(), CovarianceConstraintPolicy constraint =
CovarianceConstraintPolicy() )
```

Construct the **EMFit** (p. 266) object, optionally passing an InitialClusteringType object (just in case it needs to store state).

Setting the maximum number of iterations to 0 means that the EM algorithm will iterate until convergence (with the given tolerance).

The parameter forcePositive controls whether or not the covariance matrices are checked for positive definiteness at each iteration. This could be a time-consuming task, so, if you know your data is well-behaved, you can set it to false and save some runtime.

Parameters

<i>maxIterations</i>	Maximum number of iterations for EM.
<i>tolerance</i>	Log-likelihood tolerance required for convergence.

<i>forcePositive</i>	Check for positive-definiteness of each covariance matrix at each iteration.
<i>clusterer</i>	Object which will perform the initial clustering.

22.45.3 Member Function Documentation

22.45.3.1 `template<typename InitialClusteringType = kmeans::KMeans<>, typename CovarianceConstraintPolicy = PositiveDefiniteConstraint> const InitialClusteringType& mlpack::gmm::EMFit< InitialClusteringType, CovarianceConstraintPolicy >::Clusterer () const [inline]`

Get the clusterer.

Definition at line 114 of file `em_fit.hpp`.

References `mlpack::gmm::EMFit< InitialClusteringType, CovarianceConstraintPolicy >::clusterer`.

22.45.3.2 `template<typename InitialClusteringType = kmeans::KMeans<>, typename CovarianceConstraintPolicy = PositiveDefiniteConstraint> InitialClusteringType& mlpack::gmm::EMFit< InitialClusteringType, CovarianceConstraintPolicy >::Clusterer () [inline]`

Modify the clusterer.

Definition at line 116 of file `em_fit.hpp`.

References `mlpack::gmm::EMFit< InitialClusteringType, CovarianceConstraintPolicy >::clusterer`.

22.45.3.3 `template<typename InitialClusteringType = kmeans::KMeans<>, typename CovarianceConstraintPolicy = PositiveDefiniteConstraint> const CovarianceConstraintPolicy& mlpack::gmm::EMFit< InitialClusteringType, CovarianceConstraintPolicy >::Constraint () const [inline]`

Get the covariance constraint policy class.

Definition at line 119 of file `em_fit.hpp`.

References `mlpack::gmm::EMFit< InitialClusteringType, CovarianceConstraintPolicy >::constraint`.

22.45.3.4 `template<typename InitialClusteringType = kmeans::KMeans<>, typename CovarianceConstraintPolicy = PositiveDefiniteConstraint> CovarianceConstraintPolicy& mlpack::gmm::EMFit< InitialClusteringType, CovarianceConstraintPolicy >::Constraint () [inline]`

Modify the covariance constraint policy class.

Definition at line 121 of file `em_fit.hpp`.

References `mlpack::gmm::EMFit< InitialClusteringType, CovarianceConstraintPolicy >::constraint`.

22.45.3.5 `template<typename InitialClusteringType = kmeans::KMeans<>, typename CovarianceConstraintPolicy = PositiveDefiniteConstraint> void mlpack::gmm::EMFit< InitialClusteringType, CovarianceConstraintPolicy >::Estimate (const arma::mat & observations, std::vector< arma::vec > & means, std::vector< arma::mat > & covariances, arma::vec & weights, const bool useInitialModel = false)`

Fit the observations to a Gaussian mixture model (**GMM** (p. 271)) using the EM algorithm.

The size of the vectors (indicating the number of components) must already be set. Optionally, if `useInitialModel` is set to true, then the model given in the means, covariances, and weights parameters is used as the initial model, instead of using the `InitialClusteringType::Cluster()` option.

22.45 mlpack::gmm::EMFit< InitialClusteringType, CovarianceConstraintPolicy > Class Template Reference#69

Parameters

<i>observations</i>	List of observations to train on.
<i>means</i>	Vector to store trained means in.
<i>covariances</i>	Vector to store trained covariances in.
<i>weights</i>	Vector to store a priori weights in.
<i>useInitialModel</i>	If true, the given model is used for the initial clustering.

```
22.45.3.6  template<typename InitialClusteringType = kmeans::KMeans<>, typename CovarianceConstraintPolicy =
PositiveDefiniteConstraint> void mlpack::gmm::EMFit< InitialClusteringType, CovarianceConstraintPolicy
>::Estimate ( const arma::mat & observations, const arma::vec & probabilities, std::vector< arma::vec > & means,
std::vector< arma::mat > & covariances, arma::vec & weights, const bool useInitialModel = false )
```

Fit the observations to a Gaussian mixture model (**GMM** (p.271)) using the EM algorithm, taking into account the probabilities of each point being from this mixture.

The size of the vectors (indicating the number of components) must already be set. Optionally, if *useInitialModel* is set to true, then the model given in the means, covariances, and weights parameters is used as the initial model, instead of using the *InitialClusteringType::Cluster()* option.

Parameters

<i>observations</i>	List of observations to train on.
<i>probabilities</i>	Probability of each point being from this model.
<i>means</i>	Vector to store trained means in.
<i>covariances</i>	Vector to store trained covariances in.
<i>weights</i>	Vector to store a priori weights in.
<i>useInitialModel</i>	If true, the given model is used for the initial clustering.

```
22.45.3.7  template<typename InitialClusteringType = kmeans::KMeans<>, typename CovarianceConstraintPolicy =
PositiveDefiniteConstraint> void mlpack::gmm::EMFit< InitialClusteringType, CovarianceConstraintPolicy
>::InitialClustering ( const arma::mat & observations, std::vector< arma::vec > & means, std::vector< arma::mat > &
covariances, arma::vec & weights ) [private]
```

Run the clusterer, and then turn the cluster assignments into Gaussians.

This is a helper function for both overloads of **Estimate()** (p.268). The vectors must be already set to the number of clusters.

Parameters

<i>observations</i>	List of observations.
<i>means</i>	Vector to store means in.
<i>covariances</i>	Vector to store covariances in.
<i>weights</i>	Vector to store a priori weights in.

```
22.45.3.8  template<typename InitialClusteringType = kmeans::KMeans<>, typename CovarianceConstraintPolicy =
PositiveDefiniteConstraint> double mlpack::gmm::EMFit< InitialClusteringType, CovarianceConstraintPolicy
>::LogLikelihood ( const arma::mat & data, const std::vector< arma::vec > & means, const std::vector< arma::mat >
& covariances, const arma::vec & weights ) const [private]
```

Calculate the log-likelihood of a model.

Yes, this is reimplemented in the **GMM** (p.271) code. Intuition suggests that the log-likelihood is not the best way to determine if the EM algorithm has converged.

Parameters

<i>data</i>	Data matrix.
<i>means</i>	Vector of means.
<i>covariances</i>	Vector of covariance matrices.
<i>weights</i>	Vector of a priori weights.

```
22.45.3.9  template<typename InitialClusteringType = kmeans::KMeans<>, typename CovarianceConstraintPolicy =
           PositiveDefiniteConstraint> size_t mlpack::gmm::EMFit< InitialClusteringType, CovarianceConstraintPolicy
           >::MaxIterations ( ) const [inline]
```

Get the maximum number of iterations of the EM algorithm.

Definition at line 124 of file em_fit.hpp.

References mlpack::gmm::EMFit< InitialClusteringType, CovarianceConstraintPolicy >::maxIterations.

```
22.45.3.10 template<typename InitialClusteringType = kmeans::KMeans<>, typename CovarianceConstraintPolicy =
           PositiveDefiniteConstraint> size_t& mlpack::gmm::EMFit< InitialClusteringType, CovarianceConstraintPolicy
           >::MaxIterations ( ) [inline]
```

Modify the maximum number of iterations of the EM algorithm.

Definition at line 126 of file em_fit.hpp.

References mlpack::gmm::EMFit< InitialClusteringType, CovarianceConstraintPolicy >::maxIterations.

```
22.45.3.11 template<typename InitialClusteringType = kmeans::KMeans<>, typename CovarianceConstraintPolicy =
           PositiveDefiniteConstraint> double mlpack::gmm::EMFit< InitialClusteringType, CovarianceConstraintPolicy
           >::Tolerance ( ) const [inline]
```

Get the tolerance for the convergence of the EM algorithm.

Definition at line 129 of file em_fit.hpp.

References mlpack::gmm::EMFit< InitialClusteringType, CovarianceConstraintPolicy >::tolerance.

```
22.45.3.12 template<typename InitialClusteringType = kmeans::KMeans<>, typename CovarianceConstraintPolicy =
           PositiveDefiniteConstraint> double& mlpack::gmm::EMFit< InitialClusteringType, CovarianceConstraintPolicy
           >::Tolerance ( ) [inline]
```

Modify the tolerance for the convergence of the EM algorithm.

Definition at line 131 of file em_fit.hpp.

References mlpack::gmm::EMFit< InitialClusteringType, CovarianceConstraintPolicy >::tolerance.

22.45.4 Member Data Documentation

22.45.4.1 `template<typename InitialClusteringType = kmeans::KMeans<>, typename CovarianceConstraintPolicy = PositiveDefiniteConstraint> InitialClusteringType mlpack::gmm::EMFit< InitialClusteringType, CovarianceConstraintPolicy >::clusterer [private]`

Object which will perform the clustering.

Definition at line 169 of file `em_fit.hpp`.

Referenced by `mlpack::gmm::EMFit< InitialClusteringType, CovarianceConstraintPolicy >::Clusterer()`.

22.45.4.2 `template<typename InitialClusteringType = kmeans::KMeans<>, typename CovarianceConstraintPolicy = PositiveDefiniteConstraint> CovarianceConstraintPolicy mlpack::gmm::EMFit< InitialClusteringType, CovarianceConstraintPolicy >::constraint [private]`

Object which applies constraints to the covariance matrix.

Definition at line 171 of file `em_fit.hpp`.

Referenced by `mlpack::gmm::EMFit< InitialClusteringType, CovarianceConstraintPolicy >::Constraint()`.

22.45.4.3 `template<typename InitialClusteringType = kmeans::KMeans<>, typename CovarianceConstraintPolicy = PositiveDefiniteConstraint> size_t mlpack::gmm::EMFit< InitialClusteringType, CovarianceConstraintPolicy >::maxIterations [private]`

Maximum iterations of EM algorithm.

Definition at line 165 of file `em_fit.hpp`.

Referenced by `mlpack::gmm::EMFit< InitialClusteringType, CovarianceConstraintPolicy >::MaxIterations()`.

22.45.4.4 `template<typename InitialClusteringType = kmeans::KMeans<>, typename CovarianceConstraintPolicy = PositiveDefiniteConstraint> double mlpack::gmm::EMFit< InitialClusteringType, CovarianceConstraintPolicy >::tolerance [private]`

Tolerance for convergence of EM.

Definition at line 167 of file `em_fit.hpp`.

Referenced by `mlpack::gmm::EMFit< InitialClusteringType, CovarianceConstraintPolicy >::Tolerance()`.

The documentation for this class was generated from the following file:

- `src/mlpack/methods/gmm/em_fit.hpp`

22.46 mlpack::gmm::GMM< FittingType > Class Template Reference

A Gaussian Mixture Model (**GMM** (p. 271)).

Public Member Functions

- **GMM** ()
Create an empty Gaussian Mixture Model, with zero gaussians.
- **GMM** (const size_t **gaussians**, const size_t **dimensionality**)

- Create a **GMM** (p. 271) with the given number of Gaussians, each of which have the specified dimensionality.

 - **GMM** (const size_t **gaussians**, const size_t **dimensionality**, FittingType &fitter)

Create a **GMM** (p. 271) with the given number of Gaussians, each of which have the specified dimensionality.

 - **GMM** (const std::vector< arma::vec > &**means**, const std::vector< arma::mat > &**covariances**, const arma::vec &**weights**)

Create a **GMM** (p. 271) with the given means, covariances, and weights.

 - **GMM** (const std::vector< arma::vec > &**means**, const std::vector< arma::mat > &**covariances**, const arma::vec &**weights**, FittingType &fitter)

Create a **GMM** (p. 271) with the given means, covariances, and weights, and use the given initialized FittingType class.

 - template<typename OtherFittingType >
GMM (const **GMM**< OtherFittingType > &other)

Copy constructor for GMMs which use different fitting types.

 - **GMM** (const **GMM** &other)

Copy constructor for GMMs using the same fitting type.

 - void **Classify** (const arma::mat &observations, arma::Col< size_t > &labels) const

Classify the given observations as being from an individual component in this **GMM** (p. 271).

 - const std::vector< arma::mat > &**Covariances** () const

Return a const reference to the vector of covariance matrices (sigma).

 - std::vector< arma::mat > &**Covariances** ()

Return a reference to the vector of covariance matrices (sigma).

 - size_t **Dimensionality** () const

Return the dimensionality of the model.

 - size_t & **Dimensionality** ()

Modify the dimensionality of the model.

 - double **Estimate** (const arma::mat &observations, const size_t trials=1, const bool useExistingModel=false)

Estimate the probability distribution directly from the given observations, using the given algorithm in the FittingType class to fit the data.

 - double **Estimate** (const arma::mat &observations, const arma::vec &probabilities, const size_t trials=1, const bool useExistingModel=false)

Estimate the probability distribution directly from the given observations, taking into account the probability of each observation actually being from this distribution, and using the given algorithm in the FittingType class to fit the data.

 - const FittingType & **Fitter** () const

Return a const reference to the fitting type.

 - FittingType & **Fitter** ()

Return a reference to the fitting type.

 - size_t **Gaussians** () const

Return the number of gaussians in the model.

 - size_t & **Gaussians** ()

Modify the number of gaussians in the model.

 - void **Load** (const std::string &filename)

Load a **GMM** (p. 271) from an XML file.

 - const std::vector< arma::vec > & **Means** () const

Return a const reference to the vector of means (mu).

 - std::vector< arma::vec > & **Means** ()

Return a reference to the vector of means (mu).

 - template<typename OtherFittingType >
GMM & **operator=** (const **GMM**< OtherFittingType > &other)

Copy operator for GMMs which use different fitting types.

- **GMM & operator=** (const **GMM** &other)
Copy operator for GMMs which use the same fitting type.
- double **Probability** (const arma::vec &observation) const
Return the probability that the given observation came from this distribution.
- double **Probability** (const arma::vec &observation, const size_t component) const
Return the probability that the given observation came from the given Gaussian component in this distribution.
- arma::vec **Random** () const
Return a randomly generated observation according to the probability distribution defined by this object.
- void **Save** (const std::string &filename) const
*Save a **GMM** (p. 271) to an XML file.*
- std::string **ToString** () const
Returns a string representation of this object.
- const arma::vec & **Weights** () const
Return a const reference to the a priori weights of each Gaussian.
- arma::vec & **Weights** ()
Return a reference to the a priori weights of each Gaussian.

Private Member Functions

- double **LogLikelihood** (const arma::mat &dataPoints, const std::vector< arma::vec > &means, const std::vector< arma::mat > &covars, const arma::vec &weights) const
This function computes the loglikelihood of the given model.

Private Attributes

- std::vector< arma::mat > **covariances**
Vector of covariances; one for each Gaussian.
- size_t **dimensionality**
The dimensionality of the model.
- FittingType & **fitter**
Reference to the fitting object we should use.
- size_t **gaussians**
The number of Gaussians in the model.
- FittingType **localFitter**
Locally-stored fitting object; in case the user did not pass one.
- std::vector< arma::vec > **means**
Vector of means; one for each Gaussian.
- arma::vec **weights**
Vector of a priori weights for each Gaussian.

22.46.1 Detailed Description

```
template<typename FittingType = EMFit<>> class mlpack::gmm::GMM< FittingType >
```

A Gaussian Mixture Model (**GMM** (p. 271)).

This class uses maximum likelihood loss functions to estimate the parameters of the **GMM** (p. 271) on a given dataset via the given fitting mechanism, defined by the FittingType template parameter. The **GMM** (p. 271) can be trained using normal data, or data with probabilities of being from this **GMM** (p. 271) (see **GMM::Estimate()** (p. 277) for more information).

The FittingType template class must provide a way for the **GMM** (p. 271) to train on data. It must provide the following two functions:

```
void Estimate(const arma::mat& observations,
              std::vector<arma::vec>& means,
              std::vector<arma::mat>& covariances,
              arma::vec& weights);

void Estimate(const arma::mat& observations,
              const arma::vec& probabilities,
              std::vector<arma::vec>& means,
              std::vector<arma::mat>& covariances,
              arma::vec& weights);
```

These functions should produce a trained **GMM** (p. 271) from the given observations and probabilities. These may modify the size of the model (by increasing the size of the mean and covariance vectors as well as the weight vectors), but the method should expect that these vectors are already set to the size of the **GMM** (p. 271) as specified in the constructor.

For a sample implementation, see the **EMFit** (p. 266) class; this class uses the EM algorithm to train a **GMM** (p. 271), and is the default fitting type.

The **GMM** (p. 271), once trained, can be used to generate random points from the distribution and estimate the probability of points being from the distribution. The parameters of the **GMM** (p. 271) can be obtained through the accessors and mutators.

Example use:

```
// Set up a mixture of 5 gaussians in a 4-dimensional space (uses the default
// EM fitting mechanism).
GMM<> g(5, 4);

// Train the GMM given the data observations.
g.Estimate(data);

// Get the probability of 'observation' being observed from this GMM.
double probability = g.Probability(observation);

// Get a random observation from the GMM.
arma::vec observation = g.Random();
```

Definition at line 81 of file gmm.hpp.

22.46.2 Constructor & Destructor Documentation

22.46.2.1 `template<typename FittingType = EMFit<>> mlpack::gmm::GMM< FittingType >::GMM () [inline]`

Create an empty Gaussian Mixture Model, with zero gaussians.

Definition at line 99 of file gmm.hpp.

References `mlpack::Log::Debug`.

22.46.2.2 `template<typename FittingType = EMFit<>> mlpack::gmm::GMM< FittingType >::GMM (const size_t gaussians, const size_t dimensionality)`

Create a **GMM** (p. 271) with the given number of Gaussians, each of which have the specified dimensionality.

The means and covariances will be set to 0.

Parameters

<i>gaussians</i>	Number of Gaussians in this GMM (p. 271).
<i>dimensionality</i>	Dimensionality of each Gaussian.

22.46.2.3 `template<typename FittingType = EMFit<>> mlpack::gmm::GMM< FittingType >::GMM (const size_t gaussians, const size_t dimensionality, FittingType & fitter)`

Create a **GMM** (p. 271) with the given number of Gaussians, each of which have the specified dimensionality.

Also, pass in an initialized FittingType class; this is useful in cases where the FittingType class needs to store some state.

Parameters

<i>gaussians</i>	Number of Gaussians in this GMM (p. 271).
<i>dimensionality</i>	Dimensionality of each Gaussian.
<i>fitter</i>	Initialized fitting mechanism.

22.46.2.4 `template<typename FittingType = EMFit<>> mlpack::gmm::GMM< FittingType >::GMM (const std::vector< arma::vec > & means, const std::vector< arma::mat > & covariances, const arma::vec & weights) [inline]`

Create a **GMM** (p. 271) with the given means, covariances, and weights.

Parameters

<i>means</i>	Means of the model.
<i>covariances</i>	Covariances of the model.
<i>weights</i>	Weights of the model.

Definition at line 142 of file gmm.hpp.

22.46.2.5 `template<typename FittingType = EMFit<>> mlpack::gmm::GMM< FittingType >::GMM (const std::vector< arma::vec > & means, const std::vector< arma::mat > & covariances, const arma::vec & weights, FittingType & fitter) [inline]`

Create a **GMM** (p. 271) with the given means, covariances, and weights, and use the given initialized FittingType class.

This is useful in cases where the FittingType class needs to store some state.

Parameters

<i>means</i>	Means of the model.
<i>covariances</i>	Covariances of the model.

<i>weights</i>	Weights of the model.
----------------	-----------------------

Definition at line 162 of file gmm.hpp.

22.46.2.6 `template<typename FittingType = EMFit<>> template<typename OtherFittingType > mlpack::gmm::GMM< FittingType >::GMM (const GMM< OtherFittingType > & other)`

Copy constructor for GMMs which use different fitting types.

22.46.2.7 `template<typename FittingType = EMFit<>> mlpack::gmm::GMM< FittingType >::GMM (const GMM< FittingType > & other)`

Copy constructor for GMMs using the same fitting type.

This also copies the fitter.

22.46.3 Member Function Documentation

22.46.3.1 `template<typename FittingType = EMFit<>> void mlpack::gmm::GMM< FittingType >::Classify (const arma::mat & observations, arma::Col< size_t > & labels) const`

Classify the given observations as being from an individual component in this **GMM** (p. 271).

The resultant classifications are stored in the 'labels' object, and each label will be between 0 and (**Gaussians()** (p. 279) - 1). Supposing that a point was classified with label 2, and that our **GMM** (p. 271) object was called 'gmm', one could access the relevant Gaussian distribution as follows:

```
arma::vec mean = gmm.Means()[2];
arma::mat covariance = gmm.Covariances()[2];
double priorWeight = gmm.Weights()[2];
```

Parameters

<i>observations</i>	List of observations to classify.
<i>labels</i>	Object which will be filled with labels.

22.46.3.2 `template<typename FittingType = EMFit<>> const std::vector<arma::mat>& mlpack::gmm::GMM< FittingType >::Covariances () const [inline]`

Return a const reference to the vector of covariance matrices (sigma).

Definition at line 230 of file gmm.hpp.

References `mlpack::gmm::GMM< FittingType >::covariances`.

22.46.3.3 `template<typename FittingType = EMFit<>> std::vector<arma::mat>& mlpack::gmm::GMM< FittingType >::Covariances () [inline]`

Return a reference to the vector of covariance matrices (sigma).

Definition at line 232 of file gmm.hpp.

References `mlpack::gmm::GMM< FittingType >::covariances`.

22.46.3.4 `template<typename FittingType = EMFit<>> size_t mlpack::gmm::GMM< FittingType >::Dimensionality () const [inline]`

Return the dimensionality of the model.

Definition at line 219 of file gmm.hpp.

References mlpack::gmm::GMM< FittingType >::dimensionality.

22.46.3.5 `template<typename FittingType = EMFit<>> size_t& mlpack::gmm::GMM< FittingType >::Dimensionality () [inline]`

Modify the dimensionality of the model.

Careful! You will have to update each mean and covariance matrix yourself.

Definition at line 222 of file gmm.hpp.

References mlpack::gmm::GMM< FittingType >::dimensionality.

22.46.3.6 `template<typename FittingType = EMFit<>> double mlpack::gmm::GMM< FittingType >::Estimate (const arma::mat & observations, const size_t trials = 1, const bool useExistingModel = false)`

Estimate the probability distribution directly from the given observations, using the given algorithm in the FittingType class to fit the data.

The fitting will be performed 'trials' times; from these trials, the model with the greatest log-likelihood will be selected. By default, only one trial is performed. The log-likelihood of the best fitting is returned.

Optionally, the existing model can be used as an initial model for the estimation by setting 'useExistingModel' to true. If the fitting procedure is deterministic after the initial position is given, then 'trials' should be set to 1.

Template Parameters

<i>FittingType</i>	The type of fitting method which should be used (EMFit<> is suggested).
--------------------	---

Parameters

<i>observations</i>	Observations of the model.
<i>trials</i>	Number of trials to perform; the model in these trials with the greatest log-likelihood will be selected.
<i>useExistingModel</i>	If true, the existing model is used as an initial model for the estimation.

Returns

The log-likelihood of the best fit.

22.46.3.7 `template<typename FittingType = EMFit<>> double mlpack::gmm::GMM< FittingType >::Estimate (const arma::mat & observations, const arma::vec & probabilities, const size_t trials = 1, const bool useExistingModel = false)`

Estimate the probability distribution directly from the given observations, taking into account the probability of each observation actually being from this distribution, and using the given algorithm in the FittingType class to fit the data.

The fitting will be performed 'trials' times; from these trials, the model with the greatest log-likelihood will be selected. By default, only one trial is performed. The log-likelihood of the best fitting is returned.

Optionally, the existing model can be used as an initial model for the estimation by setting 'useExistingModel' to true. If the fitting procedure is deterministic after the initial position is given, then 'trials' should be set to 1.

Parameters

<i>observations</i>	Observations of the model.
<i>probabilities</i>	Probability of each observation being from this distribution.
<i>trials</i>	Number of trials to perform; the model in these trials with the greatest log-likelihood will be selected.
<i>useExistingModel</i>	If true, the existing model is used as an initial model for the estimation.

Returns

The log-likelihood of the best fit.

22.46.3.8 `template<typename FittingType = EMFit<>> const FittingType& mlpack::gmm::GMM< FittingType >::Fitter ()
const [inline]`

Return a const reference to the fitting type.

Definition at line 240 of file gmm.hpp.

References mlpack::gmm::GMM< FittingType >::fitter.

22.46.3.9 `template<typename FittingType = EMFit<>> FittingType& mlpack::gmm::GMM< FittingType >::Fitter ()
[inline]`

Return a reference to the fitting type.

Definition at line 242 of file gmm.hpp.

References mlpack::gmm::GMM< FittingType >::fitter.

22.46.3.10 `template<typename FittingType = EMFit<>> size_t mlpack::gmm::GMM< FittingType >::Gaussians () const
[inline]`

Return the number of gaussians in the model.

Definition at line 213 of file gmm.hpp.

References mlpack::gmm::GMM< FittingType >::gaussians.

22.46.3.11 `template<typename FittingType = EMFit<>> size_t& mlpack::gmm::GMM< FittingType >::Gaussians ()
[inline]`

Modify the number of gaussians in the model.

Careful! You will have to resize the means, covariances, and weights yourself.

Definition at line 216 of file gmm.hpp.

References mlpack::gmm::GMM< FittingType >::gaussians.

22.46.3.12 `template<typename FittingType = EMFit<>> void mlpack::gmm::GMM< FittingType >::Load (const std::string &
filename)`

Load a **GMM** (p. 271) from an XML file.

The format of the XML file should be the same as is generated by the **Save()** (p. 281) method.

Parameters

<i>filename</i>	Name of XML file containing model to be loaded.
-----------------	---

22.46.3.13 `template<typename FittingType = EMFit<>> double mlpack::gmm::GMM< FittingType >::LogLikelihood (const arma::mat & dataPoints, const std::vector< arma::vec > & means, const std::vector< arma::mat > & covars, const arma::vec & weights) const [private]`

This function computes the loglikelihood of the given model.

This function is used by **GMM::Estimate()** (p. 277).

Parameters

<i>dataPoints</i>	Observations to calculate the likelihood for.
<i>means</i>	Means of the given mixture model.
<i>covars</i>	Covariances of the given mixture model.
<i>weights</i>	Weights of the given mixture model.

22.46.3.14 `template<typename FittingType = EMFit<>> const std::vector<arma::vec>& mlpack::gmm::GMM< FittingType >::Means () const [inline]`

Return a const reference to the vector of means (μ).

Definition at line 225 of file gmm.hpp.

References mlpack::gmm::GMM< FittingType >::means.

22.46.3.15 `template<typename FittingType = EMFit<>> std::vector<arma::vec>& mlpack::gmm::GMM< FittingType >::Means () [inline]`

Return a reference to the vector of means (μ).

Definition at line 227 of file gmm.hpp.

References mlpack::gmm::GMM< FittingType >::means.

22.46.3.16 `template<typename FittingType = EMFit<>> template<typename OtherFittingType > GMM& mlpack::gmm::GMM< FittingType >::operator= (const GMM< OtherFittingType > & other)`

Copy operator for GMMs which use different fitting types.

22.46.3.17 `template<typename FittingType = EMFit<>> GMM& mlpack::gmm::GMM< FittingType >::operator= (const GMM< FittingType > & other)`

Copy operator for GMMs which use the same fitting type.

This also copies the fitter.

22.46.3.18 `template<typename FittingType = EMFit<>> double mlpack::gmm::GMM< FittingType >::Probability (const arma::vec & observation) const`

Return the probability that the given observation came from this distribution.

Parameters

<i>observation</i>	Observation to evaluate the probability of.
--------------------	---

22.46.3.19 `template<typename FittingType = EMFit<>> double mlpack::gmm::GMM< FittingType >::Probability (const arma::vec & observation, const size_t component) const`

Return the probability that the given observation came from the given Gaussian component in this distribution.

Parameters

<i>observation</i>	Observation to evaluate the probability of.
<i>component</i>	Index of the component of the GMM (p. 271) to be considered.

22.46.3.20 `template<typename FittingType = EMFit<>> arma::vec mlpack::gmm::GMM< FittingType >::Random () const`

Return a randomly generated observation according to the probability distribution defined by this object.

Returns

Random observation from this **GMM** (p. 271).

22.46.3.21 `template<typename FittingType = EMFit<>> void mlpack::gmm::GMM< FittingType >::Save (const std::string & filename) const`

Save a **GMM** (p. 271) to an XML file.

Parameters

<i>filename</i>	Name of XML file to write to.
-----------------	-------------------------------

22.46.3.22 `template<typename FittingType = EMFit<>> std::string mlpack::gmm::GMM< FittingType >::ToString () const`

Returns a string representation of this object.

22.46.3.23 `template<typename FittingType = EMFit<>> const arma::vec& mlpack::gmm::GMM< FittingType >::Weights () const [inline]`

Return a const reference to the a priori weights of each Gaussian.

Definition at line 235 of file gmm.hpp.

References mlpack::gmm::GMM< FittingType >::weights.

22.46.3.24 `template<typename FittingType = EMFit<>> arma::vec& mlpack::gmm::GMM< FittingType >::Weights () [inline]`

Return a reference to the a priori weights of each Gaussian.

Definition at line 237 of file gmm.hpp.

References mlpack::gmm::GMM< FittingType >::weights.

22.46.4 Member Data Documentation

22.46.4.1 `template<typename FittingType = EMFit<>> std::vector<arma::mat> mlpack::gmm::GMM< FittingType >::covariances [private]`

Vector of covariances; one for each Gaussian.

Definition at line 91 of file gmm.hpp.

Referenced by `mlpack::gmm::GMM< FittingType >::Covariances()`.

22.46.4.2 `template<typename FittingType = EMFit<>> size_t mlpack::gmm::GMM< FittingType >::dimensionality [private]`

The dimensionality of the model.

Definition at line 87 of file gmm.hpp.

Referenced by `mlpack::gmm::GMM< FittingType >::Dimensionality()`.

22.46.4.3 `template<typename FittingType = EMFit<>> FittingType& mlpack::gmm::GMM< FittingType >::fitter [private]`

Reference to the fitting object we should use.

Definition at line 368 of file gmm.hpp.

Referenced by `mlpack::gmm::GMM< FittingType >::Fitter()`.

22.46.4.4 `template<typename FittingType = EMFit<>> size_t mlpack::gmm::GMM< FittingType >::gaussians [private]`

The number of Gaussians in the model.

Definition at line 85 of file gmm.hpp.

Referenced by `mlpack::gmm::GMM< FittingType >::Gaussians()`.

22.46.4.5 `template<typename FittingType = EMFit<>> FittingType mlpack::gmm::GMM< FittingType >::localFitter [private]`

Locally-stored fitting object; in case the user did not pass one.

Definition at line 365 of file gmm.hpp.

22.46.4.6 `template<typename FittingType = EMFit<>> std::vector<arma::vec> mlpack::gmm::GMM< FittingType >::means [private]`

Vector of means; one for each Gaussian.

Definition at line 89 of file gmm.hpp.

Referenced by `mlpack::gmm::GMM< FittingType >::Means()`.

22.46.4.7 `template<typename FittingType = EMFit<>> arma::vec mlpack::gmm::GMM< FittingType >::weights`
`[private]`

Vector of a priori weights for each Gaussian.

Definition at line 93 of file gmm.hpp.

Referenced by `mlpack::gmm::GMM< FittingType >::Weights()`.

The documentation for this class was generated from the following file:

- `src/mlpack/methods/gmm/gmm.hpp`

22.47 mlpack::gmm::NoConstraint Class Reference

This class enforces no constraint on the covariance matrix.

Static Public Member Functions

- static void **ApplyConstraint** (const arma::mat &)
Do nothing, and do not modify the covariance matrix.

22.47.1 Detailed Description

This class enforces no constraint on the covariance matrix.

It's faster this way, although depending on your situation you may end up with a non-invertible covariance matrix.

Definition at line 27 of file no_constraint.hpp.

22.47.2 Member Function Documentation

22.47.2.1 `static void mlpack::gmm::NoConstraint::ApplyConstraint (const arma::mat &) [inline],[static]`

Do nothing, and do not modify the covariance matrix.

Definition at line 31 of file no_constraint.hpp.

The documentation for this class was generated from the following file:

- `src/mlpack/methods/gmm/no_constraint.hpp`

22.48 mlpack::gmm::PositiveDefiniteConstraint Class Reference

Given a covariance matrix, force the matrix to be positive definite.

Static Public Member Functions

- static void **ApplyConstraint** (arma::mat &covariance)
Apply the positive definiteness constraint to the given covariance matrix.

22.48.1 Detailed Description

Given a covariance matrix, force the matrix to be positive definite.

Definition at line 25 of file `positive_definite_constraint.hpp`.

22.48.2 Member Function Documentation

22.48.2.1 `static void mlpack::gmm::PositiveDefiniteConstraint::ApplyConstraint (arma::mat & covariance) [inline], [static]`

Apply the positive definiteness constraint to the given covariance matrix.

Parameters

<i>covariance</i>	Covariance matrix.
-------------------	--------------------

Definition at line 33 of file `positive_definite_constraint.hpp`.

References `mlpack::Log::Debug`.

The documentation for this class was generated from the following file:

- `src/mlpack/methods/gmm/positive_definite_constraint.hpp`

22.49 mlpack::hmm::HMM< Distribution > Class Template Reference

A class that represents a Hidden Markov Model with an arbitrary type of emission distribution.

Public Member Functions

- **HMM** (const size_t states, const Distribution emissions, const double **tolerance**=1e-5)
Create the Hidden Markov Model with the given number of hidden states and the given default distribution for emissions.
- **HMM** (const arma::vec &**initial**, const arma::mat &**transition**, const std::vector< Distribution > &**emission**, const double **tolerance**=1e-5)
Create the Hidden Markov Model with the given initial probability vector, the given transition matrix, and the given emission distributions.
- size_t **Dimensionality** () const
Get the dimensionality of observations.
- size_t & **Dimensionality** ()
Set the dimensionality of observations.
- const std::vector< Distribution > & **Emission** () const
Return the emission distributions.
- std::vector< Distribution > & **Emission** ()
Return a modifiable emission probability matrix reference.
- double **Estimate** (const arma::mat &dataSeq, arma::mat &stateProb, arma::mat &forwardProb, arma::mat &backwardProb, arma::vec &scales) const
Estimate the probabilities of each hidden state at each time step for each given data observation, using the Forward-↔ Backward algorithm.
- double **Estimate** (const arma::mat &dataSeq, arma::mat &stateProb) const

Estimate the probabilities of each hidden state at each time step of each given data observation, using the Forward-Backward algorithm.

- void **Generate** (const size_t length, arma::mat &dataSequence, arma::Col< size_t > &stateSequence, const size_t startState=0) const

Generate a random data sequence of the given length.

- const arma::vec & **Initial** () const

Return the vector of initial state probabilities.

- arma::vec & **Initial** ()

Modify the vector of initial state probabilities.

- double **LogLikelihood** (const arma::mat &dataSeq) const

Compute the log-likelihood of the given data sequence.

- double **Predict** (const arma::mat &dataSeq, arma::Col< size_t > &stateSeq) const

Compute the most probable hidden state sequence for the given data sequence, using the Viterbi algorithm, returning the log-likelihood of the most likely state sequence.

- double **Tolerance** () const

Get the tolerance of the Baum-Welch algorithm.

- double & **Tolerance** ()

Modify the tolerance of the Baum-Welch algorithm.

- std::string **ToString** () const

Returns a string representation of this object.

- void **Train** (const std::vector< arma::mat > &dataSeq)

Train the model using the Baum-Welch algorithm, with only the given unlabeled observations.

- void **Train** (const std::vector< arma::mat > &dataSeq, const std::vector< arma::Col< size_t > > &stateSeq)

Train the model using the given labeled observations; the transition and emission matrices are directly estimated.

- const arma::mat & **Transition** () const

Return the transition matrix.

- arma::mat & **Transition** ()

Return a modifiable transition matrix reference.

Private Member Functions

- void **Backward** (const arma::mat &dataSeq, const arma::vec &scales, arma::mat &backwardProb) const

The Backward algorithm (part of the Forward-Backward algorithm).

- void **Forward** (const arma::mat &dataSeq, arma::vec &scales, arma::mat &forwardProb) const

The Forward algorithm (part of the Forward-Backward algorithm).

Private Attributes

- size_t **dimensionality**

Dimensionality of observations.

- std::vector< Distribution > **emission**

Set of emission probability distributions; one for each state.

- arma::vec **initial**

Initial state probability vector.

- double **tolerance**

Tolerance of Baum-Welch algorithm.

- arma::mat **transition**

Transition probability matrix.

22.49.1 Detailed Description

```
template<typename Distribution = distribution::DiscreteDistribution>class mlpack::hmm::HMM< Distribution >
```

A class that represents a Hidden Markov Model with an arbitrary type of emission distribution.

This **HMM** (p. 284) class supports training (supervised and unsupervised), prediction of state sequences via the Viterbi algorithm, estimation of state probabilities, generation of random sequences, and calculation of the log-likelihood of a given sequence.

The template parameter, *Distribution*, specifies the distribution which the emissions follow. The class should implement the following functions:

```
class Distribution
{
public:
  // The type of observation used by this distribution.
  typedef something DataType;

  // Return the probability of the given observation.
  double Probability(const DataType& observation) const;

  // Estimate the distribution based on the given observations.
  void Estimate(const std::vector<DataType>& observations);

  // Estimate the distribution based on the given observations, given also
  // the probability of each observation coming from this distribution.
  void Estimate(const std::vector<DataType>& observations,
               const std::vector<double>& probabilities);
};
```

See the **mlpack::distribution::DiscreteDistribution** (p. 214) class for an example. One would use the **DiscreteDistribution** class when the observations are non-negative integers. Other distributions could be Gaussians, a mixture of Gaussians (GMM), or any other probability distribution implementing the four *Distribution* functions.

Usage of the **HMM** (p. 284) class generally involves either training an **HMM** (p. 284) or loading an already-known **HMM** (p. 284) and taking probability measurements of sequences. Example code for supervised training of a Gaussian **HMM** (p. 284) (that is, where the emission output distribution is a single Gaussian for each hidden state) is given below.

```
extern arma::mat observations; // Each column is an observation.
extern arma::Col<size_t> states; // Hidden states for each observation.
// Create an untrained HMM with 5 hidden states and default (N(0, 1))
// Gaussian distributions with the dimensionality of the dataset.
HMM<GaussianDistribution> hmm(5, GaussianDistribution(observations.n_rows));

// Train the HMM (the labels could be omitted to perform unsupervised
// training).
hmm.Train(observations, states);
```

Once initialized, the **HMM** (p. 284) can evaluate the probability of a certain sequence (with **LogLikelihood()** (p. 290)), predict the most likely sequence of hidden states (with **Predict()** (p. 290)), generate a sequence (with **Generate()** (p. 289)), or estimate the probabilities of each state for a sequence of observations (with **Estimate()** (p. 288)).

Template Parameters

<i>Distribution</i>	Type of emission distribution for this HMM (p. 284).
---------------------	---

Definition at line 85 of file `hmm.hpp`.

22.49.2 Constructor & Destructor Documentation

22.49.2.1 `template<typename Distribution = distribution::DiscreteDistribution> mlpack::hmm::HMM< Distribution >::HMM (const size_t states, const Distribution emissions, const double tolerance = 1e-5)`

Create the Hidden Markov Model with the given number of hidden states and the given default distribution for emissions.

The dimensionality of the observations is taken from the emissions variable, so it is important that the given default emission distribution is set with the correct dimensionality. Alternately, set the dimensionality with **Dimensionality()** (p. 288). Optionally, the tolerance for convergence of the Baum-Welch algorithm can be set.

By default, the transition matrix and initial probability vector are set to contain equal probability for each state.

Parameters

<i>states</i>	Number of states.
<i>emissions</i>	Default distribution for emissions.
<i>tolerance</i>	Tolerance for convergence of training algorithm (Baum-Welch).

```
22.49.2.2  template<typename Distribution = distribution::DiscreteDistribution> mlpack::hmm::HMM< Distribution >::HMM (
    const arma::vec & initial, const arma::mat & transition, const std::vector< Distribution > & emission, const double
    tolerance = 1e-5 )
```

Create the Hidden Markov Model with the given initial probability vector, the given transition matrix, and the given emission distributions.

The dimensionality of the observations of the **HMM** (p. 284) are taken from the given emission distributions. Alternately, the dimensionality can be set with **Dimensionality()** (p. 288).

The initial state probability vector should have length equal to the number of states, and each entry represents the probability of being in the given state at time $T = 0$ (the beginning of a sequence).

The transition matrix should be such that $T(i, j)$ is the probability of transition to state i from state j . The columns of the matrix should sum to 1.

The emission matrix should be such that $E(i, j)$ is the probability of emission i while in state j . The columns of the matrix should sum to 1.

Optionally, the tolerance for convergence of the Baum-Welch algorithm can be set.

Parameters

<i>initial</i>	Initial state probabilities.
<i>transition</i>	Transition matrix.
<i>emission</i>	Emission distributions.
<i>tolerance</i>	Tolerance for convergence of training algorithm (Baum-Welch).

22.49.3 Member Function Documentation

```
22.49.3.1  template<typename Distribution = distribution::DiscreteDistribution> void mlpack::hmm::HMM< Distribution
    >::Backward ( const arma::mat & dataSeq, const arma::vec & scales, arma::mat & backwardProb ) const
    [private]
```

The Backward algorithm (part of the Forward-Backward algorithm).

Computes backward probabilities for each state for each observation in the given data sequence, using the scaling factors found (presumably) by **Forward()** (p. 289). The returned matrix has rows equal to the number of hidden states and columns equal to the number of observations.

Parameters

<i>dataSeq</i>	Data sequence to compute probabilities for.
<i>scales</i>	Vector of scaling factors.
<i>backwardProb</i>	Matrix in which backward probabilities will be saved.

22.49.3.2 `template<typename Distribution = distribution::DiscreteDistribution> size_t mlpack::hmm::HMM< Distribution >::Dimensionality () const [inline]`

Get the dimensionality of observations.

Definition at line 286 of file `hmm.hpp`.

References `mlpack::hmm::HMM< Distribution >::dimensionality`.

22.49.3.3 `template<typename Distribution = distribution::DiscreteDistribution> size_t& mlpack::hmm::HMM< Distribution >::Dimensionality () [inline]`

Set the dimensionality of observations.

Definition at line 288 of file `hmm.hpp`.

References `mlpack::hmm::HMM< Distribution >::dimensionality`.

22.49.3.4 `template<typename Distribution = distribution::DiscreteDistribution> const std::vector<Distribution>& mlpack::hmm::HMM< Distribution >::Emission () const [inline]`

Return the emission distributions.

Definition at line 281 of file `hmm.hpp`.

References `mlpack::hmm::HMM< Distribution >::emission`.

22.49.3.5 `template<typename Distribution = distribution::DiscreteDistribution> std::vector<Distribution>& mlpack::hmm::HMM< Distribution >::Emission () [inline]`

Return a modifiable emission probability matrix reference.

Definition at line 283 of file `hmm.hpp`.

References `mlpack::hmm::HMM< Distribution >::emission`.

22.49.3.6 `template<typename Distribution = distribution::DiscreteDistribution> double mlpack::hmm::HMM< Distribution >::Estimate (const arma::mat & dataSeq, arma::mat & stateProb, arma::mat & forwardProb, arma::mat & backwardProb, arma::vec & scales) const`

Estimate the probabilities of each hidden state at each time step for each given data observation, using the Forward-↵ Backward algorithm.

Each matrix which is returned has columns equal to the number of data observations, and rows equal to the number of hidden states in the model. The log-likelihood of the most probable sequence is returned.

Parameters

<i>dataSeq</i>	Sequence of observations.
<i>stateProb</i>	Matrix in which the probabilities of each state at each time interval will be stored.
<i>forwardProb</i>	Matrix in which the forward probabilities of each state at each time interval will be stored.
<i>backwardProb</i>	Matrix in which the backward probabilities of each state at each time interval will be stored.
<i>scales</i>	Vector in which the scaling factors at each time interval will be stored.

Returns

Log-likelihood of most likely state sequence.

22.49.3.7 `template<typename Distribution = distribution::DiscreteDistribution> double mlpack::hmm::HMM< Distribution >::Estimate (const arma::mat & dataSeq, arma::mat & stateProb) const`

Estimate the probabilities of each hidden state at each time step of each given data observation, using the Forward-Backward algorithm.

The returned matrix of state probabilities has columns equal to the number of data observations, and rows equal to the number of hidden states in the model. The log-likelihood of the most probable sequence is returned.

Parameters

<i>dataSeq</i>	Sequence of observations.
<i>stateProb</i>	Probabilities of each state at each time interval.

Returns

Log-likelihood of most likely state sequence.

22.49.3.8 `template<typename Distribution = distribution::DiscreteDistribution> void mlpack::hmm::HMM< Distribution >::Forward (const arma::mat & dataSeq, arma::vec & scales, arma::mat & forwardProb) const [private]`

The Forward algorithm (part of the Forward-Backward algorithm).

Computes forward probabilities for each state for each observation in the given data sequence. The returned matrix has rows equal to the number of hidden states and columns equal to the number of observations.

Parameters

<i>dataSeq</i>	Data sequence to compute probabilities for.
<i>scales</i>	Vector in which scaling factors will be saved.
<i>forwardProb</i>	Matrix in which forward probabilities will be saved.

22.49.3.9 `template<typename Distribution = distribution::DiscreteDistribution> void mlpack::hmm::HMM< Distribution >::Generate (const size_t length, arma::mat & dataSequence, arma::Col< size_t > & stateSequence, const size_t startState = 0) const`

Generate a random data sequence of the given length.

The data sequence is stored in the dataSequence parameter, and the state sequence is stored in the stateSequence parameter. Each column of dataSequence represents a random observation.

Parameters

<i>length</i>	Length of random sequence to generate.
<i>dataSequence</i>	Vector to store data in.
<i>stateSequence</i>	Vector to store states in.
<i>startState</i>	Hidden state to start sequence in (default 0).

22.49.3.10 `template<typename Distribution = distribution::DiscreteDistribution> const arma::vec& mlpack::hmm::HMM< Distribution >::Initial () const [inline]`

Return the vector of initial state probabilities.

Definition at line 271 of file `hmm.hpp`.

References `mlpack::hmm::HMM< Distribution >::initial`.

22.49.3.11 `template<typename Distribution = distribution::DiscreteDistribution> arma::vec& mlpack::hmm::HMM< Distribution >::Initial () [inline]`

Modify the vector of initial state probabilities.

Definition at line 273 of file `hmm.hpp`.

References `mlpack::hmm::HMM< Distribution >::initial`.

22.49.3.12 `template<typename Distribution = distribution::DiscreteDistribution> double mlpack::hmm::HMM< Distribution >::LogLikelihood (const arma::mat & dataSeq) const`

Compute the log-likelihood of the given data sequence.

Parameters

<i>dataSeq</i>	Data sequence to evaluate the likelihood of.
----------------	--

Returns

Log-likelihood of the given sequence.

22.49.3.13 `template<typename Distribution = distribution::DiscreteDistribution> double mlpack::hmm::HMM< Distribution >::Predict (const arma::mat & dataSeq, arma::Col< size_t > & stateSeq) const`

Compute the most probable hidden state sequence for the given data sequence, using the Viterbi algorithm, returning the log-likelihood of the most likely state sequence.

Parameters

<i>dataSeq</i>	Sequence of observations.
<i>stateSeq</i>	Vector in which the most probable state sequence will be stored.

Returns

Log-likelihood of most probable state sequence.

22.49.3.14 `template<typename Distribution = distribution::DiscreteDistribution> double mlpack::hmm::HMM< Distribution >::Tolerance () const [inline]`

Get the tolerance of the Baum-Welch algorithm.

Definition at line 291 of file `hmm.hpp`.

References `mlpack::hmm::HMM< Distribution >::tolerance`.

22.49.3.15 `template<typename Distribution = distribution::DiscreteDistribution> double& mlpack::hmm::HMM< Distribution >::Tolerance () [inline]`

Modify the tolerance of the Baum-Welch algorithm.

Definition at line 293 of file `hmm.hpp`.

References `mlpack::hmm::HMM< Distribution >::tolerance`.

22.49.3.16 `template<typename Distribution = distribution::DiscreteDistribution> std::string mlpack::hmm::HMM< Distribution >::ToString () const`

Returns a string representation of this object.

22.49.3.17 `template<typename Distribution = distribution::DiscreteDistribution> void mlpack::hmm::HMM< Distribution >::Train (const std::vector< arma::mat > & dataSeq)`

Train the model using the Baum-Welch algorithm, with only the given unlabeled observations.

Instead of giving a guess transition and emission matrix here, do that in the constructor. Each matrix in the vector of data sequences holds an individual data sequence; each point in each individual data sequence should be a column in the matrix. The number of rows in each matrix should be equal to the dimensionality of the **HMM** (p. 284) (which is set in the constructor).

It is preferable to use the other overload of **Train()** (p. 291), with labeled data. That will produce much better results. However, if labeled data is unavailable, this will work. In addition, it is possible to use **Train()** (p. 291) with labeled data first, and then continue to train the model using this overload of **Train()** (p. 291) with unlabeled data.

The tolerance of the Baum-Welch algorithm can be set either in the constructor or with the **Tolerance()** (p. 291) method. When the change in log-likelihood of the model between iterations is less than the tolerance, the Baum-Welch algorithm terminates.

Note

Train() (p. 291) can be called multiple times with different sequences; each time it is called, it uses the current parameters of the **HMM** (p. 284) as a starting point for training.

Parameters

<i>dataSeq</i>	Vector of observation sequences.
----------------	----------------------------------

22.49.3.18 `template<typename Distribution = distribution::DiscreteDistribution> void mlpack::hmm::HMM< Distribution >::Train (const std::vector< arma::mat > & dataSeq, const std::vector< arma::Col< size_t > > & stateSeq)`

Train the model using the given labeled observations; the transition and emission matrices are directly estimated.

Each matrix in the vector of data sequences corresponds to a vector in the vector of state sequences. Each point in each individual data sequence should be a column in the matrix, and its state should be the corresponding element in the state sequence vector. For instance, `dataSeq[0].col(3)` corresponds to the fourth observation in the first data sequence, and its state is `stateSeq[0][3]`. The number of rows in each matrix should be equal to the dimensionality of the **HMM** (p. 284) (which is set in the constructor).

Note

Train() (p. 291) can be called multiple times with different sequences; each time it is called, it uses the current parameters of the **HMM** (p. 284) as a starting point for training.

Parameters

<i>dataSeq</i>	Vector of observation sequences.
<i>stateSeq</i>	Vector of state sequences, corresponding to each observation.

22.49.3.19 `template<typename Distribution = distribution::DiscreteDistribution> const arma::mat& mlpack::hmm::HMM< Distribution >::Transition () const [inline]`

Return the transition matrix.

Definition at line 276 of file `hmm.hpp`.

References `mlpack::hmm::HMM< Distribution >::transition`.

22.49.3.20 `template<typename Distribution = distribution::DiscreteDistribution> arma::mat& mlpack::hmm::HMM< Distribution >::Transition () [inline]`

Return a modifiable transition matrix reference.

Definition at line 278 of file `hmm.hpp`.

References `mlpack::hmm::HMM< Distribution >::transition`.

22.49.4 Member Data Documentation

22.49.4.1 `template<typename Distribution = distribution::DiscreteDistribution> size_t mlpack::hmm::HMM< Distribution >::dimensionality [private]`

Dimensionality of observations.

Definition at line 342 of file `hmm.hpp`.

Referenced by `mlpack::hmm::HMM< Distribution >::Dimensionality()`.

22.49.4.2 `template<typename Distribution = distribution::DiscreteDistribution> std::vector<Distribution> mlpack::hmm::HMM< Distribution >::emission [private]`

Set of emission probability distributions; one for each state.

Definition at line 339 of file `hmm.hpp`.

Referenced by `mlpack::hmm::HMM< Distribution >::Emission()`.

22.49.4.3 `template<typename Distribution = distribution::DiscreteDistribution> arma::vec mlpack::hmm::HMM< Distribution >::initial [private]`

Initial state probability vector.

Definition at line 333 of file `hmm.hpp`.

Referenced by `mlpack::hmm::HMM< Distribution >::Initial()`.

22.49.4.4 `template<typename Distribution = distribution::DiscreteDistribution> double mlpack::hmm::HMM< Distribution >::tolerance [private]`

Tolerance of Baum-Welch algorithm.

Definition at line 345 of file `hmm.hpp`.

Referenced by `mlpack::hmm::HMM< Distribution >::Tolerance()`.

22.49.4.5 `template<typename Distribution = distribution::DiscreteDistribution> arma::mat mlpack::hmm::HMM< Distribution >::transition [private]`

Transition probability matrix.

Definition at line 336 of file `hmm.hpp`.

Referenced by `mlpack::hmm::HMM< Distribution >::Transition()`.

The documentation for this class was generated from the following file:

- `src/mlpack/methods/hmm/hmm.hpp`

22.50 mlpack::kernel::CosineDistance Class Reference

The cosine distance (or cosine similarity).

Public Member Functions

- `std::string ToString () const`
Returns a string representation of this object.

Static Public Member Functions

- `template<typename VecType > static double Evaluate (const VecType &a, const VecType &b)`
Computes the cosine distance between two points.

22.50.1 Detailed Description

The cosine distance (or cosine similarity).

It is defined by

$$d(a, b) = \frac{a^T b}{||a|| ||b||}$$

and this class assumes the standard L2 inner product.

Definition at line 32 of file cosine_distance.hpp.

22.50.2 Member Function Documentation

22.50.2.1 `template<typename VecType > static double mlpack::kernel::CosineDistance::Evaluate (const VecType & a, const VecType & b) [static]`

Computes the cosine distance between two points.

Parameters

<i>a</i>	First vector.
<i>b</i>	Second vector.

Returns

`d(a, b)`.

22.50.2.2 `std::string mlpack::kernel::CosineDistance::ToString () const [inline]`

Returns a string representation of this object.

Definition at line 48 of file cosine_distance.hpp.

The documentation for this class was generated from the following file:

- src/mlpack/core/kernels/cosine_distance.hpp

22.51 mlpack::kernel::EpanechnikovKernel Class Reference

The Epanechnikov kernel, defined as.

Public Member Functions

- **EpanechnikovKernel** (const double **bandwidth**=1.0)
Instantiate the Epanechnikov kernel with the given bandwidth (default 1.0).
- `template<typename VecType >`
double ConvolutionIntegral (const VecType &a, const VecType &b)
Obtains the convolution integral [integral of $K(||x-a||) K(||b-x||) dx$] for the two vectors.
- `template<typename Vec1Type , typename Vec2Type >`
double Evaluate (const Vec1Type &a, const Vec2Type &b) const
Evaluate the Epanechnikov kernel on the given two inputs.
- **double Evaluate** (const double distance) const
Evaluate the Epanechnikov kernel given that the distance between the two input points is known.

- double **Normalizer** (const size_t dimension)
Compute the normalizer of this Epanechnikov kernel for the given dimension.
- std::string **ToString** () const

Private Attributes

- double **bandwidth**
Bandwidth of the kernel.
- double **inverseBandwidthSquared**
Cached value of the inverse bandwidth squared (to speed up computation).

22.51.1 Detailed Description

The Epanechnikov kernel, defined as.

$$K(x, y) = \max\{0, 1 - \|x - y\|_2^2 / b^2\}$$

where b is the bandwidth the of the kernel (defaults to 1.0).

Definition at line 31 of file epanechnikov_kernel.hpp.

22.51.2 Constructor & Destructor Documentation

22.51.2.1 mlpack::kernel::EpanechnikovKernel::EpanechnikovKernel (const double *bandwidth* = 1.0) [inline]

Instantiate the Epanechnikov kernel with the given bandwidth (default 1.0).

Parameters

<i>bandwidth</i>	Bandwidth of the kernel.
------------------	--------------------------

Definition at line 39 of file epanechnikov_kernel.hpp.

22.51.3 Member Function Documentation

22.51.3.1 template<typename VecType > double mlpack::kernel::EpanechnikovKernel::ConvolutionIntegral (const VecType & *a*, const VecType & *b*)

Obtains the convolution integral [integral of $K(\|x-a\|) K(\|b-x\|) dx$] for the two vectors.

Template Parameters

<i>VecType</i>	Type of vector (arma::vec, arma::spvec should be expected).
----------------	---

Parameters

<i>a</i>	First vector.
<i>b</i>	Second vector.

Returns

the convolution integral value.

22.51.3.2 `template<typename Vec1Type , typename Vec2Type > double mlpack::kernel::EpanechnikovKernel::Evaluate (const Vec1Type & a, const Vec2Type & b) const`

Evaluate the Epanechnikov kernel on the given two inputs.

Parameters

<i>a</i>	One input vector.
<i>b</i>	The other input vector.

22.51.3.3 `double mlpack::kernel::EpanechnikovKernel::Evaluate (const double distance) const`

Evaluate the Epanechnikov kernel given that the distance between the two input points is known.

22.51.3.4 `double mlpack::kernel::EpanechnikovKernel::Normalizer (const size_t dimension)`

Compute the normalizer of this Epanechnikov kernel for the given dimension.

Parameters

<i>dimension</i>	Dimension to calculate the normalizer for.
------------------	--

22.51.3.5 `std::string mlpack::kernel::EpanechnikovKernel::ToString () const`

22.51.4 Member Data Documentation

22.51.4.1 `double mlpack::kernel::EpanechnikovKernel::bandwidth [private]`

Bandwidth of the kernel.

Definition at line 84 of file `epanechnikov_kernel.hpp`.

22.51.4.2 `double mlpack::kernel::EpanechnikovKernel::inverseBandwidthSquared [private]`

Cached value of the inverse bandwidth squared (to speed up computation).

Definition at line 86 of file `epanechnikov_kernel.hpp`.

The documentation for this class was generated from the following file:

- `src/mlpack/core/kernels/epanechnikov_kernel.hpp`

22.52 mlpack::kernel::ExampleKernel Class Reference

An example kernel function.

Public Member Functions

- **ExampleKernel** ()

The default constructor, which takes no parameters.

- `std::string ToString () const`

Returns a string for the kernel object; in this case, with only the memory address for the kernel.

Static Public Member Functions

- `template<typename VecType >`
`static double ConvolutionIntegral (const VecType &a, const VecType &b)`
Obtains the convolution integral $\int K(\|x-a\|)K(\|b-x\|)dx$ for the two vectors.
- `template<typename VecType >`
`static double Evaluate (const VecType &a, const VecType &b)`
Evaluates the kernel function for two given vectors.
- `static double Normalizer ()`
Obtains the normalizing volume for the kernel with dimension $\$dimension\$$.

22.52.1 Detailed Description

An example kernel function.

This is not a useful kernel, but it implements the two functions necessary to satisfy the Kernel policy (so that a class can be used whenever an MLPACK method calls for a `typename Kernel` template parameter).

All that is necessary is a constructor and an **Evaluate()** (p. 298) function. More methods could be added; for instance, one useful idea is a constructor which takes parameters for a kernel (for instance, the width of the Gaussian for a Gaussian kernel). However, MLPACK methods cannot count on these various constructors existing, which is why most methods allow passing an already-instantiated kernel object (and by default the method will construct the kernel with the default constructor). So, for instance,

```
GaussianKernel k(5.0);
KDE<GaussianKernel> kde(dataset, k);
```

will set up KDE using a Gaussian kernel with a width of 5.0, but

```
KDE<GaussianKernel> kde(dataset);
```

will create the kernel with the default constructor. It is important (but not strictly mandatory) that your default constructor still gives a working kernel.

Note

Not all kernels require state. For instance, the regular dot product needs no parameters. In that case, no local variables are necessary and **Evaluate()** (p. 298) can (and should) be declared static. However, for greater generalization, MLPACK methods expect all kernels to require state and hence must store instantiated kernel functions; this is why a default constructor is necessary.

Definition at line 86 of file `example_kernel.hpp`.

22.52.2 Constructor & Destructor Documentation

22.52.2.1 mlpack::kernel::ExampleKernel::ExampleKernel () [inline]

The default constructor, which takes no parameters.

Because our simple example kernel has no internal parameters that need to be stored, the constructor does not need to do anything. For a more complex example, see the **GaussianKernel** (p. 299), which stores an internal parameter.

Definition at line 95 of file example_kernel.hpp.

22.52.3 Member Function Documentation

22.52.3.1 `template<typename VecType > static double mlpack::kernel::ExampleKernel::ConvolutionIntegral (const VecType & a, const VecType & b) [inline],[static]`

Obtains the convolution integral $\int K(\|x-a\|)K(\|b-x\|)dx$ for the two vectors.

In this case, because our simple example kernel has no internal parameters, we can declare the function static. For a more complex example which cannot be declared static, see the **GaussianKernel** (p. 299), which stores an internal parameter.

Template Parameters

<i>VecType</i>	Type of vector (arma::vec, arma::spvec should be expected).
----------------	---

Parameters

<i>a</i>	First vector.
<i>b</i>	Second vector.

Returns

the convolution integral value.

Definition at line 136 of file example_kernel.hpp.

22.52.3.2 `template<typename VecType > static double mlpack::kernel::ExampleKernel::Evaluate (const VecType & a, const VecType & b) [inline],[static]`

Evaluates the kernel function for two given vectors.

In this case, because our simple example kernel has no internal parameters, we can declare the function static. For a more complex example which cannot be declared static, see the **GaussianKernel** (p. 299), which stores an internal parameter.

Template Parameters

<i>VecType</i>	Type of vector (arma::vec, arma::spvec should be expected).
----------------	---

Parameters

<i>a</i>	First vector.
<i>b</i>	Second vector.

Returns

$K(a, b)$.

Definition at line 109 of file example_kernel.hpp.

22.52.3.3 `static double mlpack::kernel::ExampleKernel::Normalizer () [inline],[static]`

Obtains the normalizing volume for the kernel with dimension `$dimension$`.

In this case, because our simple example kernel has no internal parameters, we can declare the function static. For a more complex example which cannot be declared static, see the **GaussianKernel** (p. 299), which stores an internal parameter.

Parameters

<i>dimension</i>	the dimension of the space.
------------------	-----------------------------

Returns

the normalization constant.

Definition at line 149 of file `example_kernel.hpp`.

22.52.3.4 `std::string mlpack::kernel::ExampleKernel::ToString () const [inline]`

Returns a string for the kernel object; in this case, with only the memory address for the kernel.

If your kernel has any members, your **ToString()** (p. 299) method should include those as necessary as well.

Definition at line 116 of file `example_kernel.hpp`.

The documentation for this class was generated from the following file:

- `src/mlpack/core/kernels/example_kernel.hpp`

22.53 mlpack::kernel::GaussianKernel Class Reference

The standard Gaussian kernel.

Public Member Functions

- **GaussianKernel** ()
Default constructor; sets bandwidth to 1.0.
- **GaussianKernel** (const double **bandwidth**)
Construct the Gaussian kernel with a custom bandwidth.
- double **Bandwidth** () const
Get the bandwidth.
- void **Bandwidth** (const double **bandwidth**)
Modify the bandwidth.
- template<typename VecType >
double **ConvolutionIntegral** (const VecType &a, const VecType &b)
Obtain a convolution integral of the Gaussian kernel.
- template<typename VecType >
double **Evaluate** (const VecType &a, const VecType &b) const
Evaluation of the Gaussian kernel.
- double **Evaluate** (const double t) const

Evaluation of the Gaussian kernel given the distance between two points.

- double **Gamma** () const
Get the precalculated constant.
- double **Normalizer** (const size_t dimension)
Obtain the normalization constant of the Gaussian kernel.
- std::string **ToString** () const
Convert object to string.

Private Attributes

- double **bandwidth**
Kernel bandwidth.
- double **gamma**
Precalculated constant depending on the bandwidth; $\gamma = -\frac{1}{2\mu^2}$.

22.53.1 Detailed Description

The standard Gaussian kernel.

Given two vectors x , y , and a bandwidth μ (set in the constructor),

$$K(x, y) = \exp\left(-\frac{\|x - y\|^2}{2\mu^2}\right).$$

The implementation is all in the header file because it is so simple.

Definition at line 35 of file gaussian_kernel.hpp.

22.53.2 Constructor & Destructor Documentation

22.53.2.1 mlpack::kernel::GaussianKernel::GaussianKernel () [inline]

Default constructor; sets bandwidth to 1.0.

Definition at line 41 of file gaussian_kernel.hpp.

22.53.2.2 mlpack::kernel::GaussianKernel::GaussianKernel (const double *bandwidth*) [inline]

Construct the Gaussian kernel with a custom bandwidth.

Parameters

<i>bandwidth</i>	The bandwidth of the kernel (μ).
------------------	--

Definition at line 49 of file gaussian_kernel.hpp.

22.53.3 Member Function Documentation

22.53.3.1 double mlpack::kernel::GaussianKernel::Bandwidth () const [inline]

Get the bandwidth.

Definition at line 112 of file gaussian_kernel.hpp.

References bandwidth.

22.53.3.2 void mpack::kernel::GaussianKernel::Bandwidth (const double *bandwidth*) [inline]

Modify the bandwidth.

This takes an argument because we must update the precalculated constant (gamma).

Definition at line 116 of file gaussian_kernel.hpp.

References bandwidth, and gamma.

22.53.3.3 template<typename VecType > double mpack::kernel::GaussianKernel::ConvolutionIntegral (const VecType & *a*, const VecType & *b*) [inline]

Obtain a convolution integral of the Gaussian kernel.

Parameters

<i>a,first</i>	vector
<i>b,second</i>	vector

Returns

the convolution integral

Definition at line 104 of file gaussian_kernel.hpp.

References Evaluate(), mpack::metric::LMetric< Power, TakeRoot >::Evaluate(), and Normalizer().

22.53.3.4 template<typename VecType > double mpack::kernel::GaussianKernel::Evaluate (const VecType & *a*, const VecType & *b*) const [inline]

Evaluation of the Gaussian kernel.

This could be generalized to use any distance metric, not the Euclidean distance, but for now, the Euclidean distance is used.

Template Parameters

<i>VecType</i>	Type of vector (likely arma::vec or arma::spvec).
----------------	---

Parameters

<i>a</i>	First vector.
<i>b</i>	Second vector.

Returns

K(a, b) using the bandwidth (μ) specified in the constructor.

Definition at line 66 of file gaussian_kernel.hpp.

References mpack::metric::LMetric< Power, TakeRoot >::Evaluate(), and gamma.

Referenced by ConvolutionIntegral().

22.53.3.5 `double mlpack::kernel::GaussianKernel::Evaluate (const double t) const` `[inline]`

Evaluation of the Gaussian kernel given the distance between two points.

Parameters

<i>t</i>	The distance between the two points the kernel is evaluated on.
----------	---

Returns

$K(t)$ using the bandwidth (μ) specified in the constructor.

Definition at line 79 of file gaussian_kernel.hpp.

References gamma.

22.53.3.6 double mpack::kernel::GaussianKernel::Gamma () const [inline]

Get the precalculated constant.

Definition at line 123 of file gaussian_kernel.hpp.

References gamma.

22.53.3.7 double mpack::kernel::GaussianKernel::Normalizer (const size_t *dimension*) [inline]

Obtain the normalization constant of the Gaussian kernel.

Parameters

<i>dimension</i>	
------------------	--

Returns

the normalization constant

Definition at line 91 of file gaussian_kernel.hpp.

References bandwidth, and M_PI.

Referenced by ConvolutionIntegral().

22.53.3.8 std::string mpack::kernel::GaussianKernel::ToString () const [inline]

Convert object to string.

Definition at line 126 of file gaussian_kernel.hpp.

References bandwidth.

22.53.4 Member Data Documentation

22.53.4.1 double mpack::kernel::GaussianKernel::bandwidth [private]

Kernel bandwidth.

Definition at line 136 of file gaussian_kernel.hpp.

Referenced by Bandwidth(), Normalizer(), and ToString().

22.53.4.2 `double mlpack::kernel::GaussianKernel::gamma` [private]

Precalculated constant depending on the bandwidth; $\gamma = -\frac{1}{2\mu^2}$.

Definition at line 140 of file `gaussian_kernel.hpp`.

Referenced by `Bandwidth()`, `Evaluate()`, and `Gamma()`.

The documentation for this class was generated from the following file:

- `src/mlpack/core/kernels/gaussian_kernel.hpp`

22.54 mlpack::kernel::HyperbolicTangentKernel Class Reference

Hyperbolic tangent kernel.

Public Member Functions

- **HyperbolicTangentKernel** ()
This constructor sets the default scale to 1.0 and offset to 0.0.
- **HyperbolicTangentKernel** (double **scale**, double **offset**)
Construct the hyperbolic tangent kernel with custom scale factor and offset.
- `template<typename VecType >`
double **Evaluate** (const VecType &a, const VecType &b)
Evaluate the hyperbolic tangent kernel.
- double **Offset** () const
Get offset for the kernel.
- double & **Offset** ()
Modify offset for the kernel.
- double **Scale** () const
Get scale factor.
- double & **Scale** ()
Modify scale factor.
- `std::string` **ToString** () const
Convert object to string.

Private Attributes

- double **offset**
- double **scale**

22.54.1 Detailed Description

Hyperbolic tangent kernel.

For any two vectors x, y and a given scale s and offset t

$$K(x, y) = \tanh(s \langle x, y \rangle + t)$$

Definition at line 30 of file `hyperbolic_tangent_kernel.hpp`.

22.54.2 Constructor & Destructor Documentation

22.54.2.1 mpack::kernel::HyperbolicTangentKernel::HyperbolicTangentKernel () [inline]

This constructor sets the default scale to 1.0 and offset to 0.0.

Definition at line 36 of file hyperbolic_tangent_kernel.hpp.

22.54.2.2 mpack::kernel::HyperbolicTangentKernel::HyperbolicTangentKernel (double *scale*, double *offset*) [inline]

Construct the hyperbolic tangent kernel with custom scale factor and offset.

Parameters

<i>scale</i>	Scaling factor for $\langle x, y \rangle$.
<i>offset</i>	Kernel offset.

Definition at line 46 of file hyperbolic_tangent_kernel.hpp.

22.54.3 Member Function Documentation

22.54.3.1 template<typename VecType > double mpack::kernel::HyperbolicTangentKernel::Evaluate (const VecType & *a*, const VecType & *b*) [inline]

Evaluate the hyperbolic tangent kernel.

This evaluation uses Armadillo's dot() function.

Template Parameters

<i>VecType</i>	Type of vector (should be arma::vec or arma::spvec).
----------------	--

Parameters

<i>a</i>	First vector.
<i>b</i>	Second vector.

Returns

$K(a, b)$.

Definition at line 60 of file hyperbolic_tangent_kernel.hpp.

References offset, and scale.

22.54.3.2 double mpack::kernel::HyperbolicTangentKernel::Offset () const [inline]

Get offset for the kernel.

Definition at line 71 of file hyperbolic_tangent_kernel.hpp.

References offset.

22.54.3.3 double& mpack::kernel::HyperbolicTangentKernel::Offset () [inline]

Modify offset for the kernel.

Definition at line 73 of file `hyperbolic_tangent_kernel.hpp`.

References offset.

22.54.3.4 `double mpack::kernel::HyperbolicTangentKernel::Scale () const` `[inline]`

Get scale factor.

Definition at line 66 of file `hyperbolic_tangent_kernel.hpp`.

References scale.

22.54.3.5 `double& mpack::kernel::HyperbolicTangentKernel::Scale ()` `[inline]`

Modify scale factor.

Definition at line 68 of file `hyperbolic_tangent_kernel.hpp`.

References scale.

22.54.3.6 `std::string mpack::kernel::HyperbolicTangentKernel::ToString () const` `[inline]`

Convert object to string.

Definition at line 76 of file `hyperbolic_tangent_kernel.hpp`.

References offset, and scale.

22.54.4 Member Data Documentation

22.54.4.1 `double mpack::kernel::HyperbolicTangentKernel::offset` `[private]`

Definition at line 87 of file `hyperbolic_tangent_kernel.hpp`.

Referenced by `Evaluate()`, `Offset()`, and `ToString()`.

22.54.4.2 `double mpack::kernel::HyperbolicTangentKernel::scale` `[private]`

Definition at line 86 of file `hyperbolic_tangent_kernel.hpp`.

Referenced by `Evaluate()`, `Scale()`, and `ToString()`.

The documentation for this class was generated from the following file:

- `src/mpack/core/kernels/hyperbolic_tangent_kernel.hpp`

22.55 `mpack::kernel::KernelTraits< KernelType >` Class Template Reference

This is a template class that can provide information about various kernels.

Static Public Attributes

- static const bool **IsNormalized** = false

If true, then the kernel is normalized: $K(x, x) = K(y, y) = 1$ for all x .

22.55.1 Detailed Description

```
template<typename KernelType>class mpack::kernel::KernelTraits< KernelType >
```

This is a template class that can provide information about various kernels.

By default, this class will provide the weakest possible assumptions on kernels, and each kernel should override values as necessary. If a kernel doesn't need to override a value, then there's no need to write a **KernelTraits** (p. 306) specialization for that class.

Definition at line 29 of file kernel_traits.hpp.

22.55.2 Member Data Documentation

22.55.2.1 `template<typename KernelType > const bool mpack::kernel::KernelTraits< KernelType >::IsNormalized = false` [static]

If true, then the kernel is normalized: $K(x, x) = K(y, y) = 1$ for all x .

Definition at line 35 of file kernel_traits.hpp.

The documentation for this class was generated from the following file:

- src/mpack/core/kernels/**kernel_traits.hpp**

22.56 mpack::kernel::KernelTraits< CosineDistance > Class Template Reference

Kernel traits for the cosine distance.

Static Public Attributes

- static const bool **IsNormalized** = true
The cosine kernel is normalized: $K(x, x) = 1$ for all x .

22.56.1 Detailed Description

```
template<>class mpack::kernel::KernelTraits< CosineDistance >
```

Kernel traits for the cosine distance.

Definition at line 58 of file cosine_distance.hpp.

22.56.2 Member Data Documentation

22.56.2.1 `const bool mpack::kernel::KernelTraits< CosineDistance >::IsNormalized = true` [static]

The cosine kernel is normalized: $K(x, x) = 1$ for all x .

Definition at line 62 of file cosine_distance.hpp.

The documentation for this class was generated from the following file:

- src/mlpack/core/kernels/cosine_distance.hpp

22.57 mlpack::kernel::KernelTraits< EpanechnikovKernel > Class Template Reference

Kernel traits for the Epanechnikov kernel.

Static Public Attributes

- static const bool **IsNormalized** = true
The Epanechnikov kernel is normalized: $K(x, x) = 1$ for all x .

22.57.1 Detailed Description

```
template<> class mlpack::kernel::KernelTraits< EpanechnikovKernel >
```

Kernel traits for the Epanechnikov kernel.

Definition at line 92 of file epanechnikov_kernel.hpp.

22.57.2 Member Data Documentation

22.57.2.1 `const bool mlpack::kernel::KernelTraits< EpanechnikovKernel >::IsNormalized = true` [static]

The Epanechnikov kernel is normalized: $K(x, x) = 1$ for all x .

Definition at line 96 of file epanechnikov_kernel.hpp.

The documentation for this class was generated from the following file:

- src/mlpack/core/kernels/epanechnikov_kernel.hpp

22.58 mlpack::kernel::KernelTraits< GaussianKernel > Class Template Reference

Kernel traits for the Gaussian kernel.

Static Public Attributes

- static const bool **IsNormalized** = true
The Gaussian kernel is normalized: $K(x, x) = 1$ for all x .

22.58.1 Detailed Description

template<>class mlpack::kernel::KernelTraits< GaussianKernel >

Kernel traits for the Gaussian kernel.

Definition at line 145 of file gaussian_kernel.hpp.

22.58.2 Member Data Documentation

22.58.2.1 `const bool mlpack::kernel::KernelTraits< GaussianKernel >::IsNormalized = true` [static]

The Gaussian kernel is normalized: $K(x, x) = 1$ for all x .

Definition at line 149 of file gaussian_kernel.hpp.

The documentation for this class was generated from the following file:

- src/mlpack/core/kernels/gaussian_kernel.hpp

22.59 mlpack::kernel::KernelTraits< LaplacianKernel > Class Template Reference

Kernel traits of the Laplacian kernel.

Static Public Attributes

- static const bool **IsNormalized** = true
The Laplacian kernel is normalized: $K(x, x) = 1$ for all x .

22.59.1 Detailed Description

template<>class mlpack::kernel::KernelTraits< LaplacianKernel >

Kernel traits of the Laplacian kernel.

Definition at line 103 of file laplacian_kernel.hpp.

22.59.2 Member Data Documentation

22.59.2.1 `const bool mlpack::kernel::KernelTraits< LaplacianKernel >::IsNormalized = true` [static]

The Laplacian kernel is normalized: $K(x, x) = 1$ for all x .

Definition at line 107 of file laplacian_kernel.hpp.

The documentation for this class was generated from the following file:

- src/mlpack/core/kernels/laplacian_kernel.hpp

22.60 mlpack::kernel::KernelTraits< SphericalKernel > Class Template Reference

Kernel traits for the spherical kernel.

Static Public Attributes

- static const bool **IsNormalized** = true

The spherical kernel is normalized: $K(x, x) = 1$ for all x .

22.60.1 Detailed Description

```
template<> class mlpack::kernel::KernelTraits< SphericalKernel >
```

Kernel traits for the spherical kernel.

Definition at line 106 of file spherical_kernel.hpp.

22.60.2 Member Data Documentation

22.60.2.1 const bool mlpack::kernel::KernelTraits< SphericalKernel >::IsNormalized = true [static]

The spherical kernel is normalized: $K(x, x) = 1$ for all x .

Definition at line 110 of file spherical_kernel.hpp.

The documentation for this class was generated from the following file:

- src/mlpack/core/kernels/spherical_kernel.hpp

22.61 mlpack::kernel::KernelTraits< TriangularKernel > Class Template Reference

Kernel traits for the triangular kernel.

Static Public Attributes

- static const bool **IsNormalized** = true

The triangular kernel is normalized: $K(x, x) = 1$ for all x .

22.61.1 Detailed Description

```
template<> class mlpack::kernel::KernelTraits< TriangularKernel >
```

Kernel traits for the triangular kernel.

Definition at line 87 of file triangular_kernel.hpp.

22.61.2 Member Data Documentation

22.61.2.1 `const bool mlpack::kernel::KernelTraits< TriangularKernel >::IsNormalized = true` `[static]`

The triangular kernel is normalized: $K(x, x) = 1$ for all x .

Definition at line 91 of file `triangular_kernel.hpp`.

The documentation for this class was generated from the following file:

- `src/mlpack/core/kernels/triangular_kernel.hpp`

22.62 mlpack::kernel::KMeansSelection< ClusteringType > Class Template Reference

Static Public Member Functions

- static `const arma::mat * Select` (`const arma::mat &data`, `const size_t m`, `const size_t maxIterations=5`)

Use the K-Means clustering method to select the specified number of points in the dataset.

22.62.1 Detailed Description

`template<typename ClusteringType = kmeans::KMeans<>> class mlpack::kernel::KMeansSelection< ClusteringType >`

Definition at line 25 of file `kmeans_selection.hpp`.

22.62.2 Member Function Documentation

22.62.2.1 `template<typename ClusteringType = kmeans::KMeans<>> static const arma::mat* mlpack::kernel::KMeansSelection< ClusteringType >::Select (const arma::mat & data, const size_t m, const size_t maxIterations = 5)`
`[inline], [static]`

Use the K-Means clustering method to select the specified number of points in the dataset.

You are responsible for deleting the returned matrix!

Parameters

<i>data</i>	Dataset to sample from.
<i>m</i>	Number of points to select.

Returns

Matrix pointer in which centroids are stored.

Definition at line 36 of file `kmeans_selection.hpp`.

The documentation for this class was generated from the following file:

- `src/mlpack/methods/nystroem_method/kmeans_selection.hpp`

22.63 mpack::kernel::LaplacianKernel Class Reference

The standard Laplacian kernel.

Public Member Functions

- **LaplacianKernel** ()
Default constructor; sets bandwidth to 1.0.
- **LaplacianKernel** (double **bandwidth**)
Construct the Laplacian kernel with a custom bandwidth.
- double **Bandwidth** () const
Get the bandwidth.
- double & **Bandwidth** ()
Modify the bandwidth.
- template<typename VecType >
double **Evaluate** (const VecType &a, const VecType &b) const
Evaluation of the Laplacian kernel.
- double **Evaluate** (const double t) const
Evaluation of the Laplacian kernel given the distance between two points.
- std::string **ToString** () const
Return a string representation of the kernel.

Private Attributes

- double **bandwidth**
Kernel bandwidth.

22.63.1 Detailed Description

The standard Laplacian kernel.

Given two vectors x , y , and a bandwidth μ (set in the constructor),

$$K(x, y) = \exp\left(-\frac{\|x - y\|}{\mu}\right).$$

The implementation is all in the header file because it is so simple.

Definition at line 32 of file laplacian_kernel.hpp.

22.63.2 Constructor & Destructor Documentation

22.63.2.1 mpack::kernel::LaplacianKernel::LaplacianKernel () [inline]

Default constructor; sets bandwidth to 1.0.

Definition at line 38 of file laplacian_kernel.hpp.

22.63.2.2 mlpack::kernel::LaplacianKernel::LaplacianKernel (double *bandwidth*) [inline]

Construct the Laplacian kernel with a custom bandwidth.

Parameters

<i>bandwidth</i>	The bandwidth of the kernel (μ).
------------------	--

Definition at line 46 of file laplacian_kernel.hpp.

22.63.3 Member Function Documentation

22.63.3.1 `double mlpack::kernel::LaplacianKernel::Bandwidth () const [inline]`

Get the bandwidth.

Definition at line 83 of file laplacian_kernel.hpp.

References bandwidth.

22.63.3.2 `double& mlpack::kernel::LaplacianKernel::Bandwidth () [inline]`

Modify the bandwidth.

Definition at line 85 of file laplacian_kernel.hpp.

References bandwidth.

22.63.3.3 `template<typename VecType > double mlpack::kernel::LaplacianKernel::Evaluate (const VecType & a, const VecType & b) const [inline]`

Evaluation of the Laplacian kernel.

This could be generalized to use any distance metric, not the Euclidean distance, but for now, the Euclidean distance is used.

Template Parameters

<i>VecType</i>	Type of vector (likely arma::vec or arma::spvec).
----------------	---

Parameters

<i>a</i>	First vector.
<i>b</i>	Second vector.

Returns

$K(a, b)$ using the bandwidth (μ) specified in the constructor.

Definition at line 62 of file laplacian_kernel.hpp.

References bandwidth, and mlpack::metric::LMetric< Power, TakeRoot >::Evaluate().

22.63.3.4 `double mlpack::kernel::LaplacianKernel::Evaluate (const double t) const [inline]`

Evaluation of the Laplacian kernel given the distance between two points.

Parameters

t	The distance between the two points the kernel should be evaluated on.
-----	--

Returns

$K(t)$ using the bandwidth (μ) specified in the constructor.

Definition at line 76 of file laplacian_kernel.hpp.

References bandwidth.

22.63.3.5 std::string mlpack::kernel::LaplacianKernel::ToString () const [inline]

Return a string representation of the kernel.

Definition at line 88 of file laplacian_kernel.hpp.

References bandwidth.

22.63.4 Member Data Documentation

22.63.4.1 double mlpack::kernel::LaplacianKernel::bandwidth [private]

Kernel bandwidth.

Definition at line 98 of file laplacian_kernel.hpp.

Referenced by Bandwidth(), Evaluate(), and ToString().

The documentation for this class was generated from the following file:

- src/mlpack/core/kernels/laplacian_kernel.hpp

22.64 mlpack::kernel::LinearKernel Class Reference

The simple linear kernel (dot product).

Public Member Functions

- **LinearKernel** ()
This constructor does nothing; the linear kernel has no parameters to store.
- std::string **ToString** () const
Return a string representation of the kernel.

Static Public Member Functions

- template<typename VecType >
static double **Evaluate** (const VecType &a, const VecType &b)
Simple evaluation of the dot product.

22.64.1 Detailed Description

The simple linear kernel (dot product).

For any two vectors x and y ,

$$K(x, y) = x^T y$$

This kernel has no parameters and therefore the evaluation can be static.

Definition at line 34 of file linear_kernel.hpp.

22.64.2 Constructor & Destructor Documentation

22.64.2.1 `mlpack::kernel::LinearKernel::LinearKernel () [inline]`

This constructor does nothing; the linear kernel has no parameters to store.

Definition at line 41 of file linear_kernel.hpp.

22.64.3 Member Function Documentation

22.64.3.1 `template<typename VecType > static double mlpack::kernel::LinearKernel::Evaluate (const VecType & a, const VecType & b) [inline], [static]`

Simple evaluation of the dot product.

This evaluation uses Armadillo's `dot()` function.

Template Parameters

<i>VecType</i>	Type of vector (should be <code>arma::vec</code> or <code>arma::spvec</code>).
----------------	---

Parameters

<i>a</i>	First vector.
<i>b</i>	Second vector.

Returns

$K(a, b)$.

Definition at line 53 of file linear_kernel.hpp.

22.64.3.2 `std::string mlpack::kernel::LinearKernel::ToString () const [inline]`

Return a string representation of the kernel.

Definition at line 59 of file linear_kernel.hpp.

The documentation for this class was generated from the following file:

- `src/mlpack/core/kernels/linear_kernel.hpp`

22.65 mlpack::kernel::NystroemMethod< KernelType, PointSelectionPolicy > Class Template Reference

Public Member Functions

- **NystroemMethod** (const arma::mat &**data**, KernelType &**kernel**, const size_t **rank**)
Create the **NystroemMethod** (p. 317) object.
- void **Apply** (arma::mat &output)
Apply the low-rank factorization to obtain an output matrix G such that $K' = G * G^T$.
- void **GetKernelMatrix** (const arma::mat &**data**, arma::mat &miniKernel, arma::mat &semiKernel)
Construct the kernel matrix with matrix that contains the selected points.
- void **GetKernelMatrix** (const arma::Col< size_t > &selectedPoints, arma::mat &miniKernel, arma::mat &semiKernel)
Construct the kernel matrix with the selected points.

Private Attributes

- const arma::mat & **data**
The reference dataset.
- KernelType & **kernel**
The locally stored kernel, if it is necessary.
- const size_t **rank**
Rank used for matrix approximation.

22.65.1 Detailed Description

```
template<typename KernelType, typename PointSelectionPolicy = KMeansSelection<>>class mlpack::kernel::NystroemMethod<
KernelType, PointSelectionPolicy >
```

Definition at line 30 of file nystroem_method.hpp.

22.65.2 Constructor & Destructor Documentation

22.65.2.1 `template<typename KernelType, typename PointSelectionPolicy = KMeansSelection<>> mlpack::kernel::NystroemMethod< KernelType, PointSelectionPolicy >::NystroemMethod (const arma::mat & data, KernelType & kernel, const size_t rank)`

Create the **NystroemMethod** (p. 317) object.

The constructor here does not really do anything.

Parameters

<i>data</i>	Data matrix.
<i>kernel</i>	Kernel to be used for computation.

<i>rank</i>	Rank to be used for matrix approximation.
-------------	---

22.65.3 Member Function Documentation

22.65.3.1 `template<typename KernelType, typename PointSelectionPolicy = KMeansSelection<>> void
mlpack::kernel::NystroemMethod< KernelType, PointSelectionPolicy >::Apply (arma::mat & output)`

Apply the low-rank factorization to obtain an output matrix G such that $K' = G * G^T$.

Parameters

<i>output</i>	Matrix to store kernel approximation into.
---------------	--

Referenced by `mlpack::kpca::NystroemKernelRule< KernelType, PointSelectionPolicy >::ApplyKernelMatrix()`.

22.65.3.2 `template<typename KernelType, typename PointSelectionPolicy = KMeansSelection<>> void
mlpack::kernel::NystroemMethod< KernelType, PointSelectionPolicy >::GetKernelMatrix (const arma::mat * data,
arma::mat & miniKernel, arma::mat & semiKernel)`

Construct the kernel matrix with matrix that contains the selected points.

Parameters

<i>data</i>	Data matrix pointer.
<i>miniKernel</i>	to store the constructed mini-kernel matrix in.
<i>miniKernel</i>	to store the constructed semi-kernel matrix in.

22.65.3.3 `template<typename KernelType, typename PointSelectionPolicy = KMeansSelection<>> void
mlpack::kernel::NystroemMethod< KernelType, PointSelectionPolicy >::GetKernelMatrix (const arma::Col<
size_t> & selectedPoints, arma::mat & miniKernel, arma::mat & semiKernel)`

Construct the kernel matrix with the selected points.

Parameters

<i>points</i>	Indices of selected points.
<i>miniKernel</i>	to store the constructed mini-kernel matrix in.
<i>miniKernel</i>	to store the constructed semi-kernel matrix in.

22.65.4 Member Data Documentation

22.65.4.1 `template<typename KernelType, typename PointSelectionPolicy = KMeansSelection<>> const arma::mat&
mlpack::kernel::NystroemMethod< KernelType, PointSelectionPolicy >::data [private]`

The reference dataset.

Definition at line 75 of file `nystroem_method.hpp`.

22.65.4.2 `template<typename KernelType, typename PointSelectionPolicy = KMeansSelection<>> KernelType&
mlpack::kernel::NystroemMethod< KernelType, PointSelectionPolicy >::kernel [private]`

The locally stored kernel, if it is necessary.

Definition at line 77 of file nystroem_method.hpp.

22.65.4.3 `template<typename KernelType, typename PointSelectionPolicy = KMeansSelection<>> const size_t
mlpack::kernel::NystroemMethod< KernelType, PointSelectionPolicy >::rank [private]`

Rank used for matrix approximation.

Definition at line 79 of file nystroem_method.hpp.

The documentation for this class was generated from the following file:

- src/mlpack/methods/nystroem_method/nystroem_method.hpp

22.66 mlpack::kernel::OrderedSelection Class Reference

Static Public Member Functions

- static const arma::Col< size_t > **Select** (const arma::mat &, const size_t m)
Select the specified number of points in the dataset.

22.66.1 Detailed Description

Definition at line 24 of file ordered_selection.hpp.

22.66.2 Member Function Documentation

22.66.2.1 `static const arma::Col<size_t> mlpack::kernel::OrderedSelection::Select (const arma::mat & , const size_t m)
[inline],[static]`

Select the specified number of points in the dataset.

Parameters

<i>data</i>	Dataset to sample from.
<i>m</i>	Number of points to select.

Returns

Indices of selected points from the dataset.

Definition at line 34 of file ordered_selection.hpp.

The documentation for this class was generated from the following file:

- src/mlpack/methods/nystroem_method/ordered_selection.hpp

22.67 mlpack::kernel::PolynomialKernel Class Reference

The simple polynomial kernel.

Public Member Functions

- **PolynomialKernel** (const double **degree**=2.0, const double **offset**=0.0)
Construct the Polynomial Kernel with the given offset and degree.
- const double & **Degree** () const
Get the degree of the polynomial.
- double & **Degree** ()
Modify the degree of the polynomial.
- template<typename VecType >
double **Evaluate** (const VecType &a, const VecType &b) const
Simple evaluation of the dot product.
- const double & **Offset** () const
Get the offset of the dot product of the arguments.
- double & **Offset** ()
Modify the offset of the dot product of the arguments.
- std::string **ToString** () const
Return a string representation of the kernel.

Private Attributes

- double **degree**
The degree of the polynomial.
- double **offset**
The offset of the dot product of the arguments.

22.67.1 Detailed Description

The simple polynomial kernel.

For any two vectors x , y , $degree$ and $offset$,

$$K(x, y) = (x^T * y + offset)^{degree}.$$

Definition at line 30 of file polynomial_kernel.hpp.

22.67.2 Constructor & Destructor Documentation

22.67.2.1 `mlpack::kernel::PolynomialKernel::PolynomialKernel (const double degree = 2.0, const double offset = 0.0)`
`[inline]`

Construct the Polynomial Kernel with the given offset and degree.

If the arguments are omitted, the default degree is 2 and the default offset is 0.

Parameters

<i>offset</i>	Offset of the dot product of the arguments.
<i>degree</i>	Degree of the polynomial.

Definition at line 40 of file polynomial_kernel.hpp.

22.67.3 Member Function Documentation

22.67.3.1 `const double& mlpack::kernel::PolynomialKernel::Degree () const` `[inline]`

Get the degree of the polynomial.

Definition at line 61 of file polynomial_kernel.hpp.

References degree.

22.67.3.2 `double& mlpack::kernel::PolynomialKernel::Degree ()` `[inline]`

Modify the degree of the polynomial.

Definition at line 63 of file polynomial_kernel.hpp.

References degree.

22.67.3.3 `template<typename VecType> double mlpack::kernel::PolynomialKernel::Evaluate (const VecType & a, const VecType & b) const` `[inline]`

Simple evaluation of the dot product.

This evaluation uses Armadillo's dot() function.

Template Parameters

<i>VecType</i>	Type of vector (should be arma::vec or arma::spvec).
----------------	--

Parameters

<i>a</i>	First vector.
<i>b</i>	Second vector.

Returns

$K(a, b)$.

Definition at line 55 of file polynomial_kernel.hpp.

References degree, and offset.

22.67.3.4 `const double& mlpack::kernel::PolynomialKernel::Offset () const` `[inline]`

Get the offset of the dot product of the arguments.

Definition at line 66 of file polynomial_kernel.hpp.

References offset.

22.67.3.5 `double& mlpack::kernel::PolynomialKernel::Offset () [inline]`

Modify the offset of the dot product of the arguments.

Definition at line 68 of file `polynomial_kernel.hpp`.

References `offset`.

22.67.3.6 `std::string mlpack::kernel::PolynomialKernel::ToString () const [inline]`

Return a string representation of the kernel.

Definition at line 71 of file `polynomial_kernel.hpp`.

References `degree`, and `offset`.

22.67.4 Member Data Documentation

22.67.4.1 `double mlpack::kernel::PolynomialKernel::degree [private]`

The degree of the polynomial.

Definition at line 82 of file `polynomial_kernel.hpp`.

Referenced by `Degree()`, `Evaluate()`, and `ToString()`.

22.67.4.2 `double mlpack::kernel::PolynomialKernel::offset [private]`

The offset of the dot product of the arguments.

Definition at line 84 of file `polynomial_kernel.hpp`.

Referenced by `Evaluate()`, `Offset()`, and `ToString()`.

The documentation for this class was generated from the following file:

- `src/mlpack/core/kernels/polynomial_kernel.hpp`

22.68 mlpack::kernel::PSpectrumStringKernel Class Reference

The p-spectrum string kernel.

Public Member Functions

- **PSpectrumStringKernel** (const std::vector< std::vector< std::string > > &datasets, const size_t p)
*Initialize the **PSpectrumStringKernel** (p. 322) with the given string datasets.*
- const std::vector< std::vector< std::map< std::string, int > > > & **Counts** () const
Access the lists of substrings.
- std::vector< std::vector< std::map< std::string, int > > > & **Counts** ()
Modify the lists of substrings.
- template<typename VecType >
double **Evaluate** (const VecType &a, const VecType &b) const

Evaluate the kernel for the string indices given.

- `size_t P () const`

Access the value of p.

- `size_t & P ()`

Modify the value of p.

- `std::string ToString () const`

Private Attributes

- `std::vector< std::vector< std::map< std::string, int > > > counts`

Mappings of the datasets to counts of substrings.

- `const std::vector< std::vector< std::string > > & datasets`

The datasets.

- `size_t p`

The value of p to use in calculation.

22.68.1 Detailed Description

The p-spectrum string kernel.

Given a length p, the p-spectrum kernel finds the contiguous subsequence match count between two strings. The kernel will take every possible substring of length p of one string and count how many times it appears in the other string.

The string kernel, when created, must be passed a reference to a series of string datasets (`std::vector<std::vector<std::string> >&`). This is because MLPACK only supports datasets which are Armadillo matrices – and a dataset of variable-length strings cannot be easily cast into an Armadillo matrix.

Therefore, once the **PSpectrumStringKernel** (p. 322) is created with a reference to the string datasets, a "fake" Armadillo data matrix must be created, which simply holds indices to the strings they represent. This "fake" matrix has two rows and n columns (where n is the number of strings in the dataset). The first row holds the index of the dataset (remember, the kernel can have multiple datasets), and the second row holds the index of the string. A fake matrix containing only strings from dataset 0 might look like this:

```
[[0 0 0 0 0 0 0 0] [0 1 2 3 4 5 6 7 8]]
```

This fake matrix is then given to the machine learning method, which will eventually call `PSpectrumStringKernel::Evaluate(a, b)`, where a and b are two columns of the fake matrix. The string kernel will then map these fake columns back to the strings they represent, and then correctly evaluate the kernel.

Unfortunately, not every machine learning method will work with this kernel. Only machine learning methods which do not ever operate on the explicit representation of points can use this kernel. So, for instance, one cannot build a kd-tree on strings, because the `BinarySpaceTree<>` class will split the data according to the fake data matrix – resulting in a meaningless tree. This kernel was originally written for the FastMKS method; so, at the very least, it will work with that.

Definition at line 66 of file `pspectrum_string_kernel.hpp`.

22.68.2 Constructor & Destructor Documentation

- 22.68.2.1** `mlpack::kernel::PSpectrumStringKernel::PSpectrumStringKernel (const std::vector< std::vector< std::string > > & datasets, const size_t p)`

Initialize the **PSpectrumStringKernel** (p. 322) with the given string datasets.

For more information on this, see the general class documentation.

Parameters

<i>datasets</i>	Sets of string data.
<i>p</i>	The length of substrings to search.

22.68.3 Member Function Documentation

22.68.3.1 `const std::vector<std::vector<std::map<std::string, int> > >& mlpack::kernel::PSpectrumStringKernel::Counts ()`
`const [inline]`

Access the lists of substrings.

Definition at line 94 of file `pspectrum_string_kernel.hpp`.

References counts.

22.68.3.2 `std::vector<std::vector<std::map<std::string, int> > >& mlpack::kernel::PSpectrumStringKernel::Counts ()`
`[inline]`

Modify the lists of substrings.

Definition at line 97 of file `pspectrum_string_kernel.hpp`.

References counts.

22.68.3.3 `template<typename VecType > double mlpack::kernel::PSpectrumStringKernel::Evaluate (const VecType & a, const VecType & b) const`

Evaluate the kernel for the string indices given.

As mentioned in the class documentation, *a* and *b* should be 2-element vectors, where the first element contains the index of the dataset and the second element contains the index of the string. Therefore, if [2 3] is passed for *a*, the string used will be `datasets[2][3]` (`datasets` is of type `std::vector<std::vector<std::string> >&`).

Parameters

<i>a</i>	Index of string and dataset for first string.
<i>b</i>	Index of string and dataset for second string.

22.68.3.4 `size_t mlpack::kernel::PSpectrumStringKernel::P () const [inline]`

Access the value of *p*.

Definition at line 101 of file `pspectrum_string_kernel.hpp`.

References *p*.

22.68.3.5 `size_t& mlpack::kernel::PSpectrumStringKernel::P () [inline]`

Modify the value of *p*.

Definition at line 103 of file `pspectrum_string_kernel.hpp`.

References *p*.

22.68.3.6 `std::string mlpack::kernel::PSpectrumStringKernel::ToString () const` `[inline]`

Definition at line 108 of file `pspectrum_string_kernel.hpp`.

References `datasets`, and `mlpack::util::Indent()`.

22.68.4 Member Data Documentation

22.68.4.1 `std::vector<std::vector<std::map<std::string, int> > > mlpack::kernel::PSpectrumStringKernel::counts` `[private]`

Mappings of the datasets to counts of substrings.

Such a huge structure is not wonderful...

Definition at line 125 of file `pspectrum_string_kernel.hpp`.

Referenced by `Counts()`.

22.68.4.2 `const std::vector<std::vector<std::string> > & mlpack::kernel::PSpectrumStringKernel::datasets` `[private]`

The datasets.

Definition at line 121 of file `pspectrum_string_kernel.hpp`.

Referenced by `ToString()`.

22.68.4.3 `size_t mlpack::kernel::PSpectrumStringKernel::p` `[private]`

The value of `p` to use in calculation.

Definition at line 128 of file `pspectrum_string_kernel.hpp`.

Referenced by `P()`.

The documentation for this class was generated from the following file:

- `src/mlpack/core/kernels/pspectrum_string_kernel.hpp`

22.69 mlpack::kernel::RandomSelection Class Reference

Static Public Member Functions

- static const `arma::Col< size_t > Select` (const `arma::mat &data`, const `size_t m`)

Randomly select the specified number of points in the dataset.

22.69.1 Detailed Description

Definition at line 24 of file `random_selection.hpp`.

22.69.2 Member Function Documentation

22.69.2.1 `static const arma::Col<size_t> mlpack::kernel::RandomSelection::Select (const arma::mat & data, const size_t m)`
`[inline],[static]`

Randomly select the specified number of points in the dataset.

Parameters

<i>data</i>	Dataset to sample from.
<i>m</i>	Number of points to select.

Returns

Indices of selected points from the dataset.

Definition at line 34 of file random_selection.hpp.

References mlpack::math::RandInt().

The documentation for this class was generated from the following file:

- src/mlpack/methods/nystroem_method/random_selection.hpp

22.70 mlpack::kernel::SphericalKernel Class Reference

Public Member Functions

- **SphericalKernel** ()
- **SphericalKernel** (double b)
- template<typename VecType >
double **ConvolutionIntegral** (const VecType &a, const VecType &b)
Obtains the convolution integral $\int K(\|x-a\|)K(\|b-x\|)dx$ for the two vectors.
- template<typename VecType >
double **Evaluate** (const VecType &a, const VecType &b)
- double **Evaluate** (double t)
- double **Normalizer** (size_t dimension)
- std::string **ToString** () const
Return a string representation of the kernel.

Private Attributes

- double **bandwidth**
- double **bandwidthSquared**

22.70.1 Detailed Description

Definition at line 24 of file spherical_kernel.hpp.

22.70.2 Constructor & Destructor Documentation

22.70.2.1 mlpack::kernel::SphericalKernel::SphericalKernel () [inline]

Definition at line 27 of file spherical_kernel.hpp.

22.70.2.2 `mlpack::kernel::SphericalKernel::SphericalKernel (double b) [inline]`

Definition at line 30 of file `spherical_kernel.hpp`.

22.70.3 Member Function Documentation

22.70.3.1 `template<typename VecType > double mlpack::kernel::SphericalKernel::ConvolutionIntegral (const VecType & a, const VecType & b) [inline]`

Obtains the convolution integral $[\int K(\|x-a\|)K(\|b-x\|)dx]$ for the two vectors.

In this case, because our simple example kernel has no internal parameters, we can declare the function static. For a more complex example which cannot be declared static, see the **GaussianKernel** (p. 299), which stores an internal parameter.

Template Parameters

<i>VecType</i>	Type of vector (arma::vec, arma::spvec should be expected).
----------------	---

Parameters

<i>a</i>	First vector.
<i>b</i>	Second vector.

Returns

the convolution integral value.

Definition at line 54 of file `spherical_kernel.hpp`.

References `bandwidth`, `mlpack::metric::LMetric< Power, TakeRoot >::Evaluate()`, `mlpack::Log::Fatal`, and `Normalizer()`.

22.70.3.2 `template<typename VecType > double mlpack::kernel::SphericalKernel::Evaluate (const VecType & a, const VecType & b) [inline]`

Definition at line 35 of file `spherical_kernel.hpp`.

References `bandwidthSquared`, and `mlpack::metric::LMetric< Power, TakeRoot >::Evaluate()`.

22.70.3.3 `double mlpack::kernel::SphericalKernel::Evaluate (double t) [inline]`

Definition at line 85 of file `spherical_kernel.hpp`.

References `bandwidth`.

22.70.3.4 `double mlpack::kernel::SphericalKernel::Normalizer (size_t dimension) [inline]`

Definition at line 80 of file `spherical_kernel.hpp`.

References `bandwidth`, and `M_PI`.

Referenced by `ConvolutionIntegral()`.

22.70.3.5 `std::string mlpack::kernel::SphericalKernel::ToString () const [inline]`

Return a string representation of the kernel.

Definition at line 91 of file `spherical_kernel.hpp`.

References `bandwidth`.

22.70.4 Member Data Documentation

22.70.4.1 `double mlpack::kernel::SphericalKernel::bandwidth [private]`

Definition at line 100 of file `spherical_kernel.hpp`.

Referenced by `ConvolutionIntegral()`, `Evaluate()`, `Normalizer()`, and `ToString()`.

22.70.4.2 `double mlpack::kernel::SphericalKernel::bandwidthSquared [private]`

Definition at line 101 of file `spherical_kernel.hpp`.

Referenced by `Evaluate()`.

The documentation for this class was generated from the following file:

- `src/mlpack/core/kernels/spherical_kernel.hpp`

22.71 mlpack::kernel::TriangularKernel Class Reference

The trivially simple triangular kernel, defined by.

Public Member Functions

- **TriangularKernel** (const double **bandwidth**=1.0)
Initialize the triangular kernel with the given bandwidth (default 1.0).
- double **Bandwidth** () const
Get the bandwidth of the kernel.
- double & **Bandwidth** ()
Modify the bandwidth of the kernel.
- template<typename Vec1Type , typename Vec2Type >
double **Evaluate** (const Vec1Type &a, const Vec2Type &b) const
Evaluate the triangular kernel for the two given vectors.
- double **Evaluate** (const double distance) const
Evaluate the triangular kernel given that the distance between the two points is known.
- std::string **ToString** () const
Return a string representation of the kernel.

Private Attributes

- double **bandwidth**
The bandwidth of the kernel.

22.71.1 Detailed Description

The trivially simple triangular kernel, defined by.

$$K(x, y) = \max\{0, 1 - \frac{\|x - y\|_2}{b}\}$$

where b is the bandwidth of the kernel.

Definition at line 32 of file triangular_kernel.hpp.

22.71.2 Constructor & Destructor Documentation

22.71.2.1 `mlpack::kernel::TriangularKernel (const double bandwidth = 1.0) [inline]`

Initialize the triangular kernel with the given bandwidth (default 1.0).

Parameters

<i>bandwidth</i>	Bandwidth of the triangular kernel.
------------------	-------------------------------------

Definition at line 40 of file triangular_kernel.hpp.

22.71.3 Member Function Documentation

22.71.3.1 `double mlpack::kernel::TriangularKernel::Bandwidth () const [inline]`

Get the bandwidth of the kernel.

Definition at line 67 of file triangular_kernel.hpp.

References bandwidth.

22.71.3.2 `double& mlpack::kernel::TriangularKernel::Bandwidth () [inline]`

Modify the bandwidth of the kernel.

Definition at line 69 of file triangular_kernel.hpp.

References bandwidth.

22.71.3.3 `template<typename Vec1Type , typename Vec2Type > double mlpack::kernel::TriangularKernel::Evaluate (const Vec1Type & a, const Vec2Type & b) const [inline]`

Evaluate the triangular kernel for the two given vectors.

Parameters

<i>a</i>	First vector.
<i>b</i>	Second vector.

Definition at line 49 of file triangular_kernel.hpp.

References bandwidth, and `mlpack::metric::LMetric< Power, TakeRoot >::Evaluate()`.

22.71.3.4 `double mpack::kernel::TriangularKernel::Evaluate (const double distance) const` `[inline]`

Evaluate the triangular kernel given that the distance between the two points is known.

Parameters

<i>distance</i>	The distance between the two points.
-----------------	--------------------------------------

Definition at line 61 of file `triangular_kernel.hpp`.

References `bandwidth`.

22.71.3.5 `std::string mlpack::kernel::TriangularKernel::ToString () const` `[inline]`

Return a string representation of the kernel.

Definition at line 72 of file `triangular_kernel.hpp`.

References `bandwidth`.

22.71.4 Member Data Documentation

22.71.4.1 `double mlpack::kernel::TriangularKernel::bandwidth` `[private]`

The bandwidth of the kernel.

Definition at line 82 of file `triangular_kernel.hpp`.

Referenced by `Bandwidth()`, `Evaluate()`, and `ToString()`.

The documentation for this class was generated from the following file:

- `src/mlpack/core/kernels/triangular_kernel.hpp`

22.72 mlpack::kmeans::AllowEmptyClusters Class Reference

Policy which allows K-Means to create empty clusters without any error being reported.

Public Member Functions

- **AllowEmptyClusters ()**

Default constructor required by EmptyClusterPolicy policy.

Static Public Member Functions

- `template<typename MatType >`
`static size_t EmptyCluster (const MatType &, const size_t, const MatType &, arma::Col< size_t > &, arma::Col< size_t > &)`

This function does nothing.

22.72.1 Detailed Description

Policy which allows K-Means to create empty clusters without any error being reported.

Definition at line 27 of file `allow_empty_clusters.hpp`.

22.72.2 Constructor & Destructor Documentation

22.72.2.1 mlpack::kmeans::AllowEmptyClusters::AllowEmptyClusters () [inline]

Default constructor required by EmptyClusterPolicy policy.

Definition at line 31 of file allow_empty_clusters.hpp.

22.72.3 Member Function Documentation

22.72.3.1 template<typename MatType > static size_t mlpack::kmeans::AllowEmptyClusters::EmptyCluster (const MatType & , const size_t , const MatType & , arma::Col< size_t > & , arma::Col< size_t > &) [inline],[static]

This function does nothing.

It is called by K-Means when K-Means detects an empty cluster.

Template Parameters

<i>MatType</i>	Type of data (arma::mat or arma::spmat).
----------------	--

Parameters

<i>data</i>	Dataset on which clustering is being performed.
<i>emptyCluster</i>	Index of cluster which is empty.
<i>centroids</i>	Centroids of each cluster (one per column).
<i>clusterCounts</i>	Number of points in each cluster.
<i>assignments</i>	Cluster assignments of each point.

Returns

Number of points changed (0).

Definition at line 47 of file allow_empty_clusters.hpp.

The documentation for this class was generated from the following file:

- src/mlpack/methods/kmeans/allow_empty_clusters.hpp

22.73 mlpack::kmeans::KMeans< MetricType, InitialPartitionPolicy, EmptyClusterPolicy > Class Template Reference

This class implements K-Means clustering.

Public Member Functions

- **KMeans** (const size_t **maxIterations**=1000, const double **overclusteringFactor**=1.0, const MetricType **metric**=MetricType(), const InitialPartitionPolicy **partitioner**=InitialPartitionPolicy(), const EmptyClusterPolicy **emptyClusterAction**=EmptyClusterPolicy())

Create a K-Means object and (optionally) set the parameters which K-Means will be run with.

- template<typename MatType >
void **Cluster** (const MatType &data, const size_t clusters, arma::Col< size_t > &assignments, const bool initial←Guess=false) const

Perform k-means clustering on the data, returning a list of cluster assignments.

- `template<typename MatType >
void Cluster (const MatType &data, const size_t clusters, arma::Col< size_t > &assignments, MatType ¢roids, const bool initialAssignmentGuess=false, const bool initialCentroidGuess=false) const`

Perform k-means clustering on the data, returning a list of cluster assignments and also the centroids of each cluster.

- `const EmptyClusterPolicy & EmptyClusterAction () const`

Get the empty cluster policy.

- `EmptyClusterPolicy & EmptyClusterAction ()`

Modify the empty cluster policy.

- `size_t MaxIterations () const`

Get the maximum number of iterations.

- `size_t & MaxIterations ()`

Set the maximum number of iterations.

- `const MetricType & Metric () const`

Get the distance metric.

- `MetricType & Metric ()`

Modify the distance metric.

- `double OverclusteringFactor () const`

Return the overclustering factor.

- `double & OverclusteringFactor ()`

Set the overclustering factor. Must be greater than 1.

- `const InitialPartitionPolicy & Partitioner () const`

Get the initial partitioning policy.

- `InitialPartitionPolicy & Partitioner ()`

Modify the initial partitioning policy.

- `std::string ToString () const`

Private Attributes

- EmptyClusterPolicy **emptyClusterAction**

Instantiated empty cluster policy.

- size_t **maxIterations**

Maximum number of iterations before giving up.

- MetricType **metric**

Instantiated distance metric.

- double **overclusteringFactor**

Factor controlling how many clusters are actually found.

- InitialPartitionPolicy **partitioner**

Instantiated initial partitioning policy.

22.73.1 Detailed Description

```
template<typename MetricType = metric::SquaredEuclideanDistance, typename InitialPartitionPolicy = RandomPartition, typename
EmptyClusterPolicy = MaxVarianceNewCluster> class mlpack::kmeans::KMeans< MetricType, InitialPartitionPolicy, EmptyClusterPolicy >
```

This class implements K-Means clustering.

This implementation supports overclustering, which means that more clusters than are requested will be found; then, those clusters will be merged together to produce the desired number of clusters.

Two template parameters can (optionally) be supplied: the policy for how to find the initial partition of the data, and the actions to be taken when an empty cluster is encountered, as well as the distance metric to be used.

A simple example of how to run K-Means clustering is shown below.

```
extern arma::mat data; // Dataset we want to run K-Means on.
arma::Col<size_t> assignments; // Cluster assignments.

KMeans<> k; // Default options.
k.Cluster(data, 3, assignments); // 3 clusters.

// Cluster using the Manhattan distance, 100 iterations maximum, and an
// overclustering factor of 4.0.
KMeans<metric::ManhattanDistance> k(100, 4.0);
k.Cluster(data, 6, assignments); // 6 clusters.
```

Template Parameters

<i>MetricType</i>	The distance metric to use for this KMeans (p. 333); see metric::LMetric (p. 367) for an example.
<i>InitialPartitionPolicy</i>	Initial partitioning policy; must implement a default constructor and 'void Cluster(const arma::mat&, const size_t, arma::Col<size_t>&)'.
<i>EmptyClusterPolicy</i>	Policy for what to do on an empty cluster; must implement a default constructor and 'void EmptyCluster(const arma::mat&, arma::Col<size_t>&)'.

See also

RandomPartition (p. 342), **RefinedStart** (p. 343), **AllowEmptyClusters** (p. 332), **MaxVarianceNewCluster** (p. 340)

Definition at line 67 of file kmeans.hpp.

22.73.2 Constructor & Destructor Documentation

22.73.2.1 `template<typename MetricType = metric::SquaredEuclideanDistance, typename InitialPartitionPolicy = RandomPartition, typename EmptyClusterPolicy = MaxVarianceNewCluster> mlpack::kmeans::KMeans< MetricType, InitialPartitionPolicy, EmptyClusterPolicy >::KMeans (const size_t maxIterations = 1000, const double overclusteringFactor = 1.0, const MetricType metric = MetricType(), const InitialPartitionPolicy partitioner = InitialPartitionPolicy(), const EmptyClusterPolicy emptyClusterAction = EmptyClusterPolicy())`

Create a K-Means object and (optionally) set the parameters which K-Means will be run with.

This implementation allows a few strategies to improve the performance of K-Means, including "overclustering" and disallowing empty clusters.

The overclustering factor controls how many clusters are actually found; for instance, with an overclustering factor of 4, if K-Means is run to find 3 clusters, it will actually find 12, then merge the nearest clusters until only 3 are left.

Parameters

<i>maxIterations</i>	Maximum number of iterations allowed before giving up (0 is valid, but the algorithm may never terminate).
----------------------	--

<i>overclustering</i> ↔ <i>Factor</i>	Factor controlling how many extra clusters are found and then merged to get the desired number of clusters.
<i>metric</i>	Optional MetricType object; for when the metric has state it needs to store.
<i>partitioner</i>	Optional InitialPartitionPolicy object; for when a specially initialized partitioning policy is required.
<i>emptyCluster</i> ↔ <i>Action</i>	Optional EmptyClusterPolicy object; for when a specially initialized empty cluster policy is required.

22.73.3 Member Function Documentation

22.73.3.1 `template<typename MetricType = metric::SquaredEuclideanDistance, typename InitialPartitionPolicy = RandomPartition, typename EmptyClusterPolicy = MaxVarianceNewCluster> template<typename MatType > void mlpack::kmeans::KMeans< MetricType, InitialPartitionPolicy, EmptyClusterPolicy >::Cluster (const MatType & data, const size_t clusters, arma::Col< size_t > & assignments, const bool initialGuess = false) const`

Perform k-means clustering on the data, returning a list of cluster assignments.

Optionally, the vector of assignments can be set to an initial guess of the cluster assignments; to do this, set initialGuess to true.

Template Parameters

<i>MatType</i>	Type of matrix (arma::mat or arma::sp_mat).
----------------	---

Parameters

<i>data</i>	Dataset to cluster.
<i>clusters</i>	Number of clusters to compute.
<i>assignments</i>	Vector to store cluster assignments in.
<i>initialGuess</i>	If true, then it is assumed that assignments has a list of initial cluster assignments.

22.73.3.2 `template<typename MetricType = metric::SquaredEuclideanDistance, typename InitialPartitionPolicy = RandomPartition, typename EmptyClusterPolicy = MaxVarianceNewCluster> template<typename MatType > void mlpack::kmeans::KMeans< MetricType, InitialPartitionPolicy, EmptyClusterPolicy >::Cluster (const MatType & data, const size_t clusters, arma::Col< size_t > & assignments, MatType & centroids, const bool initialAssignmentGuess = false, const bool initialCentroidGuess = false) const`

Perform k-means clustering on the data, returning a list of cluster assignments and also the centroids of each cluster.

Optionally, the vector of assignments can be set to an initial guess of the cluster assignments; to do this, set initial↔AssignmentGuess to true. Another way to set initial cluster guesses is to fill the centroids matrix with the centroid guesses, and then set initialCentroidGuess to true. initialAssignmentGuess supersedes initialCentroidGuess, so if both are set to true, the assignments vector is used.

Note that if the overclustering factor is greater than 1, the centroids matrix will be resized in the method. Regardless of the overclustering factor, the centroid guess matrix (if initialCentroidGuess is set to true) should have the same number of rows as the data matrix, and number of columns equal to 'clusters'.

Template Parameters

<i>MatType</i>	Type of matrix (arma::mat or arma::sp_mat).
----------------	---

Parameters

<i>data</i>	Dataset to cluster.
<i>clusters</i>	Number of clusters to compute.
<i>assignments</i>	Vector to store cluster assignments in.
<i>centroids</i>	Matrix in which centroids are stored.
<i>initial</i> ↔ <i>Assignment</i> ↔ <i>Guess</i>	If true, then it is assumed that assignments has a list of initial cluster assignments.
<i>initialCentroid</i> ↔ <i>Guess</i>	If true, then it is assumed that centroids contains the initial centroids of each cluster.

22.73.3.3 `template<typename MetricType = metric::SquaredEuclideanDistance, typename InitialPartitionPolicy = RandomPartition, typename EmptyClusterPolicy = MaxVarianceNewCluster> const EmptyClusterPolicy& mlpack::kmeans::KMeans< MetricType, InitialPartitionPolicy, EmptyClusterPolicy >::EmptyClusterAction () const [inline]`

Get the empty cluster policy.

Definition at line 173 of file kmeans.hpp.

References mlpack::kmeans::KMeans< MetricType, InitialPartitionPolicy, EmptyClusterPolicy >::emptyClusterAction.

22.73.3.4 `template<typename MetricType = metric::SquaredEuclideanDistance, typename InitialPartitionPolicy = RandomPartition, typename EmptyClusterPolicy = MaxVarianceNewCluster> EmptyClusterPolicy& mlpack::kmeans::KMeans< MetricType, InitialPartitionPolicy, EmptyClusterPolicy >::EmptyClusterAction () [inline]`

Modify the empty cluster policy.

Definition at line 176 of file kmeans.hpp.

References mlpack::kmeans::KMeans< MetricType, InitialPartitionPolicy, EmptyClusterPolicy >::emptyClusterAction.

22.73.3.5 `template<typename MetricType = metric::SquaredEuclideanDistance, typename InitialPartitionPolicy = RandomPartition, typename EmptyClusterPolicy = MaxVarianceNewCluster> size_t mlpack::kmeans::KMeans< MetricType, InitialPartitionPolicy, EmptyClusterPolicy >::MaxIterations () const [inline]`

Get the maximum number of iterations.

Definition at line 158 of file kmeans.hpp.

References mlpack::kmeans::KMeans< MetricType, InitialPartitionPolicy, EmptyClusterPolicy >::maxIterations.

22.73.3.6 `template<typename MetricType = metric::SquaredEuclideanDistance, typename InitialPartitionPolicy = RandomPartition, typename EmptyClusterPolicy = MaxVarianceNewCluster> size_t& mlpack::kmeans::KMeans< MetricType, InitialPartitionPolicy, EmptyClusterPolicy >::MaxIterations () [inline]`

Set the maximum number of iterations.

Definition at line 160 of file kmeans.hpp.

References mlpack::kmeans::KMeans< MetricType, InitialPartitionPolicy, EmptyClusterPolicy >::maxIterations.

```
22.73.3.7 template<typename MetricType = metric::SquaredEuclideanDistance, typename InitialPartitionPolicy = RandomPartition,
typename EmptyClusterPolicy = MaxVarianceNewCluster> const MetricType& mlpack::kmeans::KMeans<
MetricType, InitialPartitionPolicy, EmptyClusterPolicy >::Metric ( ) const [inline]
```

Get the distance metric.

Definition at line 163 of file kmeans.hpp.

References mlpack::kmeans::KMeans< MetricType, InitialPartitionPolicy, EmptyClusterPolicy >::metric.

```
22.73.3.8 template<typename MetricType = metric::SquaredEuclideanDistance, typename InitialPartitionPolicy = RandomPartition,
typename EmptyClusterPolicy = MaxVarianceNewCluster> MetricType& mlpack::kmeans::KMeans< MetricType,
InitialPartitionPolicy, EmptyClusterPolicy >::Metric ( ) [inline]
```

Modify the distance metric.

Definition at line 165 of file kmeans.hpp.

References mlpack::kmeans::KMeans< MetricType, InitialPartitionPolicy, EmptyClusterPolicy >::metric.

```
22.73.3.9 template<typename MetricType = metric::SquaredEuclideanDistance, typename InitialPartitionPolicy = RandomPartition,
typename EmptyClusterPolicy = MaxVarianceNewCluster> double mlpack::kmeans::KMeans< MetricType,
InitialPartitionPolicy, EmptyClusterPolicy >::OverclusteringFactor ( ) const [inline]
```

Return the overclustering factor.

Definition at line 153 of file kmeans.hpp.

References mlpack::kmeans::KMeans< MetricType, InitialPartitionPolicy, EmptyClusterPolicy >::overclusteringFactor.

```
22.73.3.10 template<typename MetricType = metric::SquaredEuclideanDistance, typename InitialPartitionPolicy = RandomPartition,
typename EmptyClusterPolicy = MaxVarianceNewCluster> double& mlpack::kmeans::KMeans< MetricType,
InitialPartitionPolicy, EmptyClusterPolicy >::OverclusteringFactor ( ) [inline]
```

Set the overclustering factor. Must be greater than 1.

Definition at line 155 of file kmeans.hpp.

References mlpack::kmeans::KMeans< MetricType, InitialPartitionPolicy, EmptyClusterPolicy >::overclusteringFactor.

```
22.73.3.11 template<typename MetricType = metric::SquaredEuclideanDistance, typename InitialPartitionPolicy = RandomPartition,
typename EmptyClusterPolicy = MaxVarianceNewCluster> const InitialPartitionPolicy& mlpack::kmeans::KMeans<
MetricType, InitialPartitionPolicy, EmptyClusterPolicy >::Partitioner ( ) const [inline]
```

Get the initial partitioning policy.

Definition at line 168 of file kmeans.hpp.

References mlpack::kmeans::KMeans< MetricType, InitialPartitionPolicy, EmptyClusterPolicy >::partitioner.

```
22.73.3.12 template<typename MetricType = metric::SquaredEuclideanDistance, typename InitialPartitionPolicy = RandomPartition,
typename EmptyClusterPolicy = MaxVarianceNewCluster> InitialPartitionPolicy& mlpack::kmeans::KMeans<
MetricType, InitialPartitionPolicy, EmptyClusterPolicy >::Partitioner ( ) [inline]
```

Modify the initial partitioning policy.

Definition at line 170 of file kmeans.hpp.

References mlpack::kmeans::KMeans< MetricType, InitialPartitionPolicy, EmptyClusterPolicy >::partitioner.

```
22.73.3.13  template<typename MetricType = metric::SquaredEuclideanDistance, typename InitialPartitionPolicy = RandomPartition,
            typename EmptyClusterPolicy = MaxVarianceNewCluster> std::string mlpack::kmeans::KMeans< MetricType,
            InitialPartitionPolicy, EmptyClusterPolicy >::ToString ( ) const
```

22.73.4 Member Data Documentation

```
22.73.4.1  template<typename MetricType = metric::SquaredEuclideanDistance, typename InitialPartitionPolicy = RandomPartition,
            typename EmptyClusterPolicy = MaxVarianceNewCluster> EmptyClusterPolicy mlpack::kmeans::KMeans<
            MetricType, InitialPartitionPolicy, EmptyClusterPolicy >::emptyClusterAction  [private]
```

Instantiated empty cluster policy.

Definition at line 191 of file kmeans.hpp.

Referenced by mlpack::kmeans::KMeans< MetricType, InitialPartitionPolicy, EmptyClusterPolicy >::EmptyClusterAction().

```
22.73.4.2  template<typename MetricType = metric::SquaredEuclideanDistance, typename InitialPartitionPolicy = RandomPartition,
            typename EmptyClusterPolicy = MaxVarianceNewCluster> size_t mlpack::kmeans::KMeans< MetricType,
            InitialPartitionPolicy, EmptyClusterPolicy >::maxIterations  [private]
```

Maximum number of iterations before giving up.

Definition at line 185 of file kmeans.hpp.

Referenced by mlpack::kmeans::KMeans< MetricType, InitialPartitionPolicy, EmptyClusterPolicy >::MaxIterations().

```
22.73.4.3  template<typename MetricType = metric::SquaredEuclideanDistance, typename InitialPartitionPolicy = RandomPartition,
            typename EmptyClusterPolicy = MaxVarianceNewCluster> MetricType mlpack::kmeans::KMeans< MetricType,
            InitialPartitionPolicy, EmptyClusterPolicy >::metric  [private]
```

Instantiated distance metric.

Definition at line 187 of file kmeans.hpp.

Referenced by mlpack::kmeans::KMeans< MetricType, InitialPartitionPolicy, EmptyClusterPolicy >::Metric().

```
22.73.4.4  template<typename MetricType = metric::SquaredEuclideanDistance, typename InitialPartitionPolicy = RandomPartition,
            typename EmptyClusterPolicy = MaxVarianceNewCluster> double mlpack::kmeans::KMeans< MetricType,
            InitialPartitionPolicy, EmptyClusterPolicy >::overclusteringFactor  [private]
```

Factor controlling how many clusters are actually found.

Definition at line 183 of file kmeans.hpp.

Referenced by mlpack::kmeans::KMeans< MetricType, InitialPartitionPolicy, EmptyClusterPolicy >::OverclusteringFactor().

22.73.4.5 `template<typename MetricType = metric::SquaredEuclideanDistance, typename InitialPartitionPolicy = RandomPartition, typename EmptyClusterPolicy = MaxVarianceNewCluster> InitialPartitionPolicy mlpack::kmeans::KMeans< MetricType, InitialPartitionPolicy, EmptyClusterPolicy >::partitioner [private]`

Instantiated initial partitioning policy.

Definition at line 189 of file `kmeans.hpp`.

Referenced by `mlpack::kmeans::KMeans< MetricType, InitialPartitionPolicy, EmptyClusterPolicy >::Partitioner()`.

The documentation for this class was generated from the following file:

- `src/mlpack/methods/kmeans/kmeans.hpp`

22.74 mlpack::kmeans::MaxVarianceNewCluster Class Reference

When an empty cluster is detected, this class takes the point furthest from the centroid of the cluster with maximum variance as a new cluster.

Public Member Functions

- **MaxVarianceNewCluster ()**

Default constructor required by EmptyClusterPolicy.

Static Public Member Functions

- `template<typename MatType >`
`static size_t EmptyCluster (const MatType &data, const size_t emptyCluster, const MatType ¢roids, arma::Col< size_t > &clusterCounts, arma::Col< size_t > &assignments)`

Take the point furthest from the centroid of the cluster with maximum variance to be a new cluster.

22.74.1 Detailed Description

When an empty cluster is detected, this class takes the point furthest from the centroid of the cluster with maximum variance as a new cluster.

Definition at line 28 of file `max_variance_new_cluster.hpp`.

22.74.2 Constructor & Destructor Documentation

22.74.2.1 `mlpack::kmeans::MaxVarianceNewCluster::MaxVarianceNewCluster () [inline]`

Default constructor required by EmptyClusterPolicy.

Definition at line 32 of file `max_variance_new_cluster.hpp`.

22.74.3 Member Function Documentation


```
22.74.3.1  template<typename MatType > static size_t mlpack::kmeans::MaxVarianceNewCluster::EmptyCluster ( const MatType &
            data, const size_t emptyCluster, const MatType & centroids, arma::Col< size_t > & clusterCounts, arma::Col< size_t >
            & assignments ) [static]
```

Take the point furthest from the centroid of the cluster with maximum variance to be a new cluster.

Template Parameters

<i>MatType</i>	Type of data (arma::mat or arma::sp_mat).
----------------	---

Parameters

<i>data</i>	Dataset on which clustering is being performed.
<i>emptyCluster</i>	Index of cluster which is empty.
<i>centroids</i>	Centroids of each cluster (one per column).
<i>clusterCounts</i>	Number of points in each cluster.
<i>assignments</i>	Cluster assignments of each point.

Returns

Number of points changed.

The documentation for this class was generated from the following file:

- src/mlpack/methods/kmeans/**max_variance_new_cluster.hpp**

22.75 mlpack::kmeans::RandomPartition Class Reference

A very simple partitioner which partitions the data randomly into the number of desired clusters.

Public Member Functions

- **RandomPartition** ()
Empty constructor, required by the InitialPartitionPolicy policy.

Static Public Member Functions

- template<typename MatType >
static void **Cluster** (const MatType &data, const size_t clusters, arma::Col< size_t > &assignments)
Partition the given dataset into the given number of clusters.

22.75.1 Detailed Description

A very simple partitioner which partitions the data randomly into the number of desired clusters.

It has no parameters, and so an instance of the class is not even necessary.

Definition at line 28 of file random_partition.hpp.

22.75.2 Constructor & Destructor Documentation

22.75.2.1 mlpack::kmeans::RandomPartition::RandomPartition () [inline]

Empty constructor, required by the InitialPartitionPolicy policy.

Definition at line 32 of file random_partition.hpp.

22.75.3 Member Function Documentation

22.75.3.1 `template<typename MatType > static void mlpack::kmeans::RandomPartition::Cluster (const MatType & data, const size_t clusters, arma::Col< size_t > & assignments) [inline], [static]`

Partition the given dataset into the given number of clusters.

Assignments are random, and the number of points in each cluster should be equal (or approximately equal).

Template Parameters

<i>MatType</i>	Type of data (arma::mat or arma::sp_mat).
----------------	---

Parameters

<i>data</i>	Dataset to partition.
<i>clusters</i>	Number of clusters to split dataset into.
<i>assignments</i>	Vector to store cluster assignments into. Values will be between 0 and (clusters - 1).

Definition at line 46 of file random_partition.hpp.

The documentation for this class was generated from the following file:

- src/mlpack/methods/kmeans/random_partition.hpp

22.76 mlpack::kmeans::RefinedStart Class Reference

A refined approach for choosing initial points for k-means clustering.

Public Member Functions

- **RefinedStart** (const size_t **samplings**=100, const double **percentage**=0.02)
*Create the **RefinedStart** (p. 343) object, optionally specifying parameters for the number of samplings to perform and the percentage of the dataset to use in each sampling.*
- `template<typename MatType > void Cluster (const MatType &data, const size_t clusters, arma::Col< size_t > &assignments) const`
Partition the given dataset into the given number of clusters according to the random sampling scheme outlined in Bradley and Fayyad's paper.
- double **Percentage** () const
Get the percentage of the data used by each subsampling.
- double & **Percentage** ()
Modify the percentage of the data used by each subsampling.
- size_t **Samplings** () const
Get the number of samplings that will be performed.
- size_t & **Samplings** ()
Modify the number of samplings that will be performed.

Private Attributes

- double **percentage**
The percentage of the data to use for each subsampling.

- **size_t samplings**

The number of samplings to perform.

22.76.1 Detailed Description

A refined approach for choosing initial points for k-means clustering.

This approach runs k-means several times on random subsets of the data, and then clusters those solutions to select refined initial cluster assignments. It is an implementation of the following paper:

{bradley1998refining, title={Refining initial points for k-means clustering}, author={Bradley, Paul S and Fayyad, Usama M}, booktitle={Proceedings of the Fifteenth International Conference on Machine Learning (ICML 1998)}, volume={66}, year={1998} }

Definition at line 39 of file refined_start.hpp.

22.76.2 Constructor & Destructor Documentation

22.76.2.1 `mlpack::kmeans::RefinedStart::RefinedStart (const size_t samplings = 100, const double percentage = 0.02)`
`[inline]`

Create the **RefinedStart** (p.343) object, optionally specifying parameters for the number of samplings to perform and the percentage of the dataset to use in each sampling.

Definition at line 47 of file refined_start.hpp.

22.76.3 Member Function Documentation

22.76.3.1 `template<typename MatType > void mlpack::kmeans::RefinedStart::Cluster (const MatType & data, const size_t clusters, arma::Col< size_t > & assignments) const`

Partition the given dataset into the given number of clusters according to the random sampling scheme outlined in Bradley and Fayyad's paper.

Template Parameters

<i>MatType</i>	Type of data (arma::mat or arma::sp_mat).
----------------	---

Parameters

<i>data</i>	Dataset to partition.
<i>clusters</i>	Number of clusters to split dataset into.
<i>assignments</i>	Vector to store cluster assignments into. Values will be between 0 and (clusters - 1).

22.76.3.2 `double mlpack::kmeans::RefinedStart::Percentage () const` `[inline]`

Get the percentage of the data used by each subsampling.

Definition at line 72 of file refined_start.hpp.

References percentage.

22.76.3.3 `double& mlpack::kmeans::RefinedStart::Percentage () [inline]`

Modify the percentage of the data used by each subsampling.

Definition at line 74 of file `refined_start.hpp`.

References `percentage`.

22.76.3.4 `size_t mlpack::kmeans::RefinedStart::Samplings () const [inline]`

Get the number of samplings that will be performed.

Definition at line 67 of file `refined_start.hpp`.

References `samplings`.

22.76.3.5 `size_t& mlpack::kmeans::RefinedStart::Samplings () [inline]`

Modify the number of samplings that will be performed.

Definition at line 69 of file `refined_start.hpp`.

References `samplings`.

22.76.4 Member Data Documentation

22.76.4.1 `double mlpack::kmeans::RefinedStart::percentage [private]`

The percentage of the data to use for each subsampling.

Definition at line 80 of file `refined_start.hpp`.

Referenced by `Percentage()`.

22.76.4.2 `size_t mlpack::kmeans::RefinedStart::samplings [private]`

The number of samplings to perform.

Definition at line 78 of file `refined_start.hpp`.

Referenced by `Samplings()`.

The documentation for this class was generated from the following file:

- `src/mlpack/methods/kmeans/refined_start.hpp`

22.77 mlpack::kpca::KernelPCA< KernelType, KernelRule > Class Template Reference

This class performs kernel principal components analysis (Kernel PCA), for a given kernel.

Public Member Functions

- **KernelPCA** (const KernelType **kernel**=KernelType(), const bool **centerTransformedData**=false)
*Construct the **KernelPCA** (p. 345) object, optionally passing a kernel.*

- void **Apply** (const arma::mat &data, arma::mat &transformedData, arma::vec &eigval, arma::mat &eigvec, const size_t newDimension)
Apply Kernel Principal Components Analysis to the provided data set.
- void **Apply** (const arma::mat &data, arma::mat &transformedData, arma::vec &eigval, arma::mat &eigvec)
Apply Kernel Principal Components Analysis to the provided data set.
- void **Apply** (const arma::mat &data, arma::mat &transformedData, arma::vec &eigval)
Apply Kernel Principal Component Analysis to the provided data set.
- void **Apply** (arma::mat &data, const size_t newDimension)
Apply dimensionality reduction using Kernel Principal Component Analysis to the provided data set.
- bool **CenterTransformedData** () const
Return whether or not the transformed data is centered.
- bool & **CenterTransformedData** ()
Return whether or not the transformed data is centered.
- const KernelType & **Kernel** () const
Get the kernel.
- KernelType & **Kernel** ()
Modify the kernel.
- std::string **ToString** () const

Private Attributes

- bool **centerTransformedData**
*If true, the data will be scaled (by standard deviation) when **Apply()** (p. 348) is run.*
- KernelType **kernel**
The instantiated kernel.

22.77.1 Detailed Description

```
template<typename KernelType, typename KernelRule = NaiveKernelRule<KernelType>>class mlpack::kpca::KernelPCA<
KernelType, KernelRule >
```

This class performs kernel principal components analysis (Kernel PCA), for a given kernel.

This is a standard machine learning technique and is well-documented on the Internet and in standard texts. It is often used as a dimensionality reduction technique, and can also be useful in mapping linearly inseparable classes of points to different spaces where they are linearly separable.

The performance of the method is highly dependent on the kernel choice. There are numerous available kernels in the **mlpack::kernel** (p. 101) namespace (see files in mlpack/core/kernels/) and it is easy to write your own; see other implementations for examples.

Definition at line 42 of file kernel_pca.hpp.

22.77.2 Constructor & Destructor Documentation

```
22.77.2.1 template<typename KernelType , typename KernelRule = NaiveKernelRule<KernelType>>
mlpack::kpca::KernelPCA< KernelType, KernelRule >::KernelPCA ( const KernelType kernel =
KernelType(), const bool centerTransformedData = false )
```

Construct the **KernelPCA** (p. 345) object, optionally passing a kernel.

Optionally, the transformed data can be centered about the origin; to do this, pass 'true' for centerTransformedData. This will take slightly longer (but not much).

Parameters

<i>kernel</i>	Kernel to be used for computation.
<i>center</i> ↔ <i>TransformedData</i>	Center transformed data.

22.77.3 Member Function Documentation

22.77.3.1 `template<typename KernelType , typename KernelRule = NaiveKernelRule<KernelType>> void
mlpack::kpca::KernelPCA< KernelType, KernelRule >::Apply (const arma::mat & data, arma::mat &
transformedData, arma::vec & eigval, arma::mat & eigvec, const size_t newDimension)`

Apply Kernel Principal Components Analysis to the provided data set.

Parameters

<i>data</i>	Data matrix.
<i>transformedData</i>	Matrix to output results into.
<i>eigval</i>	KPCA eigenvalues will be written to this vector.
<i>eigvec</i>	KPCA eigenvectors will be written to this matrix.
<i>newDimension</i>	New dimension for the dataset.

22.77.3.2 `template<typename KernelType , typename KernelRule = NaiveKernelRule<KernelType>> void
mlpack::kpca::KernelPCA< KernelType, KernelRule >::Apply (const arma::mat & data, arma::mat &
transformedData, arma::vec & eigval, arma::mat & eigvec)`

Apply Kernel Principal Components Analysis to the provided data set.

Parameters

<i>data</i>	Data matrix.
<i>transformedData</i>	Matrix to output results into.
<i>eigval</i>	KPCA eigenvalues will be written to this vector.
<i>eigvec</i>	KPCA eigenvectors will be written to this matrix.

22.77.3.3 `template<typename KernelType , typename KernelRule = NaiveKernelRule<KernelType>> void
mlpack::kpca::KernelPCA< KernelType, KernelRule >::Apply (const arma::mat & data, arma::mat &
transformedData, arma::vec & eigval)`

Apply Kernel Principal Component Analysis to the provided data set.

Parameters

<i>data</i>	Data matrix.
<i>transformedData</i>	Matrix to output results into.
<i>eigval</i>	KPCA eigenvalues will be written to this vector.

22.77.3.4 `template<typename KernelType , typename KernelRule = NaiveKernelRule<KernelType>> void
mlpack::kpca::KernelPCA< KernelType, KernelRule >::Apply (arma::mat & data, const size_t newDimension)`

Apply dimensionality reduction using Kernel Principal Component Analysis to the provided data set.

The data matrix will be modified in-place. Note that the dimension can be larger than the existing dimension because KPCA works on the kernel matrix, not the covariance matrix. This means the new dimension can be as large as the number of points (columns) in the dataset. Note that if you specify newDimension to be larger than the current dimension of the data (the number of rows), then it's not really "dimensionality reduction"...

Parameters

<i>data</i>	Data matrix.
<i>newDimension</i>	New dimension for the dataset.

22.77.3.5 `template<typename KernelType , typename KernelRule = NaiveKernelRule<KernelType>> bool
mlpack::kpca::KernelPCA< KernelType, KernelRule >::CenterTransformedData () const [inline]`

Return whether or not the transformed data is centered.

Definition at line 117 of file kernel_pca.hpp.

References mlpack::kpca::KernelPCA< KernelType, KernelRule >::centerTransformedData.

22.77.3.6 `template<typename KernelType , typename KernelRule = NaiveKernelRule<KernelType>> bool&
mlpack::kpca::KernelPCA< KernelType, KernelRule >::CenterTransformedData () [inline]`

Return whether or not the transformed data is centered.

Definition at line 119 of file kernel_pca.hpp.

References mlpack::kpca::KernelPCA< KernelType, KernelRule >::centerTransformedData.

22.77.3.7 `template<typename KernelType , typename KernelRule = NaiveKernelRule<KernelType>> const KernelType&
mlpack::kpca::KernelPCA< KernelType, KernelRule >::Kernel () const [inline]`

Get the kernel.

Definition at line 112 of file kernel_pca.hpp.

References mlpack::kpca::KernelPCA< KernelType, KernelRule >::kernel.

22.77.3.8 `template<typename KernelType , typename KernelRule = NaiveKernelRule<KernelType>> KernelType&
mlpack::kpca::KernelPCA< KernelType, KernelRule >::Kernel () [inline]`

Modify the kernel.

Definition at line 114 of file kernel_pca.hpp.

References mlpack::kpca::KernelPCA< KernelType, KernelRule >::kernel.

22.77.3.9 `template<typename KernelType , typename KernelRule = NaiveKernelRule<KernelType>> std::string
mlpack::kpca::KernelPCA< KernelType, KernelRule >::ToString () const`

22.77.4 Member Data Documentation

22.77.4.1 `template<typename KernelType , typename KernelRule = NaiveKernelRule<KernelType>> bool
mlpack::kpca::KernelPCA< KernelType, KernelRule >::centerTransformedData [private]`

If true, the data will be scaled (by standard deviation) when **Apply()** (p. 348) is run.

Definition at line 129 of file `kernel_pca.hpp`.

Referenced by `mlpack::kpca::KernelPCA< KernelType, KernelRule >::CenterTransformedData()`.

22.77.4.2 `template<typename KernelType , typename KernelRule = NaiveKernelRule<KernelType>> KernelType
mlpack::kpca::KernelPCA< KernelType, KernelRule >::kernel [private]`

The instantiated kernel.

Definition at line 126 of file `kernel_pca.hpp`.

Referenced by `mlpack::kpca::KernelPCA< KernelType, KernelRule >::Kernel()`.

The documentation for this class was generated from the following file:

- `src/mlpack/methods/kernel_pca/kernel_pca.hpp`

22.78 mlpack::kpca::NaiveKernelRule< KernelType > Class Template Reference

Static Public Member Functions

- static void **ApplyKernelMatrix** (const arma::mat &data, arma::mat &transformedData, arma::vec &eigval, arma::mat &eigvec, const size_t, KernelType kernel=KernelType())
Construct the kernel matrix approximation using the nystroem method.

22.78.1 Detailed Description

`template<typename KernelType>class mlpack::kpca::NaiveKernelRule< KernelType >`

Definition at line 24 of file `naive_method.hpp`.

22.78.2 Member Function Documentation

22.78.2.1 `template<typename KernelType > static void mlpack::kpca::NaiveKernelRule< KernelType >::ApplyKernelMatrix
(const arma::mat & data, arma::mat & transformedData, arma::vec & eigval, arma::mat & eigvec, const size_t ,
KernelType kernel = KernelType()) [inline],[static]`

Construct the kernel matrix approximation using the nystroem method.

Parameters

<i>data</i>	Input data points.
<i>transformedData</i>	Matrix to output results into.

<i>eigval</i>	KPCA eigenvalues will be written to this vector.
<i>eigvec</i>	KPCA eigenvectors will be written to this matrix.
<i>rank</i>	Rank to be used for matrix approximation.
<i>kernel</i>	Kernel to be used for computation.

Definition at line 38 of file naive_method.hpp.

The documentation for this class was generated from the following file:

- src/mlpack/methods/kernel_pca/kernel_rules/naive_method.hpp

22.79 mlpack::kpca::NystroemKernelRule< KernelType, PointSelectionPolicy > Class Template Reference

Static Public Member Functions

- static void **ApplyKernelMatrix** (const arma::mat &data, arma::mat &transformedData, arma::vec &eigval, arma::mat &eigvec, const size_t rank, KernelType kernel=KernelType())
Construct the kernel matrix approximation using the nystroem method.

22.79.1 Detailed Description

```
template<typename KernelType, typename PointSelectionPolicy = kernel::KMeansSelection<>> class mlpack::kpca::NystroemKernelRule< KernelType, PointSelectionPolicy >
```

Definition at line 29 of file nystroem_method.hpp.

22.79.2 Member Function Documentation

```
22.79.2.1 template<typename KernelType, typename PointSelectionPolicy = kernel::KMeansSelection<>> static void mlpack::kpca::NystroemKernelRule< KernelType, PointSelectionPolicy >::ApplyKernelMatrix ( const arma::mat & data, arma::mat & transformedData, arma::vec & eigval, arma::mat & eigvec, const size_t rank, KernelType kernel = KernelType() ) [inline],[static]
```

Construct the kernel matrix approximation using the nystroem method.

Parameters

<i>data</i>	Input data points.
<i>transformedData</i>	Matrix to output results into.
<i>eigval</i>	KPCA eigenvalues will be written to this vector.
<i>eigvec</i>	KPCA eigenvectors will be written to this matrix.
<i>rank</i>	Rank to be used for matrix approximation.
<i>kernel</i>	Kernel to be used for computation.

Definition at line 42 of file nystroem_method.hpp.

References mlpack::kernel::NystroemMethod< KernelType, PointSelectionPolicy >::Apply().

The documentation for this class was generated from the following file:

- src/mlpack/methods/kernel_pca/kernel_rules/nystroem_method.hpp

22.80 mlpack::lcc::LocalCoordinateCoding< DictionaryInitializer > Class Template Reference

An implementation of Local Coordinate Coding (LCC) that codes data which approximately lives on a manifold using a variation of l_1 -norm regularized sparse coding; in LCC, the penalty on the absolute value of each point's coefficient for each atom is weighted by the squared distance of that point to that atom.

Public Member Functions

- **LocalCoordinateCoding** (const arma::mat &data, const size_t atoms, const double lambda)

*Set the parameters to **LocalCoordinateCoding** (p. 352).*

- const arma::mat & **Codes** () const

Accessor the codes.

- arma::mat & **Codes** ()

Modify the codes.

- const arma::mat & **Data** () const

Access the data.

- const arma::mat & **Dictionary** () const

Accessor for dictionary.

- arma::mat & **Dictionary** ()

Mutator for dictionary.

- void **Encode** (const size_t maxIterations=0, const double objTolerance=0.01)

Run local coordinate coding.

- double **Objective** (arma::uvec adjacencies) const

Compute objective function given the list of adjacencies.

- void **OptimizeCode** ()

Code each point via distance-weighted LARS.

- void **OptimizeDictionary** (arma::uvec adjacencies)

Learn dictionary by solving linear system.

- std::string **ToString** () const

Private Attributes

- size_t **atoms**

Number of atoms in dictionary.

- arma::mat **codes**

Codes (columns are points).

- const arma::mat & **data**

Data matrix (columns are points).

- arma::mat **dictionary**

Dictionary (columns are atoms).

- double **lambda**

l_1 regularization term.

22.80.1 Detailed Description

template<typename DictionaryInitializer = sparse_coding::DataDependentRandomInitializer>class mpack::lcc::LocalCoordinateCoding< DictionaryInitializer >

An implementation of Local Coordinate Coding (LCC) that codes data which approximately lives on a manifold using a variation of l1-norm regularized sparse coding; in LCC, the penalty on the absolute value of each point's coefficient for each atom is weighted by the squared distance of that point to that atom.

Let d be the number of dimensions in the original space, m the number of training points, and k the number of atoms in the dictionary (the dimension of the learned feature space). The training data X is a d -by- m matrix where each column is a point and each row is a dimension. The dictionary D is a d -by- k matrix, and the sparse codes matrix Z is a k -by- m matrix. This program seeks to minimize the objective: $\min_{D,Z} ||X - DZ||_{\text{Fro}}^2$

- $\lambda \sum_{i=1}^m \sum_{j=1}^k \text{dist}(X_i, D_j)^2 Z_{ij}$ where $\lambda > 0$.

This problem is solved by an algorithm that alternates between a dictionary learning step and a sparse coding step. The dictionary learning step updates the dictionary D by solving a linear system (note that the objective is a positive definite quadratic program). The sparse coding step involves solving a large number of weighted l1-norm regularized linear regression problems; this can be done efficiently using LARS, an algorithm that can solve the LASSO (paper below).

The papers are listed below.

```
@incollection{NIPS2009_0719,
  title = {Nonlinear Learning using Local Coordinate Coding},
  author = {Kai Yu and Tong Zhang and Yihong Gong},
  booktitle = {Advances in Neural Information Processing Systems 22},
  editor = {Y. Bengio and D. Schuurmans and J. Lafferty and C. K. I. Williams
    and A. Culotta},
  pages = {2223--2231},
  year = {2009}
}
```

```
@article{efron2004least,
  title={Least angle regression},
  author={Efron, B. and Hastie, T. and Johnstone, I. and Tibshirani, R.},
  journal={The Annals of statistics},
  volume={32},
  number={2},
  pages={407--499},
  year={2004},
  publisher={Institute of Mathematical Statistics}
}
```

Definition at line 83 of file lcc.hpp.

22.80.2 Constructor & Destructor Documentation

22.80.2.1 template<typename DictionaryInitializer = sparse_coding::DataDependentRandomInitializer>
mpack::lcc::LocalCoordinateCoding< DictionaryInitializer >::LocalCoordinateCoding(const arma::mat &
data, const size_t atoms, const double lambda)

Set the parameters to **LocalCoordinateCoding** (p. 352).

Parameters

<i>data</i>	Data matrix.
<i>atoms</i>	Number of atoms in dictionary.
<i>lambda</i>	Regularization parameter for weighted l1-norm penalty.

22.80.3 Member Function Documentation

22.80.3.1 `template<typename DictionaryInitializer = sparse_coding::DataDependentRandomInitializer> const arma::mat& mlpack::lcc::LocalCoordinateCoding< DictionaryInitializer >::Codes () const [inline]`

Accessor the codes.

Definition at line 136 of file lcc.hpp.

References mlpack::lcc::LocalCoordinateCoding< DictionaryInitializer >::codes.

22.80.3.2 `template<typename DictionaryInitializer = sparse_coding::DataDependentRandomInitializer> arma::mat& mlpack::lcc::LocalCoordinateCoding< DictionaryInitializer >::Codes () [inline]`

Modify the codes.

Definition at line 138 of file lcc.hpp.

References mlpack::lcc::LocalCoordinateCoding< DictionaryInitializer >::codes.

22.80.3.3 `template<typename DictionaryInitializer = sparse_coding::DataDependentRandomInitializer> const arma::mat& mlpack::lcc::LocalCoordinateCoding< DictionaryInitializer >::Data () const [inline]`

Access the data.

Definition at line 128 of file lcc.hpp.

References mlpack::lcc::LocalCoordinateCoding< DictionaryInitializer >::data.

22.80.3.4 `template<typename DictionaryInitializer = sparse_coding::DataDependentRandomInitializer> const arma::mat& mlpack::lcc::LocalCoordinateCoding< DictionaryInitializer >::Dictionary () const [inline]`

Accessor for dictionary.

Definition at line 131 of file lcc.hpp.

References mlpack::lcc::LocalCoordinateCoding< DictionaryInitializer >::dictionary.

22.80.3.5 `template<typename DictionaryInitializer = sparse_coding::DataDependentRandomInitializer> arma::mat& mlpack::lcc::LocalCoordinateCoding< DictionaryInitializer >::Dictionary () [inline]`

Mutator for dictionary.

Definition at line 133 of file lcc.hpp.

References mlpack::lcc::LocalCoordinateCoding< DictionaryInitializer >::dictionary.

```
22.80.3.6  template<typename DictionaryInitializer = sparse_coding::DataDependentRandomInitializer> void  
           mlpack::lcc::LocalCoordinateCoding< DictionaryInitializer >::Encode ( const size_t maxIterations = 0, const  
           double objTolerance = 0.01 )
```

Run local coordinate coding.

Parameters

<i>nIterations</i>	Maximum number of iterations to run algorithm.
<i>objTolerance</i>	Tolerance of objective function. When the objective function changes by a value lower than this tolerance, the optimization terminates.

22.80.3.7 `template<typename DictionaryInitializer = sparse_coding::DataDependentRandomInitializer> double
mlpack::lcc::LocalCoordinateCoding< DictionaryInitializer >::Objective (arma::uvec adjacencies) const`

Compute objective function given the list of adjacencies.

22.80.3.8 `template<typename DictionaryInitializer = sparse_coding::DataDependentRandomInitializer> void
mlpack::lcc::LocalCoordinateCoding< DictionaryInitializer >::OptimizeCode ()`

Code each point via distance-weighted LARS.

22.80.3.9 `template<typename DictionaryInitializer = sparse_coding::DataDependentRandomInitializer> void
mlpack::lcc::LocalCoordinateCoding< DictionaryInitializer >::OptimizeDictionary (arma::uvec adjacencies)`

Learn dictionary by solving linear system.

Parameters

<i>adjacencies</i>	Indices of entries (unrolled column by column) of the coding matrix Z that are non-zero (the adjacency matrix for the bipartite graph of points and atoms)
--------------------	--

22.80.3.10 `template<typename DictionaryInitializer = sparse_coding::DataDependentRandomInitializer> std::string
mlpack::lcc::LocalCoordinateCoding< DictionaryInitializer >::ToString () const`

22.80.4 Member Data Documentation

22.80.4.1 `template<typename DictionaryInitializer = sparse_coding::DataDependentRandomInitializer> size_t
mlpack::lcc::LocalCoordinateCoding< DictionaryInitializer >::atoms [private]`

Number of atoms in dictionary.

Definition at line 145 of file lcc.hpp.

22.80.4.2 `template<typename DictionaryInitializer = sparse_coding::DataDependentRandomInitializer> arma::mat
mlpack::lcc::LocalCoordinateCoding< DictionaryInitializer >::codes [private]`

Codes (columns are points).

Definition at line 154 of file lcc.hpp.

Referenced by `mlpack::lcc::LocalCoordinateCoding< DictionaryInitializer >::Codes()`.

22.80.4.3 `template<typename DictionaryInitializer = sparse_coding::DataDependentRandomInitializer> const arma::mat&
mlpack::lcc::LocalCoordinateCoding< DictionaryInitializer >::data [private]`

Data matrix (columns are points).

Definition at line 148 of file lcc.hpp.

Referenced by mlpack::lcc::LocalCoordinateCoding< DictionaryInitializer >::Data().

```
22.80.4.4  template<typename DictionaryInitializer = sparse_coding::DataDependentRandomInitializer> arma::mat
          mlpack::lcc::LocalCoordinateCoding< DictionaryInitializer >::dictionary [private]
```

Dictionary (columns are atoms).

Definition at line 151 of file lcc.hpp.

Referenced by mlpack::lcc::LocalCoordinateCoding< DictionaryInitializer >::Dictionary().

```
22.80.4.5  template<typename DictionaryInitializer = sparse_coding::DataDependentRandomInitializer> double
          mlpack::lcc::LocalCoordinateCoding< DictionaryInitializer >::lambda [private]
```

l1 regularization term.

Definition at line 157 of file lcc.hpp.

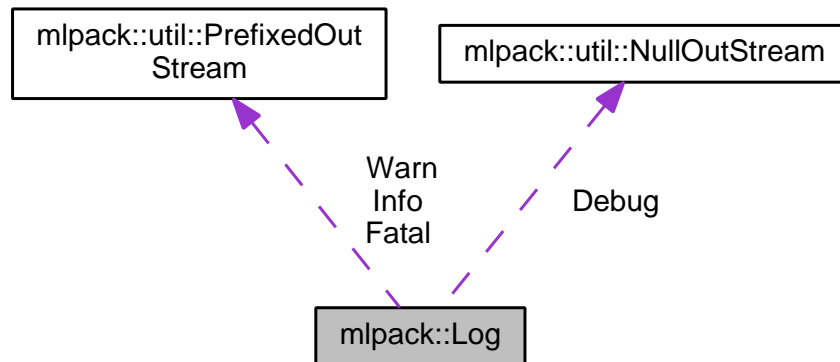
The documentation for this class was generated from the following file:

- src/mlpack/methods/local_coordinate_coding/lcc.hpp

22.81 mlpack::Log Class Reference

Provides a convenient way to give formatted output.

Collaboration diagram for mlpack::Log:



Static Public Member Functions

- static void **Assert** (bool condition, const std::string &message="Assert Failed.")
Checks if the specified condition is true.

Static Public Attributes

- static std::ostream & **cout**

Reference to cout, if necessary.

- static **util::NullOutputStream Debug**

Dumps debug output into the bit nether regions.

- static **util::PrefixedOutputStream Fatal**

Prints fatal messages prefixed with [FATAL], then terminates the program.

- static **util::PrefixedOutputStream Info**

Prints informational messages if `-verbose` is specified, prefixed with [INFO].

- static **util::PrefixedOutputStream Warn**

Prints warning messages prefixed with [WARN].

22.81.1 Detailed Description

Provides a convenient way to give formatted output.

The **Log** (p. 357) class has four members which can be used in the same way ostream can be used:

- **Log::Debug** (p. 359)
- **Log::Info** (p. 359)
- **Log::Warn** (p. 359)
- **Log::Fatal** (p. 359)

Each of these will prefix a tag to the output (for easy filtering), and the fatal output will terminate the program when a newline is encountered. An example is given below.

```
Log::Info << "Checking a condition." << std::endl;
if (!someCondition())
    Log::Warn << "someCondition() is not satisfied!" << std::endl;
Log::Info << "Checking an important condition." << std::endl;
if (!someImportantCondition())
{
    Log::Fatal << "someImportantCondition() is not satisfied! Terminating.";
    Log::Fatal << std::endl;
}
```

Any messages sent to **Log::Debug** (p. 359) will not be shown when compiling in non-debug mode. Messages to **Log::Info** (p. 359) will only be shown when the `-verbose` flag is given to the program (or rather, the **CLI** (p. 184) class).

See also

PrefixedOutputStream, NullOutputStream, **CLI** (p. 184)

Definition at line 57 of file log.hpp.

22.81.2 Member Function Documentation

22.81.2.1 static void mpack::Log::Assert (bool *condition*, const std::string & *message* = "Assert Failed.") [static]

Checks if the specified condition is true.

If not, halts program execution and prints a custom error message. Does nothing in non-debug mode.

Referenced by mpack::emst::EdgePair::EdgePair().

22.81.3 Member Data Documentation

22.81.3.1 `std::ostream& mpack::Log::cout` `[static]`

Reference to `cout`, if necessary.

Definition at line 90 of file `log.hpp`.

22.81.3.2 `util::NullOutputStream mpack::Log::Debug` `[static]`

Dumps debug output into the bit nether regions.

Definition at line 76 of file `log.hpp`.

Referenced by `mpack::gmm::PositiveDefiniteConstraint::ApplyConstraint()`, `mpack::gmm::GMM< FittingType >::GMM()`, and `mpack::distribution::DiscreteDistribution::Probability()`.

22.81.3.3 `util::PrefixedOutputStream mpack::Log::Fatal` `[static]`

Prints fatal messages prefixed with `[FATAL]`, then terminates the program.

Definition at line 87 of file `log.hpp`.

Referenced by `mpack::kernel::SphericalKernel::ConvolutionIntegral()`, and `mpack::gmm::EigenvalueRatioConstraint::EigenvalueRatioConstraint()`.

22.81.3.4 `util::PrefixedOutputStream mpack::Log::Info` `[static]`

Prints informational messages if `-verbose` is specified, prefixed with `[INFO]`.

Definition at line 81 of file `log.hpp`.

22.81.3.5 `util::PrefixedOutputStream mpack::Log::Warn` `[static]`

Prints warning messages prefixed with `[WARN]`.

Definition at line 84 of file `log.hpp`.

Referenced by `mpack::gmm::EigenvalueRatioConstraint::EigenvalueRatioConstraint()`, `mpack::amf::RandomAcolInitialization< p >::Initialize()`, and `mpack::cf::CF< FactorizerType >::NumUsersForSimilarity()`.

The documentation for this class was generated from the following file:

- `src/mpack/core/util/log.hpp`

22.82 mpack::math::Range Class Reference

Simple real-valued range.

Public Member Functions

- **Range()**
The upper bound.

- **Range** (const double point)
- **Range** (const double **lo**, const double **hi**)
Initializes to specified range.
- bool **Contains** (const double d) const
Determines if a point is contained within the range.
- bool **Contains** (const **Range** &r) const
Determines if another range overlaps with this one.
- double **Hi** () const
Get the upper bound.
- double & **Hi** ()
Modify the upper bound.
- double **Lo** () const
Get the lower bound.
- double & **Lo** ()
Modify the lower bound.
- double **Mid** () const
Gets the midpoint of this range.
- bool **operator!=** (const **Range** &rhs) const
Compare with another range for strict equality.
- **Range operator&** (const **Range** &rhs) const
Shrinks this range to be the overlap with another range; this makes an empty set if there is no overlap.
- **Range & operator&=** (const **Range** &rhs)
Shrinks this range to be the overlap with another range; this makes an empty set if there is no overlap.
- **Range operator*** (const double d) const
Scale the bounds by the given double.
- **Range & operator*=** (const double d)
Scale the bounds by the given double.
- bool **operator<** (const **Range** &rhs) const
Compare with another range.
- bool **operator==** (const **Range** &rhs) const
Compare with another range for strict equality.
- bool **operator>** (const **Range** &rhs) const
Compare with another range.
- **Range operator|** (const **Range** &rhs) const
Expands this range to include another range.
- **Range & operator|=** (const **Range** &rhs)
Expands this range to include another range.
- std::string **ToString** () const
Returns a string representation of an object.
- double **Width** () const
Gets the span of the range (hi - lo).

Private Attributes

- double **hi**
The lower bound.
- double **lo**

Friends

- **Range operator*** (const double d, const **Range** &r)

Scale the bounds by the given double.

22.82.1 Detailed Description

Simple real-valued range.

It contains an upper and lower bound.

Definition at line 23 of file range.hpp.

22.82.2 Constructor & Destructor Documentation

22.82.2.1 `mpack::math::Range::Range ()` `[inline]`

The upper bound.

Initialize to an empty set (where lo > hi).

22.82.2.2 `mpack::math::Range::Range (const double point)` `[inline]`

22.82.2.3 `mpack::math::Range::Range (const double lo, const double hi)` `[inline]`

Initializes to specified range.

Parameters

<i>lo</i>	Lower bound of the range.
<i>hi</i>	Upper bound of the range.

22.82.3 Member Function Documentation

22.82.3.1 `bool mpack::math::Range::Contains (const double d) const` `[inline]`

Determines if a point is contained within the range.

Parameters

<i>d</i>	Point to check.
----------	-----------------

22.82.3.2 `bool mpack::math::Range::Contains (const Range & r) const` `[inline]`

Determines if another range overlaps with this one.

Parameters

<i>r</i>	Other range.
----------	--------------

Returns

true if ranges overlap at all.

22.82.3.3 `double mlpack::math::Range::Hi () const` `[inline]`

Get the upper bound.

Definition at line 55 of file range.hpp.

References `hi`.

22.82.3.4 `double& mlpack::math::Range::Hi ()` `[inline]`

Modify the upper bound.

Definition at line 57 of file range.hpp.

References `hi`.

22.82.3.5 `double mlpack::math::Range::Lo () const` `[inline]`

Get the lower bound.

Definition at line 50 of file range.hpp.

References `lo`.

22.82.3.6 `double& mlpack::math::Range::Lo ()` `[inline]`

Modify the lower bound.

Definition at line 52 of file range.hpp.

References `lo`.

22.82.3.7 `double mlpack::math::Range::Mid () const` `[inline]`

Gets the midpoint of this range.

22.82.3.8 `bool mlpack::math::Range::operator!= (const Range & rhs) const` `[inline]`

Compare with another range for strict equality.

Parameters

<i>rhs</i>	Other range.
------------	--------------

22.82.3.9 `Range mlpack::math::Range::operator& (const Range & rhs) const` `[inline]`

Shrinks this range to be the overlap with another range; this makes an empty set if there is no overlap.

Parameters

<i>rhs</i>	Other range.
------------	--------------

22.82.3.10 **Range&** mpack::math::Range::operator&= (const Range & *rhs*) [inline]

Shrinks this range to be the overlap with another range; this makes an empty set if there is no overlap.

Parameters

<i>rhs</i>	Other range.
------------	--------------

22.82.3.11 **Range** mpack::math::Range::operator* (const double *d*) const [inline]

Scale the bounds by the given double.

Parameters

<i>d</i>	Scaling factor.
----------	-----------------

22.82.3.12 **Range&** mpack::math::Range::operator*= (const double *d*) [inline]

Scale the bounds by the given double.

Parameters

<i>d</i>	Scaling factor.
----------	-----------------

22.82.3.13 **bool** mpack::math::Range::operator< (const Range & *rhs*) const [inline]

Compare with another range.

For **Range** (p. 359) objects *x* and *y*, $x < y$ means that *x* is strictly less than *y* and does not overlap at all.

Parameters

<i>rhs</i>	Other range.
------------	--------------

22.82.3.14 **bool** mpack::math::Range::operator== (const Range & *rhs*) const [inline]

Compare with another range for strict equality.

Parameters

<i>rhs</i>	Other range.
------------	--------------

22.82.3.15 **bool** mpack::math::Range::operator> (const Range & *rhs*) const [inline]

Compare with another range.

For **Range** (p. 359) objects *x* and *y*, $x < y$ means that *x* is strictly less than *y* and does not overlap at all.

Parameters

<i>rhs</i>	Other range.
------------	--------------

22.82.3.16 `Range mpack::math::Range::operator| (const Range & rhs) const` `[inline]`

Expands this range to include another range.

Parameters

<i>rhs</i>	Range (p. 359) to include.
------------	-----------------------------------

22.82.3.17 `Range& mpack::math::Range::operator|= (const Range & rhs)` `[inline]`

Expands this range to include another range.

Parameters

<i>rhs</i>	Range (p. 359) to include.
------------	-----------------------------------

22.82.3.18 `std::string mpack::math::Range::ToString () const` `[inline]`

Returns a string representation of an object.

22.82.3.19 `double mpack::math::Range::Width () const` `[inline]`

Gets the span of the range (hi - lo).

22.82.4 Friends And Related Function Documentation

22.82.4.1 `Range operator* (const double d, const Range & r)` `[friend]`

Scale the bounds by the given double.

Parameters

<i>d</i>	Scaling factor.
----------	-----------------

22.82.5 Member Data Documentation

22.82.5.1 `double mpack::math::Range::hi` `[private]`

The lower bound.

Definition at line 27 of file range.hpp.

Referenced by `Hi()`.

22.82.5.2 double mlpack::math::Range::lo [private]

Definition at line 26 of file range.hpp.

Referenced by Lo().

The documentation for this class was generated from the following file:

- src/mlpack/core/math/range.hpp

22.83 mlpack::metric::IPMetric< KernelType > Class Template Reference

Public Member Functions

- **IPMetric** ()
*Create the **IPMetric** (p. 365) without an instantiated kernel.*
- **IPMetric** (KernelType &kernel)
*Create the **IPMetric** (p. 365) with an instantiated kernel.*
- **~IPMetric** ()
*Destroy the **IPMetric** (p. 365) object.*
- template<typename Vec1Type , typename Vec2Type >
double **Evaluate** (const Vec1Type &a, const Vec2Type &b)
Evaluate the metric.
- const KernelType & **Kernel** () const
Get the kernel.
- KernelType & **Kernel** ()
Modify the kernel.
- std::string **ToString** () const
Returns a string representation of this object.

Private Attributes

- KernelType & **kernel**
The reference to the kernel that is being used.
- KernelType * **localKernel**
The locally stored kernel, if it is necessary.

22.83.1 Detailed Description

```
template<typename KernelType>class mlpack::metric::IPMetric< KernelType >
```

Definition at line 22 of file ip_metric.hpp.

22.83.2 Constructor & Destructor Documentation

22.83.2.1 template<typename KernelType> mlpack::metric::IPMetric< KernelType >::IPMetric ()

Create the **IPMetric** (p. 365) without an instantiated kernel.

22.83.2.2 `template<typename KernelType> mlpack::metric::IPMetric< KernelType >::IPMetric (KernelType & kernel)`

Create the **IPMetric** (p. 365) with an instantiated kernel.

22.83.2.3 `template<typename KernelType> mlpack::metric::IPMetric< KernelType >::~~IPMetric ()`

Destroy the **IPMetric** (p. 365) object.

22.83.3 Member Function Documentation

22.83.3.1 `template<typename KernelType> template<typename Vec1Type , typename Vec2Type > double mlpack::metric::IPMetric< KernelType >::Evaluate (const Vec1Type & a, const Vec2Type & b)`

Evaluate the metric.

22.83.3.2 `template<typename KernelType> const KernelType& mlpack::metric::IPMetric< KernelType >::Kernel () const [inline]`

Get the kernel.

Definition at line 41 of file ip_metric.hpp.

References mlpack::metric::IPMetric< KernelType >::kernel.

22.83.3.3 `template<typename KernelType> KernelType& mlpack::metric::IPMetric< KernelType >::Kernel () [inline]`

Modify the kernel.

Definition at line 43 of file ip_metric.hpp.

References mlpack::metric::IPMetric< KernelType >::kernel.

22.83.3.4 `template<typename KernelType> std::string mlpack::metric::IPMetric< KernelType >::ToString () const`

Returns a string representation of this object.

22.83.4 Member Data Documentation

22.83.4.1 `template<typename KernelType> KernelType& mlpack::metric::IPMetric< KernelType >::kernel [private]`

The reference to the kernel that is being used.

Definition at line 52 of file ip_metric.hpp.

Referenced by mlpack::metric::IPMetric< KernelType >::Kernel().

22.83.4.2 `template<typename KernelType> KernelType* mlpack::metric::IPMetric< KernelType >::localKernel [private]`

The locally stored kernel, if it is necessary.

Definition at line 50 of file ip_metric.hpp.

The documentation for this class was generated from the following file:

- src/mlpack/core/metrics/ip_metric.hpp

22.84 mlpack::metric::LMetric< Power, TakeRoot > Class Template Reference

The L_p metric for arbitrary integer p, with an option to take the root.

Public Member Functions

- **LMetric** ()
- std::string **ToString** () const

Static Public Member Functions

- template<typename VecType1 , typename VecType2 >
static double **Evaluate** (const VecType1 &a, const VecType2 &b)
Computes the distance between two points.

22.84.1 Detailed Description

template<int Power, bool TakeRoot = true>class mlpack::metric::LMetric< Power, TakeRoot >

The L_p metric for arbitrary integer p, with an option to take the root.

This class implements the standard L_p metric for two arbitrary vectors x and y of dimensionality n :

$$d(x, y) = \left(\sum_{i=1}^n |x_i - y_i|^p \right)^{\frac{1}{p}}.$$

The value of p is given as a template parameter.

In addition, the function $d(x, y)$ can be simplified, neglecting the p-root calculation. This is done by specifying the TakeRoot template parameter to be false. Then,

$$d(x, y) = \sum_{i=1}^n |x_i - y_i|^p$$

It is faster to compute that distance, so TakeRoot is by default off. However, when TakeRoot is false, the distance given is not actually a true metric – it does not satisfy the triangle inequality. Some MLPACK methods do not require the triangle inequality to operate correctly (such as the BinarySpaceTree), but setting TakeRoot = false in some cases will cause incorrect results.

A few convenience typedefs are given:

- ManhattanDistance
- EuclideanDistance
- SquaredEuclideanDistance

Template Parameters

<i>Power</i>	Power of metric; i.e. Power = 1 gives the L1-norm (Manhattan distance).
<i>TakeRoot</i>	If true, the Power'th root of the result is taken before it is returned. Setting this to false causes the metric to not satisfy the Triangle Inequality (be careful!).

Definition at line 65 of file `lmetric.hpp`.

22.84.2 Constructor & Destructor Documentation

22.84.2.1 `template<int Power, bool TakeRoot = true> mpack::metric::LMetric< Power, TakeRoot >::LMetric ()`
`[inline]`

Definition at line 72 of file `lmetric.hpp`.

22.84.3 Member Function Documentation

22.84.3.1 `template<int Power, bool TakeRoot = true> template<typename VecType1 , typename VecType2 > static double`
`mpack::metric::LMetric< Power, TakeRoot >::Evaluate (const VecType1 & a, const VecType2 & b) [static]`

Computes the distance between two points.

Referenced by `mpack::kernel::SphericalKernel::ConvolutionIntegral()`, `mpack::kernel::GaussianKernel::ConvolutionIntegral()`, `mpack::kernel::SphericalKernel::Evaluate()`, `mpack::kernel::TriangularKernel::Evaluate()`, `mpack::kernel::LaplacianKernel::Evaluate()`, and `mpack::kernel::GaussianKernel::Evaluate()`.

22.84.3.2 `template<int Power, bool TakeRoot = true> std::string mpack::metric::LMetric< Power, TakeRoot >::ToString ()`
`const`

The documentation for this class was generated from the following file:

- `src/mlpack/core/metrics/lmetric.hpp`

22.85 `mpack::metric::MahalanobisDistance< TakeRoot >` Class Template Reference

The Mahalanobis distance, which is essentially a stretched Euclidean distance.

Public Member Functions

- **MahalanobisDistance** ()
Initialize the Mahalanobis distance with the empty matrix as covariance.
- **MahalanobisDistance** (const size_t dimensionality)
Initialize the Mahalanobis distance with the identity matrix of the given dimensionality.
- **MahalanobisDistance** (const arma::mat &covariance)
Initialize the Mahalanobis distance with the given covariance matrix.
- const arma::mat & **Covariance** () const
Access the covariance matrix.
- arma::mat & **Covariance** ()

Modify the covariance matrix.

- `template<typename VecType1 , typename VecType2 >`
`double Evaluate (const VecType1 &a, const VecType2 &b)`
- `std::string ToString () const`

Evaluate the distance between the two given points using this Mahalanobis distance.

Private Attributes

- `arma::mat covariance`

The covariance matrix associated with this distance.

22.85.1 Detailed Description

```
template<bool TakeRoot = true>class mlpack::metric::MahalanobisDistance< TakeRoot >
```

The Mahalanobis distance, which is essentially a stretched Euclidean distance.

Given a square covariance matrix Q of size $d \times d$, where d is the dimensionality of the points it will be evaluating, and given two vectors x and y also of dimensionality d ,

$$d(x, y) = \sqrt{(x - y)^T Q (x - y)}$$

where Q is the covariance matrix.

Because each evaluation multiplies $(x_1 - x_2)$ by the covariance matrix, it may be much quicker to use an **LMetric** (p. 367) and simply stretch the actual dataset itself before performing any evaluations. However, this class is provided for convenience.

Similar to the **LMetric** (p. 367) class, this offers a template parameter `TakeRoot` which, when set to false, will instead evaluate the distance

$$d(x, y) = (x - y)^T Q (x - y)$$

which is faster to evaluate.

Template Parameters

<i>TakeRoot</i>	If true, takes the root of the output. It is slightly faster to leave this at the default of false, but this means the metric may not satisfy the triangle inequality and may not be usable for methods that expect a true metric.
-----------------	--

Definition at line 55 of file `mahalanobis_distance.hpp`.

22.85.2 Constructor & Destructor Documentation

22.85.2.1 `template<bool TakeRoot = true> mlpack::metric::MahalanobisDistance< TakeRoot >::MahalanobisDistance () [inline]`

Initialize the Mahalanobis distance with the empty matrix as covariance.

Don't call **Evaluate()** (p. 371) until you set the covariance with **Covariance()** (p. 371)!

Definition at line 62 of file `mahalanobis_distance.hpp`.

22.85.2.2 `template<bool TakeRoot = true> mlpack::metric::MahalanobisDistance<TakeRoot >::MahalanobisDistance
(const size_t dimensionality) [inline]`

Initialize the Mahalanobis distance with the identity matrix of the given dimensionality.

Parameters

<i>dimensionality</i>	Dimesnsionality of the covariance matrix.
-----------------------	---

Definition at line 70 of file `mahalanobis_distance.hpp`.

22.85.2.3 `template<bool TakeRoot = true> mlpack::metric::MahalanobisDistance< TakeRoot >::MahalanobisDistance (const arma::mat & covariance) [inline]`

Initialize the Mahalanobis distance with the given covariance matrix.

The given covariance matrix will be copied (this is not optimal).

Parameters

<i>covariance</i>	The covariance matrix to use for this distance.
-------------------	---

Definition at line 79 of file `mahalanobis_distance.hpp`.

22.85.3 Member Function Documentation

22.85.3.1 `template<bool TakeRoot = true> const arma::mat& mlpack::metric::MahalanobisDistance< TakeRoot >::Covariance () const [inline]`

Access the covariance matrix.

Returns

Constant reference to the covariance matrix.

Definition at line 101 of file `mahalanobis_distance.hpp`.

References `mlpack::metric::MahalanobisDistance< TakeRoot >::covariance`.

22.85.3.2 `template<bool TakeRoot = true> arma::mat& mlpack::metric::MahalanobisDistance< TakeRoot >::Covariance () [inline]`

Modify the covariance matrix.

Returns

Reference to the covariance matrix.

Definition at line 108 of file `mahalanobis_distance.hpp`.

References `mlpack::metric::MahalanobisDistance< TakeRoot >::covariance`.

22.85.3.3 `template<bool TakeRoot = true> template<typename VecType1 , typename VecType2 > double mlpack::metric::MahalanobisDistance< TakeRoot >::Evaluate (const VecType1 & a, const VecType2 & b)`

22.85.3.4 `template<bool TakeRoot = true> std::string mlpack::metric::MahalanobisDistance< TakeRoot >::ToString () const`

Evaluate the distance between the two given points using this Mahalanobis distance.

If the covariance matrix has not been set (i.e. if you used the empty constructor and did not later modify the covariance matrix), calling this method will probably result in a crash.

Parameters

<i>a</i>	First vector.
<i>b</i>	Second vector.

22.85.4 Member Data Documentation

22.85.4.1 `template<bool TakeRoot = true> arma::mat mlpack::metric::MahalanobisDistance< TakeRoot >::covariance [private]`

The covariance matrix associated with this distance.

Definition at line 111 of file mahalanobis_distance.hpp.

Referenced by `mlpack::metric::MahalanobisDistance< TakeRoot >::Covariance()`.

The documentation for this class was generated from the following file:

- `src/mlpack/core/metrics/mahalanobis_distance.hpp`

22.86 mlpack::mvu::MVU Class Reference

The **MVU** (p. 372) class is meant to provide a good abstraction for users.

Public Member Functions

- **MVU** (const arma::mat &dataIn)
- void **Unfold** (const size_t newDim, const size_t numNeighbors, arma::mat &outputCoordinates)

Private Attributes

- const arma::mat & **data**

22.86.1 Detailed Description

The **MVU** (p. 372) class is meant to provide a good abstraction for users.

The dataset needs to be provided, as well as several parameters.

- dataset
- new dimensionality

Definition at line 34 of file mvu.hpp.

22.86.2 Constructor & Destructor Documentation

22.86.2.1 `mlpack::mvu::MVU::MVU (const arma::mat & dataIn)`

22.86.3 Member Function Documentation

22.86.3.1 void mlpack::mvu::MVU::Unfold (const size_t *newDim*, const size_t *numNeighbors*, arma::mat & *outputCoordinates*)

22.86.4 Member Data Documentation

22.86.4.1 const arma::mat& mlpack::mvu::MVU::data [private]

Definition at line 44 of file mvu.hpp.

The documentation for this class was generated from the following file:

- src/mlpack/methods/mvu/**mvu.hpp**

22.87 mlpack::naive_bayes::NaiveBayesClassifier< MatType > Class Template Reference

The simple Naive Bayes classifier.

Public Member Functions

- **NaiveBayesClassifier** (const MatType &data, const arma::Col< size_t > &labels, const size_t classes, const bool incrementalVariance=false)
Initializes the classifier as per the input and then trains it by calculating the sample mean and variances.
- void **Classify** (const MatType &data, arma::Col< size_t > &results)
Given a bunch of data points, this function evaluates the class of each of those data points, and puts it in the vector 'results'.
- const MatType & **Means** () const
Get the sample means for each class.
- MatType & **Means** ()
Modify the sample means for each class.
- const arma::vec & **Probabilities** () const
Get the prior probabilities for each class.
- arma::vec & **Probabilities** ()
Modify the prior probabilities for each class.
- const MatType & **Variances** () const
Get the sample variances for each class.
- MatType & **Variances** ()
Modify the sample variances for each class.

Private Attributes

- MatType **means**
Sample mean for each class.
- arma::vec **probabilities**
Class probabilities.
- MatType **variances**
Sample variances for each class.

22.87.1 Detailed Description

`template<typename MatType = arma::mat> class mlpack::naive_bayes::NaiveBayesClassifier< MatType >`

The simple Naive Bayes classifier.

This class trains on the data by calculating the sample mean and variance of the features with respect to each of the labels, and also the class probabilities. The class labels are assumed to be positive integers (starting with 0), and are expected to be the last row of the data input to the constructor.

Mathematically, it computes $P(X_i = x_i \mid Y = y_j)$ for each feature X_i for each of the labels y_j . Alongwith this, it also computes the class probabilities $P(Y = y_j)$.

For classifying a data point (x_1, x_2, \dots, x_n) , it computes the following: $\arg \max_y (P(Y = y) * P(X_1 = x_1 \mid Y = y) * \dots * P(X_n = x_n \mid Y = y))$

Example use:

```
extern arma::mat training_data, testing_data;
NaiveBayesClassifier<> nbc(training_data, 5);
arma::vec results;

nbc.Classify(testing_data, results);
```

Definition at line 50 of file `naive_bayes_classifier.hpp`.

22.87.2 Constructor & Destructor Documentation

22.87.2.1 `template<typename MatType = arma::mat> mlpack::naive_bayes::NaiveBayesClassifier< MatType >::NaiveBayesClassifier(const MatType & data, const arma::Col< size_t > & labels, const size_t classes, const bool incrementalVariance = false)`

Initializes the classifier as per the input and then trains it by calculating the sample mean and variances.

The input data is expected to have integer labels as the last row (starting with 0 and not greater than the number of classes).

Example use:

```
extern arma::mat training_data, testing_data;
extern arma::Col<size_t> labels;
NaiveBayesClassifier nbc(training_data, labels, 5);
```

Parameters

<i>data</i>	Training data points.
<i>labels</i>	Labels corresponding to training data points.
<i>classes</i>	Number of classes in this classifier.
<i>incrementalVariance</i>	If true, an incremental algorithm is used to calculate the variance; this can prevent loss of precision in some cases, but will be somewhat slower to calculate.

22.87.3 Member Function Documentation

22.87.3.1 `template<typename MatType = arma::mat> void mlpack::naive_bayes::NaiveBayesClassifier< MatType >::Classify(const MatType & data, arma::Col< size_t > & results)`

Given a bunch of data points, this function evaluates the class of each of those data points, and puts it in the vector 'results'.

```
arma::mat test_data; // each column is a test point
arma::Col<size_t> results;
...
nbc.Classify(test_data, &results);
```

Parameters

<i>data</i>	List of data points.
<i>results</i>	Vector that class predictions will be placed into.

22.87.3.2 `template<typename MatType = arma::mat> const MatType& mlpack::naive_bayes::NaiveBayesClassifier< MatType >::Means () const [inline]`

Get the sample means for each class.

Definition at line 105 of file naive_bayes_classifier.hpp.

References mlpack::naive_bayes::NaiveBayesClassifier< MatType >::means.

22.87.3.3 `template<typename MatType = arma::mat> MatType& mlpack::naive_bayes::NaiveBayesClassifier< MatType >::Means () [inline]`

Modify the sample means for each class.

Definition at line 107 of file naive_bayes_classifier.hpp.

References mlpack::naive_bayes::NaiveBayesClassifier< MatType >::means.

22.87.3.4 `template<typename MatType = arma::mat> const arma::vec& mlpack::naive_bayes::NaiveBayesClassifier< MatType >::Probabilities () const [inline]`

Get the prior probabilities for each class.

Definition at line 115 of file naive_bayes_classifier.hpp.

References mlpack::naive_bayes::NaiveBayesClassifier< MatType >::probabilities.

22.87.3.5 `template<typename MatType = arma::mat> arma::vec& mlpack::naive_bayes::NaiveBayesClassifier< MatType >::Probabilities () [inline]`

Modify the prior probabilities for each class.

Definition at line 117 of file naive_bayes_classifier.hpp.

References mlpack::naive_bayes::NaiveBayesClassifier< MatType >::probabilities.

22.87.3.6 `template<typename MatType = arma::mat> const MatType& mlpack::naive_bayes::NaiveBayesClassifier< MatType >::Variances () const [inline]`

Get the sample variances for each class.

Definition at line 110 of file naive_bayes_classifier.hpp.

References mlpack::naive_bayes::NaiveBayesClassifier< MatType >::variances.

22.87.3.7 `template<typename MatType = arma::mat> MatType& mlpack::naive_bayes::NaiveBayesClassifier< MatType >::Variances () [inline]`

Modify the sample variances for each class.

Definition at line 112 of file `naive_bayes_classifier.hpp`.

References `mlpack::naive_bayes::NaiveBayesClassifier< MatType >::variances`.

22.87.4 Member Data Documentation

22.87.4.1 `template<typename MatType = arma::mat> MatType mlpack::naive_bayes::NaiveBayesClassifier< MatType >::means [private]`

Sample mean for each class.

Definition at line 54 of file `naive_bayes_classifier.hpp`.

Referenced by `mlpack::naive_bayes::NaiveBayesClassifier< MatType >::Means()`.

22.87.4.2 `template<typename MatType = arma::mat> arma::vec mlpack::naive_bayes::NaiveBayesClassifier< MatType >::probabilities [private]`

Class probabilities.

Definition at line 60 of file `naive_bayes_classifier.hpp`.

Referenced by `mlpack::naive_bayes::NaiveBayesClassifier< MatType >::Probabilities()`.

22.87.4.3 `template<typename MatType = arma::mat> MatType mlpack::naive_bayes::NaiveBayesClassifier< MatType >::variances [private]`

Sample variances for each class.

Definition at line 57 of file `naive_bayes_classifier.hpp`.

Referenced by `mlpack::naive_bayes::NaiveBayesClassifier< MatType >::Variances()`.

The documentation for this class was generated from the following file:

- `src/mlpack/methods/naive_bayes/naive_bayes_classifier.hpp`

22.88 `mlpack::nca::NCA< MetricType, OptimizerType >` Class Template Reference

An implementation of Neighborhood Components Analysis, both a linear dimensionality reduction technique and a distance learning technique.

Public Member Functions

- **NCA** (const arma::mat &**dataset**, const arma::Col< size_t > &**labels**, MetricType **metric**=MetricType())
Construct the Neighborhood Components Analysis object.
- const arma::mat & **Dataset** () const
Get the dataset reference.

- const arma::Col< size_t > & **Labels** () const
Get the labels reference.
- void **LearnDistance** (arma::mat &outputMatrix)
Perform Neighborhood Components Analysis.
- const OptimizerType< **SoftmaxErrorFunction**< MetricType > > & **Optimizer** () const
Get the optimizer.
- OptimizerType< **SoftmaxErrorFunction**< MetricType > > & **Optimizer** ()
- std::string **ToString** () const

Private Attributes

- const arma::mat & **dataset**
Dataset reference.
- **SoftmaxErrorFunction**< MetricType > **errorFunction**
The function to optimize.
- const arma::Col< size_t > & **labels**
Labels reference.
- MetricType **metric**
Metric to be used.
- OptimizerType< **SoftmaxErrorFunction**< MetricType > > **optimizer**
The optimizer to use.

22.88.1 Detailed Description

```
template<typename MetricType = metric::SquaredEuclideanDistance, template< typename > class OptimizerType = optimization::SGD>class mlpack::nca::NCA< MetricType, OptimizerType >
```

An implementation of Neighborhood Components Analysis, both a linear dimensionality reduction technique and a distance learning technique.

The method seeks to improve k-nearest-neighbor classification on a dataset by scaling the dimensions. The method is nonparametric, and does not require a value of k. It works by using stochastic ("soft") neighbor assignments and using optimization techniques over the gradient of the accuracy of the neighbor assignments.

For more details, see the following published paper:

```
@inproceedings{Goldberger2004,
  author = {Goldberger, Jacob and Roweis, Sam and Hinton, Geoff and
    Salakhutdinov, Ruslan},
  booktitle = {Advances in Neural Information Processing Systems 17},
  pages = {513--520},
  publisher = {MIT Press},
  title = {{Neighbourhood Components Analysis}},
  year = {2004}
}
```

Definition at line 51 of file nca.hpp.

22.88.2 Constructor & Destructor Documentation

```
22.88.2.1  template<typename MetricType = metric::SquaredEuclideanDistance, template< typename > class OptimizerType =
           optimization::SGD> mlpack::nca::NCA< MetricType, OptimizerType >::NCA ( const arma::mat & dataset, const
           arma::Col< size_t > & labels, MetricType metric =MetricType() )
```

Construct the Neighborhood Components Analysis object.

This simply stores the reference to the dataset and labels as well as the parameters for optimization before the actual optimization is performed.

Parameters

<i>dataset</i>	Input dataset.
<i>labels</i>	Input dataset labels.
<i>stepSize</i>	Step size for stochastic gradient descent.
<i>maxIterations</i>	Maximum iterations for stochastic gradient descent.
<i>tolerance</i>	Tolerance for termination of stochastic gradient descent.
<i>shuffle</i>	Whether or not to shuffle the dataset during SGD.
<i>metric</i>	Instantiated metric to use.

22.88.3 Member Function Documentation

```
22.88.3.1  template<typename MetricType = metric::SquaredEuclideanDistance, template< typename > class OptimizerType =
           optimization::SGD> const arma::mat& mlpack::nca::NCA< MetricType, OptimizerType >::Dataset ( ) const
           [inline]
```

Get the dataset reference.

Definition at line 83 of file nca.hpp.

References mlpack::nca::NCA< MetricType, OptimizerType >::dataset.

```
22.88.3.2  template<typename MetricType = metric::SquaredEuclideanDistance, template< typename > class OptimizerType =
           optimization::SGD> const arma::Col<size_t>& mlpack::nca::NCA< MetricType, OptimizerType >::Labels ( ) const
           [inline]
```

Get the labels reference.

Definition at line 85 of file nca.hpp.

References mlpack::nca::NCA< MetricType, OptimizerType >::labels.

```
22.88.3.3  template<typename MetricType = metric::SquaredEuclideanDistance, template< typename > class OptimizerType
           = optimization::SGD> void mlpack::nca::NCA< MetricType, OptimizerType >::LearnDistance ( arma::mat &
           outputMatrix )
```

Perform Neighborhood Components Analysis.

The output distance learning matrix is written into the passed reference. If **LearnDistance()** (p. 378) is called with an outputMatrix which has the correct size (dataset.n_rows x dataset.n_rows), that matrix will be used as the starting point for optimization.

Parameters

<i>output_matrix</i>	Covariance matrix of Mahalanobis distance.
----------------------	--

22.88.3.4 `template<typename MetricType = metric::SquaredEuclideanDistance, template< typename > class OptimizerType = optimization::SGD> const OptimizerType<SoftmaxErrorFunction<MetricType>> & mlpack::nca::NCA< MetricType, OptimizerType >::Optimizer () const` `[inline]`

Get the optimizer.

Definition at line 88 of file nca.hpp.

References mlpack::nca::NCA< MetricType, OptimizerType >::optimizer.

22.88.3.5 `template<typename MetricType = metric::SquaredEuclideanDistance, template< typename > class OptimizerType = optimization::SGD> OptimizerType<SoftmaxErrorFunction<MetricType>> & mlpack::nca::NCA< MetricType, OptimizerType >::Optimizer ()` `[inline]`

Definition at line 90 of file nca.hpp.

References mlpack::nca::NCA< MetricType, OptimizerType >::optimizer.

22.88.3.6 `template<typename MetricType = metric::SquaredEuclideanDistance, template< typename > class OptimizerType = optimization::SGD> std::string mlpack::nca::NCA< MetricType, OptimizerType >::ToString () const`

22.88.4 Member Data Documentation

22.88.4.1 `template<typename MetricType = metric::SquaredEuclideanDistance, template< typename > class OptimizerType = optimization::SGD> const arma::mat& mlpack::nca::NCA< MetricType, OptimizerType >::dataset` `[private]`

Dataset reference.

Definition at line 98 of file nca.hpp.

Referenced by mlpack::nca::NCA< MetricType, OptimizerType >::Dataset().

22.88.4.2 `template<typename MetricType = metric::SquaredEuclideanDistance, template< typename > class OptimizerType = optimization::SGD> SoftmaxErrorFunction<MetricType> mlpack::nca::NCA< MetricType, OptimizerType >::errorFunction` `[private]`

The function to optimize.

Definition at line 106 of file nca.hpp.

22.88.4.3 `template<typename MetricType = metric::SquaredEuclideanDistance, template< typename > class OptimizerType = optimization::SGD> const arma::Col<size_t> & mlpack::nca::NCA< MetricType, OptimizerType >::labels` `[private]`

Labels reference.

Definition at line 100 of file nca.hpp.

Referenced by mlpack::nca::NCA< MetricType, OptimizerType >::Labels().

22.88.4.4 `template<typename MetricType = metric::SquaredEuclideanDistance, template< typename > class OptimizerType = optimization::SGD> MetricType mlpack::nca::NCA< MetricType, OptimizerType >::metric [private]`

Metric to be used.

Definition at line 103 of file nca.hpp.

22.88.4.5 `template<typename MetricType = metric::SquaredEuclideanDistance, template< typename > class OptimizerType = optimization::SGD> OptimizerType<SoftmaxErrorFunction<MetricType> > mlpack::nca::NCA< MetricType, OptimizerType >::optimizer [private]`

The optimizer to use.

Definition at line 109 of file nca.hpp.

Referenced by `mlpack::nca::NCA< MetricType, OptimizerType >::Optimizer()`.

The documentation for this class was generated from the following file:

- `src/mlpack/methods/nca/nca.hpp`

22.89 mlpack::nca::SoftmaxErrorFunction< MetricType > Class Template Reference

The "softmax" stochastic neighbor assignment probability function.

Public Member Functions

- **SoftmaxErrorFunction** (const arma::mat &dataset, const arma::Col< size_t > &labels, MetricType metric=MetricType())
Initialize with the given kernel; useful when the kernel has some state to store, which is set elsewhere.
- double **Evaluate** (const arma::mat &covariance)
Evaluate the softmax function for the given covariance matrix.
- double **Evaluate** (const arma::mat &covariance, const size_t i)
Evaluate the softmax objective function for the given covariance matrix on only one point of the dataset.
- const arma::mat **GetInitialPoint** () const
Get the initial point.
- void **Gradient** (const arma::mat &covariance, arma::mat &gradient)
Evaluate the gradient of the softmax function for the given covariance matrix.
- void **Gradient** (const arma::mat &covariance, const size_t i, arma::mat &gradient)
Evaluate the gradient of the softmax function for the given covariance matrix on only one point of the dataset.
- size_t **NumFunctions** () const
Get the number of functions the objective function can be decomposed into.
- std::string **ToString** () const

Private Member Functions

- void **Precalculate** (const arma::mat &coordinates)
*Precalculate the denominators and numerators that will make up the p_{ij} , but only if the coordinates matrix is different than the last coordinates the **Precalculate()** (p. 383) method was run with.*

Private Attributes

- const arma::mat & **dataset**
The dataset.
- arma::vec **denominators**
*Holds denominators for calculation of p_{ij} , for the non-separable **Evaluate()** (p. 382) and **Gradient()** (p. 382).*
- const arma::Col< size_t > & **labels**
Labels for each point in the dataset.
- arma::mat **lastCoordinates**
*Last coordinates. Used for the non-separable **Evaluate()** (p. 382) and **Gradient()** (p. 382).*
- MetricType **metric**
The instantiated metric.
- arma::vec **p**
*Holds calculated p_i , for the non-separable **Evaluate()** (p. 382) and **Gradient()** (p. 382).*
- bool **precalculated**
False if nothing has ever been precalculated (only at construction time).
- arma::mat **stretchedDataset**
Stretched dataset. Kept internal to avoid memory reallocations.

22.89.1 Detailed Description

```
template<typename MetricType = metric::SquaredEuclideanDistance> class mlpack::nca::SoftmaxErrorFunction< MetricType >
```

The "softmax" stochastic neighbor assignment probability function.

The actual function is

$$p_{ij} = (\exp(-\|A x_i - A x_j\|^2)) / (\sum_{k \neq i} (\exp(-\|A x_i - A x_k\|^2)))$$

where x_n represents a point and A is the current scaling matrix.

This class is more flexible than the original paper, allowing an arbitrary metric function to be used in place of $\|A x_i - A x_j\|^2$, meaning that the squared Euclidean distance is not the only allowed metric for **NCA** (p. 376). However, that is probably the best way to use this class.

In addition to the standard **Evaluate()** (p. 382) and **Gradient()** (p. 382) functions which MLPACK optimizers use, overloads of **Evaluate()** (p. 382) and **Gradient()** (p. 382) are given which only operate on one point in the dataset. This is useful for optimizers like stochastic gradient descent (see **mlpack::optimization::SGD** (p. 485)).

Definition at line 44 of file `nca_softmax_error_function.hpp`.

22.89.2 Constructor & Destructor Documentation

22.89.2.1 `template<typename MetricType = metric::SquaredEuclideanDistance> mlpack::nca::SoftmaxErrorFunction< MetricType >::SoftmaxErrorFunction (const arma::mat & dataset, const arma::Col< size_t > & labels, MetricType metric = MetricType())`

Initialize with the given kernel; useful when the kernel has some state to store, which is set elsewhere.

If no kernel is given, an empty kernel is used; this way, you can call the constructor with no arguments. A reference to the dataset we will be optimizing over is also required.

Parameters

<i>dataset</i>	Matrix containing the dataset.
<i>labels</i>	Vector of class labels for each point in the dataset.
<i>kernel</i>	Instantiated kernel (optional).

22.89.3 Member Function Documentation

22.89.3.1 `template<typename MetricType = metric::SquaredEuclideanDistance> double mlpack::nca::SoftmaxErrorFunction< MetricType >::Evaluate (const arma::mat & covariance)`

Evaluate the softmax function for the given covariance matrix.

This is the non-separable implementation, where the objective function is not decomposed into the sum of several objective functions.

Parameters

<i>covariance</i>	Covariance matrix of Mahalanobis distance.
-------------------	--

22.89.3.2 `template<typename MetricType = metric::SquaredEuclideanDistance> double mlpack::nca::SoftmaxErrorFunction< MetricType >::Evaluate (const arma::mat & covariance, const size_t i)`

Evaluate the softmax objective function for the given covariance matrix on only one point of the dataset.

This is the separable implementation, where the objective function is decomposed into the sum of many objective functions, and here, only one of those constituent objective functions is returned.

Parameters

<i>covariance</i>	Covariance matrix of Mahalanobis distance.
<i>i</i>	Index of point to use for objective function.

22.89.3.3 `template<typename MetricType = metric::SquaredEuclideanDistance> const arma::mat mlpack::nca::SoftmaxErrorFunction< MetricType >::GetInitialPoint () const`

Get the initial point.

22.89.3.4 `template<typename MetricType = metric::SquaredEuclideanDistance> void mlpack::nca::SoftmaxErrorFunction< MetricType >::Gradient (const arma::mat & covariance, arma::mat & gradient)`

Evaluate the gradient of the softmax function for the given covariance matrix.

This is the non-separable implementation, where the objective function is not decomposed into the sum of several objective functions.

Parameters

<i>covariance</i>	Covariance matrix of Mahalanobis distance.
<i>gradient</i>	Matrix to store the calculated gradient in.

22.89.3.5 `template<typename MetricType = metric::SquaredEuclideanDistance> void mlpack::nca::SoftmaxErrorFunction< MetricType >::Gradient (const arma::mat & covariance, const size_t i, arma::mat & gradient)`

Evaluate the gradient of the softmax function for the given covariance matrix on only one point of the dataset.

This is the separable implementation, where the objective function is decomposed into the sum of many objective functions, and here, only one of those constituent objective functions is returned.

Parameters

<i>covariance</i>	Covariance matrix of Mahalanobis distance.
<i>i</i>	Index of point to use for objective function.
<i>gradient</i>	Matrix to store the calculated gradient in.

22.89.3.6 `template<typename MetricType = metric::SquaredEuclideanDistance> size_t mlpack::nca::SoftmaxErrorFunction< MetricType >::NumFunctions () const [inline]`

Get the number of functions the objective function can be decomposed into.

This is just the number of points in the dataset.

Definition at line 116 of file `nca_softmax_error_function.hpp`.

22.89.3.7 `template<typename MetricType = metric::SquaredEuclideanDistance> void mlpack::nca::SoftmaxErrorFunction< MetricType >::Precalculate (const arma::mat & coordinates) [private]`

Precalculate the denominators and numerators that will make up the p_{ij} , but only if the coordinates matrix is different than the last coordinates the **Precalculate()** (p. 383) method was run with.

This method is only called by the non-separable **Evaluate()** (p. 382) and **Gradient()** (p. 382).

This will update `last_coordinates_` and `stretched_dataset_`, and also calculate the p_i and `denominators_` which are used in the calculation of p_i or p_{ij} . The calculation will be $O((n * (n + 1)) / 2)$, which is not great.

Parameters

<i>coordinates</i>	Coordinates matrix to use for precalculation.
--------------------	---

22.89.3.8 `template<typename MetricType = metric::SquaredEuclideanDistance> std::string mlpack::nca::SoftmaxErrorFunction< MetricType >::ToString () const`

22.89.4 Member Data Documentation

22.89.4.1 `template<typename MetricType = metric::SquaredEuclideanDistance> const arma::mat& mlpack::nca::SoftmaxErrorFunction< MetricType >::dataset [private]`

The dataset.

Definition at line 123 of file `nca_softmax_error_function.hpp`.

22.89.4.2 `template<typename MetricType = metric::SquaredEuclideanDistance> arma::vec mlpack::nca::SoftmaxErrorFunction< MetricType >::denominators [private]`

Holds denominators for calculation of p_{ij} , for the non-separable **Evaluate()** (p. 382) and **Gradient()** (p. 382).

Definition at line 138 of file `nca_softmax_error_function.hpp`.

22.89.4.3 `template<typename MetricType = metric::SquaredEuclideanDistance> const arma::Col<size_t>& mlpack::nca::SoftmaxErrorFunction< MetricType >::labels [private]`

Labels for each point in the dataset.

Definition at line 125 of file `nca_softmax_error_function.hpp`.

22.89.4.4 `template<typename MetricType = metric::SquaredEuclideanDistance> arma::mat mlpack::nca::SoftmaxErrorFunction< MetricType >::lastCoordinates [private]`

Last coordinates. Used for the non-separable **Evaluate()** (p. 382) and **Gradient()** (p. 382).

Definition at line 131 of file `nca_softmax_error_function.hpp`.

22.89.4.5 `template<typename MetricType = metric::SquaredEuclideanDistance> MetricType mlpack::nca::SoftmaxErrorFunction< MetricType >::metric [private]`

The instantiated metric.

Definition at line 128 of file `nca_softmax_error_function.hpp`.

22.89.4.6 `template<typename MetricType = metric::SquaredEuclideanDistance> arma::vec mlpack::nca::SoftmaxErrorFunction< MetricType >::p [private]`

Holds calculated p_i , for the non-separable **Evaluate()** (p. 382) and **Gradient()** (p. 382).

Definition at line 135 of file `nca_softmax_error_function.hpp`.

22.89.4.7 `template<typename MetricType = metric::SquaredEuclideanDistance> bool mlpack::nca::SoftmaxErrorFunction< MetricType >::precalculated [private]`

False if nothing has ever been precalculated (only at construction time).

Definition at line 141 of file `nca_softmax_error_function.hpp`.

22.89.4.8 `template<typename MetricType = metric::SquaredEuclideanDistance> arma::mat mlpack::nca::SoftmaxErrorFunction< MetricType >::stretchedDataset [private]`

Stretched dataset. Kept internal to avoid memory reallocations.

Definition at line 133 of file `nca_softmax_error_function.hpp`.

The documentation for this class was generated from the following file:

- `src/mlpack/methods/nca/nca_softmax_error_function.hpp`

22.90 mlpack::neighbor::FurthestNeighborSort Class Reference

This class implements the necessary methods for the SortPolicy template parameter of the **NeighborSearch** (p. 399) class.

Static Public Member Functions

- static double **BestDistance** ()
Return what should represent the best possible distance with this particular sort policy.
- template<typename TreeType >
static double **BestNodeToNodeDistance** (const TreeType *queryNode, const TreeType *referenceNode)
Return the best possible distance between two nodes.
- template<typename TreeType >
static double **BestNodeToNodeDistance** (const TreeType *queryNode, const TreeType *referenceNode, const double centerToCenterDistance)
Return the best possible distance between two nodes, given that the distance between the centers of the two nodes has already been calculated.
- template<typename TreeType >
static double **BestNodeToNodeDistance** (const TreeType *queryNode, const TreeType *referenceNode, const TreeType *referenceChildNode, const double centerToCenterDistance)
Return the best possible distance between the query node and the reference child node given the base case distance between the query node and the reference node.
- template<typename VecType, typename TreeType >
static double **BestPointToNodeDistance** (const VecType &queryPoint, const TreeType *referenceNode)
Return the best possible distance between a node and a point.
- template<typename VecType, typename TreeType >
static double **BestPointToNodeDistance** (const VecType &queryPoint, const TreeType *referenceNode, const double pointToCenterDistance)
Return the best possible distance between a point and a node, given that the distance between the point and the center of the node has already been calculated.
- static double **CombineBest** (const double a, const double b)
Return the best combination of the two distances.
- static double **CombineWorst** (const double a, const double b)
Return the worst combination of the two distances.
- static bool **IsBetter** (const double value, const double ref)
Return whether or not value is "better" than ref.
- static size_t **SortDistance** (const arma::vec &list, const arma::Col< size_t > &indices, double newDistance)
Return the index in the vector where the new distance should be inserted, or size_t() - 1 if it should not be inserted (i.e.
- static double **WorstDistance** ()
Return what should represent the worst possible distance with this particular sort policy.

22.90.1 Detailed Description

This class implements the necessary methods for the SortPolicy template parameter of the **NeighborSearch** (p. 399) class.

The sorting policy here is that the minimum distance is the best (so, when used with **NeighborSearch** (p. 399), the output is furthest neighbors).

Definition at line 29 of file furthest_neighbor_sort.hpp.

22.90.2 Member Function Documentation

22.90.2.1 `static double mlpack::neighbor::FurthestNeighborSort::BestDistance () [inline],[static]`

Return what should represent the best possible distance with this particular sort policy.

In our case, this should be the maximum possible distance, DBL_MAX.

Returns

DBL_MAX

Definition at line 138 of file `furthest_neighbor_sort.hpp`.

22.90.2.2 `template<typename TreeType > static double mlpack::neighbor::FurthestNeighborSort::BestNodeToNodeDistance (const TreeType * queryNode, const TreeType * referenceNode) [static]`

Return the best possible distance between two nodes.

In our case, this is the maximum distance between the two tree nodes using the given distance function.

22.90.2.3 `template<typename TreeType > static double mlpack::neighbor::FurthestNeighborSort::BestNodeToNodeDistance (const TreeType * queryNode, const TreeType * referenceNode, const double centerToCenterDistance) [static]`

Return the best possible distance between two nodes, given that the distance between the centers of the two nodes has already been calculated.

This is used in conjunction with trees that have self-children (like cover trees).

22.90.2.4 `template<typename TreeType > static double mlpack::neighbor::FurthestNeighborSort::BestNodeToNodeDistance (const TreeType * queryNode, const TreeType * referenceNode, const TreeType * referenceChildNode, const double centerToCenterDistance) [static]`

Return the best possible distance between the query node and the reference child node given the base case distance between the query node and the reference node.

`TreeType::ParentDistance()` must be implemented to use this.

Parameters

<i>queryNode</i>	Query node.
<i>referenceNode</i>	Reference node.
<i>referenceChildNode</i>	Child of reference node which is being inspected.
<i>centerToCenterDistance</i>	Distance between centers of query node and reference node.

22.90.2.5 `template<typename VecType , typename TreeType > static double mlpack::neighbor::FurthestNeighborSort::BestPointToNodeDistance (const VecType & queryPoint, const TreeType * referenceNode) [static]`

Return the best possible distance between a node and a point.

In our case, this is the maximum distance between the tree node and the point using the given distance function.

22.90.2.6 `template<typename VecType , typename TreeType > static double mpack::neighbor::FurthestNeighborSort::BestPointToNodeDistance (const VecType & queryPoint, const TreeType * referenceNode, const double pointToCenterDistance) [static]`

Return the best possible distance between a point and a node, given that the distance between the point and the center of the node has already been calculated.

This is used in conjunction with trees that have self-children (like cover trees).

22.90.2.7 `static double mpack::neighbor::FurthestNeighborSort::CombineBest (const double a, const double b) [inline], [static]`

Return the best combination of the two distances.

Definition at line 143 of file `furthest_neighbor_sort.hpp`.

22.90.2.8 `static double mpack::neighbor::FurthestNeighborSort::CombineWorst (const double a, const double b) [inline], [static]`

Return the worst combination of the two distances.

Definition at line 153 of file `furthest_neighbor_sort.hpp`.

22.90.2.9 `static bool mpack::neighbor::FurthestNeighborSort::IsBetter (const double value, const double ref) [inline], [static]`

Return whether or not *value* is "better" than *ref*.

In this case, that means that the *value* is greater than the reference.

Parameters

<i>value</i>	Value to compare
<i>ref</i>	Value to compare with

Returns

bool indicating whether or not (*value* > *ref*).

Definition at line 59 of file `furthest_neighbor_sort.hpp`.

22.90.2.10 `static size_t mpack::neighbor::FurthestNeighborSort::SortDistance (const arma::vec & list, const arma::Col< size_t > & indices, double newDistance) [static]`

Return the index in the vector where the new distance should be inserted, or `size_t() - 1` if it should not be inserted (i.e. if it is not any better than any of the existing points in the list). The list should be sorted such that the best point is the first in the list. The actual insertion is not performed.

Parameters

<i>list</i>	Vector of existing distance points, sorted such that the best point is the first in the list.
<i>new_distance</i>	Distance to try to insert.

Returns

size_t containing the position to insert into, or (size_t() - 1) if the new distance should not be inserted.

22.90.2.11 static double mlpack::neighbor::FurthestNeighborSort::WorstDistance () [inline],[static]

Return what should represent the worst possible distance with this particular sort policy.

In our case, this should be the minimum possible distance, 0.

Returns

0

Definition at line 129 of file furthest_neighbor_sort.hpp.

The documentation for this class was generated from the following file:

- src/mlpack/methods/neighbor_search/sort_policies/furthest_neighbor_sort.hpp

22.91 mlpack::neighbor::LSHSearch< SortPolicy > Class Template Reference

The **LSHSearch** (p.388) class – This class builds a hash on the reference set and uses this hash to compute the distance-approximate nearest-neighbors of the given queries.

Public Member Functions

- **LSHSearch** (const arma::mat &referenceSet, const arma::mat &querySet, const size_t numProj, const size_t numTables, const double hashWidth=0.0, const size_t secondHashSize=99901, const size_t bucketSize=500)
This function initializes the LSH class.
- **LSHSearch** (const arma::mat &referenceSet, const size_t numProj, const size_t numTables, const double hashWidth=0.0, const size_t secondHashSize=99901, const size_t bucketSize=500)
This function initializes the LSH class.
- void **Search** (const size_t k, arma::Mat< size_t > &resultingNeighbors, arma::mat &distances, const size_t numTablesToSearch=0)
Compute the nearest neighbors and store the output in the given matrices.
- std::string **ToString** () const

Private Member Functions

- double **BaseCase** (const size_t queryIndex, const size_t referenceIndex)
This is a helper function that computes the distance of the query to the neighbor candidates and appropriately stores the best 'k' candidates.
- void **BuildHash** ()
This function builds a hash table with two levels of hashing as presented in the paper.
- void **InsertNeighbor** (const size_t queryIndex, const size_t pos, const size_t neighbor, const double distance)

This is a helper function that efficiently inserts better neighbor candidates into an existing set of neighbor candidates.

- void **ReturnIndicesFromTable** (const size_t queryIndex, arma::uvec &referenceIndices, size_t numTablesToSearch)

This function takes a query and hashes it into each of the hash tables to get keys for the query and then the key is hashed to a bucket of the second hash table and all the points (if any) in those buckets are collected as the potential neighbor candidates.

Private Attributes

- arma::Col< size_t > **bucketContentSize**
The number of elements present in each hash bucket; should be secondHashSize.
- arma::Col< size_t > **bucketRowInHashTable**
For a particular hash value, points to the row in secondHashTable corresponding to this value.
- const size_t **bucketSize**
The bucket size of the second hash.
- arma::mat * **distancePtr**
The pointer to the nearest neighbor distances.
- double **hashWidth**
The hash width.
- **metric::SquaredEuclideanDistance** **metric**
Instantiation of the metric.
- arma::Mat< size_t > * **neighborPtr**
The pointer to the nearest neighbor indices.
- const size_t **numProj**
The number of projections.
- const size_t **numTables**
The number of hash tables.
- arma::mat **offsets**
The list of the offset 'b' for each of the projection for each table.
- std::vector< arma::mat > **projections**
The std::vector containing the projection matrix of each table.
- const arma::mat & **querySet**
Query dataset (may not be given).
- const arma::mat & **referenceSet**
Reference dataset.
- const size_t **secondHashSize**
The big prime representing the size of the second hash.
- arma::Mat< size_t > **secondHashTable**
The final hash table; should be (< secondHashSize) x bucketSize.
- arma::vec **secondHashWeights**
The weights of the second hash.

22.91.1 Detailed Description

```
template<typename SortPolicy = NearestNeighborSort>class mlpack::neighbor::LSHSearch< SortPolicy >
```

The **LSHSearch** (p.388) class – This class builds a hash on the reference set and uses this hash to compute the distance-approximate nearest-neighbors of the given queries.

Template Parameters

<i>SortPolicy</i>	The sort policy for distances; see NearestNeighborSort (p. 395).
-------------------	---

Definition at line 51 of file `lsh_search.hpp`.

22.91.2 Constructor & Destructor Documentation

22.91.2.1 `template<typename SortPolicy = NearestNeighborSort> mpack::neighbor::LSHSearch< SortPolicy >::LSHSearch (const arma::mat & referenceSet, const arma::mat & querySet, const size_t numProj, const size_t numTables, const double hashWidth = 0.0, const size_t secondHashSize = 99901, const size_t bucketSize = 500)`

This function initializes the LSH class.

It builds the hash on the reference set with 2-stable distributions. See the individual functions performing the hashing for details on how the hashing is done.

Parameters

<i>referenceSet</i>	Set of reference points.
<i>querySet</i>	Set of query points.
<i>numProj</i>	Number of projections in each hash table (anything between 10-50 might be a decent choice).
<i>numTables</i>	Total number of hash tables (anything between 10-20 should suffice).
<i>hashWidth</i>	The width of hash for every table. If 0 (the default) is provided, then the hash width is automatically obtained by computing the average pairwise distance of 25 pairs. This should be a reasonable upper bound on the nearest-neighbor distance in general.
<i>secondHashSize</i>	The size of the second hash table. This should be a large prime number.
<i>bucketSize</i>	The size of the bucket in the second hash table. This is the maximum number of points that can be hashed into single bucket. Default values are already provided here.

22.91.2.2 `template<typename SortPolicy = NearestNeighborSort> mpack::neighbor::LSHSearch< SortPolicy >::LSHSearch (const arma::mat & referenceSet, const size_t numProj, const size_t numTables, const double hashWidth = 0.0, const size_t secondHashSize = 99901, const size_t bucketSize = 500)`

This function initializes the LSH class.

It builds the hash on the reference set with 2-stable distributions. See the individual functions performing the hashing for details on how the hashing is done.

Parameters

<i>referenceSet</i>	Set of reference points and the set of queries.
<i>numProj</i>	Number of projections in each hash table (anything between 10-50 might be a decent choice).
<i>numTables</i>	Total number of hash tables (anything between 10-20 should suffice).
<i>hashWidth</i>	The width of hash for every table. If 0 (the default) is provided, then the hash width is automatically obtained by computing the average pairwise distance of 25 pairs. This should be a reasonable upper bound on the nearest-neighbor distance in general.
<i>secondHashSize</i>	The size of the second hash table. This should be a large prime number.
<i>bucketSize</i>	The size of the bucket in the second hash table. This is the maximum number of points that can be hashed into single bucket. Default values are already provided here.

22.91.3 Member Function Documentation

```
22.91.3.1  template<typename SortPolicy = NearestNeighborSort> double mlpack::neighbor::LSHSearch< SortPolicy
           >::BaseCase( const size_t queryIndex, const size_t referenceIndex ) [private]
```

This is a helper function that computes the distance of the query to the neighbor candidates and appropriately stores the best 'k' candidates.

Parameters

<i>queryIndex</i>	The index of the query in question
<i>referenceIndex</i>	The index of the neighbor candidate in question

22.91.3.2 `template<typename SortPolicy = NearestNeighborSort> void mlpack::neighbor::LSHSearch< SortPolicy >::BuildHash () [private]`

This function builds a hash table with two levels of hashing as presented in the paper.

This function first hashes the points with 'numProj' random projections to a single hash table creating (key, point ID) pairs where the key is a 'numProj'-dimensional integer vector.

Then each key in this hash table is hashed into a second hash table using a standard hash.

This function does not have any parameters and relies on parameters which are private members of this class, initialized during the class initialization.

22.91.3.3 `template<typename SortPolicy = NearestNeighborSort> void mlpack::neighbor::LSHSearch< SortPolicy >::InsertNeighbor (const size_t queryIndex, const size_t pos, const size_t neighbor, const double distance) [private]`

This is a helper function that efficiently inserts better neighbor candidates into an existing set of neighbor candidates.

This function is only called by the 'BaseCase' function.

Parameters

<i>queryIndex</i>	This is the index of the query being processed currently
<i>pos</i>	The position of the neighbor candidate in the current list of neighbor candidates.
<i>neighbor</i>	The neighbor candidate that is being inserted into the list of the best 'k' candidates for the query in question.
<i>distance</i>	The distance of the query to the neighbor candidate.

22.91.3.4 `template<typename SortPolicy = NearestNeighborSort> void mlpack::neighbor::LSHSearch< SortPolicy >::ReturnIndicesFromTable (const size_t queryIndex, arma::uvec & referenceIndices, size_t numTablesToSearch) [private]`

This function takes a query and hashes it into each of the hash tables to get keys for the query and then the key is hashed to a bucket of the second hash table and all the points (if any) in those buckets are collected as the potential neighbor candidates.

Parameters

<i>queryIndex</i>	The index of the query currently being processed.
<i>referenceIndices</i>	The list of neighbor candidates obtained from hashing the query into all the hash tables and eventually into multiple buckets of the second hash table.

22.91.3.5 `template<typename SortPolicy = NearestNeighborSort> void mlpack::neighbor::LSHSearch< SortPolicy >::Search (const size_t k, arma::Mat< size_t > & resultingNeighbors, arma::mat & distances, const size_t numTablesToSearch = 0)`

Compute the nearest neighbors and store the output in the given matrices.

The matrices will be set to the size of n columns by k rows, where n is the number of points in the query dataset and k is the number of neighbors being searched for.

Parameters

<i>k</i>	Number of neighbors to search for.
<i>resulting↔ Neighbors</i>	Matrix storing lists of neighbors for each query point.
<i>distances</i>	Matrix storing distances of neighbors for each query point.
<i>numTablesTo↔ Search</i>	This parameter allows the user to have control over the number of hash tables to be searched. This allows the user to pick the number of tables it can afford for the time available without having to build hashing for every table size. By default, this is set to zero in which case all tables are considered.

22.91.3.6 `template<typename SortPolicy = NearestNeighborSort> std::string mpack::neighbor::LSHSearch< SortPolicy >::ToString () const`

22.91.4 Member Data Documentation

22.91.4.1 `template<typename SortPolicy = NearestNeighborSort> arma::Col<size_t> mpack::neighbor::LSHSearch< SortPolicy >::bucketContentSize [private]`

The number of elements present in each hash bucket; should be secondHashSize.

Definition at line 229 of file lsh_search.hpp.

22.91.4.2 `template<typename SortPolicy = NearestNeighborSort> arma::Col<size_t> mpack::neighbor::LSHSearch< SortPolicy >::bucketRowInHashTable [private]`

For a particular hash value, points to the row in secondHashTable corresponding to this value.

Should be secondHashSize.

Definition at line 233 of file lsh_search.hpp.

22.91.4.3 `template<typename SortPolicy = NearestNeighborSort> const size_t mpack::neighbor::LSHSearch< SortPolicy >::bucketSize [private]`

The bucket size of the second hash.

Definition at line 219 of file lsh_search.hpp.

22.91.4.4 `template<typename SortPolicy = NearestNeighborSort> arma::mat* mpack::neighbor::LSHSearch< SortPolicy >::distancePtr [private]`

The pointer to the nearest neighbor distances.

Definition at line 236 of file lsh_search.hpp.

22.91.4.5 `template<typename SortPolicy = NearestNeighborSort> double mpack::neighbor::LSHSearch< SortPolicy >::hashWidth [private]`

The hash width.

Definition at line 210 of file `lsh_search.hpp`.

22.91.4.6 `template<typename SortPolicy = NearestNeighborSort> metric::SquaredEuclideanDistance
mlpack::neighbor::LSHSearch< SortPolicy >::metric [private]`

Instantiation of the metric.

Definition at line 222 of file `lsh_search.hpp`.

22.91.4.7 `template<typename SortPolicy = NearestNeighborSort> arma::Mat<size_t>* mlpack::neighbor::LSHSearch<
SortPolicy >::neighborPtr [private]`

The pointer to the nearest neighbor indices.

Definition at line 239 of file `lsh_search.hpp`.

22.91.4.8 `template<typename SortPolicy = NearestNeighborSort> const size_t mlpack::neighbor::LSHSearch< SortPolicy
>::numProj [private]`

The number of projections.

Definition at line 198 of file `lsh_search.hpp`.

22.91.4.9 `template<typename SortPolicy = NearestNeighborSort> const size_t mlpack::neighbor::LSHSearch< SortPolicy
>::numTables [private]`

The number of hash tables.

Definition at line 201 of file `lsh_search.hpp`.

22.91.4.10 `template<typename SortPolicy = NearestNeighborSort> arma::mat mlpack::neighbor::LSHSearch< SortPolicy
>::offsets [private]`

The list of the offset 'b' for each of the projection for each table.

Definition at line 207 of file `lsh_search.hpp`.

22.91.4.11 `template<typename SortPolicy = NearestNeighborSort> std::vector<arma::mat>
mlpack::neighbor::LSHSearch< SortPolicy >::projections [private]`

The `std::vector` containing the projection matrix of each table.

Definition at line 204 of file `lsh_search.hpp`.

22.91.4.12 `template<typename SortPolicy = NearestNeighborSort> const arma::mat& mlpack::neighbor::LSHSearch<
SortPolicy >::querySet [private]`

Query dataset (may not be given).

Definition at line 195 of file `lsh_search.hpp`.

22.91.4.13 `template<typename SortPolicy = NearestNeighborSort> const arma::mat& mlpack::neighbor::LSHSearch<SortPolicy>::referenceSet [private]`

Reference dataset.

Definition at line 192 of file lsh_search.hpp.

22.91.4.14 `template<typename SortPolicy = NearestNeighborSort> const size_t mlpack::neighbor::LSHSearch<SortPolicy>::secondHashSize [private]`

The big prime representing the size of the second hash.

Definition at line 213 of file lsh_search.hpp.

22.91.4.15 `template<typename SortPolicy = NearestNeighborSort> arma::Mat<size_t> mlpack::neighbor::LSHSearch<SortPolicy>::secondHashTable [private]`

The final hash table; should be (< secondHashSize) x bucketSize.

Definition at line 225 of file lsh_search.hpp.

22.91.4.16 `template<typename SortPolicy = NearestNeighborSort> arma::vec mlpack::neighbor::LSHSearch<SortPolicy>::secondHashWeights [private]`

The weights of the second hash.

Definition at line 216 of file lsh_search.hpp.

The documentation for this class was generated from the following file:

- src/mlpack/methods/lsh/lsh_search.hpp

22.92 mlpack::neighbor::NearestNeighborSort Class Reference

This class implements the necessary methods for the SortPolicy template parameter of the **NeighborSearch** (p. 399) class.

Static Public Member Functions

- static double **BestDistance** ()
Return what should represent the best possible distance with this particular sort policy.
- template<typename TreeType >
static double **BestNodeToNodeDistance** (const TreeType *queryNode, const TreeType *referenceNode)
Return the best possible distance between two nodes.
- template<typename TreeType >
static double **BestNodeToNodeDistance** (const TreeType *queryNode, const TreeType *referenceNode, const double centerToCenterDistance)
Return the best possible distance between two nodes, given that the distance between the centers of the two nodes has already been calculated.

- `template<typename TreeType >`
`static double BestNodeToNodeDistance (const TreeType *queryNode, const TreeType *referenceNode, const TreeType *referenceChildNode, const double centerToCenterDistance)`
Return the best possible distance between the query node and the reference child node given the base case distance between the query node and the reference node.
- `template<typename VecType , typename TreeType >`
`static double BestPointToNodeDistance (const VecType &queryPoint, const TreeType *referenceNode)`
Return the best possible distance between a node and a point.
- `template<typename VecType , typename TreeType >`
`static double BestPointToNodeDistance (const VecType &queryPoint, const TreeType *referenceNode, const double pointToCenterDistance)`
Return the best possible distance between a point and a node, given that the distance between the point and the center of the node has already been calculated.
- `static double CombineBest (const double a, const double b)`
Return the best combination of the two distances.
- `static double CombineWorst (const double a, const double b)`
Return the worst combination of the two distances.
- `static bool IsBetter (const double value, const double ref)`
Return whether or not value is "better" than ref.
- `static size_t SortDistance (const arma::vec &list, const arma::Col< size_t > &indices, double newDistance)`
Return the index in the vector where the new distance should be inserted, or (size_t) - 1) if it should not be inserted (i.e.
- `static double WorstDistance ()`
Return what should represent the worst possible distance with this particular sort policy.

22.92.1 Detailed Description

This class implements the necessary methods for the SortPolicy template parameter of the **NeighborSearch** (p. 399) class.

The sorting policy here is that the minimum distance is the best (so, when used with **NeighborSearch** (p. 399), the output is nearest neighbors).

This class is also meant to serve as a guide to implement a custom SortPolicy. All of the methods implemented here must be implemented by any other SortPolicy classes.

Definition at line 33 of file nearest_neighbor_sort.hpp.

22.92.2 Member Function Documentation

22.92.2.1 `static double mpack::neighbor::NearestNeighborSort::BestDistance () [inline], [static]`

Return what should represent the best possible distance with this particular sort policy.

In our case, this should be the minimum possible distance, 0.0.

Returns

0.0

Definition at line 141 of file nearest_neighbor_sort.hpp.

22.92.2.2 `template<typename TreeType > static double mpack::neighbor::NearestNeighborSort::BestNodeToNodeDistance (const TreeType * queryNode, const TreeType * referenceNode) [static]`

Return the best possible distance between two nodes.

In our case, this is the minimum distance between the two tree nodes using the given distance function.

22.92.2.3 `template<typename TreeType > static double mpack::neighbor::NearestNeighborSort::BestNodeToNodeDistance (const TreeType * queryNode, const TreeType * referenceNode, const double centerToCenterDistance) [static]`

Return the best possible distance between two nodes, given that the distance between the centers of the two nodes has already been calculated.

This is used in conjunction with trees that have self-children (like cover trees).

22.92.2.4 `template<typename TreeType > static double mpack::neighbor::NearestNeighborSort::BestNodeToNodeDistance (const TreeType * queryNode, const TreeType * referenceNode, const TreeType * referenceChildNode, const double centerToCenterDistance) [static]`

Return the best possible distance between the query node and the reference child node given the base case distance between the query node and the reference node.

TreeType::ParentDistance() must be implemented to use this.

Parameters

<i>queryNode</i>	Query node.
<i>referenceNode</i>	Reference node.
<i>referenceChildNode</i>	Child of reference node which is being inspected.
<i>centerToCenterDistance</i>	Distance between centers of query node and reference node.

22.92.2.5 `template<typename VecType , typename TreeType > static double mpack::neighbor::NearestNeighborSort::BestPointToNodeDistance (const VecType & queryPoint, const TreeType * referenceNode) [static]`

Return the best possible distance between a node and a point.

In our case, this is the minimum distance between the tree node and the point using the given distance function.

22.92.2.6 `template<typename VecType , typename TreeType > static double mpack::neighbor::NearestNeighborSort::BestPointToNodeDistance (const VecType & queryPoint, const TreeType * referenceNode, const double pointToCenterDistance) [static]`

Return the best possible distance between a point and a node, given that the distance between the point and the center of the node has already been calculated.

This is used in conjunction with trees that have self-children (like cover trees).

22.92.2.7 `static double mlpack::neighbor::NearestNeighborSort::CombineBest (const double a, const double b)` `[inline]`,
`[static]`

Return the best combination of the two distances.

Definition at line 146 of file nearest_neighbor_sort.hpp.

22.92.2.8 `static double mlpack::neighbor::NearestNeighborSort::CombineWorst (const double a, const double b)` `[inline]`,
`[static]`

Return the worst combination of the two distances.

Definition at line 154 of file nearest_neighbor_sort.hpp.

22.92.2.9 `static bool mlpack::neighbor::NearestNeighborSort::IsBetter (const double value, const double ref)` `[inline]`,
`[static]`

Return whether or not *value* is "better" than *ref*.

In this case, that means that the *value* is less than the reference.

Parameters

<i>value</i>	Value to compare
<i>ref</i>	Value to compare with

Returns

bool indicating whether or not (*value* < *ref*).

Definition at line 63 of file nearest_neighbor_sort.hpp.

22.92.2.10 `static size_t mlpack::neighbor::NearestNeighborSort::SortDistance (const arma::vec & list, const arma::Col< size_t > & indices, double newDistance)` `[static]`

Return the index in the vector where the new distance should be inserted, or (*size_t*() - 1) if it should not be inserted (i.e. if it is not any better than any of the existing points in the list). The list should be sorted such that the best point is the first in the list. The actual insertion is not performed.

Parameters

<i>list</i>	Vector of existing distance points, sorted such that the best point is first in the list.
<i>new_distance</i>	Distance to try to insert

Returns

size_t containing the position to insert into, or (*size_t*() - 1) if the new distance should not be inserted.

22.92.2.11 `static double mlpack::neighbor::NearestNeighborSort::WorstDistance ()` `[inline]`, `[static]`

Return what should represent the worst possible distance with this particular sort policy.

In our case, this should be the maximum possible distance, DBL_MAX.

Returns

DBL_MAX

Definition at line 132 of file nearest_neighbor_sort.hpp.

The documentation for this class was generated from the following file:

- src/mlpack/methods/neighbor_search/sort_policies/nearest_neighbor_sort.hpp

22.93 mlpack::neighbor::NeighborSearch< SortPolicy, MetricType, TreeType > Class Template Reference

The **NeighborSearch** (p. 399) class is a template class for performing distance-based neighbor searches.

Public Member Functions

- **NeighborSearch** (const typename TreeType::Mat &**referenceSet**, const typename TreeType::Mat &**querySet**, const bool **naive**=false, const bool **singleMode**=false, const MetricType **metric**=MetricType())
*Initialize the **NeighborSearch** (p. 399) object, passing both a query and reference dataset.*
- **NeighborSearch** (const typename TreeType::Mat &**referenceSet**, const bool **naive**=false, const bool **singleMode**=false, const MetricType **metric**=MetricType())
*Initialize the **NeighborSearch** (p. 399) object, passing only one dataset, which is used as both the query and the reference dataset.*
- **NeighborSearch** (TreeType ***referenceTree**, TreeType ***queryTree**, const typename TreeType::Mat &**referenceSet**, const typename TreeType::Mat &**querySet**, const bool **singleMode**=false, const MetricType **metric**=MetricType())
*Initialize the **NeighborSearch** (p. 399) object with the given datasets and pre-constructed trees.*
- **NeighborSearch** (TreeType ***referenceTree**, const typename TreeType::Mat &**referenceSet**, const bool **singleMode**=false, const MetricType **metric**=MetricType())
*Initialize the **NeighborSearch** (p. 399) object with the given reference dataset and pre-constructed tree.*
- **~NeighborSearch** ()
*Delete the **NeighborSearch** (p. 399) object.*
- void **Search** (const size_t k, arma::Mat< size_t > &resultingNeighbors, arma::mat &distances)
Compute the nearest neighbors and store the output in the given matrices.
- std::string **ToString** () const

Private Attributes

- bool **hasQuerySet**
Indicates if a separate query set was passed.
- MetricType **metric**
Instantiation of metric.
- bool **naive**
Indicates if $O(n^2)$ naive search is being used.
- std::vector< size_t > **oldFromNewQueries**
Permutations of query points during tree building.
- std::vector< size_t > **oldFromNewReferences**

Permutations of reference points during tree building.

- `TreeType::Mat queryCopy`

Copy of query dataset (if we need it, because tree building modifies it).

- `const TreeType::Mat & querySet`

Query dataset (may not be given).

- `TreeType * queryTree`

Pointer to the root of the query tree (might not exist).

- `TreeType::Mat referenceCopy`

Copy of reference dataset (if we need it, because tree building modifies it).

- `const TreeType::Mat & referenceSet`

Reference dataset.

- `TreeType * referenceTree`

Pointer to the root of the reference tree.

- `bool singleMode`

Indicates if single-tree search is being used (opposed to dual-tree).

- `bool treeOwner`

If true, this object created the trees and is responsible for them.

22.93.1 Detailed Description

```
template<typename SortPolicy = NearestNeighborSort, typename MetricType = mpack::metric::SquaredEuclideanDistance, type-
name TreeType = tree::BinarySpaceTree<bound::HRectBound<2>, NeighborSearchStat<SortPolicy>>>class mpack::neighbor<
::NeighborSearch< SortPolicy, MetricType, TreeType >
```

The **NeighborSearch** (p. 399) class is a template class for performing distance-based neighbor searches.

It takes a query dataset and a reference dataset (or just a reference dataset) and, for each point in the query dataset, finds the k neighbors in the reference dataset which have the 'best' distance according to a given sorting policy. A constructor is given which takes only a reference dataset, and if that constructor is used, the given reference dataset is also used as the query dataset.

The template parameters `SortPolicy` and `Metric` define the sort function used and the metric (distance function) used. More information on those classes can be found in the **NearestNeighborSort** (p. 395) class and the **kernel::Example**↵
Kernel (p. 296) class.

Template Parameters

<i>SortPolicy</i>	The sort policy for distances; see NearestNeighborSort (p. 395).
<i>MetricType</i>	The metric to use for computation.
<i>TreeType</i>	The tree type to use.

Definition at line 55 of file `neighbor_search.hpp`.

22.93.2 Constructor & Destructor Documentation

```
22.93.2.1 template<typename SortPolicy = NearestNeighborSort, typename MetricType = mpack::metric::SquaredEuclidean<
Distance, typename TreeType = tree::BinarySpaceTree<bound::HRectBound<2>, NeighborSearchStat<SortPolicy>>>
>> mpack::neighbor::NeighborSearch< SortPolicy, MetricType, TreeType >::NeighborSearch ( const
typename TreeType::Mat & referenceSet, const typename TreeType::Mat & querySet, const bool naive = false, const
bool singleMode = false, const MetricType metric = MetricType() )
```

Initialize the **NeighborSearch** (p. 399) object, passing both a query and reference dataset.

Optionally, perform the computation in naive mode or single-tree mode, and set the leaf size used for tree-building. An initialized distance metric can be given, for cases where the metric has internal data (i.e. the distance::Mahalanobis← Distance class).

This method will copy the matrices to internal copies, which are rearranged during tree-building. You can avoid this extra copy by pre-constructing the trees and passing them using a different constructor.

Parameters

<i>referenceSet</i>	Set of reference points.
<i>querySet</i>	Set of query points.
<i>naive</i>	If true, $O(n^2)$ naive search will be used (as opposed to dual-tree search). This overrides singleMode (if it is set to true).
<i>singleMode</i>	If true, single-tree search will be used (as opposed to dual-tree search).
<i>leafSize</i>	Leaf size for tree construction (ignored if tree is given).
<i>metric</i>	An optional instance of the MetricType class.

```
22.93.2.2  template<typename SortPolicy = NearestNeighborSort, typename MetricType = mlpack::metric::SquaredEuclidean←
Distance, typename TreeType = tree::BinarySpaceTree<bound::HRectBound<2>, NeighborSearchStat<SortPolicy>
>> mlpack::neighbor::NeighborSearch< SortPolicy, MetricType, TreeType >::NeighborSearch ( const
typename TreeType::Mat & referenceSet, const bool naive = false, const bool singleMode = false, const MetricType
metric = MetricType() )
```

Initialize the **NeighborSearch** (p.399) object, passing only one dataset, which is used as both the query and the reference dataset.

Optionally, perform the computation in naive mode or single-tree mode, and set the leaf size used for tree-building. An initialized distance metric can be given, for cases where the metric has internal data (i.e. the distance::Mahalanobis← Distance class).

If naive mode is being used and a pre-built tree is given, it may not work: naive mode operates by building a one-node tree (the root node holds all the points). If that condition is not satisfied with the pre-built tree, then naive mode will not work.

Parameters

<i>referenceSet</i>	Set of reference points.
<i>naive</i>	If true, $O(n^2)$ naive search will be used (as opposed to dual-tree search). This overrides singleMode (if it is set to true).
<i>singleMode</i>	If true, single-tree search will be used (as opposed to dual-tree search).
<i>leafSize</i>	Leaf size for tree construction (ignored if tree is given).
<i>metric</i>	An optional instance of the MetricType class.

```
22.93.2.3  template<typename SortPolicy = NearestNeighborSort, typename MetricType = mlpack::metric::SquaredEuclidean←
Distance, typename TreeType = tree::BinarySpaceTree<bound::HRectBound<2>, NeighborSearchStat<SortPolicy>
>> mlpack::neighbor::NeighborSearch< SortPolicy, MetricType, TreeType >::NeighborSearch ( TreeType *
referenceTree, TreeType * queryTree, const typename TreeType::Mat & referenceSet, const typename TreeType::Mat &
querySet, const bool singleMode = false, const MetricType metric = MetricType() )
```

Initialize the **NeighborSearch** (p.399) object with the given datasets and pre-constructed trees.

It is assumed that the points in referenceSet and querySet correspond to the points in referenceTree and queryTree, respectively. Optionally, choose to use single-tree mode. Naive mode is not available as an option for this constructor; instead, to run naive computation, construct a tree with all of the points in one leaf (i.e. leafSize = number of points). Additionally, an instantiated distance metric can be given, for cases where the distance metric holds data.

There is no copying of the data matrices in this constructor (because tree-building is not necessary), so this is the constructor to use when copies absolutely must be avoided.

Note

Because tree-building (at least with `BinarySpaceTree`) modifies the ordering of a matrix, be sure you pass the modified matrix to this object! In addition, mapping the points of the matrix back to their original indices is not done when this constructor is used.

Parameters

<i>referenceTree</i>	Pre-built tree for reference points.
<i>queryTree</i>	Pre-built tree for query points.
<i>referenceSet</i>	Set of reference points corresponding to <i>referenceTree</i> .
<i>querySet</i>	Set of query points corresponding to <i>queryTree</i> .
<i>singleMode</i>	Whether single-tree computation should be used (as opposed to dual-tree computation).
<i>metric</i>	Instantiated distance metric.

```
22.93.2.4  template<typename SortPolicy = NearestNeighborSort, typename MetricType = mlpack::metric::SquaredEuclidean↵
Distance, typename TreeType = tree::BinarySpaceTree<bound::HRectBound<2>, NeighborSearchStat<SortPolicy>↵
>> mlpack::neighbor::NeighborSearch< SortPolicy, MetricType, TreeType >::NeighborSearch ( TreeType *↵
referenceTree, const typename TreeType::Mat & referenceSet, const bool singleMode = false, const MetricType metric↵
= MetricType() )
```

Initialize the **NeighborSearch** (p. 399) object with the given reference dataset and pre-constructed tree.

It is assumed that the points in *referenceSet* correspond to the points in *referenceTree*. Optionally, choose to use single-tree mode. Naive mode is not available as an option for this constructor; instead, to run naive computation, construct a tree with all the points in one leaf (i.e. *leafSize* = number of points). Additionally, an instantiated distance metric can be given, for the case where the distance metric holds data.

There is no copying of the data matrices in this constructor (because tree-building is not necessary), so this is the constructor to use when copies absolutely must be avoided.

Note

Because tree-building (at least with `BinarySpaceTree`) modifies the ordering of a matrix, be sure you pass the modified matrix to this object! In addition, mapping the points of the matrix back to their original indices is not done when this constructor is used.

Parameters

<i>referenceTree</i>	Pre-built tree for reference points.
<i>referenceSet</i>	Set of reference points corresponding to <i>referenceTree</i> .
<i>singleMode</i>	Whether single-tree computation should be used (as opposed to dual-tree computation).
<i>metric</i>	Instantiated distance metric.

```
22.93.2.5  template<typename SortPolicy = NearestNeighborSort, typename MetricType = mlpack::metric::SquaredEuclidean↵
Distance, typename TreeType = tree::BinarySpaceTree<bound::HRectBound<2>, NeighborSearchStat<SortPolicy>↵
>> mlpack::neighbor::NeighborSearch< SortPolicy, MetricType, TreeType >::~NeighborSearch ( )
```

Delete the **NeighborSearch** (p. 399) object.

The tree is the only member we are responsible for deleting. The others will take care of themselves.

22.93.3 Member Function Documentation

22.93.3.1 `template<typename SortPolicy = NearestNeighborSort, typename MetricType = mpack::metric::SquaredEuclideanDistance, typename TreeType = tree::BinarySpaceTree<bound::HRectBound<2>, NeighborSearchStat<SortPolicy>>> void mpack::neighbor::NeighborSearch< SortPolicy, MetricType, TreeType >::Search (const size_t k, arma::Mat< size_t > & resultingNeighbors, arma::mat & distances)`

Compute the nearest neighbors and store the output in the given matrices.

The matrices will be set to the size of n columns by k rows, where n is the number of points in the query dataset and k is the number of neighbors being searched for.

Parameters

<i>k</i>	Number of neighbors to search for.
<i>resultingNeighbors</i>	Matrix storing lists of neighbors for each query point.
<i>distances</i>	Matrix storing distances of neighbors for each query point.

22.93.3.2 `template<typename SortPolicy = NearestNeighborSort, typename MetricType = mpack::metric::SquaredEuclideanDistance, typename TreeType = tree::BinarySpaceTree<bound::HRectBound<2>, NeighborSearchStat<SortPolicy>>> std::string mpack::neighbor::NeighborSearch< SortPolicy, MetricType, TreeType >::ToString () const`

22.93.4 Member Data Documentation

22.93.4.1 `template<typename SortPolicy = NearestNeighborSort, typename MetricType = mpack::metric::SquaredEuclideanDistance, typename TreeType = tree::BinarySpaceTree<bound::HRectBound<2>, NeighborSearchStat<SortPolicy>>> bool mpack::neighbor::NeighborSearch< SortPolicy, MetricType, TreeType >::hasQuerySet [private]`

Indicates if a separate query set was passed.

Definition at line 224 of file neighbor_search.hpp.

22.93.4.2 `template<typename SortPolicy = NearestNeighborSort, typename MetricType = mpack::metric::SquaredEuclideanDistance, typename TreeType = tree::BinarySpaceTree<bound::HRectBound<2>, NeighborSearchStat<SortPolicy>>> MetricType mpack::neighbor::NeighborSearch< SortPolicy, MetricType, TreeType >::metric [private]`

Instantiation of metric.

Definition at line 232 of file neighbor_search.hpp.

22.93.4.3 `template<typename SortPolicy = NearestNeighborSort, typename MetricType = mpack::metric::SquaredEuclideanDistance, typename TreeType = tree::BinarySpaceTree<bound::HRectBound<2>, NeighborSearchStat<SortPolicy>>> bool mpack::neighbor::NeighborSearch< SortPolicy, MetricType, TreeType >::naive [private]`

Indicates if $O(n^2)$ naive search is being used.

Definition at line 227 of file neighbor_search.hpp.

```
22.93.4.4  template<typename SortPolicy = NearestNeighborSort, typename MetricType = mlpack::metric::SquaredEuclidean↵
Distance, typename TreeType = tree::BinarySpaceTree<bound::HRectBound<2>, NeighborSearchStat<SortPolicy>↵
>> std::vector<size_t> mlpack::neighbor::NeighborSearch< SortPolicy, MetricType, TreeType
>::oldFromNewQueries    [private]
```

Permutations of query points during tree building.

Definition at line 237 of file neighbor_search.hpp.

```
22.93.4.5  template<typename SortPolicy = NearestNeighborSort, typename MetricType = mlpack::metric::SquaredEuclidean↵
Distance, typename TreeType = tree::BinarySpaceTree<bound::HRectBound<2>, NeighborSearchStat<SortPolicy>↵
>> std::vector<size_t> mlpack::neighbor::NeighborSearch< SortPolicy, MetricType, TreeType
>::oldFromNewReferences  [private]
```

Permutations of reference points during tree building.

Definition at line 235 of file neighbor_search.hpp.

```
22.93.4.6  template<typename SortPolicy = NearestNeighborSort, typename MetricType = mlpack::metric::SquaredEuclidean↵
Distance, typename TreeType = tree::BinarySpaceTree<bound::HRectBound<2>, NeighborSearchStat<SortPolicy>↵
>> TreeType::Mat mlpack::neighbor::NeighborSearch< SortPolicy, MetricType, TreeType >::queryCopy
[private]
```

Copy of query dataset (if we need it, because tree building modifies it).

Definition at line 209 of file neighbor_search.hpp.

```
22.93.4.7  template<typename SortPolicy = NearestNeighborSort, typename MetricType = mlpack::metric::SquaredEuclidean↵
Distance, typename TreeType = tree::BinarySpaceTree<bound::HRectBound<2>, NeighborSearchStat<SortPolicy>↵
>> const TreeType::Mat& mlpack::neighbor::NeighborSearch< SortPolicy, MetricType, TreeType >::querySet
[private]
```

Query dataset (may not be given).

Definition at line 214 of file neighbor_search.hpp.

```
22.93.4.8  template<typename SortPolicy = NearestNeighborSort, typename MetricType = mlpack::metric::SquaredEuclidean↵
Distance, typename TreeType = tree::BinarySpaceTree<bound::HRectBound<2>, NeighborSearchStat<SortPolicy>↵
>> TreeType* mlpack::neighbor::NeighborSearch< SortPolicy, MetricType, TreeType >::queryTree
[private]
```

Pointer to the root of the query tree (might not exist).

Definition at line 219 of file neighbor_search.hpp.

```
22.93.4.9  template<typename SortPolicy = NearestNeighborSort, typename MetricType = mlpack::metric::SquaredEuclidean↵
Distance, typename TreeType = tree::BinarySpaceTree<bound::HRectBound<2>, NeighborSearchStat<SortPolicy>↵
>> TreeType::Mat mlpack::neighbor::NeighborSearch< SortPolicy, MetricType, TreeType >::referenceCopy
[private]
```

Copy of reference dataset (if we need it, because tree building modifies it).

Definition at line 207 of file neighbor_search.hpp.


```
22.93.4.10 template<typename SortPolicy = NearestNeighborSort, typename MetricType = mlpack::metric::SquaredEuclidean↵
Distance, typename TreeType = tree::BinarySpaceTree<bound::HRectBound<2>, NeighborSearchStat<SortPolicy>↵
>> const TreeType::Mat& mlpack::neighbor::NeighborSearch< SortPolicy, MetricType, TreeType↵
>::referenceSet [private]
```

Reference dataset.

Definition at line 212 of file neighbor_search.hpp.

```
22.93.4.11 template<typename SortPolicy = NearestNeighborSort, typename MetricType = mlpack::metric::SquaredEuclidean↵
Distance, typename TreeType = tree::BinarySpaceTree<bound::HRectBound<2>, NeighborSearchStat<SortPolicy>↵
>> TreeType* mlpack::neighbor::NeighborSearch< SortPolicy, MetricType, TreeType >::referenceTree↵
[private]
```

Pointer to the root of the reference tree.

Definition at line 217 of file neighbor_search.hpp.

```
22.93.4.12 template<typename SortPolicy = NearestNeighborSort, typename MetricType = mlpack::metric::SquaredEuclidean↵
Distance, typename TreeType = tree::BinarySpaceTree<bound::HRectBound<2>, NeighborSearchStat<SortPolicy>↵
>> bool mlpack::neighbor::NeighborSearch< SortPolicy, MetricType, TreeType >::singleMode [private]
```

Indicates if single-tree search is being used (opposed to dual-tree).

Definition at line 229 of file neighbor_search.hpp.

```
22.93.4.13 template<typename SortPolicy = NearestNeighborSort, typename MetricType = mlpack::metric::SquaredEuclidean↵
Distance, typename TreeType = tree::BinarySpaceTree<bound::HRectBound<2>, NeighborSearchStat<SortPolicy>↵
>> bool mlpack::neighbor::NeighborSearch< SortPolicy, MetricType, TreeType >::treeOwner [private]
```

If true, this object created the trees and is responsible for them.

Definition at line 222 of file neighbor_search.hpp.

The documentation for this class was generated from the following file:

- src/mlpack/methods/neighbor_search/**neighbor_search.hpp**

22.94 mlpack::neighbor::NeighborSearchRules< SortPolicy, MetricType, TreeType > Class Template Reference

Public Types

- typedef **NeighborSearchTraversallInfo**< TreeType > **TraversallInfoType**
Convenience typedef.

Public Member Functions

- **NeighborSearchRules** (const typename TreeType::Mat &**referenceSet**, const typename TreeType::Mat &**querySet**, arma::Mat< size_t > &**neighbors**, arma::mat &**distances**, MetricType &**metric**)
- double **BaseCase** (const size_t queryIndex, const size_t referenceIndex)

- Get the distance from the query point to the reference point.*

 - `size_t BaseCases () const`

Get the number of base cases that have been performed.
 - `size_t & BaseCases ()`

Modify the number of base cases that have been performed.
 - `double Rescore (const size_t queryIndex, TreeType &referenceNode, const double oldScore) const`

Re-evaluate the score for recursion order.
 - `double Rescore (TreeType &queryNode, TreeType &referenceNode, const double oldScore) const`

Re-evaluate the score for recursion order.
 - `double Score (const size_t queryIndex, TreeType &referenceNode)`

Get the score for recursion order.
 - `double Score (TreeType &queryNode, TreeType &referenceNode)`

Get the score for recursion order.
 - `size_t Scores () const`

Get the number of scores that have been performed.
 - `size_t & Scores ()`

Modify the number of scores that have been performed.
 - `const TraversallInfoType & TraversallInfo () const`

Get the traversal info.
 - `TraversallInfoType & TraversallInfo ()`

Modify the traversal info.

Private Member Functions

- `double CalculateBound (TreeType &queryNode) const`

Recalculate the bound for a given query node.
- `void InsertNeighbor (const size_t queryIndex, const size_t pos, const size_t neighbor, const double distance)`

Insert a point into the neighbors and distances matrices; this is a helper function.

Private Attributes

- `size_t baseCases`

The number of base cases that have been performed.
- `arma::mat & distances`

The matrix the resultant neighbor distances should be stored in.
- `double lastBaseCase`

The last base case result.
- `size_t lastQueryIndex`

*The last query point **BaseCase()** (p. 407) was called with.*
- `size_t lastReferenceIndex`

*The last reference point **BaseCase()** (p. 407) was called with.*
- `MetricType & metric`

The instantiated metric.
- `arma::Mat< size_t > & neighbors`

The matrix the resultant neighbor indices should be stored in.
- `const TreeType::Mat & querySet`

The query set.

- const TreeType::Mat & **referenceSet**

The reference set.

- size_t **scores**

The number of scores that have been performed.

- **TraversallInfoType** **traversalInfo**

*Traversal info for the parent combination; this is updated by the traversal before each call to **Score()** (p. 409).*

22.94.1 Detailed Description

```
template<typename SortPolicy, typename MetricType, typename TreeType>class mlpack::neighbor::NeighborSearchRules< SortPolicy, MetricType, TreeType >
```

Definition at line 24 of file neighbor_search_rules.hpp.

22.94.2 Member Typedef Documentation

22.94.2.1 `template<typename SortPolicy , typename MetricType , typename TreeType > typedef NeighborSearchTraversallInfo<TreeType> mlpack::neighbor::NeighborSearchRules< SortPolicy, MetricType, TreeType >::TraversallInfoType`

Convenience typedef.

Definition at line 103 of file neighbor_search_rules.hpp.

22.94.3 Constructor & Destructor Documentation

22.94.3.1 `template<typename SortPolicy , typename MetricType , typename TreeType > mlpack::neighbor::NeighborSearchRules< SortPolicy, MetricType, TreeType >::NeighborSearchRules(const typename TreeType::Mat & referenceSet, const typename TreeType::Mat & querySet, arma::Mat< size_t > & neighbors, arma::mat & distances, MetricType & metric)`

22.94.4 Member Function Documentation

22.94.4.1 `template<typename SortPolicy , typename MetricType , typename TreeType > double mlpack::neighbor::NeighborSearchRules< SortPolicy, MetricType, TreeType >::BaseCase(const size_t queryIndex, const size_t referenceIndex)`

Get the distance from the query point to the reference point.

This will update the "neighbor" matrix with the new point if appropriate and will track the number of base cases (number of points evaluated).

Parameters

<i>queryIndex</i>	Index of query point.
<i>referenceIndex</i>	Index of reference point.

```
22.94.4.2  template<typename SortPolicy , typename MetricType , typename TreeType > size_t mlpack↵
           ::neighbor::NeighborSearchRules< SortPolicy, MetricType, TreeType >::BaseCases (    ) const
           [inline]
```

Get the number of base cases that have been performed.

Definition at line 93 of file neighbor_search_rules.hpp.

References mlpack::neighbor::NeighborSearchRules< SortPolicy, MetricType, TreeType >::baseCases.

```
22.94.4.3  template<typename SortPolicy , typename MetricType , typename TreeType > size_t& mlpack↵
           ::neighbor::NeighborSearchRules< SortPolicy, MetricType, TreeType >::BaseCases (    )
           [inline]
```

Modify the number of base cases that have been performed.

Definition at line 95 of file neighbor_search_rules.hpp.

References mlpack::neighbor::NeighborSearchRules< SortPolicy, MetricType, TreeType >::baseCases.

```
22.94.4.4  template<typename SortPolicy , typename MetricType , typename TreeType > double mlpack::neighbor::↵
           NeighborSearchRules< SortPolicy, MetricType, TreeType >::CalculateBound ( TreeType & queryNode ) const
           [private]
```

Recalculate the bound for a given query node.

```
22.94.4.5  template<typename SortPolicy , typename MetricType , typename TreeType > void mlpack::neighbor::Neighbor↵
           SearchRules< SortPolicy, MetricType, TreeType >::InsertNeighbor ( const size_t queryIndex, const size_t pos, const
           size_t neighbor, const double distance ) [private]
```

Insert a point into the neighbors and distances matrices; this is a helper function.

Parameters

<i>queryIndex</i>	Index of point whose neighbors we are inserting into.
<i>pos</i>	Position in list to insert into.
<i>neighbor</i>	Index of reference point which is being inserted.
<i>distance</i>	Distance from query point to reference point.

```
22.94.4.6  template<typename SortPolicy , typename MetricType , typename TreeType > double mlpack::neighbor::↵
           NeighborSearchRules< SortPolicy, MetricType, TreeType >::Rescore ( const size_t queryIndex, TreeType &
           referenceNode, const double oldScore ) const
```

Re-evaluate the score for recursion order.

A low score indicates priority for recursion, while DBL_MAX indicates that the node should not be recursed into at all (it should be pruned). This is used when the score has already been calculated, but another recursion may have modified the bounds for pruning. So the old score is checked against the new pruning bound.

Parameters

<i>queryIndex</i>	Index of query point.
<i>referenceNode</i>	Candidate node to be recursed into.
<i>oldScore</i>	Old score produced by Score() (p. 409) (or Rescore() (p. 408)).

22.94.4.7 `template<typename SortPolicy , typename MetricType , typename TreeType > double mlpack::neighbor::↵
NeighborSearchRules< SortPolicy, MetricType, TreeType >::Rescore (TreeType & queryNode, TreeType &
referenceNode, const double oldScore) const`

Re-evaluate the score for recursion order.

A low score indicates priority for recursion, while DBL_MAX indicates that the node should not be recursed into at all (it should be pruned). This is used when the score has already been calculated, but another recursion may have modified the bounds for pruning. So the old score is checked against the new pruning bound.

Parameters

<i>queryNode</i>	Candidate query node to recurse into.
<i>referenceNode</i>	Candidate reference node to recurse into.
<i>oldScore</i>	Old score produced by Socre() (or Rescore() (p. 408)).

22.94.4.8 `template<typename SortPolicy , typename MetricType , typename TreeType > double mlpack::neighbor::↵
NeighborSearchRules< SortPolicy, MetricType, TreeType >::Score (const size_t queryIndex, TreeType &
referenceNode)`

Get the score for recursion order.

A low score indicates priority for recursion, while DBL_MAX indicates that the node should not be recursed into at all (it should be pruned).

Parameters

<i>queryIndex</i>	Index of query point.
<i>referenceNode</i>	Candidate node to be recursed into.

22.94.4.9 `template<typename SortPolicy , typename MetricType , typename TreeType > double mlpack::neighbor::↵
NeighborSearchRules< SortPolicy, MetricType, TreeType >::Score (TreeType & queryNode, TreeType &
referenceNode)`

Get the score for recursion order.

A low score indicates priority for recursionm while DBL_MAX indicates that the node should not be recursed into at all (it should be pruned).

Parameters

<i>queryNode</i>	Candidate query node to recurse into.
<i>referenceNode</i>	Candidate reference node to recurse into.

```
22.94.4.10 template<typename SortPolicy , typename MetricType , typename TreeType > size_t mlpack↵
      ::neighbor::NeighborSearchRules< SortPolicy, MetricType, TreeType >::Scores ( ) const
      [inline]
```

Get the number of scores that have been performed.

Definition at line 98 of file neighbor_search_rules.hpp.

References mlpack::neighbor::NeighborSearchRules< SortPolicy, MetricType, TreeType >::scores.

```
22.94.4.11 template<typename SortPolicy , typename MetricType , typename TreeType > size_t&
      mlpack::neighbor::NeighborSearchRules< SortPolicy, MetricType, TreeType >::Scores ( ) [inline]
```

Modify the number of scores that have been performed.

Definition at line 100 of file neighbor_search_rules.hpp.

References mlpack::neighbor::NeighborSearchRules< SortPolicy, MetricType, TreeType >::scores.

```
22.94.4.12 template<typename SortPolicy , typename MetricType , typename TreeType > const TraversalInfoType&
      mlpack::neighbor::NeighborSearchRules< SortPolicy, MetricType, TreeType >::TraversalInfo ( ) const
      [inline]
```

Get the traversal info.

Definition at line 106 of file neighbor_search_rules.hpp.

References mlpack::neighbor::NeighborSearchRules< SortPolicy, MetricType, TreeType >::traversalInfo.

```
22.94.4.13 template<typename SortPolicy , typename MetricType , typename TreeType > TraversalInfoType&
      mlpack::neighbor::NeighborSearchRules< SortPolicy, MetricType, TreeType >::TraversalInfo ( )
      [inline]
```

Modify the traversal info.

Definition at line 108 of file neighbor_search_rules.hpp.

References mlpack::neighbor::NeighborSearchRules< SortPolicy, MetricType, TreeType >::traversalInfo.

22.94.5 Member Data Documentation

```
22.94.5.1 template<typename SortPolicy , typename MetricType , typename TreeType > size_t
      mlpack::neighbor::NeighborSearchRules< SortPolicy, MetricType, TreeType >::baseCases [private]
```

The number of base cases that have been performed.

Definition at line 134 of file neighbor_search_rules.hpp.

Referenced by mlpack::neighbor::NeighborSearchRules< SortPolicy, MetricType, TreeType >::BaseCases().

```
22.94.5.2 template<typename SortPolicy , typename MetricType , typename TreeType > arma::mat&
      mlpack::neighbor::NeighborSearchRules< SortPolicy, MetricType, TreeType >::distances [private]
```

The matrix the resultant neighbor distances should be stored in.

Definition at line 121 of file neighbor_search_rules.hpp.

22.94.5.3 `template<typename SortPolicy , typename MetricType , typename TreeType > double
mlpack::neighbor::NeighborSearchRules< SortPolicy, MetricType, TreeType >::lastBaseCase [private]`

The last base case result.

Definition at line 131 of file neighbor_search_rules.hpp.

22.94.5.4 `template<typename SortPolicy , typename MetricType , typename TreeType > size_t mlpack↵
::neighbor::NeighborSearchRules< SortPolicy, MetricType, TreeType >::lastQueryIndex
[private]`

The last query point **BaseCase()** (p. 407) was called with.

Definition at line 127 of file neighbor_search_rules.hpp.

22.94.5.5 `template<typename SortPolicy , typename MetricType , typename TreeType > size_t mlpack↵
::neighbor::NeighborSearchRules< SortPolicy, MetricType, TreeType >::lastReferenceIndex
[private]`

The last reference point **BaseCase()** (p. 407) was called with.

Definition at line 129 of file neighbor_search_rules.hpp.

22.94.5.6 `template<typename SortPolicy , typename MetricType , typename TreeType > MetricType&
mlpack::neighbor::NeighborSearchRules< SortPolicy, MetricType, TreeType >::metric [private]`

The instantiated metric.

Definition at line 124 of file neighbor_search_rules.hpp.

22.94.5.7 `template<typename SortPolicy , typename MetricType , typename TreeType > arma::Mat<size_t>&
mlpack::neighbor::NeighborSearchRules< SortPolicy, MetricType, TreeType >::neighbors [private]`

The matrix the resultant neighbor indices should be stored in.

Definition at line 118 of file neighbor_search_rules.hpp.

22.94.5.8 `template<typename SortPolicy , typename MetricType , typename TreeType > const TreeType::Mat&
mlpack::neighbor::NeighborSearchRules< SortPolicy, MetricType, TreeType >::querySet [private]`

The query set.

Definition at line 115 of file neighbor_search_rules.hpp.

22.94.5.9 `template<typename SortPolicy , typename MetricType , typename TreeType > const TreeType::Mat&
mlpack::neighbor::NeighborSearchRules< SortPolicy, MetricType, TreeType >::referenceSet [private]`

The reference set.

Definition at line 112 of file neighbor_search_rules.hpp.

22.94.5.10 `template<typename SortPolicy , typename MetricType , typename TreeType > size_t
mlpack::neighbor::NeighborSearchRules< SortPolicy, MetricType, TreeType >::scores [private]`

The number of scores that have been performed.

Definition at line 136 of file `neighbor_search_rules.hpp`.

Referenced by `mlpack::neighbor::NeighborSearchRules< SortPolicy, MetricType, TreeType >::Scores()`.

22.94.5.11 `template<typename SortPolicy , typename MetricType , typename TreeType > TraversalInfoType
mlpack::neighbor::NeighborSearchRules< SortPolicy, MetricType, TreeType >::traversalInfo [private]`

Traversal info for the parent combination; this is updated by the traversal before each call to **Score()** (p. 409).

Definition at line 140 of file `neighbor_search_rules.hpp`.

Referenced by `mlpack::neighbor::NeighborSearchRules< SortPolicy, MetricType, TreeType >::TraversalInfo()`.

The documentation for this class was generated from the following file:

- `src/mlpack/methods/neighbor_search/neighbor_search_rules.hpp`

22.95 mlpack::neighbor::NeighborSearchStat< SortPolicy > Class Template Reference

Extra data for each node in the tree.

Public Member Functions

- **NeighborSearchStat** ()
Initialize the statistic with the worst possible distance according to our sorting policy.
- `template<typename TreeType >
NeighborSearchStat (TreeType &)`
Initialization for a fully initialized node.
- `double Bound () const`
Get the overall bound (the better of the two bounds).
- `double & Bound ()`
Modify the overall bound (it should be the better of the two bounds).
- `double FirstBound () const`
Get the first bound.
- `double & FirstBound ()`
Modify the first bound.
- `double LastDistance () const`
Get the last distance calculation.
- `double & LastDistance ()`
Modify the last distance calculation.
- `void * LastDistanceNode () const`
Get the last distance evaluation node.
- `void *& LastDistanceNode ()`
Modify the last distance evaluation node.
- `double SecondBound () const`

Get the second bound.

- double & **SecondBound** ()

Modify the second bound.

Private Attributes

- double **bound**

The better of the two bounds.

- double **firstBound**

The first bound on the node's neighbor distances (B_1).

- double **lastDistance**

The last distance evaluation.

- void * **lastDistanceNode**

The last distance evaluation node.

- double **secondBound**

The second bound on the node's neighbor distances (B_2).

22.95.1 Detailed Description

```
template<typename SortPolicy>class mlpack::neighbor::NeighborSearchStat< SortPolicy >
```

Extra data for each node in the tree.

For neighbor searches, each node only needs to store a bound on neighbor distances.

Definition at line 28 of file neighbor_search_stat.hpp.

22.95.2 Constructor & Destructor Documentation

```
22.95.2.1 template<typename SortPolicy > mlpack::neighbor::NeighborSearchStat< SortPolicy
>::NeighborSearchStat( ) [inline]
```

Initialize the statistic with the worst possible distance according to our sorting policy.

Definition at line 52 of file neighbor_search_stat.hpp.

```
22.95.2.2 template<typename SortPolicy > template<typename TreeType > mlpack::neighbor::NeighborSearchStat<
SortPolicy >::NeighborSearchStat( TreeType & ) [inline]
```

Initialization for a fully initialized node.

In this case, we don't need to worry about the node.

Definition at line 64 of file neighbor_search_stat.hpp.

22.95.3 Member Function Documentation

```
22.95.3.1 template<typename SortPolicy > double mlpack::neighbor::NeighborSearchStat< SortPolicy >::Bound ( )
const [inline]
```

Get the overall bound (the better of the two bounds).

Definition at line 80 of file neighbor_search_stat.hpp.

References `mlpack::neighbor::NeighborSearchStat< SortPolicy >::bound`.

22.95.3.2 `template<typename SortPolicy > double& mlpack::neighbor::NeighborSearchStat< SortPolicy >::Bound ()`
`[inline]`

Modify the overall bound (it should be the better of the two bounds).

Definition at line 82 of file neighbor_search_stat.hpp.

References `mlpack::neighbor::NeighborSearchStat< SortPolicy >::bound`.

22.95.3.3 `template<typename SortPolicy > double mlpack::neighbor::NeighborSearchStat< SortPolicy >::FirstBound ()`
`const [inline]`

Get the first bound.

Definition at line 72 of file neighbor_search_stat.hpp.

References `mlpack::neighbor::NeighborSearchStat< SortPolicy >::firstBound`.

22.95.3.4 `template<typename SortPolicy > double& mlpack::neighbor::NeighborSearchStat< SortPolicy >::FirstBound ()`
`[inline]`

Modify the first bound.

Definition at line 74 of file neighbor_search_stat.hpp.

References `mlpack::neighbor::NeighborSearchStat< SortPolicy >::firstBound`.

22.95.3.5 `template<typename SortPolicy > double mlpack::neighbor::NeighborSearchStat< SortPolicy >::LastDistance ()`
`const [inline]`

Get the last distance calculation.

Definition at line 88 of file neighbor_search_stat.hpp.

References `mlpack::neighbor::NeighborSearchStat< SortPolicy >::lastDistance`.

22.95.3.6 `template<typename SortPolicy > double& mlpack::neighbor::NeighborSearchStat< SortPolicy >::LastDistance ()`
`[inline]`

Modify the last distance calculation.

Definition at line 90 of file neighbor_search_stat.hpp.

References `mlpack::neighbor::NeighborSearchStat< SortPolicy >::lastDistance`.

22.95.3.7 `template<typename SortPolicy > void* mlpack::neighbor::NeighborSearchStat< SortPolicy >::LastDistanceNode ()`
`const [inline]`

Get the last distance evaluation node.

Definition at line 84 of file neighbor_search_stat.hpp.

References mlpack::neighbor::NeighborSearchStat< SortPolicy >::lastDistanceNode.

22.95.3.8 `template<typename SortPolicy > void*& mlpack::neighbor::NeighborSearchStat< SortPolicy >::LastDistanceNode () [inline]`

Modify the last distance evaluation node.

Definition at line 86 of file neighbor_search_stat.hpp.

References mlpack::neighbor::NeighborSearchStat< SortPolicy >::lastDistanceNode.

22.95.3.9 `template<typename SortPolicy > double mlpack::neighbor::NeighborSearchStat< SortPolicy >::SecondBound () const [inline]`

Get the second bound.

Definition at line 76 of file neighbor_search_stat.hpp.

References mlpack::neighbor::NeighborSearchStat< SortPolicy >::secondBound.

22.95.3.10 `template<typename SortPolicy > double& mlpack::neighbor::NeighborSearchStat< SortPolicy >::SecondBound () [inline]`

Modify the second bound.

Definition at line 78 of file neighbor_search_stat.hpp.

References mlpack::neighbor::NeighborSearchStat< SortPolicy >::secondBound.

22.95.4 Member Data Documentation

22.95.4.1 `template<typename SortPolicy > double mlpack::neighbor::NeighborSearchStat< SortPolicy >::bound [private]`

The better of the two bounds.

Definition at line 40 of file neighbor_search_stat.hpp.

Referenced by mlpack::neighbor::NeighborSearchStat< SortPolicy >::Bound().

22.95.4.2 `template<typename SortPolicy > double mlpack::neighbor::NeighborSearchStat< SortPolicy >::firstBound [private]`

The first bound on the node's neighbor distances (B₁).

This represents the worst candidate distance of any descendants of this node.

Definition at line 33 of file neighbor_search_stat.hpp.

Referenced by mlpack::neighbor::NeighborSearchStat< SortPolicy >::FirstBound().

22.95.4.3 `template<typename SortPolicy > double mlpack::neighbor::NeighborSearchStat< SortPolicy >::lastDistance [private]`

The last distance evaluation.

Definition at line 45 of file neighbor_search_stat.hpp.

Referenced by `mlpack::neighbor::NeighborSearchStat< SortPolicy >::LastDistance()`.

22.95.4.4 `template<typename SortPolicy > void* mlpack::neighbor::NeighborSearchStat< SortPolicy >::lastDistanceNode [private]`

The last distance evaluation node.

Definition at line 43 of file neighbor_search_stat.hpp.

Referenced by `mlpack::neighbor::NeighborSearchStat< SortPolicy >::LastDistanceNode()`.

22.95.4.5 `template<typename SortPolicy > double mlpack::neighbor::NeighborSearchStat< SortPolicy >::secondBound [private]`

The second bound on the node's neighbor distances (`B_2`).

This represents a bound on the worst distance of any descendants of this node assembled using the best descendant candidate distance modified by the furthest descendant distance.

Definition at line 38 of file neighbor_search_stat.hpp.

Referenced by `mlpack::neighbor::NeighborSearchStat< SortPolicy >::SecondBound()`.

The documentation for this class was generated from the following file:

- `src/mlpack/methods/neighbor_search/neighbor_search_stat.hpp`

22.96 `mlpack::neighbor::NeighborSearchTraversalInfo< TreeType >` Class Template Reference

Traversal information for **NeighborSearch** (p. 399).

Public Member Functions

- **NeighborSearchTraversalInfo** ()
*Create the **TraversalInfo** (p. 673) object and initialize the pointers to NULL.*
- double **LastBaseCase** () const
Get the base case associated with the last node combination.
- double & **LastBaseCase** ()
Modify the base case associated with the last node combination.
- `TreeType *` **LastQueryNode** () const
Get the last query node.
- `TreeType *&` **LastQueryNode** ()
Modify the last query node.
- `TreeType *` **LastReferenceNode** () const
Get the last reference node.
- `TreeType *&` **LastReferenceNode** ()
Modify the last reference node.
- double **LastScore** () const

Get the score associated with the last query and reference nodes.

- double & **LastScore** ()

Modify the score associated with the last query and reference nodes.

Private Attributes

- double **lastBaseCase**

The last base case.

- TreeType * **lastQueryNode**

The last query node.

- TreeType * **lastReferenceNode**

The last reference node.

- double **lastScore**

The last distance.

22.96.1 Detailed Description

```
template<typename TreeType>class mlpack::neighbor::NeighborSearchTraversallInfo< TreeType >
```

Traversal information for **NeighborSearch** (p. 399).

This information is used to make parent-child prunes or parent-parent prunes in Score() without needing to evaluate the distance between two nodes.

The information held by this class is the last node combination visited before the current node combination was recursed into and the distance between the node centroids.

Definition at line 31 of file ns_traversal_info.hpp.

22.96.2 Constructor & Destructor Documentation

22.96.2.1 `template<typename TreeType > mlpack::neighbor::NeighborSearchTraversallInfo< TreeType >::NeighborSearchTraversallInfo() [inline]`

Create the **TraversallInfo** (p. 673) object and initialize the pointers to NULL.

Definition at line 37 of file ns_traversal_info.hpp.

22.96.3 Member Function Documentation

22.96.3.1 `template<typename TreeType > double mlpack::neighbor::NeighborSearchTraversallInfo< TreeType >::LastBaseCase() const [inline]`

Get the base case associated with the last node combination.

Definition at line 59 of file ns_traversal_info.hpp.

References mlpack::neighbor::NeighborSearchTraversallInfo< TreeType >::lastBaseCase.

22.96.3.2 `template<typename TreeType > double& mpack::neighbor::NeighborSearchTraversalInfo< TreeType >::LastBaseCase () [inline]`

Modify the base case associated with the last node combination.

Definition at line 61 of file ns_traversal_info.hpp.

References mpack::neighbor::NeighborSearchTraversalInfo< TreeType >::lastBaseCase.

22.96.3.3 `template<typename TreeType > TreeType* mpack::neighbor::NeighborSearchTraversalInfo< TreeType >::LastQueryNode () const [inline]`

Get the last query node.

Definition at line 44 of file ns_traversal_info.hpp.

References mpack::neighbor::NeighborSearchTraversalInfo< TreeType >::lastQueryNode.

22.96.3.4 `template<typename TreeType > TreeType*& mpack::neighbor::NeighborSearchTraversalInfo< TreeType >::LastQueryNode () [inline]`

Modify the last query node.

Definition at line 46 of file ns_traversal_info.hpp.

References mpack::neighbor::NeighborSearchTraversalInfo< TreeType >::lastQueryNode.

22.96.3.5 `template<typename TreeType > TreeType* mpack::neighbor::NeighborSearchTraversalInfo< TreeType >::LastReferenceNode () const [inline]`

Get the last reference node.

Definition at line 49 of file ns_traversal_info.hpp.

References mpack::neighbor::NeighborSearchTraversalInfo< TreeType >::lastReferenceNode.

22.96.3.6 `template<typename TreeType > TreeType*& mpack::neighbor::NeighborSearchTraversalInfo< TreeType >::LastReferenceNode () [inline]`

Modify the last reference node.

Definition at line 51 of file ns_traversal_info.hpp.

References mpack::neighbor::NeighborSearchTraversalInfo< TreeType >::lastReferenceNode.

22.96.3.7 `template<typename TreeType > double mpack::neighbor::NeighborSearchTraversalInfo< TreeType >::LastScore () const [inline]`

Get the score associated with the last query and reference nodes.

Definition at line 54 of file ns_traversal_info.hpp.

References mpack::neighbor::NeighborSearchTraversalInfo< TreeType >::lastScore.

22.96.3.8 `template<typename TreeType > double& mlpack::neighbor::NeighborSearchTraversallInfo< TreeType >::LastScore () [inline]`

Modify the score associated with the last query and reference nodes.

Definition at line 56 of file ns_traversal_info.hpp.

References mlpack::neighbor::NeighborSearchTraversallInfo< TreeType >::lastScore.

22.96.4 Member Data Documentation

22.96.4.1 `template<typename TreeType > double mlpack::neighbor::NeighborSearchTraversallInfo< TreeType >::lastBaseCase [private]`

The last base case.

Definition at line 71 of file ns_traversal_info.hpp.

Referenced by mlpack::neighbor::NeighborSearchTraversallInfo< TreeType >::LastBaseCase().

22.96.4.2 `template<typename TreeType > TreeType* mlpack::neighbor::NeighborSearchTraversallInfo< TreeType >::lastQueryNode [private]`

The last query node.

Definition at line 65 of file ns_traversal_info.hpp.

Referenced by mlpack::neighbor::NeighborSearchTraversallInfo< TreeType >::LastQueryNode().

22.96.4.3 `template<typename TreeType > TreeType* mlpack::neighbor::NeighborSearchTraversallInfo< TreeType >::lastReferenceNode [private]`

The last reference node.

Definition at line 67 of file ns_traversal_info.hpp.

Referenced by mlpack::neighbor::NeighborSearchTraversallInfo< TreeType >::LastReferenceNode().

22.96.4.4 `template<typename TreeType > double mlpack::neighbor::NeighborSearchTraversallInfo< TreeType >::lastScore [private]`

The last distance.

Definition at line 69 of file ns_traversal_info.hpp.

Referenced by mlpack::neighbor::NeighborSearchTraversallInfo< TreeType >::LastScore().

The documentation for this class was generated from the following file:

- src/mlpack/methods/neighbor_search/ns_traversal_info.hpp

22.97 mlpack::neighbor::RASearchRules< SortPolicy, MetricType, TreeType > Class Template Reference

Public Types

- typedef **neighbor::NeighborSearchTraversallInfo** < TreeType > **TraversallInfoType**

Public Member Functions

- **RASearchRules** (const arma::mat &**referenceSet**, const arma::mat &**querySet**, arma::Mat< size_t > &**neighbors**, arma::mat &**distances**, MetricType &**metric**, const double tau=5, const double **alpha**=0.95, const bool naive=false, const bool **sampleAtLeaves**=false, const bool **firstLeafExact**=false, const size_t **singleSampleLimit**=20)
- double **BaseCase** (const size_t queryIndex, const size_t referenceIndex)
- size_t **NumDistComputations** ()
- size_t **NumEffectiveSamples** ()
- double **Rescore** (const size_t queryIndex, TreeType &referenceNode, const double oldScore)
Re-evaluate the score for recursion order.
- double **Rescore** (TreeType &queryNode, TreeType &referenceNode, const double oldScore)
Re-evaluate the score for recursion order.
- double **Score** (const size_t queryIndex, TreeType &referenceNode)
Get the score for recursion order.
- double **Score** (const size_t queryIndex, TreeType &referenceNode, const double baseCaseResult)
Get the score for recursion order.
- double **Score** (TreeType &queryNode, TreeType &referenceNode)
Get the score for recursion order.
- double **Score** (TreeType &queryNode, TreeType &referenceNode, const double baseCaseResult)
Get the score for recursion order, passing the base case result (in the situation where it may be needed to calculate the recursion order).
- const **TraversallInfoType** & **TraversallInfo** () const
- **TraversallInfoType** & **TraversallInfo** ()

Private Member Functions

- void **InsertNeighbor** (const size_t queryIndex, const size_t pos, const size_t neighbor, const double distance)
Insert a point into the neighbors and distances matrices; this is a helper function.
- size_t **MinimumSamplesReqd** (const size_t n, const size_t k, const double tau, const double **alpha**) const
Compute the minimum number of samples required to guarantee the given rank-approximation and success probability.
- void **ObtainDistinctSamples** (const size_t numSamples, const size_t rangeUpperBound, arma::uvec &distinctSamples) const
Pick up desired number of samples (with replacement) from a given range of integers so that only the distinct samples are returned from the range [0 - specified upper bound)
- double **Score** (const size_t queryIndex, TreeType &referenceNode, const double distance, const double bestDistance)
Perform actual scoring for single-tree case.
- double **Score** (TreeType &queryNode, TreeType &referenceNode, const double distance, const double bestDistance)
Perform actual scoring for dual-tree case.
- double **SuccessProbability** (const size_t n, const size_t k, const size_t m, const size_t t) const
Compute the success probability of obtaining 'k'-neighbors from a set of size 'n' within the top 't' neighbors if 'm' samples are made.

Private Attributes

- arma::mat & **distances**
The matrix the resultant neighbor distances should be stored in.
- bool **firstLeafExact**
Whether to do exact computation on the first leaf before any sampling.
- MetricType & **metric**
The instantiated metric.
- arma::Mat< size_t > & **neighbors**
The matrix the resultant neighbor indices should be stored in.
- size_t **numDistComputations**
- arma::Col< size_t > **numSamplesMade**
The number of samples made for every query.
- size_t **numSamplesReqd**
The minimum number of samples required per query.
- const arma::mat & **querySet**
The query set.
- const arma::mat & **referenceSet**
The reference set.
- bool **sampleAtLeaves**
Whether to sample at leaves or just use all of it.
- double **samplingRatio**
The sampling ratio.
- size_t **singleSampleLimit**
The limit on the largest node that can be approximated by sampling.
- TraversalInfoType **traversalInfo**

Friends

- class **RASearch**< SortPolicy, MetricType, TreeType >

22.97.1 Detailed Description

template<typename SortPolicy, typename MetricType, typename TreeType>class mlpack::neighbor::RASearchRules< SortPolicy, MetricType, TreeType >

Definition at line 26 of file ra_search_rules.hpp.

22.97.2 Member Typedef Documentation

22.97.2.1 template<typename SortPolicy , typename MetricType , typename TreeType > typedef neighbor::Neighbor↔ SearchTraversalInfo<TreeType> mlpack::neighbor::RASearchRules< SortPolicy, MetricType, TreeType >::TraversalInfoType

Definition at line 197 of file ra_search_rules.hpp.

22.97.3 Constructor & Destructor Documentation

22.97.3.1 `template<typename SortPolicy , typename MetricType , typename TreeType > mlpack::neighbor::RASearchRules< SortPolicy, MetricType, TreeType >::RASearchRules (const arma::mat & referenceSet, const arma::mat & querySet, arma::Mat< size_t > & neighbors, arma::mat & distances, MetricType & metric, const double tau = 5, const double alpha = 0.95, const bool naive = false, const bool sampleAtLeaves = false, const bool firstLeafExact = false, const size_t singleSampleLimit = 20)`

22.97.4 Member Function Documentation

22.97.4.1 `template<typename SortPolicy , typename MetricType , typename TreeType > double mlpack::neighbor::RA↔SearchRules< SortPolicy, MetricType, TreeType >::BaseCase (const size_t queryIndex, const size_t referenceIndex)`

22.97.4.2 `template<typename SortPolicy , typename MetricType , typename TreeType > void mlpack::neighbor::RASearch↔Rules< SortPolicy, MetricType, TreeType >::InsertNeighbor (const size_t queryIndex, const size_t pos, const size_t neighbor, const double distance) [private]`

Insert a point into the neighbors and distances matrices; this is a helper function.

Parameters

<i>queryIndex</i>	Index of point whose neighbors we are inserting into.
<i>pos</i>	Position in list to insert into.
<i>neighbor</i>	Index of reference point which is being inserted.
<i>distance</i>	Distance from query point to reference point.

22.97.4.3 `template<typename SortPolicy , typename MetricType , typename TreeType > size_t mlpack::neighbor::RA↔SearchRules< SortPolicy, MetricType, TreeType >::MinimumSamplesReqd (const size_t n, const size_t k, const double tau, const double alpha) const [private]`

Compute the minimum number of samples required to guarantee the given rank-approximation and success probability.

Parameters

<i>n</i>	Size of the set to be sampled from.
<i>k</i>	The number of neighbors required within the rank-approximation.
<i>tau</i>	The rank-approximation in percentile of the data.
<i>alpha</i>	The success probability desired.

22.97.4.4 `template<typename SortPolicy , typename MetricType , typename TreeType > size_t mlpack↔::neighbor::RASearchRules< SortPolicy, MetricType, TreeType >::NumDistComputations () [inline]`

Definition at line 188 of file `ra_search_rules.hpp`.

References `mlpack::neighbor::RASearchRules< SortPolicy, MetricType, TreeType >::numDistComputations`.

22.97.4.5 `template<typename SortPolicy , typename MetricType , typename TreeType > size_t mlpack::neighbor::RASearchRules< SortPolicy, MetricType, TreeType >::NumEffectiveSamples ()`
`[inline]`

Definition at line 189 of file `ra_search_rules.hpp`.

References `mlpack::neighbor::RASearchRules< SortPolicy, MetricType, TreeType >::numSamplesMade`.

22.97.4.6 `template<typename SortPolicy , typename MetricType , typename TreeType > void mlpack::neighbor::RASearchRules< SortPolicy, MetricType, TreeType >::ObtainDistinctSamples (const size_t numSamples, const size_t rangeUpperBound, arma::uvec & distinctSamples) const` `[private]`

Pick up desired number of samples (with replacement) from a given range of integers so that only the distinct samples are returned from the range [0 - specified upper bound)

Parameters

<i>numSamples</i>	Number of random samples.
<i>rangeUpperBound</i>	The upper bound on the range of integers.
<i>distinctSamples</i>	The list of the distinct samples.

22.97.4.7 `template<typename SortPolicy , typename MetricType , typename TreeType > double mlpack::neighbor::RASearchRules< SortPolicy, MetricType, TreeType >::Rescore (const size_t queryIndex, TreeType & referenceNode, const double oldScore)`

Re-evaluate the score for recursion order.

A low score indicates priority for recursion, while `DBL_MAX` indicates that the node should not be recursed into at all (it should be pruned). This is used when the score has already been calculated, but another recursion may have modified the bounds for pruning. So the old score is checked against the new pruning bound.

For rank-approximation, it also checks if the number of samples left for a query to satisfy the rank constraint is small enough at this point of the algorithm, then this node is approximated by sampling and given a new score of 'DBL_MAX'.

Parameters

<i>queryIndex</i>	Index of query point.
<i>referenceNode</i>	Candidate node to be recursed into.
<i>oldScore</i>	Old score produced by Score() (p. 425) (or Rescore() (p. 423)).

22.97.4.8 `template<typename SortPolicy , typename MetricType , typename TreeType > double mlpack::neighbor::RASearchRules< SortPolicy, MetricType, TreeType >::Rescore (TreeType & queryNode, TreeType & referenceNode, const double oldScore)`

Re-evaluate the score for recursion order.

A low score indicates priority for recursion, while `DBL_MAX` indicates that the node should not be recursed into at all (it should be pruned). This is used when the score has already been calculated, but another recursion may have modified the bounds for pruning. So the old score is checked against the new pruning bound.

For the rank-approximation, we check if the referenceNode can be approximated by sampling. If it can be, enough samples are made for every query in the queryNode. No further query-tree traversal is performed.

The 'NumSamplesMade' query stat is propagated up the tree. And then if pruning occurs (by distance or by sampling), the 'NumSamplesMade' stat is not propagated down the tree. If no pruning occurs, the stat is propagated down the tree.

Parameters

<i>queryNode</i>	Candidate query node to recurse into.
<i>referenceNode</i>	Candidate reference node to recurse into.
<i>oldScore</i>	Old score produced by Socre() (or Rescore() (p. 423)).

22.97.4.9 `template<typename SortPolicy , typename MetricType , typename TreeType > double mlpack::neighbor::RA↔
SearchRules< SortPolicy, MetricType, TreeType >::Score (const size_t queryIndex, TreeType & referenceNode
)`

Get the score for recursion order.

A low score indicates priority for recursion, while DBL_MAX indicates that the node should not be recursed into at all (it should be pruned).

For rank-approximation, the scoring function first checks if pruning by distance is possible. If yes, then the node is given the score of 'DBL_MAX' and the expected number of samples from that node are added to the number of samples made for the query.

If no, then the function tries to see if the node can be pruned by approximation. If number of samples required from this node is small enough, then that number of samples are acquired from this node and the score is set to be 'DBL_MAX'.

If the pruning by approximation is not possible either, the algorithm continues with the usual tree-traversal.

Parameters

<i>queryIndex</i>	Index of query point.
<i>referenceNode</i>	Candidate node to be recursed into.

22.97.4.10 `template<typename SortPolicy , typename MetricType , typename TreeType > double mlpack::neighbor::RA↔
SearchRules< SortPolicy, MetricType, TreeType >::Score (const size_t queryIndex, TreeType & referenceNode,
const double baseCaseResult)`

Get the score for recursion order.

A low score indicates priority for recursion, while DBL_MAX indicates that the node should not be recursed into at all (it should be pruned).

For rank-approximation, the scoring function first checks if pruning by distance is possible. If yes, then the node is given the score of 'DBL_MAX' and the expected number of samples from that node are added to the number of samples made for the query.

If no, then the function tries to see if the node can be pruned by approximation. If number of samples required from this node is small enough, then that number of samples are acquired from this node and the score is set to be 'DBL_MAX'.

If the pruning by approximation is not possible either, the algorithm continues with the usual tree-traversal.

Parameters

<i>queryIndex</i>	Index of query point.
<i>referenceNode</i>	Candidate node to be recursed into.
<i>baseCaseResult</i>	Result of BaseCase(queryIndex, referenceNode).

```
22.97.4.11  template<typename SortPolicy , typename MetricType , typename TreeType > double mlpack::neighbor::RA↵
SearchRules< SortPolicy, MetricType, TreeType >::Score ( TreeType & queryNode, TreeType & referenceNode
)
```

Get the score for recursion order.

A low score indicates priority for recursionm while DBL_MAX indicates that the node should not be recursed into at all (it should be pruned).

For the rank-approximation, we check if the referenceNode can be approximated by sampling. If it can be, enough samples are made for every query in the queryNode. No further query-tree traversal is performed.

The 'NumSamplesMade' query stat is propagated up the tree. And then if pruning occurs (by distance or by sampling), the 'NumSamplesMade' stat is not propagated down the tree. If no pruning occurs, the stat is propagated down the tree.

Parameters

<i>queryNode</i>	Candidate query node to recurse into.
<i>referenceNode</i>	Candidate reference node to recurse into.

```
22.97.4.12  template<typename SortPolicy , typename MetricType , typename TreeType > double mlpack::neighbor::RA↵
SearchRules< SortPolicy, MetricType, TreeType >::Score ( TreeType & queryNode, TreeType & referenceNode, const
double baseCaseResult )
```

Get the score for recursion order, passing the base case result (in the situation where it may be needed to calculate the recursion order).

A low score indicates priority for recursion, while DBL_MAX indicates that the node should not be recursed into at all (it should be pruned).

For the rank-approximation, we check if the referenceNode can be approximated by sampling. If it can be, enough samples are made for every query in the queryNode. No further query-tree traversal is performed.

The 'NumSamplesMade' query stat is propagated up the tree. And then if pruning occurs (by distance or by sampling), the 'NumSamplesMade' stat is not propagated down the tree. If no pruning occurs, the stat is propagated down the tree.

Parameters

<i>queryNode</i>	Candidate query node to recurse into.
<i>referenceNode</i>	Candidate reference node to recurse into.
<i>baseCaseResult</i>	Result of BaseCase(queryIndex, referenceNode).

```
22.97.4.13  template<typename SortPolicy , typename MetricType , typename TreeType > double mlpack::neighbor::RA↵
SearchRules< SortPolicy, MetricType, TreeType >::Score ( const size_t queryIndex, TreeType & referenceNode,
const double distance, const double bestDistance ) [private]
```

Perform actual scoring for single-tree case.

```
22.97.4.14  template<typename SortPolicy , typename MetricType , typename TreeType > double mlpack::neighbor::RA↵
SearchRules< SortPolicy, MetricType, TreeType >::Score ( TreeType & queryNode, TreeType & referenceNode, const
double distance, const double bestDistance ) [private]
```

Perform actual scoring for dual-tree case.

22.97.4.15 `template<typename SortPolicy , typename MetricType , typename TreeType > double mlpack::neighbor::RA←
SearchRules< SortPolicy, MetricType, TreeType >::SuccessProbability (const size_t n, const size_t k, const size_t
m, const size_t t) const [private]`

Compute the success probability of obtaining 'k'-neighbors from a set of size 'n' within the top 't' neighbors if 'm' samples are made.

Parameters

n	Size of the set being sampled from.
k	The number of neighbors required within the rank-approximation.
m	The number of random samples.
t	The desired rank-approximation.

22.97.4.16 `template<typename SortPolicy , typename MetricType , typename TreeType > const TraversalInfoType& mlpack::neighbor::RASearchRules< SortPolicy, MetricType, TreeType >::TraversalInfo () const [inline]`

Definition at line 199 of file `ra_search_rules.hpp`.

References `mlpack::neighbor::RASearchRules< SortPolicy, MetricType, TreeType >::traversalInfo`.

22.97.4.17 `template<typename SortPolicy , typename MetricType , typename TreeType > TraversalInfoType& mlpack::neighbor::RASearchRules< SortPolicy, MetricType, TreeType >::TraversalInfo () [inline]`

Definition at line 200 of file `ra_search_rules.hpp`.

References `mlpack::neighbor::RASearchRules< SortPolicy, MetricType, TreeType >::traversalInfo`.

22.97.5 Friends And Related Function Documentation

22.97.5.1 `template<typename SortPolicy , typename MetricType , typename TreeType > friend class RASearch< SortPolicy, MetricType, TreeType > [friend]`

Definition at line 315 of file `ra_search_rules.hpp`.

22.97.6 Member Data Documentation

22.97.6.1 `template<typename SortPolicy , typename MetricType , typename TreeType > arma::mat& mlpack::neighbor::RASearchRules< SortPolicy, MetricType, TreeType >::distances [private]`

The matrix the resultant neighbor distances should be stored in.

Definition at line 213 of file `ra_search_rules.hpp`.

22.97.6.2 `template<typename SortPolicy , typename MetricType , typename TreeType > bool mlpack::neighbor::RASearchRules< SortPolicy, MetricType, TreeType >::firstLeafExact [private]`

Whether to do exact computation on the first leaf before any sampling.

Definition at line 222 of file `ra_search_rules.hpp`.

22.97.6.3 `template<typename SortPolicy , typename MetricType , typename TreeType > MetricType& mlpack::neighbor::RASearchRules< SortPolicy, MetricType, TreeType >::metric [private]`

The instantiated metric.

Definition at line 216 of file `ra_search_rules.hpp`.

22.97.6.4 `template<typename SortPolicy , typename MetricType , typename TreeType > arma::Mat<size_t>&
mlpack::neighbor::RASearchRules< SortPolicy, MetricType, TreeType >::neighbors [private]`

The matrix the resultant neighbor indices should be stored in.

Definition at line 210 of file `ra_search_rules.hpp`.

22.97.6.5 `template<typename SortPolicy , typename MetricType , typename TreeType > size_t mlpack←
::neighbor::RASearchRules< SortPolicy, MetricType, TreeType >::numDistComputations
[private]`

Definition at line 237 of file `ra_search_rules.hpp`.

Referenced by `mlpack::neighbor::RASearchRules< SortPolicy, MetricType, TreeType >::NumDistComputations()`.

22.97.6.6 `template<typename SortPolicy , typename MetricType , typename TreeType > arma::Col<size_t>
mlpack::neighbor::RASearchRules< SortPolicy, MetricType, TreeType >::numSamplesMade [private]`

The number of samples made for every query.

Definition at line 231 of file `ra_search_rules.hpp`.

Referenced by `mlpack::neighbor::RASearchRules< SortPolicy, MetricType, TreeType >::NumEffectiveSamples()`.

22.97.6.7 `template<typename SortPolicy , typename MetricType , typename TreeType > size_t
mlpack::neighbor::RASearchRules< SortPolicy, MetricType, TreeType >::numSamplesReqd [private]`

The minimum number of samples required per query.

Definition at line 228 of file `ra_search_rules.hpp`.

22.97.6.8 `template<typename SortPolicy , typename MetricType , typename TreeType > const arma::mat&
mlpack::neighbor::RASearchRules< SortPolicy, MetricType, TreeType >::querySet [private]`

The query set.

Definition at line 207 of file `ra_search_rules.hpp`.

22.97.6.9 `template<typename SortPolicy , typename MetricType , typename TreeType > const arma::mat&
mlpack::neighbor::RASearchRules< SortPolicy, MetricType, TreeType >::referenceSet [private]`

The reference set.

Definition at line 204 of file `ra_search_rules.hpp`.

22.97.6.10 `template<typename SortPolicy , typename MetricType , typename TreeType > bool
mlpack::neighbor::RASearchRules< SortPolicy, MetricType, TreeType >::sampleAtLeaves [private]`

Whether to sample at leaves or just use all of it.

Definition at line 219 of file `ra_search_rules.hpp`.

22.97.6.11 `template<typename SortPolicy , typename MetricType , typename TreeType > double
mlpack::neighbor::RASearchRules< SortPolicy, MetricType, TreeType >::samplingRatio [private]`

The sampling ratio.

Definition at line 234 of file `ra_search_rules.hpp`.

22.97.6.12 `template<typename SortPolicy , typename MetricType , typename TreeType > size_t
mlpack::neighbor::RASearchRules< SortPolicy, MetricType, TreeType >::singleSampleLimit [private]`

The limit on the largest node that can be approximated by sampling.

Definition at line 225 of file `ra_search_rules.hpp`.

22.97.6.13 `template<typename SortPolicy , typename MetricType , typename TreeType > TraversalInfoType
mlpack::neighbor::RASearchRules< SortPolicy, MetricType, TreeType >::traversalInfo [private]`

Definition at line 239 of file `ra_search_rules.hpp`.

Referenced by `mlpack::neighbor::RASearchRules< SortPolicy, MetricType, TreeType >::TraversalInfo()`.

The documentation for this class was generated from the following file:

- `src/mlpack/methods/rann/ra_search_rules.hpp`

22.98 mlpack::nn::SparseAutoencoder< OptimizerType > Class Template Reference

A sparse autoencoder is a neural network whose aim to learn compressed representations of the data, typically for dimensionality reduction, with a constraint on the activity of the neurons in the network.

Public Member Functions

- **SparseAutoencoder** (const arma::mat &data, const size_t **visibleSize**, const size_t **hiddenSize**, const double **lambda**=0.0001, const double **beta**=3, const double **rho**=0.01)
Construct the sparse autoencoder model with the given training data.
- **SparseAutoencoder** (OptimizerType< **SparseAutoencoderFunction** > &optimizer)
Construct the sparse autoencoder model with the given training data.
- void **Beta** (const double b)
Sets the KL divergence parameter.
- double **Beta** () const
Gets the KL divergence parameter.
- void **GetNewFeatures** (arma::mat &data, arma::mat &features)
Transforms the provided data into the representation learned by the sparse autoencoder.
- void **HiddenSize** (const size_t hidden)
Sets size of the hidden layer.
- size_t **HiddenSize** () const
Gets the size of the hidden layer.
- void **Lambda** (const double l)
Sets the L2-regularization parameter.

- double **Lambda** () const
Gets the L2-regularization parameter.
- void **Rho** (const double r)
Sets the sparsity parameter.
- double **Rho** () const
Gets the sparsity parameter.
- void **Sigmoid** (const arma::mat &x, arma::mat &output) const
Returns the elementwise sigmoid of the passed matrix, where the sigmoid function of a real number 'x' is $[1 / (1 + \exp(-x))]$.
- void **VisibleSize** (const size_t visible)
Sets size of the visible layer.
- size_t **VisibleSize** () const
Gets size of the visible layer.

Private Attributes

- double **beta**
KL divergence parameter.
- size_t **hiddenSize**
Size of the hidden layer.
- double **lambda**
L2-regularization parameter.
- arma::mat **parameters**
Parameters after optimization.
- double **rho**
Sparsity parameter.
- size_t **visibleSize**
Size of the visible layer.

22.98.1 Detailed Description

```
template<template< typename > class OptimizerType = mlpack::optimization::L_BFGS>class mlpack::nn::SparseAutoencoder<
OptimizerType >
```

A sparse autoencoder is a neural network whose aim to learn compressed representations of the data, typically for dimensionality reduction, with a constraint on the activity of the neurons in the network.

Sparse autoencoders can be stacked together to learn a hierarchy of features, which provide a better representation of the data for classification. This is a method used in the recently developed field of deep learning. More technical details about the model can be found on the following webpage:

http://deeplearning.stanford.edu/wiki/index.php/UFLDL_Tutorial

An example of how to use the interface is shown below:

```
arma::mat data; // Data matrix.
const size_t vSize = 64; // Size of visible layer, depends on the data.
const size_t hSize = 25; // Size of hidden layer, depends on requirements.

// Train the model using default options.
SparseAutoencoder encoder1(data, vSize, hSize);

const size_t numBasis = 5; // Parameter required for L-BFGS algorithm.
const size_t numIterations = 100; // Maximum number of iterations.
```

```
// Use an instantiated optimizer for the training.
SparseAutoencoderFunction saf(data, vSize, hSize);
L_BFGS<SparseAutoencoderFunction> optimizer(saf, numBasis, numIterations);
SparseAutoencoder<L_BFGS> encoder2(optimizer);

arma::mat features1, features2; // Matrices for storing new representations.

// Get new representations from the trained models.
encoder1.GetNewFeatures(data, features1);
encoder2.GetNewFeatures(data, features2);
```

This implementation allows the use of arbitrary mlpack optimizers via the `OptimizerType` template parameter.

Template Parameters

<i>OptimizerType</i>	The optimizer to use; by default this is L-BFGS. Any mlpack optimizer can be used here.
----------------------	---

Definition at line 70 of file `sparse_autoencoder.hpp`.

22.98.2 Constructor & Destructor Documentation

22.98.2.1 `template<template< typename > class OptimizerType = mlpack::optimization::L_BFGS> mlpack::nn::SparseAutoencoder< OptimizerType >::SparseAutoencoder(const arma::mat & data, const size_t visibleSize, const size_t hiddenSize, const double lambda = 0.0001, const double beta = 3, const double rho = 0.01)`

Construct the sparse autoencoder model with the given training data.

This will train the model. The parameters 'lambda', 'beta' and 'rho' can be set optionally. Changing these parameters will have an effect on regularization and sparsity of the model.

Parameters

<i>data</i>	Input data with each column as one example.
<i>visibleSize</i>	Size of input vector expected at the visible layer.
<i>hiddenSize</i>	Size of input vector expected at the hidden layer.
<i>lambda</i>	L2-regularization parameter.
<i>beta</i>	KL divergence parameter.
<i>rho</i>	Sparsity parameter.

22.98.2.2 `template<template< typename > class OptimizerType = mlpack::optimization::L_BFGS> mlpack::nn::SparseAutoencoder< OptimizerType >::SparseAutoencoder(OptimizerType< SparseAutoencoderFunction > & optimizer)`

Construct the sparse autoencoder model with the given training data.

This will train the model. This overload takes an already instantiated optimizer and uses it to train the model. The optimizer should hold an instantiated **SparseAutoencoderFunction** (p. 436) object for the function to operate upon. This option should be preferred when the optimizer options are to be changed.

Parameters

<i>optimizer</i>	Instantiated optimizer with instantiated error function.
------------------	--

22.98.3 Member Function Documentation

22.98.3.1 `template<template< typename > class OptimizerType = mlpack::optimization::L_BFGS> void
mlpack::nn::SparseAutoencoder< OptimizerType >::Beta (const double b) [inline]`

Sets the KL divergence parameter.

Definition at line 162 of file `sparse_autoencoder.hpp`.

References `mlpack::nn::SparseAutoencoder< OptimizerType >::beta`.

22.98.3.2 `template<template< typename > class OptimizerType = mlpack::optimization::L_BFGS> double
mlpack::nn::SparseAutoencoder< OptimizerType >::Beta () const [inline]`

Gets the KL divergence parameter.

Definition at line 168 of file `sparse_autoencoder.hpp`.

References `mlpack::nn::SparseAutoencoder< OptimizerType >::beta`.

22.98.3.3 `template<template< typename > class OptimizerType = mlpack::optimization::L_BFGS> void
mlpack::nn::SparseAutoencoder< OptimizerType >::GetNewFeatures (arma::mat & data, arma::mat & features)`

Transforms the provided data into the representation learned by the sparse autoencoder.

The function basically performs a feedforward computation using the learned weights, and returns the hidden layer activations.

Parameters

<i>data</i>	Matrix of the provided data.
<i>features</i>	The hidden layer representation of the provided data.

22.98.3.4 `template<template< typename > class OptimizerType = mlpack::optimization::L_BFGS> void
mlpack::nn::SparseAutoencoder< OptimizerType >::HiddenSize (const size_t hidden) [inline]`

Sets size of the hidden layer.

Definition at line 138 of file `sparse_autoencoder.hpp`.

22.98.3.5 `template<template< typename > class OptimizerType = mlpack::optimization::L_BFGS> size_t
mlpack::nn::SparseAutoencoder< OptimizerType >::HiddenSize () const [inline]`

Gets the size of the hidden layer.

Definition at line 144 of file `sparse_autoencoder.hpp`.

References `mlpack::nn::SparseAutoencoder< OptimizerType >::hiddenSize`.

22.98.3.6 `template<template< typename > class OptimizerType = mlpack::optimization::L_BFGS> void
mlpack::nn::SparseAutoencoder< OptimizerType >::Lambda (const double l) [inline]`

Sets the L2-regularization parameter.

Definition at line 150 of file `sparse_autoencoder.hpp`.

References `mlpack::nn::SparseAutoencoder< OptimizerType >::lambda`.

22.98.3.7 `template<template< typename > class OptimizerType = mlpack::optimization::L_BFGS> double
mlpack::nn::SparseAutoencoder< OptimizerType >::Lambda () const [inline]`

Gets the L2-regularization parameter.

Definition at line 156 of file `sparse_autoencoder.hpp`.

References `mlpack::nn::SparseAutoencoder< OptimizerType >::lambda`.

22.98.3.8 `template<template< typename > class OptimizerType = mlpack::optimization::L_BFGS> void
mlpack::nn::SparseAutoencoder< OptimizerType >::Rho (const double r) [inline]`

Sets the sparsity parameter.

Definition at line 174 of file `sparse_autoencoder.hpp`.

References `mlpack::nn::SparseAutoencoder< OptimizerType >::rho`.

22.98.3.9 `template<template< typename > class OptimizerType = mlpack::optimization::L_BFGS> double
mlpack::nn::SparseAutoencoder< OptimizerType >::Rho () const [inline]`

Gets the sparsity parameter.

Definition at line 180 of file `sparse_autoencoder.hpp`.

References `mlpack::nn::SparseAutoencoder< OptimizerType >::rho`.

22.98.3.10 `template<template< typename > class OptimizerType = mlpack::optimization::L_BFGS> void
mlpack::nn::SparseAutoencoder< OptimizerType >::Sigmoid (const arma::mat & x, arma::mat & output) const
[inline]`

Returns the elementwise sigmoid of the passed matrix, where the sigmoid function of a real number 'x' is $[1 / (1 + \exp(-x))]$.

Parameters

<code>x</code>	Matrix of real values for which we require the sigmoid activation.
----------------	--

Definition at line 120 of file `sparse_autoencoder.hpp`.

22.98.3.11 `template<template< typename > class OptimizerType = mlpack::optimization::L_BFGS> void
mlpack::nn::SparseAutoencoder< OptimizerType >::VisibleSize (const size_t visible) [inline]`

Sets size of the visible layer.

Definition at line 126 of file `sparse_autoencoder.hpp`.

22.98.3.12 `template<template< typename > class OptimizerType = mlpack::optimization::L_BFGS> size_t
mlpack::nn::SparseAutoencoder< OptimizerType >::VisibleSize () const [inline]`

Gets size of the visible layer.

Definition at line 132 of file `sparse_autoencoder.hpp`.

References `mlpack::nn::SparseAutoencoder< OptimizerType >::visibleSize`.

22.98.4 Member Data Documentation

22.98.4.1 `template<template< typename > class OptimizerType = mlpack::optimization::L_BFGS> double
mlpack::nn::SparseAutoencoder< OptimizerType >::beta [private]`

KL divergence parameter.

Definition at line 195 of file `sparse_autoencoder.hpp`.

Referenced by `mlpack::nn::SparseAutoencoder< OptimizerType >::Beta()`.

22.98.4.2 `template<template< typename > class OptimizerType = mlpack::optimization::L_BFGS> size_t
mlpack::nn::SparseAutoencoder< OptimizerType >::hiddenSize [private]`

Size of the hidden layer.

Definition at line 191 of file `sparse_autoencoder.hpp`.

Referenced by `mlpack::nn::SparseAutoencoder< OptimizerType >::HiddenSize()`.

22.98.4.3 `template<template< typename > class OptimizerType = mlpack::optimization::L_BFGS> double
mlpack::nn::SparseAutoencoder< OptimizerType >::lambda [private]`

L2-regularization parameter.

Definition at line 193 of file `sparse_autoencoder.hpp`.

Referenced by `mlpack::nn::SparseAutoencoder< OptimizerType >::Lambda()`.

22.98.4.4 `template<template< typename > class OptimizerType = mlpack::optimization::L_BFGS> arma::mat
mlpack::nn::SparseAutoencoder< OptimizerType >::parameters [private]`

Parameters after optimization.

Definition at line 187 of file `sparse_autoencoder.hpp`.

22.98.4.5 `template<template< typename > class OptimizerType = mlpack::optimization::L_BFGS> double
mlpack::nn::SparseAutoencoder< OptimizerType >::rho [private]`

Sparsity parameter.

Definition at line 197 of file `sparse_autoencoder.hpp`.

Referenced by `mlpack::nn::SparseAutoencoder< OptimizerType >::Rho()`.

22.98.4.6 `template<template< typename > class OptimizerType = mlpack::optimization::L_BFGS> size_t
mlpack::nn::SparseAutoencoder< OptimizerType >::visibleSize [private]`

Size of the visible layer.

Definition at line 189 of file `sparse_autoencoder.hpp`.

Referenced by `mlpack::nn::SparseAutoencoder< OptimizerType >::VisibleSize()`.

The documentation for this class was generated from the following file:

- `src/mlpack/methods/sparse_autoencoder/sparse_autoencoder.hpp`

22.99 mlpack::nn::SparseAutoencoderFunction Class Reference

This is a class for the sparse autoencoder objective function.

Public Member Functions

- **SparseAutoencoderFunction** (const arma::mat &**data**, const size_t **visibleSize**, const size_t **hiddenSize**, const double **lambda**=0.0001, const double **beta**=3, const double **rho**=0.01)
Construct the sparse autoencoder objective function with the given parameters.
- void **Beta** (const double b)
Sets the KL divergence parameter.
- double **Beta** () const
Gets the KL divergence parameter.
- double **Evaluate** (const arma::mat ¶meters) const
Evaluates the objective function of the sparse autoencoder model using the given parameters.
- const arma::mat & **GetInitialPoint** () const
Return the initial point for the optimization.
- void **Gradient** (const arma::mat ¶meters, arma::mat &gradient) const
Evaluates the gradient values of the objective function given the current set of parameters.
- void **HiddenSize** (const size_t hidden)
Sets size of the hidden layer.
- size_t **HiddenSize** () const
Gets the size of the hidden layer.
- const arma::mat **InitializeWeights** ()
Initializes the parameters of the model to suitable values.
- void **Lambda** (const double l)
Sets the L2-regularization parameter.
- double **Lambda** () const
Gets the L2-regularization parameter.
- void **Rho** (const double r)
Sets the sparsity parameter.
- double **Rho** () const
Gets the sparsity parameter.
- void **Sigmoid** (const arma::mat &x, arma::mat &output) const
Returns the elementwise sigmoid of the passed matrix, where the sigmoid function of a real number 'x' is $[1 / (1 + \exp(-x))]$.
- void **VisibleSize** (const size_t visible)
Sets size of the visible layer.
- size_t **VisibleSize** () const
Gets size of the visible layer.

Private Attributes

- double **beta**
KL divergence parameter.
- const arma::mat & **data**
The matrix of data points.
- size_t **hiddenSize**

Size of the hidden layer.

- arma::mat **initialPoint**

Initial parameter vector.

- double **lambda**

L2-regularization parameter.

- double **rho**

Sparsity parameter.

- size_t **visibleSize**

Size of the visible layer.

22.99.1 Detailed Description

This is a class for the sparse autoencoder objective function.

It can be used to create learning models like self-taught learning, stacked autoencoders, conditional random fields (CRFs), and so forth.

Definition at line 28 of file `sparse_autoencoder_function.hpp`.

22.99.2 Constructor & Destructor Documentation

22.99.2.1 `mlpack::nn::SparseAutoencoderFunction::SparseAutoencoderFunction (const arma::mat & data, const size_t visibleSize, const size_t hiddenSize, const double lambda = 0.0001, const double beta = 3, const double rho = 0.01)`

Construct the sparse autoencoder objective function with the given parameters.

Parameters

<i>data</i>	The data matrix.
<i>visibleSize</i>	Size of input vector expected at the visible layer.
<i>hiddenSize</i>	Size of input vector expected at the hidden layer.
<i>lambda</i>	L2-regularization parameter.
<i>beta</i>	KL divergence parameter.
<i>rho</i>	Sparsity parameter.

22.99.3 Member Function Documentation

22.99.3.1 `void mlpack::nn::SparseAutoencoderFunction::Beta (const double b) [inline]`

Sets the KL divergence parameter.

Definition at line 126 of file `sparse_autoencoder_function.hpp`.

References `beta`.

22.99.3.2 `double mlpack::nn::SparseAutoencoderFunction::Beta () const [inline]`

Gets the KL divergence parameter.

Definition at line 132 of file `sparse_autoencoder_function.hpp`.

References `beta`.

22.99.3.3 `double mlpack::nn::SparseAutoencoderFunction::Evaluate (const arma::mat & parameters) const`

Evaluates the objective function of the sparse autoencoder model using the given parameters.

The cost function has terms for the reconstruction error, regularization cost and the sparsity cost. The objective function takes a low value when the model is able to reconstruct the data well using weights which are low in value and when the average activations of neurons in the hidden layers agrees well with the sparsity parameter 'rho'.

Parameters

<i>parameters</i>	Current values of the model parameters.
-------------------	---

22.99.3.4 `const arma::mat& mlpack::nn::SparseAutoencoderFunction::GetInitialPoint () const` `[inline]`

Return the initial point for the optimization.

Definition at line 87 of file `sparse_autoencoder_function.hpp`.

References `initialPoint`.

22.99.3.5 `void mlpack::nn::SparseAutoencoderFunction::Gradient (const arma::mat & parameters, arma::mat & gradient) const`

Evaluates the gradient values of the objective function given the current set of parameters.

The function performs a feedforward pass and computes the error in reconstructing the data points. It then uses the backpropagation algorithm to compute the gradient values.

Parameters

<i>parameters</i>	Current values of the model parameters.
<i>gradient</i>	Matrix where gradient values will be stored.

22.99.3.6 `void mlpack::nn::SparseAutoencoderFunction::HiddenSize (const size_t hidden)` `[inline]`

Sets size of the hidden layer.

Definition at line 102 of file `sparse_autoencoder_function.hpp`.

22.99.3.7 `size_t mlpack::nn::SparseAutoencoderFunction::HiddenSize () const` `[inline]`

Gets the size of the hidden layer.

Definition at line 108 of file `sparse_autoencoder_function.hpp`.

References `hiddenSize`.

22.99.3.8 `const arma::mat mlpack::nn::SparseAutoencoderFunction::InitializeWeights ()`

Initializes the parameters of the model to suitable values.

22.99.3.9 `void mlpack::nn::SparseAutoencoderFunction::Lambda (const double l)` `[inline]`

Sets the L2-regularization parameter.

Definition at line 114 of file sparse_autoencoder_function.hpp.

References `lambda`.

22.99.3.10 `double mlpack::nn::SparseAutoencoderFunction::Lambda () const` `[inline]`

Gets the L2-regularization parameter.

Definition at line 120 of file sparse_autoencoder_function.hpp.

References `lambda`.

22.99.3.11 `void mlpack::nn::SparseAutoencoderFunction::Rho (const double r)` `[inline]`

Sets the sparsity parameter.

Definition at line 138 of file sparse_autoencoder_function.hpp.

References `rho`.

22.99.3.12 `double mlpack::nn::SparseAutoencoderFunction::Rho () const` `[inline]`

Gets the sparsity parameter.

Definition at line 144 of file sparse_autoencoder_function.hpp.

References `rho`.

22.99.3.13 `void mlpack::nn::SparseAutoencoderFunction::Sigmoid (const arma::mat & x, arma::mat & output) const`
`[inline]`

Returns the elementwise sigmoid of the passed matrix, where the sigmoid function of a real number '*x*' is $[1 / (1 + \exp(-x))]$.

Parameters

<i>x</i>	Matrix of real values for which we require the sigmoid activation.
----------	--

Definition at line 81 of file sparse_autoencoder_function.hpp.

22.99.3.14 `void mlpack::nn::SparseAutoencoderFunction::VisibleSize (const size_t visible)` `[inline]`

Sets size of the visible layer.

Definition at line 90 of file sparse_autoencoder_function.hpp.

22.99.3.15 `size_t mlpack::nn::SparseAutoencoderFunction::VisibleSize () const` `[inline]`

Gets size of the visible layer.

Definition at line 96 of file sparse_autoencoder_function.hpp.

References `visibleSize`.

22.99.4 Member Data Documentation

22.99.4.1 `double mpack::nn::SparseAutoencoderFunction::beta` [private]

KL divergence parameter.

Definition at line 161 of file `sparse_autoencoder_function.hpp`.

Referenced by `Beta()`.

22.99.4.2 `const arma::mat& mpack::nn::SparseAutoencoderFunction::data` [private]

The matrix of data points.

Definition at line 151 of file `sparse_autoencoder_function.hpp`.

22.99.4.3 `size_t mpack::nn::SparseAutoencoderFunction::hiddenSize` [private]

Size of the hidden layer.

Definition at line 157 of file `sparse_autoencoder_function.hpp`.

Referenced by `HiddenSize()`.

22.99.4.4 `arma::mat mpack::nn::SparseAutoencoderFunction::initialPoint` [private]

Initial parameter vector.

Definition at line 153 of file `sparse_autoencoder_function.hpp`.

Referenced by `GetInitialPoint()`.

22.99.4.5 `double mpack::nn::SparseAutoencoderFunction::lambda` [private]

L2-regularization parameter.

Definition at line 159 of file `sparse_autoencoder_function.hpp`.

Referenced by `Lambda()`.

22.99.4.6 `double mpack::nn::SparseAutoencoderFunction::rho` [private]

Sparsity parameter.

Definition at line 163 of file `sparse_autoencoder_function.hpp`.

Referenced by `Rho()`.

22.99.4.7 `size_t mpack::nn::SparseAutoencoderFunction::visibleSize` [private]

Size of the visible layer.

Definition at line 155 of file `sparse_autoencoder_function.hpp`.

Referenced by `VisibleSize()`.

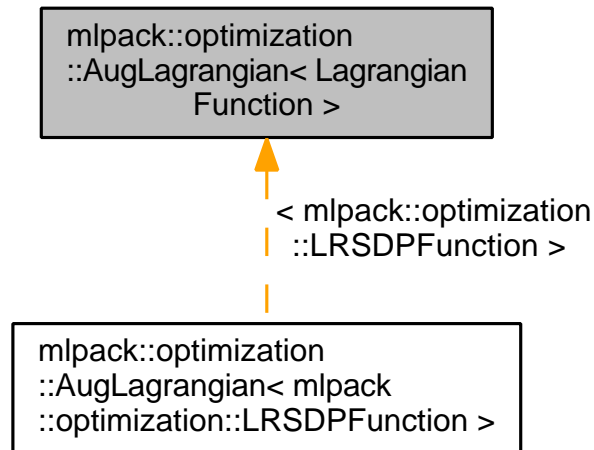
The documentation for this class was generated from the following file:

- src/mlpack/methods/sparse_autoencoder/sparse_autoencoder_function.hpp

22.100 mlpack::optimization::AugLagrangian< LagrangianFunction > Class Template Reference

The **AugLagrangian** (p. 441) class implements the Augmented Lagrangian method of optimization.

Inheritance diagram for mlpack::optimization::AugLagrangian< LagrangianFunction >:



Collaboration diagram for mlpack::optimization::AugLagrangian< LagrangianFunction >:



Public Types

- typedef **L_BFGS< AugLagrangianFunction< LagrangianFunction > > L_BFGSType**
Shorthand for the type of the L-BFGS optimizer we'll be using.

Public Member Functions

- **AugLagrangian** (LagrangianFunction &function)
Initialize the Augmented Lagrangian with the default L-BFGS optimizer.
- **AugLagrangian** (AugLagrangianFunction< LagrangianFunction > &augfunc, L_BFGSType &lbfgs)
Initialize the Augmented Lagrangian with a custom L-BFGS optimizer.
- const LagrangianFunction & **Function** () const
Get the LagrangianFunction.
- LagrangianFunction & **Function** ()
Modify the LagrangianFunction.

- `const arma::vec & Lambda () const`
Get the Lagrange multipliers.
- `arma::vec & Lambda ()`
Modify the Lagrange multipliers (i.e. set them before optimization).
- `const L_BFGSType & LBFGS () const`
Get the L-BFGS object used for the actual optimization.
- `L_BFGSType & LBFGS ()`
Modify the L-BFGS object used for the actual optimization.
- `bool Optimize (arma::mat &coordinates, const size_t maxIterations=1000)`
Optimize the function.
- `bool Optimize (arma::mat &coordinates, const arma::vec &initLambda, const double initSigma, const size_t ←
t maxIterations=1000)`
Optimize the function, giving initial estimates for the Lagrange multipliers.
- `double Sigma () const`
Get the penalty parameter.
- `double & Sigma ()`
Modify the penalty parameter.
- `std::string ToString () const`

Private Attributes

- `AugLagrangianFunction< LagrangianFunction > augfunc`
*Internally used **AugLagrangianFunction** (p. 446) which holds the function we are optimizing.*
- `LagrangianFunction & function`
Function to be optimized.
- `L_BFGSType & lbfgs`
The L-BFGS optimizer that we will use.
- `L_BFGSType lbfgsInternal`
*If the user did not pass an **L_BFGS** (p. 455) object, we'll use our own internal one.*

22.100.1 Detailed Description

`template<typename LagrangianFunction>class mpack::optimization::AugLagrangian< LagrangianFunction >`

The **AugLagrangian** (p. 441) class implements the Augmented Lagrangian method of optimization.

In this scheme, a penalty term is added to the Lagrangian. This method is also called the "method of multipliers".

The template class LagrangianFunction must implement the following five methods:

- `double Evaluate(const arma::mat& coordinates);`
- `void Gradient(const arma::mat& coordinates, arma::mat& gradient);`
- `size_t NumConstraints();`
- `double EvaluateConstraint(size_t index, const arma::mat& coordinates);`
- `double GradientConstraint(size_t index, const arma::mat& coordinates, arma::mat& gradient);`

The number of constraints must be greater than or equal to 0, and EvaluateConstraint() should evaluate the constraint at the given index for the given coordinates. Evaluate() should provide the objective function value for the given coordinates.

Template Parameters

<i>LagrangianFunction</i>	Function which can be optimized by this class.
---------------------------	--

Definition at line 51 of file aug_lagrangian.hpp.

22.100.2 Member Typedef Documentation

22.100.2.1 `template<typename LagrangianFunction> typedef L_BFGS<AugLagrangianFunction<LagrangianFunction> > mpack::optimization::AugLagrangian<LagrangianFunction>::L_BFGSType`

Shorthand for the type of the L-BFGS optimizer we'll be using.

Definition at line 56 of file aug_lagrangian.hpp.

22.100.3 Constructor & Destructor Documentation

22.100.3.1 `template<typename LagrangianFunction> mpack::optimization::AugLagrangian<LagrangianFunction>::AugLagrangian (LagrangianFunction & function)`

Initialize the Augmented Lagrangian with the default L-BFGS optimizer.

We limit the number of L-BFGS iterations to 1000, rather than the unlimited default L-BFGS.

Parameters

<i>function</i>	The function to be optimized.
-----------------	-------------------------------

22.100.3.2 `template<typename LagrangianFunction> mpack::optimization::AugLagrangian<LagrangianFunction>::AugLagrangian (AugLagrangianFunction<LagrangianFunction> & augfunc, L_BFGSType & lbfgs)`

Initialize the Augmented Lagrangian with a custom L-BFGS optimizer.

Parameters

<i>function</i>	The function to be optimized. This must be a pre-created utility AugLagrangianFunction (p. 446).
<i>lbfgs</i>	The custom L-BFGS optimizer to be used. This should have already been initialized with the given AugLagrangianFunction (p. 446).

22.100.4 Member Function Documentation

22.100.4.1 `template<typename LagrangianFunction> const LagrangianFunction& mpack::optimization::AugLagrangian<LagrangianFunction>::Function () const [inline]`

Get the LagrangianFunction.

Definition at line 109 of file aug_lagrangian.hpp.

22.100.4.2 `template<typename LagrangianFunction> LagrangianFunction& mpack::optimization::AugLagrangian<LagrangianFunction>::Function () [inline]`

Modify the LagrangianFunction.

Definition at line 111 of file aug_lagrangian.hpp.

```
22.100.4.3  template<typename LagrangianFunction> const arma::vec& mlpack::optimization::AugLagrangian<
            LagrangianFunction >::Lambda ( ) const  [inline]
```

Get the Lagrange multipliers.

Definition at line 119 of file aug_lagrangian.hpp.

```
22.100.4.4  template<typename LagrangianFunction> arma::vec& mlpack::optimization::AugLagrangian<
            LagrangianFunction >::Lambda ( )  [inline]
```

Modify the Lagrange multipliers (i.e. set them before optimization).

Definition at line 121 of file aug_lagrangian.hpp.

```
22.100.4.5  template<typename LagrangianFunction> const L_BFGSType& mlpack::optimization::AugLagrangian<
            LagrangianFunction >::LBFGS ( ) const  [inline]
```

Get the L-BFGS object used for the actual optimization.

Definition at line 114 of file aug_lagrangian.hpp.

```
22.100.4.6  template<typename LagrangianFunction> L_BFGSType& mlpack::optimization::AugLagrangian<
            LagrangianFunction >::LBFGS ( )  [inline]
```

Modify the L-BFGS object used for the actual optimization.

Definition at line 116 of file aug_lagrangian.hpp.

```
22.100.4.7  template<typename LagrangianFunction> bool mlpack::optimization::AugLagrangian< LagrangianFunction
            >::Optimize ( arma::mat & coordinates, const size_t maxIterations = 1000 )
```

Optimize the function.

The value '1' is used for the initial value of each Lagrange multiplier. To set the Lagrange multipliers yourself, use the other overload of **Optimize()** (p. 444).

Parameters

<i>coordinates</i>	Output matrix to store the optimized coordinates in.
<i>maxIterations</i>	Maximum number of iterations of the Augmented Lagrangian algorithm. 0 indicates no maximum.
<i>sigma</i>	Initial penalty parameter.

```
22.100.4.8  template<typename LagrangianFunction> bool mlpack::optimization::AugLagrangian< LagrangianFunction
            >::Optimize ( arma::mat & coordinates, const arma::vec & initLambda, const double initSigma, const size_t
            maxIterations = 1000 )
```

Optimize the function, giving initial estimates for the Lagrange multipliers.

The vector of Lagrange multipliers will be modified to contain the Lagrange multipliers of the final solution (if one is found).

Parameters

<i>coordinates</i>	Output matrix to store the optimized coordinates in.
<i>initLambda</i>	Vector of initial Lagrange multipliers. Should have length equal to the number of constraints.
<i>initSigma</i>	Initial penalty parameter.
<i>maxIterations</i>	Maximum number of iterations of the Augmented Lagrangian algorithm. 0 indicates no maximum.

22.100.4.9 `template<typename LagrangianFunction> double mpack::optimization::AugLagrangian<LagrangianFunction>::Sigma () const [inline]`

Get the penalty parameter.

Definition at line 124 of file `aug_lagrangian.hpp`.

22.100.4.10 `template<typename LagrangianFunction> double& mpack::optimization::AugLagrangian<LagrangianFunction>::Sigma () [inline]`

Modify the penalty parameter.

Definition at line 126 of file `aug_lagrangian.hpp`.

22.100.4.11 `template<typename LagrangianFunction> std::string mpack::optimization::AugLagrangian<LagrangianFunction>::ToString () const`

22.100.5 Member Data Documentation

22.100.5.1 `template<typename LagrangianFunction> AugLagrangianFunction<LagrangianFunction> mpack::optimization::AugLagrangian<LagrangianFunction>::augfunc [private]`

Internally used **AugLagrangianFunction** (p. 446) which holds the function we are optimizing.

This isn't publically accessible, but we provide ways to get to the Lagrange multipliers and the penalty parameter sigma.

Definition at line 138 of file `aug_lagrangian.hpp`.

Referenced by `mpack::optimization::AugLagrangian< mpack::optimization::LRSDPFunction >::Lambda()`, and `mpack::optimization::AugLagrangian< mpack::optimization::LRSDPFunction >::Sigma()`.

22.100.5.2 `template<typename LagrangianFunction> LagrangianFunction& mpack::optimization::AugLagrangian<LagrangianFunction>::function [private]`

Function to be optimized.

Definition at line 133 of file `aug_lagrangian.hpp`.

22.100.5.3 `template<typename LagrangianFunction> L_BFGSType& mpack::optimization::AugLagrangian<LagrangianFunction>::lbfgs [private]`

The L-BFGS optimizer that we will use.

Definition at line 144 of file `aug_lagrangian.hpp`.

Referenced by `mpack::optimization::AugLagrangian< mpack::optimization::LRSDPFunction >::LBFGS()`.

22.100.5.4 `template<typename LagrangianFunction> L_BFGSType mpack::optimization::AugLagrangian<LagrangianFunction >::lbfgsInternal [private]`

If the user did not pass an **L_BFGS** (p. 455) object, we'll use our own internal one.

Definition at line 141 of file `aug_lagrangian.hpp`.

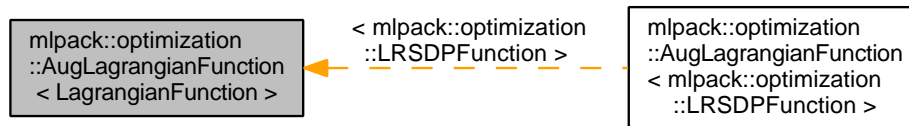
The documentation for this class was generated from the following file:

- `src/mlpack/core/optimizers/aug_lagrangian/aug_lagrangian.hpp`

22.101 mpack::optimization::AugLagrangianFunction< LagrangianFunction > Class Template Reference

This is a utility class used by **AugLagrangian** (p. 441), meant to wrap a `LagrangianFunction` into a function usable by a simple optimizer like L-BFGS.

Inheritance diagram for `mpack::optimization::AugLagrangianFunction< LagrangianFunction >`:



Public Member Functions

- **AugLagrangianFunction** (`LagrangianFunction &function`)
Initialize the **AugLagrangianFunction** (p. 446), but don't set the Lagrange multipliers or penalty parameters yet.
- **AugLagrangianFunction** (`LagrangianFunction &function`, `const arma::vec &lambda`, `const double sigma`)
Initialize the **AugLagrangianFunction** (p. 446) with the given `LagrangianFunction`, Lagrange multipliers, and initial penalty parameter.
- `double Evaluate` (`const arma::mat &coordinates`) `const`
Evaluate the objective function of the Augmented Lagrangian function, which is the standard Lagrangian function evaluation plus a penalty term, which penalizes unsatisfied constraints.
- `template<>`
`double Evaluate` (`const arma::mat &coordinates`) `const`
- `const LagrangianFunction &Function` () `const`
Get the Lagrangian function.
- `LagrangianFunction &Function` ()
Modify the Lagrangian function.
- `const arma::mat &GetInitialPoint` () `const`
Get the initial point of the optimization (supplied by the `LagrangianFunction`).
- `void Gradient` (`const arma::mat &coordinates`, `arma::mat &gradient`) `const`
Evaluate the gradient of the Augmented Lagrangian function.
- `template<>`
`void Gradient` (`const arma::mat &coordinates`, `arma::mat &gradient`) `const`
- `const arma::vec &Lambda` () `const`
Get the Lagrange multipliers.
- `arma::vec &Lambda` ()

Modify the Lagrange multipliers.

- double **Sigma** () const

Get sigma (the penalty parameter).

- double & **Sigma** ()

Modify sigma (the penalty parameter).

- std::string **ToString** () const

Private Attributes

- LagrangianFunction & **function**

Instantiation of the function to be optimized.

- arma::vec **lambda**

The Lagrange multipliers.

- double **sigma**

The penalty parameter.

22.101.1 Detailed Description

template<typename LagrangianFunction>class mpack::optimization::AugLagrangianFunction< LagrangianFunction >

This is a utility class used by **AugLagrangian** (p. 441), meant to wrap a LagrangianFunction into a function usable by a simple optimizer like L-BFGS.

Given a LagrangianFunction which follows the format outlined in the documentation for **AugLagrangian** (p. 441), this class provides **Evaluate()** (p. 448), **Gradient()** (p. 449), and **GetInitialPoint()** (p. 449) functions which allow this class to be used with a simple optimizer like L-BFGS.

This class can be specialized for your particular implementation – commonly, a faster method for computing the overall objective and gradient of the augmented Lagrangian function can be implemented than the naive, default implementation given. Use class template specialization and re-implement all of the methods (unfortunately, C++ specialization rules mean you have to re-implement everything).

Template Parameters

<i>LagrangianFunction</i>	Lagrangian function to be used.
---------------------------	---------------------------------

Definition at line 40 of file aug_lagrangian_function.hpp.

22.101.2 Constructor & Destructor Documentation

22.101.2.1 template<typename LagrangianFunction> mpack::optimization::AugLagrangianFunction< LagrangianFunction >::AugLagrangianFunction (LagrangianFunction & function)

Initialize the **AugLagrangianFunction** (p. 446), but don't set the Lagrange multipliers or penalty parameters yet.

Make sure you set the Lagrange multipliers before you use this...

Parameters

<i>function</i>	Lagrangian function.
-----------------	----------------------

22.101.2.2 `template<typename LagrangianFunction> mpack::optimization::AugLagrangianFunction<LagrangianFunction >::AugLagrangianFunction (LagrangianFunction & function, const arma::vec & lambda, const double sigma)`

Initialize the **AugLagrangianFunction** (p.446) with the given LagrangianFunction, Lagrange multipliers, and initial penalty parameter.

Parameters

<i>function</i>	Lagrangian function.
<i>lambda</i>	Initial Lagrange multipliers.
<i>sigma</i>	Initial penalty parameter.

22.101.3 Member Function Documentation

22.101.3.1 `template<typename LagrangianFunction> double mpack::optimization::AugLagrangianFunction<LagrangianFunction >::Evaluate (const arma::mat & coordinates) const`

Evaluate the objective function of the Augmented Lagrangian function, which is the standard Lagrangian function evaluation plus a penalty term, which penalizes unsatisfied constraints.

Parameters

<i>coordinates</i>	Coordinates to evaluate function at.
--------------------	--------------------------------------

Returns

Objective function.

22.101.3.2 `template<> double mpack::optimization::AugLagrangianFunction<LRSDPFunction >::Evaluate (const arma::mat & coordinates) const`

22.101.3.3 `template<typename LagrangianFunction> const LagrangianFunction& mpack::optimization::AugLagrangianFunction<LagrangianFunction >::Function () const [inline]`

Get the Lagrangian function.

Definition at line 100 of file `aug_lagrangian_function.hpp`.

22.101.3.4 `template<typename LagrangianFunction> LagrangianFunction& mpack::optimization::AugLagrangianFunction<LagrangianFunction >::Function () [inline]`

Modify the Lagrangian function.

Definition at line 102 of file `aug_lagrangian_function.hpp`.

22.101.3.5 `template<typename LagrangianFunction> const arma::mat& mpack::optimization::AugLagrangianFunction< LagrangianFunction >::GetInitialPoint () const`

Get the initial point of the optimization (supplied by the LagrangianFunction).

Returns

Initial point.

22.101.3.6 `template<typename LagrangianFunction> void mpack::optimization::AugLagrangianFunction< LagrangianFunction >::Gradient (const arma::mat & coordinates, arma::mat & gradient) const`

Evaluate the gradient of the Augmented Lagrangian function.

Parameters

<i>coordinates</i>	Coordinates to evaluate gradient at.
<i>gradient</i>	Matrix to store gradient into.

22.101.3.7 `template<> void mpack::optimization::AugLagrangianFunction< LRSDPFunction >::Gradient (const arma::mat & coordinates, arma::mat & gradient) const`

22.101.3.8 `template<typename LagrangianFunction> const arma::vec& mpack::optimization::AugLagrangianFunction< LagrangianFunction >::Lambda () const [inline]`

Get the Lagrange multipliers.

Definition at line 90 of file `aug_lagrangian_function.hpp`.

22.101.3.9 `template<typename LagrangianFunction> arma::vec& mpack::optimization::AugLagrangianFunction< LagrangianFunction >::Lambda () [inline]`

Modify the Lagrange multipliers.

Definition at line 92 of file `aug_lagrangian_function.hpp`.

22.101.3.10 `template<typename LagrangianFunction> double mpack::optimization::AugLagrangianFunction< LagrangianFunction >::Sigma () const [inline]`

Get sigma (the penalty parameter).

Definition at line 95 of file `aug_lagrangian_function.hpp`.

22.101.3.11 `template<typename LagrangianFunction> double& mpack::optimization::AugLagrangianFunction< LagrangianFunction >::Sigma () [inline]`

Modify sigma (the penalty parameter).

Definition at line 97 of file `aug_lagrangian_function.hpp`.

22.101.3.12 `template<typename LagrangianFunction> std::string mlpack::optimization::AugLagrangianFunction<LagrangianFunction >::ToString () const`

22.101.4 Member Data Documentation

22.101.4.1 `template<typename LagrangianFunction> LagrangianFunction& mlpack::optimization::AugLagrangianFunction<LagrangianFunction >::function [private]`

Instantiation of the function to be optimized.

Definition at line 109 of file `aug_lagrangian_function.hpp`.

22.101.4.2 `template<typename LagrangianFunction> arma::vec mlpack::optimization::AugLagrangianFunction<LagrangianFunction >::lambda [private]`

The Lagrange multipliers.

Definition at line 112 of file `aug_lagrangian_function.hpp`.

Referenced by `mlpack::optimization::AugLagrangianFunction< mlpack::optimization::LRSDPFunction >::Lambda()`.

22.101.4.3 `template<typename LagrangianFunction> double mlpack::optimization::AugLagrangianFunction<LagrangianFunction >::sigma [private]`

The penalty parameter.

Definition at line 114 of file `aug_lagrangian_function.hpp`.

Referenced by `mlpack::optimization::AugLagrangianFunction< mlpack::optimization::LRSDPFunction >::Sigma()`.

The documentation for this class was generated from the following file:

- `src/mlpack/core/optimizers/aug_lagrangian/aug_lagrangian_function.hpp`

22.102 mlpack::optimization::AugLagrangianTestFunction Class Reference

This function is taken from "Practical Mathematical Optimization" (Snyman), section 5.3.8 ("Application of the Augmented Lagrangian Method").

Public Member Functions

- **AugLagrangianTestFunction** ()
- **AugLagrangianTestFunction** (const arma::mat &initial_point)
- double **Evaluate** (const arma::mat &coordinates)
- double **EvaluateConstraint** (const size_t index, const arma::mat &coordinates)
- const arma::mat & **GetInitialPoint** () const
- void **Gradient** (const arma::mat &coordinates, arma::mat &gradient)
- void **GradientConstraint** (const size_t index, const arma::mat &coordinates, arma::mat &gradient)
- size_t **NumConstraints** () const
- std::string **ToString** () const

Private Attributes

- arma::mat **initialPoint**

22.102.1 Detailed Description

This function is taken from "Practical Mathematical Optimization" (Snyman), section 5.3.8 ("Application of the Augmented Lagrangian Method").

It has only one constraint.

The minimum that satisfies the constraint is $x = [1, 4]$, with an objective value of 70.

Definition at line 30 of file `aug_lagrangian_test_functions.hpp`.

22.102.2 Constructor & Destructor Documentation

22.102.2.1 `mpack::optimization::AugLagrangianTestFunction::AugLagrangianTestFunction ()`

22.102.2.2 `mpack::optimization::AugLagrangianTestFunction::AugLagrangianTestFunction (const arma::mat & initial_point)`

22.102.3 Member Function Documentation

22.102.3.1 `double mpack::optimization::AugLagrangianTestFunction::Evaluate (const arma::mat & coordinates)`

22.102.3.2 `double mpack::optimization::AugLagrangianTestFunction::EvaluateConstraint (const size_t index, const arma::mat & coordinates)`

22.102.3.3 `const arma::mat& mpack::optimization::AugLagrangianTestFunction::GetInitialPoint () const` `[inline]`

Definition at line 46 of file `aug_lagrangian_test_functions.hpp`.

References `initialPoint`.

22.102.3.4 `void mpack::optimization::AugLagrangianTestFunction::Gradient (const arma::mat & coordinates, arma::mat & gradient)`

22.102.3.5 `void mpack::optimization::AugLagrangianTestFunction::GradientConstraint (const size_t index, const arma::mat & coordinates, arma::mat & gradient)`

22.102.3.6 `size_t mpack::optimization::AugLagrangianTestFunction::NumConstraints () const` `[inline]`

Definition at line 39 of file `aug_lagrangian_test_functions.hpp`.

22.102.3.7 `std::string mpack::optimization::AugLagrangianTestFunction::ToString () const`

22.102.4 Member Data Documentation

22.102.4.1 `arma::mat mpack::optimization::AugLagrangianTestFunction::initialPoint` `[private]`

Definition at line 52 of file `aug_lagrangian_test_functions.hpp`.

Referenced by GetInitialPoint().

The documentation for this class was generated from the following file:

- src/mlpack/core/optimizers/aug_lagrangian/**aug_lagrangian_test_functions.hpp**

22.103 mlpack::optimization::ExponentialSchedule Class Reference

The exponential cooling schedule cools the temperature T at every step according to the equation.

Public Member Functions

- **ExponentialSchedule** (const double **lambda**=0.001)
- double **Lambda** () const
Get the cooling speed, lambda.
- double & **Lambda** ()
Modify the cooling speed, lambda.
- double **NextTemperature** (const double currentTemperature, const double)
Returns the next temperature given current status.

Private Attributes

- double **lambda**
The cooling speed.

22.103.1 Detailed Description

The exponential cooling schedule cools the temperature T at every step according to the equation.

$$T_{n+1} = (1 - \lambda)T_n$$

where $0 < \lambda < 1$ is the cooling speed. The smaller λ is, the slower the cooling speed, and better the final result will be. Some literature uses $\alpha = (-1\lambda)$ instead. In practice, α is very close to 1 and will be awkward to input (e.g. alpha = 0.999999 vs lambda = 1e-6).

Definition at line 34 of file exponential_schedule.hpp.

22.103.2 Constructor & Destructor Documentation

22.103.2.1 mlpack::optimization::ExponentialSchedule::ExponentialSchedule (const double *lambda* = 0.001) [inline]

Definition at line 42 of file exponential_schedule.hpp.

22.103.3 Member Function Documentation

22.103.3.1 `double mlpack::optimization::ExponentialSchedule::Lambda () const` `[inline]`

Get the cooling speed, lambda.

Definition at line 59 of file `exponential_schedule.hpp`.

References `lambda`.

22.103.3.2 `double& mlpack::optimization::ExponentialSchedule::Lambda ()` `[inline]`

Modify the cooling speed, lambda.

Definition at line 61 of file `exponential_schedule.hpp`.

References `lambda`.

22.103.3.3 `double mlpack::optimization::ExponentialSchedule::NextTemperature (const double currentTemperature, const double)` `[inline]`

Returns the next temperature given current status.

The current system's energy is not used in this calculation.

Parameters

<i>currentTemperature</i>	Current temperature of system.
<i>currentEnergy</i>	Current energy of system (not used).

Definition at line 51 of file `exponential_schedule.hpp`.

References `lambda`.

22.103.4 Member Data Documentation

22.103.4.1 `double mlpack::optimization::ExponentialSchedule::lambda` `[private]`

The cooling speed.

Definition at line 65 of file `exponential_schedule.hpp`.

Referenced by `Lambda()`, and `NextTemperature()`.

The documentation for this class was generated from the following file:

- `src/mlpack/core/optimizers/sa/exponential_schedule.hpp`

22.104 mlpack::optimization::GockenbachFunction Class Reference

This function is taken from M.

Public Member Functions

- **GockenbachFunction** ()
- **GockenbachFunction** (const arma::mat &initial_point)
- double **Evaluate** (const arma::mat &coordinates)
- double **EvaluateConstraint** (const size_t index, const arma::mat &coordinates)
- const arma::mat & **GetInitialPoint** () const
- void **Gradient** (const arma::mat &coordinates, arma::mat &gradient)
- void **GradientConstraint** (const size_t index, const arma::mat &coordinates, arma::mat &gradient)
- size_t **NumConstraints** () const

Private Attributes

- arma::mat **initialPoint**

22.104.1 Detailed Description

This function is taken from M.

Gockenbach's lectures on general nonlinear programs, found at: <http://www.math.mtu.edu/~msgocken/ma5630spring20pdf>

The program we are using is example 2.5 from this document. I have arbitrarily decided that this will be called the Gockenbach function.

The minimum that satisfies the two constraints is given as $x = [0.12288, -1.1078, 0.015100]$, with an objective value of about 29.634.

Definition at line 66 of file `aug_lagrangian_test_functions.hpp`.

22.104.2 Constructor & Destructor Documentation

22.104.2.1 `mlpack::optimization::GockenbachFunction::GockenbachFunction ()`

22.104.2.2 `mlpack::optimization::GockenbachFunction::GockenbachFunction (const arma::mat & initial_point)`

22.104.3 Member Function Documentation

22.104.3.1 `double mlpack::optimization::GockenbachFunction::Evaluate (const arma::mat & coordinates)`

22.104.3.2 `double mlpack::optimization::GockenbachFunction::EvaluateConstraint (const size_t index, const arma::mat & coordinates)`

22.104.3.3 `const arma::mat& mlpack::optimization::GockenbachFunction::GetInitialPoint () const` `[inline]`

Definition at line 82 of file `aug_lagrangian_test_functions.hpp`.

References `initialPoint`.

22.104.3.4 `void mlpack::optimization::GockenbachFunction::Gradient (const arma::mat & coordinates, arma::mat & gradient)`

22.104.3.5 void mlpack::optimization::GockenbachFunction::GradientConstraint (const size_t *index*, const arma::mat & *coordinates*, arma::mat & *gradient*)

22.104.3.6 size_t mlpack::optimization::GockenbachFunction::NumConstraints () const [inline]

Definition at line 75 of file aug_lagrangian_test_functions.hpp.

22.104.4 Member Data Documentation

22.104.4.1 arma::mat mlpack::optimization::GockenbachFunction::initialPoint [private]

Definition at line 85 of file aug_lagrangian_test_functions.hpp.

Referenced by GetInitialPoint().

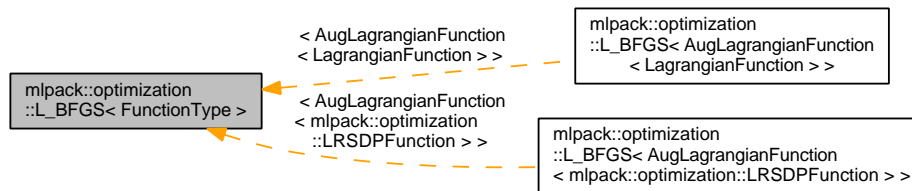
The documentation for this class was generated from the following file:

- src/mlpack/core/optimizers/aug_lagrangian/aug_lagrangian_test_functions.hpp

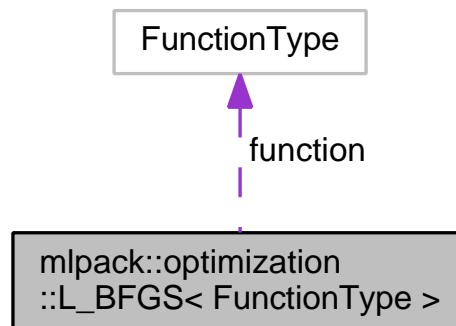
22.105 mlpack::optimization::L_BFGS< FunctionType > Class Template Reference

The generic L-BFGS optimizer, which uses a back-tracking line search algorithm to minimize a function.

Inheritance diagram for mlpack::optimization::L_BFGS< FunctionType >:



Collaboration diagram for mlpack::optimization::L_BFGS< FunctionType >:



Public Member Functions

- **L_BFGS** (FunctionType &**function**, const size_t **numBasis**=5, const size_t **maxIterations**=0, const double

armijoConstant=1e-4, const double wolfe=0.9, const double minGradientNorm=1e-10, const size_t maxLine↵
SearchTrials=50, const double minStep=1e-20, const double maxStep=1e20)

Initialize the L-BFGS object.

- double **ArmijoConstant** () const

Get the Armijo condition constant.

- double & **ArmijoConstant** ()

Modify the Armijo condition constant.

- const FunctionType & **Function** () const

Return the function that is being optimized.

- FunctionType & **Function** ()

Modify the function that is being optimized.

- size_t **MaxIterations** () const

Get the maximum number of iterations.

- size_t & **MaxIterations** ()

Modify the maximum number of iterations.

- size_t **MaxLineSearchTrials** () const

Get the maximum number of line search trials.

- size_t & **MaxLineSearchTrials** ()

Modify the maximum number of line search trials.

- double **MaxStep** () const

Return the maximum line search step size.

- double & **MaxStep** ()

Modify the maximum line search step size.

- double **MinGradientNorm** () const

Get the minimum gradient norm.

- double & **MinGradientNorm** ()

Modify the minimum gradient norm.

- const std::pair< arma::mat, double > & **MinPointIterate** () const

Return the point where the lowest function value has been found.

- double **MinStep** () const

Return the minimum line search step size.

- double & **MinStep** ()

Modify the minimum line search step size.

- size_t **NumBasis** () const

Get the memory size.

- size_t & **NumBasis** ()

Modify the memory size.

- double **Optimize** (arma::mat &iterate)

Use L-BFGS to optimize the given function, starting at the given iterate point and finding the minimum.

- double **Optimize** (arma::mat &iterate, const size_t **maxIterations**)

Use L-BFGS to optimize (minimize) the given function, starting at the given iterate point, and performing no more than the given maximum number of iterations (the class variable maxIterations is ignored for this run, but not modified).

- std::string **ToString** () const

- double **Wolfe** () const

Get the Wolfe parameter.

- double & **Wolfe** ()

Modify the Wolfe parameter.

Private Member Functions

- double **ChooseScalingFactor** (const size_t iterationNum, const arma::mat &gradient)
Calculate the scaling factor, gamma, which is used to scale the Hessian approximation matrix.
- double **Evaluate** (const arma::mat &iterate)
Evaluate the function at the given iterate point and store the result if it is a new minimum.
- bool **GradientNormTooSmall** (const arma::mat &gradient)
Check to make sure that the norm of the gradient is not smaller than 1e-5.
- bool **LineSearch** (double &functionValue, arma::mat &iterate, arma::mat &gradient, const arma::mat &search←
Direction)
Perform a back-tracking line search along the search direction to calculate a step size satisfying the Wolfe conditions.
- void **SearchDirection** (const arma::mat &gradient, const size_t iterationNum, const double scalingFactor, arma←
::mat &searchDirection)
Find the L-BFGS search direction.
- void **UpdateBasisSet** (const size_t iterationNum, const arma::mat &iterate, const arma::mat &oldIterate, const
arma::mat &gradient, const arma::mat &oldGradient)
*Update the y and s matrices, which store the differences between the iterate and old iterate and the differences between
the gradient and the old gradient, respectively.*

Private Attributes

- double **armijoConstant**
Parameter for determining the Armijo condition.
- FunctionType & **function**
Internal reference to the function we are optimizing.
- size_t **maxIterations**
Maximum number of iterations.
- size_t **maxLineSearchTrials**
Maximum number of trials for the line search.
- double **maxStep**
Maximum step of the line search.
- double **minGradientNorm**
Minimum gradient norm required to continue the optimization.
- std::pair< arma::mat, double > **minPointIterate**
Best point found so far.
- double **minStep**
Minimum step of the line search.
- arma::mat **newIterateTmp**
Position of the new iterate.
- size_t **numBasis**
Size of memory for this L-BFGS optimizer.
- arma::cube **s**
Stores all the s matrices in memory.
- double **wolfe**
Parameter for detecting the Wolfe condition.
- arma::cube **y**
Stores all the y matrices in memory.

22.105.1 Detailed Description

```
template<typename FunctionType> class mpack::optimization::L_BFGS< FunctionType >
```

The generic L-BFGS optimizer, which uses a back-tracking line search algorithm to minimize a function.

The parameters for the algorithm (number of memory points, maximum step size, and so forth) are all configurable via either the constructor or standalone modifier functions. A function which can be optimized by this class must implement the following methods:

- a default constructor
- double **Evaluate**(const arma::mat& coordinates) (p. 459);
- void Gradient(const arma::mat& coordinates, arma::mat& gradient);
- arma::mat& GetInitialPoint();

Definition at line 36 of file lbfgs.hpp.

22.105.2 Constructor & Destructor Documentation

22.105.2.1 `template<typename FunctionType> mpack::optimization::L_BFGS< FunctionType >::L_BFGS (FunctionType & function, const size_t numBasis = 5, const size_t maxIterations = 0, const double armijoConstant = 1e-4, const double wolfe = 0.9, const double minGradientNorm = 1e-10, const size_t maxLineSearchTrials = 50, const double minStep = 1e-20, const double maxStep = 1e20)`

Initialize the L-BFGS object.

Store a reference to the function we will be optimizing and set the size of the memory for the algorithm. There are many parameters that can be set for the optimization, but default values are given for each of them.

Parameters

<i>function</i>	Instance of function to be optimized.
<i>numBasis</i>	Number of memory points to be stored (default 5).
<i>maxIterations</i>	Maximum number of iterations for the optimization (default 0 – may run indefinitely).
<i>armijoConstant</i>	Controls the accuracy of the line search routine for determining the Armijo condition.
<i>wolfe</i>	Parameter for detecting the Wolfe condition.
<i>minGradient↔ Norm</i>	Minimum gradient norm required to continue the optimization.
<i>maxLine↔ SearchTrials</i>	The maximum number of trials for the line search (before giving up).
<i>minStep</i>	The minimum step of the line search.
<i>maxStep</i>	The maximum step of the line search.

22.105.3 Member Function Documentation

22.105.3.1 `template<typename FunctionType> double mpack::optimization::L_BFGS< FunctionType >::ArmijoConstant () const [inline]`

Get the Armijo condition constant.

Definition at line 120 of file lbfgs.hpp.

22.105.3.2 `template<typename FunctionType> double& mlpack::optimization::L_BFGS< FunctionType >::ArmijoConstant () [inline]`

Modify the Armijo condition constant.

Definition at line 122 of file lbfgs.hpp.

22.105.3.3 `template<typename FunctionType> double mlpack::optimization::L_BFGS< FunctionType >::ChooseScalingFactor (const size_t iterationNum, const arma::mat & gradient) [private]`

Calculate the scaling factor, gamma, which is used to scale the Hessian approximation matrix.

See method M3 in Section 4 of Liu and Nocedal (1989).

Returns

The calculated scaling factor.

22.105.3.4 `template<typename FunctionType> double mlpack::optimization::L_BFGS< FunctionType >::Evaluate (const arma::mat & iterate) [private]`

Evaluate the function at the given iterate point and store the result if it is a new minimum.

Returns

The value of the function.

22.105.3.5 `template<typename FunctionType> const FunctionType& mlpack::optimization::L_BFGS< FunctionType >::Function () const [inline]`

Return the function that is being optimized.

Definition at line 105 of file lbfgs.hpp.

22.105.3.6 `template<typename FunctionType> FunctionType& mlpack::optimization::L_BFGS< FunctionType >::Function () [inline]`

Modify the function that is being optimized.

Definition at line 107 of file lbfgs.hpp.

22.105.3.7 `template<typename FunctionType> bool mlpack::optimization::L_BFGS< FunctionType >::GradientNormTooSmall (const arma::mat & gradient) [private]`

Check to make sure that the norm of the gradient is not smaller than 1e-5.

Currently that value is not configurable.

Returns

(norm < minGradientNorm).

22.105.3.8 `template<typename FunctionType> bool mlpack::optimization::L_BFGS< FunctionType >::LineSearch (double & functionValue, arma::mat & iterate, arma::mat & gradient, const arma::mat & searchDirection) [private]`

Perform a back-tracking line search along the search direction to calculate a step size satisfying the Wolfe conditions.

The parameter `iterate` will be modified if the method is successful.

Parameters

<i>functionValue</i>	Value of the function at the initial point
<i>iterate</i>	The initial point to begin the line search from
<i>gradient</i>	The gradient at the initial point
<i>searchDirection</i>	A vector specifying the search direction
<i>stepSize</i>	Variable the calculated step size will be stored in

Returns

false if no step size is suitable, true otherwise.

22.105.3.9 `template<typename FunctionType> size_t mlpack::optimization::L_BFGS< FunctionType >::MaxIterations () const [inline]`

Get the maximum number of iterations.

Definition at line 115 of file `lbfgs.hpp`.

22.105.3.10 `template<typename FunctionType> size_t& mlpack::optimization::L_BFGS< FunctionType >::MaxIterations () [inline]`

Modify the maximum number of iterations.

Definition at line 117 of file `lbfgs.hpp`.

22.105.3.11 `template<typename FunctionType> size_t mlpack::optimization::L_BFGS< FunctionType >::MaxLineSearchTrials () const [inline]`

Get the maximum number of line search trials.

Definition at line 135 of file `lbfgs.hpp`.

22.105.3.12 `template<typename FunctionType> size_t& mlpack::optimization::L_BFGS< FunctionType >::MaxLineSearchTrials () [inline]`

Modify the maximum number of line search trials.

Definition at line 137 of file `lbfgs.hpp`.

22.105.3.13 `template<typename FunctionType> double mlpack::optimization::L_BFGS< FunctionType >::MaxStep () const [inline]`

Return the maximum line search step size.

Definition at line 145 of file `lbfgs.hpp`.

22.105.3.14 `template<typename FunctionType> double& mpack::optimization::L_BFGS< FunctionType >::MaxStep ()`
`[inline]`

Modify the maximum line search step size.

Definition at line 147 of file lbfgs.hpp.

22.105.3.15 `template<typename FunctionType> double mpack::optimization::L_BFGS< FunctionType >::MinGradientNorm`
`() const [inline]`

Get the minimum gradient norm.

Definition at line 130 of file lbfgs.hpp.

22.105.3.16 `template<typename FunctionType> double& mpack::optimization::L_BFGS< FunctionType`
`>::MinGradientNorm () [inline]`

Modify the minimum gradient norm.

Definition at line 132 of file lbfgs.hpp.

22.105.3.17 `template<typename FunctionType> const std::pair<arma::mat, double>& mpack::optimization::L_BFGS<`
`FunctionType >::MinPointIterate () const`

Return the point where the lowest function value has been found.

Returns

arma::vec representing the point and a double with the function value at that point.

22.105.3.18 `template<typename FunctionType> double mpack::optimization::L_BFGS< FunctionType >::MinStep ()`
`const [inline]`

Return the minimum line search step size.

Definition at line 140 of file lbfgs.hpp.

22.105.3.19 `template<typename FunctionType> double& mpack::optimization::L_BFGS< FunctionType >::MinStep ()`
`[inline]`

Modify the minimum line search step size.

Definition at line 142 of file lbfgs.hpp.

22.105.3.20 `template<typename FunctionType> size_t mpack::optimization::L_BFGS< FunctionType >::NumBasis ()`
`const [inline]`

Get the memory size.

Definition at line 110 of file lbfgs.hpp.

22.105.3.21 `template<typename FunctionType> size_t& mlpack::optimization::L_BFGS<FunctionType>::NumBasis ()`
`[inline]`

Modify the memory size.

Definition at line 112 of file lbfgs.hpp.

22.105.3.22 `template<typename FunctionType> double mlpack::optimization::L_BFGS<FunctionType>::Optimize (arma::mat & iterate)`

Use L-BFGS to optimize the given function, starting at the given iterate point and finding the minimum.

The maximum number of iterations is set in the constructor (or with **MaxIterations()** (p.460)). Alternately, another overload is provided which takes a maximum number of iterations as a parameter. The given starting point will be modified to store the finishing point of the algorithm, and the final objective value is returned.

Parameters

<i>iterate</i>	Starting point (will be modified).
----------------	------------------------------------

Returns

Objective value of the final point.

22.105.3.23 `template<typename FunctionType> double mlpack::optimization::L_BFGS<FunctionType>::Optimize (arma::mat & iterate, const size_t maxIterations)`

Use L-BFGS to optimize (minimize) the given function, starting at the given iterate point, and performing no more than the given maximum number of iterations (the class variable maxIterations is ignored for this run, but not modified).

The given starting point will be modified to store the finishing point of the algorithm, and the final objective value is returned.

Parameters

<i>iterate</i>	Starting point (will be modified).
<i>maxIterations</i>	Maximum number of iterations (0 specifies no limit).

Returns

Objective value of the final point.

22.105.3.24 `template<typename FunctionType> void mlpack::optimization::L_BFGS<FunctionType>::SearchDirection (const arma::mat & gradient, const size_t iterationNum, const double scalingFactor, arma::mat & searchDirection)`
`[private]`

Find the L-BFGS search direction.

Parameters

<i>gradient</i>	The gradient at the current point
<i>iteration_num</i>	The iteration number
<i>scaling_factor</i>	Scaling factor to use (see ChooseScalingFactor_())
<i>search_direction</i>	Vector to store search direction in

22.105.3.25 `template<typename FunctionType> std::string mlpack::optimization::L_BFGS< FunctionType >::ToString () const`

22.105.3.26 `template<typename FunctionType> void mlpack::optimization::L_BFGS< FunctionType >::UpdateBasisSet (const size_t iterationNum, const arma::mat & iterate, const arma::mat & oldIterate, const arma::mat & gradient, const arma::mat & oldGradient) [private]`

Update the y and s matrices, which store the differences between the iterate and old iterate and the differences between the gradient and the old gradient, respectively.

Parameters

<i>iterationNum</i>	Iteration number
<i>iterate</i>	Current point
<i>oldIterate</i>	Point at last iteration
<i>gradient</i>	Gradient at current point (iterate)
<i>oldGradient</i>	Gradient at last iteration point (oldIterate)

22.105.3.27 `template<typename FunctionType> double mlpack::optimization::L_BFGS< FunctionType >::Wolfe () const [inline]`

Get the Wolfe parameter.

Definition at line 125 of file lbfgs.hpp.

22.105.3.28 `template<typename FunctionType> double& mlpack::optimization::L_BFGS< FunctionType >::Wolfe () [inline]`

Modify the Wolfe parameter.

Definition at line 127 of file lbfgs.hpp.

22.105.4 Member Data Documentation

22.105.4.1 `template<typename FunctionType> double mlpack::optimization::L_BFGS< FunctionType >::armijoConstant [private]`

Parameter for determining the Armijo condition.

Definition at line 168 of file lbfgs.hpp.

Referenced by `mlpack::optimization::L_BFGS< AugLagrangianFunction< mlpack::optimization::LRSDPFFunction >::ArmijoConstant()`.

22.105.4.2 `template<typename FunctionType> FunctionType& mlpack::optimization::L_BFGS<FunctionType>::function`
`[private]`

Internal reference to the function we are optimizing.

Definition at line 154 of file lbfgs.hpp.

22.105.4.3 `template<typename FunctionType> size_t mlpack::optimization::L_BFGS<FunctionType>::maxIterations`
`[private]`

Maximum number of iterations.

Definition at line 166 of file lbfgs.hpp.

Referenced by `mlpack::optimization::L_BFGS< AugLagrangianFunction< mlpack::optimization::LRSDPFunction > >::MaxIterations()`.

22.105.4.4 `template<typename FunctionType> size_t mlpack::optimization::L_BFGS<FunctionType>::maxLineSearchTrials`
`[private]`

Maximum number of trials for the line search.

Definition at line 174 of file lbfgs.hpp.

Referenced by `mlpack::optimization::L_BFGS< AugLagrangianFunction< mlpack::optimization::LRSDPFunction > >::MaxLineSearchTrials()`.

22.105.4.5 `template<typename FunctionType> double mlpack::optimization::L_BFGS<FunctionType>::maxStep`
`[private]`

Maximum step of the line search.

Definition at line 178 of file lbfgs.hpp.

Referenced by `mlpack::optimization::L_BFGS< AugLagrangianFunction< mlpack::optimization::LRSDPFunction > >::MaxStep()`.

22.105.4.6 `template<typename FunctionType> double mlpack::optimization::L_BFGS<FunctionType>::minGradientNorm`
`[private]`

Minimum gradient norm required to continue the optimization.

Definition at line 172 of file lbfgs.hpp.

Referenced by `mlpack::optimization::L_BFGS< AugLagrangianFunction< mlpack::optimization::LRSDPFunction > >::MinGradientNorm()`.

22.105.4.7 `template<typename FunctionType> std::pair<arma::mat, double> mlpack::optimization::L_BFGS<FunctionType>::minPointIterate`
`[private]`

Best point found so far.

Definition at line 181 of file lbfgs.hpp.

22.105.4.8 `template<typename FunctionType> double mlpack::optimization::L_BFGS< FunctionType >::minStep`
`[private]`

Minimum step of the line search.

Definition at line 176 of file lbfgs.hpp.

Referenced by `mlpack::optimization::L_BFGS< AugLagrangianFunction< mlpack::optimization::LRSDPFunction >>::MinStep()`.

22.105.4.9 `template<typename FunctionType> arma::mat mlpack::optimization::L_BFGS< FunctionType >::newIterateTmp`
`[private]`

Position of the new iterate.

Definition at line 157 of file lbfgs.hpp.

22.105.4.10 `template<typename FunctionType> size_t mlpack::optimization::L_BFGS< FunctionType >::numBasis`
`[private]`

Size of memory for this L-BFGS optimizer.

Definition at line 164 of file lbfgs.hpp.

Referenced by `mlpack::optimization::L_BFGS< AugLagrangianFunction< mlpack::optimization::LRSDPFunction >>::NumBasis()`.

22.105.4.11 `template<typename FunctionType> arma::cube mlpack::optimization::L_BFGS< FunctionType >::s`
`[private]`

Stores all the s matrices in memory.

Definition at line 159 of file lbfgs.hpp.

22.105.4.12 `template<typename FunctionType> double mlpack::optimization::L_BFGS< FunctionType >::wolfe`
`[private]`

Parameter for detecting the Wolfe condition.

Definition at line 170 of file lbfgs.hpp.

Referenced by `mlpack::optimization::L_BFGS< AugLagrangianFunction< mlpack::optimization::LRSDPFunction >>::Wolfe()`.

22.105.4.13 `template<typename FunctionType> arma::cube mlpack::optimization::L_BFGS< FunctionType >::y`
`[private]`

Stores all the y matrices in memory.

Definition at line 161 of file lbfgs.hpp.

The documentation for this class was generated from the following file:

- `src/mlpack/core/optimizers/lbfgs/lbfgs.hpp`

22.106 mpack::optimization::LovaszThetaSDP Class Reference

This function is the Lovasz-Theta semidefinite program, as implemented in the following paper:

Public Member Functions

- **LovaszThetaSDP** ()
- **LovaszThetaSDP** (const arma::mat &edges)
 - Initialize the Lovasz-Theta SDP with the given set of edges.*
- const arma::mat & **Edges** () const
- arma::mat & **Edges** ()
- double **Evaluate** (const arma::mat &coordinates)
- double **EvaluateConstraint** (const size_t index, const arma::mat &coordinates)
- const arma::mat & **GetInitialPoint** ()
- void **Gradient** (const arma::mat &coordinates, arma::mat &gradient)
- void **GradientConstraint** (const size_t index, const arma::mat &coordinates, arma::mat &gradient)
- size_t **NumConstraints** () const

Private Attributes

- arma::mat **edges**
- arma::mat **initialPoint**
- size_t **vertices**

22.106.1 Detailed Description

This function is the Lovasz-Theta semidefinite program, as implemented in the following paper:

S. Burer, R. Monteiro "A nonlinear programming algorithm for solving semidefinite programs via low-rank factorization." Journal of Mathematical Programming, 2004

Given a simple, undirected graph $G = (V, E)$, the Lovasz-Theta SDP is defined by:

$$\min_X \{ \text{Tr}(-(\mathbf{e} \mathbf{e}^T)^T X) : \text{Tr}(X) = 1, X_{ij} = 0 \text{ for all } (i, j) \text{ in } E, X \succeq 0 \}$$

where \mathbf{e} is the vector of all ones and X has dimension $|V| \times |V|$.

In the Monteiro-Burer formulation, we take $X = R * R^T$, where R is the coordinates given to the **Evaluate()** (p. 467), **Gradient()** (p. 467), **EvaluateConstraint()** (p. 467), and **GradientConstraint()** (p. 467) functions.

Definition at line 110 of file aug_lagrangian_test_functions.hpp.

22.106.2 Constructor & Destructor Documentation

22.106.2.1 mpack::optimization::LovaszThetaSDP::LovaszThetaSDP ()

22.106.2.2 mpack::optimization::LovaszThetaSDP::LovaszThetaSDP (const arma::mat & edges)

Initialize the Lovasz-Theta SDP with the given set of edges.

The edge matrix should consist of rows of two dimensions, where dimension 0 is the first vertex of the edge and dimension 1 is the second edge (or vice versa, as it doesn't make a difference).

Parameters

<i>edges</i>	Matrix of edges.
--------------	------------------

22.106.3 Member Function Documentation

22.106.3.1 `const arma::mat& mlpack::optimization::LovaszThetaSDP::Edges () const` `[inline]`

Definition at line 137 of file `aug_lagrangian_test_functions.hpp`.

References `edges`.

22.106.3.2 `arma::mat& mlpack::optimization::LovaszThetaSDP::Edges ()` `[inline]`

Definition at line 138 of file `aug_lagrangian_test_functions.hpp`.

References `edges`.

22.106.3.3 `double mlpack::optimization::LovaszThetaSDP::Evaluate (const arma::mat & coordinates)`

22.106.3.4 `double mlpack::optimization::LovaszThetaSDP::EvaluateConstraint (const size_t index, const arma::mat & coordinates)`

22.106.3.5 `const arma::mat& mlpack::optimization::LovaszThetaSDP::GetInitialPoint ()`

22.106.3.6 `void mlpack::optimization::LovaszThetaSDP::Gradient (const arma::mat & coordinates, arma::mat & gradient)`

22.106.3.7 `void mlpack::optimization::LovaszThetaSDP::GradientConstraint (const size_t index, const arma::mat & coordinates, arma::mat & gradient)`

22.106.3.8 `size_t mlpack::optimization::LovaszThetaSDP::NumConstraints () const`

22.106.4 Member Data Documentation

22.106.4.1 `arma::mat mlpack::optimization::LovaszThetaSDP::edges` `[private]`

Definition at line 141 of file `aug_lagrangian_test_functions.hpp`.

Referenced by `Edges()`.

22.106.4.2 `arma::mat mlpack::optimization::LovaszThetaSDP::initialPoint` `[private]`

Definition at line 144 of file `aug_lagrangian_test_functions.hpp`.

22.106.4.3 `size_t mlpack::optimization::LovaszThetaSDP::vertices` `[private]`

Definition at line 142 of file `aug_lagrangian_test_functions.hpp`.

The documentation for this class was generated from the following file:

- `src/mlpack/core/optimizers/aug_lagrangian/aug_lagrangian_test_functions.hpp`

Private Attributes

- **AugLagrangian**< LRSDPFunction > **augLag**

The *AugLagrangian* (p. 441) object which will be used for optimization.

- **LRSDPFunction** function

Function to optimize, which the *AugLagrangian* (p. 441) object holds.

22.107.1 Detailed Description

LRSDP (p. 468) is the implementation of Monteiro and Burer's formulation of low-rank semidefinite programs (LR-SDP).

This solver uses the augmented Lagrangian optimizer to solve low-rank semidefinite programs.

Definition at line 31 of file lrsdp.hpp.

22.107.2 Constructor & Destructor Documentation

22.107.2.1 mpack::optimization::LRSDP::LRSDP (const size_t numConstraints, const arma::mat & initialPoint)

Create an **LRSDP** (p. 468) to be optimized.

The solution will end up being a matrix of size (rank) x (rows). To construct each constraint and the objective function, use the functions **A()** (p. 470), **B()** (p. 470), and **C()** (p. 471) to set them correctly.

Parameters

<i>numConstraints</i>	Number of constraints in the problem.
<i>rank</i>	Rank of the solution (<= rows).
<i>rows</i>	Number of rows in the solution.

22.107.2.2 mpack::optimization::LRSDP::LRSDP (const size_t numConstraints, const arma::mat & initialPoint, AugLagrangian< LRSDPFunction > & augLagrangian)

Create an **LRSDP** (p. 468) to be optimized, passing in an already-created **AugLagrangian** (p. 441) object.

The given initial point should be set to the size (rows) x (rank), where (rank) is the reduced rank of the problem.

Parameters

<i>numConstraints</i>	Number of constraints in the problem.
<i>initialPoint</i>	Initial point of the optimization.
<i>auglag</i>	Pre-initialized AugLagrangian<LRSDP> object.

22.107.3 Member Function Documentation

22.107.3.1 const std::vector<arma::mat>& mpack::optimization::LRSDP::A () const [inline]

Return the vector of A matrices (which correspond to the constraints).

Definition at line 73 of file lrsdp.hpp.

22.107.3.2 `std::vector<arma::mat>& mlpack::optimization::LRSDP::A () [inline]`

Modify the vector of A matrices (which correspond to the constraints).

Definition at line 75 of file lrsdp.hpp.

22.107.3.3 `const arma::uvec& mlpack::optimization::LRSDP::AModes () const [inline]`

Return the vector of modes for the A matrices.

Definition at line 78 of file lrsdp.hpp.

22.107.3.4 `arma::uvec& mlpack::optimization::LRSDP::AModes () [inline]`

Modify the vector of modes for the A matrices.

Definition at line 80 of file lrsdp.hpp.

22.107.3.5 `const AugLagrangian<LRSDPFunction>& mlpack::optimization::LRSDP::AugLag () const [inline]`

Return the augmented Lagrangian object.

Definition at line 93 of file lrsdp.hpp.

References `augLag`.

22.107.3.6 `AugLagrangian<LRSDPFunction>& mlpack::optimization::LRSDP::AugLag () [inline]`

Modify the augmented Lagrangian object.

Definition at line 95 of file lrsdp.hpp.

References `augLag`.

22.107.3.7 `const arma::vec& mlpack::optimization::LRSDP::B () const [inline]`

Return the vector of B values.

Definition at line 83 of file lrsdp.hpp.

22.107.3.8 `arma::vec& mlpack::optimization::LRSDP::B () [inline]`

Modify the vector of B values.

Definition at line 85 of file lrsdp.hpp.

22.107.3.9 `const arma::mat& mlpack::optimization::LRSDP::C () const [inline]`

Return the objective function matrix (C).

Definition at line 68 of file lrsdp.hpp.

22.107.3.10 `arma::mat& mlpack::optimization::LRSDP::C () [inline]`

Modify the objective function matrix (C).

Definition at line 70 of file `lrstdp.hpp`.

22.107.3.11 `const LRSDPFunction& mlpack::optimization::LRSDP::Function () const [inline]`

Return the function to be optimized.

Definition at line 88 of file `lrstdp.hpp`.

22.107.3.12 `LRSDPFunction& mlpack::optimization::LRSDP::Function () [inline]`

Modify the function to be optimized.

Definition at line 90 of file `lrstdp.hpp`.

22.107.3.13 `double mlpack::optimization::LRSDP::Optimize (arma::mat & coordinates)`

Optimize the **LRSDP** (p. 468) and return the final objective value.

The given coordinates will be modified to contain the final solution.

Parameters

<i>coordinates</i>	Starting coordinates for the optimization.
--------------------	--

22.107.3.14 `std::string mlpack::optimization::LRSDP::ToString () const`

Return a string representation of the object.

22.107.4 Member Data Documentation

22.107.4.1 `AugLagrangian<LRSDPFunction> mlpack::optimization::LRSDP::augLag [private]`

The **AugLagrangian** (p. 441) object which will be used for optimization.

Definition at line 105 of file `lrstdp.hpp`.

Referenced by `AugLag()`.

22.107.4.2 `LRSDPFunction mlpack::optimization::LRSDP::function [private]`

Function to optimize, which the **AugLagrangian** (p. 441) object holds.

Definition at line 102 of file `lrstdp.hpp`.

The documentation for this class was generated from the following file:

- `src/mlpack/core/optimizers/lrstdp/lrstdp.hpp`

22.108 mlpack::optimization::LRSDPFunction Class Reference

The objective function that **LRSDP** (p. 468) is trying to optimize.

Public Member Functions

- **LRSDPFunction** (const size_t numConstraints, const arma::mat &initialPoint)
Construct the **LRSDPFunction** (p. 472) with the given initial point and number of constraints.
- const std::vector< arma::mat > & **A** () const
Return the vector of *A* matrices (which correspond to the constraints).
- std::vector< arma::mat > & **A** ()
Modify the vector of *A* matrices (which correspond to the constraints).
- const arma::uvec & **AModes** () const
Return the vector of modes for the *A* matrices.
- arma::uvec & **AModes** ()
Modify the vector of modes for the *A* matrices.
- const arma::vec & **B** () const
Return the vector of *B* values.
- arma::vec & **B** ()
Modify the vector of *B* values.
- const arma::mat & **C** () const
Return the objective function matrix (*C*).
- arma::mat & **C** ()
Modify the objective function matrix (*C*).
- double **Evaluate** (const arma::mat &coordinates) const
Evaluate the objective function of the **LRSDP** (p. 468) (no constraints) at the given coordinates.
- double **EvaluateConstraint** (const size_t index, const arma::mat &coordinates) const
Evaluate a particular constraint of the **LRSDP** (p. 468) at the given coordinates.
- const arma::mat & **GetInitialPoint** () const
Get the initial point of the **LRSDP** (p. 468).
- void **Gradient** (const arma::mat &coordinates, arma::mat &gradient) const
Evaluate the gradient of the **LRSDP** (p. 468) (no constraints) at the given coordinates.
- void **GradientConstraint** (const size_t index, const arma::mat &coordinates, arma::mat &gradient) const
Evaluate the gradient of a particular constraint of the **LRSDP** (p. 468) at the given coordinates.
- size_t **NumConstraints** () const
Get the number of constraints in the **LRSDP** (p. 468).
- std::string **ToString** () const
Return string representation of object.

Private Attributes

- std::vector< arma::mat > **a**
A_i for each constraint.
- arma::uvec **aModes**
1 if entries in matrix, 0 for normal.
- arma::vec **b**

b_i for each constraint.

- arma::mat **c**

Objective function matrix c .

- arma::mat **initialPoint**

Initial point.

22.108.1 Detailed Description

The objective function that **LRSDP** (p. 468) is trying to optimize.

Definition at line 27 of file lrsdp_function.hpp.

22.108.2 Constructor & Destructor Documentation

22.108.2.1 mpack::optimization::LRSDPFunction::LRSDPFunction (const size_t numConstraints, const arma::mat & initialPoint)

Construct the **LRSDPFunction** (p. 472) with the given initial point and number of constraints.

Set the A, B, and C matrices for each constraint using the **A()** (p. 473), **B()** (p. 474), and **C()** (p. 474) functions.

22.108.3 Member Function Documentation

22.108.3.1 const std::vector<arma::mat> & mpack::optimization::LRSDPFunction::A () const [inline]

Return the vector of A matrices (which correspond to the constraints).

Definition at line 75 of file lrsdp_function.hpp.

References a.

22.108.3.2 std::vector<arma::mat> & mpack::optimization::LRSDPFunction::A () [inline]

Modify the vector of A matrices (which correspond to the constraints).

Definition at line 77 of file lrsdp_function.hpp.

References a.

22.108.3.3 const arma::uvec & mpack::optimization::LRSDPFunction::AModes () const [inline]

Return the vector of modes for the A matrices.

Definition at line 80 of file lrsdp_function.hpp.

References aModes.

22.108.3.4 arma::uvec & mpack::optimization::LRSDPFunction::AModes () [inline]

Modify the vector of modes for the A matrices.

Definition at line 82 of file lrsdp_function.hpp.

References aModes.

22.108.3.5 `const arma::vec& mlpack::optimization::LRSDPFunction::B () const` `[inline]`

Return the vector of B values.

Definition at line 85 of file `lrscp_function.hpp`.

References `b`.

22.108.3.6 `arma::vec& mlpack::optimization::LRSDPFunction::B ()` `[inline]`

Modify the vector of B values.

Definition at line 87 of file `lrscp_function.hpp`.

References `b`.

22.108.3.7 `const arma::mat& mlpack::optimization::LRSDPFunction::C () const` `[inline]`

Return the objective function matrix (C).

Definition at line 70 of file `lrscp_function.hpp`.

References `c`.

22.108.3.8 `arma::mat& mlpack::optimization::LRSDPFunction::C ()` `[inline]`

Modify the objective function matrix (C).

Definition at line 72 of file `lrscp_function.hpp`.

References `c`.

22.108.3.9 `double mlpack::optimization::LRSDPFunction::Evaluate (const arma::mat & coordinates) const`

Evaluate the objective function of the **LRSDP** (p. 468) (no constraints) at the given coordinates.

22.108.3.10 `double mlpack::optimization::LRSDPFunction::EvaluateConstraint (const size_t index, const arma::mat & coordinates) const`

Evaluate a particular constraint of the **LRSDP** (p. 468) at the given coordinates.

22.108.3.11 `const arma::mat& mlpack::optimization::LRSDPFunction::GetInitialPoint () const` `[inline]`

Get the initial point of the **LRSDP** (p. 468).

Definition at line 67 of file `lrscp_function.hpp`.

References `initialPoint`.

22.108.3.12 `void mlpack::optimization::LRSDPFunction::Gradient (const arma::mat & coordinates, arma::mat & gradient) const`

Evaluate the gradient of the **LRSDP** (p. 468) (no constraints) at the given coordinates.

22.108.3.13 `void mpack::optimization::LRSDPFunction::GradientConstraint (const size_t index, const arma::mat & coordinates, arma::mat & gradient) const`

Evaluate the gradient of a particular constraint of the **LRSDP** (p. 468) at the given coordinates.

22.108.3.14 `size_t mpack::optimization::LRSDPFunction::NumConstraints () const` `[inline]`

Get the number of constraints in the **LRSDP** (p. 468).

Definition at line 64 of file `lrmdp_function.hpp`.

References `b`.

22.108.3.15 `std::string mpack::optimization::LRSDPFunction::ToString () const`

Return string representation of object.

22.108.4 Member Data Documentation

22.108.4.1 `std::vector<arma::mat> mpack::optimization::LRSDPFunction::a` `[private]`

A_i for each constraint.

Definition at line 96 of file `lrmdp_function.hpp`.

Referenced by `A()`.

22.108.4.2 `arma::uvec mpack::optimization::LRSDPFunction::aModes` `[private]`

1 if entries in matrix, 0 for normal.

Definition at line 103 of file `lrmdp_function.hpp`.

Referenced by `AModes()`.

22.108.4.3 `arma::vec mpack::optimization::LRSDPFunction::b` `[private]`

b_i for each constraint.

Definition at line 98 of file `lrmdp_function.hpp`.

Referenced by `B()`, and `NumConstraints()`.

22.108.4.4 `arma::mat mpack::optimization::LRSDPFunction::c` `[private]`

Objective function matrix `c`.

Definition at line 94 of file `lrmdp_function.hpp`.

Referenced by `C()`.

22.108.4.5 `arma::mat mpack::optimization::LRSDPFunction::initialPoint` `[private]`

Initial point.

Definition at line 101 of file `lrsdp_function.hpp`.

Referenced by `GetInitialPoint()`.

The documentation for this class was generated from the following file:

- `src/mlpack/core/optimizers/lrsdp/lrsdp_function.hpp`

22.109 `mlpack::optimization::SA< FunctionType, CoolingScheduleType >` Class Template Reference

Simulated Annealing is an stochastic optimization algorithm which is able to deliver near-optimal results quickly without knowing the gradient of the function being optimized.

Public Member Functions

- **SA** (`FunctionType &function`, `CoolingScheduleType &coolingSchedule`, `const size_t maxIterations=1000000`, `const double initT=10000.`, `const size_t initMoves=1000`, `const size_t moveCtrlSweep=100`, `const double tolerance=1e-5`, `const size_t maxToleranceSweep=3`, `const double maxMoveCoef=20`, `const double initMoveCoef=0.3`, `const double gain=0.3`)
Construct the SA (p. 476) optimizer with the given function and parameters.
- `const FunctionType &Function () const`
Get the instantiated function to be optimized.
- `FunctionType &Function ()`
Modify the instantiated function.
- `double Gain () const`
Get the gain.
- `double &Gain ()`
Modify the gain.
- `size_t InitMoves () const`
Get the initial moves.
- `size_t &InitMoves ()`
Modify the initial moves.
- `size_t MaxIterations () const`
Get the maximum number of iterations.
- `size_t &MaxIterations ()`
Modify the maximum number of iterations.
- `arma::mat MaxMove () const`
Get the maximum move size of each parameter.
- `arma::mat &MaxMove ()`
Modify the maximum move size of each parameter.
- `size_t MaxToleranceSweep () const`
Get the maxToleranceSweep.
- `size_t &MaxToleranceSweep ()`
Modify the maxToleranceSweep.
- `size_t MoveCtrlSweep () const`
Get sweeps per move control.
- `size_t &MoveCtrlSweep ()`

- Modify sweeps per move control.*
- arma::mat **MoveSize** () const
Get move size of each parameter.
- arma::mat & **MoveSize** ()
Modify move size of each parameter.
- double **Optimize** (arma::mat &iterate)
Optimize the given function using simulated annealing.
- double **Temperature** () const
Get the temperature.
- double & **Temperature** ()
Modify the temperature.
- double **Tolerance** () const
Get the tolerance.
- double & **Tolerance** ()
Modify the tolerance.
- std::string **ToString** () const
Return a string representation of this object.

Private Member Functions

- void **GenerateMove** (arma::mat &iterate, arma::mat &accept, double &energy, size_t &idx, size_t &sweep↔ Counter)
GenerateMove proposes a move on element iterate(idx), and determines if that move is acceptable or not according to the Metropolis criterion.
- void **MoveControl** (const size_t nMoves, arma::mat &accept)
***MoveControl()** (p. 481) uses a proportional feedback control to determine the size parameter to pass to the move generation distribution.*

Private Attributes

- CoolingScheduleType & **coolingSchedule**
The cooling schedule being used.
- FunctionType & **function**
The function to be optimized.
- double **gain**
Proportional control in feedback move control.
- size_t **initMoves**
The number of initial moves before reducing the temperature.
- size_t **maxIterations**
The maximum number of iterations.
- arma::mat **maxMove**
Maximum move size of each parameter.
- size_t **maxToleranceSweep**
Number of sweeps in tolerance before system is considered frozen.
- size_t **moveCtrlSweep**
*The number of sweeps before a **MoveControl()** (p. 481) call.*
- arma::mat **moveSize**

Move size of each parameter.

- double **temperature**

The current temperature.

- double **tolerance**

Tolerance for convergence.

22.109.1 Detailed Description

```
template<typename FunctionType, typename CoolingScheduleType = ExponentialSchedule>class mlpack::optimization::SA<
FunctionType, CoolingScheduleType >
```

Simulated Annealing is an stochastic optimization algorithm which is able to deliver near-optimal results quickly without knowing the gradient of the function being optimized.

It has unique hill climbing capability that makes it less vulnerable to local minima. This implementation uses exponential cooling schedule and feedback move control by default, but the cooling schedule can be changed via a template parameter.

The algorithm keeps the temperature at initial temperature for `initMove` steps to get rid of the dependency on the initial condition. After that, it cools every step until the system is considered frozen or `maxIterations` is reached.

At each step, **SA** (p. 476) only perturbs one parameter at a time. When **SA** (p. 476) has perturbed all parameters in a problem, a sweep has been completed. Every `moveCtrlSweep` sweeps, the algorithm does feedback move control to change the average move size depending on the responsiveness of each parameter. Parameter gain controls the proportion of the feedback control.

The system is considered "frozen" when its score fails to change more than tolerance for `maxToleranceSweep` consecutive sweeps.

For **SA** (p. 476) to work, the `FunctionType` parameter must implement the following two methods:

```
double Evaluate(const arma::mat& coordinates); arma::mat& GetInitialPoint();
```

and the `CoolingScheduleType` parameter must implement the following method:

```
double NextTemperature(const double currentTemperature, const double currentValue);
```

which returns the next temperature given current temperature and the value of the function being optimized.

Template Parameters

<i>FunctionType</i>	objective function type to be minimized.
<i>CoolingScheduleType</i>	type for cooling schedule

Definition at line 67 of file `sa.hpp`.

22.109.2 Constructor & Destructor Documentation

```
22.109.2.1 template<typename FunctionType , typename CoolingScheduleType = ExponentialSchedule>
mlpack::optimization::SA< FunctionType, CoolingScheduleType >::SA ( FunctionType & function,
CoolingScheduleType & coolingSchedule, const size_t maxIterations = 1000000, const double initT = 10000 .,
const size_t initMoves = 1000, const size_t moveCtrlSweep = 100, const double tolerance = 1e-5, const size_t
maxToleranceSweep = 3, const double maxMoveCoef = 20, const double initMoveCoef = 0.3, const double gain =
0.3 )
```

Construct the **SA** (p. 476) optimizer with the given function and parameters.

Parameters

<i>function</i>	Function to be minimized.
<i>coolingSchedule</i>	Instantiated cooling schedule.
<i>maxIterations</i>	Maximum number of iterations allowed (0 indicates no limit).
<i>initT</i>	Initial temperature.
<i>initMoves</i>	Number of initial iterations without changing temperature.
<i>moveCtrlSweep</i>	Sweeps per feedback move control.
<i>tolerance</i>	Tolerance to consider system frozen.
<i>maxToleranceSweep</i>	Maximum sweeps below tolerance to consider system frozen.
<i>maxMoveCoef</i>	Maximum move size.
<i>initMoveCoef</i>	Initial move size.
<i>gain</i>	Proportional control in feedback move control.

22.109.3 Member Function Documentation

22.109.3.1 `template<typename FunctionType , typename CoolingScheduleType = ExponentialSchedule> const FunctionType& mlpack::optimization::SA< FunctionType, CoolingScheduleType >::Function () const [inline]`

Get the instantiated function to be optimized.

Definition at line 109 of file sa.hpp.

22.109.3.2 `template<typename FunctionType , typename CoolingScheduleType = ExponentialSchedule> FunctionType& mlpack::optimization::SA< FunctionType, CoolingScheduleType >::Function () [inline]`

Modify the instantiated function.

Definition at line 111 of file sa.hpp.

22.109.3.3 `template<typename FunctionType , typename CoolingScheduleType = ExponentialSchedule> double mlpack::optimization::SA< FunctionType, CoolingScheduleType >::Gain () const [inline]`

Get the gain.

Definition at line 139 of file sa.hpp.

References mlpack::optimization::SA< FunctionType, CoolingScheduleType >::gain.

22.109.3.4 `template<typename FunctionType , typename CoolingScheduleType = ExponentialSchedule> double& mlpack::optimization::SA< FunctionType, CoolingScheduleType >::Gain () [inline]`

Modify the gain.

Definition at line 141 of file sa.hpp.

References mlpack::optimization::SA< FunctionType, CoolingScheduleType >::gain.

22.109.3.5 `template<typename FunctionType , typename CoolingScheduleType = ExponentialSchedule> void
mlpack::optimization::SA< FunctionType, CoolingScheduleType >::GenerateMove (arma::mat & iterate,
arma::mat & accept, double & energy, size_t & idx, size_t & sweepCounter) [private]`

GenerateMove proposes a move on element `iterate(idx)`, and determines if that move is acceptable or not according to the Metropolis criterion.

After that it increments `idx` so the next call will make a move on next parameters. When all elements of the state have been moved (a sweep), it resets `idx` and increments `sweepCounter`. When `sweepCounter` reaches `moveCtrlSweep`, it performs **MoveControl()** (p. 481) and resets `sweepCounter`.

Parameters

<i>iterate</i>	Current optimization position.
<i>accept</i>	Matrix representing which parameters have had accepted moves.
<i>energy</i>	Current energy of the system.
<i>idx</i>	Current parameter to modify.
<i>sweepCounter</i>	Current counter representing how many sweeps have been completed.

22.109.3.6 `template<typename FunctionType , typename CoolingScheduleType = ExponentialSchedule> size_t
mlpack::optimization::SA< FunctionType, CoolingScheduleType >::InitMoves () const [inline]`

Get the initial moves.

Definition at line 119 of file `sa.hpp`.

References `mlpack::optimization::SA< FunctionType, CoolingScheduleType >::initMoves`.

22.109.3.7 `template<typename FunctionType , typename CoolingScheduleType = ExponentialSchedule> size_t&
mlpack::optimization::SA< FunctionType, CoolingScheduleType >::InitMoves () [inline]`

Modify the initial moves.

Definition at line 121 of file `sa.hpp`.

References `mlpack::optimization::SA< FunctionType, CoolingScheduleType >::initMoves`.

22.109.3.8 `template<typename FunctionType , typename CoolingScheduleType = ExponentialSchedule> size_t
mlpack::optimization::SA< FunctionType, CoolingScheduleType >::MaxIterations () const [inline]`

Get the maximum number of iterations.

Definition at line 144 of file `sa.hpp`.

References `mlpack::optimization::SA< FunctionType, CoolingScheduleType >::maxIterations`.

22.109.3.9 `template<typename FunctionType , typename CoolingScheduleType = ExponentialSchedule> size_t&
mlpack::optimization::SA< FunctionType, CoolingScheduleType >::MaxIterations () [inline]`

Modify the maximum number of iterations.

Definition at line 146 of file `sa.hpp`.

References `mlpack::optimization::SA< FunctionType, CoolingScheduleType >::maxIterations`.

```
22.109.3.10 template<typename FunctionType , typename CoolingScheduleType = ExponentialSchedule> arma::mat
mlpack::optimization::SA< FunctionType, CoolingScheduleType >::MaxMove ( ) const [inline]
```

Get the maximum move size of each parameter.

Definition at line 149 of file sa.hpp.

References mlpack::optimization::SA< FunctionType, CoolingScheduleType >::maxMove.

```
22.109.3.11 template<typename FunctionType , typename CoolingScheduleType = ExponentialSchedule> arma::mat&
mlpack::optimization::SA< FunctionType, CoolingScheduleType >::MaxMove ( ) [inline]
```

Modify the maximum move size of each parameter.

Definition at line 151 of file sa.hpp.

References mlpack::optimization::SA< FunctionType, CoolingScheduleType >::maxMove.

```
22.109.3.12 template<typename FunctionType , typename CoolingScheduleType = ExponentialSchedule> size_t
mlpack::optimization::SA< FunctionType, CoolingScheduleType >::MaxToleranceSweep ( ) const
[inline]
```

Get the maxToleranceSweep.

Definition at line 134 of file sa.hpp.

References mlpack::optimization::SA< FunctionType, CoolingScheduleType >::maxToleranceSweep.

```
22.109.3.13 template<typename FunctionType , typename CoolingScheduleType = ExponentialSchedule> size_t&
mlpack::optimization::SA< FunctionType, CoolingScheduleType >::MaxToleranceSweep ( ) [inline]
```

Modify the maxToleranceSweep.

Definition at line 136 of file sa.hpp.

References mlpack::optimization::SA< FunctionType, CoolingScheduleType >::maxToleranceSweep.

```
22.109.3.14 template<typename FunctionType , typename CoolingScheduleType = ExponentialSchedule> void
mlpack::optimization::SA< FunctionType, CoolingScheduleType >::MoveControl ( const size_t nMoves,
arma::mat & accept ) [private]
```

MoveControl() (p. 481) uses a proportional feedback control to determine the size parameter to pass to the move generation distribution.

The target of such move control is to make the acceptance ratio, $\text{accept}/n\text{Moves}$, be as close to 0.44 as possible. Generally speaking, the larger the move size is, the larger the function value change of the move will be, and less likely such move will be accepted by the Metropolis criterion. Thus, the move size is controlled by

$$\log(\text{moveSize}) = \log(\text{moveSize}) + \text{gain} * (\text{accept}/n\text{Moves} - \text{target})$$

For more theory and the mysterious 0.44 value, see Jimmy K.-C. Lam and Jean-Marc Delosme. 'An efficient simulated annealing schedule: derivation'. Technical Report 8816, Yale University, 1988.

Parameters

<i>nMoves</i>	Number of moves since last call.
<i>accept</i>	Matrix representing which parameters have had accepted moves.

22.109.3.15 `template<typename FunctionType , typename CoolingScheduleType = ExponentialSchedule> size_t
mlpack::optimization::SA< FunctionType, CoolingScheduleType >::MoveCtrlSweep () const [inline]`

Get sweeps per move control.

Definition at line 124 of file sa.hpp.

References `mlpack::optimization::SA< FunctionType, CoolingScheduleType >::moveCtrlSweep`.

22.109.3.16 `template<typename FunctionType , typename CoolingScheduleType = ExponentialSchedule> size_t&
mlpack::optimization::SA< FunctionType, CoolingScheduleType >::MoveCtrlSweep () [inline]`

Modify sweeps per move control.

Definition at line 126 of file sa.hpp.

References `mlpack::optimization::SA< FunctionType, CoolingScheduleType >::moveCtrlSweep`.

22.109.3.17 `template<typename FunctionType , typename CoolingScheduleType = ExponentialSchedule> arma::mat
mlpack::optimization::SA< FunctionType, CoolingScheduleType >::MoveSize () const [inline]`

Get move size of each parameter.

Definition at line 154 of file sa.hpp.

References `mlpack::optimization::SA< FunctionType, CoolingScheduleType >::moveSize`.

22.109.3.18 `template<typename FunctionType , typename CoolingScheduleType = ExponentialSchedule> arma::mat&
mlpack::optimization::SA< FunctionType, CoolingScheduleType >::MoveSize () [inline]`

Modify move size of each parameter.

Definition at line 156 of file sa.hpp.

References `mlpack::optimization::SA< FunctionType, CoolingScheduleType >::moveSize`.

22.109.3.19 `template<typename FunctionType , typename CoolingScheduleType = ExponentialSchedule> double
mlpack::optimization::SA< FunctionType, CoolingScheduleType >::Optimize (arma::mat & iterate)`

Optimize the given function using simulated annealing.

The given starting point will be modified to store the finishing point of the algorithm, and the final objective value is returned.

Parameters

<i>iterate</i>	Starting point (will be modified).
----------------	------------------------------------

Returns

Objective value of the final point.

22.109.3.20 `template<typename FunctionType , typename CoolingScheduleType = ExponentialSchedule> double
mlpack::optimization::SA< FunctionType, CoolingScheduleType >::Temperature () const [inline]`

Get the temperature.

Definition at line 114 of file sa.hpp.

References mlpack::optimization::SA< FunctionType, CoolingScheduleType >::temperature.

22.109.3.21 `template<typename FunctionType , typename CoolingScheduleType = ExponentialSchedule> double&
mlpack::optimization::SA< FunctionType, CoolingScheduleType >::Temperature () [inline]`

Modify the temperature.

Definition at line 116 of file sa.hpp.

References mlpack::optimization::SA< FunctionType, CoolingScheduleType >::temperature.

22.109.3.22 `template<typename FunctionType , typename CoolingScheduleType = ExponentialSchedule> double
mlpack::optimization::SA< FunctionType, CoolingScheduleType >::Tolerance () const [inline]`

Get the tolerance.

Definition at line 129 of file sa.hpp.

References mlpack::optimization::SA< FunctionType, CoolingScheduleType >::tolerance.

22.109.3.23 `template<typename FunctionType , typename CoolingScheduleType = ExponentialSchedule> double&
mlpack::optimization::SA< FunctionType, CoolingScheduleType >::Tolerance () [inline]`

Modify the tolerance.

Definition at line 131 of file sa.hpp.

References mlpack::optimization::SA< FunctionType, CoolingScheduleType >::tolerance.

22.109.3.24 `template<typename FunctionType , typename CoolingScheduleType = ExponentialSchedule> std::string
mlpack::optimization::SA< FunctionType, CoolingScheduleType >::ToString () const`

Return a string representation of this object.

22.109.4 Member Data Documentation

22.109.4.1 `template<typename FunctionType , typename CoolingScheduleType = ExponentialSchedule> CoolingScheduleType&
mlpack::optimization::SA< FunctionType, CoolingScheduleType >::coolingSchedule [private]`

The cooling schedule being used.

Definition at line 164 of file sa.hpp.

22.109.4.2 `template<typename FunctionType , typename CoolingScheduleType = ExponentialSchedule> FunctionType&
mlpack::optimization::SA< FunctionType, CoolingScheduleType >::function [private]`

The function to be optimized.

Definition at line 162 of file sa.hpp.

22.109.4.3 `template<typename FunctionType , typename CoolingScheduleType = ExponentialSchedule> double
mlpack::optimization::SA< FunctionType, CoolingScheduleType >::gain [private]`

Proportional control in feedback move control.

Definition at line 178 of file sa.hpp.

Referenced by `mlpack::optimization::SA< FunctionType, CoolingScheduleType >::Gain()`.

22.109.4.4 `template<typename FunctionType , typename CoolingScheduleType = ExponentialSchedule> size_t
mlpack::optimization::SA< FunctionType, CoolingScheduleType >::initMoves [private]`

The number of initial moves before reducing the temperature.

Definition at line 170 of file sa.hpp.

Referenced by `mlpack::optimization::SA< FunctionType, CoolingScheduleType >::InitMoves()`.

22.109.4.5 `template<typename FunctionType , typename CoolingScheduleType = ExponentialSchedule> size_t
mlpack::optimization::SA< FunctionType, CoolingScheduleType >::maxIterations [private]`

The maximum number of iterations.

Definition at line 166 of file sa.hpp.

Referenced by `mlpack::optimization::SA< FunctionType, CoolingScheduleType >::MaxIterations()`.

22.109.4.6 `template<typename FunctionType , typename CoolingScheduleType = ExponentialSchedule> arma::mat
mlpack::optimization::SA< FunctionType, CoolingScheduleType >::maxMove [private]`

Maximum move size of each parameter.

Definition at line 181 of file sa.hpp.

Referenced by `mlpack::optimization::SA< FunctionType, CoolingScheduleType >::MaxMove()`.

22.109.4.7 `template<typename FunctionType , typename CoolingScheduleType = ExponentialSchedule> size_t
mlpack::optimization::SA< FunctionType, CoolingScheduleType >::maxToleranceSweep [private]`

Number of sweeps in tolerance before system is considered frozen.

Definition at line 176 of file sa.hpp.

Referenced by `mlpack::optimization::SA< FunctionType, CoolingScheduleType >::MaxToleranceSweep()`.

22.109.4.8 `template<typename FunctionType , typename CoolingScheduleType = ExponentialSchedule> size_t
mlpack::optimization::SA< FunctionType, CoolingScheduleType >::moveCtrlSweep [private]`

The number of sweeps before a **MoveControl()** (p. 481) call.

Definition at line 172 of file sa.hpp.

Referenced by mlpack::optimization::SA< FunctionType, CoolingScheduleType >::MoveCtrlSweep().

22.109.4.9 `template<typename FunctionType , typename CoolingScheduleType = ExponentialSchedule> arma::mat
mlpack::optimization::SA< FunctionType, CoolingScheduleType >::moveSize [private]`

Move size of each parameter.

Definition at line 183 of file sa.hpp.

Referenced by mlpack::optimization::SA< FunctionType, CoolingScheduleType >::MoveSize().

22.109.4.10 `template<typename FunctionType , typename CoolingScheduleType = ExponentialSchedule> double
mlpack::optimization::SA< FunctionType, CoolingScheduleType >::temperature [private]`

The current temperature.

Definition at line 168 of file sa.hpp.

Referenced by mlpack::optimization::SA< FunctionType, CoolingScheduleType >::Temperature().

22.109.4.11 `template<typename FunctionType , typename CoolingScheduleType = ExponentialSchedule> double
mlpack::optimization::SA< FunctionType, CoolingScheduleType >::tolerance [private]`

Tolerance for convergence.

Definition at line 174 of file sa.hpp.

Referenced by mlpack::optimization::SA< FunctionType, CoolingScheduleType >::Tolerance().

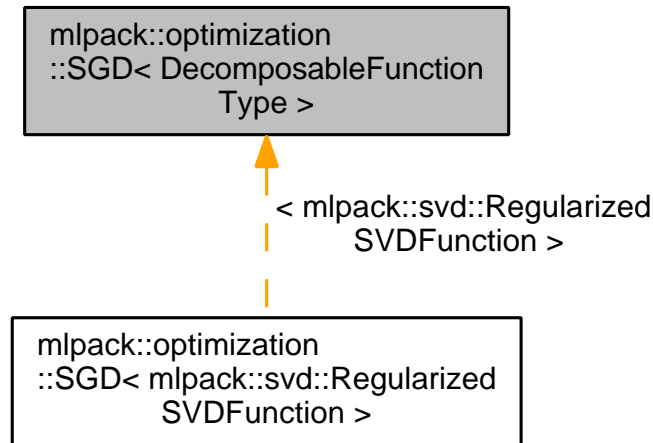
The documentation for this class was generated from the following file:

- src/mlpack/core/optimizers/sa/sa.hpp

22.110 mlpack::optimization::SGD< DecomposableFunctionType > Class Template Reference

Stochastic Gradient Descent is a technique for minimizing a function which can be expressed as a sum of other functions.

Inheritance diagram for `mlpack::optimization::SGD< DecomposableFunctionType >`:



Public Member Functions

- **SGD** (`DecomposableFunctionType &function`, `const double stepSize=0.01`, `const size_t maxIterations=100000`, `const double tolerance=1e-5`, `const bool shuffle=true`)
*Construct the **SGD** (p. 485) optimizer with the given function and parameters.*
- `const DecomposableFunctionType & Function () const`
Get the instantiated function to be optimized.
- `DecomposableFunctionType & Function ()`
Modify the instantiated function.
- `size_t MaxIterations () const`
Get the maximum number of iterations (0 indicates no limit).
- `size_t & MaxIterations ()`
Modify the maximum number of iterations (0 indicates no limit).
- `double Optimize (arma::mat &iterate)`
Optimize the given function using stochastic gradient descent.
- `template<> double Optimize (arma::mat ¶meters)`
- `bool Shuffle () const`
Get whether or not the individual functions are shuffled.
- `bool & Shuffle ()`
Modify whether or not the individual functions are shuffled.
- `double StepSize () const`
Get the step size.
- `double & StepSize ()`
Modify the step size.
- `double Tolerance () const`
Get the tolerance for termination.
- `double & Tolerance ()`
Modify the tolerance for termination.
- `std::string ToString () const`

Private Attributes

- **DecomposableFunctionType & function**
The instantiated function.
- **size_t maxIterations**
The maximum number of allowed iterations.
- **bool shuffle**
Controls whether or not the individual functions are shuffled when iterating.
- **double stepSize**
The step size for each example.
- **double tolerance**
The tolerance for termination.

22.110.1 Detailed Description

template<typename DecomposableFunctionType>class mpack::optimization::SGD< DecomposableFunctionType >

Stochastic Gradient Descent is a technique for minimizing a function which can be expressed as a sum of other functions. That is, suppose we have

$$f(A) = \sum_{i=0}^n f_i(A)$$

and our task is to minimize A . Stochastic gradient descent iterates over each function $f_i(A)$, producing the following update scheme:

$$A_{j+1} = A_j + \alpha \nabla f_i(A)$$

where α is a parameter which specifies the step size. i is chosen according to j (the iteration number). The **SGD** (p. 485) class supports either scanning through each of the n functions $f_i(A)$ linearly, or in a random sequence. The algorithm continues until j reaches the maximum number of iterations – or when a full sequence of updates through each of the n functions $f_i(A)$ produces an improvement within a certain tolerance ϵ . That is,

$$|f(A_{j+n}) - f(A_j)| < \epsilon.$$

The parameter ϵ is specified by the tolerance parameter to the constructor; n is specified by the maxIterations parameter.

This class is useful for data-dependent functions whose objective function can be expressed as a sum of objective functions operating on an individual point. Then, **SGD** (p. 485) considers the gradient of the objective function operating on an individual point in its update of A .

For **SGD** (p. 485) to work, a DecomposableFunctionType template parameter is required. This class must implement the following function:

```
size_t NumFunctions(); double Evaluate(const arma::mat& coordinates, const size_t i); void Gradient(const arma::mat& coordinates, const size_t i, arma::mat& gradient);
```

NumFunctions() should return the number of functions (n), and in the other two functions, the parameter i refers to which individual function (or gradient) is being evaluated. So, for the case of a data-dependent function, such as NCA (see **mpack::nca::NCA** (p. 376)), NumFunctions() should return the number of points in the dataset, and Evaluate(coordinates, 0) will evaluate the objective function on the first point in the dataset (presumably, the dataset is held internally in the DecomposableFunctionType).

Template Parameters

<i>DecomposableFunctionType</i>	Decomposable objective function type to be minimized.
---------------------------------	---

Definition at line 78 of file sgd.hpp.

22.110.2 Constructor & Destructor Documentation

22.110.2.1 `template<typename DecomposableFunctionType> mpack::optimization::SGD< DecomposableFunctionType >::SGD (DecomposableFunctionType & function, const double stepSize = 0.01, const size_t maxIterations = 100000, const double tolerance = 1e-5, const bool shuffle = true)`

Construct the **SGD** (p. 485) optimizer with the given function and parameters.

Parameters

<i>function</i>	Function to be optimized (minimized).
<i>stepSize</i>	Step size for each iteration.
<i>maxIterations</i>	Maximum number of iterations allowed (0 means no limit).
<i>tolerance</i>	Maximum absolute tolerance to terminate algorithm.
<i>shuffle</i>	If true, the function order is shuffled; otherwise, each function is visited in linear order.

22.110.3 Member Function Documentation

22.110.3.1 `template<typename DecomposableFunctionType> const DecomposableFunctionType& mpack::optimization::SGD< DecomposableFunctionType >::Function () const [inline]`

Get the instantiated function to be optimized.

Definition at line 109 of file sgd.hpp.

22.110.3.2 `template<typename DecomposableFunctionType> DecomposableFunctionType& mpack::optimization::SGD< DecomposableFunctionType >::Function () [inline]`

Modify the instantiated function.

Definition at line 111 of file sgd.hpp.

22.110.3.3 `template<typename DecomposableFunctionType> size_t mpack::optimization::SGD< DecomposableFunctionType >::MaxIterations () const [inline]`

Get the maximum number of iterations (0 indicates no limit).

Definition at line 119 of file sgd.hpp.

22.110.3.4 `template<typename DecomposableFunctionType> size_t& mpack::optimization::SGD< DecomposableFunctionType >::MaxIterations () [inline]`

Modify the maximum number of iterations (0 indicates no limit).

Definition at line 121 of file sgd.hpp.

22.110.3.5 `template<typename DecomposableFunctionType> double mpack::optimization::SGD< DecomposableFunctionType >::Optimize (arma::mat & iterate)`

Optimize the given function using stochastic gradient descent.

The given starting point will be modified to store the finishing point of the algorithm, and the final objective value is returned.

Parameters

<i>iterate</i>	Starting point (will be modified).
----------------	------------------------------------

Returns

Objective value of the final point.

22.110.3.6 `template<> double mpack::optimization::SGD< mpack::svd::RegularizedSVDFunction >::Optimize (arma::mat & parameters)`

Used because the gradient affects only a small number of parameters per example, and thus the normal abstraction does not work as fast as we might like it to.

22.110.3.7 `template<typename DecomposableFunctionType> bool mpack::optimization::SGD< DecomposableFunctionType >::Shuffle () const [inline]`

Get whether or not the individual functions are shuffled.

Definition at line 129 of file sgd.hpp.

22.110.3.8 `template<typename DecomposableFunctionType> bool& mpack::optimization::SGD< DecomposableFunctionType >::Shuffle () [inline]`

Modify whether or not the individual functions are shuffled.

Definition at line 131 of file sgd.hpp.

22.110.3.9 `template<typename DecomposableFunctionType> double mpack::optimization::SGD< DecomposableFunctionType >::StepSize () const [inline]`

Get the step size.

Definition at line 114 of file sgd.hpp.

22.110.3.10 `template<typename DecomposableFunctionType> double& mpack::optimization::SGD< DecomposableFunctionType >::StepSize () [inline]`

Modify the step size.

Definition at line 116 of file sgd.hpp.

22.110.3.11 `template<typename DecomposableFunctionType> double mlpack::optimization::SGD< DecomposableFunctionType >::Tolerance () const [inline]`

Get the tolerance for termination.

Definition at line 124 of file `sgd.hpp`.

22.110.3.12 `template<typename DecomposableFunctionType> double& mlpack::optimization::SGD< DecomposableFunctionType >::Tolerance () [inline]`

Modify the tolerance for termination.

Definition at line 126 of file `sgd.hpp`.

22.110.3.13 `template<typename DecomposableFunctionType> std::string mlpack::optimization::SGD< DecomposableFunctionType >::ToString () const`

22.110.4 Member Data Documentation

22.110.4.1 `template<typename DecomposableFunctionType> DecomposableFunctionType& mlpack::optimization::SGD< DecomposableFunctionType >::function [private]`

The instantiated function.

Definition at line 138 of file `sgd.hpp`.

22.110.4.2 `template<typename DecomposableFunctionType> size_t mlpack::optimization::SGD< DecomposableFunctionType >::maxIterations [private]`

The maximum number of allowed iterations.

Definition at line 144 of file `sgd.hpp`.

Referenced by `mlpack::optimization::SGD< mlpack::svd::RegularizedSVDFunction >::MaxIterations()`.

22.110.4.3 `template<typename DecomposableFunctionType> bool mlpack::optimization::SGD< DecomposableFunctionType >::shuffle [private]`

Controls whether or not the individual functions are shuffled when iterating.

Definition at line 151 of file `sgd.hpp`.

Referenced by `mlpack::optimization::SGD< mlpack::svd::RegularizedSVDFunction >::Shuffle()`.

22.110.4.4 `template<typename DecomposableFunctionType> double mlpack::optimization::SGD< DecomposableFunctionType >::stepSize [private]`

The step size for each example.

Definition at line 141 of file `sgd.hpp`.

Referenced by `mlpack::optimization::SGD< mlpack::svd::RegularizedSVDFunction >::StepSize()`.

22.110.4.5 `template<typename DecomposableFunctionType> double mlpack::optimization::SGD< DecomposableFunctionType >::tolerance [private]`

The tolerance for termination.

Definition at line 147 of file sgd.hpp.

Referenced by `mlpack::optimization::SGD< mlpack::svd::RegularizedSVDFunction >::Tolerance()`.

The documentation for this class was generated from the following file:

- `src/mlpack/core/optimizers/sgd/sgd.hpp`

22.111 mlpack::optimization::test::GeneralizedRosenbrockFunction Class Reference

The Generalized Rosenbrock function in n dimensions, defined by $f(x) = \sum_{i=1}^{n-1} (f_i(x))$ $f_i(x) = 100 * (x_i^2 - x_{i+1})^2 + (1 - x_i)^2$ $x_0 = [-1.2, 1, -1.2, 1, \dots]$.

Public Member Functions

- **GeneralizedRosenbrockFunction** (int n)
- double **Evaluate** (const arma::mat &coordinates) const
- double **Evaluate** (const arma::mat &coordinates, const size_t i) const
- const arma::mat & **GetInitialPoint** () const
- void **Gradient** (const arma::mat &coordinates, arma::mat &gradient) const
- void **Gradient** (const arma::mat &coordinates, const size_t i , arma::mat &gradient) const
- size_t **NumFunctions** () const

Private Attributes

- arma::mat **initialPoint**
- int n

22.111.1 Detailed Description

The Generalized Rosenbrock function in n dimensions, defined by $f(x) = \sum_{i=1}^{n-1} (f_i(x))$ $f_i(x) = 100 * (x_i^2 - x_{i+1})^2 + (1 - x_i)^2$ $x_0 = [-1.2, 1, -1.2, 1, \dots]$.

This should optimize to $f(x) = 0$, at $x = [1, 1, 1, 1, \dots]$.

This function can also be used for stochastic gradient descent (**SGD** (p.485)) as a decomposable function (`DecomposableFunctionType`), so there are other overloads of **Evaluate()** (p.492) and **Gradient()** (p.492) implemented, as well as **NumFunctions()** (p.492).

"An analysis of the behavior of a glass of genetic adaptive systems." K.A. De Jong. Ph.D. thesis, University of Michigan, 1975.

Definition at line 115 of file test_functions.hpp.

22.111.2 Constructor & Destructor Documentation

22.111.2.1 `mlpack::optimization::test::GeneralizedRosenbrockFunction::GeneralizedRosenbrockFunction (int n)`

22.111.3 Member Function Documentation

22.111.3.1 `double mlpack::optimization::test::GeneralizedRosenbrockFunction::Evaluate (const arma::mat & coordinates) const`

22.111.3.2 `double mlpack::optimization::test::GeneralizedRosenbrockFunction::Evaluate (const arma::mat & coordinates, const size_t i) const`

22.111.3.3 `const arma::mat& mlpack::optimization::test::GeneralizedRosenbrockFunction::GetInitialPoint () const`

22.111.3.4 `void mlpack::optimization::test::GeneralizedRosenbrockFunction::Gradient (const arma::mat & coordinates, arma::mat & gradient) const`

22.111.3.5 `void mlpack::optimization::test::GeneralizedRosenbrockFunction::Gradient (const arma::mat & coordinates, const size_t i, arma::mat & gradient) const`

22.111.3.6 `size_t mlpack::optimization::test::GeneralizedRosenbrockFunction::NumFunctions () const` `[inline]`

Definition at line 128 of file `test_functions.hpp`.

22.111.4 Member Data Documentation

22.111.4.1 `arma::mat mlpack::optimization::test::GeneralizedRosenbrockFunction::initialPoint` `[private]`

Definition at line 137 of file `test_functions.hpp`.

22.111.4.2 `int mlpack::optimization::test::GeneralizedRosenbrockFunction::n` `[private]`

Definition at line 138 of file `test_functions.hpp`.

The documentation for this class was generated from the following file:

- `src/mlpack/core/optimizers/lbfgs/test_functions.hpp`

22.112 mlpack::optimization::test::RosenbrockFunction Class Reference

The Rosenbrock function, defined by $f(x) = f_1(x) + f_2(x)$ $f_1(x) = 100 (x_2 - x_1^2)^2$ $f_2(x) = (1 - x_1)^2$ $x_0 = [-1.2, 1]$.

Public Member Functions

- **RosenbrockFunction** ()
- double **Evaluate** (const arma::mat &coordinates)
- const arma::mat & **GetInitialPoint** () const
- void **Gradient** (const arma::mat &coordinates, arma::mat &gradient)

Private Attributes

- arma::mat **initialPoint**

22.112.1 Detailed Description

The Rosenbrock function, defined by $f(x) = f_1(x) + f_2(x)$ $f_1(x) = 100 (x_2 - x_1^2)^2$ $f_2(x) = (1 - x_1)^2$ $x_0 = [-1.2, 1]$.

This should optimize to $f(x) = 0$, at $x = [1, 1]$.

"An automatic method for finding the greatest or least value of a function." H.H. Rosenbrock. 1960. Comput. J. 3., 175-184.

Definition at line 55 of file test_functions.hpp.

22.112.2 Constructor & Destructor Documentation

22.112.2.1 mlpack::optimization::test::RosenbrockFunction::RosenbrockFunction ()

22.112.3 Member Function Documentation

22.112.3.1 double mlpack::optimization::test::RosenbrockFunction::Evaluate (const arma::mat & *coordinates*)

22.112.3.2 const arma::mat& mlpack::optimization::test::RosenbrockFunction::GetInitialPoint () const

22.112.3.3 void mlpack::optimization::test::RosenbrockFunction::Gradient (const arma::mat & *coordinates*, arma::mat & *gradient*)

22.112.4 Member Data Documentation

22.112.4.1 arma::mat mlpack::optimization::test::RosenbrockFunction::initialPoint [private]

Definition at line 66 of file test_functions.hpp.

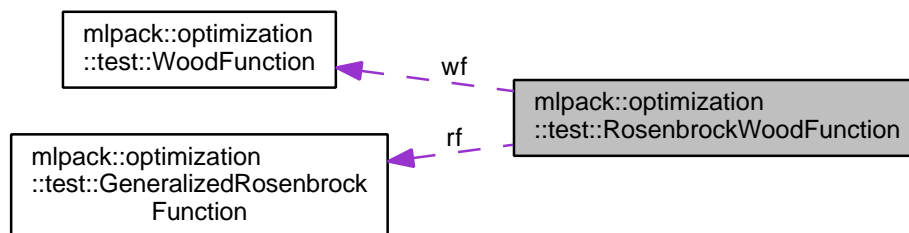
The documentation for this class was generated from the following file:

- src/mlpack/core/optimizers/lbfgs/test_functions.hpp

22.113 mlpack::optimization::test::RosenbrockWoodFunction Class Reference

The Generalized Rosenbrock function in 4 dimensions with the Wood Function in four dimensions.

Collaboration diagram for mlpack::optimization::test::RosenbrockWoodFunction:



Public Member Functions

- **RosenbrockWoodFunction** ()
- double **Evaluate** (const arma::mat &coordinates)
- const arma::mat & **GetInitialPoint** () const
- void **Gradient** (const arma::mat &coordinates, arma::mat &gradient)

Private Attributes

- arma::mat **initialPoint**
- **GeneralizedRosenbrockFunction** rf
- **WoodFunction** wf

22.113.1 Detailed Description

The Generalized Rosenbrock function in 4 dimensions with the Wood Function in four dimensions.

In this function we are actually optimizing a 2x4 matrix of coordinates, not a vector.

Definition at line 146 of file test_functions.hpp.

22.113.2 Constructor & Destructor Documentation

22.113.2.1 `mlpack::optimization::test::RosenbrockWoodFunction::RosenbrockWoodFunction ()`

22.113.3 Member Function Documentation

22.113.3.1 `double mlpack::optimization::test::RosenbrockWoodFunction::Evaluate (const arma::mat & coordinates)`

22.113.3.2 `const arma::mat& mlpack::optimization::test::RosenbrockWoodFunction::GetInitialPoint () const`

22.113.3.3 `void mlpack::optimization::test::RosenbrockWoodFunction::Gradient (const arma::mat & coordinates, arma::mat & gradient)`

22.113.4 Member Data Documentation

22.113.4.1 `arma::mat mlpack::optimization::test::RosenbrockWoodFunction::initialPoint [private]`

Definition at line 157 of file test_functions.hpp.

22.113.4.2 `GeneralizedRosenbrockFunction mlpack::optimization::test::RosenbrockWoodFunction::rf [private]`

Definition at line 158 of file test_functions.hpp.

22.113.4.3 `WoodFunction mlpack::optimization::test::RosenbrockWoodFunction::wf [private]`

Definition at line 159 of file test_functions.hpp.

The documentation for this class was generated from the following file:

- src/mlpack/core/optimizers/lbfgs/test_functions.hpp

22.114 mlpack::optimization::test::SGDTestFunction Class Reference

Very, very simple test function which is the composite of three other functions.

Public Member Functions

- **SGDTestFunction** ()
Nothing to do for the constructor.
- double **Evaluate** (const arma::mat &coordinates, const size_t i) const
Evaluate a function.
- arma::mat **GetInitialPoint** () const
Get the starting point.
- void **Gradient** (const arma::mat &coordinates, const size_t i, arma::mat &gradient) const
Evaluate the gradient of a function.
- size_t **NumFunctions** () const
Return 3 (the number of functions).

22.114.1 Detailed Description

Very, very simple test function which is the composite of three other functions.

The gradient is not very steep far away from the optimum, so a larger step size may be required to optimize it in a reasonable number of iterations.

Definition at line 27 of file test_function.hpp.

22.114.2 Constructor & Destructor Documentation

22.114.2.1 mlpack::optimization::test::SGDTestFunction::SGDTestFunction () `[inline]`

Nothing to do for the constructor.

Definition at line 31 of file test_function.hpp.

22.114.3 Member Function Documentation

22.114.3.1 double mlpack::optimization::test::SGDTestFunction::Evaluate (const arma::mat & *coordinates*, const size_t *i*) const

Evaluate a function.

22.114.3.2 arma::mat mlpack::optimization::test::SGDTestFunction::GetInitialPoint () const `[inline]`

Get the starting point.

Definition at line 37 of file test_function.hpp.

22.114.3.3 void mlpack::optimization::test::SGDTestFunction::Gradient (const arma::mat & *coordinates*, const size_t *i*, arma::mat & *gradient*) const

Evaluate the gradient of a function.

22.114.3.4 `size_t mpack::optimization::test::SGDTestFunction::NumFunctions () const` `[inline]`

Return 3 (the number of functions).

Definition at line 34 of file `test_function.hpp`.

The documentation for this class was generated from the following file:

- `src/mlpack/core/optimizers/sgd/test_function.hpp`

22.115 mpack::optimization::test::WoodFunction Class Reference

The Wood function, defined by $f(x) = f_1(x) + f_2(x) + f_3(x) + f_4(x) + f_5(x) + f_6(x)$ $f_1(x) = 100 (x_2 - x_1^2)^2$ $f_2(x) = (1 - x_1)^2$ $f_3(x) = 90 (x_4 - x_3^2)^2$ $f_4(x) = (1 - x_3)^2$ $f_5(x) = 10 (x_2 + x_4 - 2)^2$ $f_6(x) = (1 / 10) (x_2 - x_4)^2$ $x_0 = [-3, -1, -3, -1]$.

Public Member Functions

- **WoodFunction** ()
- double **Evaluate** (const arma::mat &coordinates)
- const arma::mat & **GetInitialPoint** () const
- void **Gradient** (const arma::mat &coordinates, arma::mat &gradient)

Private Attributes

- arma::mat **initialPoint**

22.115.1 Detailed Description

The Wood function, defined by $f(x) = f_1(x) + f_2(x) + f_3(x) + f_4(x) + f_5(x) + f_6(x)$ $f_1(x) = 100 (x_2 - x_1^2)^2$ $f_2(x) = (1 - x_1)^2$ $f_3(x) = 90 (x_4 - x_3^2)^2$ $f_4(x) = (1 - x_3)^2$ $f_5(x) = 10 (x_2 + x_4 - 2)^2$ $f_6(x) = (1 / 10) (x_2 - x_4)^2$ $x_0 = [-3, -1, -3, -1]$.

This should optimize to $f(x) = 0$, at $x = [1, 1, 1, 1]$.

"A comparative study of nonlinear programming codes." A.R. Colville. 1968. Rep. 320-2949, IBM N.Y. Scientific Center.

Definition at line 85 of file `test_functions.hpp`.

22.115.2 Constructor & Destructor Documentation

22.115.2.1 `mpack::optimization::test::WoodFunction::WoodFunction ()`

22.115.3 Member Function Documentation

22.115.3.1 `double mpack::optimization::test::WoodFunction::Evaluate (const arma::mat & coordinates)`

22.115.3.2 `const arma::mat& mpack::optimization::test::WoodFunction::GetInitialPoint () const`

22.115.3.3 `void mpack::optimization::test::WoodFunction::Gradient (const arma::mat & coordinates, arma::mat & gradient)`

22.115.4 Member Data Documentation

22.115.4.1 arma::mat mlpack::optimization::test::WoodFunction::initialPoint [private]

Definition at line 96 of file test_functions.hpp.

The documentation for this class was generated from the following file:

- src/mlpack/core/optimizers/lbfgs/test_functions.hpp

22.116 mlpack::ParamData Struct Reference

Aids in the extensibility of **CLI** (p. 184) by focusing potential changes into one structure.

Public Attributes

- std::string **desc**
Description of this parameter, if any.
- bool **isFlag**
True if the wasPassed value should not be ignored.
- std::string **name**
Name of this parameter.
- std::string **tname**
Type information of this parameter.
- boost::any **value**
The actual value of this parameter.
- bool **wasPassed**
True if this parameter was passed in via command line or file.

22.116.1 Detailed Description

Aids in the extensibility of **CLI** (p. 184) by focusing potential changes into one structure.

Definition at line 382 of file cli.hpp.

22.116.2 Member Data Documentation

22.116.2.1 std::string mlpack::ParamData::desc

Description of this parameter, if any.

Definition at line 387 of file cli.hpp.

22.116.2.2 bool mlpack::ParamData::isFlag

True if the wasPassed value should not be ignored.

Definition at line 395 of file cli.hpp.

22.116.2.3 `std::string mlpack::ParamData::name`

Name of this parameter.

Definition at line 385 of file cli.hpp.

22.116.2.4 `std::string mlpack::ParamData::tname`

Type information of this parameter.

Definition at line 389 of file cli.hpp.

22.116.2.5 `boost::any mlpack::ParamData::value`

The actual value of this parameter.

Definition at line 391 of file cli.hpp.

22.116.2.6 `bool mlpack::ParamData::wasPassed`

True if this parameter was passed in via command line or file.

Definition at line 393 of file cli.hpp.

The documentation for this struct was generated from the following file:

- `src/mlpack/core/util/cli.hpp`

22.117 `mlpack::pca::PCA` Class Reference

This class implements principal components analysis (**PCA** (p. 498)).

Public Member Functions

- **PCA** (const bool **scaleData**=false)
*Create the **PCA** (p. 498) object, specifying if the data should be scaled in each dimension by standard deviation when **PCA** (p. 498) is performed.*
- void **Apply** (const arma::mat &data, arma::mat &transformedData, arma::vec &eigval, arma::mat &eigvec) const
Apply Principal Component Analysis to the provided data set.
- void **Apply** (const arma::mat &data, arma::mat &transformedData, arma::vec &eigVal) const
Apply Principal Component Analysis to the provided data set.
- double **Apply** (arma::mat &data, const size_t newDimension) const
*Use **PCA** (p. 498) for dimensionality reduction on the given dataset.*
- double **Apply** (arma::mat &data, const int newDimension) const
This overload is here to make sure int gets casted right to size_t.
- double **Apply** (arma::mat &data, const double varRetained) const
*Use **PCA** (p. 498) for dimensionality reduction on the given dataset.*
- bool **ScaleData** () const
*Get whether or not this **PCA** (p. 498) object will scale (by standard deviation) the data when **PCA** (p. 498) is performed.*
- bool & **ScaleData** ()

Modify whether or not this **PCA** (p. 498) object will scale (by standard deviation) the data when **PCA** (p. 498) is performed.

- `std::string ToString () const`

Private Attributes

- `bool scaleData`

Whether or not the data will be scaled by standard deviation when **PCA** (p. 498) is performed.

22.117.1 Detailed Description

This class implements principal components analysis (**PCA** (p. 498)).

This is a common, widely-used technique that is often used for either dimensionality reduction or transforming data into a better basis. Further information on **PCA** (p. 498) can be found in almost any statistics or machine learning textbook, and all over the internet.

Definition at line 30 of file `pca.hpp`.

22.117.2 Constructor & Destructor Documentation

22.117.2.1 `mlpack::pca::PCA::PCA (const bool scaleData = false)`

Create the **PCA** (p. 498) object, specifying if the data should be scaled in each dimension by standard deviation when **PCA** (p. 498) is performed.

Parameters

<i>scaleData</i>	Whether or not to scale the data.
------------------	-----------------------------------

22.117.3 Member Function Documentation

22.117.3.1 `void mlpack::pca::PCA::Apply (const arma::mat & data, arma::mat & transformedData, arma::vec & eigval, arma::mat & eigvec) const`

Apply Principal Component Analysis to the provided data set.

It is safe to pass the same matrix reference for both data and transformedData.

Parameters

<i>data</i>	Data matrix.
<i>transformedData</i>	Matrix to put results of PCA (p. 498) into.
<i>eigval</i>	Vector to put eigenvalues into.
<i>eigvec</i>	Matrix to put eigenvectors (loadings) into.

Referenced by `Apply()`.

22.117.3.2 `void mlpack::pca::PCA::Apply (const arma::mat & data, arma::mat & transformedData, arma::vec & eigVal) const`

Apply Principal Component Analysis to the provided data set.

It is safe to pass the same matrix reference for both data and transformedData.

Parameters

<i>data</i>	Data matrix.
<i>transformedData</i>	Matrix to store results of PCA (p. 498) in.
<i>eigval</i>	Vector to put eigenvalues into.

22.117.3.3 `double mlpack::pca::PCA::Apply (arma::mat & data, const size_t newDimension) const`

Use **PCA** (p. 498) for dimensionality reduction on the given dataset.

This will save the newDimension largest principal components of the data and remove the rest. The parameter returned is the amount of variance of the data that is retained; this is a value between 0 and 1. For instance, a value of 0.9 indicates that 90% of the variance present in the data was retained.

Parameters

<i>data</i>	Data matrix.
<i>newDimension</i>	New dimension of the data.

Returns

Amount of the variance of the data retained (between 0 and 1).

22.117.3.4 `double mlpack::pca::PCA::Apply (arma::mat & data, const int newDimension) const` `[inline]`

This overload is here to make sure int gets casted right to size_t.

Definition at line 81 of file pca.hpp.

References Apply().

22.117.3.5 `double mlpack::pca::PCA::Apply (arma::mat & data, const double varRetained) const`

Use **PCA** (p. 498) for dimensionality reduction on the given dataset.

This will save as many dimensions as necessary to retain at least the given amount of variance (specified by parameter varRetained). The amount should be between 0 and 1; if the amount is 0, then only 1 dimension will be retained. If the amount is 1, then all dimensions will be retained.

The method returns the actual amount of variance retained, which will always be greater than or equal to the varRetained parameter.

Parameters

<i>data</i>	Data matrix.
<i>varRetained</i>	Lower bound on amount of variance to retain; should be between 0 and 1.

Returns

Actual amount of variance retained (between 0 and 1).

22.117.3.6 `bool mlpack::pca::PCA::ScaleData () const` `[inline]`

Get whether or not this **PCA** (p. 498) object will scale (by standard deviation) the data when **PCA** (p. 498) is performed.

Definition at line 105 of file `pca.hpp`.

References `scaleData`.

22.117.3.7 `bool& mlpack::pca::PCA::ScaleData () [inline]`

Modify whether or not this **PCA** (p. 498) object will scale (by standard deviation) the data when **PCA** (p. 498) is performed.

Definition at line 108 of file `pca.hpp`.

References `scaleData`.

22.117.3.8 `std::string mlpack::pca::PCA::ToString () const`

22.117.4 Member Data Documentation

22.117.4.1 `bool mlpack::pca::PCA::scaleData [private]`

Whether or not the data will be scaled by standard deviation when **PCA** (p. 498) is performed.

Definition at line 116 of file `pca.hpp`.

Referenced by `ScaleData()`.

The documentation for this class was generated from the following file:

- `src/mlpack/methods/pca/pca.hpp`

22.118 `mlpack::perceptron::Perceptron< LearnPolicy, WeightInitializationPolicy, MatType >` Class Template Reference

This class implements a simple perceptron (i.e., a single layer neural network).

Public Member Functions

- **Perceptron** (const MatType &data, const arma::Row< size_t > &labels, int iterations)
Constructor - constructs the perceptron by building the weightVectors matrix, which is later used in Classification.
- **Perceptron** (const **Perceptron**<> &other, MatType &data, const arma::rowvec &D, const arma::Row< size_t > &labels)
Alternate constructor which copies parameters from an already initiated perceptron.
- void **Classify** (const MatType &test, arma::Row< size_t > &predictedLabels)
Classification function.

Private Member Functions

- void **Train** (const arma::rowvec &D)
Training Function.

Private Attributes

- `arma::Row< size_t > classLabels`
Stores the class labels for the input data.
- `size_t iter`
To store the number of iterations.
- `arma::mat trainData`
Stores the training data to be used later on in `UpdateWeights`.
- `arma::mat weightVectors`
Stores the weight vectors for each of the input class labels.

22.118.1 Detailed Description

`template<typename LearnPolicy = SimpleWeightUpdate, typename WeightInitializationPolicy = ZeroInitialization, typename MatType = arma::mat> class mlpack::perceptron::Perceptron< LearnPolicy, WeightInitializationPolicy, MatType >`

This class implements a simple perceptron (i.e., a single layer neural network).

It converges if the supplied training dataset is linearly separable.

Template Parameters

<i>LearnPolicy</i>	Options of SimpleWeightUpdate (p. 505) and GradientDescent .
<i>WeightInitializationPolicy</i>	Option of ZeroInitialization (p. 505) and RandomInitialization (p. 504).

Definition at line 38 of file `perceptron.hpp`.

22.118.2 Constructor & Destructor Documentation

22.118.2.1 `template<typename LearnPolicy = SimpleWeightUpdate, typename WeightInitializationPolicy = ZeroInitialization, typename MatType = arma::mat> mlpack::perceptron::Perceptron< LearnPolicy, WeightInitializationPolicy, MatType >::Perceptron (const MatType & data, const arma::Row< size_t > & labels, int iterations)`

Constructor - constructs the perceptron by building the `weightVectors` matrix, which is later used in `Classification`.

It adds a bias input vector of 1 to the input data to take care of the bias weights.

Parameters

<i>data</i>	Input, training data.
<i>labels</i>	Labels of dataset.
<i>iterations</i>	Maximum number of iterations for the perceptron learning algorithm.

22.118.2.2 `template<typename LearnPolicy = SimpleWeightUpdate, typename WeightInitializationPolicy = ZeroInitialization, typename MatType = arma::mat> mlpack::perceptron::Perceptron< LearnPolicy, WeightInitializationPolicy, MatType >::Perceptron (const Perceptron<> & other, MatType & data, const arma::rowvec & D, const arma::Row< size_t > & labels)`

Alternate constructor which copies parameters from an already initiated perceptron.

Parameters

<i>other</i>	The other initiated Perceptron (p. 501) object from which we copy the values from.
<i>data</i>	The data on which to train this Perceptron (p. 501) object on.
<i>D</i>	Weight vector to use while training. For boosting purposes.
<i>labels</i>	The labels of data.

22.118.3 Member Function Documentation

22.118.3.1 `template<typename LearnPolicy = SimpleWeightUpdate, typename WeightInitializationPolicy = ZeroInitialization, typename MatType = arma::mat> void mlpack::perceptron::Perceptron<LearnPolicy, WeightInitializationPolicy, MatType>::Classify (const MatType & test, arma::Row< size_t > & predictedLabels)`

Classification function.

After training, use the `weightVectors` matrix to classify test, and put the predicted classes in `predictedLabels`.

Parameters

<i>test</i>	Testing data or data to classify.
<i>predictedLabels</i>	Vector to store the predicted classes after classifying test.

22.118.3.2 `template<typename LearnPolicy = SimpleWeightUpdate, typename WeightInitializationPolicy = ZeroInitialization, typename MatType = arma::mat> void mlpack::perceptron::Perceptron<LearnPolicy, WeightInitializationPolicy, MatType>::Train (const arma::rowvec & D) [private]`

Training Function.

It trains on `trainData` using the cost matrix `D`

Parameters

<i>D</i>	Cost matrix. Stores the cost of mispredicting instances
----------	---

22.118.4 Member Data Documentation

22.118.4.1 `template<typename LearnPolicy = SimpleWeightUpdate, typename WeightInitializationPolicy = ZeroInitialization, typename MatType = arma::mat> arma::Row<size_t> mlpack::perceptron::Perceptron<LearnPolicy, WeightInitializationPolicy, MatType>::classLabels [private]`

Stores the class labels for the input data.

Definition at line 80 of file `perceptron.hpp`.

22.118.4.2 `template<typename LearnPolicy = SimpleWeightUpdate, typename WeightInitializationPolicy = ZeroInitialization, typename MatType = arma::mat> size_t mlpack::perceptron::Perceptron<LearnPolicy, WeightInitializationPolicy, MatType>::iter [private]`

To store the number of iterations.

Definition at line 77 of file `perceptron.hpp`.

```
22.118.4.3  template<typename LearnPolicy = SimpleWeightUpdate, typename WeightInitializationPolicy = ZeroInitialization,
            typename MatType = arma::mat> arma::mat mlpack::perceptron::Perceptron< LearnPolicy,
            WeightInitializationPolicy, MatType >::trainData  [private]
```

Stores the training data to be used later on in UpdateWeights.

Definition at line 86 of file perceptron.hpp.

```
22.118.4.4  template<typename LearnPolicy = SimpleWeightUpdate, typename WeightInitializationPolicy = ZeroInitialization,
            typename MatType = arma::mat> arma::mat mlpack::perceptron::Perceptron< LearnPolicy,
            WeightInitializationPolicy, MatType >::weightVectors  [private]
```

Stores the weight vectors for each of the input class labels.

Definition at line 83 of file perceptron.hpp.

The documentation for this class was generated from the following file:

- src/mlpack/methods/perceptron/**perceptron.hpp**

22.119 mlpack::perceptron::RandomInitialization Class Reference

This class is used to initialize weights for the weightVectors matrix in a random manner.

Public Member Functions

- **RandomInitialization** ()

Static Public Member Functions

- static void **Initialize** (arma::mat &W, const size_t row, const size_t col)

22.119.1 Detailed Description

This class is used to initialize weights for the weightVectors matrix in a random manner.

Definition at line 26 of file random_init.hpp.

22.119.2 Constructor & Destructor Documentation

```
22.119.2.1  mlpack::perceptron::RandomInitialization::RandomInitialization ( )  [inline]
```

Definition at line 29 of file random_init.hpp.

22.119.3 Member Function Documentation

```
22.119.3.1  static void mlpack::perceptron::RandomInitialization::Initialize ( arma::mat & W, const size_t row, const size_t col )
            [inline], [static]
```

Definition at line 31 of file random_init.hpp.

The documentation for this class was generated from the following file:

- src/mlpack/methods/perceptron/initialization_methods/**random_init.hpp**

22.120 mlpack::perceptron::SimpleWeightUpdate Class Reference

Public Member Functions

- void **UpdateWeights** (const arma::mat &trainData, arma::mat &weightVectors, const size_t labelIndex, const size_t vectorIndex, const size_t rowIndex, const arma::rowvec &D)

This function is called to update the weightVectors matrix.

22.120.1 Detailed Description

Definition at line 32 of file simple_weight_update.hpp.

22.120.2 Member Function Documentation

22.120.2.1 void mlpack::perceptron::SimpleWeightUpdate::UpdateWeights (const arma::mat & *trainData*, arma::mat & *weightVectors*, const size_t *labelIndex*, const size_t *vectorIndex*, const size_t *rowIndex*, const arma::rowvec & *D*)
[inline]

This function is called to update the weightVectors matrix.

It decreases the weights of the incorrectly classified class while increasing the weight of the correct class it should have been classified to.

Parameters

<i>trainData</i>	The training dataset.
<i>weightVectors</i>	Matrix of weight vectors.
<i>rowIndex</i>	Index of the row which has been incorrectly predicted.
<i>labelIndex</i>	Index of the vector in trainData.
<i>vectorIndex</i>	Index of the class which should have been predicted.
<i>D</i>	Cost of mispredicting the labelIndex instance.

Definition at line 47 of file simple_weight_update.hpp.

The documentation for this class was generated from the following file:

- src/mlpack/methods/perceptron/learning_policies/**simple_weight_update.hpp**

22.121 mlpack::perceptron::ZeroInitialization Class Reference

This class is used to initialize the matrix weightVectors to zero.

Public Member Functions

- **ZeroInitialization** ()

Static Public Member Functions

- static void **Initialize** (arma::mat &W, const size_t row, const size_t col)

22.121.1 Detailed Description

This class is used to initialize the matrix weightVectors to zero.

Definition at line 25 of file zero_init.hpp.

22.121.2 Constructor & Destructor Documentation

22.121.2.1 `mlpack::perceptron::ZeroInitialization::ZeroInitialization ()` `[inline]`

Definition at line 28 of file zero_init.hpp.

22.121.3 Member Function Documentation

22.121.3.1 `static void mlpack::perceptron::ZeroInitialization::Initialize (arma::mat & W, const size_t row, const size_t col)`
`[inline], [static]`

Definition at line 30 of file zero_init.hpp.

The documentation for this class was generated from the following file:

- src/mlpack/methods/perceptron/initialization_methods/**zero_init.hpp**

22.122 mlpack::radical::Radical Class Reference

An implementation of RADICAL, an algorithm for independent component analysis (ICA).

Public Member Functions

- **Radical** (const double **noiseStdDev**=0.175, const size_t **replicates**=30, const size_t **angles**=150, const size_t **sweeps**=0, const size_t **m**=0)
Set the parameters to RADICAL.
- size_t **Angles** () const
Get the number of angles considered during brute-force search.
- size_t & **Angles** ()
Modify the number of angles considered during brute-force search.
- void **CopyAndPerturb** (arma::mat &xNew, const arma::mat &x) const
*Make replicates of each data point (the number of replicates is set in either the constructor or with **Replicates()** (p. 510)) and perturb data with Gaussian noise with standard deviation noiseStdDev.*
- void **DoRadical** (const arma::mat &matX, arma::mat &matY, arma::mat &matW)
Run RADICAL.
- double **DoRadical2D** (const arma::mat &matX)
Two-dimensional version of RADICAL.
- double **NoiseStdDev** () const

Get the standard deviation of the additive Gaussian noise.

- double & **NoiseStdDev** ()

Modify the standard deviation of the additive Gaussian noise.

- size_t **Replicates** () const

Get the number of Gaussian-perturbed replicates used per point.

- size_t & **Replicates** ()

Modify the number of Gaussian-perturbed replicates used per point.

- size_t **Sweeps** () const

Get the number of sweeps.

- size_t & **Sweeps** ()

Modify the number of sweeps.

- std::string **ToString** () const

- double **Vasicek** (arma::vec &x) const

Vasicek's m-spacing estimator of entropy, with overlap modification from (Learned-Miller and Fisher, 2003).

Private Attributes

- size_t **angles**

Number of angles to consider in brute-force search during Radical2D.

- arma::mat **candidate**

Internal matrix, held as member variable to prevent memory reallocations.

- size_t **m**

Value of m to use for Vasicek's m-spacing estimator of entropy.

- double **noiseStdDev**

Standard deviation of the Gaussian noise added to the replicates of the data points during Radical2D.

- arma::mat **perturbed**

Internal matrix, held as member variable to prevent memory reallocations.

- size_t **replicates**

Number of Gaussian-perturbed replicates to use (per point) in Radical2D.

- size_t **sweeps**

Number of sweeps; each sweep calls Radical2D once for each pair of dimensions.

22.122.1 Detailed Description

An implementation of RADICAL, an algorithm for independent component analysis (ICA).

Let X be a matrix where each column is a point and each row a dimension. The goal is to find a square unmixing matrix W such that $Y = W X$ and the rows of Y are independent components.

For more details, see the following paper:

```
@article{learned2003ica,
  title = {ICA Using Spacings Estimates of Entropy},
  author = {Learned-Miller, E.G. and Fisher III, J.W.},
  journal = {Journal of Machine Learning Research},
  volume = {4},
  pages = {1271--1295},
  year = {2003}
}
```

Definition at line 45 of file radical.hpp.

22.122.2 Constructor & Destructor Documentation

22.122.2.1 `mlpack::radical::Radical (const double noiseStdDev = 0.175, const size_t replicates = 30, const size_t angles = 150, const size_t sweeps = 0, const size_t m = 0)`

Set the parameters to RADICAL.

Parameters

<i>noiseStdDev</i>	Standard deviation of the Gaussian noise added to the replicates of the data points during Radical2D
<i>replicates</i>	Number of Gaussian-perturbed replicates to use (per point) in Radical2D
<i>angles</i>	Number of angles to consider in brute-force search during Radical2D
<i>sweeps</i>	Number of sweeps. Each sweep calls Radical2D once for each pair of dimensions
<i>m</i>	The variable m from Vasicek's m-spacing estimator of entropy.

22.122.3 Member Function Documentation

22.122.3.1 `size_t mlpack::radical::Radical::Angles () const` `[inline]`

Get the number of angles considered during brute-force search.

Definition at line 107 of file radical.hpp.

References angles.

22.122.3.2 `size_t& mlpack::radical::Radical::Angles ()` `[inline]`

Modify the number of angles considered during brute-force search.

Definition at line 109 of file radical.hpp.

References angles.

22.122.3.3 `void mlpack::radical::Radical::CopyAndPerturb (arma::mat & xNew, const arma::mat & x) const`

Make replicates of each data point (the number of replicates is set in either the constructor or with **Replicates()** (p. 510)) and perturb data with Gaussian noise with standard deviation noiseStdDev.

22.122.3.4 `void mlpack::radical::Radical::DoRadical (const arma::mat & matX, arma::mat & matY, arma::mat & matW)`

Run RADICAL.

Parameters

<i>matX</i>	Input data into the algorithm - a matrix where each column is a point and each row is a dimension.
<i>matY</i>	Estimated independent components - a matrix where each column is a point and each row is an estimated independent component.
<i>matW</i>	Estimated unmixing matrix, where $\text{matY} = \text{matW} * \text{matX}$.

22.122.3.5 `double mlpack::radical::Radical::DoRadical2D (const arma::mat & matX)`

Two-dimensional version of RADICAL.

22.122.3.6 `double mlpack::radical::Radical::NoiseStdDev () const` `[inline]`

Get the standard deviation of the additive Gaussian noise.

Definition at line 97 of file radical.hpp.

References noiseStdDev.

22.122.3.7 `double& mlpack::radical::Radical::NoiseStdDev () [inline]`

Modify the standard deviation of the additive Gaussian noise.

Definition at line 99 of file radical.hpp.

References noiseStdDev.

22.122.3.8 `size_t mlpack::radical::Radical::Replicates () const [inline]`

Get the number of Gaussian-perturbed replicates used per point.

Definition at line 102 of file radical.hpp.

References replicates.

22.122.3.9 `size_t& mlpack::radical::Radical::Replicates () [inline]`

Modify the number of Gaussian-perturbed replicates used per point.

Definition at line 104 of file radical.hpp.

References replicates.

22.122.3.10 `size_t mlpack::radical::Radical::Sweeps () const [inline]`

Get the number of sweeps.

Definition at line 112 of file radical.hpp.

References sweeps.

22.122.3.11 `size_t& mlpack::radical::Radical::Sweeps () [inline]`

Modify the number of sweeps.

Definition at line 114 of file radical.hpp.

References sweeps.

22.122.3.12 `std::string mlpack::radical::Radical::ToString () const`

22.122.3.13 `double mlpack::radical::Radical::Vasicek (arma::vec & x) const`

Vasicek's m-spacing estimator of entropy, with overlap modification from (Learned-Miller and Fisher, 2003).

Parameters

<code>x</code>	Empirical sample (one-dimensional) over which to estimate entropy.
----------------	--

22.122.4 Member Data Documentation

22.122.4.1 `size_t mlpack::radical::Radical::angles` `[private]`

Number of angles to consider in brute-force search during Radical2D.

Definition at line 128 of file radical.hpp.

Referenced by Angles().

22.122.4.2 `arma::mat mlpack::radical::Radical::candidate` `[private]`

Internal matrix, held as member variable to prevent memory reallocations.

Definition at line 140 of file radical.hpp.

22.122.4.3 `size_t mlpack::radical::Radical::m` `[private]`

Value of m to use for Vasicek's m-spacing estimator of entropy.

Definition at line 135 of file radical.hpp.

22.122.4.4 `double mlpack::radical::Radical::noiseStdDev` `[private]`

Standard deviation of the Gaussian noise added to the replicates of the data points during Radical2D.

Definition at line 122 of file radical.hpp.

Referenced by NoiseStdDev().

22.122.4.5 `arma::mat mlpack::radical::Radical::perturbed` `[private]`

Internal matrix, held as member variable to prevent memory reallocations.

Definition at line 138 of file radical.hpp.

22.122.4.6 `size_t mlpack::radical::Radical::replicates` `[private]`

Number of Gaussian-perturbed replicates to use (per point) in Radical2D.

Definition at line 125 of file radical.hpp.

Referenced by Replicates().

22.122.4.7 `size_t mlpack::radical::Radical::sweeps` `[private]`

Number of sweeps; each sweep calls Radical2D once for each pair of dimensions.

Definition at line 132 of file radical.hpp.

Referenced by Sweeps().

The documentation for this class was generated from the following file:

- `src/mlpack/methods/radical/radical.hpp`

22.123 mlpack::range::RangeSearch< MetricType, TreeType > Class Template Reference

The **RangeSearch** (p. 512) class is a template class for performing range searches.

Public Member Functions

- **RangeSearch** (const typename TreeType::Mat &**referenceSet**, const typename TreeType::Mat &**querySet**, const bool **naive**=false, const bool **singleMode**=false, const MetricType **metric**=MetricType())
*Initialize the **RangeSearch** (p. 512) object with a different reference set and a query set.*
- **RangeSearch** (const typename TreeType::Mat &**referenceSet**, const bool **naive**=false, const bool **singleMode**=false, const MetricType **metric**=MetricType())
*Initialize the **RangeSearch** (p. 512) object with only a reference set, which will also be used as a query set.*
- **RangeSearch** (TreeType ***referenceTree**, TreeType ***queryTree**, const typename TreeType::Mat &**referenceSet**, const typename TreeType::Mat &**querySet**, const bool **singleMode**=false, const MetricType **metric**=MetricType())
*Initialize the **RangeSearch** (p. 512) object with the given datasets and pre-constructed trees.*
- **RangeSearch** (TreeType ***referenceTree**, const typename TreeType::Mat &**referenceSet**, const bool **singleMode**=false, const MetricType **metric**=MetricType())
*Initialize the **RangeSearch** (p. 512) object with the given reference dataset and pre-constructed tree.*
- **~RangeSearch** ()
*Destroy the **RangeSearch** (p. 512) object.*
- void **Search** (const math::Range &range, std::vector< std::vector< size_t > > &neighbors, std::vector< std::vector< double > > &distances)
Search for all points in the given range, returning the results in the neighbors and distances objects.
- std::string **ToString** () const

Private Attributes

- bool **hasQuerySet**
If true, a query set was passed; if false, the query set is the reference set.
- MetricType **metric**
Instantiated distance metric.
- bool **naive**
If true, $O(n^2)$ naive computation is used.
- size_t **numPrunes**
The number of pruned nodes during computation.
- std::vector< size_t > **oldFromNewQueries**
Mappings to old query indices (used when this object builds trees).
- std::vector< size_t > **oldFromNewReferences**
Mappings to old reference indices (used when this object builds trees).
- TreeType::Mat **queryCopy**
Copy of query matrix; used when a tree is built internally.
- const TreeType::Mat & **querySet**
Query set (data should be accessed using this).
- TreeType * **queryTree**
Query tree (may be NULL).
- TreeType::Mat **referenceCopy**
Copy of reference matrix; used when a tree is built internally.

- const TreeType::Mat & **referenceSet**
Reference set (data should be accessed using this).
- TreeType * **referenceTree**
Reference tree.
- bool **singleMode**
If true, single-tree computation is used.
- bool **treeOwner**
If true, this object is responsible for deleting the trees.

22.123.1 Detailed Description

```
template<typename MetricType = mlpack::metric::EuclideanDistance, typename TreeType = tree::BinarySpaceTree<bound::HRectBound<2>, RangeSearchStat>> class mlpack::range::RangeSearch< MetricType, TreeType >
```

The **RangeSearch** (p. 512) class is a template class for performing range searches.

It is implemented in the style of a generalized tree-independent dual-tree algorithm; for more details on the actual algorithm, see the **RangeSearchRules** (p. 518) class.

Definition at line 35 of file range_search.hpp.

22.123.2 Constructor & Destructor Documentation

```
22.123.2.1 template<typename MetricType = mlpack::metric::EuclideanDistance, typename TreeType = tree::BinarySpaceTree<bound::HRectBound<2>, RangeSearchStat>> mlpack::range::RangeSearch< MetricType, TreeType >::RangeSearch ( const typename TreeType::Mat & referenceSet, const typename TreeType::Mat & querySet, const bool naive = false, const bool singleMode = false, const MetricType metric = MetricType() )
```

Initialize the **RangeSearch** (p. 512) object with a different reference set and a query set.

Optionally, perform the computation in naive mode or single-tree mode, and set the leaf size used for tree-building. Additionally, an instantiated metric can be given, for cases where the distance metric holds data.

This method will copy the matrices to internal copies, which are rearranged during tree-building. You can avoid this extra copy by pre-constructing the trees and passing them using a different constructor.

Parameters

<i>referenceSet</i>	Reference dataset.
<i>querySet</i>	Query dataset.
<i>naive</i>	Whether the computation should be done in $O(n^2)$ naive mode.
<i>singleMode</i>	Whether single-tree computation should be used (as opposed to dual-tree computation).
<i>leafSize</i>	The leaf size to be used during tree construction.
<i>metric</i>	Instantiated distance metric.

```
22.123.2.2 template<typename MetricType = mlpack::metric::EuclideanDistance, typename TreeType = tree::BinarySpaceTree<bound::HRectBound<2>, RangeSearchStat>> mlpack::range::RangeSearch< MetricType, TreeType >::RangeSearch ( const typename TreeType::Mat & referenceSet, const bool naive = false, const bool singleMode = false, const MetricType metric = MetricType() )
```

Initialize the **RangeSearch** (p. 512) object with only a reference set, which will also be used as a query set.

Optionally, perform the computation in naive mode or single-tree mode, and set the leaf size used for tree-building. Additionally an instantiated metric can be given, for cases where the distance metric holds data.

This method will copy the reference matrix to an internal copy, which is rearranged during tree-building. You can avoid this extra copy by pre-constructing the reference tree and passing it using a different constructor.

Parameters

<i>referenceSet</i>	Reference dataset.
<i>naive</i>	Whether the computation should be done in $O(n^2)$ naive mode.
<i>singleMode</i>	Whether single-tree computation should be used (as opposed to dual-tree computation).
<i>leafSize</i>	The leaf size to be used during tree construction.
<i>metric</i>	Instantiated distance metric.

```
22.123.2.3 template<typename MetricType = mlpack::metric::EuclideanDistance, typename TreeType =
tree::BinarySpaceTree<bound::HRectBound<2>, RangeSearchStat>> mlpack::range::RangeSearch<
MetricType, TreeType >::RangeSearch ( TreeType * referenceTree, TreeType * queryTree, const typename
TreeType::Mat & referenceSet, const typename TreeType::Mat & querySet, const bool singleMode = false, const
MetricType metric = MetricType() )
```

Initialize the **RangeSearch** (p. 512) object with the given datasets and pre-constructed trees.

It is assumed that the points in referenceSet and querySet correspond to the points in referenceTree and queryTree, respectively. Optionally, choose to use single-tree mode. Naive mode is not available as an option for this constructor; instead, to run naive computation, construct a tree with all the points in one leaf (i.e. leafSize = number of points). Additionally, an instantiated distance metric can be given, for cases where the distance metric holds data.

There is no copying of the data matrices in this constructor (because tree-building is not necessary), so this is the constructor to use when copies absolutely must be avoided.

Note

Because tree-building (at least with BinarySpaceTree) modifies the ordering of a matrix, be sure you pass the modified matrix to this object! In addition, mapping the points of the matrix back to their original indices is not done when this constructor is used.

Parameters

<i>referenceTree</i>	Pre-built tree for reference points.
<i>queryTree</i>	Pre-built tree for query points.
<i>referenceSet</i>	Set of reference points corresponding to referenceTree.
<i>querySet</i>	Set of query points corresponding to queryTree.
<i>singleMode</i>	Whether single-tree computation should be used (as opposed to dual-tree computation).
<i>metric</i>	Instantiated distance metric.

```
22.123.2.4 template<typename MetricType = mlpack::metric::EuclideanDistance, typename TreeType =
tree::BinarySpaceTree<bound::HRectBound<2>, RangeSearchStat>> mlpack::range::RangeSearch<
MetricType, TreeType >::RangeSearch ( TreeType * referenceTree, const typename TreeType::Mat & referenceSet,
const bool singleMode = false, const MetricType metric = MetricType() )
```

Initialize the **RangeSearch** (p. 512) object with the given reference dataset and pre-constructed tree.

It is assumed that the points in referenceSet correspond to the points in referenceTree. Optionally, choose to use single-tree mode. Naive mode is not available as an option for this constructor; instead, to run naive computation, construct a

tree with all the points in one leaf (i.e. leafSize = number of points). Additionally, an instantiated distance metric can be given, for the case where the distance metric holds data.

There is no copying of the data matrices in this constructor (because tree-building is not necessary), so this is the constructor to use when copies absolutely must be avoided.

Note

Because tree-building (at least with BinarySpaceTree) modifies the ordering of a matrix, be sure you pass the modified matrix to this object! In addition, mapping the points of the matrix back to their original indices is not done when this constructor is used.

Parameters

<i>referenceTree</i>	Pre-built tree for reference points.
<i>referenceSet</i>	Set of reference points corresponding to referenceTree.
<i>singleMode</i>	Whether single-tree computation should be used (as opposed to dual-tree computation).
<i>metric</i>	Instantiated distance metric.

```
22.123.2.5  template<typename MetricType = mlpack::metric::EuclideanDistance, typename TreeType =
             tree::BinarySpaceTree<bound::HRectBound<2>, RangeSearchStat>> mlpack::range::RangeSearch<
             MetricType, TreeType >::~~RangeSearch ( )
```

Destroy the **RangeSearch** (p. 512) object.

If trees were created, they will be deleted.

22.123.3 Member Function Documentation

```
22.123.3.1  template<typename MetricType = mlpack::metric::EuclideanDistance, typename TreeType =
             tree::BinarySpaceTree<bound::HRectBound<2>, RangeSearchStat>> void mlpack::range::RangeSearch<
             MetricType, TreeType >::Search ( const math::Range & range, std::vector< std::vector< size_t > > & neighbors,
             std::vector< std::vector< double > > & distances )
```

Search for all points in the given range, returning the results in the neighbors and distances objects.

Each entry in the external vector corresponds to a query point. Each of these entries holds a vector which contains the indices and distances of the reference points falling into the given range.

That is:

- neighbors.size() and distances.size() both equal the number of query points.
- neighbors[i] contains the indices of all the points in the reference set which have distances inside the given range to query point i.
- distances[i] contains all of the distances corresponding to the indices contained in neighbors[i].
- neighbors[i] and distances[i] are not sorted in any particular order.

Parameters

<i>range</i>	Range of distances in which to search.
<i>neighbors</i>	Object which will hold the list of neighbors for each point which fell into the given range, for each query point.
<i>distances</i>	Object which will hold the list of distances for each point which fell into the given range, for each query point.

22.123.3.2 `template<typename MetricType = mlpack::metric::EuclideanDistance, typename TreeType = tree::BinarySpaceTree<bound::HRectBound<2>, RangeSearchStat>> std::string mlpack::range::RangeSearch< MetricType, TreeType >::ToString () const`

22.123.4 Member Data Documentation

22.123.4.1 `template<typename MetricType = mlpack::metric::EuclideanDistance, typename TreeType = tree::BinarySpaceTree<bound::HRectBound<2>, RangeSearchStat>> bool mlpack::range::RangeSearch< MetricType, TreeType >::hasQuerySet [private]`

If true, a query set was passed; if false, the query set is the reference set.

Definition at line 219 of file range_search.hpp.

22.123.4.2 `template<typename MetricType = mlpack::metric::EuclideanDistance, typename TreeType = tree::BinarySpaceTree<bound::HRectBound<2>, RangeSearchStat>> MetricType mlpack::range::RangeSearch< MetricType, TreeType >::metric [private]`

Instantiated distance metric.

Definition at line 227 of file range_search.hpp.

22.123.4.3 `template<typename MetricType = mlpack::metric::EuclideanDistance, typename TreeType = tree::BinarySpaceTree<bound::HRectBound<2>, RangeSearchStat>> bool mlpack::range::RangeSearch< MetricType, TreeType >::naive [private]`

If true, $O(n^2)$ naive computation is used.

Definition at line 222 of file range_search.hpp.

22.123.4.4 `template<typename MetricType = mlpack::metric::EuclideanDistance, typename TreeType = tree::BinarySpaceTree<bound::HRectBound<2>, RangeSearchStat>> size_t mlpack::range::RangeSearch< MetricType, TreeType >::numPrunes [private]`

The number of pruned nodes during computation.

Definition at line 230 of file range_search.hpp.

22.123.4.5 `template<typename MetricType = mlpack::metric::EuclideanDistance, typename TreeType = tree::BinarySpaceTree<bound::HRectBound<2>, RangeSearchStat>> std::vector<size_t> mlpack::range::RangeSearch< MetricType, TreeType >::oldFromNewQueries [private]`

Mappings to old query indices (used when this object builds trees).

Definition at line 213 of file range_search.hpp.

```
22.123.4.6  template<typename MetricType = mpack::metric::EuclideanDistance, typename TreeType = tree::BinarySpaceTree<bound::HRectBound<2>, RangeSearchStat>> std::vector<size_t> mpack::range::RangeSearch< MetricType, TreeType >::oldFromNewReferences  [private]
```

Mappings to old reference indices (used when this object builds trees).

Definition at line 211 of file range_search.hpp.

```
22.123.4.7  template<typename MetricType = mpack::metric::EuclideanDistance, typename TreeType = tree::BinarySpaceTree<bound::HRectBound<2>, RangeSearchStat>> TreeType::Mat mpack::range::RangeSearch< MetricType, TreeType >::queryCopy  [private]
```

Copy of query matrix; used when a tree is built internally.

Definition at line 198 of file range_search.hpp.

```
22.123.4.8  template<typename MetricType = mpack::metric::EuclideanDistance, typename TreeType = tree::BinarySpaceTree<bound::HRectBound<2>, RangeSearchStat>> const TreeType::Mat& mpack::range::RangeSearch< MetricType, TreeType >::querySet  [private]
```

Query set (data should be accessed using this).

Definition at line 203 of file range_search.hpp.

```
22.123.4.9  template<typename MetricType = mpack::metric::EuclideanDistance, typename TreeType = tree::BinarySpaceTree<bound::HRectBound<2>, RangeSearchStat>> TreeType* mpack::range::RangeSearch< MetricType, TreeType >::queryTree  [private]
```

Query tree (may be NULL).

Definition at line 208 of file range_search.hpp.

```
22.123.4.10 template<typename MetricType = mpack::metric::EuclideanDistance, typename TreeType = tree::BinarySpaceTree<bound::HRectBound<2>, RangeSearchStat>> TreeType::Mat mpack::range::RangeSearch< MetricType, TreeType >::referenceCopy  [private]
```

Copy of reference matrix; used when a tree is built internally.

Definition at line 196 of file range_search.hpp.

```
22.123.4.11 template<typename MetricType = mpack::metric::EuclideanDistance, typename TreeType = tree::BinarySpaceTree<bound::HRectBound<2>, RangeSearchStat>> const TreeType::Mat& mpack::range::RangeSearch< MetricType, TreeType >::referenceSet  [private]
```

Reference set (data should be accessed using this).

Definition at line 201 of file range_search.hpp.

```
22.123.4.12  template<typename MetricType = mlpack::metric::EuclideanDistance, typename TreeType = tree::BinarySpaceTree<bound::HRectBound<2>, RangeSearchStat>> TreeType* mlpack::range::RangeSearch< MetricType, TreeType >::referenceTree [private]
```

Reference tree.

Definition at line 206 of file range_search.hpp.

```
22.123.4.13  template<typename MetricType = mlpack::metric::EuclideanDistance, typename TreeType = tree::BinarySpaceTree<bound::HRectBound<2>, RangeSearchStat>> bool mlpack::range::RangeSearch< MetricType, TreeType >::singleMode [private]
```

If true, single-tree computation is used.

Definition at line 224 of file range_search.hpp.

```
22.123.4.14  template<typename MetricType = mlpack::metric::EuclideanDistance, typename TreeType = tree::BinarySpaceTree<bound::HRectBound<2>, RangeSearchStat>> bool mlpack::range::RangeSearch< MetricType, TreeType >::treeOwner [private]
```

If true, this object is responsible for deleting the trees.

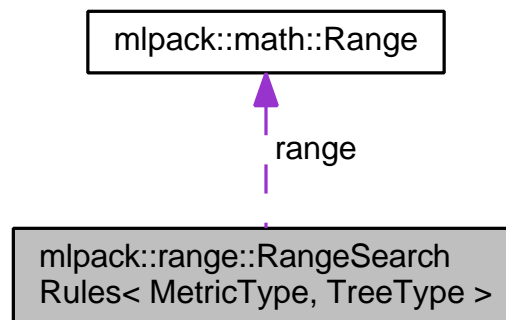
Definition at line 216 of file range_search.hpp.

The documentation for this class was generated from the following file:

- src/mlpack/methods/range_search/range_search.hpp

22.124 mlpack::range::RangeSearchRules< MetricType, TreeType > Class Template Reference

Collaboration diagram for mlpack::range::RangeSearchRules< MetricType, TreeType >:



Public Types

- typedef `neighbor::NeighborSearchTraversalInfo< TreeType >` `TraversalInfoType`

Public Member Functions

- **RangeSearchRules** (const arma::mat &**referenceSet**, const arma::mat &**querySet**, const **math::Range** &**range**, std::vector< std::vector< size_t > > &**neighbors**, std::vector< std::vector< double > > &**distances**, MetricType &**metric**)
Construct the **RangeSearchRules** (p. 518) object.
- double **BaseCase** (const size_t queryIndex, const size_t referenceIndex)
Compute the base case between the given query point and reference point.
- double **Rescore** (const size_t queryIndex, TreeType &referenceNode, const double oldScore) const
Re-evaluate the score for recursion order.
- double **Rescore** (TreeType &queryNode, TreeType &referenceNode, const double oldScore) const
Re-evaluate the score for recursion order.
- double **Score** (const size_t queryIndex, TreeType &referenceNode)
Get the score for recursion order.
- double **Score** (TreeType &queryNode, TreeType &referenceNode)
Get the score for recursion order.
- const **TraversallInfoType** & **TraversallInfo** () const
- **TraversallInfoType** & **TraversallInfo** ()

Private Member Functions

- void **AddResult** (const size_t queryIndex, TreeType &referenceNode)
Add all the points in the given node to the results for the given query point.

Private Attributes

- std::vector< std::vector< double > > & **distances**
The vector the resultant neighbor distances should be stored in.
- size_t **lastQueryIndex**
The last query index.
- size_t **lastReferenceIndex**
The last reference index.
- MetricType & **metric**
The instantiated metric.
- std::vector< std::vector< size_t > > & **neighbors**
The vector the resultant neighbor indices should be stored in.
- const arma::mat & **querySet**
The query set.
- const **math::Range** & **range**
The range of distances for which we are searching.
- const arma::mat & **referenceSet**
The reference set.
- **TraversallInfoType** **traversallInfo**

22.124.1 Detailed Description

```
template<typename MetricType, typename TreeType>class mlpack::range::RangeSearchRules< MetricType, TreeType >
```

Definition at line 24 of file range_search_rules.hpp.

22.124.2 Member Typedef Documentation

22.124.2.1 `template<typename MetricType , typename TreeType > typedef neighbor::NeighborSearch←
TraversallInfo<TreeType> mlpack::range::RangeSearchRules< MetricType, TreeType
>::TraversallInfoType`

Definition at line 103 of file `range_search_rules.hpp`.

22.124.3 Constructor & Destructor Documentation

22.124.3.1 `template<typename MetricType , typename TreeType > mlpack::range::RangeSearchRules< MetricType,
TreeType >::RangeSearchRules (const arma::mat & referenceSet, const arma::mat & querySet, const
math::Range & range, std::vector< std::vector< size_t > > & neighbors, std::vector< std::vector< double > > &
distances, MetricType & metric)`

Construct the **RangeSearchRules** (p. 518) object.

This is usually done from within the **RangeSearch** (p. 512) class at search time.

Parameters

<i>referenceSet</i>	Set of reference data.
<i>querySet</i>	Set of query data.
<i>range</i>	Range to search for.
<i>neighbors</i>	Vector to store resulting neighbors in.
<i>distances</i>	Vector to store resulting distances in.
<i>metric</i>	Instantiated metric.

22.124.4 Member Function Documentation

22.124.4.1 `template<typename MetricType , typename TreeType > void mlpack::range::RangeSearchRules< MetricType,
TreeType >::AddResult (const size_t queryIndex, TreeType & referenceNode) [private]`

Add all the points in the given node to the results for the given query point.

If the base case has already been calculated, we make sure to not add that to the results twice.

22.124.4.2 `template<typename MetricType , typename TreeType > double mlpack::range::RangeSearchRules< MetricType,
TreeType >::BaseCase (const size_t queryIndex, const size_t referenceIndex)`

Compute the base case between the given query point and reference point.

Parameters

<i>queryIndex</i>	Index of query point.
<i>referenceIndex</i>	Index of reference point.

22.124.4.3 `template<typename MetricType , typename TreeType > double mlpack::range::RangeSearchRules< MetricType,
TreeType >::Rescore (const size_t queryIndex, TreeType & referenceNode, const double oldScore) const`

Re-evaluate the score for recursion order.

A low score indicates priority for recursion, while DBL_MAX indicates that the node should not be recursed into at all (it should be pruned). This is used when the score has already been calculated, but another recursion may have modified the bounds for pruning. So the old score is checked against the new pruning bound.

Parameters

<i>queryIndex</i>	Index of query point.
<i>referenceNode</i>	Candidate node to be recursed into.
<i>oldScore</i>	Old score produced by Score() (p. 521) (or Rescore() (p. 520)).

22.124.4.4 `template<typename MetricType , typename TreeType > double mlpack::range::RangeSearchRules< MetricType, TreeType >::Rescore (TreeType & queryNode, TreeType & referenceNode, const double oldScore) const`

Re-evaluate the score for recursion order.

A low score indicates priority for recursion, while DBL_MAX indicates that the node should not be recursed into at all (it should be pruned). This is used when the score has already been calculated, but another recursion may have modified the bounds for pruning. So the old score is checked against the new pruning bound.

Parameters

<i>queryNode</i>	Candidate query node to recurse into.
<i>referenceNode</i>	Candidate reference node to recurse into.
<i>oldScore</i>	Old score produced by Score() (p. 521) (or Rescore() (p. 520)).

22.124.4.5 `template<typename MetricType , typename TreeType > double mlpack::range::RangeSearchRules< MetricType, TreeType >::Score (const size_t queryIndex, TreeType & referenceNode)`

Get the score for recursion order.

A low score indicates priority for recursion, while DBL_MAX indicates that the node should not be recursed into at all (it should be pruned).

Parameters

<i>queryIndex</i>	Index of query point.
<i>referenceNode</i>	Candidate node to be recursed into.

22.124.4.6 `template<typename MetricType , typename TreeType > double mlpack::range::RangeSearchRules< MetricType, TreeType >::Score (TreeType & queryNode, TreeType & referenceNode)`

Get the score for recursion order.

A low score indicates priority for recursion, while DBL_MAX indicates that the node should not be recursed into at all (it should be pruned).

Parameters

<i>queryNode</i>	Candidate query node to recurse into.
<i>referenceNode</i>	Candidate reference node to recurse into.

22.124.4.7 `template<typename MetricType , typename TreeType > const TraversalInfoType& mlpack::range::RangeSearchRules< MetricType, TreeType >::TraversalInfo() const [inline]`

Definition at line 105 of file range_search_rules.hpp.

References `mlpack::range::RangeSearchRules< MetricType, TreeType >::traversalInfo`.

22.124.4.8 `template<typename MetricType , typename TreeType > TraversalInfoType& mlpack::range::RangeSearchRules< MetricType, TreeType >::TraversalInfo() [inline]`

Definition at line 106 of file range_search_rules.hpp.

References `mlpack::range::RangeSearchRules< MetricType, TreeType >::traversalInfo`.

22.124.5 Member Data Documentation

22.124.5.1 `template<typename MetricType , typename TreeType > std::vector<std::vector<double> >& mlpack::range::RangeSearchRules< MetricType, TreeType >::distances [private]`

The vector the resultant neighbor distances should be stored in.

Definition at line 122 of file range_search_rules.hpp.

22.124.5.2 `template<typename MetricType , typename TreeType > size_t mlpack::range::RangeSearchRules< MetricType, TreeType >::lastQueryIndex [private]`

The last query index.

Definition at line 128 of file range_search_rules.hpp.

22.124.5.3 `template<typename MetricType , typename TreeType > size_t mlpack::range::RangeSearchRules< MetricType, TreeType >::lastReferenceIndex [private]`

The last reference index.

Definition at line 130 of file range_search_rules.hpp.

22.124.5.4 `template<typename MetricType , typename TreeType > MetricType& mlpack::range::RangeSearchRules< MetricType, TreeType >::metric [private]`

The instantiated metric.

Definition at line 125 of file range_search_rules.hpp.

22.124.5.5 `template<typename MetricType , typename TreeType > std::vector<std::vector<size_t> >& mlpack::range::RangeSearchRules< MetricType, TreeType >::neighbors [private]`

The vector the resultant neighbor indices should be stored in.

Definition at line 119 of file range_search_rules.hpp.

22.124.5.6 `template<typename MetricType , typename TreeType > const arma::mat& mlpack::range::RangeSearchRules< MetricType, TreeType >::querySet [private]`

The query set.

Definition at line 113 of file range_search_rules.hpp.

22.124.5.7 `template<typename MetricType , typename TreeType > const math::Range& mlpack::range::RangeSearchRules< MetricType, TreeType >::range [private]`

The range of distances for which we are searching.

Definition at line 116 of file range_search_rules.hpp.

22.124.5.8 `template<typename MetricType , typename TreeType > const arma::mat& mlpack::range::RangeSearchRules< MetricType, TreeType >::referenceSet [private]`

The reference set.

Definition at line 110 of file range_search_rules.hpp.

22.124.5.9 `template<typename MetricType , typename TreeType > TraversalInfoType mlpack::range::RangeSearchRules< MetricType, TreeType >::traversalInfo [private]`

Definition at line 138 of file range_search_rules.hpp.

Referenced by mlpack::range::RangeSearchRules< MetricType, TreeType >::TraversalInfo().

The documentation for this class was generated from the following file:

- src/mlpack/methods/range_search/range_search_rules.hpp

22.125 mlpack::range::RangeSearchStat Class Reference

Statistic class for **RangeSearch** (p.512), to be set to the StatisticType of the tree type that range search is being performed with.

Public Member Functions

- **RangeSearchStat** ()
Initialize the statistic.
- `template<typename TreeType >`
RangeSearchStat (TreeType &)
Initialize the statistic given a tree node that this statistic belongs to.
- double **LastDistance** () const
Get the last distance evaluation.
- double & **LastDistance** ()
Modify the last distance evaluation.
- void * **LastDistanceNode** () const
Get the last distance evaluation node.

- void *& **LastDistanceNode** ()

Modify the last distance evaluation node.

Private Attributes

- double **lastDistance**

The last distance evaluation.

- void * **lastDistanceNode**

The last distance evaluation node.

22.125.1 Detailed Description

Statistic class for **RangeSearch** (p. 512), to be set to the `StatisticType` of the tree type that range search is being performed with.

This class just holds the last visited node and the corresponding base case result.

Definition at line 28 of file `range_search_stat.hpp`.

22.125.2 Constructor & Destructor Documentation

22.125.2.1 `mlpack::range::RangeSearchStat::RangeSearchStat ()` `[inline]`

Initialize the statistic.

Definition at line 34 of file `range_search_stat.hpp`.

22.125.2.2 `template<typename TreeType> mlpack::range::RangeSearchStat::RangeSearchStat (TreeType &)` `[inline]`

Initialize the statistic given a tree node that this statistic belongs to.

In this case, we ignore the node.

Definition at line 41 of file `range_search_stat.hpp`.

22.125.3 Member Function Documentation

22.125.3.1 `double mlpack::range::RangeSearchStat::LastDistance () const` `[inline]`

Get the last distance evaluation.

Definition at line 50 of file `range_search_stat.hpp`.

References `lastDistance`.

22.125.3.2 `double& mlpack::range::RangeSearchStat::LastDistance ()` `[inline]`

Modify the last distance evaluation.

Definition at line 52 of file `range_search_stat.hpp`.

References `lastDistance`.

22.125.3.3 `void* mlpack::range::RangeSearchStat::LastDistanceNode () const [inline]`

Get the last distance evaluation node.

Definition at line 46 of file `range_search_stat.hpp`.

References `LastDistanceNode`.

22.125.3.4 `void*& mlpack::range::RangeSearchStat::LastDistanceNode () [inline]`

Modify the last distance evaluation node.

Definition at line 48 of file `range_search_stat.hpp`.

References `LastDistanceNode`.

22.125.4 Member Data Documentation

22.125.4.1 `double mlpack::range::RangeSearchStat::lastDistance [private]`

The last distance evaluation.

Definition at line 58 of file `range_search_stat.hpp`.

Referenced by `LastDistance()`.

22.125.4.2 `void* mlpack::range::RangeSearchStat::lastDistanceNode [private]`

The last distance evaluation node.

Definition at line 56 of file `range_search_stat.hpp`.

Referenced by `LastDistanceNode()`.

The documentation for this class was generated from the following file:

- `src/mlpack/methods/range_search/range_search_stat.hpp`

22.126 mlpack::regression::LARS Class Reference

An implementation of **LARS** (p. 525), a stage-wise homotopy-based algorithm for l_1 -regularized linear regression (LASSO) and $l_1 + l_2$ regularized linear regression (Elastic Net).

Public Member Functions

- **LARS** (const bool **useCholesky**, const double **lambda1**=0.0, const double **lambda2**=0.0, const double **tolerance**=1e-16)
*Set the parameters to **LARS** (p. 525).*
- **LARS** (const bool **useCholesky**, const arma::mat &gramMatrix, const double **lambda1**=0.0, const double **lambda2**=0.0, const double **tolerance**=1e-16)
*Set the parameters to **LARS** (p. 525), and pass in a precalculated Gram matrix.*
- const std::vector< size_t > & **ActiveSet** () const
Access the set of active dimensions.

- `const std::vector< arma::vec > & BetaPath ()` const
Access the set of coefficients after each iteration; the solution is the last element.
- `const std::vector< double > & LambdaPath ()` const
Access the set of values for lambda1 after each iteration; the solution is the last element.
- `const arma::mat & MatUtriCholFactor ()` const
Access the upper triangular cholesky factor.
- `void Regress` (const arma::mat &data, const arma::vec &responses, arma::vec &beta, const bool transpose←Data=true)
*Run **LARS** (p. 525).*
- `std::string ToString ()` const

Private Member Functions

- `void Activate` (const size_t varInd)
Add dimension varInd to active set.
- `void CholeskyDelete` (const size_t colToKill)
- `void CholeskyInsert` (const arma::vec &newX, const arma::mat &X)
- `void CholeskyInsert` (double sqNormNewX, const arma::vec &newGramCol)
- `void ComputeYHatDirection` (const arma::mat &matX, const arma::vec &betaDirection, arma::vec &yHat←Direction)
- `void Deactivate` (const size_t activeVarInd)
Remove activeVarInd'th element from active set.
- `void GivensRotate` (const arma::vec::fixed< 2 > &x, arma::vec::fixed< 2 > &rotatedX, arma::mat &G)
- `void Ignore` (const size_t varInd)
Add dimension varInd to ignores set (never removed).
- `void InterpolateBeta ()`

Private Attributes

- `std::vector< size_t > activeSet`
Active set of dimensions.
- `std::vector< arma::vec > betaPath`
Solution path.
- `bool elasticNet`
True if this is the elastic net problem.
- `std::vector< size_t > ignoreSet`
Set of ignored variables (for dimensions in span{active set dimensions}).
- `std::vector< bool > isActive`
Active set membership indicator (for each dimension).
- `std::vector< bool > isIgnored`
Membership indicator for set of ignored variables.
- `double lambda1`
Regularization parameter for l1 penalty.
- `double lambda2`
Regularization parameter for l2 penalty.
- `std::vector< double > lambdaPath`
Value of lambda_1 for each solution in solution path.

- bool **lasso**

True if this is the LASSO problem.

- const arma::mat & **matGram**

Reference to the Gram matrix we will use.

- arma::mat **matGramInternal**

Gram matrix.

- arma::mat **matUtriCholFactor**

Upper triangular cholesky factor; initially 0x0 matrix.

- double **tolerance**

Tolerance for main loop.

- bool **useCholesky**

Whether or not to use Cholesky decomposition when solving linear system.

22.126.1 Detailed Description

An implementation of **LARS** (p. 525), a stage-wise homotopy-based algorithm for l1-regularized linear regression (LASSO) and l1+l2 regularized linear regression (Elastic Net).

Let X be a matrix where each row is a point and each column is a dimension and let y be a vector of responses.

The Elastic Net problem is to solve

$$\min_{\beta} 0.5 \|X\beta - y\|_2^2 + \lambda_1 \|\beta\|_1 + 0.5\lambda_2 \|\beta\|_2^2$$

where β is the vector of regression coefficients.

If $\lambda_1 > 0$ and $\lambda_2 = 0$, the problem is the LASSO. If $\lambda_1 > 0$ and $\lambda_2 > 0$, the problem is the elastic net. If $\lambda_1 = 0$ and $\lambda_2 > 0$, the problem is ridge regression. If $\lambda_1 = 0$ and $\lambda_2 = 0$, the problem is unregularized linear regression.

Note: This algorithm is not recommended for use (in terms of efficiency) when $\lambda_1 = 0$.

For more details, see the following papers:

```
@article{efron2004least,
  title={Least angle regression},
  author={Efron, B. and Hastie, T. and Johnstone, I. and Tibshirani, R.},
  journal={The Annals of statistics},
  volume={32},
  number={2},
  pages={407--499},
  year={2004},
  publisher={Institute of Mathematical Statistics}
}
```

```
@article{zou2005regularization,
  title={Regularization and variable selection via the elastic net},
  author={Zou, H. and Hastie, T.},
  journal={Journal of the Royal Statistical Society Series B},
  volume={67},
  number={2},
  pages={301--320},
  year={2005},
  publisher={Royal Statistical Society}
}
```

Definition at line 91 of file lars.hpp.

22.126.2 Constructor & Destructor Documentation

22.126.2.1 `mlpack::regression::LARS::LARS (const bool useCholesky, const double lambda1 = 0.0, const double lambda2 = 0.0, const double tolerance = 1e-16)`

Set the parameters to **LARS** (p. 525).

Both `lambda1` and `lambda2` default to 0.

Parameters

<i>useCholesky</i>	Whether or not to use Cholesky decomposition when solving linear system (as opposed to using the full Gram matrix).
<i>lambda1</i>	Regularization parameter for l1-norm penalty.
<i>lambda2</i>	Regularization parameter for l2-norm penalty.
<i>tolerance</i>	Run until the maximum correlation of elements in $(X^T y)$ is less than this.

22.126.2.2 `mlpack::regression::LARS::LARS (const bool useCholesky, const arma::mat & gramMatrix, const double lambda1 = 0.0, const double lambda2 = 0.0, const double tolerance = 1e-16)`

Set the parameters to **LARS** (p. 525), and pass in a precalculated Gram matrix.

Both `lambda1` and `lambda2` default to 0.

Parameters

<i>useCholesky</i>	Whether or not to use Cholesky decomposition when solving linear system (as opposed to using the full Gram matrix).
<i>gramMatrix</i>	Gram matrix.
<i>lambda1</i>	Regularization parameter for l1-norm penalty.
<i>lambda2</i>	Regularization parameter for l2-norm penalty.
<i>tolerance</i>	Run until the maximum correlation of elements in $(X^T y)$ is less than this.

22.126.3 Member Function Documentation

22.126.3.1 `void mlpack::regression::LARS::Activate (const size_t varInd) [private]`

Add dimension `varInd` to active set.

Parameters

<i>varInd</i>	Dimension to add to active set.
---------------	---------------------------------

22.126.3.2 `const std::vector<size_t> & mlpack::regression::LARS::ActiveSet () const [inline]`

Access the set of active dimensions.

Definition at line 147 of file `lars.hpp`.

References `activeSet`.

22.126.3.3 `const std::vector<arma::vec> & mlpack::regression::LARS::BetaPath () const [inline]`

Access the set of coefficients after each iteration; the solution is the last element.

Definition at line 151 of file lars.hpp.

References `betaPath`.

22.126.3.4 `void mlpack::regression::LARS::CholeskyDelete (const size_t colToKill) [private]`

22.126.3.5 `void mlpack::regression::LARS::CholeskyInsert (const arma::vec & newX, const arma::mat & X) [private]`

22.126.3.6 `void mlpack::regression::LARS::CholeskyInsert (double sqNormNewX, const arma::vec & newGramCol) [private]`

22.126.3.7 `void mlpack::regression::LARS::ComputeYHatDirection (const arma::mat & matX, const arma::vec & betaDirection, arma::vec & yHatDirection) [private]`

22.126.3.8 `void mlpack::regression::LARS::Deactivate (const size_t activeVarInd) [private]`

Remove `activeVarInd`'th element from active set.

Parameters

<i>activeVarInd</i>	Index of element to remove from active set.
---------------------	---

22.126.3.9 `void mlpack::regression::LARS::GivensRotate (const arma::vec::fixed< 2 > & x, arma::vec::fixed< 2 > & rotatedX, arma::mat & G) [private]`

22.126.3.10 `void mlpack::regression::LARS::Ignore (const size_t varInd) [private]`

Add dimension `varInd` to ignores set (never removed).

Parameters

<i>varInd</i>	Dimension to add to ignores set.
---------------	----------------------------------

22.126.3.11 `void mlpack::regression::LARS::InterpolateBeta () [private]`

22.126.3.12 `const std::vector<double>& mlpack::regression::LARS::LambdaPath () const [inline]`

Access the set of values for `lambda1` after each iteration; the solution is the last element.

Definition at line 155 of file lars.hpp.

References `lambdaPath`.

22.126.3.13 `const arma::mat& mlpack::regression::LARS::MatUtriCholFactor () const [inline]`

Access the upper triangular cholesky factor.

Definition at line 158 of file lars.hpp.

References `matUtriCholFactor`.

22.126.3.14 `void mlpack::regression::LARS::Regress (const arma::mat & data, const arma::vec & responses, arma::vec & beta, const bool transposeData = true)`

Run **LARS** (p. 525).

The input matrix (like all MLPACK matrices) should be column-major – each column is an observation and each row is a dimension. However, because **LARS** (p. 525) is more efficient on a row-major matrix, this method will (internally) transpose the matrix. If this transposition is not necessary (i.e., you want to pass in a row-major matrix), pass 'false' for the `transposeData` parameter.

Parameters

<i>data</i>	Column-major input data (or row-major input data if <code>rowMajor = true</code>).
<i>responses</i>	A vector of targets.
<i>beta</i>	Vector to store the solution (the coefficients) in.
<i>rowMajor</i>	Set to false if the data is row-major.

22.126.3.15 `std::string mlpack::regression::LARS::ToString () const`

22.126.4 Member Data Documentation

22.126.4.1 `std::vector<size_t> mlpack::regression::LARS::activeSet` [private]

Active set of dimensions.

Definition at line 196 of file `lars.hpp`.

Referenced by `ActiveSet()`.

22.126.4.2 `std::vector<arma::vec> mlpack::regression::LARS::betaPath` [private]

Solution path.

Definition at line 190 of file `lars.hpp`.

Referenced by `BetaPath()`.

22.126.4.3 `bool mlpack::regression::LARS::elasticNet` [private]

True if this is the elastic net problem.

Definition at line 182 of file `lars.hpp`.

22.126.4.4 `std::vector<size_t> mlpack::regression::LARS::ignoreSet` [private]

Set of ignored variables (for dimensions in `span{active set dimensions}`).

Definition at line 204 of file `lars.hpp`.

22.126.4.5 `std::vector<bool> mlpack::regression::LARS::isActive` [private]

Active set membership indicator (for each dimension).

Definition at line 199 of file `lars.hpp`.

22.126.4.6 `std::vector<bool> mpack::regression::LARS::isIgnored` [private]

Membership indicator for set of ignored variables.

Definition at line 207 of file lars.hpp.

22.126.4.7 `double mpack::regression::LARS::lambda1` [private]

Regularization parameter for l1 penalty.

Definition at line 179 of file lars.hpp.

22.126.4.8 `double mpack::regression::LARS::lambda2` [private]

Regularization parameter for l2 penalty.

Definition at line 184 of file lars.hpp.

22.126.4.9 `std::vector<double> mpack::regression::LARS::lambdaPath` [private]

Value of lambda_1 for each solution in solution path.

Definition at line 193 of file lars.hpp.

Referenced by LambdaPath().

22.126.4.10 `bool mpack::regression::LARS::lasso` [private]

True if this is the LASSO problem.

Definition at line 177 of file lars.hpp.

22.126.4.11 `const arma::mat& mpack::regression::LARS::matGram` [private]

Reference to the Gram matrix we will use.

Definition at line 168 of file lars.hpp.

22.126.4.12 `arma::mat mpack::regression::LARS::matGramInternal` [private]

Gram matrix.

Definition at line 165 of file lars.hpp.

22.126.4.13 `arma::mat mpack::regression::LARS::matUtriCholFactor` [private]

Upper triangular cholesky factor; initially 0x0 matrix.

Definition at line 171 of file lars.hpp.

Referenced by MatUtriCholFactor().

22.126.4.14 `double mlpack::regression::LARS::tolerance` [private]

Tolerance for main loop.

Definition at line 187 of file lars.hpp.

22.126.4.15 `bool mlpack::regression::LARS::useCholesky` [private]

Whether or not to use Cholesky decomposition when solving linear system.

Definition at line 174 of file lars.hpp.

The documentation for this class was generated from the following file:

- `src/mlpack/methods/lars/lars.hpp`

22.127 `mlpack::regression::LinearRegression` Class Reference

A simple linear regression algorithm using ordinary least squares.

Public Member Functions

- **LinearRegression** (const arma::mat &predictors, const arma::vec &responses, const double **lambda**=0, const bool **intercept**=true, const arma::vec &weights=arma::vec())
Creates the model.
- **LinearRegression** (const std::string &filename)
Initialize the model from a file.
- **LinearRegression** (const **LinearRegression** &linearRegression)
Copy constructor.
- **LinearRegression** ()
Empty constructor.
- double **ComputeError** (const arma::mat &points, const arma::vec &responses) const
Calculate the L2 squared error on the given predictors and responses using this linear regression model.
- double **Lambda** () const
Return the Tikhonov regularization parameter for ridge regression.
- double & **Lambda** ()
Modify the Tikhonov regularization parameter for ridge regression.
- const arma::vec & **Parameters** () const
Return the parameters (the b vector).
- arma::vec & **Parameters** ()
Modify the parameters (the b vector).
- void **Predict** (const arma::mat &points, arma::vec &predictions) const
Calculate y_i for each data point in points.
- std::string **ToString** () const

Private Attributes

- bool **intercept**
Indicates whether first parameter is intercept.
- double **lambda**
The Tikhonov regularization parameter for ridge regression (0 for linear regression).
- arma::vec **parameters**
The calculated B.

22.127.1 Detailed Description

A simple linear regression algorithm using ordinary least squares.

Optionally, this class can perform ridge regression, if the lambda parameter is set to a number greater than zero.

Definition at line 28 of file linear_regression.hpp.

22.127.2 Constructor & Destructor Documentation

22.127.2.1 mlpack::regression::LinearRegression::LinearRegression (const arma::mat & *predictors*, const arma::vec & *responses*, const double *lambda* = 0, const bool *intercept* = true, const arma::vec & *weights* = arma::vec())

Creates the model.

Parameters

<i>predictors</i>	X, matrix of data points to create B with.
<i>responses</i>	y, the measured data for each point in X
<i>intercept</i>	include intercept?
<i>weights</i>	observation weights

22.127.2.2 mlpack::regression::LinearRegression::LinearRegression (const std::string & *filename*)

Initialize the model from a file.

Parameters

<i>filename</i>	the name of the file to load the model from.
-----------------	--

22.127.2.3 mlpack::regression::LinearRegression::LinearRegression (const LinearRegression & *linearRegression*)

Copy constructor.

Parameters

<i>linearRegression</i>	the other instance to copy parameters from.
-------------------------	---

22.127.2.4 mlpack::regression::LinearRegression::LinearRegression () `[inline]`

Empty constructor.

Definition at line 63 of file linear_regression.hpp.

22.127.3 Member Function Documentation

22.127.3.1 `double mlpack::regression::LinearRegression::ComputeError (const arma::mat & points, const arma::vec & responses) const`

Calculate the L2 squared error on the given predictors and responses using this linear regression model.

This calculation returns

$$(1/n) * \|y - XB\|_2^2$$

where y is the responses vector, X is the matrix of predictors, and B is the parameters of the trained linear regression model.

As this number decreases to 0, the linear regression fit is better.

Parameters

<i>predictors</i>	Matrix of predictors (X).
<i>responses</i>	Vector of responses (y).

22.127.3.2 `double mlpack::regression::LinearRegression::Lambda () const` `[inline]`

Return the Tikhonov regularization parameter for ridge regression.

Definition at line 99 of file `linear_regression.hpp`.

References `lambda`.

22.127.3.3 `double& mlpack::regression::LinearRegression::Lambda ()` `[inline]`

Modify the Tikhonov regularization parameter for ridge regression.

Definition at line 101 of file `linear_regression.hpp`.

References `lambda`.

22.127.3.4 `const arma::vec& mlpack::regression::LinearRegression::Parameters () const` `[inline]`

Return the parameters (the b vector).

Definition at line 94 of file `linear_regression.hpp`.

References `parameters`.

22.127.3.5 `arma::vec& mlpack::regression::LinearRegression::Parameters ()` `[inline]`

Modify the parameters (the b vector).

Definition at line 96 of file `linear_regression.hpp`.

References `parameters`.

22.127.3.6 void mlpack::regression::LinearRegression::Predict (const arma::mat & *points*, arma::vec & *predictions*) const

Calculate y_i for each data point in *points*.

Parameters

<i>points</i>	the data points to calculate with.
<i>predictions</i>	y, will contain calculated values on completion.

22.127.3.7 `std::string mlpack::regression::LinearRegression::ToString () const`

22.127.4 Member Data Documentation

22.127.4.1 `bool mlpack::regression::LinearRegression::intercept [private]`

Indicates whether first parameter is intercept.

Definition at line 118 of file `linear_regression.hpp`.

22.127.4.2 `double mlpack::regression::LinearRegression::lambda [private]`

The Tikhonov regularization parameter for ridge regression (0 for linear regression).

Definition at line 116 of file `linear_regression.hpp`.

Referenced by `Lambda()`.

22.127.4.3 `arma::vec mlpack::regression::LinearRegression::parameters [private]`

The calculated B.

Initialized and filled by constructor to hold the least squares solution.

Definition at line 111 of file `linear_regression.hpp`.

Referenced by `Parameters()`.

The documentation for this class was generated from the following file:

- `src/mlpack/methods/linear_regression/linear_regression.hpp`

22.128 `mlpack::regression::LogisticRegression< OptimizerType >` Class Template Reference

Public Member Functions

- **LogisticRegression** (const arma::mat &predictors, const arma::vec &responses, const double **lambda**=0)
Construct the **LogisticRegression** (p. 536) class with the given labeled training data.
- **LogisticRegression** (const arma::mat &predictors, const arma::vec &responses, const arma::mat &initialPoint, const double **lambda**=0)
Construct the **LogisticRegression** (p. 536) class with the given labeled training data.
- **LogisticRegression** (OptimizerType< **LogisticRegressionFunction** > &optimizer)
Construct the **LogisticRegression** (p. 536) class with the given labeled training data.
- **LogisticRegression** (const arma::vec ¶meters, const double **lambda**=0)
Construct a logistic regression model from the given parameters, without performing any training.

- double **ComputeAccuracy** (const arma::mat &predictors, const arma::vec &responses, const double decisionBoundary=0.5) const
Compute the accuracy of the model on the given predictors and responses, optionally using the given decision boundary.
- double **ComputeError** (const arma::mat &predictors, const arma::vec &responses) const
Compute the error of the model.
- const double & **Lambda** () const
Return the lambda value for L2-regularization.
- double & **Lambda** ()
Modify the lambda value for L2-regularization.
- const arma::vec & **Parameters** () const
Return the parameters (the b vector).
- arma::vec & **Parameters** ()
Modify the parameters (the b vector).
- void **Predict** (const arma::mat &predictors, arma::vec &responses, const double decisionBoundary=0.5) const
Predict the responses to a given set of predictors.
- std::string **ToString** () const

Private Attributes

- double **lambda**
L2-regularization penalty parameter.
- arma::vec **parameters**
Vector of trained parameters.

22.128.1 Detailed Description

```
template<template< typename > class OptimizerType = mlpack::optimization::L_BFGS>class mlpack::regression::LogisticRegression< OptimizerType >
```

Definition at line 29 of file logistic_regression.hpp.

22.128.2 Constructor & Destructor Documentation

22.128.2.1 `template<template< typename > class OptimizerType = mlpack::optimization::L_BFGS> mlpack::regression::LogisticRegression< OptimizerType >::LogisticRegression (const arma::mat & predictors, const arma::vec & responses, const double lambda = 0)`

Construct the **LogisticRegression** (p. 536) class with the given labeled training data.

This will train the model. Optionally, specify lambda, which is the penalty parameter for L2-regularization. If not specified, it is set to 0, which results in standard (unregularized) logistic regression.

Parameters

<i>predictors</i>	Input training variables.
<i>responses</i>	Outputs resulting from input training variables.

<i>lambda</i>	L2-regularization parameter.
---------------	------------------------------

22.128.2.2 `template<template< typename > class OptimizerType = mlpack::optimization::L_BFGS>
mlpack::regression::LogisticRegression< OptimizerType >::LogisticRegression (const arma::mat &
predictors, const arma::vec & responses, const arma::mat & initialPoint, const double lambda = 0)`

Construct the **LogisticRegression** (p. 536) class with the given labeled training data.

This will train the model. Optionally, specify lambda, which is the penalty parameter for L2-regularization. If not specified, it is set to 0, which results in standard (unregularized) logistic regression.

Parameters

<i>predictors</i>	Input training variables.
<i>responses</i>	Outputs results from input training variables.
<i>initialPoint</i>	Initial model to train with.
<i>lambda</i>	L2-regularization parameter.

22.128.2.3 `template<template< typename > class OptimizerType = mlpack::optimization::L_BFGS>
mlpack::regression::LogisticRegression< OptimizerType >::LogisticRegression (OptimizerType<
LogisticRegressionFunction > & optimizer)`

Construct the **LogisticRegression** (p. 536) class with the given labeled training data.

This will train the model. This overload takes an already instantiated optimizer (which holds the **LogisticRegressionFunction** (p. 541) error function, which must also be instantiated), so that the optimizer can be configured before the training is run by this constructor. The predictors and responses and initial point are all taken from the error function contained in the optimizer.

Parameters

<i>optimizer</i>	Instantiated optimizer with instantiated error function.
------------------	--

22.128.2.4 `template<template< typename > class OptimizerType = mlpack::optimization::L_BFGS>
mlpack::regression::LogisticRegression< OptimizerType >::LogisticRegression (const arma::vec &
parameters, const double lambda = 0)`

Construct a logistic regression model from the given parameters, without performing any training.

The lambda parameter is used for the **ComputeAccuracy()** (p. 539) and **ComputeError()** (p. 539) functions; this constructor does not train the model itself.

Parameters

<i>parameters</i>	Parameters making up the model.
<i>lambda</i>	L2-regularization penalty parameter.

22.128.3 Member Function Documentation

22.128.3.1 `template<template< typename > class OptimizerType = mlpack::optimization::L_BFGS> double
mlpack::regression::LogisticRegression< OptimizerType >::ComputeAccuracy (const arma::mat & predictors,
const arma::vec & responses, const double decisionBoundary = 0.5) const`

Compute the accuracy of the model on the given predictors and responses, optionally using the given decision boundary.

The responses should be either 0 or 1. Logistic regression returns a value between 0 and 1. If the value is greater than the decision boundary, the response is taken to be 1; otherwise, it is 0. By default, the decision boundary is 0.5.

The accuracy is returned as a percentage, between 0 and 100.

Parameters

<i>predictors</i>	Input predictors.
<i>responses</i>	Vector of responses.
<i>decisionBoundary</i>	Decision boundary (default 0.5).

Returns

Percentage of responses that are predicted correctly.

22.128.3.2 `template<template< typename > class OptimizerType = mlpack::optimization::L_BFGS> double
mlpack::regression::LogisticRegression< OptimizerType >::ComputeError (const arma::mat & predictors,
const arma::vec & responses) const`

Compute the error of the model.

This returns the negative objective function of the logistic regression log-likelihood function. For the model to be optimal, the negative log-likelihood function should be minimized.

Parameters

<i>predictors</i>	Input predictors.
<i>responses</i>	Vector of responses.

22.128.3.3 `template<template< typename > class OptimizerType = mlpack::optimization::L_BFGS> const double&
mlpack::regression::LogisticRegression< OptimizerType >::Lambda () const [inline]`

Return the lambda value for L2-regularization.

Definition at line 92 of file logistic_regression.hpp.

References mlpack::regression::LogisticRegression< OptimizerType >::lambda.

22.128.3.4 `template<template< typename > class OptimizerType = mlpack::optimization::L_BFGS> double&
mlpack::regression::LogisticRegression< OptimizerType >::Lambda () [inline]`

Modify the lambda value for L2-regularization.

Definition at line 94 of file logistic_regression.hpp.

References mlpack::regression::LogisticRegression< OptimizerType >::lambda.

22.128.3.5 `template<template< typename > class OptimizerType = mlpack::optimization::L_BFGS> const arma::vec& mlpack::regression::LogisticRegression< OptimizerType >::Parameters () const [inline]`

Return the parameters (the b vector).

Definition at line 87 of file logistic_regression.hpp.

References mlpack::regression::LogisticRegression< OptimizerType >::parameters.

22.128.3.6 `template<template< typename > class OptimizerType = mlpack::optimization::L_BFGS> arma::vec& mlpack::regression::LogisticRegression< OptimizerType >::Parameters () [inline]`

Modify the parameters (the b vector).

Definition at line 89 of file logistic_regression.hpp.

References mlpack::regression::LogisticRegression< OptimizerType >::parameters.

22.128.3.7 `template<template< typename > class OptimizerType = mlpack::optimization::L_BFGS> void mlpack::regression::LogisticRegression< OptimizerType >::Predict (const arma::mat & predictors, arma::vec & responses, const double decisionBoundary = 0.5) const`

Predict the responses to a given set of predictors.

The responses will be either 0 or 1. Optionally, specify the decision boundary; logistic regression returns a value between 0 and 1. If the value is greater than the decision boundary, the response is taken to be 1; otherwise, it is 0. By default the decision boundary is 0.5.

Parameters

<i>predictors</i>	Input predictors.
<i>responses</i>	Vector to put output predictions of responses into.
<i>decisionBoundary</i>	Decision boundary (default 0.5).

22.128.3.8 `template<template< typename > class OptimizerType = mlpack::optimization::L_BFGS> std::string mlpack::regression::LogisticRegression< OptimizerType >::ToString () const`

22.128.4 Member Data Documentation

22.128.4.1 `template<template< typename > class OptimizerType = mlpack::optimization::L_BFGS> double mlpack::regression::LogisticRegression< OptimizerType >::lambda [private]`

L2-regularization penalty parameter.

Definition at line 147 of file logistic_regression.hpp.

Referenced by mlpack::regression::LogisticRegression< OptimizerType >::Lambda().

22.128.4.2 `template<template< typename > class OptimizerType = mlpack::optimization::L_BFGS> arma::vec mlpack::regression::LogisticRegression< OptimizerType >::parameters [private]`

Vector of trained parameters.

Definition at line 145 of file logistic_regression.hpp.

Referenced by mlpack::regression::LogisticRegression< OptimizerType >::Parameters().

The documentation for this class was generated from the following file:

- src/mlpack/methods/logistic_regression/logistic_regression.hpp

22.129 mlpack::regression::LogisticRegressionFunction Class Reference

The log-likelihood function for the logistic regression objective function.

Public Member Functions

- **LogisticRegressionFunction** (const arma::mat &**predictors**, const arma::vec &**responses**, const double **lambda**=0)
- **LogisticRegressionFunction** (const arma::mat &**predictors**, const arma::vec &**responses**, const arma::mat &**initialPoint**, const double **lambda**=0)
- double **Evaluate** (const arma::mat ¶meters) const
Evaluate the logistic regression log-likelihood function with the given parameters.
- double **Evaluate** (const arma::mat ¶meters, const size_t i) const
Evaluate the logistic regression log-likelihood function with the given parameters, but using only one data point.
- const arma::mat & **GetInitialPoint** () const
Return the initial point for the optimization.
- void **Gradient** (const arma::mat ¶meters, arma::mat &gradient) const
Evaluate the gradient of the logistic regression log-likelihood function with the given parameters.
- void **Gradient** (const arma::mat ¶meters, const size_t i, arma::mat &gradient) const
Evaluate the gradient of the logistic regression log-likelihood function with the given parameters, and with respect to only one point in the dataset.
- const arma::mat & **InitialPoint** () const
Return the initial point for the optimization.
- arma::mat & **InitialPoint** ()
Modify the initial point for the optimization.
- const double & **Lambda** () const
Return the regularization parameter (lambda).
- double & **Lambda** ()
Modify the regularization parameter (lambda).
- size_t **NumFunctions** () const
Return the number of separable functions (the number of predictor points).
- const arma::mat & **Predictors** () const
Return the matrix of predictors.
- const arma::vec & **Responses** () const
Return the vector of responses.

Private Attributes

- arma::mat **initialPoint**
The initial point, from which to start the optimization.
- double **lambda**
The regularization parameter for L2-regularization.
- const arma::mat & **predictors**
The matrix of data points (predictors).
- const arma::vec & **responses**
The vector of responses to the input data points.

22.129.1 Detailed Description

The log-likelihood function for the logistic regression objective function.

This is used by various mlpack optimizers to train a logistic regression model.

Definition at line 29 of file logistic_regression_function.hpp.

22.129.2 Constructor & Destructor Documentation

22.129.2.1 mlpack::regression::LogisticRegressionFunction::LogisticRegressionFunction (const arma::mat & *predictors*, const arma::vec & *responses*, const double *lambda* = 0)

22.129.2.2 mlpack::regression::LogisticRegressionFunction::LogisticRegressionFunction (const arma::mat & *predictors*, const arma::vec & *responses*, const arma::mat & *initialPoint*, const double *lambda* = 0)

22.129.3 Member Function Documentation

22.129.3.1 double mlpack::regression::LogisticRegressionFunction::Evaluate (const arma::mat & *parameters*) const

Evaluate the logistic regression log-likelihood function with the given parameters.

Note that if a point has 0 probability of being classified directly with the given parameters, then **Evaluate()** (p. 542) will return nan (this is kind of a corner case and should not happen for reasonable models).

The optimum (minimum) of this function is 0.0, and occurs when each point is classified correctly with very high probability.

Parameters

<i>parameters</i>	Vector of logistic regression parameters.
-------------------	---

22.129.3.2 double mlpack::regression::LogisticRegressionFunction::Evaluate (const arma::mat & *parameters*, const size_t *i*) const

Evaluate the logistic regression log-likelihood function with the given parameters, but using only one data point.

This is useful for optimizers such as SGD, which require a separable objective function. Note that if the point has 0 probability of being classified correctly with the given parameters, then **Evaluate()** (p. 542) will return nan (this is kind of a corner case and should not happen for reasonable models).

The optimum (minimum) of this function is 0.0, and occurs when the point is classified correctly with very high probability.

Parameters

<i>parameters</i>	Vector of logistic regression parameters.
<i>i</i>	Index of point to use for objective function evaluation.

22.129.3.3 `const arma::mat& mlpack::regression::LogisticRegressionFunction::GetInitialPoint () const` `[inline]`

Return the initial point for the optimization.

Definition at line 109 of file `logistic_regression_function.hpp`.

References `initialPoint`.

22.129.3.4 `void mlpack::regression::LogisticRegressionFunction::Gradient (const arma::mat & parameters, arma::mat & gradient) const`

Evaluate the gradient of the logistic regression log-likelihood function with the given parameters.

Parameters

<i>parameters</i>	Vector of logistic regression parameters.
<i>gradient</i>	Vector to output gradient into.

22.129.3.5 `void mlpack::regression::LogisticRegressionFunction::Gradient (const arma::mat & parameters, const size_t i, arma::mat & gradient) const`

Evaluate the gradient of the logistic regression log-likelihood function with the given parameters, and with respect to only one point in the dataset.

This is useful for optimizers such as SGD, which require a separable objective function.

Parameters

<i>parameters</i>	Vector of logistic regression parameters.
<i>i</i>	Index of points to use for objective function gradient evaluation.
<i>gradient</i>	Vector to output gradient into.

22.129.3.6 `const arma::mat& mlpack::regression::LogisticRegressionFunction::InitialPoint () const` `[inline]`

Return the initial point for the optimization.

Definition at line 42 of file `logistic_regression_function.hpp`.

References `initialPoint`.

22.129.3.7 `arma::mat& mlpack::regression::LogisticRegressionFunction::InitialPoint ()` `[inline]`

Modify the initial point for the optimization.

Definition at line 44 of file `logistic_regression_function.hpp`.

References `initialPoint`.

22.129.3.8 `const double& mlpack::regression::LogisticRegressionFunction::Lambda () const` `[inline]`

Return the regularization parameter (lambda).

Definition at line 47 of file `logistic_regression_function.hpp`.

References `lambda`.

22.129.3.9 `double& mlpack::regression::LogisticRegressionFunction::Lambda ()` `[inline]`

Modify the regularization parameter (lambda).

Definition at line 49 of file `logistic_regression_function.hpp`.

References `lambda`.

22.129.3.10 `size_t mlpack::regression::LogisticRegressionFunction::NumFunctions () const` `[inline]`

Return the number of separable functions (the number of predictor points).

Definition at line 112 of file `logistic_regression_function.hpp`.

22.129.3.11 `const arma::mat& mlpack::regression::LogisticRegressionFunction::Predictors () const` `[inline]`

Return the matrix of predictors.

Definition at line 52 of file `logistic_regression_function.hpp`.

References `predictors`.

22.129.3.12 `const arma::vec& mlpack::regression::LogisticRegressionFunction::Responses () const` `[inline]`

Return the vector of responses.

Definition at line 54 of file `logistic_regression_function.hpp`.

References `responses`.

22.129.4 Member Data Documentation

22.129.4.1 `arma::mat mlpack::regression::LogisticRegressionFunction::initialPoint` `[private]`

The initial point, from which to start the optimization.

Definition at line 116 of file `logistic_regression_function.hpp`.

Referenced by `GetInitialPoint()`, and `InitialPoint()`.

22.129.4.2 `double mlpack::regression::LogisticRegressionFunction::lambda` `[private]`

The regularization parameter for L2-regularization.

Definition at line 122 of file `logistic_regression_function.hpp`.

Referenced by `Lambda()`.

22.129.4.3 `const arma::mat& mlpack::regression::LogisticRegressionFunction::predictors` `[private]`

The matrix of data points (predictors).

Definition at line 118 of file `logistic_regression_function.hpp`.

Referenced by `Predictors()`.

22.129.4.4 `const arma::vec& mlpack::regression::LogisticRegressionFunction::responses` `[private]`

The vector of responses to the input data points.

Definition at line 120 of file `logistic_regression_function.hpp`.

Referenced by `Responses()`.

The documentation for this class was generated from the following file:

- `src/mlpack/methods/logistic_regression/logistic_regression_function.hpp`

22.130 mlpack::sparse_coding::DataDependentRandomInitializer Class Reference

A data-dependent random dictionary initializer for **SparseCoding** (p. 547).

Static Public Member Functions

- static void **Initialize** (const arma::mat &data, const size_t atoms, arma::mat &dictionary)
Initialize the dictionary by adding together three random observations from the data, and then normalizing the atom.

22.130.1 Detailed Description

A data-dependent random dictionary initializer for **SparseCoding** (p. 547).

This creates random dictionary atoms by adding three random observations from the data together, and then normalizing the atom.

Definition at line 27 of file `data_dependent_random_initializer.hpp`.

22.130.2 Member Function Documentation

22.130.2.1 `static void mlpack::sparse_coding::DataDependentRandomInitializer::Initialize (const arma::mat & data, const size_t atoms, arma::mat & dictionary)` `[inline]`, `[static]`

Initialize the dictionary by adding together three random observations from the data, and then normalizing the atom.

This implementation is simple enough to be included with the definition.

Parameters

<i>data</i>	Dataset to initialize the dictionary with.
<i>atoms</i>	Number of atoms in dictionary.
<i>dictionary</i>	Dictionary to initialize.

Definition at line 39 of file `data_dependent_random_initializer.hpp`.

References `mlpack::math::RandInt()`.

The documentation for this class was generated from the following file:

- `src/mlpack/methods/sparse_coding/data_dependent_random_initializer.hpp`

22.131 `mlpack::sparse_coding::NothingInitializer` Class Reference

A DictionaryInitializer for **SparseCoding** (p. 547) which does not initialize anything; it is useful for when the dictionary is already known and will be set with **SparseCoding::Dictionary()** (p. 550).

Static Public Member Functions

- static void **Initialize** (const arma::mat &, const size_t, arma::mat &)

This function does not initialize the dictionary.

22.131.1 Detailed Description

A DictionaryInitializer for **SparseCoding** (p. 547) which does not initialize anything; it is useful for when the dictionary is already known and will be set with **SparseCoding::Dictionary()** (p. 550).

Definition at line 29 of file `nothing_initializer.hpp`.

22.131.2 Member Function Documentation

22.131.2.1 static void `mlpack::sparse_coding::NothingInitializer::Initialize` (const arma::mat &, const size_t, arma::mat &)
[inline], [static]

This function does not initialize the dictionary.

This will cause problems for **SparseCoding** (p. 547) if the dictionary is not set manually before running the method.

Definition at line 37 of file `nothing_initializer.hpp`.

The documentation for this class was generated from the following file:

- `src/mlpack/methods/sparse_coding/nothing_initializer.hpp`

22.132 `mlpack::sparse_coding::RandomInitializer` Class Reference

A DictionaryInitializer for use with the **SparseCoding** (p. 547) class.

Static Public Member Functions

- static void **Initialize** (const arma::mat &data, const size_t atoms, arma::mat &dictionary)
Initialize the dictionary randomly from a normal distribution, such that each atom has a norm of 1.

22.132.1 Detailed Description

A DictionaryInitializer for use with the **SparseCoding** (p. 547) class.

This provides a random, normally distributed dictionary, such that each atom has a norm of 1.

Definition at line 27 of file random_initializer.hpp.

22.132.2 Member Function Documentation

22.132.2.1 static void mlpack::sparse_coding::RandomInitializer::Initialize (const arma::mat & data, const size_t atoms, arma::mat & dictionary) [inline],[static]

Initialize the dictionary randomly from a normal distribution, such that each atom has a norm of 1.

This is simple enough to be included with the definition.

Parameters

<i>data</i>	Dataset to use for initialization.
<i>atoms</i>	Number of atoms (columns) in the dictionary.
<i>dictionary</i>	Dictionary to initialize.

Definition at line 39 of file random_initializer.hpp.

The documentation for this class was generated from the following file:

- src/mlpack/methods/sparse_coding/random_initializer.hpp

22.133 mlpack::sparse_coding::SparseCoding< DictionaryInitializer > Class Template Reference

An implementation of Sparse Coding with Dictionary Learning that achieves sparsity via an l1-norm regularizer on the codes (LASSO) or an (l1+l2)-norm regularizer on the codes (the Elastic Net).

Public Member Functions

- **SparseCoding** (const arma::mat &data, const size_t atoms, const double lambda1, const double lambda2=0)
*Set the parameters to **SparseCoding** (p. 547).*
- const arma::mat & **Codes** () const
Access the sparse codes.
- arma::mat & **Codes** ()
Modify the sparse codes.
- const arma::mat & **Data** () const
Access the data.
- const arma::mat & **Dictionary** () const

- *Access the dictionary.*
- arma::mat & **Dictionary** ()
- *Modify the dictionary.*
- void **Encode** (const size_t maxIterations=0, const double objTolerance=0.01, const double newtonTolerance=1e-6)
- *Run Sparse Coding with Dictionary Learning.*
- double **Objective** () const
- *Compute the objective function.*
- void **OptimizeCode** ()
- *Sparse code each point via LARS.*
- double **OptimizeDictionary** (const arma::uvec &adjacencies, const double newtonTolerance=1e-6, const size_t maxIterations=50)
- *Learn dictionary via Newton method based on Lagrange dual.*
- void **ProjectDictionary** ()
- *Project each atom of the dictionary back onto the unit ball, if necessary.*
- std::string **ToString** () const

Private Attributes

- size_t **atoms**
- *Number of atoms.*
- arma::mat **codes**
- *Sparse codes (columns are points).*
- const arma::mat & **data**
- *Data matrix (columns are points).*
- arma::mat **dictionary**
- *Dictionary (columns are atoms).*
- double **lambda1**
- *l1 regularization term.*
- double **lambda2**
- *l2 regularization term.*

22.133.1 Detailed Description

```
template<typename DictionaryInitializer = DataDependentRandomInitializer> class mlpack::sparse_coding::SparseCoding<
DictionaryInitializer >
```

An implementation of Sparse Coding with Dictionary Learning that achieves sparsity via an l1-norm regularizer on the codes (LASSO) or an (l1+l2)-norm regularizer on the codes (the Elastic Net).

Let d be the number of dimensions in the original space, m the number of training points, and k the number of atoms in the dictionary (the dimension of the learned feature space). The training data X is a d -by- m matrix where each column is a point and each row is a dimension. The dictionary D is a d -by- k matrix, and the sparse codes matrix Z is a k -by- m matrix. This program seeks to minimize the objective:

$$\min_{D,Z} 0.5 \|X - DZ\|_F^2 + \lambda_1 \sum_{i=1}^m \|Z_i\|_1 + 0.5 \lambda_2 \sum_{i=1}^m \|Z_i\|_2^2$$

subject to $\|D_j\|_2 \leq 1$ for $1 \leq j \leq k$ where typically $\lambda_1 > 0$ and $\lambda_2 = 0$.

This problem is solved by an algorithm that alternates between a dictionary learning step and a sparse coding step. The dictionary learning step updates the dictionary D using a Newton method based on the Lagrange dual (see the paper below for details). The sparse coding step involves solving a large number of sparse linear regression problems; this can be done efficiently using LARS, an algorithm that can solve the LASSO or the Elastic Net (papers below).

Here are those papers:

```
@incollection{lee2007efficient,
  title = {Efficient sparse coding algorithms},
  author = {Honglak Lee and Alexis Battle and Rajat Raina and Andrew Y. Ng},
  booktitle = {Advances in Neural Information Processing Systems 19},
  editor = {B. Schölkopf and J. Platt and T. Hoffman},
  publisher = {MIT Press},
  address = {Cambridge, MA},
  pages = {801--808},
  year = {2007}
}

@article{efron2004least,
  title={Least angle regression},
  author={Efron, B. and Hastie, T. and Johnstone, I. and Tibshirani, R.},
  journal={The Annals of Statistics},
  volume={32},
  number={2},
  pages={407--499},
  year={2004},
  publisher={Institute of Mathematical Statistics}
}

@article{zou2005regularization,
  title={Regularization and variable selection via the elastic net},
  author={Zou, H. and Hastie, T.},
  journal={Journal of the Royal Statistical Society Series B},
  volume={67},
  number={2},
  pages={301--320},
  year={2005},
  publisher={Royal Statistical Society}
}
```

Before the method is run, the dictionary is initialized using the `DictionaryInitializationPolicy` class. Possible choices include the **RandomInitializer** (p. 546), which provides an entirely random dictionary, the **DataDependentRandomInitializer** (p. 545), which provides a random dictionary based loosely on characteristics of the dataset, and the **NothingInitializer** (p. 546), which does not initialize the dictionary – instead, the user should set the dictionary using the **Dictionary()** (p. 550) mutator method.

Template Parameters

<i>DictionaryInitializationPolicy</i>	The class to use to initialize the dictionary; must have 'void Initialize(const arma::mat& data, arma::mat& dictionary)' function.
---------------------------------------	--

Definition at line 111 of file `sparse_coding.hpp`.

22.133.2 Constructor & Destructor Documentation

22.133.2.1 `template<typename DictionaryInitializer = DataDependentRandomInitializer> mlpack::sparse_coding::SparseCoding< DictionaryInitializer >::SparseCoding (const arma::mat & data, const size_t atoms, const double lambda1, const double lambda2 = 0)`

Set the parameters to **SparseCoding** (p. 547).

`lambda2` defaults to 0.

Parameters

<i>data</i>	Data matrix
<i>atoms</i>	Number of atoms in dictionary
<i>lambda1</i>	Regularization parameter for l1-norm penalty
<i>lambda2</i>	Regularization parameter for l2-norm penalty

22.133.3 Member Function Documentation

22.133.3.1 `template<typename DictionaryInitializer = DataDependentRandomInitializer> const arma::mat& mlpack::sparse_coding::SparseCoding< DictionaryInitializer >::Codes () const [inline]`

Access the sparse codes.

Definition at line 182 of file `sparse_coding.hpp`.

References `mlpack::sparse_coding::SparseCoding< DictionaryInitializer >::codes`.

22.133.3.2 `template<typename DictionaryInitializer = DataDependentRandomInitializer> arma::mat& mlpack::sparse_coding::SparseCoding< DictionaryInitializer >::Codes () [inline]`

Modify the sparse codes.

Definition at line 184 of file `sparse_coding.hpp`.

References `mlpack::sparse_coding::SparseCoding< DictionaryInitializer >::codes`.

22.133.3.3 `template<typename DictionaryInitializer = DataDependentRandomInitializer> const arma::mat& mlpack::sparse_coding::SparseCoding< DictionaryInitializer >::Data () const [inline]`

Access the data.

Definition at line 174 of file `sparse_coding.hpp`.

References `mlpack::sparse_coding::SparseCoding< DictionaryInitializer >::data`.

22.133.3.4 `template<typename DictionaryInitializer = DataDependentRandomInitializer> const arma::mat& mlpack::sparse_coding::SparseCoding< DictionaryInitializer >::Dictionary () const [inline]`

Access the dictionary.

Definition at line 177 of file `sparse_coding.hpp`.

References `mlpack::sparse_coding::SparseCoding< DictionaryInitializer >::dictionary`.

22.133.3.5 `template<typename DictionaryInitializer = DataDependentRandomInitializer> arma::mat& mlpack::sparse_coding::SparseCoding< DictionaryInitializer >::Dictionary () [inline]`

Modify the dictionary.

Definition at line 179 of file `sparse_coding.hpp`.

References `mlpack::sparse_coding::SparseCoding< DictionaryInitializer >::dictionary`.

22.133.3.6 `template<typename DictionaryInitializer = DataDependentRandomInitializer> void mlpack::sparse_coding::`
`SparseCoding< DictionaryInitializer >::Encode (const size_t maxIterations = 0, const double objTolerance =`
`0.01, const double newtonTolerance = 1e-6)`

Run Sparse Coding with Dictionary Learning.

Parameters

<i>maxIterations</i>	Maximum number of iterations to run algorithm. If 0, the algorithm will run until convergence (or forever).
<i>objTolerance</i>	Tolerance for objective function. When an iteration of the algorithm produces an improvement smaller than this, the algorithm will terminate.
<i>newtonTolerance</i>	Tolerance for the Newton's method dictionary optimization step.

22.133.3.7 `template<typename DictionaryInitializer = DataDependentRandomInitializer> double
mlpack::sparse_coding::SparseCoding< DictionaryInitializer >::Objective () const`

Compute the objective function.

22.133.3.8 `template<typename DictionaryInitializer = DataDependentRandomInitializer> void mlpack::sparse_coding::
SparseCoding< DictionaryInitializer >::OptimizeCode ()`

Sparse code each point via LARS.

22.133.3.9 `template<typename DictionaryInitializer = DataDependentRandomInitializer> double mlpack::sparse_coding::
SparseCoding< DictionaryInitializer >::OptimizeDictionary (const arma::uvec & adjacencies, const double
newtonTolerance = 1e-6, const size_t maxIterations = 50)`

Learn dictionary via Newton method based on Lagrange dual.

Parameters

<i>adjacencies</i>	Indices of entries (unrolled column by column) of the coding matrix Z that are non-zero (the adjacency matrix for the bipartite graph of points and atoms).
<i>newtonTolerance</i>	Tolerance of the Newton's method optimizer.
<i>maxIterations</i>	Maximum number of iterations to run the Newton's method. If 0, the method will run until convergence (or forever).

Returns

the norm of the gradient of the Lagrange dual with respect to the dual variables

22.133.3.10 `template<typename DictionaryInitializer = DataDependentRandomInitializer> void
mlpack::sparse_coding::SparseCoding< DictionaryInitializer >::ProjectDictionary ()`

Project each atom of the dictionary back onto the unit ball, if necessary.

22.133.3.11 `template<typename DictionaryInitializer = DataDependentRandomInitializer> std::string
mlpack::sparse_coding::SparseCoding< DictionaryInitializer >::ToString () const`

22.133.4 Member Data Documentation

22.133.4.1 `template<typename DictionaryInitializer = DataDependentRandomInitializer> size_t
mlpack::sparse_coding::SparseCoding< DictionaryInitializer >::atoms [private]`

Number of atoms.

Definition at line 191 of file sparse_coding.hpp.

22.133.4.2 `template<typename DictionaryInitializer = DataDependentRandomInitializer> arma::mat
mlpack::sparse_coding::SparseCoding< DictionaryInitializer >::codes [private]`

Sparse codes (columns are points).

Definition at line 200 of file sparse_coding.hpp.

Referenced by `mlpack::sparse_coding::SparseCoding< DictionaryInitializer >::Codes()`.

22.133.4.3 `template<typename DictionaryInitializer = DataDependentRandomInitializer> const arma::mat&
mlpack::sparse_coding::SparseCoding< DictionaryInitializer >::data [private]`

Data matrix (columns are points).

Definition at line 194 of file sparse_coding.hpp.

Referenced by `mlpack::sparse_coding::SparseCoding< DictionaryInitializer >::Data()`.

22.133.4.4 `template<typename DictionaryInitializer = DataDependentRandomInitializer> arma::mat
mlpack::sparse_coding::SparseCoding< DictionaryInitializer >::dictionary [private]`

Dictionary (columns are atoms).

Definition at line 197 of file sparse_coding.hpp.

Referenced by `mlpack::sparse_coding::SparseCoding< DictionaryInitializer >::Dictionary()`.

22.133.4.5 `template<typename DictionaryInitializer = DataDependentRandomInitializer> double
mlpack::sparse_coding::SparseCoding< DictionaryInitializer >::lambda1 [private]`

l1 regularization term.

Definition at line 203 of file sparse_coding.hpp.

22.133.4.6 `template<typename DictionaryInitializer = DataDependentRandomInitializer> double
mlpack::sparse_coding::SparseCoding< DictionaryInitializer >::lambda2 [private]`

l2 regularization term.

Definition at line 206 of file sparse_coding.hpp.

The documentation for this class was generated from the following file:

- `src/mlpack/methods/sparse_coding/sparse_coding.hpp`

22.134 mlpack::svd::QUIC_SVD Class Reference

Public Member Functions

- **QUIC_SVD** (const arma::mat &**dataset**, arma::mat &u, arma::mat &v, arma::mat &sigma, const double **epsilon**=0.03, const double **delta**=0.1)

Constructor which implements the QUIC-SVD algorithm.

- void **ExtractSVD** (arma::mat &u, arma::mat &v, arma::mat &sigma)

This function uses the vector subspace created using a cosine tree to calculate an approximate SVD of the original matrix.

Private Attributes

- arma::mat **basis**

Subspace basis of the input dataset.

- const arma::mat & **dataset**

Matrix for which cosine tree is constructed.

- double **delta**

Cumulative probability for Monte Carlo error lower bound.

- double **epsilon**

Error tolerance fraction for calculated subspace.

22.134.1 Detailed Description

Definition at line 23 of file quic_svd.hpp.

22.134.2 Constructor & Destructor Documentation

22.134.2.1 `mlpack::svd::QUIC_SVD::QUIC_SVD (const arma::mat & dataset, arma::mat & u, arma::mat & v, arma::mat & sigma, const double epsilon = 0.03, const double delta = 0.1)`

Constructor which implements the QUIC-SVD algorithm.

The function calls the CosineTree constructor to create a subspace basis, where the original matrix's projection has minimum reconstruction error. The constructor then uses the **ExtractSVD()** (p. 554) function to calculate the SVD of the original dataset in that subspace.

Parameters

<i>dataset</i>	Matrix for which SVD is calculated.
<i>u</i>	First unitary matrix.
<i>v</i>	Second unitary matrix.
<i>sigma</i>	Diagonal matrix of singular values.
<i>epsilon</i>	Error tolerance fraction for calculated subspace.
<i>delta</i>	Cumulative probability for Monte Carlo error lower bound.

22.134.3 Member Function Documentation

22.134.3.1 `void mlpack::svd::QUIC_SVD::ExtractSVD (arma::mat & u, arma::mat & v, arma::mat & sigma)`

This function uses the vector subspace created using a cosine tree to calculate an approximate SVD of the original matrix.

Parameters

u	First unitary matrix.
v	Second unitary matrix.
σ	Diagonal matrix of singular values.

22.134.4 Member Data Documentation

22.134.4.1 arma::mat mlpack::svd::QUIC_SVD::basis [private]

Subspace basis of the input dataset.

Definition at line 68 of file quic_svd.hpp.

22.134.4.2 const arma::mat& mlpack::svd::QUIC_SVD::dataset [private]

Matrix for which cosine tree is constructed.

Definition at line 62 of file quic_svd.hpp.

22.134.4.3 double mlpack::svd::QUIC_SVD::delta [private]

Cumulative probability for Monte Carlo error lower bound.

Definition at line 66 of file quic_svd.hpp.

22.134.4.4 double mlpack::svd::QUIC_SVD::epsilon [private]

Error tolerance fraction for calculated subspace.

Definition at line 64 of file quic_svd.hpp.

The documentation for this class was generated from the following file:

- src/mlpack/methods/quic_svd/quic_svd.hpp

22.135 mlpack::svd::RegularizedSVD< OptimizerType > Class Template Reference

Collaboration diagram for mlpack::svd::RegularizedSVD< OptimizerType >:



Public Member Functions

- **RegularizedSVD** (const arma::mat &data, arma::mat &u, arma::mat &v, const size_t rank, const size_t iterations=10, const double alpha=0.01, const double lambda=0.02)

Constructor for Regularized SVD.

Private Attributes

- double **alpha**
Learning rate for the SGD optimizer.
- const arma::mat & **data**
Rating data.
- size_t **iterations**
Number of optimization iterations.
- double **lambda**
Regularization parameter for the optimization.
- **mlpack::optimization::SGD**< **RegularizedSVDFunction** > **optimizer**
Default SGD optimizer for the class.
- size_t **rank**
Rank used for matrix factorization.
- **RegularizedSVDFunction** **rSVDFunc**
Function that will be held by the optimizer.

22.135.1 Detailed Description

```
template<template< typename > class OptimizerType = mlpack::optimization::SGD> class mlpack::svd::RegularizedSVD<
OptimizerType >
```

Definition at line 29 of file regularized_svd.hpp.

22.135.2 Constructor & Destructor Documentation

22.135.2.1 `template<template< typename > class OptimizerType = mlpack::optimization::SGD> mlpack::svd::RegularizedSVD< OptimizerType >::RegularizedSVD(const arma::mat & data, arma::mat & u, arma::mat & v, const size_t rank, const size_t iterations = 10, const double alpha = 0.01, const double lambda = 0.02)`

Constructor for Regularized SVD.

Obtains the user and item matrices after training on the passed data. The constructor initiates an object of class **RegularizedSVDFunction** (p. 557) for optimization. It uses the SGD optimizer by default. The optimizer uses a template specialization of `Optimize()`.

Parameters

<i>data</i>	Dataset for which SVD is calculated.
<i>u</i>	User matrix in the matrix decomposition.
<i>v</i>	Item matrix in the matrix decomposition.
<i>rank</i>	Rank used for matrix factorization.
<i>iterations</i>	Number of optimization iterations.
<i>lambda</i>	Regularization parameter for the optimization.

22.135.3 Member Data Documentation

22.135.3.1 `template<template< typename > class OptimizerType = mlpack::optimization::SGD> double mlpack::svd::RegularizedSVD< OptimizerType >::alpha [private]`

Learning rate for the SGD optimizer.

Definition at line 62 of file regularized_svd.hpp.

22.135.3.2 `template<template< typename > class OptimizerType = mlpack::optimization::SGD> const arma::mat& mlpack::svd::RegularizedSVD< OptimizerType >::data [private]`

Rating data.

Definition at line 56 of file regularized_svd.hpp.

22.135.3.3 `template<template< typename > class OptimizerType = mlpack::optimization::SGD> size_t mlpack::svd::RegularizedSVD< OptimizerType >::iterations [private]`

Number of optimization iterations.

Definition at line 60 of file regularized_svd.hpp.

22.135.3.4 `template<template< typename > class OptimizerType = mlpack::optimization::SGD> double mlpack::svd::RegularizedSVD< OptimizerType >::lambda [private]`

Regularization parameter for the optimization.

Definition at line 64 of file regularized_svd.hpp.

22.135.3.5 `template<template< typename > class OptimizerType = mlpack::optimization::SGD> mlpack::optimization::SGD<RegularizedSVDFunction> mlpack::svd::RegularizedSVD< OptimizerType >::optimizer [private]`

Default SGD optimizer for the class.

Definition at line 68 of file regularized_svd.hpp.

22.135.3.6 `template<template< typename > class OptimizerType = mlpack::optimization::SGD> size_t mlpack::svd::RegularizedSVD< OptimizerType >::rank [private]`

Rank used for matrix factorization.

Definition at line 58 of file regularized_svd.hpp.

22.135.3.7 `template<template< typename > class OptimizerType = mlpack::optimization::SGD> RegularizedSVDFunction mlpack::svd::RegularizedSVD< OptimizerType >::rSVDFunc [private]`

Function that will be held by the optimizer.

Definition at line 66 of file regularized_svd.hpp.

The documentation for this class was generated from the following file:

- src/mlpack/methods/regularized_svd/regularized_svd.hpp

22.136 mlpack::svd::RegularizedSVDFunction Class Reference

Public Member Functions

- **RegularizedSVDFunction** (const arma::mat &**data**, const size_t **rank**, const double **lambda**)

*Constructor for **RegularizedSVDFunction** (p. 557) class.*

- const arma::mat & **Dataset** () const

Return the dataset passed into the constructor.

- double **Evaluate** (const arma::mat ¶meters) const

Evaluates the cost function over all examples in the data.

- double **Evaluate** (const arma::mat ¶meters, const size_t i) const

Evaluates the cost function for one training example.

- const arma::mat & **GetInitialPoint** () const

Return the initial point for the optimization.

- void **Gradient** (const arma::mat ¶meters, arma::mat &gradient) const

Evaluates the full gradient of the cost function over all the training examples.

- double **Lambda** () const

Return the regularization parameters.

- size_t **NumFunctions** () const

Return the number of training examples. Useful for SGD optimizer.

- size_t **NumItems** () const

Return the number of items in the data.

- size_t **NumUsers** () const

Return the number of users in the data.

- size_t **Rank** () const

Return the rank used for the factorization.

Private Attributes

- const arma::mat & **data**

Rating data.

- arma::mat **initialPoint**

Initial parameter point.

- double **lambda**

Regularization parameter for the optimization.

- size_t **numItems**

Number of items in the given dataset.

- size_t **numUsers**

Number of users in the given dataset.

- size_t **rank**

Rank used for matrix factorization.

22.136.1 Detailed Description

Definition at line 24 of file regularized_svd_function.hpp.

22.136.2 Constructor & Destructor Documentation

22.136.2.1 `mlpack::svd::RegularizedSVDFunction::RegularizedSVDFunction (const arma::mat & data, const size_t rank, const double lambda)`

Constructor for **RegularizedSVDFunction** (p. 557) class.

The constructor calculates the number of users and items in the passed data. It also randomly initializes the parameter values.

Parameters

<i>data</i>	Dataset for which SVD is calculated.
<i>rank</i>	Rank used for matrix factorization.
<i>lambda</i>	Regularization parameter used for optimization.

22.136.3 Member Function Documentation

22.136.3.1 `const arma::mat& mlpack::svd::RegularizedSVDFunction::Dataset () const [inline]`

Return the dataset passed into the constructor.

Definition at line 72 of file `regularized_svd_function.hpp`.

References `data`.

22.136.3.2 `double mlpack::svd::RegularizedSVDFunction::Evaluate (const arma::mat & parameters) const`

Evaluates the cost function over all examples in the data.

Parameters

<i>parameters</i>	Parameters(user/item matrices) of the decomposition.
-------------------	--

22.136.3.3 `double mlpack::svd::RegularizedSVDFunction::Evaluate (const arma::mat & parameters, const size_t i) const`

Evaluates the cost function for one training example.

Useful for the SGD optimizer abstraction which uses one training example at a time.

Parameters

<i>parameters</i>	Parameters(user/item matrices) of the decomposition.
<i>i</i>	Index of the training example to be used.

22.136.3.4 `const arma::mat& mlpack::svd::RegularizedSVDFunction::GetInitialPoint () const [inline]`

Return the initial point for the optimization.

Definition at line 69 of file `regularized_svd_function.hpp`.

References `initialPoint`.

22.136.3.5 `void mlpack::svd::RegularizedSVDFunction::Gradient (const arma::mat & parameters, arma::mat & gradient) const`

Evaluates the full gradient of the cost function over all the training examples.

Parameters

<i>parameters</i>	Parameters(user/item matrices) of the decomposition.
<i>gradient</i>	Calculated gradient for the parameters.

22.136.3.6 `double mlpack::svd::RegularizedSVDFunction::Lambda () const` `[inline]`

Return the regularization parameters.

Definition at line 84 of file `regularized_svd_function.hpp`.

References `lambda`.

22.136.3.7 `size_t mlpack::svd::RegularizedSVDFunction::NumFunctions () const` `[inline]`

Return the number of training examples. Useful for SGD optimizer.

Definition at line 75 of file `regularized_svd_function.hpp`.

22.136.3.8 `size_t mlpack::svd::RegularizedSVDFunction::NumItems () const` `[inline]`

Return the number of items in the data.

Definition at line 81 of file `regularized_svd_function.hpp`.

References `numItems`.

22.136.3.9 `size_t mlpack::svd::RegularizedSVDFunction::NumUsers () const` `[inline]`

Return the number of users in the data.

Definition at line 78 of file `regularized_svd_function.hpp`.

References `numUsers`.

22.136.3.10 `size_t mlpack::svd::RegularizedSVDFunction::Rank () const` `[inline]`

Return the rank used for the factorization.

Definition at line 87 of file `regularized_svd_function.hpp`.

References `rank`.

22.136.4 Member Data Documentation

22.136.4.1 `const arma::mat& mlpack::svd::RegularizedSVDFunction::data` `[private]`

Rating data.

Definition at line 91 of file `regularized_svd_function.hpp`.

Referenced by `Dataset()`.

22.136.4.2 `arma::mat mlpack::svd::RegularizedSVDFunction::initialPoint` `[private]`

Initial parameter point.

Definition at line 93 of file `regularized_svd_function.hpp`.

Referenced by `GetInitialPoint()`.

22.136.4.3 `double mlpack::svd::RegularizedSVDFunction::lambda` `[private]`

Regularization parameter for the optimization.

Definition at line 97 of file `regularized_svd_function.hpp`.

Referenced by `Lambda()`.

22.136.4.4 `size_t mlpack::svd::RegularizedSVDFunction::numItems` `[private]`

Number of items in the given dataset.

Definition at line 101 of file `regularized_svd_function.hpp`.

Referenced by `NumItems()`.

22.136.4.5 `size_t mlpack::svd::RegularizedSVDFunction::numUsers` `[private]`

Number of users in the given dataset.

Definition at line 99 of file `regularized_svd_function.hpp`.

Referenced by `NumUsers()`.

22.136.4.6 `size_t mlpack::svd::RegularizedSVDFunction::rank` `[private]`

Rank used for matrix factorization.

Definition at line 95 of file `regularized_svd_function.hpp`.

Referenced by `Rank()`.

The documentation for this class was generated from the following file:

- `src/mlpack/methods/regularized_svd/regularized_svd_function.hpp`

22.137 mlpack::Timer Class Reference

The timer class provides a way for MLPACK methods to be timed.

Static Public Member Functions

- static timeval **Get** (const std::string &name)
Get the value of the given timer.
- static void **Start** (const std::string &name)
Start the given timer.

- static void **Stop** (const std::string &name)

Stop the given timer.

22.137.1 Detailed Description

The timer class provides a way for MLPACK methods to be timed.

The three methods contained in this class allow a named timer to be started and stopped, and its value to be obtained.

Definition at line 65 of file timers.hpp.

22.137.2 Member Function Documentation

22.137.2.1 static timeval mlpack::Timer::Get (const std::string & *name*) [static]

Get the value of the given timer.

Parameters

<i>name</i>	Name of timer to return value of.
-------------	-----------------------------------

22.137.2.2 static void mlpack::Timer::Start (const std::string & *name*) [static]

Start the given timer.

If a timer is started, then stopped, then re-started, then re-stopped, the final value of the timer is the length of both runs – that is, MLPACK timers are additive for each time they are run, and do not reset.

Note

Undefined behavior will occur if a timer is started twice.

Parameters

<i>name</i>	Name of timer to be started.
-------------	------------------------------

22.137.2.3 static void mlpack::Timer::Stop (const std::string & *name*) [static]

Stop the given timer.

Note

Undefined behavior will occur if a timer is started twice.

Parameters

<i>name</i>	Name of timer to be stopped.
-------------	------------------------------

The documentation for this class was generated from the following file:

- src/mlpack/core/util/**timers.hpp**

22.138 mpack::Timers Class Reference

Public Member Functions

- **Timers** ()
Nothing to do for the constructor.
- `std::map< std::string, timeval > & GetAllTimers` ()
Returns a copy of all the timers used via this interface.
- `timeval GetTimer` (const std::string &timerName)
Returns a copy of the timer specified.
- `void PrintTimer` (const std::string &timerName)
Prints the specified timer.
- `void StartTimer` (const std::string &timerName)
** Initializes a timer, available like a normal value specified on * the command line.*
- `void StopTimer` (const std::string &timerName)
** Halts the timer, and replaces it's value with * the delta time from it's start * *.*

Private Member Functions

- `void FileTimeToTimeVal` (timeval *tv)
- `void GetTime` (timeval *tv)

Private Attributes

- `std::map< std::string, timeval > timers`

22.138.1 Detailed Description

Definition at line 97 of file timers.hpp.

22.138.2 Constructor & Destructor Documentation

22.138.2.1 mpack::Timers::Timers () [inline]

Nothing to do for the constructor.

Definition at line 101 of file timers.hpp.

22.138.3 Member Function Documentation

22.138.3.1 void mpack::Timers::FileTimeToTimeVal (timeval * tv) [private]

22.138.3.2 std::map<std::string, timeval>& mpack::Timers::GetAllTimers ()

Returns a copy of all the timers used via this interface.

22.138.3.3 void mpack::Timers::GetTime (timeval * *tv*) [private]

22.138.3.4 timeval mpack::Timers::GetTimer (const std::string & *timerName*)

Returns a copy of the timer specified.

Parameters

<i>timerName</i>	The name of the timer in question.
------------------	------------------------------------

22.138.3.5 void mlpack::Timers::PrintTimer (const std::string & *timerName*)

Prints the specified timer.

If it took longer than a minute to complete the timer will be displayed in days, hours, and minutes as well.

Parameters

<i>timerName</i>	The name of the timer in question.
------------------	------------------------------------

22.138.3.6 void mlpack::Timers::StartTimer (const std::string & *timerName*)

* Initializes a timer, available like a normal value specified on * the command line.

Timers (p. 564) are of type timeval. If a timer is started, then stopped, then re-started, then stopped, the final timer value will be the length of both runs of the timer. * *

Parameters

<i>timerName</i>	The name of the timer in question.
------------------	------------------------------------

22.138.3.7 void mlpack::Timers::StopTimer (const std::string & *timerName*)

* Halts the timer, and replaces it's value with * the delta time from it's start * *.

Parameters

<i>timerName</i>	The name of the timer in question.
------------------	------------------------------------

22.138.4 Member Data Documentation

22.138.4.1 std::map<std::string, timeval> mlpack::Timers::timers [private]

Definition at line 142 of file timers.hpp.

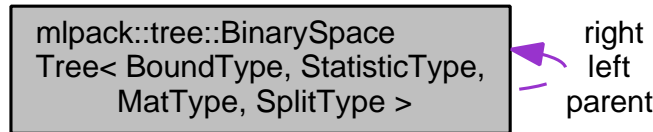
The documentation for this class was generated from the following file:

- src/mlpack/core/util/**timers.hpp**

22.139 mlpack::tree::BinarySpaceTree< BoundType, StatisticType, MatType, SplitType > Class Template Reference

A binary space partitioning tree, such as a KD-tree or a ball tree.

Collaboration diagram for mlpack::tree::BinarySpaceTree< BoundType, StatisticType, MatType, SplitType >:



Classes

- class **DualTreeTraverser**
A dual-tree traverser for binary space trees; see dual_tree_traverser.hpp.
- class **SingleTreeTraverser**
A single-tree traverser for binary space trees; see single_tree_traverser.hpp for implementation.

Public Types

- typedef MatType **Mat**
So other classes can use TreeType::Mat.

Public Member Functions

- **BinarySpaceTree** (MatType &data, const size_t **maxLeafSize**=20)
Construct this as the root node of a binary space tree using the given dataset.
- **BinarySpaceTree** (MatType &data, std::vector< size_t > &oldFromNew, const size_t **maxLeafSize**=20)
Construct this as the root node of a binary space tree using the given dataset.
- **BinarySpaceTree** (MatType &data, std::vector< size_t > &oldFromNew, std::vector< size_t > &newFromOld, const size_t **maxLeafSize**=20)
Construct this as the root node of a binary space tree using the given dataset.
- **BinarySpaceTree** (MatType &data, const size_t **begin**, const size_t **count**, **BinarySpaceTree** *parent=NULL, const size_t **maxLeafSize**=20)
Construct this node on a subset of the given matrix, starting at column begin and using count points.
- **BinarySpaceTree** (MatType &data, const size_t **begin**, const size_t **count**, std::vector< size_t > &oldFromNew, **BinarySpaceTree** *parent=NULL, const size_t **maxLeafSize**=20)
Construct this node on a subset of the given matrix, starting at column begin_in and using count_in points.
- **BinarySpaceTree** (MatType &data, const size_t **begin**, const size_t **count**, std::vector< size_t > &oldFromNew, std::vector< size_t > &newFromOld, **BinarySpaceTree** *parent=NULL, const size_t **maxLeafSize**=20)
Construct this node on a subset of the given matrix, starting at column begin_in and using count_in points.
- **BinarySpaceTree** (const **BinarySpaceTree** &other)
Create a binary space tree by copying the other tree.
- ~**BinarySpaceTree** ()
Deletes this node, deallocating the memory for the children and calling their destructors in turn.
- size_t **Begin** () const
Return the index of the beginning point of this subset.
- size_t & **Begin** ()
Modify the index of the beginning point of this subset.
- const BoundType & **Bound** () const

- Return the bound object for this node.*

 - **BoundType & Bound ()**

Return the bound object for this node.
- void **Centroid** (arma::vec ¢roid)

Get the centroid of the node and store it in the given vector.
- **BinarySpaceTree & Child** (const size_t child) const

Return the specified child (0 will be left, 1 will be right).
- size_t **Count** () const

Return the number of points in this subset.
- size_t & **Count** ()

Modify the number of points in this subset.
- const MatType & **Dataset** () const

Get the dataset which the tree is built on.
- MatType & **Dataset** ()

Modify the dataset which the tree is built on. Be careful!
- size_t **Descendant** (const size_t index) const

Return the index (with reference to the dataset) of a particular descendant of this node.
- size_t **End** () const

Gets the index one beyond the last index in the subset.
- size_t **ExtendTree** (const size_t level)

Fills the tree to the specified level.
- const **BinarySpaceTree * FindByBeginCount** (size_t begin, size_t count) const

Find a node in this tree by its begin and count (const).
- **BinarySpaceTree * FindByBeginCount** (size_t begin, size_t count)

Find a node in this tree by its begin and count.
- double **FurthestDescendantDistance** () const

Return the furthest possible descendant distance.
- double **FurthestPointDistance** () const

Return the furthest distance to a point held in this node.
- size_t **GetSplitDimension** () const

Returns the dimension this parent's children are split on.
- bool **IsLeaf** () const

Return whether or not this node is a leaf (true if it has no children).
- **BinarySpaceTree * Left** () const

Gets the left child of this node.
- **BinarySpaceTree *& Left** ()

Modify the left child of this node.
- double **MaxDistance** (const **BinarySpaceTree** *other) const

Return the maximum distance to another node.
- template<typename VecType >
double **MaxDistance** (const VecType &point, typename boost::enable_if< **IsVector**< VecType > >::type *=0)
const

Return the maximum distance to another point.
- size_t **MaxLeafSize** () const

Return the max leaf size.
- size_t & **MaxLeafSize** ()

Modify the max leaf size.

- BoundType::MetricType **Metric** () const
Get the metric which the tree uses.
- double **MinDistance** (const **BinarySpaceTree** *other) const
Return the minimum distance to another node.
- template<typename VecType >
double **MinDistance** (const VecType &point, typename boost::enable_if< **IsVector**< VecType > >::type *=0)
const
Return the minimum distance to another point.
- double **MinimumBoundDistance** () const
Return the minimum distance from the center of the node to any bound edge.
- size_t **NumChildren** () const
Return the number of children in this node.
- size_t **NumDescendants** () const
Return the number of descendants of this node.
- size_t **NumPoints** () const
Return the number of points in this node (0 if not a leaf).
- **BinarySpaceTree** * **Parent** () const
Gets the parent of this node.
- **BinarySpaceTree** *& **Parent** ()
Modify the parent of this node.
- double **ParentDistance** () const
Return the distance from the center of this node to the center of the parent node.
- double & **ParentDistance** ()
Modify the distance from the center of this node to the center of the parent node.
- size_t **Point** (const size_t index) const
Return the index (with reference to the dataset) of a particular point in this node.
- **math::Range** **RangeDistance** (const **BinarySpaceTree** *other) const
Return the minimum and maximum distance to another node.
- template<typename VecType >
math::Range **RangeDistance** (const VecType &point, typename boost::enable_if< **IsVector**< VecType > >::type *=0) const
Return the minimum and maximum distance to another point.
- **BinarySpaceTree** * **Right** () const
Gets the right child of this node.
- **BinarySpaceTree** *& **Right** ()
Modify the right child of this node.
- size_t **SplitDimension** () const
Get the split dimension for this node.
- size_t & **SplitDimension** ()
Modify the split dimension for this node.
- const StatisticType & **Stat** () const
Return the statistic object for this node.
- StatisticType & **Stat** ()
Return the statistic object for this node.
- std::string **ToString** () const
Returns a string representation of this object.
- size_t **TreeDepth** () const
Obtains the number of levels below this node in the tree, starting with this.
- size_t **TreeSize** () const
Obtains the number of nodes in the tree, starting with this.

Static Public Member Functions

- static bool **HasSelfChildren** ()
Returns false: this tree type does not have self children.

Private Member Functions

- **BinarySpaceTree** (const size_t **begin**, const size_t **count**, BoundType **bound**, StatisticType **stat**, const int **maxLeafSize**=20)
Private copy constructor, available only to fill (pad) the tree to a specified level.
- **BinarySpaceTree** * **CopyMe** ()
- void **SplitNode** (MatType &data)
Splits the current node, assigning its left and right children recursively.
- void **SplitNode** (MatType &data, std::vector< size_t > &oldFromNew)
Splits the current node, assigning its left and right children recursively.

Private Attributes

- size_t **begin**
The index of the first point in the dataset contained in this node (and its children).
- BoundType **bound**
The bound object for this node.
- size_t **count**
The number of points of the dataset contained in this node (and its children).
- MatType & **dataset**
The dataset.
- double **furthestDescendantDistance**
The worst possible distance to the furthest descendant, cached to speed things up.
- **BinarySpaceTree** * **left**
The left child node.
- size_t **maxLeafSize**
The max leaf size.
- double **minimumBoundDistance**
The minimum distance from the center to any edge of the bound.
- **BinarySpaceTree** * **parent**
The parent node (NULL if this is the root of the tree).
- double **parentDistance**
The distance from the centroid of this node to the centroid of the parent.
- **BinarySpaceTree** * **right**
The right child node.
- size_t **splitDimension**
The dimension this node split on if it is a parent.
- StatisticType **stat**
Any extra data contained in the node.

22.139.1 Detailed Description

```
template<typename BoundType, typename StatisticType = EmptyStatistic, typename MatType = arma::mat, typename SplitType = MeanSplit<BoundType, MatType>> class mlpack::tree::BinarySpaceTree< BoundType, StatisticType, MatType, SplitType >
```

A binary space partitioning tree, such as a KD-tree or a ball tree.

Once the bound and type of dataset is defined, the tree will construct itself. Call the constructor with the dataset to build the tree on, and the entire tree will be built.

This particular tree does not allow growth, so you cannot add or delete nodes from it. If you need to add or delete a node, the better procedure is to rebuild the tree entirely.

This tree does take one runtime parameter in the constructor, which is the max leaf size to be used.

Template Parameters

<i>BoundType</i>	The bound used for each node. The valid types of bounds and the necessary skeleton interface for this class can be found in <code>bounds/</code> .
<i>StatisticType</i>	Extra data contained in the node. See statistic.hpp (p. 755) for the necessary skeleton interface.
<i>MatType</i>	The dataset class.
<i>SplitType</i>	The class that partitions the dataset/points at a particular node into two parts. Its definition decides the way this split is done.

Definition at line 51 of file `binary_space_tree.hpp`.

22.139.2 Member Typedef Documentation

22.139.2.1 `template<typename BoundType , typename StatisticType = EmptyStatistic, typename MatType = arma::mat, typename SplitType = MeanSplit<BoundType, MatType>> typedef MatType mlpack::tree::BinarySpaceTree< BoundType, StatisticType, MatType, SplitType >::Mat`

So other classes can use `TreeType::Mat`.

Definition at line 86 of file `binary_space_tree.hpp`.

22.139.3 Constructor & Destructor Documentation

22.139.3.1 `template<typename BoundType , typename StatisticType = EmptyStatistic, typename MatType = arma::mat, typename SplitType = MeanSplit<BoundType, MatType>> mlpack::tree::BinarySpaceTree< BoundType, StatisticType, MatType, SplitType >::BinarySpaceTree(MatType & data, const size_t maxLeafSize = 20)`

Construct this as the root node of a binary space tree using the given dataset.

This will modify the ordering of the points in the dataset!

Parameters

<i>data</i>	Dataset to create tree from. This will be modified!
<i>maxLeafSize</i>	Size of each leaf in the tree.

Referenced by `mlpack::tree::BinarySpaceTree< BoundType, StatisticType, MatType, SplitType >::CopyMe()`.

```
22.139.3.2  template<typename BoundType , typename StatisticType = EmptyStatistic, typename MatType = arma::mat, typename
            SplitType = MeanSplit<BoundType, MatType>> mlpack::tree::BinarySpaceTree< BoundType, StatisticType,
            MatType, SplitType >::BinarySpaceTree( MatType & data, std::vector< size_t > & oldFromNew, const size_t
            maxLeafSize = 20 )
```

Construct this as the root node of a binary space tree using the given dataset.

This will modify the ordering of points in the dataset! A mapping of the old point indices to the new point indices is filled.

Parameters

<i>data</i>	Dataset to create tree from. This will be modified!
<i>oldFromNew</i>	Vector which will be filled with the old positions for each new point.
<i>maxLeafSize</i>	Size of each leaf in the tree.

```
22.139.3.3  template<typename BoundType , typename StatisticType = EmptyStatistic, typename MatType = arma::mat, typename
            SplitType = MeanSplit<BoundType, MatType>> mlpack::tree::BinarySpaceTree< BoundType, StatisticType,
            MatType, SplitType >::BinarySpaceTree( MatType & data, std::vector< size_t > & oldFromNew, std::vector<
            size_t > & newFromOld, const size_t maxLeafSize = 20 )
```

Construct this as the root node of a binary space tree using the given dataset.

This will modify the ordering of points in the dataset! A mapping of the old point indices to the new point indices is filled, as well as a mapping of the new point indices to the old point indices.

Parameters

<i>data</i>	Dataset to create tree from. This will be modified!
<i>oldFromNew</i>	Vector which will be filled with the old positions for each new point.
<i>newFromOld</i>	Vector which will be filled with the new positions for each old point.
<i>maxLeafSize</i>	Size of each leaf in the tree.

```
22.139.3.4  template<typename BoundType , typename StatisticType = EmptyStatistic, typename MatType = arma::mat,
            typename SplitType = MeanSplit<BoundType, MatType>> mlpack::tree::BinarySpaceTree< BoundType,
            StatisticType, MatType, SplitType >::BinarySpaceTree( MatType & data, const size_t begin, const size_t count,
            BinarySpaceTree< BoundType, StatisticType, MatType, SplitType > * parent = NULL, const size_t maxLeafSize =
            20 )
```

Construct this node on a subset of the given matrix, starting at column begin and using count points.

The ordering of that subset of points will be modified! This is used for recursive tree-building by the other constructors which don't specify point indices.

Parameters

<i>data</i>	Dataset to create tree from. This will be modified!
<i>begin</i>	Index of point to start tree construction with.
<i>count</i>	Number of points to use to construct tree.
<i>maxLeafSize</i>	Size of each leaf in the tree.


```
22.139.3.5  template<typename BoundType , typename StatisticType = EmptyStatistic, typename MatType = arma::mat, typename
SplitType = MeanSplit<BoundType, MatType>> mlpack::tree::BinarySpaceTree< BoundType, StatisticType,
MatType, SplitType >::BinarySpaceTree( MatType & data, const size_t begin, const size_t count, std::vector<
size_t > & oldFromNew, BinarySpaceTree< BoundType, StatisticType, MatType, SplitType > * parent = NULL,
const size_t maxLeafSize = 20 )
```

Construct this node on a subset of the given matrix, starting at column `begin_in` and using `count_in` points.

The ordering of that subset of points will be modified! This is used for recursive tree-building by the other constructors which don't specify point indices.

A mapping of the old point indices to the new point indices is filled, but it is expected that the vector is already allocated with size greater than or equal to $(\text{begin_in} + \text{count_in})$, and if that is not true, invalid memory reads (and writes) will occur.

Parameters

<i>data</i>	Dataset to create tree from. This will be modified!
<i>begin</i>	Index of point to start tree construction with.
<i>count</i>	Number of points to use to construct tree.
<i>oldFromNew</i>	Vector which will be filled with the old positions for each new point.
<i>maxLeafSize</i>	Size of each leaf in the tree.

```
22.139.3.6  template<typename BoundType , typename StatisticType = EmptyStatistic, typename MatType = arma::mat, typename
SplitType = MeanSplit<BoundType, MatType>> mlpack::tree::BinarySpaceTree< BoundType, StatisticType,
MatType, SplitType >::BinarySpaceTree( MatType & data, const size_t begin, const size_t count, std::vector<
size_t > & oldFromNew, std::vector< size_t > & newFromOld, BinarySpaceTree< BoundType, StatisticType,
MatType, SplitType > * parent = NULL, const size_t maxLeafSize = 20 )
```

Construct this node on a subset of the given matrix, starting at column `begin_in` and using `count_in` points.

The ordering of that subset of points will be modified! This is used for recursive tree-building by the other constructors which don't specify point indices.

A mapping of the old point indices to the new point indices is filled, as well as a mapping of the new point indices to the old point indices. It is expected that the vector is already allocated with size greater than or equal to $(\text{begin_in} + \text{count_in})$, and if that is not true, invalid memory reads (and writes) will occur.

Parameters

<i>data</i>	Dataset to create tree from. This will be modified!
<i>begin</i>	Index of point to start tree construction with.
<i>count</i>	Number of points to use to construct tree.
<i>oldFromNew</i>	Vector which will be filled with the old positions for each new point.
<i>newFromOld</i>	Vector which will be filled with the new positions for each old point.
<i>maxLeafSize</i>	Size of each leaf in the tree.

```
22.139.3.7  template<typename BoundType , typename StatisticType = EmptyStatistic, typename MatType = arma::mat, typename
SplitType = MeanSplit<BoundType, MatType>> mlpack::tree::BinarySpaceTree< BoundType, StatisticType,
MatType, SplitType >::BinarySpaceTree( const BinarySpaceTree< BoundType, StatisticType, MatType, SplitType
> & other )
```

Create a binary space tree by copying the other tree.

Be careful! This can take a long time and use a lot of memory.

Parameters

<i>other</i>	Tree to be replicated.
--------------	------------------------

22.139.3.8 `template<typename BoundType , typename StatisticType = EmptyStatistic, typename MatType = arma::mat, typename SplitType = MeanSplit<BoundType, MatType>> mlpack::tree::BinarySpaceTree< BoundType, StatisticType, MatType, SplitType >::~BinarySpaceTree ()`

Deletes this node, deallocating the memory for the children and calling their destructors in turn.

This will invalidate any pointers or references to any nodes which are children of this one.

22.139.3.9 `template<typename BoundType , typename StatisticType = EmptyStatistic, typename MatType = arma::mat, typename SplitType = MeanSplit<BoundType, MatType>> mlpack::tree::BinarySpaceTree< BoundType, StatisticType, MatType, SplitType >::~BinarySpaceTree(const size_t begin, const size_t count, BoundType bound, StatisticType stat, const int maxLeafSize = 20) [inline], [private]`

Private copy constructor, available only to fill (pad) the tree to a specified level.

Definition at line 452 of file `binary_space_tree.hpp`.

22.139.4 Member Function Documentation

22.139.4.1 `template<typename BoundType , typename StatisticType = EmptyStatistic, typename MatType = arma::mat, typename SplitType = MeanSplit<BoundType, MatType>> size_t mlpack::tree::BinarySpaceTree< BoundType, StatisticType, MatType, SplitType >::Begin () const [inline]`

Return the index of the beginning point of this subset.

Definition at line 430 of file `binary_space_tree.hpp`.

References `mlpack::tree::BinarySpaceTree< BoundType, StatisticType, MatType, SplitType >::begin`.

22.139.4.2 `template<typename BoundType , typename StatisticType = EmptyStatistic, typename MatType = arma::mat, typename SplitType = MeanSplit<BoundType, MatType>> size_t& mlpack::tree::BinarySpaceTree< BoundType, StatisticType, MatType, SplitType >::Begin () [inline]`

Modify the index of the beginning point of this subset.

Definition at line 432 of file `binary_space_tree.hpp`.

References `mlpack::tree::BinarySpaceTree< BoundType, StatisticType, MatType, SplitType >::begin`.

22.139.4.3 `template<typename BoundType , typename StatisticType = EmptyStatistic, typename MatType = arma::mat, typename SplitType = MeanSplit<BoundType, MatType>> const BoundType& mlpack::tree::BinarySpaceTree< BoundType, StatisticType, MatType, SplitType >::Bound () const [inline]`

Return the bound object for this node.

Definition at line 252 of file `binary_space_tree.hpp`.

References `mlpack::tree::BinarySpaceTree< BoundType, StatisticType, MatType, SplitType >::bound`.

Referenced by mlpack::tree::BinarySpaceTree< BoundType, StatisticType, MatType, SplitType >::MaxDistance(), mlpack::tree::BinarySpaceTree< BoundType, StatisticType, MatType, SplitType >::MinDistance(), and mlpack::tree::BinarySpaceTree< BoundType, StatisticType, MatType, SplitType >::RangeDistance().

22.139.4.4 `template<typename BoundType , typename StatisticType = EmptyStatistic, typename MatType = arma::mat, typename SplitType = MeanSplit<BoundType, MatType>> BoundType& mlpack::tree::BinarySpaceTree< BoundType, StatisticType, MatType, SplitType >::Bound () [inline]`

Return the bound object for this node.

Definition at line 254 of file `binary_space_tree.hpp`.

References mlpack::tree::BinarySpaceTree< BoundType, StatisticType, MatType, SplitType >::bound.

22.139.4.5 `template<typename BoundType , typename StatisticType = EmptyStatistic, typename MatType = arma::mat, typename SplitType = MeanSplit<BoundType, MatType>> void mlpack::tree::BinarySpaceTree< BoundType, StatisticType, MatType, SplitType >::Centroid (arma::vec & centroid) [inline]`

Get the centroid of the node and store it in the given vector.

Definition at line 301 of file `binary_space_tree.hpp`.

22.139.4.6 `template<typename BoundType , typename StatisticType = EmptyStatistic, typename MatType = arma::mat, typename SplitType = MeanSplit<BoundType, MatType>> BinarySpaceTree& mlpack::tree::BinarySpaceTree< BoundType, StatisticType, MatType, SplitType >::Child (const size_t child) const`

Return the specified child (0 will be left, 1 will be right).

If the index is greater than 1, this will return the right child.

Parameters

<i>child</i>	Index of child to return.
--------------	---------------------------

22.139.4.7 `template<typename BoundType , typename StatisticType = EmptyStatistic, typename MatType = arma::mat, typename SplitType = MeanSplit<BoundType, MatType>> BinarySpaceTree* mlpack::tree::BinarySpaceTree< BoundType, StatisticType, MatType, SplitType >::CopyMe () [inline],[private]`

Definition at line 465 of file `binary_space_tree.hpp`.

References mlpack::tree::BinarySpaceTree< BoundType, StatisticType, MatType, SplitType >::BinarySpaceTree().

22.139.4.8 `template<typename BoundType , typename StatisticType = EmptyStatistic, typename MatType = arma::mat, typename SplitType = MeanSplit<BoundType, MatType>> size_t mlpack::tree::BinarySpaceTree< BoundType, StatisticType, MatType, SplitType >::Count () const [inline]`

Return the number of points in this subset.

Definition at line 440 of file `binary_space_tree.hpp`.

References mlpack::tree::BinarySpaceTree< BoundType, StatisticType, MatType, SplitType >::count.

```
22.139.4.9  template<typename BoundType , typename StatisticType = EmptyStatistic, typename MatType = arma::mat, typename
            SplitType = MeanSplit<BoundType, MatType>> size_t& mlpack::tree::BinarySpaceTree< BoundType,
            StatisticType, MatType, SplitType >::Count ( ) [inline]
```

Modify the number of points in this subset.

Definition at line 442 of file `binary_space_tree.hpp`.

References `mlpack::tree::BinarySpaceTree< BoundType, StatisticType, MatType, SplitType >::count`.

```
22.139.4.10 template<typename BoundType , typename StatisticType = EmptyStatistic, typename MatType = arma::mat, typename
            SplitType = MeanSplit<BoundType, MatType>> const MatType& mlpack::tree::BinarySpaceTree< BoundType,
            StatisticType, MatType, SplitType >::Dataset ( ) const [inline]
```

Get the dataset which the tree is built on.

Definition at line 293 of file `binary_space_tree.hpp`.

References `mlpack::tree::BinarySpaceTree< BoundType, StatisticType, MatType, SplitType >::dataset`.

```
22.139.4.11 template<typename BoundType , typename StatisticType = EmptyStatistic, typename MatType = arma::mat, typename
            SplitType = MeanSplit<BoundType, MatType>> MatType& mlpack::tree::BinarySpaceTree< BoundType,
            StatisticType, MatType, SplitType >::Dataset ( ) [inline]
```

Modify the dataset which the tree is built on. Be careful!

Definition at line 295 of file `binary_space_tree.hpp`.

References `mlpack::tree::BinarySpaceTree< BoundType, StatisticType, MatType, SplitType >::dataset`.

```
22.139.4.12 template<typename BoundType , typename StatisticType = EmptyStatistic, typename MatType = arma::mat, typename
            SplitType = MeanSplit<BoundType, MatType>> size_t mlpack::tree::BinarySpaceTree< BoundType,
            StatisticType, MatType, SplitType >::Descendant ( const size_t index ) const
```

Return the index (with reference to the dataset) of a particular descendant of this node.

The index should be greater than zero but less than the number of descendants.

Parameters

<i>index</i>	Index of the descendant.
--------------	--------------------------

```
22.139.4.13 template<typename BoundType , typename StatisticType = EmptyStatistic, typename MatType = arma::mat, typename
            SplitType = MeanSplit<BoundType, MatType>> size_t mlpack::tree::BinarySpaceTree< BoundType,
            StatisticType, MatType, SplitType >::End ( ) const
```

Gets the index one beyond the last index in the subset.

```
22.139.4.14 template<typename BoundType , typename StatisticType = EmptyStatistic, typename MatType = arma::mat, typename
            SplitType = MeanSplit<BoundType, MatType>> size_t mlpack::tree::BinarySpaceTree< BoundType,
            StatisticType, MatType, SplitType >::ExtendTree ( const size_t level )
```

Fills the tree to the specified level.

22.139.4.15 `template<typename BoundType , typename StatisticType = EmptyStatistic, typename MatType = arma::mat, typename SplitType = MeanSplit<BoundType, MatType>> const BinarySpaceTree* mlpack::tree::BinarySpaceTree< BoundType, StatisticType, MatType, SplitType >::FindByBeginCount (size_t begin, size_t count) const`

Find a node in this tree by its begin and count (const).

Every node is uniquely identified by these two numbers. This is useful for communicating position over the network, when pointers would be invalid.

Parameters

<i>begin</i>	The begin() (p. 584) of the node to find.
<i>count</i>	The count() (p. 584) of the node to find.

Returns

The found node, or NULL if not found.

22.139.4.16 `template<typename BoundType , typename StatisticType = EmptyStatistic, typename MatType = arma::mat, typename SplitType = MeanSplit<BoundType, MatType>> BinarySpaceTree* mlpack::tree::BinarySpaceTree< BoundType, StatisticType, MatType, SplitType >::FindByBeginCount (size_t begin, size_t count)`

Find a node in this tree by its begin and count.

Every node is uniquely identified by these two numbers. This is useful for communicating position over the network, when pointers would be invalid.

Parameters

<i>begin</i>	The begin() (p. 584) of the node to find.
<i>count</i>	The count() (p. 584) of the node to find.

Returns

The found node, or NULL if not found.

22.139.4.17 `template<typename BoundType , typename StatisticType = EmptyStatistic, typename MatType = arma::mat, typename SplitType = MeanSplit<BoundType, MatType>> double mlpack::tree::BinarySpaceTree< BoundType, StatisticType, MatType, SplitType >::FurthestDescendantDistance () const`

Return the furthest possible descendant distance.

This returns the maximum distance from the centroid to the edge of the bound and not the empirical quantity which is the actual furthest descendant distance. So the actual furthest descendant distance may be less than what this method returns (but it will never be greater than this).

22.139.4.18 `template<typename BoundType , typename StatisticType = EmptyStatistic, typename MatType = arma::mat, typename SplitType = MeanSplit<BoundType, MatType>> double mlpack::tree::BinarySpaceTree< BoundType, StatisticType, MatType, SplitType >::FurthestPointDistance () const`

Return the furthest distance to a point held in this node.

If this is not a leaf node, then the distance is 0 because the node holds no points.

```
22.139.4.19  template<typename BoundType , typename StatisticType = EmptyStatistic, typename MatType = arma::mat, typename
              SplitType = MeanSplit<BoundType, MatType>> size_t mpack::tree::BinarySpaceTree< BoundType,
              StatisticType, MatType, SplitType >::GetSplitDimension ( ) const
```

Returns the dimension this parent's children are split on.

```
22.139.4.20  template<typename BoundType , typename StatisticType = EmptyStatistic, typename MatType = arma::mat, typename
              SplitType = MeanSplit<BoundType, MatType>> static bool mpack::tree::BinarySpaceTree< BoundType,
              StatisticType, MatType, SplitType >::HasSelfChildren ( ) [inline],[static]
```

Returns false: this tree type does not have self children.

Definition at line 445 of file binary_space_tree.hpp.

```
22.139.4.21  template<typename BoundType , typename StatisticType = EmptyStatistic, typename MatType = arma::mat,
              typename SplitType = MeanSplit<BoundType, MatType>> bool mpack::tree::BinarySpaceTree< BoundType,
              StatisticType, MatType, SplitType >::IsLeaf ( ) const
```

Return whether or not this node is a leaf (true if it has no children).

```
22.139.4.22  template<typename BoundType , typename StatisticType = EmptyStatistic, typename MatType = arma::mat, typename
              SplitType = MeanSplit<BoundType, MatType>> BinarySpaceTree* mpack::tree::BinarySpaceTree<
              BoundType, StatisticType, MatType, SplitType >::Left ( ) const [inline]
```

Gets the left child of this node.

Definition at line 273 of file binary_space_tree.hpp.

References mpack::tree::BinarySpaceTree< BoundType, StatisticType, MatType, SplitType >::left.

```
22.139.4.23  template<typename BoundType , typename StatisticType = EmptyStatistic, typename MatType = arma::mat, typename
              SplitType = MeanSplit<BoundType, MatType>> BinarySpaceTree*& mpack::tree::BinarySpaceTree<
              BoundType, StatisticType, MatType, SplitType >::Left ( ) [inline]
```

Modify the left child of this node.

Definition at line 275 of file binary_space_tree.hpp.

References mpack::tree::BinarySpaceTree< BoundType, StatisticType, MatType, SplitType >::left.

```
22.139.4.24  template<typename BoundType , typename StatisticType = EmptyStatistic, typename MatType = arma::mat, typename
              SplitType = MeanSplit<BoundType, MatType>> double mpack::tree::BinarySpaceTree< BoundType,
              StatisticType, MatType, SplitType >::MaxDistance ( const BinarySpaceTree< BoundType, StatisticType, MatType,
              SplitType > * other ) const [inline]
```

Return the maximum distance to another node.

Definition at line 375 of file binary_space_tree.hpp.

References mpack::tree::BinarySpaceTree< BoundType, StatisticType, MatType, SplitType >::Bound().

```
22.139.4.25  template<typename BoundType , typename StatisticType = EmptyStatistic, typename MatType = arma::mat,
              typename SplitType = MeanSplit<BoundType, MatType>>> template<typename VecType > double
              mlpack::tree::BinarySpaceTree< BoundType, StatisticType, MatType, SplitType >::MaxDistance ( const VecType
              & point, typename boost::enable_if< IsVector< VecType >>::type * = 0 ) const  [inline]
```

Return the maximum distance to another point.

Definition at line 397 of file `binary_space_tree.hpp`.

```
22.139.4.26  template<typename BoundType , typename StatisticType = EmptyStatistic, typename MatType = arma::mat, typename
              SplitType = MeanSplit<BoundType, MatType>>> size_t mlpack::tree::BinarySpaceTree< BoundType,
              StatisticType, MatType, SplitType >::MaxLeafSize ( ) const  [inline]
```

Return the max leaf size.

Definition at line 265 of file `binary_space_tree.hpp`.

References `mlpack::tree::BinarySpaceTree< BoundType, StatisticType, MatType, SplitType >::maxLeafSize`.

```
22.139.4.27  template<typename BoundType , typename StatisticType = EmptyStatistic, typename MatType = arma::mat, typename
              SplitType = MeanSplit<BoundType, MatType>>> size_t& mlpack::tree::BinarySpaceTree< BoundType,
              StatisticType, MatType, SplitType >::MaxLeafSize ( )  [inline]
```

Modify the max leaf size.

Definition at line 267 of file `binary_space_tree.hpp`.

References `mlpack::tree::BinarySpaceTree< BoundType, StatisticType, MatType, SplitType >::maxLeafSize`.

```
22.139.4.28  template<typename BoundType , typename StatisticType = EmptyStatistic, typename MatType = arma::mat, typename
              SplitType = MeanSplit<BoundType, MatType>>> BoundType::MetricType mlpack::tree::BinarySpaceTree<
              BoundType, StatisticType, MatType, SplitType >::Metric ( ) const  [inline]
```

Get the metric which the tree uses.

Definition at line 298 of file `binary_space_tree.hpp`.

```
22.139.4.29  template<typename BoundType , typename StatisticType = EmptyStatistic, typename MatType = arma::mat, typename
              SplitType = MeanSplit<BoundType, MatType>>> double mlpack::tree::BinarySpaceTree< BoundType,
              StatisticType, MatType, SplitType >::MinDistance ( const BinarySpaceTree< BoundType, StatisticType, MatType,
              SplitType > * other ) const  [inline]
```

Return the minimum distance to another node.

Definition at line 369 of file `binary_space_tree.hpp`.

References `mlpack::tree::BinarySpaceTree< BoundType, StatisticType, MatType, SplitType >::Bound()`.

```
22.139.4.30  template<typename BoundType , typename StatisticType = EmptyStatistic, typename MatType = arma::mat,
              typename SplitType = MeanSplit<BoundType, MatType>>> template<typename VecType > double
              mlpack::tree::BinarySpaceTree< BoundType, StatisticType, MatType, SplitType >::MinDistance ( const VecType
              & point, typename boost::enable_if< IsVector< VecType >>::type * = 0 ) const  [inline]
```

Return the minimum distance to another point.

Definition at line 388 of file `binary_space_tree.hpp`.

```
22.139.4.31  template<typename BoundType , typename StatisticType = EmptyStatistic, typename MatType = arma::mat, typename
              SplitType = MeanSplit<BoundType, MatType>> double mlpack::tree::BinarySpaceTree< BoundType,
              StatisticType, MatType, SplitType >::MinimumBoundDistance ( ) const
```

Return the minimum distance from the center of the node to any bound edge.

```
22.139.4.32  template<typename BoundType , typename StatisticType = EmptyStatistic, typename MatType = arma::mat, typename
              SplitType = MeanSplit<BoundType, MatType>> size_t mlpack::tree::BinarySpaceTree< BoundType,
              StatisticType, MatType, SplitType >::NumChildren ( ) const
```

Return the number of children in this node.

```
22.139.4.33  template<typename BoundType , typename StatisticType = EmptyStatistic, typename MatType = arma::mat, typename
              SplitType = MeanSplit<BoundType, MatType>> size_t mlpack::tree::BinarySpaceTree< BoundType,
              StatisticType, MatType, SplitType >::NumDescendants ( ) const
```

Return the number of descendants of this node.

For a non-leaf in a binary space tree, this is the number of points at the descendant leaves. For a leaf, this is the number of points in the leaf.

```
22.139.4.34  template<typename BoundType , typename StatisticType = EmptyStatistic, typename MatType = arma::mat, typename
              SplitType = MeanSplit<BoundType, MatType>> size_t mlpack::tree::BinarySpaceTree< BoundType,
              StatisticType, MatType, SplitType >::NumPoints ( ) const
```

Return the number of points in this node (0 if not a leaf).

```
22.139.4.35  template<typename BoundType , typename StatisticType = EmptyStatistic, typename MatType = arma::mat, typename
              SplitType = MeanSplit<BoundType, MatType>> BinarySpaceTree* mlpack::tree::BinarySpaceTree<
              BoundType, StatisticType, MatType, SplitType >::Parent ( ) const [inline]
```

Gets the parent of this node.

Definition at line 283 of file `binary_space_tree.hpp`.

References `mlpack::tree::BinarySpaceTree< BoundType, StatisticType, MatType, SplitType >::parent`.

```
22.139.4.36  template<typename BoundType , typename StatisticType = EmptyStatistic, typename MatType = arma::mat, typename
              SplitType = MeanSplit<BoundType, MatType>> BinarySpaceTree*& mlpack::tree::BinarySpaceTree<
              BoundType, StatisticType, MatType, SplitType >::Parent ( ) [inline]
```

Modify the parent of this node.

Definition at line 285 of file `binary_space_tree.hpp`.

References `mlpack::tree::BinarySpaceTree< BoundType, StatisticType, MatType, SplitType >::parent`.


```
22.139.4.37  template<typename BoundType , typename StatisticType = EmptyStatistic, typename MatType = arma::mat, typename
              SplitType = MeanSplit<BoundType, MatType>> double mlpack::tree::BinarySpaceTree< BoundType,
              StatisticType, MatType, SplitType >::ParentDistance ( ) const    [inline]
```

Return the distance from the center of this node to the center of the parent node.

Definition at line 326 of file binary_space_tree.hpp.

References mlpack::tree::BinarySpaceTree< BoundType, StatisticType, MatType, SplitType >::parentDistance.

```
22.139.4.38  template<typename BoundType , typename StatisticType = EmptyStatistic, typename MatType = arma::mat, typename
              SplitType = MeanSplit<BoundType, MatType>> double& mlpack::tree::BinarySpaceTree< BoundType,
              StatisticType, MatType, SplitType >::ParentDistance ( )    [inline]
```

Modify the distance from the center of this node to the center of the parent node.

Definition at line 329 of file binary_space_tree.hpp.

References mlpack::tree::BinarySpaceTree< BoundType, StatisticType, MatType, SplitType >::parentDistance.

```
22.139.4.39  template<typename BoundType , typename StatisticType = EmptyStatistic, typename MatType = arma::mat, typename
              SplitType = MeanSplit<BoundType, MatType>> size_t mlpack::tree::BinarySpaceTree< BoundType,
              StatisticType, MatType, SplitType >::Point ( const size_t index ) const
```

Return the index (with reference to the dataset) of a particular point in this node.

This will happily return invalid indices if the given index is greater than the number of points in this node (obtained with **NumPoints()** (p. 580)) – be careful.

Parameters

<i>index</i>	Index of point for which a dataset index is wanted.
--------------	---

```
22.139.4.40  template<typename BoundType , typename StatisticType = EmptyStatistic, typename MatType = arma::mat, typename
              SplitType = MeanSplit<BoundType, MatType>> math::Range mlpack::tree::BinarySpaceTree< BoundType,
              StatisticType, MatType, SplitType >::RangeDistance ( const BinarySpaceTree< BoundType, StatisticType, MatType,
              SplitType > * other ) const    [inline]
```

Return the minimum and maximum distance to another node.

Definition at line 381 of file binary_space_tree.hpp.

References mlpack::tree::BinarySpaceTree< BoundType, StatisticType, MatType, SplitType >::Bound().

```
22.139.4.41  template<typename BoundType , typename StatisticType = EmptyStatistic, typename MatType = arma::mat,
              typename SplitType = MeanSplit<BoundType, MatType>> template<typename VecType > math::Range
              mlpack::tree::BinarySpaceTree< BoundType, StatisticType, MatType, SplitType >::RangeDistance ( const
              VecType & point, typename boost::enable_if< IsVector< VecType > >::type * = 0 ) const    [inline]
```

Return the minimum and maximum distance to another point.

Definition at line 407 of file binary_space_tree.hpp.

```
22.139.4.42  template<typename BoundType , typename StatisticType = EmptyStatistic, typename MatType = arma::mat, typename
              SplitType = MeanSplit<BoundType, MatType>>> BinarySpaceTree* mlpack::tree::BinarySpaceTree<
              BoundType, StatisticType, MatType, SplitType >::Right ( ) const [inline]
```

Gets the right child of this node.

Definition at line 278 of file `binary_space_tree.hpp`.

References `mlpack::tree::BinarySpaceTree< BoundType, StatisticType, MatType, SplitType >::right`.

```
22.139.4.43  template<typename BoundType , typename StatisticType = EmptyStatistic, typename MatType = arma::mat, typename
              SplitType = MeanSplit<BoundType, MatType>>> BinarySpaceTree*& mlpack::tree::BinarySpaceTree<
              BoundType, StatisticType, MatType, SplitType >::Right ( ) [inline]
```

Modify the right child of this node.

Definition at line 280 of file `binary_space_tree.hpp`.

References `mlpack::tree::BinarySpaceTree< BoundType, StatisticType, MatType, SplitType >::right`.

```
22.139.4.44  template<typename BoundType , typename StatisticType = EmptyStatistic, typename MatType = arma::mat, typename
              SplitType = MeanSplit<BoundType, MatType>>> size_t mlpack::tree::BinarySpaceTree< BoundType,
              StatisticType, MatType, SplitType >::SplitDimension ( ) const [inline]
```

Get the split dimension for this node.

Definition at line 288 of file `binary_space_tree.hpp`.

References `mlpack::tree::BinarySpaceTree< BoundType, StatisticType, MatType, SplitType >::splitDimension`.

```
22.139.4.45  template<typename BoundType , typename StatisticType = EmptyStatistic, typename MatType = arma::mat, typename
              SplitType = MeanSplit<BoundType, MatType>>> size_t& mlpack::tree::BinarySpaceTree< BoundType,
              StatisticType, MatType, SplitType >::SplitDimension ( ) [inline]
```

Modify the split dimension for this node.

Definition at line 290 of file `binary_space_tree.hpp`.

References `mlpack::tree::BinarySpaceTree< BoundType, StatisticType, MatType, SplitType >::splitDimension`.

```
22.139.4.46  template<typename BoundType , typename StatisticType = EmptyStatistic, typename MatType = arma::mat,
              typename SplitType = MeanSplit<BoundType, MatType>>> void mlpack::tree::BinarySpaceTree< BoundType,
              StatisticType, MatType, SplitType >::SplitNode ( MatType & data ) [private]
```

Splits the current node, assigning its left and right children recursively.

Parameters

<i>data</i>	Dataset which we are using.
-------------	-----------------------------

```
22.139.4.47  template<typename BoundType , typename StatisticType = EmptyStatistic, typename MatType = arma::mat,
            typename SplitType = MeanSplit<BoundType, MatType>> void mlpack::tree::BinarySpaceTree< BoundType,
            StatisticType, MatType, SplitType >::SplitNode ( MatType & data, std::vector< size_t > & oldFromNew )
            [private]
```

Splits the current node, assigning its left and right children recursively.

Also returns a list of the changed indices.

Parameters

<i>data</i>	Dataset which we are using.
<i>oldFromNew</i>	Vector holding permuted indices.

```
22.139.4.48  template<typename BoundType , typename StatisticType = EmptyStatistic, typename MatType = arma::mat, typename
            SplitType = MeanSplit<BoundType, MatType>> const StatisticType& mlpack::tree::BinarySpaceTree<
            BoundType, StatisticType, MatType, SplitType >::Stat ( ) const [inline]
```

Return the statistic object for this node.

Definition at line 257 of file binary_space_tree.hpp.

References mlpack::tree::BinarySpaceTree< BoundType, StatisticType, MatType, SplitType >::stat.

```
22.139.4.49  template<typename BoundType , typename StatisticType = EmptyStatistic, typename MatType = arma::mat, typename
            SplitType = MeanSplit<BoundType, MatType>> StatisticType& mlpack::tree::BinarySpaceTree< BoundType,
            StatisticType, MatType, SplitType >::Stat ( ) [inline]
```

Return the statistic object for this node.

Definition at line 259 of file binary_space_tree.hpp.

References mlpack::tree::BinarySpaceTree< BoundType, StatisticType, MatType, SplitType >::stat.

```
22.139.4.50  template<typename BoundType , typename StatisticType = EmptyStatistic, typename MatType = arma::mat, typename
            SplitType = MeanSplit<BoundType, MatType>> std::string mlpack::tree::BinarySpaceTree< BoundType,
            StatisticType, MatType, SplitType >::ToString ( ) const
```

Returns a string representation of this object.

```
22.139.4.51  template<typename BoundType , typename StatisticType = EmptyStatistic, typename MatType = arma::mat, typename
            SplitType = MeanSplit<BoundType, MatType>> size_t mlpack::tree::BinarySpaceTree< BoundType,
            StatisticType, MatType, SplitType >::TreeDepth ( ) const
```

Obtains the number of levels below this node in the tree, starting with this.

```
22.139.4.52  template<typename BoundType , typename StatisticType = EmptyStatistic, typename MatType = arma::mat, typename
            SplitType = MeanSplit<BoundType, MatType>> size_t mlpack::tree::BinarySpaceTree< BoundType,
            StatisticType, MatType, SplitType >::TreeSize ( ) const
```

Obtains the number of nodes in the tree, starting with this.

22.139.5 Member Data Documentation

22.139.5.1 `template<typename BoundType , typename StatisticType = EmptyStatistic, typename MatType = arma::mat, typename SplitType = MeanSplit<BoundType, MatType>>> size_t mlpack::tree::BinarySpaceTree< BoundType, StatisticType, MatType, SplitType >::begin [private]`

The index of the first point in the dataset contained in this node (and its children).

Definition at line 62 of file `binary_space_tree.hpp`.

Referenced by `mlpack::tree::BinarySpaceTree< BoundType, StatisticType, MatType, SplitType >::Begin()`.

22.139.5.2 `template<typename BoundType , typename StatisticType = EmptyStatistic, typename MatType = arma::mat, typename SplitType = MeanSplit<BoundType, MatType>>> BoundType mlpack::tree::BinarySpaceTree< BoundType, StatisticType, MatType, SplitType >::bound [private]`

The bound object for this node.

Definition at line 69 of file `binary_space_tree.hpp`.

Referenced by `mlpack::tree::BinarySpaceTree< BoundType, StatisticType, MatType, SplitType >::Bound()`.

22.139.5.3 `template<typename BoundType , typename StatisticType = EmptyStatistic, typename MatType = arma::mat, typename SplitType = MeanSplit<BoundType, MatType>>> size_t mlpack::tree::BinarySpaceTree< BoundType, StatisticType, MatType, SplitType >::count [private]`

The number of points of the dataset contained in this node (and its children).

Definition at line 65 of file `binary_space_tree.hpp`.

Referenced by `mlpack::tree::BinarySpaceTree< BoundType, StatisticType, MatType, SplitType >::Count()`.

22.139.5.4 `template<typename BoundType , typename StatisticType = EmptyStatistic, typename MatType = arma::mat, typename SplitType = MeanSplit<BoundType, MatType>>> MatType& mlpack::tree::BinarySpaceTree< BoundType, StatisticType, MatType, SplitType >::dataset [private]`

The dataset.

Definition at line 82 of file `binary_space_tree.hpp`.

Referenced by `mlpack::tree::BinarySpaceTree< BoundType, StatisticType, MatType, SplitType >::Dataset()`.

22.139.5.5 `template<typename BoundType , typename StatisticType = EmptyStatistic, typename MatType = arma::mat, typename SplitType = MeanSplit<BoundType, MatType>>> double mlpack::tree::BinarySpaceTree< BoundType, StatisticType, MatType, SplitType >::furthestDescendantDistance [private]`

The worst possible distance to the furthest descendant, cached to speed things up.

Definition at line 78 of file `binary_space_tree.hpp`.

22.139.5.6 `template<typename BoundType , typename StatisticType = EmptyStatistic, typename MatType = arma::mat, typename SplitType = MeanSplit<BoundType, MatType>>> BinarySpaceTree* mlpack::tree::BinarySpaceTree< BoundType, StatisticType, MatType, SplitType >::left [private]`

The left child node.

Definition at line 55 of file binary_space_tree.hpp.

Referenced by mpack::tree::BinarySpaceTree< BoundType, StatisticType, MatType, SplitType >::Left().

```
22.139.5.7  template<typename BoundType , typename StatisticType = EmptyStatistic, typename MatType = arma::mat, typename
            SplitType = MeanSplit<BoundType, MatType>> size_t mpack::tree::BinarySpaceTree< BoundType,
            StatisticType, MatType, SplitType >::maxLeafSize  [private]
```

The max leaf size.

Definition at line 67 of file binary_space_tree.hpp.

Referenced by mpack::tree::BinarySpaceTree< BoundType, StatisticType, MatType, SplitType >::MaxLeafSize().

```
22.139.5.8  template<typename BoundType , typename StatisticType = EmptyStatistic, typename MatType = arma::mat, typename
            SplitType = MeanSplit<BoundType, MatType>> double mpack::tree::BinarySpaceTree< BoundType,
            StatisticType, MatType, SplitType >::minimumBoundDistance  [private]
```

The minimum distance from the center to any edge of the bound.

Definition at line 80 of file binary_space_tree.hpp.

```
22.139.5.9  template<typename BoundType , typename StatisticType = EmptyStatistic, typename MatType = arma::mat, typename
            SplitType = MeanSplit<BoundType, MatType>> BinarySpaceTree* mpack::tree::BinarySpaceTree<
            BoundType, StatisticType, MatType, SplitType >::parent  [private]
```

The parent node (NULL if this is the root of the tree).

Definition at line 59 of file binary_space_tree.hpp.

Referenced by mpack::tree::BinarySpaceTree< BoundType, StatisticType, MatType, SplitType >::Parent().

```
22.139.5.10 template<typename BoundType , typename StatisticType = EmptyStatistic, typename MatType = arma::mat, typename
            SplitType = MeanSplit<BoundType, MatType>> double mpack::tree::BinarySpaceTree< BoundType,
            StatisticType, MatType, SplitType >::parentDistance  [private]
```

The distance from the centroid of this node to the centroid of the parent.

Definition at line 75 of file binary_space_tree.hpp.

Referenced by mpack::tree::BinarySpaceTree< BoundType, StatisticType, MatType, SplitType >::ParentDistance().

```
22.139.5.11 template<typename BoundType , typename StatisticType = EmptyStatistic, typename MatType = arma::mat, typename
            SplitType = MeanSplit<BoundType, MatType>> BinarySpaceTree* mpack::tree::BinarySpaceTree<
            BoundType, StatisticType, MatType, SplitType >::right  [private]
```

The right child node.

Definition at line 57 of file binary_space_tree.hpp.

Referenced by mpack::tree::BinarySpaceTree< BoundType, StatisticType, MatType, SplitType >::Right().

```
22.139.5.12 template<typename BoundType , typename StatisticType = EmptyStatistic, typename MatType = arma::mat, typename
SplitType = MeanSplit<BoundType, MatType>> size_t mlpack::tree::BinarySpaceTree< BoundType,
StatisticType, MatType, SplitType >::splitDimension [private]
```

The dimension this node split on if it is a parent.

Definition at line 73 of file `binary_space_tree.hpp`.

Referenced by `mlpack::tree::BinarySpaceTree< BoundType, StatisticType, MatType, SplitType >::SplitDimension()`.

```
22.139.5.13 template<typename BoundType , typename StatisticType = EmptyStatistic, typename MatType = arma::mat, typename
SplitType = MeanSplit<BoundType, MatType>> StatisticType mlpack::tree::BinarySpaceTree< BoundType,
StatisticType, MatType, SplitType >::stat [private]
```

Any extra data contained in the node.

Definition at line 71 of file `binary_space_tree.hpp`.

Referenced by `mlpack::tree::BinarySpaceTree< BoundType, StatisticType, MatType, SplitType >::Stat()`.

The documentation for this class was generated from the following file:

- `src/mlpack/core/tree/binary_space_tree/binary_space_tree.hpp`

22.140 mlpack::tree::BinarySpaceTree< BoundType, StatisticType, MatType, SplitType >::DualTreeTraverser< RuleType > Class Template Reference

A dual-tree traverser for binary space trees; see `dual_tree_traverser.hpp`.

Public Member Functions

- **DualTreeTraverser** (RuleType &rule)
Instantiate the dual-tree traverser with the given rule set.
- `size_t NumBaseCases ()` const
Get the number of times a base case was calculated.
- `size_t & NumBaseCases ()`
Modify the number of times a base case was calculated.
- `size_t NumPrunes ()` const
Get the number of prunes.
- `size_t & NumPrunes ()`
Modify the number of prunes.
- `size_t NumScores ()` const
Get the number of times a node combination was scored.
- `size_t & NumScores ()`
Modify the number of times a node combination was scored.
- `size_t NumVisited ()` const
Get the number of visited combinations.
- `size_t & NumVisited ()`
Modify the number of visited combinations.
- `void Traverse (BinarySpaceTree &queryNode, BinarySpaceTree &referenceNode)`
Traverse the two trees.

Private Attributes

- `size_t numBaseCases`
The number of times a base case was calculated.
- `size_t numPrunes`
The number of prunes.
- `size_t numScores`
The number of times a node combination was scored.
- `size_t numVisited`
The number of node combinations that have been visited during traversal.
- `RuleType & rule`
Reference to the rules with which the trees will be traversed.
- `RuleType::TraversalInfoType traversalInfo`
Traversal information, held in the class so that it isn't continually being reallocated.

22.140.1 Detailed Description

```
template<typename BoundType, typename StatisticType = EmptyStatistic, typename MatType = arma::mat, typename SplitType = MeanSplit<BoundType, MatType>> template<typename RuleType> class mlpack::tree::BinarySpaceTree< BoundType, StatisticType, MatType, SplitType >::DualTreeTraverser< RuleType >
```

A dual-tree traverser for binary space trees; see `dual_tree_traverser.hpp`.

Definition at line 95 of file `binary_space_tree.hpp`.

22.140.2 Constructor & Destructor Documentation

```
22.140.2.1 template<typename BoundType , typename StatisticType = EmptyStatistic, typename MatType = arma::mat, typename SplitType = MeanSplit<BoundType, MatType>> template<typename RuleType > mlpack::tree::BinarySpaceTree< BoundType, StatisticType, MatType, SplitType >::DualTreeTraverser< RuleType >::DualTreeTraverser ( RuleType & rule )
```

Instantiate the dual-tree traverser with the given rule set.

22.140.3 Member Function Documentation

```
22.140.3.1 template<typename BoundType , typename StatisticType = EmptyStatistic, typename MatType = arma::mat, typename SplitType = MeanSplit<BoundType, MatType>> template<typename RuleType > size_t mlpack::tree::BinarySpaceTree< BoundType, StatisticType, MatType, SplitType >::DualTreeTraverser< RuleType >::NumBaseCases ( ) const [inline]
```

Get the number of times a base case was calculated.

Definition at line 67 of file `dual_tree_traverser.hpp`.

```
22.140.3.2 template<typename BoundType , typename StatisticType = EmptyStatistic, typename MatType = arma::mat, typename SplitType = MeanSplit<BoundType, MatType>> template<typename RuleType > size_t & mlpack::tree::BinarySpaceTree< BoundType, StatisticType, MatType, SplitType >::DualTreeTraverser< RuleType >::NumBaseCases ( ) [inline]
```

Modify the number of times a base case was calculated.

Definition at line 69 of file dual_tree_traverser.hpp.

```
22.140.3.3  template<typename BoundType , typename StatisticType = EmptyStatistic, typename MatType = arma::mat,
            typename SplitType = MeanSplit<BoundType, MatType>>> template<typename RuleType > size_t
            mlpack::tree::BinarySpaceTree< BoundType, StatisticType, MatType, SplitType >::DualTreeTraverser<
            RuleType >::NumPrunes ( ) const    [inline]
```

Get the number of prunes.

Definition at line 52 of file dual_tree_traverser.hpp.

```
22.140.3.4  template<typename BoundType , typename StatisticType = EmptyStatistic, typename MatType = arma::mat,
            typename SplitType = MeanSplit<BoundType, MatType>>> template<typename RuleType > size_t&
            mlpack::tree::BinarySpaceTree< BoundType, StatisticType, MatType, SplitType >::DualTreeTraverser<
            RuleType >::NumPrunes ( ) [inline]
```

Modify the number of prunes.

Definition at line 54 of file dual_tree_traverser.hpp.

```
22.140.3.5  template<typename BoundType , typename StatisticType = EmptyStatistic, typename MatType = arma::mat,
            typename SplitType = MeanSplit<BoundType, MatType>>> template<typename RuleType > size_t
            mlpack::tree::BinarySpaceTree< BoundType, StatisticType, MatType, SplitType >::DualTreeTraverser<
            RuleType >::NumScores ( ) const    [inline]
```

Get the number of times a node combination was scored.

Definition at line 62 of file dual_tree_traverser.hpp.

```
22.140.3.6  template<typename BoundType , typename StatisticType = EmptyStatistic, typename MatType = arma::mat,
            typename SplitType = MeanSplit<BoundType, MatType>>> template<typename RuleType > size_t&
            mlpack::tree::BinarySpaceTree< BoundType, StatisticType, MatType, SplitType >::DualTreeTraverser<
            RuleType >::NumScores ( ) [inline]
```

Modify the number of times a node combination was scored.

Definition at line 64 of file dual_tree_traverser.hpp.

```
22.140.3.7  template<typename BoundType , typename StatisticType = EmptyStatistic, typename MatType = arma::mat,
            typename SplitType = MeanSplit<BoundType, MatType>>> template<typename RuleType > size_t
            mlpack::tree::BinarySpaceTree< BoundType, StatisticType, MatType, SplitType >::DualTreeTraverser<
            RuleType >::NumVisited ( ) const    [inline]
```

Get the number of visited combinations.

Definition at line 57 of file dual_tree_traverser.hpp.


```
22.140.3.8  template<typename BoundType , typename StatisticType = EmptyStatistic, typename MatType = arma::mat,
            typename SplitType = MeanSplit<BoundType, MatType>> template<typename RuleType > size_t&
            mlpack::tree::BinarySpaceTree< BoundType, StatisticType, MatType, SplitType >::DualTreeTraverser<
            RuleType >::NumVisited ( ) [inline]
```

Modify the number of visited combinations.

Definition at line 59 of file dual_tree_traverser.hpp.

```
22.140.3.9  template<typename BoundType , typename StatisticType = EmptyStatistic, typename MatType = arma::mat,
            typename SplitType = MeanSplit<BoundType, MatType>> template<typename RuleType > void
            mlpack::tree::BinarySpaceTree< BoundType, StatisticType, MatType, SplitType >::DualTreeTraverser<
            RuleType >::Traverse ( BinarySpaceTree & queryNode, BinarySpaceTree & referenceNode )
```

Traverse the two trees.

This does not reset the number of prunes.

Parameters

<i>queryNode</i>	The query node to be traversed.
<i>referenceNode</i>	The reference node to be traversed.
<i>score</i>	The score of the current node combination.

22.140.4 Member Data Documentation

```
22.140.4.1  template<typename BoundType , typename StatisticType = EmptyStatistic, typename MatType = arma::mat,
            typename SplitType = MeanSplit<BoundType, MatType>> template<typename RuleType > size_t
            mlpack::tree::BinarySpaceTree< BoundType, StatisticType, MatType, SplitType >::DualTreeTraverser<
            RuleType >::numBaseCases [private]
```

The number of times a base case was calculated.

Definition at line 85 of file dual_tree_traverser.hpp.

```
22.140.4.2  template<typename BoundType , typename StatisticType = EmptyStatistic, typename MatType = arma::mat,
            typename SplitType = MeanSplit<BoundType, MatType>> template<typename RuleType > size_t
            mlpack::tree::BinarySpaceTree< BoundType, StatisticType, MatType, SplitType >::DualTreeTraverser<
            RuleType >::numPrunes [private]
```

The number of prunes.

Definition at line 76 of file dual_tree_traverser.hpp.

```
22.140.4.3  template<typename BoundType , typename StatisticType = EmptyStatistic, typename MatType = arma::mat,
            typename SplitType = MeanSplit<BoundType, MatType>> template<typename RuleType > size_t
            mlpack::tree::BinarySpaceTree< BoundType, StatisticType, MatType, SplitType >::DualTreeTraverser<
            RuleType >::numScores [private]
```

The number of times a node combination was scored.

Definition at line 82 of file dual_tree_traverser.hpp.

```
22.140.4.4  template<typename BoundType , typename StatisticType = EmptyStatistic, typename MatType = arma::mat,
            typename SplitType = MeanSplit<BoundType, MatType>>> template<typename RuleType > size_t
            mlpack::tree::BinarySpaceTree< BoundType, StatisticType, MatType, SplitType >::DualTreeTraverser<
            RuleType >::numVisited    [private]
```

The number of node combinations that have been visited during traversal.

Definition at line 79 of file dual_tree_traverser.hpp.

```
22.140.4.5  template<typename BoundType , typename StatisticType = EmptyStatistic, typename MatType = arma::mat,
            typename SplitType = MeanSplit<BoundType, MatType>>> template<typename RuleType > RuleType&
            mlpack::tree::BinarySpaceTree< BoundType, StatisticType, MatType, SplitType >::DualTreeTraverser<
            RuleType >::rule    [private]
```

Reference to the rules with which the trees will be traversed.

Definition at line 73 of file dual_tree_traverser.hpp.

```
22.140.4.6  template<typename BoundType , typename StatisticType = EmptyStatistic, typename MatType = arma::mat, typename
            SplitType = MeanSplit<BoundType, MatType>>> template<typename RuleType > RuleType::TraversallInfoType
            mlpack::tree::BinarySpaceTree< BoundType, StatisticType, MatType, SplitType >::DualTreeTraverser<
            RuleType >::traversallInfo    [private]
```

Traversal information, held in the class so that it isn't continually being reallocated.

Definition at line 89 of file dual_tree_traverser.hpp.

The documentation for this class was generated from the following files:

- src/mlpack/core/tree/binary_space_tree/**binary_space_tree.hpp**
- src/mlpack/core/tree/binary_space_tree/**dual_tree_traverser.hpp**

22.141 mlpack::tree::BinarySpaceTree< BoundType, StatisticType, MatType, SplitType >::SingleTreeTraverser< RuleType > Class Template Reference

A single-tree traverser for binary space trees; see single_tree_traverser.hpp for implementation.

Public Member Functions

- **SingleTreeTraverser** (RuleType &rule)
Instantiate the single tree traverser with the given rule set.
- size_t **NumPrunes** () const
Get the number of prunes.
- size_t & **NumPrunes** ()
Modify the number of prunes.
- void **Traverse** (const size_t queryIndex, **BinarySpaceTree** &referenceNode)
Traverse the tree with the given point.

Private Attributes

- `size_t numPrunes`

The number of nodes which have been pruned during traversal.

- `RuleType & rule`

Reference to the rules with which the tree will be traversed.

22.141.1 Detailed Description

```
template<typename BoundType, typename StatisticType = EmptyStatistic, typename MatType = arma::mat, typename SplitType = MeanSplit<BoundType, MatType>>> template<typename RuleType> class mlpack::tree::BinarySpaceTree< BoundType, StatisticType, MatType, SplitType >::SingleTreeTraverser< RuleType >
```

A single-tree traverser for binary space trees; see `single_tree_traverser.hpp` for implementation.

Definition at line 91 of file `binary_space_tree.hpp`.

22.141.2 Constructor & Destructor Documentation

```
22.141.2.1 template<typename BoundType , typename StatisticType = EmptyStatistic, typename MatType = arma::mat, typename SplitType = MeanSplit<BoundType, MatType>>> template<typename RuleType > mlpack::tree::BinarySpaceTree< BoundType, StatisticType, MatType, SplitType >::SingleTreeTraverser< RuleType >::SingleTreeTraverser( RuleType & rule )
```

Instantiate the single tree traverser with the given rule set.

22.141.3 Member Function Documentation

```
22.141.3.1 template<typename BoundType , typename StatisticType = EmptyStatistic, typename MatType = arma::mat, typename SplitType = MeanSplit<BoundType, MatType>>> template<typename RuleType > size_t mlpack::tree::BinarySpaceTree< BoundType, StatisticType, MatType, SplitType >::SingleTreeTraverser< RuleType >::NumPrunes( ) const [inline]
```

Get the number of prunes.

Definition at line 50 of file `single_tree_traverser.hpp`.

```
22.141.3.2 template<typename BoundType , typename StatisticType = EmptyStatistic, typename MatType = arma::mat, typename SplitType = MeanSplit<BoundType, MatType>>> template<typename RuleType > size_t& mlpack::tree::BinarySpaceTree< BoundType, StatisticType, MatType, SplitType >::SingleTreeTraverser< RuleType >::NumPrunes( ) [inline]
```

Modify the number of prunes.

Definition at line 52 of file `single_tree_traverser.hpp`.

```
22.141.3.3  template<typename BoundType , typename StatisticType = EmptyStatistic, typename MatType = arma::mat,
            typename SplitType = MeanSplit<BoundType, MatType>> template<typename RuleType > void
            mlpack::tree::BinarySpaceTree< BoundType, StatisticType, MatType, SplitType >::SingleTreeTraverser<
            RuleType >:: Traverse ( const size_t queryIndex, BinarySpaceTree & referenceNode )
```

Traverse the tree with the given point.

Parameters

<i>queryIndex</i>	The index of the point in the query set which is being used as the query point.
<i>referenceNode</i>	The tree node to be traversed.

22.141.4 Member Data Documentation

22.141.4.1 `template<typename BoundType , typename StatisticType = EmptyStatistic, typename MatType = arma::mat, typename SplitType = MeanSplit<BoundType, MatType>> template<typename RuleType > size_t mlpack::tree::BinarySpaceTree< BoundType, StatisticType, MatType, SplitType >::SingleTreeTraverser< RuleType >::numPrunes [private]`

The number of nodes which have been pruned during traversal.

Definition at line 59 of file `single_tree_traverser.hpp`.

22.141.4.2 `template<typename BoundType , typename StatisticType = EmptyStatistic, typename MatType = arma::mat, typename SplitType = MeanSplit<BoundType, MatType>> template<typename RuleType > RuleType& mlpack::tree::BinarySpaceTree< BoundType, StatisticType, MatType, SplitType >::SingleTreeTraverser< RuleType >::rule [private]`

Reference to the rules with which the tree will be traversed.

Definition at line 56 of file `single_tree_traverser.hpp`.

The documentation for this class was generated from the following files:

- `src/mlpack/core/tree/binary_space_tree/binary_space_tree.hpp`
- `src/mlpack/core/tree/binary_space_tree/single_tree_traverser.hpp`

22.142 mlpack::tree::CompareCosineNode Class Reference

Public Member Functions

- `bool operator() (const CosineTree *a, const CosineTree *b) const`

22.142.1 Detailed Description

Definition at line 245 of file `cosine_tree.hpp`.

22.142.2 Member Function Documentation

22.142.2.1 `bool mlpack::tree::CompareCosineNode::operator() (const CosineTree * a, const CosineTree * b) const [inline]`

Definition at line 250 of file `cosine_tree.hpp`.

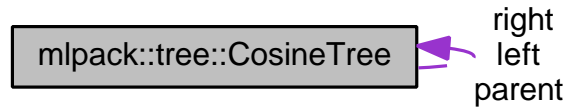
References `mlpack::tree::CosineTree::L2Error()`.

The documentation for this class was generated from the following file:

- `src/mlpack/core/tree/cosine_tree/cosine_tree.hpp`

22.143 mlpack::tree::CosineTree Class Reference

Collaboration diagram for mlpack::tree::CosineTree:



Public Member Functions

- **CosineTree** (const arma::mat &dataset)

CosineTree (p. 594) constructor for the root node of the tree.
- **CosineTree** (CosineTree &parentNode, const std::vector< size_t > &subIndices)

CosineTree (p. 594) constructor for nodes other than the root node of the tree.
- **CosineTree** (const arma::mat &dataset, const double epsilon, const double delta)

Construct the *CosineTree* (p. 594) and the basis for the given matrix, and passed 'epsilon' and 'delta' parameters.
- ~**CosineTree** ()

Destroy the cosine tree and all of its children (take care of the memory allocations too).
- void **BasisVector** (arma::vec &bVector)

Set the basis vector of the node.
- arma::vec & **BasisVector** ()

Get the basis vector of the node.
- size_t **BinarySearch** (arma::vec &cDistribution, double value, size_t start, size_t end)

Sample a column based on the cumulative Length-Squared distribution of the cosine node, and a randomly generated value in the range [0, 1].
- void **CalculateCentroid** ()

Calculate centroid of the columns present in the node.
- void **CalculateCosines** (arma::vec &cosines)

Calculate cosines of the columns present in the node, with respect to the sampled splitting point.
- arma::vec & **Centroid** ()

Get pointer to the centroid vector.
- size_t **ColumnSampleLS** ()

Sample a point from the Length-Squared distribution of the cosine node.
- void **ColumnSamplesLS** (std::vector< size_t > &sampledIndices, arma::vec &probabilities, size_t numSamples)

Sample 'numSamples' points from the Length-Squared distribution of the cosine node.
- void **ConstructBasis** (CosineNodeQueue &treeQueue)

Constructs the final basis matrix, after the cosine tree construction.
- void **CosineNodeSplit** ()

This function splits the cosine node into two children based on the cosines of the columns contained in the node, with respect to the sampled splitting point.
- double **FrobNormSquared** () const

Get the Frobenius norm squared of columns in the node.
- const arma::mat & **GetDataset** () const

Get pointer to the dataset matrix.
- void **GetFinalBasis** (arma::mat &finalBasis)

Returns the basis of the constructed subspace.

- void **L2Error** (const double error)
Set the Monte Carlo error.
- double **L2Error** () const
Get the Monte Carlo error.
- **CosineTree** * **Left** ()
Get pointer to the left child of the node.
- void **ModifiedGramSchmidt** (**CosineNodeQueue** &treeQueue, arma::vec &**centroid**, arma::vec &newBasis←
Vector, arma::vec *addBasisVector=NULL)
Calculates the orthonormalization of the passed centroid, with respect to the current vector subspace.
- double **MonteCarloError** (**CosineTree** *node, **CosineNodeQueue** &treeQueue, arma::vec *addBasis←
Vector1=NULL, arma::vec *addBasisVector2=NULL)
Estimates the squared error of the projection of the input node's matrix onto the current vector subspace.
- size_t **NumColumns** () const
Get number of columns of input matrix in the node.
- **CosineTree** * **Right** ()
Get pointer to the right child of the node.
- size_t **SplitPointIndex** () const
Get the column index of split point of the node.
- std::vector< size_t > & **VectorIndices** ()
Get the indices of columns in the node.

Private Attributes

- arma::mat **basis**
Subspace basis of the input dataset.
- arma::vec **basisVector**
Orthonormalized basis vector of the node.
- arma::vec **centroid**
Centroid of columns of input matrix in the node.
- const arma::mat & **dataset**
Matrix for which cosine tree is constructed.
- double **delta**
Cumulative probability for Monte Carlo error lower bound.
- double **epsilon**
Error tolerance fraction for calculated subspace.
- double **frobNormSquared**
Frobenius norm squared of columns in the node.
- std::vector< size_t > **indices**
Indices of columns of input matrix in the node.
- double **l2Error**
Monte Carlo error for this node.
- arma::vec **l2NormsSquared**
L2-norm squared of columns in the node.
- **CosineTree** * **left**
Left child of the node.
- size_t **numColumns**
Number of columns of input matrix in the node.

- **CosineTree * parent**
Parent of the node.
- **CosineTree * right**
Right child of the node.
- **size_t splitPointIndex**
Index of split point of cosine node.

22.143.1 Detailed Description

Definition at line 32 of file cosine_tree.hpp.

22.143.2 Constructor & Destructor Documentation

22.143.2.1 mlpack::tree::CosineTree::CosineTree (const arma::mat & dataset)

CosineTree (p. 594) constructor for the root node of the tree.

It initializes the necessary variables required for splitting of the node, and building the tree further. It takes a pointer to the input matrix and calculates the relevant variables using it.

Parameters

<i>dataset</i>	Matrix for which cosine tree is constructed.
----------------	--

22.143.2.2 mlpack::tree::CosineTree::CosineTree (CosineTree & parentNode, const std::vector< size_t > & subIndices)

CosineTree (p. 594) constructor for nodes other than the root node of the tree.

It takes in a pointer to the parent node and a list of column indices which mentions the columns to be included in the node. The function calculate the relevant variables just like the constructor above.

Parameters

<i>parentNode</i>	Pointer to the parent cosine node.
<i>subIndices</i>	Pointer to vector of column indices to be included.

22.143.2.3 mlpack::tree::CosineTree::CosineTree (const arma::mat & dataset, const double epsilon, const double delta)

Construct the **CosineTree** (p. 594) and the basis for the given matrix, and passed 'epsilon' and 'delta' parameters.

The **CosineTree** (p. 594) is constructed by splitting nodes in the direction of maximum error, stored using a priority queue. Basis vectors are added from the left and right children of the split node. The basis vector from a node is the orthonormalized centroid of its columns. The splitting continues till the Monte Carlo estimate of the input matrix's projection on the obtained subspace is less than a fraction of the norm of the input matrix.

Parameters

<i>dataset</i>	Matrix for which the CosineTree (p. 594) is constructed.
----------------	---

<i>epsilon</i>	Error tolerance fraction for calculated subspace.
<i>delta</i>	Cumulative probability for Monte Carlo error lower bound.

22.143.2.4 mpack::tree::CosineTree::~~CosineTree ()

Destroy the cosine tree and all of its children (take care of the memory allocations too).

22.143.3 Member Function Documentation

22.143.3.1 void mpack::tree::CosineTree::BasisVector (arma::vec & *bVector*) [inline]

Set the basis vector of the node.

Definition at line 192 of file cosine_tree.hpp.

References basisVector.

22.143.3.2 arma::vec& mpack::tree::CosineTree::BasisVector () [inline]

Get the basis vector of the node.

Definition at line 195 of file cosine_tree.hpp.

References basisVector.

22.143.3.3 size_t mpack::tree::CosineTree::BinarySearch (arma::vec & *cDistribution*, double *value*, size_t *start*, size_t *end*)

Sample a column based on the cumulative Length-Squared distribution of the cosine node, and a randomly generated value in the range [0, 1].

Binary search is more efficient than searching linearly for the same. This leads a significant speedup when there are large number of columns to choose from and when a number of samples are to be drawn from the distribution.

Parameters

<i>cDistribution</i>	Cumulative LS distibution of columns in the node.
<i>value</i>	Randomly generated value in the range [0, 1].
<i>start</i>	Starting index of the distribution interval to search in.
<i>end</i>	Ending index of the distribution interval to search in.

22.143.3.4 void mpack::tree::CosineTree::CalculateCentroid ()

Calculate centroid of the columns present in the node.

The calculated centroid is used as a basis vector for the cosine tree being constructed.

22.143.3.5 void mpack::tree::CosineTree::CalculateCosines (arma::vec & *cosines*)

Calculate cosines of the columns present in the node, with respect to the sampled splitting point.

The calculated cosine values are useful for splitting the node into its children.

Parameters

<i>cosines</i>	Vector to store the cosine values in.
----------------	---------------------------------------

22.143.3.6 `arma::vec& mlpack::tree::CosineTree::Centroid ()` `[inline]`

Get pointer to the centroid vector.

Definition at line 189 of file `cosine_tree.hpp`.

References `centroid`.

22.143.3.7 `size_t mlpack::tree::CosineTree::ColumnSampleLS ()`

Sample a point from the Length-Squared distribution of the cosine node.

The function uses 'l2NormsSquared' to calculate the cumulative probability distribution of the column vectors. The sampling is based on a randomly generated value in the range [0, 1].

22.143.3.8 `void mlpack::tree::CosineTree::ColumnSamplesLS (std::vector< size_t > & sampledIndices, arma::vec & probabilities, size_t numSamples)`

Sample 'numSamples' points from the Length-Squared distribution of the cosine node.

The function uses 'l2NormsSquared' to calculate the cumulative probability distribution of the column vectors. The sampling is based on a randomly generated values in the range [0, 1].

22.143.3.9 `void mlpack::tree::CosineTree::ConstructBasis (CosineNodeQueue & treeQueue)`

Constructs the final basis matrix, after the cosine tree construction.

Parameters

<i>treeQueue</i>	Priority queue of cosine nodes.
------------------	---------------------------------

22.143.3.10 `void mlpack::tree::CosineTree::CosineNodeSplit ()`

This function splits the cosine node into two children based on the cosines of the columns contained in the node, with respect to the sampled splitting point.

The function also calls the **CosineTree** (p. 594) constructor for the children.

22.143.3.11 `double mlpack::tree::CosineTree::FrobNormSquared () const` `[inline]`

Get the Frobenius norm squared of columns in the node.

Definition at line 207 of file `cosine_tree.hpp`.

References `frobNormSquared`.

22.143.3.12 `const arma::mat& mlpack::tree::CosineTree::GetDataset () const` `[inline]`

Get pointer to the dataset matrix.

Definition at line 177 of file cosine_tree.hpp.

References dataset.

22.143.3.13 void mlpack::tree::CosineTree::GetFinalBasis (arma::mat & *finalBasis*) [inline]

Returns the basis of the constructed subspace.

Definition at line 174 of file cosine_tree.hpp.

References basis.

22.143.3.14 void mlpack::tree::CosineTree::L2Error (const double *error*) [inline]

Set the Monte Carlo error.

Definition at line 183 of file cosine_tree.hpp.

References l2Error.

Referenced by mlpack::tree::CompareCosineNode::operator()().

22.143.3.15 double mlpack::tree::CosineTree::L2Error () const [inline]

Get the Monte Carlo error.

Definition at line 186 of file cosine_tree.hpp.

References l2Error.

22.143.3.16 CosineTree* mlpack::tree::CosineTree::Left () [inline]

Get pointer to the left child of the node.

Definition at line 198 of file cosine_tree.hpp.

References left.

22.143.3.17 void mlpack::tree::CosineTree::ModifiedGramSchmidt (CosineNodeQueue & *treeQueue*, arma::vec & *centroid*, arma::vec & *newBasisVector*, arma::vec * *addBasisVector* = NULL)

Calculates the orthonormalization of the passed centroid, with respect to the current vector subspace.

Parameters

<i>treeQueue</i>	Priority queue of cosine nodes.
<i>centroid</i>	Centroid of the node being added to the basis.
<i>newBasisVector</i>	Orthonormalized centroid of the node.
<i>addBasisVector</i>	Address to additional basis vector.

22.143.3.18 double mlpack::tree::CosineTree::MonteCarloError (CosineTree * *node*, CosineNodeQueue & *treeQueue*, arma::vec * *addBasisVector1* = NULL, arma::vec * *addBasisVector2* = NULL)

Estimates the squared error of the projection of the input node's matrix onto the current vector subspace.

A normal distribution is fit using weighted norms of projections of samples drawn from the input node's matrix columns. The error is calculated as the difference between the Frobenius norm of the input node's matrix and lower bound of the normal distribution.

Parameters

<i>node</i>	Node for which Monte Carlo estimate is calculated.
<i>treeQueue</i>	Priority queue of cosine nodes.
<i>addBasisVector1</i>	Address to first additional basis vector.
<i>addBasisVector2</i>	Address to second additional basis vector.

22.143.3.19 `size_t mlpack::tree::CosineTree::NumColumns () const [inline]`

Get number of columns of input matrix in the node.

Definition at line 204 of file cosine_tree.hpp.

References numColumns.

22.143.3.20 `CosineTree* mlpack::tree::CosineTree::Right () [inline]`

Get pointer to the right child of the node.

Definition at line 201 of file cosine_tree.hpp.

References right.

22.143.3.21 `size_t mlpack::tree::CosineTree::SplitPointIndex () const [inline]`

Get the column index of split point of the node.

Definition at line 210 of file cosine_tree.hpp.

References indices, and splitPointIndex.

22.143.3.22 `std::vector<size_t>& mlpack::tree::CosineTree::VectorIndices () [inline]`

Get the indices of columns in the node.

Definition at line 180 of file cosine_tree.hpp.

References indices.

22.143.4 Member Data Documentation

22.143.4.1 `arma::mat mlpack::tree::CosineTree::basis [private]`

Subspace basis of the input dataset.

Definition at line 220 of file cosine_tree.hpp.

Referenced by GetFinalBasis().

22.143.4.2 arma::vec mlpack::tree::CosineTree::basisVector [private]

Orthonormalized basis vector of the node.

Definition at line 234 of file cosine_tree.hpp.

Referenced by BasisVector().

22.143.4.3 arma::vec mlpack::tree::CosineTree::centroid [private]

Centroid of columns of input matrix in the node.

Definition at line 232 of file cosine_tree.hpp.

Referenced by Centroid().

22.143.4.4 const arma::mat& mlpack::tree::CosineTree::dataset [private]

Matrix for which cosine tree is constructed.

Definition at line 214 of file cosine_tree.hpp.

Referenced by GetDataset().

22.143.4.5 double mlpack::tree::CosineTree::delta [private]

Cumulative probability for Monte Carlo error lower bound.

Definition at line 218 of file cosine_tree.hpp.

22.143.4.6 double mlpack::tree::CosineTree::epsilon [private]

Error tolerance fraction for calculated subspace.

Definition at line 216 of file cosine_tree.hpp.

22.143.4.7 double mlpack::tree::CosineTree::frobNormSquared [private]

Frobenius norm squared of columns in the node.

Definition at line 242 of file cosine_tree.hpp.

Referenced by FrobNormSquared().

22.143.4.8 std::vector<size_t> mlpack::tree::CosineTree::indices [private]

Indices of columns of input matrix in the node.

Definition at line 228 of file cosine_tree.hpp.

Referenced by SplitPointIndex(), and VectorIndices().

22.143.4.9 double mlpack::tree::CosineTree::l2Error [private]

Monte Carlo error for this node.

Definition at line 240 of file cosine_tree.hpp.

Referenced by L2Error().

22.143.4.10 arma::vec mlpack::tree::CosineTree::l2NormsSquared [private]

L2-norm squared of columns in the node.

Definition at line 230 of file cosine_tree.hpp.

22.143.4.11 CosineTree* mlpack::tree::CosineTree::left [private]

Left child of the node.

Definition at line 224 of file cosine_tree.hpp.

Referenced by Left().

22.143.4.12 size_t mlpack::tree::CosineTree::numColumns [private]

Number of columns of input matrix in the node.

Definition at line 238 of file cosine_tree.hpp.

Referenced by NumColumns().

22.143.4.13 CosineTree* mlpack::tree::CosineTree::parent [private]

Parent of the node.

Definition at line 222 of file cosine_tree.hpp.

22.143.4.14 CosineTree* mlpack::tree::CosineTree::right [private]

Right child of the node.

Definition at line 226 of file cosine_tree.hpp.

Referenced by Right().

22.143.4.15 size_t mlpack::tree::CosineTree::splitPointIndex [private]

Index of split point of cosine node.

Definition at line 236 of file cosine_tree.hpp.

Referenced by SplitPointIndex().

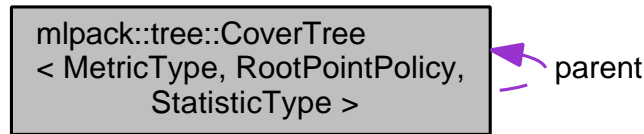
The documentation for this class was generated from the following file:

- src/mlpack/core/tree/cosine_tree/cosine_tree.hpp

22.144 mlpack::tree::CoverTree< MetricType, RootPointPolicy, StatisticType > Class Template Reference

A cover tree is a tree specifically designed to speed up nearest-neighbor computation in high-dimensional spaces.

Collaboration diagram for mlpack::tree::CoverTree< MetricType, RootPointPolicy, StatisticType >:



Classes

- class **DualTreeTraverser**
A dual-tree cover tree traverser; see dual_tree_traverser.hpp.
- class **SingleTreeTraverser**
A single-tree cover tree traverser; see single_tree_traverser.hpp for implementation.

Public Types

- typedef arma::mat **Mat**

Public Member Functions

- **CoverTree** (const arma::mat &**dataset**, const double **base**=2.0, MetricType ***metric**=NULL)
Create the cover tree with the given dataset and given base.
- **CoverTree** (const arma::mat &**dataset**, MetricType &**metric**, const double **base**=2.0)
Create the cover tree with the given dataset and the given instantiated metric.
- **CoverTree** (const arma::mat &**dataset**, const double **base**, const size_t pointIndex, const int **scale**, **CoverTree** ***parent**, const double **parentDistance**, arma::Col< size_t > &indices, arma::vec &distances, size_t nearSet↔Size, size_t &farSetSize, size_t &usedSetSize, MetricType &**metric**=NULL)
Construct a child cover tree node.
- **CoverTree** (const arma::mat &**dataset**, const double **base**, const size_t pointIndex, const int **scale**, **CoverTree** ***parent**, const double **parentDistance**, const double **furthestDescendantDistance**, MetricType ***metric**=NULL)
*Manually construct a cover tree node; no tree assembly is done in this constructor, and children must be added manually (use **Children()** (p. 611)).*
- **CoverTree** (const **CoverTree** &other)
Create a cover tree from another tree.
- ~**CoverTree** ()
Delete this cover tree node and its children.
- double **Base** () const
Get the base.
- double & **Base** ()
Modify the base; don't do this, you'll break everything.

- void **Centroid** (arma::vec ¢roid) const
Get the centroid of the node and store it in the given vector.
- const **CoverTree** & **Child** (const size_t index) const
Get a particular child node.
- **CoverTree** & **Child** (const size_t index)
Modify a particular child node.
- const std::vector< **CoverTree** * > & **Children** () const
Get the children.
- std::vector< **CoverTree** * > & **Children** ()
Modify the children manually (maybe not a great idea).
- const arma::mat & **Dataset** () const
Get a reference to the dataset.
- size_t **Descendant** (const size_t index) const
Get the index of a particular descendant point.
- size_t **DistanceComps** () const
- size_t & **DistanceComps** ()
- double **FurthestDescendantDistance** () const
Get the distance from the center of the node to the furthest descendant.
- double & **FurthestDescendantDistance** ()
Modify the distance from the center of the node to the furthest descendant.
- double **FurthestPointDistance** () const
Get the distance to the furthest point. This is always 0 for cover trees.
- bool **IsLeaf** () const
- double **MaxDistance** (const **CoverTree** *other) const
Return the maximum distance to another node.
- double **MaxDistance** (const **CoverTree** *other, const double distance) const
Return the maximum distance to another node given that the point-to-point distance has already been calculated.
- double **MaxDistance** (const arma::vec &other) const
Return the maximum distance to another point.
- double **MaxDistance** (const arma::vec &other, const double distance) const
Return the maximum distance to another point given that the distance from the center to the point has already been calculated.
- MetricType & **Metric** () const
Get the instantiated metric.
- double **MinDistance** (const **CoverTree** *other) const
Return the minimum distance to another node.
- double **MinDistance** (const **CoverTree** *other, const double distance) const
Return the minimum distance to another node given that the point-to-point distance has already been calculated.
- double **MinDistance** (const arma::vec &other) const
Return the minimum distance to another point.
- double **MinDistance** (const arma::vec &other, const double distance) const
Return the minimum distance to another point given that the distance from the center to the point has already been calculated.
- double **MinimumBoundDistance** () const
Get the minimum distance from the center to any bound edge (this is the same as furthestDescendantDistance).
- size_t **NumChildren** () const
Get the number of children.
- size_t **NumDescendants** () const

Get the number of descendant points.

- `size_t NumPoints () const`
- `CoverTree * Parent () const`

Get the parent node.

- `CoverTree *& Parent ()`

Modify the parent node.

- `double ParentDistance () const`

Get the distance to the parent.

- `double & ParentDistance ()`

Modify the distance to the parent.

- `size_t Point () const`

Get the index of the point which this node represents.

- `size_t Point (const size_t) const`

For compatibility with other trees; the argument is ignored.

- `math::Range RangeDistance (const CoverTree *other) const`

Return the minimum and maximum distance to another node.

- `math::Range RangeDistance (const CoverTree *other, const double distance) const`

Return the minimum and maximum distance to another node given that the point-to-point distance has already been calculated.

- `math::Range RangeDistance (const arma::vec &other) const`

Return the minimum and maximum distance to another point.

- `math::Range RangeDistance (const arma::vec &other, const double distance) const`

Return the minimum and maximum distance to another point given that the point-to-point distance has already been calculated.

- `int Scale () const`

Get the scale of this node.

- `int & Scale ()`

Modify the scale of this node. Be careful...

- `const StatisticType & Stat () const`

Get the statistic for this node.

- `StatisticType & Stat ()`

Modify the statistic for this node.

- `std::string ToString () const`

Returns a string representation of this object.

Static Public Member Functions

- `static bool HasSelfChildren ()`

Returns true: this tree does have self-children.

Private Member Functions

- `void ComputeDistances (const size_t pointIndex, const arma::Col< size_t > &indices, arma::vec &distances, const size_t pointSetSize)`

Fill the vector of distances with the distances between the point specified by pointIndex and each point in the indices array.

- `void CreateChildren (arma::Col< size_t > &indices, arma::vec &distances, size_t nearSetSize, size_t &farSetSize, size_t &usedSetSize)`

Create the children for this node.

- void **MoveToUsedSet** (arma::Col< size_t > &indices, arma::vec &distances, size_t &nearSetSize, size_t &farSetSize, size_t &usedSetSize, arma::Col< size_t > &childIndices, const size_t childFarSetSize, const size_t childUsedSetSize)
- size_t **PruneFarSet** (arma::Col< size_t > &indices, arma::vec &distances, const double bound, const size_t nearSetSize, const size_t pointSetSize)
- void **RemoveNewImplicitNodes** ()

Take a look at the last child (the most recently created one) and remove any implicit nodes that have been created.

- size_t **SortPointSet** (arma::Col< size_t > &indices, arma::vec &distances, const size_t childFarSetSize, const size_t childUsedSetSize, const size_t farSetSize)

Assuming that the list of indices and distances is sorted as [childFarSet | childUsedSet | farSet | usedSet], resort the sets so the organization is [childFarSet | farSet | childUsedSet | usedSet].

- size_t **SplitNearFar** (arma::Col< size_t > &indices, arma::vec &distances, const double bound, const size_t pointSetSize)

Split the given indices and distances into a near and a far set, returning the number of points in the near set.

Private Attributes

- double **base**

The base used to construct the tree.

- std::vector< **CoverTree** * > **children**

The list of children; the first is the self-child.

- const arma::mat & **dataset**

Reference to the matrix which this tree is built on.

- size_t **distanceComps**
- double **furthestDescendantDistance**

Distance to the furthest descendant.

- bool **localMetric**

Whether or not we need to destroy the metric in the destructor.

- MetricType * **metric**

The metric used for this tree.

- size_t **numDescendants**

The number of descendant points.

- **CoverTree** * **parent**

The parent node (NULL if this is the root of the tree).

- double **parentDistance**

Distance to the parent.

- size_t **point**

Index of the point in the matrix which this node represents.

- int **scale**

Scale level of the node.

- StatisticType **stat**

The instantiated statistic.

22.144.1 Detailed Description

```
template<typename MetricType = metric::LMetric<2, true>, typename RootPointPolicy = FirstPointIsRoot, typename StatisticType = EmptyStatistic>class mlpack::tree::CoverTree< MetricType, RootPointPolicy, StatisticType >
```

A cover tree is a tree specifically designed to speed up nearest-neighbor computation in high-dimensional spaces.

Each non-leaf node references a point and has a nonzero number of children, including a "self-child" which references the same point. A leaf node represents only one point.

The tree can be thought of as a hierarchy with the root node at the top level and the leaf nodes at the bottom level. Each level in the tree has an assigned 'scale' i . The tree follows these three conditions:

- nesting: the level C_i is a subset of the level C_{i-1} .
- covering: all node in level C_{i-1} have at least one node in the level C_i with distance less than or equal to b^i (exactly one of these is a parent of the point in level C_{i-1}).
- separation: all nodes in level C_i have distance greater than b^i to all other nodes in level C_i .

The value 'b' refers to the base, which is a parameter of the tree. These three properties make the cover tree very good for fast, high-dimensional nearest-neighbor search.

The theoretical structure of the tree contains many 'implicit' nodes which only have a "self-child" (a child referencing the same point, but at a lower scale level). This practical implementation only constructs explicit nodes – non-leaf nodes with more than one child. A leaf node has no children, and its scale level is `INT_MIN`.

For more information on cover trees, see

```
@inproceedings{
  author = {Beygelzimer, Alina and Kakade, Sham and Langford, John},
  title = {Cover trees for nearest neighbor},
  booktitle = {Proceedings of the 23rd International Conference on Machine
    Learning},
  series = {ICML '06},
  year = {2006},
  pages = {97--104]
}
```

For information on runtime bounds of the nearest-neighbor computation using cover trees, see the following paper, presented at NIPS 2009:

```
@inproceedings{
  author = {Ram, P., and Lee, D., and March, W.B., and Gray, A.G.},
  title = {Linear-time Algorithms for Pairwise Statistical Problems},
  booktitle = {Advances in Neural Information Processing Systems 22},
  editor = {Y. Bengio and D. Schuurmans and J. Lafferty and C.K.I. Williams
    and A. Culotta},
  pages = {1527--1535},
  year = {2009}
}
```

The **CoverTree** (p. 603) class offers three template parameters; a custom metric type can be used with `MetricType` (this class defaults to the L2-squared metric). The root node's point can be chosen with the `RootPointPolicy`; by default, the **FirstPointIsRoot** (p. 638) policy is used, meaning the first point in the dataset is used. The `StatisticType` policy allows you to define statistics which can be gathered during the creation of the tree.

Template Parameters

<i>MetricType</i>	Metric type to use during tree construction.
<i>RootPointPolicy</i>	Determines which point to use as the root node.
<i>StatisticType</i>	Statistic to be used during tree creation.

Definition at line 95 of file `cover_tree.hpp`.

22.144.2 Member Typedef Documentation

22.144.2.1 `template<typename MetricType = metric::LMetric<2, true>, typename RootPointPolicy = FirstPointIsRoot, typename StatisticType = EmptyStatistic> typedef arma::mat mlpack::tree::CoverTree< MetricType, RootPointPolicy, StatisticType >::Mat`

Definition at line 98 of file `cover_tree.hpp`.

22.144.3 Constructor & Destructor Documentation

22.144.3.1 `template<typename MetricType = metric::LMetric<2, true>, typename RootPointPolicy = FirstPointIsRoot, typename StatisticType = EmptyStatistic> mlpack::tree::CoverTree< MetricType, RootPointPolicy, StatisticType >::CoverTree (const arma::mat & dataset, const double base = 2.0, MetricType * metric = NULL)`

Create the cover tree with the given dataset and given base.

The dataset will not be modified during the building procedure (unlike **BinarySpaceTree** (p. 566)).

The last argument will be removed in mlpack 1.1.0 (see #274 and #273).

Parameters

<i>dataset</i>	Reference to the dataset to build a tree on.
<i>base</i>	Base to use during tree building (default 2.0).

22.144.3.2 `template<typename MetricType = metric::LMetric<2, true>, typename RootPointPolicy = FirstPointIsRoot, typename StatisticType = EmptyStatistic> mlpack::tree::CoverTree< MetricType, RootPointPolicy, StatisticType >::CoverTree (const arma::mat & dataset, MetricType & metric, const double base = 2.0)`

Create the cover tree with the given dataset and the given instantiated metric.

Optionally, set the base. The dataset will not be modified during the building procedure (unlike **BinarySpaceTree** (p. 566)).

Parameters

<i>dataset</i>	Reference to the dataset to build a tree on.
<i>metric</i>	Instantiated metric to use during tree building.
<i>base</i>	Base to use during tree building (default 2.0).

```
22.144.3.3 template<typename MetricType = metric::LMetric<2, true>, typename RootPointPolicy = FirstPointIsRoot, typename
StatisticType = EmptyStatistic> mlpack::tree::CoverTree< MetricType, RootPointPolicy, StatisticType
>::CoverTree( const arma::mat & dataset, const double base, const size_t pointIndex, const int scale, CoverTree<
MetricType, RootPointPolicy, StatisticType > * parent, const double parentDistance, arma::Col< size_t > & indices,
arma::vec & distances, size_t nearSetSize, size_t & farSetSize, size_t & usedSetSize, MetricType & metric = NULL )
```

Construct a child cover tree node.

This constructor is not meant to be used externally, but it could be used to insert another node into a tree. This procedure uses only one vector for the near set, the far set, and the used set (this is to prevent unnecessary memory allocation in recursive calls to this constructor). Therefore, the size of the near set, far set, and used set must be passed in. The near set will be entirely used up, and some of the far set may be used. The value of usedSetSize will be set to the number of points used in the construction of this node, and the value of farSetSize will be modified to reflect the number of points in the far set *after* the construction of this node.

If you are calling this manually, be careful that the given scale is as small as possible, or you may be creating an implicit node in your tree.

Parameters

<i>dataset</i>	Reference to the dataset to build a tree on.
<i>base</i>	Base to use during tree building.
<i>pointIndex</i>	Index of the point this node references.
<i>scale</i>	Scale of this level in the tree.
<i>parent</i>	Parent of this node (NULL indicates no parent).
<i>parentDistance</i>	Distance to the parent node.
<i>indices</i>	Array of indices, ordered [nearSet farSet usedSet]; will be modified to [farSet usedSet].
<i>distances</i>	Array of distances, ordered the same way as the indices. These represent the distances between the point specified by pointIndex and each point in the indices array.
<i>nearSetSize</i>	Size of the near set; if 0, this will be a leaf.
<i>farSetSize</i>	Size of the far set; may be modified (if this node uses any points in the far set).
<i>usedSetSize</i>	The number of points used will be added to this number.

```
22.144.3.4 template<typename MetricType = metric::LMetric<2, true>, typename RootPointPolicy = FirstPointIsRoot, typename
StatisticType = EmptyStatistic> mlpack::tree::CoverTree< MetricType, RootPointPolicy, StatisticType
>::CoverTree( const arma::mat & dataset, const double base, const size_t pointIndex, const int scale,
CoverTree< MetricType, RootPointPolicy, StatisticType > * parent, const double parentDistance, const double
furthestDescendantDistance, MetricType * metric = NULL )
```

Manually construct a cover tree node; no tree assembly is done in this constructor, and children must be added manually (use **Children()** (p. 611)).

This constructor is useful when the tree is being "imported" into the **CoverTree** (p. 603) class after being created in some other manner.

Parameters

<i>dataset</i>	Reference to the dataset this node is a part of.
<i>base</i>	Base that was used for tree building.
<i>pointIndex</i>	Index of the point in the dataset which this node refers to.
<i>scale</i>	Scale of this node's level in the tree.

<i>parent</i>	Parent node (NULL indicates no parent).
<i>parentDistance</i>	Distance to parent node point.
<i>furthest↔ Descendant↔ Distance</i>	Distance to furthest descendant point.
<i>metric</i>	Instantiated metric (optional).

22.144.3.5 `template<typename MetricType = metric::LMetric<2, true>, typename RootPointPolicy = FirstPointIsRoot, typename StatisticType = EmptyStatistic> mpack::tree::CoverTree< MetricType, RootPointPolicy, StatisticType >::CoverTree (const CoverTree< MetricType, RootPointPolicy, StatisticType > & other)`

Create a cover tree from another tree.

Be careful! This may use a lot of memory and take a lot of time.

Parameters

<i>other</i>	Cover tree to copy from.
--------------	--------------------------

22.144.3.6 `template<typename MetricType = metric::LMetric<2, true>, typename RootPointPolicy = FirstPointIsRoot, typename StatisticType = EmptyStatistic> mpack::tree::CoverTree< MetricType, RootPointPolicy, StatisticType >::~~CoverTree ()`

Delete this cover tree node and its children.

22.144.4 Member Function Documentation

22.144.4.1 `template<typename MetricType = metric::LMetric<2, true>, typename RootPointPolicy = FirstPointIsRoot, typename StatisticType = EmptyStatistic> double mpack::tree::CoverTree< MetricType, RootPointPolicy, StatisticType >::Base () const [inline]`

Get the base.

Definition at line 254 of file cover_tree.hpp.

References `mpack::tree::CoverTree< MetricType, RootPointPolicy, StatisticType >::base`.

22.144.4.2 `template<typename MetricType = metric::LMetric<2, true>, typename RootPointPolicy = FirstPointIsRoot, typename StatisticType = EmptyStatistic> double& mpack::tree::CoverTree< MetricType, RootPointPolicy, StatisticType >::Base () [inline]`

Modify the base; don't do this, you'll break everything.

Definition at line 256 of file cover_tree.hpp.

References `mpack::tree::CoverTree< MetricType, RootPointPolicy, StatisticType >::base`.

22.144.4.3 `template<typename MetricType = metric::LMetric<2, true>, typename RootPointPolicy = FirstPointIsRoot, typename StatisticType = EmptyStatistic> void mpack::tree::CoverTree< MetricType, RootPointPolicy, StatisticType >::Centroid (arma::vec & centroid) const [inline]`

Get the centroid of the node and store it in the given vector.

Definition at line 335 of file cover_tree.hpp.

References mlpack::tree::CoverTree< MetricType, RootPointPolicy, StatisticType >::dataset, and mlpack::tree::CoverTree< MetricType, RootPointPolicy, StatisticType >::point.

```
22.144.4.4  template<typename MetricType = metric::LMetric<2, true>, typename RootPointPolicy = FirstPointIsRoot, typename
            StatisticType = EmptyStatistic> const CoverTree& mlpack::tree::CoverTree< MetricType, RootPointPolicy,
            StatisticType >::Child ( const size_t index ) const    [inline]
```

Get a particular child node.

Definition at line 230 of file cover_tree.hpp.

References mlpack::tree::CoverTree< MetricType, RootPointPolicy, StatisticType >::children.

```
22.144.4.5  template<typename MetricType = metric::LMetric<2, true>, typename RootPointPolicy = FirstPointIsRoot, typename
            StatisticType = EmptyStatistic> CoverTree& mlpack::tree::CoverTree< MetricType, RootPointPolicy, StatisticType
            >::Child ( const size_t index )    [inline]
```

Modify a particular child node.

Definition at line 232 of file cover_tree.hpp.

References mlpack::tree::CoverTree< MetricType, RootPointPolicy, StatisticType >::children.

```
22.144.4.6  template<typename MetricType = metric::LMetric<2, true>, typename RootPointPolicy = FirstPointIsRoot, typename
            StatisticType = EmptyStatistic> const std::vector<CoverTree*>& mlpack::tree::CoverTree< MetricType,
            RootPointPolicy, StatisticType >::Children ( ) const    [inline]
```

Get the children.

Definition at line 238 of file cover_tree.hpp.

References mlpack::tree::CoverTree< MetricType, RootPointPolicy, StatisticType >::children.

```
22.144.4.7  template<typename MetricType = metric::LMetric<2, true>, typename RootPointPolicy = FirstPointIsRoot, typename
            StatisticType = EmptyStatistic> std::vector<CoverTree*>& mlpack::tree::CoverTree< MetricType,
            RootPointPolicy, StatisticType >::Children ( )    [inline]
```

Modify the children manually (maybe not a great idea).

Definition at line 240 of file cover_tree.hpp.

References mlpack::tree::CoverTree< MetricType, RootPointPolicy, StatisticType >::children.

```
22.144.4.8  template<typename MetricType = metric::LMetric<2, true>, typename RootPointPolicy = FirstPointIsRoot, typename
            StatisticType = EmptyStatistic> void mlpack::tree::CoverTree< MetricType, RootPointPolicy, StatisticType
            >::ComputeDistances ( const size_t pointIndex, const arma::Col< size_t > & indices, arma::vec & distances, const
            size_t pointSetSize )    [private]
```

Fill the vector of distances with the distances between the point specified by pointIndex and each point in the indices array.

The distances of the first pointSetSize points in indices are calculated (so, this does not necessarily need to use all of the points in the arrays).

Parameters

<i>pointIndex</i>	Point to build the distances for.
<i>indices</i>	List of indices to compute distances for.
<i>distances</i>	Vector to store calculated distances in.
<i>pointSetSize</i>	Number of points in arrays to calculate distances for.

22.144.4.9 `template<typename MetricType = metric::LMetric<2, true>, typename RootPointPolicy = FirstPointIsRoot, typename StatisticType = EmptyStatistic> void mlpack::tree::CoverTree< MetricType, RootPointPolicy, StatisticType >::CreateChildren (arma::Col< size_t > & indices, arma::vec & distances, size_t nearSetSize, size_t & farSetSize, size_t & usedSetSize) [private]`

Create the children for this node.

22.144.4.10 `template<typename MetricType = metric::LMetric<2, true>, typename RootPointPolicy = FirstPointIsRoot, typename StatisticType = EmptyStatistic> const arma::mat& mlpack::tree::CoverTree< MetricType, RootPointPolicy, StatisticType >::Dataset () const [inline]`

Get a reference to the dataset.

Definition at line 219 of file cover_tree.hpp.

References mlpack::tree::CoverTree< MetricType, RootPointPolicy, StatisticType >::dataset.

22.144.4.11 `template<typename MetricType = metric::LMetric<2, true>, typename RootPointPolicy = FirstPointIsRoot, typename StatisticType = EmptyStatistic> size_t mlpack::tree::CoverTree< MetricType, RootPointPolicy, StatisticType >::Descendant (const size_t index) const`

Get the index of a particular descendant point.

22.144.4.12 `template<typename MetricType = metric::LMetric<2, true>, typename RootPointPolicy = FirstPointIsRoot, typename StatisticType = EmptyStatistic> size_t mlpack::tree::CoverTree< MetricType, RootPointPolicy, StatisticType >::DistanceComps () const [inline]`

Definition at line 471 of file cover_tree.hpp.

References mlpack::tree::CoverTree< MetricType, RootPointPolicy, StatisticType >::distanceComps.

22.144.4.13 `template<typename MetricType = metric::LMetric<2, true>, typename RootPointPolicy = FirstPointIsRoot, typename StatisticType = EmptyStatistic> size_t& mlpack::tree::CoverTree< MetricType, RootPointPolicy, StatisticType >::DistanceComps () [inline]`

Definition at line 472 of file cover_tree.hpp.

References mlpack::tree::CoverTree< MetricType, RootPointPolicy, StatisticType >::distanceComps.

22.144.4.14 `template<typename MetricType = metric::LMetric<2, true>, typename RootPointPolicy = FirstPointIsRoot, typename StatisticType = EmptyStatistic> double mlpack::tree::CoverTree< MetricType, RootPointPolicy, StatisticType >::FurthestDescendantDistance () const [inline]`

Get the distance from the center of the node to the furthest descendant.

Definition at line 324 of file cover_tree.hpp.

References mpack::tree::CoverTree< MetricType, RootPointPolicy, StatisticType >::furthestDescendantDistance.

```
22.144.4.15  template<typename MetricType = metric::LMetric<2, true>, typename RootPointPolicy = FirstPointIsRoot, typename
              StatisticType = EmptyStatistic> double& mpack::tree::CoverTree< MetricType, RootPointPolicy, StatisticType
              >::FurthestDescendantDistance ( ) [inline]
```

Modify the distance from the center of the node to the furthest descendant.

Definition at line 328 of file cover_tree.hpp.

References mpack::tree::CoverTree< MetricType, RootPointPolicy, StatisticType >::furthestDescendantDistance.

```
22.144.4.16  template<typename MetricType = metric::LMetric<2, true>, typename RootPointPolicy = FirstPointIsRoot, typename
              StatisticType = EmptyStatistic> double mpack::tree::CoverTree< MetricType, RootPointPolicy, StatisticType
              >::FurthestPointDistance ( ) const [inline]
```

Get the distance to the furthest point. This is always 0 for cover trees.

Definition at line 321 of file cover_tree.hpp.

```
22.144.4.17  template<typename MetricType = metric::LMetric<2, true>, typename RootPointPolicy = FirstPointIsRoot, typename
              StatisticType = EmptyStatistic> static bool mpack::tree::CoverTree< MetricType, RootPointPolicy, StatisticType
              >::HasSelfChildren ( ) [inline], [static]
```

Returns true: this tree does have self-children.

Definition at line 308 of file cover_tree.hpp.

```
22.144.4.18  template<typename MetricType = metric::LMetric<2, true>, typename RootPointPolicy = FirstPointIsRoot, typename
              StatisticType = EmptyStatistic> bool mpack::tree::CoverTree< MetricType, RootPointPolicy, StatisticType
              >::IsLeaf ( ) const [inline]
```

Definition at line 226 of file cover_tree.hpp.

References mpack::tree::CoverTree< MetricType, RootPointPolicy, StatisticType >::children.

```
22.144.4.19  template<typename MetricType = metric::LMetric<2, true>, typename RootPointPolicy = FirstPointIsRoot, typename
              StatisticType = EmptyStatistic> double mpack::tree::CoverTree< MetricType, RootPointPolicy, StatisticType
              >::MaxDistance ( const CoverTree< MetricType, RootPointPolicy, StatisticType > * other ) const
```

Return the maximum distance to another node.

```
22.144.4.20  template<typename MetricType = metric::LMetric<2, true>, typename RootPointPolicy = FirstPointIsRoot, typename
              StatisticType = EmptyStatistic> double mpack::tree::CoverTree< MetricType, RootPointPolicy, StatisticType
              >::MaxDistance ( const CoverTree< MetricType, RootPointPolicy, StatisticType > * other, const double distance )
              const
```

Return the maximum distance to another node given that the point-to-point distance has already been calculated.

```
22.144.4.21  template<typename MetricType = metric::LMetric<2, true>, typename RootPointPolicy = FirstPointIsRoot, typename
StatisticType = EmptyStatistic> double mpack::tree::CoverTree< MetricType, RootPointPolicy, StatisticType
>::MaxDistance ( const arma::vec & other ) const
```

Return the maximum distance to another point.

```
22.144.4.22  template<typename MetricType = metric::LMetric<2, true>, typename RootPointPolicy = FirstPointIsRoot, typename
StatisticType = EmptyStatistic> double mpack::tree::CoverTree< MetricType, RootPointPolicy, StatisticType
>::MaxDistance ( const arma::vec & other, const double distance ) const
```

Return the maximum distance to another point given that the distance from the center to the point has already been calculated.

```
22.144.4.23  template<typename MetricType = metric::LMetric<2, true>, typename RootPointPolicy = FirstPointIsRoot, typename
StatisticType = EmptyStatistic> MetricType& mpack::tree::CoverTree< MetricType, RootPointPolicy, StatisticType
>::Metric ( ) const [inline]
```

Get the instantiated metric.

Definition at line 338 of file cover_tree.hpp.

References mpack::tree::CoverTree< MetricType, RootPointPolicy, StatisticType >::metric.

```
22.144.4.24  template<typename MetricType = metric::LMetric<2, true>, typename RootPointPolicy = FirstPointIsRoot, typename
StatisticType = EmptyStatistic> double mpack::tree::CoverTree< MetricType, RootPointPolicy, StatisticType
>::MinDistance ( const CoverTree< MetricType, RootPointPolicy, StatisticType > * other ) const
```

Return the minimum distance to another node.

```
22.144.4.25  template<typename MetricType = metric::LMetric<2, true>, typename RootPointPolicy = FirstPointIsRoot, typename
StatisticType = EmptyStatistic> double mpack::tree::CoverTree< MetricType, RootPointPolicy, StatisticType
>::MinDistance ( const CoverTree< MetricType, RootPointPolicy, StatisticType > * other, const double distance )
const
```

Return the minimum distance to another node given that the point-to-point distance has already been calculated.

```
22.144.4.26  template<typename MetricType = metric::LMetric<2, true>, typename RootPointPolicy = FirstPointIsRoot, typename
StatisticType = EmptyStatistic> double mpack::tree::CoverTree< MetricType, RootPointPolicy, StatisticType
>::MinDistance ( const arma::vec & other ) const
```

Return the minimum distance to another point.

```
22.144.4.27  template<typename MetricType = metric::LMetric<2, true>, typename RootPointPolicy = FirstPointIsRoot, typename
StatisticType = EmptyStatistic> double mpack::tree::CoverTree< MetricType, RootPointPolicy, StatisticType
>::MinDistance ( const arma::vec & other, const double distance ) const
```

Return the minimum distance to another point given that the distance from the center to the point has already been calculated.

```
22.144.4.28  template<typename MetricType = metric::LMetric<2, true>, typename RootPointPolicy = FirstPointIsRoot, typename
StatisticType = EmptyStatistic> double mlpack::tree::CoverTree< MetricType, RootPointPolicy, StatisticType
>::MinimumBoundDistance ( ) const [inline]
```

Get the minimum distance from the center to any bound edge (this is the same as furthestDescendantDistance).

Definition at line 332 of file cover_tree.hpp.

References mlpack::tree::CoverTree< MetricType, RootPointPolicy, StatisticType >::furthestDescendantDistance.

```
22.144.4.29  template<typename MetricType = metric::LMetric<2, true>, typename RootPointPolicy = FirstPointIsRoot, typename
StatisticType = EmptyStatistic> void mlpack::tree::CoverTree< MetricType, RootPointPolicy, StatisticType
>::MoveToUsedSet ( arma::Col< size_t > & indices, arma::vec & distances, size_t & nearSetSize, size_t & farSetSize,
size_t & usedSetSize, arma::Col< size_t > & childIndices, const size_t childFarSetSize, const size_t childUsedSetSize
) [private]
```

```
22.144.4.30  template<typename MetricType = metric::LMetric<2, true>, typename RootPointPolicy = FirstPointIsRoot, typename
StatisticType = EmptyStatistic> size_t mlpack::tree::CoverTree< MetricType, RootPointPolicy, StatisticType
>::NumChildren ( ) const [inline]
```

Get the number of children.

Definition at line 235 of file cover_tree.hpp.

References mlpack::tree::CoverTree< MetricType, RootPointPolicy, StatisticType >::children.

```
22.144.4.31  template<typename MetricType = metric::LMetric<2, true>, typename RootPointPolicy = FirstPointIsRoot, typename
StatisticType = EmptyStatistic> size_t mlpack::tree::CoverTree< MetricType, RootPointPolicy, StatisticType
>::NumDescendants ( ) const
```

Get the number of descendant points.

```
22.144.4.32  template<typename MetricType = metric::LMetric<2, true>, typename RootPointPolicy = FirstPointIsRoot, typename
StatisticType = EmptyStatistic> size_t mlpack::tree::CoverTree< MetricType, RootPointPolicy, StatisticType
>::NumPoints ( ) const [inline]
```

Definition at line 227 of file cover_tree.hpp.

```
22.144.4.33  template<typename MetricType = metric::LMetric<2, true>, typename RootPointPolicy = FirstPointIsRoot, typename
StatisticType = EmptyStatistic> CoverTree* mlpack::tree::CoverTree< MetricType, RootPointPolicy,
StatisticType >::Parent ( ) const [inline]
```

Get the parent node.

Definition at line 311 of file cover_tree.hpp.

References mlpack::tree::CoverTree< MetricType, RootPointPolicy, StatisticType >::parent.

```
22.144.4.34  template<typename MetricType = metric::LMetric<2, true>, typename RootPointPolicy = FirstPointIsRoot, typename
StatisticType = EmptyStatistic> CoverTree* & mlpack::tree::CoverTree< MetricType, RootPointPolicy,
StatisticType >::Parent ( ) [inline]
```

Modify the parent node.

Definition at line 313 of file cover_tree.hpp.

References `mlpack::tree::CoverTree< MetricType, RootPointPolicy, StatisticType >::parent`.

```
22.144.4.35  template<typename MetricType = metric::LMetric<2, true>, typename RootPointPolicy = FirstPointIsRoot, typename
StatisticType = EmptyStatistic> double mlpack::tree::CoverTree< MetricType, RootPointPolicy, StatisticType
>::ParentDistance ( ) const [inline]
```

Get the distance to the parent.

Definition at line 316 of file cover_tree.hpp.

References `mlpack::tree::CoverTree< MetricType, RootPointPolicy, StatisticType >::parentDistance`.

```
22.144.4.36  template<typename MetricType = metric::LMetric<2, true>, typename RootPointPolicy = FirstPointIsRoot, typename
StatisticType = EmptyStatistic> double& mlpack::tree::CoverTree< MetricType, RootPointPolicy, StatisticType
>::ParentDistance ( ) [inline]
```

Modify the distance to the parent.

Definition at line 318 of file cover_tree.hpp.

References `mlpack::tree::CoverTree< MetricType, RootPointPolicy, StatisticType >::parentDistance`.

```
22.144.4.37  template<typename MetricType = metric::LMetric<2, true>, typename RootPointPolicy = FirstPointIsRoot, typename
StatisticType = EmptyStatistic> size_t mlpack::tree::CoverTree< MetricType, RootPointPolicy, StatisticType
>::Point ( ) const [inline]
```

Get the index of the point which this node represents.

Definition at line 222 of file cover_tree.hpp.

References `mlpack::tree::CoverTree< MetricType, RootPointPolicy, StatisticType >::point`.

```
22.144.4.38  template<typename MetricType = metric::LMetric<2, true>, typename RootPointPolicy = FirstPointIsRoot, typename
StatisticType = EmptyStatistic> size_t mlpack::tree::CoverTree< MetricType, RootPointPolicy, StatisticType
>::Point ( const size_t ) const [inline]
```

For compatibility with other trees; the argument is ignored.

Definition at line 224 of file cover_tree.hpp.

References `mlpack::tree::CoverTree< MetricType, RootPointPolicy, StatisticType >::point`.

```
22.144.4.39  template<typename MetricType = metric::LMetric<2, true>, typename RootPointPolicy = FirstPointIsRoot, typename
StatisticType = EmptyStatistic> size_t mlpack::tree::CoverTree< MetricType, RootPointPolicy, StatisticType
>::PruneFarSet ( arma::Col< size_t > & indices, arma::vec & distances, const double bound, const size_t
nearSetSize, const size_t pointSetSize ) [private]
```

```
22.144.4.40  template<typename MetricType = metric::LMetric<2, true>, typename RootPointPolicy = FirstPointIsRoot, typename
StatisticType = EmptyStatistic> math::Range mlpack::tree::CoverTree< MetricType, RootPointPolicy,
StatisticType >::RangeDistance ( const CoverTree< MetricType, RootPointPolicy, StatisticType > * other ) const
```

Return the minimum and maximum distance to another node.

```
22.144.4.41  template<typename MetricType = metric::LMetric<2, true>, typename RootPointPolicy = FirstPointIsRoot, typename
StatisticType = EmptyStatistic> math::Range mpack::tree::CoverTree< MetricType, RootPointPolicy,
StatisticType >::RangeDistance ( const CoverTree< MetricType, RootPointPolicy, StatisticType > * other, const
double distance ) const
```

Return the minimum and maximum distance to another node given that the point-to-point distance has already been calculated.

```
22.144.4.42  template<typename MetricType = metric::LMetric<2, true>, typename RootPointPolicy = FirstPointIsRoot, typename
StatisticType = EmptyStatistic> math::Range mpack::tree::CoverTree< MetricType, RootPointPolicy,
StatisticType >::RangeDistance ( const arma::vec & other ) const
```

Return the minimum and maximum distance to another point.

```
22.144.4.43  template<typename MetricType = metric::LMetric<2, true>, typename RootPointPolicy = FirstPointIsRoot, typename
StatisticType = EmptyStatistic> math::Range mpack::tree::CoverTree< MetricType, RootPointPolicy,
StatisticType >::RangeDistance ( const arma::vec & other, const double distance ) const
```

Return the minimum and maximum distance to another point given that the point-to-point distance has already been calculated.

```
22.144.4.44  template<typename MetricType = metric::LMetric<2, true>, typename RootPointPolicy = FirstPointIsRoot, typename
StatisticType = EmptyStatistic> void mpack::tree::CoverTree< MetricType, RootPointPolicy, StatisticType
>::RemoveNewImplicitNodes ( ) [private]
```

Take a look at the last child (the most recently created one) and remove any implicit nodes that have been created.

```
22.144.4.45  template<typename MetricType = metric::LMetric<2, true>, typename RootPointPolicy = FirstPointIsRoot, typename
StatisticType = EmptyStatistic> int mpack::tree::CoverTree< MetricType, RootPointPolicy, StatisticType >::Scale
( ) const [inline]
```

Get the scale of this node.

Definition at line 249 of file cover_tree.hpp.

References mpack::tree::CoverTree< MetricType, RootPointPolicy, StatisticType >::scale.

```
22.144.4.46  template<typename MetricType = metric::LMetric<2, true>, typename RootPointPolicy = FirstPointIsRoot, typename
StatisticType = EmptyStatistic> int& mpack::tree::CoverTree< MetricType, RootPointPolicy, StatisticType
>::Scale ( ) [inline]
```

Modify the scale of this node. Be careful...

Definition at line 251 of file cover_tree.hpp.

References mpack::tree::CoverTree< MetricType, RootPointPolicy, StatisticType >::scale.

```
22.144.4.47 template<typename MetricType = metric::LMetric<2, true>, typename RootPointPolicy = FirstPointIsRoot, typename
StatisticType = EmptyStatistic> size_t mpack::tree::CoverTree< MetricType, RootPointPolicy, StatisticType
>::SortPointSet ( arma::Col< size_t > & indices, arma::vec & distances, const size_t childFarSetSize, const size_t
childUsedSetSize, const size_t farSetSize ) [private]
```

Assuming that the list of indices and distances is sorted as [childFarSet | childUsedSet | farSet | usedSet], resort the sets so the organization is [childFarSet | farSet | childUsedSet | usedSet].

The size_t parameters specify the sizes of each set in the array. Only the ordering of the indices and distances arrays will be modified (not their actual contents).

The size of any of the four sets can be zero and this method will handle that case accordingly.

Parameters

<i>indices</i>	List of indices to sort.
<i>distances</i>	List of distances to sort.
<i>childFarSetSize</i>	Number of points in child far set (childFarSet).
<i>childUsedSetSize</i>	Number of points in child used set (childUsedSet).
<i>farSetSize</i>	Number of points in far set (farSet).

```
22.144.4.48 template<typename MetricType = metric::LMetric<2, true>, typename RootPointPolicy = FirstPointIsRoot, typename
StatisticType = EmptyStatistic> size_t mpack::tree::CoverTree< MetricType, RootPointPolicy, StatisticType
>::SplitNearFar ( arma::Col< size_t > & indices, arma::vec & distances, const double bound, const size_t
pointSetSize ) [private]
```

Split the given indices and distances into a near and a far set, returning the number of points in the near set.

The distances must already be initialized. This will order the indices and distances such that the points in the near set make up the first part of the array and the far set makes up the rest: [nearSet | farSet].

Parameters

<i>indices</i>	List of indices; will be reordered.
<i>distances</i>	List of distances; will be reordered.
<i>bound</i>	If the distance is less than or equal to this bound, the point is placed into the near set.
<i>pointSetSize</i>	Size of point set (because we may be sorting a smaller list than the indices vector will hold).

```
22.144.4.49 template<typename MetricType = metric::LMetric<2, true>, typename RootPointPolicy = FirstPointIsRoot, typename
StatisticType = EmptyStatistic> const StatisticType& mpack::tree::CoverTree< MetricType, RootPointPolicy,
StatisticType >::Stat ( ) const [inline]
```

Get the statistic for this node.

Definition at line 259 of file cover_tree.hpp.

References mpack::tree::CoverTree< MetricType, RootPointPolicy, StatisticType >::stat.

```
22.144.4.50 template<typename MetricType = metric::LMetric<2, true>, typename RootPointPolicy = FirstPointIsRoot, typename
StatisticType = EmptyStatistic> StatisticType& mpack::tree::CoverTree< MetricType, RootPointPolicy,
StatisticType >::Stat ( ) [inline]
```

Modify the statistic for this node.

Definition at line 261 of file cover_tree.hpp.

References mlpack::tree::CoverTree< MetricType, RootPointPolicy, StatisticType >::stat.

```
22.144.4.51  template<typename MetricType = metric::LMetric<2, true>, typename RootPointPolicy = FirstPointIsRoot, typename
              StatisticType = EmptyStatistic> std::string mlpack::tree::CoverTree< MetricType, RootPointPolicy, StatisticType
              >::ToString ( ) const
```

Returns a string representation of this object.

22.144.5 Member Data Documentation

```
22.144.5.1  template<typename MetricType = metric::LMetric<2, true>, typename RootPointPolicy = FirstPointIsRoot, typename
              StatisticType = EmptyStatistic> double mlpack::tree::CoverTree< MetricType, RootPointPolicy, StatisticType
              >::base [private]
```

The base used to construct the tree.

Definition at line 354 of file cover_tree.hpp.

Referenced by mlpack::tree::CoverTree< MetricType, RootPointPolicy, StatisticType >::Base().

```
22.144.5.2  template<typename MetricType = metric::LMetric<2, true>, typename RootPointPolicy = FirstPointIsRoot,
              typename StatisticType = EmptyStatistic> std::vector<CoverTree*> mlpack::tree::CoverTree< MetricType,
              RootPointPolicy, StatisticType >::children [private]
```

The list of children; the first is the self-child.

Definition at line 348 of file cover_tree.hpp.

Referenced by mlpack::tree::CoverTree< MetricType, RootPointPolicy, StatisticType >::Child(), mlpack::tree::CoverTree< MetricType, RootPointPolicy, StatisticType >::Children(), mlpack::tree::CoverTree< MetricType, RootPointPolicy, StatisticType >::IsLeaf(), and mlpack::tree::CoverTree< MetricType, RootPointPolicy, StatisticType >::NumChildren().

```
22.144.5.3  template<typename MetricType = metric::LMetric<2, true>, typename RootPointPolicy = FirstPointIsRoot, typename
              StatisticType = EmptyStatistic> const arma::mat& mlpack::tree::CoverTree< MetricType, RootPointPolicy,
              StatisticType >::dataset [private]
```

Reference to the matrix which this tree is built on.

Definition at line 342 of file cover_tree.hpp.

Referenced by mlpack::tree::CoverTree< MetricType, RootPointPolicy, StatisticType >::Centroid(), and mlpack::tree::CoverTree< MetricType, RootPointPolicy, StatisticType >::Dataset().

```
22.144.5.4  template<typename MetricType = metric::LMetric<2, true>, typename RootPointPolicy = FirstPointIsRoot, typename
              StatisticType = EmptyStatistic> size_t mlpack::tree::CoverTree< MetricType, RootPointPolicy, StatisticType
              >::distanceComps [private]
```

Definition at line 475 of file cover_tree.hpp.

Referenced by mlpack::tree::CoverTree< MetricType, RootPointPolicy, StatisticType >::DistanceComps().

```
22.144.5.5  template<typename MetricType = metric::LMetric<2, true>, typename RootPointPolicy = FirstPointIsRoot, typename
           StatisticType = EmptyStatistic> double mlpack::tree::CoverTree< MetricType, RootPointPolicy, StatisticType
           >::furthestDescendantDistance  [private]
```

Distance to the furthest descendant.

Definition at line 369 of file cover_tree.hpp.

Referenced by mlpack::tree::CoverTree< MetricType, RootPointPolicy, StatisticType >::FurthestDescendantDistance(), and mlpack::tree::CoverTree< MetricType, RootPointPolicy, StatisticType >::MinimumBoundDistance().

```
22.144.5.6  template<typename MetricType = metric::LMetric<2, true>, typename RootPointPolicy = FirstPointIsRoot, typename
           StatisticType = EmptyStatistic> bool mlpack::tree::CoverTree< MetricType, RootPointPolicy, StatisticType
           >::localMetric  [private]
```

Whether or not we need to destroy the metric in the destructor.

Definition at line 372 of file cover_tree.hpp.

```
22.144.5.7  template<typename MetricType = metric::LMetric<2, true>, typename RootPointPolicy = FirstPointIsRoot, typename
           StatisticType = EmptyStatistic> MetricType* mlpack::tree::CoverTree< MetricType, RootPointPolicy, StatisticType
           >::metric  [private]
```

The metric used for this tree.

Definition at line 375 of file cover_tree.hpp.

Referenced by mlpack::tree::CoverTree< MetricType, RootPointPolicy, StatisticType >::Metric().

```
22.144.5.8  template<typename MetricType = metric::LMetric<2, true>, typename RootPointPolicy = FirstPointIsRoot, typename
           StatisticType = EmptyStatistic> size_t mlpack::tree::CoverTree< MetricType, RootPointPolicy, StatisticType
           >::numDescendants  [private]
```

The number of descendant points.

Definition at line 360 of file cover_tree.hpp.

```
22.144.5.9  template<typename MetricType = metric::LMetric<2, true>, typename RootPointPolicy = FirstPointIsRoot, typename
           StatisticType = EmptyStatistic> CoverTree* mlpack::tree::CoverTree< MetricType, RootPointPolicy, StatisticType
           >::parent  [private]
```

The parent node (NULL if this is the root of the tree).

Definition at line 363 of file cover_tree.hpp.

Referenced by mlpack::tree::CoverTree< MetricType, RootPointPolicy, StatisticType >::Parent().

```
22.144.5.10 template<typename MetricType = metric::LMetric<2, true>, typename RootPointPolicy = FirstPointIsRoot, typename
           StatisticType = EmptyStatistic> double mlpack::tree::CoverTree< MetricType, RootPointPolicy, StatisticType
           >::parentDistance  [private]
```

Distance to the parent.

Definition at line 366 of file cover_tree.hpp.

Referenced by mlpack::tree::CoverTree< MetricType, RootPointPolicy, StatisticType >::ParentDistance().

22.144.5.11 `template<typename MetricType = metric::LMetric<2, true>, typename RootPointPolicy = FirstPointIsRoot, typename StatisticType = EmptyStatistic> size_t mlpack::tree::CoverTree< MetricType, RootPointPolicy, StatisticType >::point [private]`

Index of the point in the matrix which this node represents.

Definition at line 345 of file `cover_tree.hpp`.

Referenced by `mlpack::tree::CoverTree< MetricType, RootPointPolicy, StatisticType >::Centroid()`, and `mlpack::tree::CoverTree< MetricType, RootPointPolicy, StatisticType >::Point()`.

22.144.5.12 `template<typename MetricType = metric::LMetric<2, true>, typename RootPointPolicy = FirstPointIsRoot, typename StatisticType = EmptyStatistic> int mlpack::tree::CoverTree< MetricType, RootPointPolicy, StatisticType >::scale [private]`

Scale level of the node.

Definition at line 351 of file `cover_tree.hpp`.

Referenced by `mlpack::tree::CoverTree< MetricType, RootPointPolicy, StatisticType >::Scale()`.

22.144.5.13 `template<typename MetricType = metric::LMetric<2, true>, typename RootPointPolicy = FirstPointIsRoot, typename StatisticType = EmptyStatistic> StatisticType mlpack::tree::CoverTree< MetricType, RootPointPolicy, StatisticType >::stat [private]`

The instantiated statistic.

Definition at line 357 of file `cover_tree.hpp`.

Referenced by `mlpack::tree::CoverTree< MetricType, RootPointPolicy, StatisticType >::Stat()`.

The documentation for this class was generated from the following file:

- `src/mlpack/core/tree/cover_tree/cover_tree.hpp`

22.145 `mlpack::tree::CoverTree< MetricType, RootPointPolicy, StatisticType >::DualTreeTraverser< RuleType >` ↵ Class Template Reference

A dual-tree cover tree traverser; see `dual_tree_traverser.hpp`.

Classes

- struct **DualCoverTreeMapEntry**
Struct used for traversal.

Public Member Functions

- **DualTreeTraverser** (RuleType &rule)
Initialize the dual tree traverser with the given rule type.
- `size_t NumBaseCases ()` const
- `size_t NumPrunes ()` const

Get the number of pruned nodes.

- `size_t & NumPrunes ()`

Modify the number of pruned nodes.

- `size_t NumScores () const`
- `size_t NumVisited () const`
- `void Traverse (CoverTree &queryNode, CoverTree &referenceNode)`

Traverse the two specified trees.

Private Member Functions

- `void PruneMap (CoverTree &queryNode, std::map< int, std::vector< DualCoverTreeMapEntry > > &referenceMap, std::map< int, std::vector< DualCoverTreeMapEntry > > &childMap)`

Prepare map for recursion.

- `void ReferenceRecursion (CoverTree &queryNode, std::map< int, std::vector< DualCoverTreeMapEntry > > &referenceMap)`
- `void Traverse (CoverTree &queryNode, std::map< int, std::vector< DualCoverTreeMapEntry > > &referenceMap)`

Helper function for traversal of the two trees.

Private Attributes

- `size_t numPrunes`

The number of pruned nodes.

- `RuleType & rule`

The instantiated rule set for pruning branches.

22.145.1 Detailed Description

```
template<typename MetricType = metric::LMetric<2, true>, typename RootPointPolicy = FirstPointIsRoot, typename StatisticType = EmptyStatistic>
template<typename RuleType> class mpack::tree::CoverTree< MetricType, RootPointPolicy, StatisticType >::DualTreeTraverser< RuleType >
```

A dual-tree cover tree traverser; see `dual_tree_traverser.hpp`.

Definition at line 216 of file `cover_tree.hpp`.

22.145.2 Constructor & Destructor Documentation

22.145.2.1 `template<typename MetricType = metric::LMetric<2, true>, typename RootPointPolicy = FirstPointIsRoot, typename StatisticType = EmptyStatistic> template<typename RuleType > mpack::tree::CoverTree< MetricType, RootPointPolicy, StatisticType >::DualTreeTraverser< RuleType >::DualTreeTraverser (RuleType & rule)`

Initialize the dual tree traverser with the given rule type.

22.145.3 Member Function Documentation

22.145.3.1 `template<typename MetricType = metric::LMetric<2, true>, typename RootPointPolicy = FirstPointIsRoot, typename StatisticType = EmptyStatistic> template<typename RuleType > size_t mlpack::tree::CoverTree< MetricType, RootPointPolicy, StatisticType >::DualTreeTraverser< RuleType >::NumBaseCases () const [inline]`

Definition at line 50 of file dual_tree_traverser.hpp.

22.145.3.2 `template<typename MetricType = metric::LMetric<2, true>, typename RootPointPolicy = FirstPointIsRoot, typename StatisticType = EmptyStatistic> template<typename RuleType > size_t mlpack::tree::CoverTree< MetricType, RootPointPolicy, StatisticType >::DualTreeTraverser< RuleType >::NumPrunes () const [inline]`

Get the number of pruned nodes.

Definition at line 42 of file dual_tree_traverser.hpp.

22.145.3.3 `template<typename MetricType = metric::LMetric<2, true>, typename RootPointPolicy = FirstPointIsRoot, typename StatisticType = EmptyStatistic> template<typename RuleType > size_t& mlpack::tree::CoverTree< MetricType, RootPointPolicy, StatisticType >::DualTreeTraverser< RuleType >::NumPrunes () [inline]`

Modify the number of pruned nodes.

Definition at line 44 of file dual_tree_traverser.hpp.

22.145.3.4 `template<typename MetricType = metric::LMetric<2, true>, typename RootPointPolicy = FirstPointIsRoot, typename StatisticType = EmptyStatistic> template<typename RuleType > size_t mlpack::tree::CoverTree< MetricType, RootPointPolicy, StatisticType >::DualTreeTraverser< RuleType >::NumScores () const [inline]`

Definition at line 49 of file dual_tree_traverser.hpp.

22.145.3.5 `template<typename MetricType = metric::LMetric<2, true>, typename RootPointPolicy = FirstPointIsRoot, typename StatisticType = EmptyStatistic> template<typename RuleType > size_t mlpack::tree::CoverTree< MetricType, RootPointPolicy, StatisticType >::DualTreeTraverser< RuleType >::NumVisited () const [inline]`

Definition at line 48 of file dual_tree_traverser.hpp.

22.145.3.6 `template<typename MetricType = metric::LMetric<2, true>, typename RootPointPolicy = FirstPointIsRoot, typename StatisticType = EmptyStatistic> template<typename RuleType > void mlpack::tree::CoverTree< MetricType, RootPointPolicy, StatisticType >::DualTreeTraverser< RuleType >::PruneMap (CoverTree & queryNode, std::map< int, std::vector< DualCoverTreeMapEntry > > & referenceMap, std::map< int, std::vector< DualCoverTreeMapEntry > > & childMap) [private]`

Prepare map for recursion.

22.145.3.7 `template<typename MetricType = metric::LMetric<2, true>, typename RootPointPolicy = FirstPointIsRoot, typename StatisticType = EmptyStatistic> template<typename RuleType > void mlpack::tree::CoverTree< MetricType, RootPointPolicy, StatisticType >::DualTreeTraverser< RuleType >::ReferenceRecursion (CoverTree & queryNode, std::map< int, std::vector< DualCoverTreeMapEntry > > & referenceMap) [private]`

22.145.3.8 `template<typename MetricType = metric::LMetric<2, true>, typename RootPointPolicy = FirstPointIsRoot, typename
StatisticType = EmptyStatistic> template<typename RuleType > void mlpack::tree::CoverTree< MetricType,
RootPointPolicy, StatisticType >::DualTreeTraverser< RuleType >::Traverse (CoverTree & queryNode,
CoverTree & referenceNode)`

Traverse the two specified trees.

Parameters

<code>queryNode</code>	Root of query tree.
<code>referenceNode</code>	Root of reference tree.

22.145.3.9 `template<typename MetricType = metric::LMetric<2, true>, typename RootPointPolicy = FirstPointIsRoot, typename StatisticType = EmptyStatistic> template<typename RuleType > void mlpack::tree::CoverTree< MetricType, RootPointPolicy, StatisticType >::DualTreeTraverser< RuleType >:: Traverse (CoverTree & queryNode, std::map< int, std::vector< DualCoverTreeMapEntry > > & referenceMap) [private]`

Helper function for traversal of the two trees.

22.145.4 Member Data Documentation

22.145.4.1 `template<typename MetricType = metric::LMetric<2, true>, typename RootPointPolicy = FirstPointIsRoot, typename StatisticType = EmptyStatistic> template<typename RuleType > size_t mlpack::tree::CoverTree< MetricType, RootPointPolicy, StatisticType >::DualTreeTraverser< RuleType >::numPrunes [private]`

The number of pruned nodes.

Definition at line 57 of file `dual_tree_traverser.hpp`.

22.145.4.2 `template<typename MetricType = metric::LMetric<2, true>, typename RootPointPolicy = FirstPointIsRoot, typename StatisticType = EmptyStatistic> template<typename RuleType > RuleType& mlpack::tree::CoverTree< MetricType, RootPointPolicy, StatisticType >::DualTreeTraverser< RuleType >::rule [private]`

The instantiated rule set for pruning branches.

Definition at line 54 of file `dual_tree_traverser.hpp`.

The documentation for this class was generated from the following files:

- `src/mlpack/core/tree/cover_tree/cover_tree.hpp`
- `src/mlpack/core/tree/cover_tree/dual_tree_traverser.hpp`

22.146 `mlpack::tree::CoverTree< MetricType, RootPointPolicy, StatisticType >::DualTreeTraverser< RuleType >::DualCoverTreeMapEntry` Struct Reference

Struct used for traversal.

Public Member Functions

- `bool operator< (const DualCoverTreeMapEntry &other) const`
Comparison operator, for sorting within the map.

Public Attributes

- `double baseCase`

The base case.

- **CoverTree**< MetricType, RootPointPolicy, StatisticType > * **referenceNode**

The node this entry refers to.

- double **score**

The score of the node.

- RuleType::TraversalInfoType **traversalInfo**

The traversal info associated with the call to Score() for this entry.

22.146.1 Detailed Description

```
template<typename MetricType = metric::LMetric<2, true>, typename RootPointPolicy = FirstPointIsRoot, typename StatisticType = EmptyStatistic>
template<typename RuleType> struct mlpack::tree::CoverTree< MetricType, RootPointPolicy, StatisticType >::DualTreeTraverser< RuleType >::DualCoverTreeMapEntry
```

Struct used for traversal.

Definition at line 60 of file dual_tree_traverser.hpp.

22.146.2 Member Function Documentation

22.146.2.1 `template<typename MetricType = metric::LMetric<2, true>, typename RootPointPolicy = FirstPointIsRoot, typename StatisticType = EmptyStatistic> template<typename RuleType > bool mlpack::tree::CoverTree< MetricType, RootPointPolicy, StatisticType >::DualTreeTraverser< RuleType >::DualCoverTreeMapEntry::operator< (const DualCoverTreeMapEntry & other) const [inline]`

Comparison operator, for sorting within the map.

Definition at line 72 of file dual_tree_traverser.hpp.

References `mlpack::tree::CoverTree< MetricType, RootPointPolicy, StatisticType >::DualTreeTraverser< RuleType >::DualCoverTreeMapEntry::baseCase`, and `mlpack::tree::CoverTree< MetricType, RootPointPolicy, StatisticType >::DualTreeTraverser< RuleType >::DualCoverTreeMapEntry::score`.

22.146.3 Member Data Documentation

22.146.3.1 `template<typename MetricType = metric::LMetric<2, true>, typename RootPointPolicy = FirstPointIsRoot, typename StatisticType = EmptyStatistic> template<typename RuleType > double mlpack::tree::CoverTree< MetricType, RootPointPolicy, StatisticType >::DualTreeTraverser< RuleType >::DualCoverTreeMapEntry::baseCase`

The base case.

Definition at line 67 of file dual_tree_traverser.hpp.

Referenced by `mlpack::tree::CoverTree< MetricType, RootPointPolicy, StatisticType >::DualTreeTraverser< RuleType >::DualCoverTreeMapEntry::operator<()`.

22.146.3.2 `template<typename MetricType = metric::LMetric<2, true>, typename RootPointPolicy = FirstPointIsRoot, typename StatisticType = EmptyStatistic> template<typename RuleType > CoverTree<MetricType, RootPointPolicy, StatisticType>* mlpack::tree::CoverTree< MetricType, RootPointPolicy, StatisticType >::DualTreeTraverser< RuleType >::DualCoverTreeMapEntry::referenceNode`

The node this entry refers to.

Definition at line 63 of file `dual_tree_traverser.hpp`.

22.146.3.3 `template<typename MetricType = metric::LMetric<2, true>, typename RootPointPolicy = FirstPointIsRoot, typename StatisticType = EmptyStatistic> template<typename RuleType > double mlpack::tree::CoverTree< MetricType, RootPointPolicy, StatisticType >::DualTreeTraverser< RuleType >::DualCoverTreeMapEntry::score`

The score of the node.

Definition at line 65 of file `dual_tree_traverser.hpp`.

Referenced by `mlpack::tree::CoverTree< MetricType, RootPointPolicy, StatisticType >::DualTreeTraverser< RuleType >::DualCoverTreeMapEntry::operator<()`.

22.146.3.4 `template<typename MetricType = metric::LMetric<2, true>, typename RootPointPolicy = FirstPointIsRoot, typename StatisticType = EmptyStatistic> template<typename RuleType > RuleType::TraversallInfoType mlpack::tree::CoverTree< MetricType, RootPointPolicy, StatisticType >::DualTreeTraverser< RuleType >::DualCoverTreeMapEntry::traversalInfo`

The traversal info associated with the call to `Score()` for this entry.

Definition at line 69 of file `dual_tree_traverser.hpp`.

The documentation for this struct was generated from the following file:

- `src/mlpack/core/tree/cover_tree/dual_tree_traverser.hpp`

22.147 `mlpack::tree::CoverTree< MetricType, RootPointPolicy, StatisticType >::SingleTreeTraverser< RuleType >` Class Template Reference

A single-tree cover tree traverser; see `single_tree_traverser.hpp` for implementation.

Public Member Functions

- **SingleTreeTraverser** (`RuleType &rule`)
Initialize the single tree traverser with the given rule.
- `size_t NumPrunes ()` `const`
Get the number of prunes so far.
- `size_t & NumPrunes ()`
Set the number of prunes (good for a reset to 0).
- `void Traverse (const size_t queryIndex, CoverTree &referenceNode)`
Traverse the tree with the given point.

Private Attributes

- `size_t numPrunes`
The number of nodes which have been pruned during traversal.
- `RuleType & rule`
Reference to the rules with which the tree will be traversed.

22.147.1 Detailed Description

```
template<typename MetricType = metric::LMetric<2, true>, typename RootPointPolicy = FirstPointIsRoot, typename StatisticType = EmptyStatistic>
template<typename RuleType> class mlpack::tree::CoverTree< MetricType, RootPointPolicy, StatisticType >::SingleTreeTraverser< RuleType >
```

A single-tree cover tree traverser; see `single_tree_traverser.hpp` for implementation.

Definition at line 212 of file `cover_tree.hpp`.

22.147.2 Constructor & Destructor Documentation

```
22.147.2.1 template<typename MetricType = metric::LMetric<2, true>, typename RootPointPolicy = FirstPointIsRoot, typename StatisticType = EmptyStatistic>
template<typename RuleType > mlpack::tree::CoverTree< MetricType, RootPointPolicy, StatisticType >::SingleTreeTraverser< RuleType >::SingleTreeTraverser ( RuleType & rule )
```

Initialize the single tree traverser with the given rule.

22.147.3 Member Function Documentation

```
22.147.3.1 template<typename MetricType = metric::LMetric<2, true>, typename RootPointPolicy = FirstPointIsRoot, typename StatisticType = EmptyStatistic>
template<typename RuleType > size_t mlpack::tree::CoverTree< MetricType, RootPointPolicy, StatisticType >::SingleTreeTraverser< RuleType >::NumPrunes ( ) const [inline]
```

Get the number of prunes so far.

Definition at line 46 of file `single_tree_traverser.hpp`.

```
22.147.3.2 template<typename MetricType = metric::LMetric<2, true>, typename RootPointPolicy = FirstPointIsRoot, typename StatisticType = EmptyStatistic>
template<typename RuleType > size_t& mlpack::tree::CoverTree< MetricType, RootPointPolicy, StatisticType >::SingleTreeTraverser< RuleType >::NumPrunes ( ) [inline]
```

Set the number of prunes (good for a reset to 0).

Definition at line 48 of file `single_tree_traverser.hpp`.

```
22.147.3.3 template<typename MetricType = metric::LMetric<2, true>, typename RootPointPolicy = FirstPointIsRoot, typename StatisticType = EmptyStatistic>
template<typename RuleType > void mlpack::tree::CoverTree< MetricType, RootPointPolicy, StatisticType >::SingleTreeTraverser< RuleType >::Traverse ( const size_t queryIndex, CoverTree & referenceNode )
```

Traverse the tree with the given point.

Parameters

<i>queryIndex</i>	The index of the point in the query set which is used as the query point.
<i>referenceNode</i>	The tree node to be traversed.

22.147.4 Member Data Documentation

22.147.4.1 `template<typename MetricType = metric::LMetric<2, true>, typename RootPointPolicy = FirstPointIsRoot, typename StatisticType = EmptyStatistic> template<typename RuleType > size_t mlpack::tree::CoverTree< MetricType, RootPointPolicy, StatisticType >::SingleTreeTraverser< RuleType >::numPrunes [private]`

The number of nodes which have been pruned during traversal.

Definition at line 55 of file `single_tree_traverser.hpp`.

22.147.4.2 `template<typename MetricType = metric::LMetric<2, true>, typename RootPointPolicy = FirstPointIsRoot, typename StatisticType = EmptyStatistic> template<typename RuleType > RuleType& mlpack::tree::CoverTree< MetricType, RootPointPolicy, StatisticType >::SingleTreeTraverser< RuleType >::rule [private]`

Reference to the rules with which the tree will be traversed.

Definition at line 52 of file `single_tree_traverser.hpp`.

The documentation for this class was generated from the following files:

- `src/mlpack/core/tree/cover_tree/cover_tree.hpp`
- `src/mlpack/core/tree/cover_tree/single_tree_traverser.hpp`

22.148 mlpack::tree::EmptyStatistic Class Reference

Empty statistic if you are not interested in storing statistics in your tree.

Public Member Functions

- **EmptyStatistic ()**
- `template<typename TreeType > EmptyStatistic (TreeType &)`
This constructor is called when a node is finished being created.
- **~EmptyStatistic ()**
- `std::string ToString () const`
Returns a string representation of this object.

22.148.1 Detailed Description

Empty statistic if you are not interested in storing statistics in your tree.

Use this as a template for your own.

Definition at line 26 of file `statistic.hpp`.

22.148.2 Constructor & Destructor Documentation

22.148.2.1 `mlpack::tree::EmptyStatistic::EmptyStatistic () [inline]`

Definition at line 29 of file `statistic.hpp`.

22.148.2.2 `mlpack::tree::EmptyStatistic::~~EmptyStatistic ()` `[inline]`

Definition at line 30 of file `statistic.hpp`.

22.148.2.3 `template<typename TreeType> mlpack::tree::EmptyStatistic::EmptyStatistic (TreeType &)` `[inline]`

This constructor is called when a node is finished being created.

The node is finished, and its children are finished, but it is not necessarily true that the statistics of other nodes are initialized yet.

Parameters

<i>node</i>	Node which this corresponds to.
-------------	---------------------------------

Definition at line 40 of file `statistic.hpp`.

22.148.3 Member Function Documentation

22.148.3.1 `std::string mlpack::tree::EmptyStatistic::ToString () const` `[inline]`

Returns a string representation of this object.

Definition at line 46 of file `statistic.hpp`.

The documentation for this class was generated from the following file:

- `src/mlpack/core/tree/statistic.hpp`

22.149 `mlpack::tree::ExampleTree< MetricType, StatisticType, MatType >` Class Template Reference

This is not an actual space tree but instead an example tree that exists to show and document all the functions that mlpack trees must implement.

Public Member Functions

- **ExampleTree** (const MatType &dataset, MetricType &metric)
This constructor will build the tree given a dataset and an instantiated metric.
- void **Centroid** (arma::vec ¢roid) const
Fill the given vector with the center of the node.
- const **ExampleTree & Child** (const size_t i) const
Return a particular child of this node.
- **ExampleTree & Child** (const size_t i)
Modify a particular child of this node.
- size_t **Descendant** (const size_t i) const
Get the index of a particular descendant point.
- double **FurthestDescendantDistance** () const
Get the distance from the center of the node to the furthest descendant point of this node.
- double **MaxDistance** (const MatType &point) const

- Return the maximum distance between this node and a point.*

 - double **MaxDistance** (const **ExampleTree** &other) const

Return the maximum distance between this node and another node.
- const MetricType & **Metric** () const

Get the instantiated metric for this node.
- MetricType & **Metric** ()

Modify the instantiated metric for this node.
- double **MinDistance** (const MatType &point) const

Return the minimum distance between this node and a point.
- double **MinDistance** (const **ExampleTree** &other) const

Return the minimum distance between this node and another node.
- size_t **NumChildren** () const

Return the number of children of this node.
- size_t **NumDescendants** () const

Get the number of descendant points.
- size_t **NumPoints** () const

Return the number of points held in this node.
- **ExampleTree** * **Parent** () const

Return the parent node (NULL if this is the root of the tree).
- double **ParentDistance** () const

Get the distance from the center of this node to the center of the parent node.
- size_t **Point** (const size_t i) const

Return the index of a particular point of this node.
- **math::Range** **RangeDistance** (const MatType &point) const

*Return both the minimum and maximum distances between this node and a point as a **math::Range** (p. 359) object.*
- **math::Range** **RangeDistance** (const **ExampleTree** &other) const

*Return both the minimum and maximum distances between this node and another node as a **math::Range** (p. 359) object.*
- const StatisticType & **Stat** () const

Get the statistic for this node.
- StatisticType & **Stat** ()

Modify the statistic for this node.

Private Attributes

- MetricType & **metric**

*This member is just here so the **ExampleTree** (p. 630) compiles without warnings.*
- StatisticType **stat**

*This member is just here so the **ExampleTree** (p. 630) compiles without warnings.*

22.149.1 Detailed Description

```
template<typename MetricType = metric::LMetric<2, true>, typename StatisticType = EmptyStatistic, typename MatType = arma::mat>
class mlpack::tree::ExampleTree< MetricType, StatisticType, MatType >
```

This is not an actual space tree but instead an example tree that exists to show and document all the functions that mlpack trees must implement.

For a better overview of trees, see `trees`. Also be aware that the implementations of each of the methods in this example tree are entirely fake and do not work; this example tree exists for its API, not its implementation.

Note that trees often have different properties. These properties are known at compile-time through the `mlpack::tree::TreeTraits` (p. 645) class, and some properties may imply the existence (or non-existence) of certain functions. Refer to the `TreeTraits` (p. 645) for more documentation on that.

The three template parameters below must be template parameters to the tree, in the order given below. More template parameters are fine, but they must come after the first three.

Template Parameters

<i>MetricType</i>	This defines the space in which the tree will be built. For some trees, arbitrary metrics cannot be used, and a template metaprogramming approach should be used to issue a compile-time error if a metric cannot be used with a specific tree type. One example is the <code>tree::BinarySpaceTree</code> (p. 566) tree type, which cannot work with the <code>metric::IPMetric</code> (p. 365) class.
<i>StatisticType</i>	A tree node can hold a statistic, which is sometimes useful for various dual-tree algorithms. The tree itself does not need to know anything about how the statistic works, but it needs to hold a <code>StatisticType</code> in each node. It can be assumed that the <code>StatisticType</code> class has a constructor <code>StatisticType(const ExampleTree&)</code> .
<i>MatType</i>	A tree could be built on a dense matrix or a sparse matrix. All <code>mlpack</code> trees should be able to support any Armadillo-compatible matrix type. When the tree is written it should be assumed that <code>MatType</code> has the same functionality as <code>arma::mat</code> .

Definition at line 58 of file `example_tree.hpp`.

22.149.2 Constructor & Destructor Documentation

22.149.2.1 `template<typename MetricType = metric::LMetric<2, true>, typename StatisticType = EmptyStatistic, typename MatType = arma::mat> mlpack::tree::ExampleTree< MetricType, StatisticType, MatType >::ExampleTree (const MatType & dataset, MetricType & metric)`

This constructor will build the tree given a dataset and an instantiated metric.

Note that the parameter is a `MatType&` and not an `arma::mat&`. The dataset is not modified by the tree-building process (if it is, see the documentation for `mlpack::tree::TreeTraits::RearrangesDataset` (p. 647) for how to deal with that situation). The `MetricType` parameter is necessary even though some metrics do not hold any state. This is so that the tree does not have to worry about instantiating the metric (if the tree had to worry about this, this would almost certainly incur additional runtime complexity and a larger runtime size of the tree node objects, which is to be avoided). The metric can't be `const`, in case `MetricType::Evaluate()` is non-`const`.

When this constructor is finished, the entire tree will be built and ready to use. The constructor should call the constructor of the statistic for each node that is built (see `tree::EmptyStatistic` (p. 629) for more information).

Parameters

<i>dataset</i>	The dataset that the tree will be built on.
<i>metric</i>	The instantiated metric to use to build the dataset.

22.149.3 Member Function Documentation

```
22.149.3.1  template<typename MetricType = metric::LMetric<2, true>, typename StatisticType = EmptyStatistic, typename MatType = arma::mat> void mlpack::tree::ExampleTree< MetricType, StatisticType, MatType >::Centroid ( arma::vec & centroid ) const
```

Fill the given vector with the center of the node.

Parameters

<i>centroid</i>	Vector to be filled with the center of the node.
-----------------	--

22.149.3.2 `template<typename MetricType = metric::LMetric<2, true>, typename StatisticType = EmptyStatistic, typename MatType = arma::mat> const ExampleTree& mlpack::tree::ExampleTree< MetricType, StatisticType, MatType >::Child (const size_t i) const`

Return a particular child of this node.

22.149.3.3 `template<typename MetricType = metric::LMetric<2, true>, typename StatisticType = EmptyStatistic, typename MatType = arma::mat> ExampleTree& mlpack::tree::ExampleTree< MetricType, StatisticType, MatType >::Child (const size_t i)`

Modify a particular child of this node.

22.149.3.4 `template<typename MetricType = metric::LMetric<2, true>, typename StatisticType = EmptyStatistic, typename MatType = arma::mat> size_t mlpack::tree::ExampleTree< MetricType, StatisticType, MatType >::Descendant (const size_t i) const`

Get the index of a particular descendant point.

The ordering of the descendants does not matter, as long as calling Descendant(0) through Descendant(**Num↵** Descendants()) (p. 636) - 1) will return the indices of every unique descendant point of the node.

22.149.3.5 `template<typename MetricType = metric::LMetric<2, true>, typename StatisticType = EmptyStatistic, typename MatType = arma::mat> double mlpack::tree::ExampleTree< MetricType, StatisticType, MatType >::FurthestDescendantDistance () const`

Get the distance from the center of the node to the furthest descendant point of this node.

This does not necessarily need to be the exact furthest descendant distance but instead can be an upper bound. See the definitions in trees for more information.

22.149.3.6 `template<typename MetricType = metric::LMetric<2, true>, typename StatisticType = EmptyStatistic, typename MatType = arma::mat> double mlpack::tree::ExampleTree< MetricType, StatisticType, MatType >::MaxDistance (const MatType & point) const`

Return the maximum distance between this node and a point.

It is not required that the exact maximum distance between the node and the point is returned but instead an upper bound on the maximum distance will suffice. See the definitions in trees for more information.

Parameters

<i>point</i>	Point to return [upper bound on] maximum distance to.
--------------	---

```
22.149.3.7  template<typename MetricType = metric::LMetric<2, true>, typename StatisticType = EmptyStatistic, typename MatType = arma::mat> double mlpack::tree::ExampleTree< MetricType, StatisticType, MatType >::MaxDistance ( const ExampleTree< MetricType, StatisticType, MatType > & other ) const
```

Return the maximum distance between this node and another node.

It is not required that the exact maximum distance between the two nodes be returned but instead an upper bound on the maximum distance will suffice. See the definitions in trees for more information.

Parameters

<i>node</i>	Node to return [upper bound on] maximum distance to.
-------------	--

```
22.149.3.8  template<typename MetricType = metric::LMetric<2, true>, typename StatisticType = EmptyStatistic, typename MatType = arma::mat> const MetricType& mlpack::tree::ExampleTree< MetricType, StatisticType, MatType >::Metric ( ) const
```

Get the instantiated metric for this node.

```
22.149.3.9  template<typename MetricType = metric::LMetric<2, true>, typename StatisticType = EmptyStatistic, typename MatType = arma::mat> MetricType& mlpack::tree::ExampleTree< MetricType, StatisticType, MatType >::Metric ( )
```

Modify the instantiated metric for this node.

```
22.149.3.10 template<typename MetricType = metric::LMetric<2, true>, typename StatisticType = EmptyStatistic, typename MatType = arma::mat> double mlpack::tree::ExampleTree< MetricType, StatisticType, MatType >::MinDistance ( const MatType & point ) const
```

Return the minimum distance between this node and a point.

It is not required that the exact minimum distance between the node and the point is returned but instead a lower bound on the minimum distance will suffice. See the definitions in trees for more information.

Parameters

<i>point</i>	Point to return [lower bound on] minimum distance to.
--------------	---

```
22.149.3.11 template<typename MetricType = metric::LMetric<2, true>, typename StatisticType = EmptyStatistic, typename MatType = arma::mat> double mlpack::tree::ExampleTree< MetricType, StatisticType, MatType >::MinDistance ( const ExampleTree< MetricType, StatisticType, MatType > & other ) const
```

Return the minimum distance between this node and another node.

It is not required that the exact minimum distance between the two nodes be returned but instead a lower bound on the minimum distance will suffice. See the definitions in trees for more information.

Parameters

<i>node</i>	Node to return [lower bound on] minimum distance to.
-------------	--

```
22.149.3.12  template<typename MetricType = metric::LMetric<2, true>, typename StatisticType = EmptyStatistic, typename
              MatType = arma::mat> size_t mlpack::tree::ExampleTree< MetricType, StatisticType, MatType >::NumChildren (
              ) const
```

Return the number of children of this node.

```
22.149.3.13  template<typename MetricType = metric::LMetric<2, true>, typename StatisticType = EmptyStatistic,
              typename MatType = arma::mat> size_t mlpack::tree::ExampleTree< MetricType, StatisticType, MatType
              >::NumDescendants ( ) const
```

Get the number of descendant points.

This is the number of unique points held in this node plus the number of points held in all descendant nodes. This could be calculated at build-time and cached, or could be calculated at run-time. This may be harder to calculate for trees that may hold points in multiple nodes (like cover trees and spill trees, for instance).

```
22.149.3.14  template<typename MetricType = metric::LMetric<2, true>, typename StatisticType = EmptyStatistic, typename
              MatType = arma::mat> size_t mlpack::tree::ExampleTree< MetricType, StatisticType, MatType >::NumPoints ( )
              const
```

Return the number of points held in this node.

```
22.149.3.15  template<typename MetricType = metric::LMetric<2, true>, typename StatisticType = EmptyStatistic, typename
              MatType = arma::mat> ExampleTree* mlpack::tree::ExampleTree< MetricType, StatisticType, MatType
              >::Parent ( ) const
```

Return the parent node (NULL if this is the root of the tree).

```
22.149.3.16  template<typename MetricType = metric::LMetric<2, true>, typename StatisticType = EmptyStatistic, typename
              MatType = arma::mat> double mlpack::tree::ExampleTree< MetricType, StatisticType, MatType >::ParentDistance
              ( ) const
```

Get the distance from the center of this node to the center of the parent node.

```
22.149.3.17  template<typename MetricType = metric::LMetric<2, true>, typename StatisticType = EmptyStatistic, typename
              MatType = arma::mat> size_t mlpack::tree::ExampleTree< MetricType, StatisticType, MatType >::Point ( const
              size_t i ) const
```

Return the index of a particular point of this node.

mlpack trees do not, in general, hold the actual dataset, and instead just hold the indices of the points they contain. Thus, you might use this function in code like this:

```
arma::vec thirdPoint = dataset.col(treeNode.Point(2));
```

```
22.149.3.18  template<typename MetricType = metric::LMetric<2, true>, typename StatisticType = EmptyStatistic, typename
              MatType = arma::mat> math::Range mlpack::tree::ExampleTree< MetricType, StatisticType, MatType
              >::RangeDistance ( const MatType & point ) const
```

Return both the minimum and maximum distances between this node and a point as a **math::Range** (p. 359) object.

This overload is given because it is possible that, for some tree types, calculation of both at once is faster than a call to **MinDistance()** (p. 635) then **MaxDistance()** (p. 634). It is not necessary that the minimum and maximum distances be exact; it is sufficient to return a lower bound on the minimum distance and an upper bound on the maximum distance. See the definitions in trees for more information.

Parameters

<i>point</i>	Point to return [bounds on] minimum and maximum distances to.
--------------	---

```
22.149.3.19  template<typename MetricType = metric::LMetric<2, true>, typename StatisticType = EmptyStatistic, typename
              MatType = arma::mat> math::Range mlpack::tree::ExampleTree< MetricType, StatisticType, MatType
              >::RangeDistance ( const ExampleTree< MetricType, StatisticType, MatType > & other ) const
```

Return both the minimum and maximum distances between this node and another node as a **math::Range** (p. 359) object.

This overload is given because it is possible that, for some tree types, calculation of both at once is faster than a call to **MinDistance()** (p. 635) then **MaxDistance()** (p. 634). It is not necessary that the minimum and maximum distances be exact; it is sufficient to return a lower bound on the minimum distance and an upper bound on the maximum distance. See the definitions in trees for more information.

Parameters

<i>node</i>	Node to return [bounds on] minimum and maximum distances to.
-------------	--

```
22.149.3.20  template<typename MetricType = metric::LMetric<2, true>, typename StatisticType = EmptyStatistic, typename
              MatType = arma::mat> const StatisticType& mlpack::tree::ExampleTree< MetricType, StatisticType, MatType
              >::Stat ( ) const
```

Get the statistic for this node.

```
22.149.3.21  template<typename MetricType = metric::LMetric<2, true>, typename StatisticType = EmptyStatistic, typename
              MatType = arma::mat> StatisticType& mlpack::tree::ExampleTree< MetricType, StatisticType, MatType >::Stat (
              )
```

Modify the statistic for this node.

22.149.4 Member Data Documentation

```
22.149.4.1  template<typename MetricType = metric::LMetric<2, true>, typename StatisticType = EmptyStatistic, typename
              MatType = arma::mat> MetricType& mlpack::tree::ExampleTree< MetricType, StatisticType, MatType >::metric
              [private]
```

This member is just here so the **ExampleTree** (p. 630) compiles without warnings.

It is not required to be a member in every type of tree. Be aware that storing the metric as a member and not a reference may mean that for some metrics (such as **metric::MahalanobisDistance** (p. 368) in high dimensionality) may incur lots of unnecessary matrix copying.

Definition at line 236 of file example_tree.hpp.

```
22.149.4.2 template<typename MetricType = metric::LMetric<2, true>, typename StatisticType = EmptyStatistic, typename
           MatType = arma::mat> StatisticType mlpack::tree::ExampleTree< MetricType, StatisticType, MatType >::stat
           [private]
```

This member is just here so the **ExampleTree** (p. 630) compiles without warnings.

It is not required to be a member in every type of tree.

Definition at line 227 of file example_tree.hpp.

The documentation for this class was generated from the following file:

- src/mlpack/core/tree/example_tree.hpp

22.150 mlpack::tree::FirstPointIsRoot Class Reference

This class is meant to be used as a choice for the policy class RootPointPolicy of the **CoverTree** (p. 603) class.

Static Public Member Functions

- static size_t **ChooseRoot** (const arma::mat &)
Return the point to be used as the root point of the cover tree.

22.150.1 Detailed Description

This class is meant to be used as a choice for the policy class RootPointPolicy of the **CoverTree** (p. 603) class.

This policy determines which point is used for the root node of the cover tree. This particular implementation simply chooses the first point in the dataset as the root. A more complex implementation might choose, for instance, the point with least maximum distance to other points (the closest to the "middle").

Definition at line 31 of file first_point_is_root.hpp.

22.150.2 Member Function Documentation

```
22.150.2.1 static size_t mlpack::tree::FirstPointIsRoot::ChooseRoot ( const arma::mat & ) [inline], [static]
```

Return the point to be used as the root point of the cover tree.

This just returns 0.

Definition at line 38 of file first_point_is_root.hpp.

The documentation for this class was generated from the following file:

- src/mlpack/core/tree/cover_tree/first_point_is_root.hpp

22.151 mlpack::tree::MeanSplit< BoundType, MatType > Class Template Reference

A binary space partitioning tree node is split into its left and right child.

Static Public Member Functions

- static bool **SplitNode** (const BoundType &bound, MatType &data, const size_t begin, const size_t count, size_t &splitDimension, size_t &splitCol)
Split the node according to the mean value in the dimension with maximum width.
- static bool **SplitNode** (const BoundType &bound, MatType &data, const size_t begin, const size_t count, size_t &splitDimension, size_t &splitCol, std::vector< size_t > &oldFromNew)
Split the node according to the mean value in the dimension with maximum width and return a list of changed indices.

Static Private Member Functions

- static size_t **PerformSplit** (MatType &data, const size_t begin, const size_t count, const size_t splitDimension, const double splitVal)
Reorder the dataset into two parts such that they lie on either side of splitCol.
- static size_t **PerformSplit** (MatType &data, const size_t begin, const size_t count, const size_t splitDimension, const double splitVal, std::vector< size_t > &oldFromNew)
Reorder the dataset into two parts such that they lie on either side of splitCol.

22.151.1 Detailed Description

```
template<typename BoundType, typename MatType = arma::mat>class mlpack::tree::MeanSplit< BoundType, MatType >
```

A binary space partitioning tree node is split into its left and right child.

The split is done in the dimension that has the maximum width. The points are divided into two parts based on the mean in this dimension.

Definition at line 30 of file mean_split.hpp.

22.151.2 Member Function Documentation

22.151.2.1 `template<typename BoundType , typename MatType = arma::mat> static size_t mlpack::tree::MeanSplit< BoundType, MatType >::PerformSplit (MatType & data, const size_t begin, const size_t count, const size_t splitDimension, const double splitVal) [static], [private]`

Reorder the dataset into two parts such that they lie on either side of splitCol.

Parameters

<i>data</i>	The dataset used by the binary space tree.
<i>begin</i>	Index of the starting point in the dataset that belongs to this node.
<i>count</i>	Number of points in this node.
<i>splitDimension</i>	The dimension to split the node on.
<i>splitVal</i>	The split in dimension splitDimension is based on this value.

22.151.2.2 `template<typename BoundType , typename MatType = arma::mat> static size_t mlpack::tree::MeanSplit< BoundType, MatType >::PerformSplit (MatType & data, const size_t begin, const size_t count, const size_t splitDimension, const double splitVal, std::vector< size_t > & oldFromNew) [static], [private]`

Reorder the dataset into two parts such that they lie on either side of splitCol.

Also returns a list of changed indices.

Parameters

<i>data</i>	The dataset used by the binary space tree.
<i>begin</i>	Index of the starting point in the dataset that belongs to this node.
<i>count</i>	Number of points in this node.
<i>splitDimension</i>	The dimension to split the node on.
<i>splitVal</i>	The split in dimension <i>splitDimension</i> is based on this value.
<i>oldFromNew</i>	Vector which will be filled with the old positions for each new point.

22.151.2.3 `template<typename BoundType , typename MatType = arma::mat> static bool mpack::tree::MeanSplit< BoundType, MatType >::SplitNode (const BoundType & bound, MatType & data, const size_t begin, const size_t count, size_t & splitDimension, size_t & splitCol) [static]`

Split the node according to the mean value in the dimension with maximum width.

Parameters

<i>bound</i>	The bound used for this node.
<i>data</i>	The dataset used by the binary space tree.
<i>begin</i>	Index of the starting point in the dataset that belongs to this node.
<i>count</i>	Number of points in this node.
<i>splitDimension</i>	This will be filled with the dimension the node is to be split on.
<i>splitCol</i>	The index at which the dataset is divided into two parts after the rearrangement.

22.151.2.4 `template<typename BoundType , typename MatType = arma::mat> static bool mpack::tree::MeanSplit< BoundType, MatType >::SplitNode (const BoundType & bound, MatType & data, const size_t begin, const size_t count, size_t & splitDimension, size_t & splitCol, std::vector< size_t > & oldFromNew) [static]`

Split the node according to the mean value in the dimension with maximum width and return a list of changed indices.

Parameters

<i>bound</i>	The bound used for this node.
<i>data</i>	The dataset used by the binary space tree.
<i>begin</i>	Index of the starting point in the dataset that belongs to this node.
<i>count</i>	Number of points in this node.
<i>splitDimension</i>	This will be filled with the dimension the node is to be split on.
<i>splitCol</i>	The index at which the dataset is divided into two parts after the rearrangement.
<i>oldFromNew</i>	Vector which will be filled with the old positions for each new point.

The documentation for this class was generated from the following file:

- `src/mpack/core/tree/binary_space_tree/mean_split.hpp`

22.152 mpack::tree::MRKDStatistic Class Reference

Statistic for multi-resolution kd-trees.

Collaboration diagram for mlpack::tree::MRKDStatistic:



Public Member Functions

- **MRKDStatistic** ()
Initialize an empty statistic.
- template<typename TreeType >
MRKDStatistic (const TreeType &)
This constructor is called when a node is finished initializing.
- size_t **Begin** () const
Get the index of the initial item in the dataset.
- size_t & **Begin** ()
Modify the index of the initial item in the dataset.
- const arma::colvec & **CenterOfMass** () const
Get the center of mass.
- arma::colvec & **CenterOfMass** ()
Modify the center of mass.
- size_t **Count** () const
Get the number of items in the dataset.
- size_t & **Count** ()
Modify the number of items in the dataset.
- size_t **DominatingCentroid** () const
Get the index of the dominating centroid.
- size_t & **DominatingCentroid** ()
Modify the index of the dominating centroid.
- std::string **ToString** () const
Returns a string representation of this object.
- const std::vector< size_t > & **Whitelist** () const
Access the whitelist.
- std::vector< size_t > & **Whitelist** ()
Modify the whitelist.

Private Attributes

- size_t **begin**
The initial item in the dataset, so we don't have to make a copy.
- arma::colvec **centerOfMass**
The center of mass for this dataset.
- size_t **count**
The number of items in the dataset.
- const arma::mat * **dataset**

The data points this object contains.

- `size_t dominatingCentroid`

The index of the dominating centroid of the associated hyperrectangle.

- `bool isWhitelistValid`

Whether or not the whitelist is valid.

- `const MRKDStatistic * leftStat`

The left child.

- `const MRKDStatistic * parentStat`

A link to the parent node; NULL if this is the root.

- `const MRKDStatistic * rightStat`

The right child.

- `double sumOfSquaredNorms`

The sum of the squared Euclidean norms for this dataset.

- `std::vector< size_t > whitelist`

The list of centroids that cannot own this hyperrectangle.

22.152.1 Detailed Description

Statistic for multi-resolution kd-trees.

Definition at line 25 of file `mrkd_statistic.hpp`.

22.152.2 Constructor & Destructor Documentation

22.152.2.1 `mlpack::tree::MRKDStatistic::MRKDStatistic ()`

Initialize an empty statistic.

22.152.2.2 `template<typename TreeType> mlpack::tree::MRKDStatistic::MRKDStatistic (const TreeType &)`

This constructor is called when a node is finished initializing.

Parameters

<i>node</i>	The node that has been finished.
-------------	----------------------------------

22.152.3 Member Function Documentation

22.152.3.1 `size_t mlpack::tree::MRKDStatistic::Begin () const [inline]`

Get the index of the initial item in the dataset.

Definition at line 45 of file `mrkd_statistic.hpp`.

References `begin`.

22.152.3.2 `size_t& mlpack::tree::MRKDStatistic::Begin ()` `[inline]`

Modify the index of the initial item in the dataset.

Definition at line 47 of file `mrkd_statistic.hpp`.

References `begin`.

22.152.3.3 `const arma::colvec& mlpack::tree::MRKDStatistic::CenterOfMass () const` `[inline]`

Get the center of mass.

Definition at line 55 of file `mrkd_statistic.hpp`.

References `centerOfMass`.

22.152.3.4 `arma::colvec& mlpack::tree::MRKDStatistic::CenterOfMass ()` `[inline]`

Modify the center of mass.

Definition at line 57 of file `mrkd_statistic.hpp`.

References `centerOfMass`.

22.152.3.5 `size_t mlpack::tree::MRKDStatistic::Count () const` `[inline]`

Get the number of items in the dataset.

Definition at line 50 of file `mrkd_statistic.hpp`.

References `count`.

22.152.3.6 `size_t& mlpack::tree::MRKDStatistic::Count ()` `[inline]`

Modify the number of items in the dataset.

Definition at line 52 of file `mrkd_statistic.hpp`.

References `count`.

22.152.3.7 `size_t mlpack::tree::MRKDStatistic::DominatingCentroid () const` `[inline]`

Get the index of the dominating centroid.

Definition at line 60 of file `mrkd_statistic.hpp`.

References `dominatingCentroid`.

22.152.3.8 `size_t& mlpack::tree::MRKDStatistic::DominatingCentroid ()` `[inline]`

Modify the index of the dominating centroid.

Definition at line 62 of file `mrkd_statistic.hpp`.

References `dominatingCentroid`.

22.152.3.9 `std::string mlpack::tree::MRKDStatistic::ToString () const`

Returns a string representation of this object.

22.152.3.10 `const std::vector<size_t>& mlpack::tree::MRKDStatistic::Whitelist () const` `[inline]`

Access the whitelist.

Definition at line 65 of file `mrkd_statistic.hpp`.

References `whitelist`.

22.152.3.11 `std::vector<size_t>& mlpack::tree::MRKDStatistic::Whitelist ()` `[inline]`

Modify the whitelist.

Definition at line 67 of file `mrkd_statistic.hpp`.

References `whitelist`.

22.152.4 Member Data Documentation

22.152.4.1 `size_t mlpack::tree::MRKDStatistic::begin` `[private]`

The initial item in the dataset, so we don't have to make a copy.

Definition at line 73 of file `mrkd_statistic.hpp`.

Referenced by `Begin()`.

22.152.4.2 `arma::colvec mlpack::tree::MRKDStatistic::centerOfMass` `[private]`

The center of mass for this dataset.

Definition at line 85 of file `mrkd_statistic.hpp`.

Referenced by `CenterOfMass()`.

22.152.4.3 `size_t mlpack::tree::MRKDStatistic::count` `[private]`

The number of items in the dataset.

Definition at line 75 of file `mrkd_statistic.hpp`.

Referenced by `Count()`.

22.152.4.4 `const arma::mat* mlpack::tree::MRKDStatistic::dataset` `[private]`

The data points this object contains.

Definition at line 71 of file `mrkd_statistic.hpp`.

22.152.4.5 `size_t mlpack::tree::MRKDStatistic::dominatingCentroid` [private]

The index of the dominating centroid of the associated hyperrectangle.

Definition at line 91 of file `mrkd_statistic.hpp`.

Referenced by `DominatingCentroid()`.

22.152.4.6 `bool mlpack::tree::MRKDStatistic::isWhitelistValid` [private]

Whether or not the whitelist is valid.

Definition at line 96 of file `mrkd_statistic.hpp`.

22.152.4.7 `const MRKDStatistic* mlpack::tree::MRKDStatistic::leftStat` [private]

The left child.

Definition at line 77 of file `mrkd_statistic.hpp`.

22.152.4.8 `const MRKDStatistic* mlpack::tree::MRKDStatistic::parentStat` [private]

A link to the parent node; NULL if this is the root.

Definition at line 81 of file `mrkd_statistic.hpp`.

22.152.4.9 `const MRKDStatistic* mlpack::tree::MRKDStatistic::rightStat` [private]

The right child.

Definition at line 79 of file `mrkd_statistic.hpp`.

22.152.4.10 `double mlpack::tree::MRKDStatistic::sumOfSquaredNorms` [private]

The sum of the squared Euclidean norms for this dataset.

Definition at line 87 of file `mrkd_statistic.hpp`.

22.152.4.11 `std::vector<size_t> mlpack::tree::MRKDStatistic::whitelist` [private]

The list of centroids that cannot own this hyperrectangle.

Definition at line 94 of file `mrkd_statistic.hpp`.

Referenced by `Whitelist()`.

The documentation for this class was generated from the following file:

- `src/mlpack/core/tree/mrkd_statistic.hpp`

22.153 mlpack::tree::TreeTraits< TreeType > Class Template Reference

The **TreeTraits** (p. 645) class provides compile-time information on the characteristics of a given tree type.

Static Public Attributes

- static const bool **FirstPointIsCentroid** = false
This is true if Point(0) is the centroid of the node.
- static const bool **HasOverlappingChildren** = true
This is true if the subspaces represented by the children of a node can overlap.
- static const bool **HasSelfChildren** = false
This is true if the points contained in the first child of a node (Child(0)) are also contained in that node.
- static const bool **RearrangesDataset** = false
This is true if the tree rearranges points in the dataset when it is built.

22.153.1 Detailed Description

`template<typename TreeType>class mlpack::tree::TreeTraits< TreeType >`

The **TreeTraits** (p. 645) class provides compile-time information on the characteristics of a given tree type.

These include traits such as whether or not a node knows the distance to its parent node, or whether or not the subspaces represented by children can overlap.

These traits can be used for static compile-time optimization:

```
// This if statement will be optimized out at compile time!
if (TreeTraits<TreeType>::HasOverlappingChildren == false)
{
    // Do a simpler computation because no children overlap.
}
else
{
    // Do the full, complex calculation.
}
```

The traits can also be used in conjunction with SFINAE to write specialized versions of functions:

```
template<typename TreeType>
void Compute(TreeType& node,
    boost::enable_if<
        TreeTraits<TreeType>::RearrangesDataset>::type*)
{
    // Computation where special dataset-rearranging tree constructor is
    // called.
}

template<typename TreeType>
void Compute(TreeType& node,
    boost::enable_if<
        !TreeTraits<TreeType>::RearrangesDataset>::type*)
{
    // Computation where normal tree constructor is called.
}
```

In those two examples, the `boost::enable_if<>` class takes a boolean template parameter which allows that function to be called when the boolean is true.

Each trait must be a static const value and not a function; only const values can be used as template parameters (with the exception of constexprs, which are a C++11 feature; but MLPACK is not using C++11). By default (the unspecialized implementation of **TreeTraits** (p. 645)), each parameter is set to make as few assumptions about the tree as possible; so, even if **TreeTraits** (p. 645) is not specialized for a particular tree type, tree-based algorithms should still work.

When you write your own tree, you must specialize the **TreeTraits** (p. 645) class to your tree type and set the corresponding values appropriately. See `mlpack/core/tree/binary_space_tree/traits.hpp` (p. 745) for an example.

Definition at line 80 of file `tree_traits.hpp`.

22.153.2 Member Data Documentation

22.153.2.1 `template<typename TreeType > const bool mlpack::tree::TreeTraits< TreeType >::FirstPointIsCentroid = false`
[static]

This is true if `Point(0)` is the centroid of the node.

Definition at line 92 of file `tree_traits.hpp`.

22.153.2.2 `template<typename TreeType > const bool mlpack::tree::TreeTraits< TreeType >::HasOverlappingChildren = true`
[static]

This is true if the subspaces represented by the children of a node can overlap.

Definition at line 87 of file `tree_traits.hpp`.

22.153.2.3 `template<typename TreeType > const bool mlpack::tree::TreeTraits< TreeType >::HasSelfChildren = false`
[static]

This is true if the points contained in the first child of a node (`Child(0)`) are also contained in that node.

Definition at line 98 of file `tree_traits.hpp`.

22.153.2.4 `template<typename TreeType > const bool mlpack::tree::TreeTraits< TreeType >::RearrangesDataset = false`
[static]

This is true if the tree rearranges points in the dataset when it is built.

Definition at line 103 of file `tree_traits.hpp`.

The documentation for this class was generated from the following file:

- `src/mlpack/core/tree/tree_traits.hpp`

22.154 `mlpack::tree::TreeTraits< BinarySpaceTree< BoundType, StatisticType, MatType > >` Class Template Reference

This is a specialization of the `TreeType` class to the **BinarySpaceTree** (p. 566) tree type.

Static Public Attributes

- static const bool **FirstPointIsCentroid** = false
There is no guarantee that the first point in a node is its centroid.
- static const bool **HasOverlappingChildren** = false
Each binary space tree node has two children which represent non-overlapping subsets of the space which the node represents.
- static const bool **HasSelfChildren** = false
Points are not contained at multiple levels of the binary space tree.
- static const bool **RearrangesDataset** = true
Points are rearranged during building of the tree.

22.154.1 Detailed Description

```
template<typename BoundType, typename StatisticType, typename MatType>class mlpack::tree::TreeTraits< BinarySpaceTree<
BoundType, StatisticType, MatType > >
```

This is a specialization of the `TreeType` class to the **BinarySpaceTree** (p. 566) tree type.

It defines characteristics of the binary space tree, and is used to help write tree-independent (but still optimized) tree-based algorithms. See `mlpack/core/tree/tree_traits.hpp` (p. 757) for more information.

Definition at line 31 of file `traits.hpp`.

22.154.2 Member Data Documentation

```
22.154.2.1 template<typename BoundType , typename StatisticType , typename MatType > const bool
mlpack::tree::TreeTraits< BinarySpaceTree< BoundType, StatisticType, MatType > >::FirstPointIsCentroid =
false [static]
```

There is no guarantee that the first point in a node is its centroid.

Definition at line 44 of file `traits.hpp`.

```
22.154.2.2 template<typename BoundType , typename StatisticType , typename MatType > const bool
mlpack::tree::TreeTraits< BinarySpaceTree< BoundType, StatisticType, MatType > >::HasOverlappingChildren
= false [static]
```

Each binary space tree node has two children which represent non-overlapping subsets of the space which the node represents.

Therefore, children are not overlapping.

Definition at line 39 of file `traits.hpp`.

```
22.154.2.3 template<typename BoundType , typename StatisticType , typename MatType > const bool
mlpack::tree::TreeTraits< BinarySpaceTree< BoundType, StatisticType, MatType > >::HasSelfChildren = false
[static]
```

Points are not contained at multiple levels of the binary space tree.

Definition at line 49 of file `traits.hpp`.

```
22.154.2.4 template<typename BoundType , typename StatisticType , typename MatType > const bool
mlpack::tree::TreeTraits< BinarySpaceTree< BoundType, StatisticType, MatType > >::RearrangesDataset =
true [static]
```

Points are rearranged during building of the tree.

Definition at line 54 of file `traits.hpp`.

The documentation for this class was generated from the following file:

- `src/mlpack/core/tree/binary_space_tree/traits.hpp`

22.155 `mlpack::tree::TreeTraits< CoverTree< MetricType, RootPointPolicy, StatisticType > >` Class Template Reference

The specialization of the **TreeTraits** (p. 645) class for the **CoverTree** (p. 603) tree type.

Static Public Attributes

- static const bool **FirstPointIsCentroid** = true
Each cover tree node contains only one point, and that point is its centroid.
- static const bool **HasOverlappingChildren** = true
The cover tree (or, this implementation of it) does not require that children represent non-overlapping subsets of the parent node.
- static const bool **HasSelfChildren** = true
Cover trees do have self-children.
- static const bool **RearrangesDataset** = false
Points are not rearranged when the tree is built.

22.155.1 Detailed Description

```
template<typename MetricType, typename RootPointPolicy, typename StatisticType>class mlpack::tree::TreeTraits< CoverTree<
MetricType, RootPointPolicy, StatisticType > >
```

The specialization of the **TreeTraits** (p. 645) class for the **CoverTree** (p. 603) tree type.

It defines characteristics of the cover tree, and is used to help write tree-independent (but still optimized) tree-based algorithms. See `mlpack/core/tree/tree_traits.hpp` (p. 757) for more information.

Definition at line 32 of file `traits.hpp`.

22.155.2 Member Data Documentation

22.155.2.1 `template<typename MetricType , typename RootPointPolicy , typename StatisticType > const bool mlpack::tree::TreeTraits< CoverTree< MetricType, RootPointPolicy, StatisticType > >::FirstPointIsCentroid = true [static]`

Each cover tree node contains only one point, and that point is its centroid.

Definition at line 45 of file `traits.hpp`.

22.155.2.2 `template<typename MetricType , typename RootPointPolicy , typename StatisticType > const bool mlpack::tree::TreeTraits< CoverTree< MetricType, RootPointPolicy, StatisticType > >::HasOverlappingChildren = true [static]`

The cover tree (or, this implementation of it) does not require that children represent non-overlapping subsets of the parent node.

Definition at line 39 of file `traits.hpp`.

```
22.155.2.3  template<typename MetricType , typename RootPointPolicy , typename StatisticType > const bool
            mlpack::tree::TreeTraits< CoverTree< MetricType, RootPointPolicy, StatisticType > >::HasSelfChildren = true
            [static]
```

Cover trees do have self-children.

Definition at line 50 of file traits.hpp.

```
22.155.2.4  template<typename MetricType , typename RootPointPolicy , typename StatisticType > const bool
            mlpack::tree::TreeTraits< CoverTree< MetricType, RootPointPolicy, StatisticType > >::RearrangesDataset =
            false [static]
```

Points are not rearranged when the tree is built.

Definition at line 55 of file traits.hpp.

The documentation for this class was generated from the following file:

- src/mlpack/core/tree/cover_tree/traits.hpp

22.156 mlpack::util::CLIDeleter Class Reference

Extremely simple class whose only job is to delete the existing **CLI** (p. 184) object at the end of execution.

Public Member Functions

- **CLIDeleter** ()
- **~CLIDeleter** ()

22.156.1 Detailed Description

Extremely simple class whose only job is to delete the existing **CLI** (p. 184) object at the end of execution.

This is meant to allow the user to avoid typing '**CLI::Destroy()** (p. 192)' at the end of their program. The file also defines a static **CLIDeleter** (p. 650) class, which will be initialized at the beginning of the program and deleted at the end. The destructor destroys the **CLI** (p. 184) singleton.

Definition at line 27 of file cli_deleter.hpp.

22.156.2 Constructor & Destructor Documentation

```
22.156.2.1  mlpack::util::CLIDeleter::CLIDeleter ( )
```

```
22.156.2.2  mlpack::util::CLIDeleter::~~CLIDeleter ( )
```

The documentation for this class was generated from the following file:

- src/mlpack/core/util/cli_deleter.hpp

22.157 mpack::util::NullOutputStream Class Reference

Used for **Log::Debug** (p. 359) when not compiled with debugging symbols.

Public Member Functions

- **NullOutputStream** ()
Does nothing.
- **NullOutputStream** (const **NullOutputStream** &)
Does nothing.
- **NullOutputStream** & **operator**<< (bool)
Does nothing.
- **NullOutputStream** & **operator**<< (short)
Does nothing.
- **NullOutputStream** & **operator**<< (unsigned short)
Does nothing.
- **NullOutputStream** & **operator**<< (int)
Does nothing.
- **NullOutputStream** & **operator**<< (unsigned int)
Does nothing.
- **NullOutputStream** & **operator**<< (long)
Does nothing.
- **NullOutputStream** & **operator**<< (unsigned long)
Does nothing.
- **NullOutputStream** & **operator**<< (float)
Does nothing.
- **NullOutputStream** & **operator**<< (double)
Does nothing.
- **NullOutputStream** & **operator**<< (long double)
Does nothing.
- **NullOutputStream** & **operator**<< (void *)
Does nothing.
- **NullOutputStream** & **operator**<< (const char *)
Does nothing.
- **NullOutputStream** & **operator**<< (std::string &)
Does nothing.
- **NullOutputStream** & **operator**<< (std::streambuf *)
Does nothing.
- **NullOutputStream** & **operator**<< (std::ostream &(*)(std::ostream &))
Does nothing.
- **NullOutputStream** & **operator**<< (std::ios &(*)(std::ios &))
Does nothing.
- **NullOutputStream** & **operator**<< (std::ios_base &(*)(std::ios_base &))
Does nothing.
- template<typename T >
NullOutputStream & **operator**<< (const T &)
Does nothing.

22.157.1 Detailed Description

Used for **Log::Debug** (p. 359) when not compiled with debugging symbols.

This class does nothing and should be optimized out entirely by the compiler.

Definition at line 29 of file nullostream.hpp.

22.157.2 Constructor & Destructor Documentation

22.157.2.1 `mlpack::util::NullOutputStream::NullOutputStream ()` `[inline]`

Does nothing.

Definition at line 35 of file nullostream.hpp.

22.157.2.2 `mlpack::util::NullOutputStream::NullOutputStream (const NullOutputStream &)` `[inline]`

Does nothing.

Definition at line 40 of file nullostream.hpp.

22.157.3 Member Function Documentation

22.157.3.1 `NullOutputStream& mlpack::util::NullOutputStream::operator<< (bool)` `[inline]`

Does nothing.

Definition at line 43 of file nullostream.hpp.

22.157.3.2 `NullOutputStream& mlpack::util::NullOutputStream::operator<< (short)` `[inline]`

Does nothing.

Definition at line 45 of file nullostream.hpp.

22.157.3.3 `NullOutputStream& mlpack::util::NullOutputStream::operator<< (unsigned short)` `[inline]`

Does nothing.

Definition at line 47 of file nullostream.hpp.

22.157.3.4 `NullOutputStream& mlpack::util::NullOutputStream::operator<< (int)` `[inline]`

Does nothing.

Definition at line 49 of file nullostream.hpp.

22.157.3.5 `NullOutputStream& mlpack::util::NullOutputStream::operator<< (unsigned int)` `[inline]`

Does nothing.

Definition at line 51 of file nullostream.hpp.

22.157.3.6 `NullOutputStream& mpack::util::NullOutputStream::operator<< (long) [inline]`

Does nothing.

Definition at line 53 of file nullostream.hpp.

22.157.3.7 `NullOutputStream& mpack::util::NullOutputStream::operator<< (unsigned long) [inline]`

Does nothing.

Definition at line 55 of file nullostream.hpp.

22.157.3.8 `NullOutputStream& mpack::util::NullOutputStream::operator<< (float) [inline]`

Does nothing.

Definition at line 57 of file nullostream.hpp.

22.157.3.9 `NullOutputStream& mpack::util::NullOutputStream::operator<< (double) [inline]`

Does nothing.

Definition at line 59 of file nullostream.hpp.

22.157.3.10 `NullOutputStream& mpack::util::NullOutputStream::operator<< (long double) [inline]`

Does nothing.

Definition at line 61 of file nullostream.hpp.

22.157.3.11 `NullOutputStream& mpack::util::NullOutputStream::operator<< (void *) [inline]`

Does nothing.

Definition at line 63 of file nullostream.hpp.

22.157.3.12 `NullOutputStream& mpack::util::NullOutputStream::operator<< (const char *) [inline]`

Does nothing.

Definition at line 65 of file nullostream.hpp.

22.157.3.13 `NullOutputStream& mpack::util::NullOutputStream::operator<< (std::string &) [inline]`

Does nothing.

Definition at line 67 of file nullostream.hpp.

22.157.3.14 `NullOutputStream& mpack::util::NullOutputStream::operator<< (std::streambuf *) [inline]`

Does nothing.

Definition at line 69 of file nullostream.hpp.

22.157.3.15 `NullOutputStream& mpack::util::NullOutputStream::operator<< (std::ostream &*)(std::ostream &)` `[inline]`

Does nothing.

Definition at line 71 of file `nullostream.hpp`.

22.157.3.16 `NullOutputStream& mpack::util::NullOutputStream::operator<< (std::ios &*)(std::ios &)` `[inline]`

Does nothing.

Definition at line 73 of file `nullostream.hpp`.

22.157.3.17 `NullOutputStream& mpack::util::NullOutputStream::operator<< (std::ios_base &*)(std::ios_base &)` `[inline]`

Does nothing.

Definition at line 75 of file `nullostream.hpp`.

22.157.3.18 `template<typename T > NullOutputStream& mpack::util::NullOutputStream::operator<< (const T &)` `[inline]`

Does nothing.

Definition at line 80 of file `nullostream.hpp`.

The documentation for this class was generated from the following file:

- `src/mpack/core/util/nullostream.hpp`

22.158 `mpack::util::Option< N >` Class Template Reference

A static object whose constructor registers a parameter with the **CLI** (p. 184) class.

Public Member Functions

- **Option** (bool ignoreTemplate, N defaultValue, const std::string &identifier, const std::string &description, const std::string &parent=std::string(""), bool required=false)
*Construct an **Option** (p. 654) object.*
- **Option** (const std::string &identifier, const std::string &description, const std::string &parent=std::string(""))
*Constructs an **Option** (p. 654) object.*

22.158.1 Detailed Description

`template<typename N> class mpack::util::Option< N >`

A static object whose constructor registers a parameter with the **CLI** (p. 184) class.

This should not be used outside of **CLI** (p. 184) itself, and you should use the **PARAM_FLAG()** (p. 762), **PARAM_DO↵UBLE()** (p. 761), **PARAM_INT()** (p. 763), **PARAM_STRING()** (p. 764), or other similar macros to declare these objects instead of declaring them directly.

See also

core/io/cli.hpp, **mlpack::CLI** (p. 184)

Definition at line 34 of file option.hpp.

22.158.2 Constructor & Destructor Documentation

22.158.2.1 `template<typename N> mlpack::util::Option<N>::Option (bool ignoreTemplate, N defaultValue, const std::string & identifier, const std::string & description, const std::string & parent = std::string(""), bool required = false)`

Construct an **Option** (p. 654) object.

When constructed, it will register itself with **CLI** (p. 184).

Parameters

<i>ignoreTemplate</i>	Whether or not the template type matters for this option. Essentially differs options with no value (flags) from those that do, and thus require a type.
<i>defaultValue</i>	Default value this parameter will be initialized to.
<i>identifier</i>	The name of the option (no dashes in front; for <code>--help</code> , we would pass "help").
<i>description</i>	A short string describing the option.
<i>parent</i>	Full pathname of the parent module that "owns" this option. The default is the root node (an empty string).
<i>required</i>	Whether or not the option is required at runtime.

22.158.2.2 `template<typename N> mlpack::util::Option<N>::Option (const std::string & identifier, const std::string & description, const std::string & parent = std::string(""))`

Constructs an **Option** (p. 654) object.

When constructed, it will register a flag with **CLI** (p. 184).

Parameters

<i>identifier</i>	The name of the option (no dashes in front); for <code>--help</code> we would pass "help".
<i>description</i>	A short string describing the option.
<i>parent</i>	Full pathname of the parent module that "owns" this option. The default is the root node (an empty string).

The documentation for this class was generated from the following file:

- src/mlpack/core/util/**option.hpp**

22.159 mlpack::util::PrefixedOutputStream Class Reference

Allows us to output to an ostream with a prefix at the beginning of each line, in the same way we would output to cout or cerr.

Public Member Functions

- **PrefixedOutputStream** (std::ostream &destination, const char *prefix, bool ignoreInput=false, bool fatal=false)

Set up the **PrefixOutputStream** (p. 655).

- **PrefixOutputStream & operator<<** (bool val)
Write a bool to the stream.
- **PrefixOutputStream & operator<<** (short val)
Write a short to the stream.
- **PrefixOutputStream & operator<<** (unsigned short val)
Write an unsigned short to the stream.
- **PrefixOutputStream & operator<<** (int val)
Write an int to the stream.
- **PrefixOutputStream & operator<<** (unsigned int val)
Write an unsigned int to the stream.
- **PrefixOutputStream & operator<<** (long val)
Write a long to the stream.
- **PrefixOutputStream & operator<<** (unsigned long val)
Write an unsigned long to the stream.
- **PrefixOutputStream & operator<<** (float val)
Write a float to the stream.
- **PrefixOutputStream & operator<<** (double val)
Write a double to the stream.
- **PrefixOutputStream & operator<<** (long double val)
Write a long double to the stream.
- **PrefixOutputStream & operator<<** (void *val)
Write a void pointer to the stream.
- **PrefixOutputStream & operator<<** (const char *str)
Write a character array to the stream.
- **PrefixOutputStream & operator<<** (std::string &str)
Write a string to the stream.
- **PrefixOutputStream & operator<<** (std::streambuf *sb)
Write a streambuf to the stream.
- **PrefixOutputStream & operator<<** (std::ostream &(*pf)(std::ostream &))
Write an ostream manipulator function to the stream.
- **PrefixOutputStream & operator<<** (std::ios &(*pf)(std::ios &))
Write an ios manipulator function to the stream.
- **PrefixOutputStream & operator<<** (std::ios_base &(*pf)(std::ios_base &))
Write an ios_base manipulator function to the stream.
- template<typename T >
PrefixOutputStream & operator<< (const T &s)
Write anything else to the stream.

Public Attributes

- std::ostream & **destination**
The output stream that all data is to be sent too; example: std::cout.
- bool **ignoreInput**
Discards input, prints nothing if true.

Private Member Functions

- template<typename T >
void **BaseLogic** (const T &val)
Conducts the base logic required in all the operator << overloads.
- template<typename T >
void **CallBaseLogic** (const T &s, typename boost::disable_if< boost::is_class< T > >::type *=0)
This handles forwarding all primitive types transparently.
- template<typename T >
void **CallBaseLogic** (const T &s, typename boost::enable_if< boost::is_class< T > >::type *=0, typename boost::disable_if< HasToString< T, std::string(T::*)() const > >::type *=0)
Forward all objects that do not implement a ToString() method.
- template<typename T >
void **CallBaseLogic** (const T &s, typename boost::enable_if< boost::is_class< T > >::type *=0, typename boost::enable_if< HasToString< T, std::string(T::*)() const > >::type *=0)
Call ToString() on all objects that implement ToString() before forwarding.
- void **PrefixIfNeeded** ()
Output the prefix, but only if we need to and if we are allowed to.

Private Attributes

- bool **carriageReturned**
If true, the previous call to operator<< encountered a CR, and a prefix will be necessary.
- bool **fatal**
If true, the application will terminate with an error code when a CR is encountered.
- std::string **prefix**
Contains the prefix we must prepend to each line.

22.159.1 Detailed Description

Allows us to output to an ostream with a prefix at the beginning of each line, in the same way we would output to cout or cerr.

The prefix is specified in the constructor (as well as the destination ostream). A newline must be passed to the stream, and then the prefix will be prepended to the next line. For example,

```
PrefixedOutputStream ostr(std::cout, "[TEST] ");
ostr << "Hello world I like " << 7.5;
ostr << "...Continue" << std::endl;
ostr << "After the CR\n" << std::endl;
```

would give, on std::cout,

```
[TEST] Hello world I like 7.5...Continue
[TEST] After the CR
[TEST]
```

These objects are used for the **mpack::Log** (p. 357) levels (DEBUG, INFO, WARN, and FATAL).

Definition at line 58 of file prefixedostream.hpp.

22.159.2 Constructor & Destructor Documentation

22.159.2.1 `mlpack::util::PrefixedOutputStream::PrefixedOutputStream (std::ostream & destination, const char * prefix, bool ignoreInput = false, bool fatal = false) [inline]`

Set up the **PrefixedOutputStream** (p. 655).

Parameters

<i>destination</i>	ostream which receives output from this object.
<i>prefix</i>	The prefix to prepend to each line.

Definition at line 67 of file prefixedostream.hpp.

22.159.3 Member Function Documentation

22.159.3.1 `template<typename T> void mpack::util::PrefixedOutputStream::BaseLogic (const T & val) [private]`

Conducts the base logic required in all the operator << overloads.

Mostly just a good idea to reduce copy-pasta.

Template Parameters

<i>T</i>	The type of the data to output.
----------	---------------------------------

Parameters

<i>val</i>	The The data to be output.
------------	----------------------------

22.159.3.2 `template<typename T> void mpack::util::PrefixedOutputStream::CallBaseLogic (const T & s, typename boost::disable_if< boost::is_class< T>>::type * = 0) [private]`

This handles forwarding all primitive types transparently.

22.159.3.3 `template<typename T> void mpack::util::PrefixedOutputStream::CallBaseLogic (const T & s, typename boost::enable_if< boost::is_class< T>>::type * = 0, typename boost::disable_if< HasToString< T, std::string(T::*)() const>>::type * = 0) [private]`

Forward all objects that do not implement a ToString() method.

22.159.3.4 `template<typename T> void mpack::util::PrefixedOutputStream::CallBaseLogic (const T & s, typename boost::enable_if< boost::is_class< T>>::type * = 0, typename boost::enable_if< HasToString< T, std::string(T::*)() const>>::type * = 0) [private]`

Call ToString() on all objects that implement ToString() before forwarding.

22.159.3.5 `PrefixedOutputStream& mpack::util::PrefixedOutputStream::operator<< (bool val)`

Write a bool to the stream.

22.159.3.6 `PrefixedOutputStream& mpack::util::PrefixedOutputStream::operator<< (short val)`

Write a short to the stream.

22.159.3.7 `PrefixedOutputStream& mpack::util::PrefixedOutputStream::operator<< (unsigned short val)`

Write an unsigned short to the stream.

22.159.3.8 `PrefixOutputStream& mpack::util::PrefixOutputStream::operator<< (int val)`

Write an int to the stream.

22.159.3.9 `PrefixOutputStream& mpack::util::PrefixOutputStream::operator<< (unsigned int val)`

Write an unsigned int to the stream.

22.159.3.10 `PrefixOutputStream& mpack::util::PrefixOutputStream::operator<< (long val)`

Write a long to the stream.

22.159.3.11 `PrefixOutputStream& mpack::util::PrefixOutputStream::operator<< (unsigned long val)`

Write an unsigned long to the stream.

22.159.3.12 `PrefixOutputStream& mpack::util::PrefixOutputStream::operator<< (float val)`

Write a float to the stream.

22.159.3.13 `PrefixOutputStream& mpack::util::PrefixOutputStream::operator<< (double val)`

Write a double to the stream.

22.159.3.14 `PrefixOutputStream& mpack::util::PrefixOutputStream::operator<< (long double val)`

Write a long double to the stream.

22.159.3.15 `PrefixOutputStream& mpack::util::PrefixOutputStream::operator<< (void * val)`

Write a void pointer to the stream.

22.159.3.16 `PrefixOutputStream& mpack::util::PrefixOutputStream::operator<< (const char * str)`

Write a character array to the stream.

22.159.3.17 `PrefixOutputStream& mpack::util::PrefixOutputStream::operator<< (std::string & str)`

Write a string to the stream.

22.159.3.18 `PrefixOutputStream& mpack::util::PrefixOutputStream::operator<< (std::streambuf * sb)`

Write a streambuf to the stream.

22.159.3.19 `PrefixedOutputStream& mpack::util::PrefixedOutputStream::operator<< (std::ostream &(*)(std::ostream &) pf)`

Write an ostream manipulator function to the stream.

22.159.3.20 `PrefixedOutputStream& mpack::util::PrefixedOutputStream::operator<< (std::ios &(*)(std::ios &) pf)`

Write an ios manipulator function to the stream.

22.159.3.21 `PrefixedOutputStream& mpack::util::PrefixedOutputStream::operator<< (std::ios_base &(*)(std::ios_base &) pf)`

Write an ios_base manipulator function to the stream.

22.159.3.22 `template<typename T> PrefixedOutputStream& mpack::util::PrefixedOutputStream::operator<< (const T & s)`

Write anything else to the stream.

22.159.3.23 `void mpack::util::PrefixedOutputStream::PrefixIfNeeded () [inline],[private]`

Output the prefix, but only if we need to and if we are allowed to.

22.159.4 Member Data Documentation

22.159.4.1 `bool mpack::util::PrefixedOutputStream::carriageReturned [private]`

If true, the previous call to operator<< encountered a CR, and a prefix will be necessary.

Definition at line 175 of file prefixedostream.hpp.

22.159.4.2 `std::ostream& mpack::util::PrefixedOutputStream::destination`

The output stream that all data is to be sent too; example: std::cout.

Definition at line 120 of file prefixedostream.hpp.

22.159.4.3 `bool mpack::util::PrefixedOutputStream::fatal [private]`

If true, the application will terminate with an error code when a CR is encountered.

Definition at line 179 of file prefixedostream.hpp.

22.159.4.4 `bool mpack::util::PrefixedOutputStream::ignoreInput`

Discards input, prints nothing if true.

Definition at line 123 of file prefixedostream.hpp.

22.159.4.5 `std::string mlpack::util::PrefixedOutputStream::prefix` [private]

Contains the prefix we must prepend to each line.

Definition at line 171 of file `prefixedostream.hpp`.

The documentation for this class was generated from the following file:

- `src/mlpack/core/util/prefixedostream.hpp`

22.160 mlpack::util::ProgramDoc Class Reference

A static object whose constructor registers program documentation with the **CLI** (p. 184) class.

Public Member Functions

- **ProgramDoc** (const std::string &**programName**, const std::string &**documentation**)
*Construct a **ProgramDoc** (p. 662) object.*

Public Attributes

- std::string **documentation**
Documentation for what the program does.
- std::string **programName**
The name of the program.

22.160.1 Detailed Description

A static object whose constructor registers program documentation with the **CLI** (p. 184) class.

This should not be used outside of **CLI** (p. 184) itself, and you should use the **PROGRAM_INFO()** (p. 766) macro to declare these objects. Only one **ProgramDoc** (p. 662) object should ever exist.

See also

`core/io/cli.hpp`, **mlpack::CLI** (p. 184)

Definition at line 82 of file `option.hpp`.

22.160.2 Constructor & Destructor Documentation

22.160.2.1 `mlpack::util::ProgramDoc::ProgramDoc (const std::string & programName, const std::string & documentation)`

Construct a **ProgramDoc** (p. 662) object.

When constructed, it will register itself with **CLI** (p. 184).

Parameters

<i>programName</i>	Short string representing the name of the program.
<i>documentation</i>	Long string containing documentation on how to use the program and what it is. No newline characters are necessary; this is taken care of by CLI (p. 184) later.

22.160.3 Member Data Documentation

22.160.3.1 std::string mlpack::util::ProgramDoc::documentation

Documentation for what the program does.

Definition at line 100 of file option.hpp.

22.160.3.2 std::string mlpack::util::ProgramDoc::programName

The name of the program.

Definition at line 98 of file option.hpp.

The documentation for this class was generated from the following file:

- src/mlpack/core/util/**option.hpp**

22.161 mlpack::util::SaveRestoreUtility Class Reference

Public Member Functions

- **SaveRestoreUtility** ()
- **~SaveRestoreUtility** ()
- template<typename T >
T & **LoadParameter** (T &t, const std::string &name)
LoadParameter loads a parameter from the parameters map.
- template<typename T >
std::vector< T > & **LoadParameter** (std::vector< T > &v, const std::string &name)
LoadParameter loads a parameter from the parameters map.
- char **LoadParameter** (char c, const std::string &name)
LoadParameter loads a character from the parameters map.
- std::string **LoadParameter** (std::string &str, const std::string &name)
LoadParameter loads a string from the parameters map.
- arma::mat & **LoadParameter** (arma::mat &matrix, const std::string &name)
LoadParameter loads an arma::mat from the parameters map.
- template<>
arma::vec & **LoadParameter** (arma::vec &t, const std::string &name)
Specialization for arma::vec.
- bool **ReadFile** (const std::string &filename)
ReadFile reads an XML tree from a file.
- template<typename T >
void **SaveParameter** (const T &t, const std::string &name)
SaveParameter saves a parameter to the parameters map.

- `template<typename T >`
`void SaveParameter (const std::vector< T > &v, const std::string &name)`
SaveParameter saves a parameter to the parameters map.
- `void SaveParameter (const char c, const std::string &name)`
SaveParameter saves a character to the parameters map.
- `void SaveParameter (const arma::mat &mat, const std::string &name)`
SaveParameter saves an arma::mat to the parameters map.
- `template<>`
`void SaveParameter (const arma::vec &t, const std::string &name)`
Specialization for arma::vec.
- `bool WriteFile (const std::string &filename)`
WriteFile writes the XML tree to a file.

Private Member Functions

- `void RecurseOnNodes (xmlNode *n)`
RecurseOnNodes performs a depth first search of the XML tree.

Private Attributes

- `std::map< std::string, std::string > parameters`
parameters contains a list of names and parameters in string form.

22.161.1 Detailed Description

Definition at line 34 of file `save_restore_utility.hpp`.

22.161.2 Constructor & Destructor Documentation

22.161.2.1 `mlpack::util::SaveRestoreUtility::SaveRestoreUtility ()` `[inline]`

Definition at line 48 of file `save_restore_utility.hpp`.

22.161.2.2 `mlpack::util::SaveRestoreUtility::~SaveRestoreUtility ()` `[inline]`

Definition at line 49 of file `save_restore_utility.hpp`.

22.161.3 Member Function Documentation

22.161.3.1 `template<typename T > T& mlpack::util::SaveRestoreUtility::LoadParameter (T &t, const std::string &name)`

LoadParameter loads a parameter from the parameters map.

22.161.3.2 `template<typename T > std::vector<T>& mlpack::util::SaveRestoreUtility::LoadParameter (std::vector< T > &v, const std::string &name)`

LoadParameter loads a parameter from the parameters map.

22.161.3.3 `char mlpack::util::SaveRestoreUtility::LoadParameter (char c, const std::string & name)`

LoadParameter loads a character from the parameters map.

22.161.3.4 `std::string mlpack::util::SaveRestoreUtility::LoadParameter (std::string & str, const std::string & name)`

LoadParameter loads a string from the parameters map.

22.161.3.5 `arma::mat& mlpack::util::SaveRestoreUtility::LoadParameter (arma::mat & matrix, const std::string & name)`

LoadParameter loads an arma::mat from the parameters map.

22.161.3.6 `template<> arma::vec& mlpack::util::SaveRestoreUtility::LoadParameter (arma::vec & t, const std::string & name)`

Specialization for arma::vec.

22.161.3.7 `bool mlpack::util::SaveRestoreUtility::ReadFile (const std::string & filename)`

ReadFile reads an XML tree from a file.

22.161.3.8 `void mlpack::util::SaveRestoreUtility::RecurseOnNodes (xmlNode * n) [private]`

RecurseOnNodes performs a depth first search of the XML tree.

22.161.3.9 `template<typename T> void mlpack::util::SaveRestoreUtility::SaveParameter (const T & t, const std::string & name)`

SaveParameter saves a parameter to the parameters map.

22.161.3.10 `template<typename T> void mlpack::util::SaveRestoreUtility::SaveParameter (const std::vector< T> & v, const std::string & name)`

SaveParameter saves a parameter to the parameters map.

22.161.3.11 `void mlpack::util::SaveRestoreUtility::SaveParameter (const char c, const std::string & name)`

SaveParameter saves a character to the parameters map.

22.161.3.12 `void mlpack::util::SaveRestoreUtility::SaveParameter (const arma::mat & mat, const std::string & name)`

SaveParameter saves an arma::mat to the parameters map.

22.161.3.13 `template<> void mlpack::util::SaveRestoreUtility::SaveParameter (const arma::vec & t, const std::string & name)`

Specialization for arma::vec.

22.161.3.14 `bool mlpack::util::SaveRestoreUtility::WriteFile (const std::string & filename)`

WriteFile writes the XML tree to a file.

22.161.4 Member Data Documentation

22.161.4.1 `std::map<std::string, std::string> mlpack::util::SaveRestoreUtility::parameters [private]`

parameters contains a list of names and parameters in string form.

Definition at line 40 of file `save_restore_utility.hpp`.

The documentation for this class was generated from the following file:

- `src/mlpack/core/util/save_restore_utility.hpp`

22.162 `RASearch< SortPolicy, MetricType, TreeType >` Class Template Reference

The **RASearch** (p. 666) class: This class provides a generic manner to perform rank-approximate search via random-sampling.

Public Member Functions

- **RASearch** (const typename TreeType::Mat &**referenceSet**, const typename TreeType::Mat &**querySet**, const bool **naive**=false, const bool **singleMode**=false, const MetricType **metric**=MetricType())
*Initialize the **RASearch** (p. 666) object, passing both a query and reference dataset.*
- **RASearch** (const typename TreeType::Mat &**referenceSet**, const bool **naive**=false, const bool **singleMode**=false, const MetricType **metric**=MetricType())
*Initialize the **RASearch** (p. 666) object, passing only one dataset, which is used as both the query and the reference dataset.*
- **RASearch** (TreeType ***referenceTree**, TreeType ***queryTree**, const typename TreeType::Mat &**referenceSet**, const typename TreeType::Mat &**querySet**, const bool **singleMode**=false, const MetricType **metric**=MetricType())
*Initialize the **RASearch** (p. 666) object with the given datasets and pre-constructed trees.*
- **RASearch** (TreeType ***referenceTree**, const typename TreeType::Mat &**referenceSet**, const bool **singleMode**=false, const MetricType **metric**=MetricType())
*Initialize the **RASearch** (p. 666) object with the given reference dataset and pre-constructed tree.*
- **~RASearch** ()
*Delete the **RASearch** (p. 666) object.*
- void **ResetQueryTree** ()
*This function recursively resets the **RAQueryStat** of the **queryTree** to set 'bound' to **WorstDistance** and the 'numSamplesMade' to 0.*
- void **Search** (const size_t k, arma::Mat< size_t > &resultingNeighbors, arma::mat &distances, const double tau=5, const double **alpha**=0.95, const bool sampleAtLeaves=false, const bool firstLeafExact=false, const size_t singleSampleLimit=20)
Compute the rank approximate nearest neighbors and store the output in the given matrices.
- std::string **ToString** () const

Private Member Functions

- void **ResetRAQueryStat** (TreeType *treeNode)

Private Attributes

- bool **hasQuerySet**
Indicates if a separate query set was passed.
- MetricType **metric**
Instantiation of kernel.
- bool **naive**
Indicates if naive random sampling on the set is being used.
- size_t **numberOfPrunes**
Total number of pruned nodes during the neighbor search.
- std::vector< size_t > **oldFromNewQueries**
Permutations of query points during tree building.
- std::vector< size_t > **oldFromNewReferences**
Permutations of reference points during tree building.
- arma::mat **queryCopy**
Copy of query dataset (if we need it, because tree building modifies it).
- const arma::mat & **querySet**
Query dataset (may not be given).
- TreeType * **queryTree**
Pointer to the root of the query tree (might not exist).
- arma::mat **referenceCopy**
Copy of reference dataset (if we need it, because tree building modifies it).
- const arma::mat & **referenceSet**
Reference dataset.
- TreeType * **referenceTree**
Pointer to the root of the reference tree.
- bool **singleMode**
Indicates if single-tree search is being used (opposed to dual-tree).
- bool **treeOwner**
If true, this object created the trees and is responsible for them.

22.162.1 Detailed Description

```
template<typename SortPolicy = NearestNeighborSort, typename MetricType = mlpack::metric::SquaredEuclideanDistance, typename
TreeType = tree::BinarySpaceTree<bound::HRectBound<2, false>, RAQueryStat<SortPolicy> >>class RASearch< SortPolicy,
MetricType, TreeType >
```

The **RASearch** (p. 666) class: This class provides a generic manner to perform rank-approximate search via random-sampling.

If the 'naive' option is chosen, this rank-approximate search will be done by randomly sampled from the whole set. If the 'naive' option is not chosen, the sampling is done in a stratified manner in the tree as mentioned in the algorithms in Figure 2 of the following paper:

```
{ram2009rank, title={{Rank-Approximate Nearest Neighbor Search: Retaining Meaning and Speed in High Dimen-
sions}}, author={{Ram, P. and Lee, D. and Ouyang, H. and Gray, A. G.}}, booktitle={{Advances of Neural Information
Processing Systems}}, year={2009} }
```

RASearch (p. 666) is currently known to not work with ball trees (#356).

Template Parameters

<i>SortPolicy</i>	The sort policy for distances; see NearestNeighborSort.
<i>MetricType</i>	The metric to use for computation.
<i>TreeType</i>	The tree type to use.

Definition at line 63 of file `ra_search.hpp`.

22.162.2 Constructor & Destructor Documentation

22.162.2.1 `template<typename SortPolicy = NearestNeighborSort, typename MetricType = mlpack::metric::SquaredEuclideanDistance, typename TreeType = tree::BinarySpaceTree<bound::HRectBound<2, false>, RAQueryStat<SortPolicy>>> RASearch<SortPolicy, MetricType, TreeType>::RASearch (const typename TreeType::Mat & referenceSet, const typename TreeType::Mat & querySet, const bool naive = false, const bool singleMode = false, const MetricType metric = MetricType())`

Initialize the **RASearch** (p. 666) object, passing both a query and reference dataset.

Optionally, perform the computation in naive mode or single-tree mode, and set the leaf size used for tree-building. An initialized distance metric can be given, for cases where the metric has internal data (i.e. the `distance::MahalanobisDistance` class).

This method will copy the matrices to internal copies, which are rearranged during tree-building. You can avoid this extra copy by pre-constructing the trees and passing them using a different constructor.

Parameters

<i>referenceSet</i>	Set of reference points.
<i>querySet</i>	Set of query points.
<i>naive</i>	If true, the rank-approximate search will be performed by directly sampling the whole set instead of using the stratified sampling on the tree.
<i>singleMode</i>	If true, single-tree search will be used (as opposed to dual-tree search).
<i>leafSize</i>	Leaf size for tree construction (ignored if tree is given).
<i>metric</i>	An optional instance of the <code>MetricType</code> class.

22.162.2.2 `template<typename SortPolicy = NearestNeighborSort, typename MetricType = mlpack::metric::SquaredEuclideanDistance, typename TreeType = tree::BinarySpaceTree<bound::HRectBound<2, false>, RAQueryStat<SortPolicy>>> RASearch<SortPolicy, MetricType, TreeType>::RASearch (const typename TreeType::Mat & referenceSet, const bool naive = false, const bool singleMode = false, const MetricType metric = MetricType())`

Initialize the **RASearch** (p. 666) object, passing only one dataset, which is used as both the query and the reference dataset.

Optionally, perform the computation in naive mode or single-tree mode, and set the leaf size used for tree-building. An initialized distance metric can be given, for cases where the metric has internal data (i.e. the `distance::MahalanobisDistance` class).

If naive mode is being used and a pre-built tree is given, it may not work: naive mode operates by building a one-node tree (the root node holds all the points). If that condition is not satisfied with the pre-built tree, then naive mode will not work.

Parameters

<i>referenceSet</i>	Set of reference points.
<i>naive</i>	If true, the rank-approximate search will be performed by directly sampling the whole set instead of using the stratified sampling on the tree.
<i>singleMode</i>	If true, single-tree search will be used (as opposed to dual-tree search).
<i>leafSize</i>	Leaf size for tree construction (ignored if tree is given).
<i>metric</i>	An optional instance of the MetricType class.

```
22.162.2.3  template<typename SortPolicy = NearestNeighborSort, typename MetricType = mpack::metric::SquaredEuclidean<
Distance, typename TreeType = tree::BinarySpaceTree<bound::HRectBound<2, false>, RAQueryStat<SortPolicy>
>> RASearch< SortPolicy, MetricType, TreeType >::RASearch ( TreeType * referenceTree, TreeType * queryTree,
const typename TreeType::Mat & referenceSet, const typename TreeType::Mat & querySet, const bool singleMode =
false, const MetricType metric = MetricType() )
```

Initialize the **RASearch** (p. 666) object with the given datasets and pre-constructed trees.

It is assumed that the points in *referenceSet* and *querySet* correspond to the points in *referenceTree* and *queryTree*, respectively. Optionally, choose to use single-tree mode. Naive mode is not available as an option for this constructor; instead, to run naive computation, construct a tree with all of the points in one leaf (i.e. *leafSize* = number of points). Additionally, an instantiated distance metric can be given, for cases where the distance metric holds data.

There is no copying of the data matrices in this constructor (because tree-building is not necessary), so this is the constructor to use when copies absolutely must be avoided.

Note

Because tree-building (at least with *BinarySpaceTree*) modifies the ordering of a matrix, be sure you pass the modified matrix to this object! In addition, mapping the points of the matrix back to their original indices is not done when this constructor is used.

Parameters

<i>referenceTree</i>	Pre-built tree for reference points.
<i>queryTree</i>	Pre-built tree for query points.
<i>referenceSet</i>	Set of reference points corresponding to <i>referenceTree</i> .
<i>querySet</i>	Set of query points corresponding to <i>queryTree</i> .
<i>singleMode</i>	Whether single-tree computation should be used (as opposed to dual-tree computation).
<i>metric</i>	Instantiated distance metric.

```
22.162.2.4  template<typename SortPolicy = NearestNeighborSort, typename MetricType = mpack::metric::SquaredEuclidean<
Distance, typename TreeType = tree::BinarySpaceTree<bound::HRectBound<2, false>, RAQueryStat<SortPolicy>
>> RASearch< SortPolicy, MetricType, TreeType >::RASearch ( TreeType * referenceTree, const typename
TreeType::Mat & referenceSet, const bool singleMode = false, const MetricType metric = MetricType() )
```

Initialize the **RASearch** (p. 666) object with the given reference dataset and pre-constructed tree.

It is assumed that the points in *referenceSet* correspond to the points in *referenceTree*. Optionally, choose to use single-tree mode. Naive mode is not available as an option for this constructor; instead, to run naive computation, construct a tree with all the points in one leaf (i.e. *leafSize* = number of points). Additionally, an instantiated distance metric can be given, for the case where the distance metric holds data.

There is no copying of the data matrices in this constructor (because tree-building is not necessary), so this is the constructor to use when copies absolutely must be avoided.

Note

Because tree-building (at least with BinarySpaceTree) modifies the ordering of a matrix, be sure you pass the modified matrix to this object! In addition, mapping the points of the matrix back to their original indices is not done when this constructor is used.

Parameters

<i>referenceTree</i>	Pre-built tree for reference points.
<i>referenceSet</i>	Set of reference points corresponding to referenceTree.
<i>singleMode</i>	Whether single-tree computation should be used (as opposed to dual-tree computation).
<i>metric</i>	Instantiated distance metric.

```
22.162.2.5  template<typename SortPolicy = NearestNeighborSort, typename MetricType = mpack::metric::SquaredEuclidean↵
Distance, typename TreeType = tree::BinarySpaceTree<bound::HRectBound<2, false>, RAQueryStat<SortPolicy>↵
>> RASearch< SortPolicy, MetricType, TreeType >::~~RASearch ( )
```

Delete the **RASearch** (p. 666) object.

The tree is the only member we are responsible for deleting. The others will take care of themselves.

22.162.3 Member Function Documentation

```
22.162.3.1  template<typename SortPolicy = NearestNeighborSort, typename MetricType = mpack::metric::SquaredEuclidean↵
Distance, typename TreeType = tree::BinarySpaceTree<bound::HRectBound<2, false>, RAQueryStat<SortPolicy>↵
>> void RASearch< SortPolicy, MetricType, TreeType >::ResetQueryTree ( )
```

This function recursively resets the RAQueryStat of the queryTree to set 'bound' to WorstDistance and the 'num↵SamplesMade' to 0.

This allows a user to perform multiple searches on the same pair of trees, possibly with different levels of approximation without requiring to build a new pair of trees for every new (approximate) search.

```
22.162.3.2  template<typename SortPolicy = NearestNeighborSort, typename MetricType = mpack::metric::SquaredEuclidean↵
Distance, typename TreeType = tree::BinarySpaceTree<bound::HRectBound<2, false>, RAQueryStat<SortPolicy>↵
>> void RASearch< SortPolicy, MetricType, TreeType >::ResetRAQueryStat ( TreeType * treeNode )
[private]
```

Parameters

<i>treeNode</i>	The node of the tree whose RAQueryStat is reset and whose children are to be explored recursively.
-----------------	--

```
22.162.3.3  template<typename SortPolicy = NearestNeighborSort, typename MetricType = mpack::metric::SquaredEuclidean↵
Distance, typename TreeType = tree::BinarySpaceTree<bound::HRectBound<2, false>, RAQueryStat<SortPolicy>↵
>> void RASearch< SortPolicy, MetricType, TreeType >::Search ( const size_t k, arma::Mat< size_t > &
resultingNeighbors, arma::mat & distances, const double tau = 5, const double alpha = 0.95, const bool
sampleAtLeaves = false, const bool firstLeafExact = false, const size_t singleSampleLimit = 20 )
```

Compute the rank approximate nearest neighbors and store the output in the given matrices.

The matrices will be set to the size of n columns by k rows, where n is the number of points in the query dataset and k is the number of neighbors being searched for.

Note that tau, the rank-approximation parameter, specifies that we are looking for k neighbors with probability alpha of being in the top tau percent of nearest neighbors. So, as an example, if our dataset has 1000 points, and we want 5 nearest neighbors with 95% probability of being in the top 5% of nearest neighbors (or, the top 50 nearest neighbors), we set k = 5, tau = 5, and alpha = 0.95.

The method will fail (and issue a failure message) if the value of tau is too low: tau must be set such that the number of points in the corresponding percentile of the data is greater than k. Thus, if we choose tau = 0.1 with a dataset of 1000 points and k = 5, then we are attempting to choose 5 nearest neighbors out of the closest 1 point – this is invalid.

Parameters

<i>k</i>	Number of neighbors to search for.
<i>resulting↔ Neighbors</i>	Matrix storing lists of neighbors for each query point.
<i>distances</i>	Matrix storing distances of neighbors for each query point.
<i>tau</i>	The rank-approximation in percentile of the data. The default value is 5%.
<i>alpha</i>	The desired success probability. The default value is 0.95.
<i>sampleAtLeaves</i>	Sample at leaves for faster but less accurate computation. This defaults to 'false'.
<i>firstLeafExact</i>	Traverse to the first leaf without approximation. This can ensure that the query definitely finds its (near) duplicate if there exists one. This defaults to 'false' for now.
<i>singleSample↔ Limit</i>	The limit on the largest node that can be approximated by sampling. This defaults to 20.

```
22.162.3.4  template<typename SortPolicy = NearestNeighborSort, typename MetricType = mpack::metric::SquaredEuclidean↔
Distance, typename TreeType = tree::BinarySpaceTree<bound::HRectBound<2, false>, RAQueryStat<SortPolicy>
>> std::string RASearch< SortPolicy, MetricType, TreeType >::ToString ( ) const
```

22.162.4 Member Data Documentation

```
22.162.4.1  template<typename SortPolicy = NearestNeighborSort, typename MetricType = mpack::metric::SquaredEuclidean↔
Distance, typename TreeType = tree::BinarySpaceTree<bound::HRectBound<2, false>, RAQueryStat<SortPolicy>
>> bool RASearch< SortPolicy, MetricType, TreeType >::hasQuerySet [private]
```

Indicates if a separate query set was passed.

Definition at line 271 of file ra_search.hpp.

```
22.162.4.2  template<typename SortPolicy = NearestNeighborSort, typename MetricType = mpack::metric::SquaredEuclidean↔
Distance, typename TreeType = tree::BinarySpaceTree<bound::HRectBound<2, false>, RAQueryStat<SortPolicy>
>> MetricType RASearch< SortPolicy, MetricType, TreeType >::metric [private]
```

Instantiation of kernel.

Definition at line 279 of file ra_search.hpp.

```
22.162.4.3  template<typename SortPolicy = NearestNeighborSort, typename MetricType = mpack::metric::SquaredEuclidean↔
Distance, typename TreeType = tree::BinarySpaceTree<bound::HRectBound<2, false>, RAQueryStat<SortPolicy>
>> bool RASearch< SortPolicy, MetricType, TreeType >::naive [private]
```

Indicates if naive random sampling on the set is being used.

Definition at line 274 of file ra_search.hpp.

```
22.162.4.4  template<typename SortPolicy = NearestNeighborSort, typename MetricType = mpack::metric::SquaredEuclidean↵
Distance, typename TreeType = tree::BinarySpaceTree<bound::HRectBound<2, false>, RAQueryStat<SortPolicy>
>> size_t RASearch< SortPolicy, MetricType, TreeType >::numberOfPrunes  [private]
```

Total number of pruned nodes during the neighbor search.

Definition at line 287 of file ra_search.hpp.

```
22.162.4.5  template<typename SortPolicy = NearestNeighborSort, typename MetricType = mpack::metric::SquaredEuclidean↵
Distance, typename TreeType = tree::BinarySpaceTree<bound::HRectBound<2, false>, RAQueryStat<SortPolicy>
>> std::vector<size_t> RASearch< SortPolicy, MetricType, TreeType >::oldFromNewQueries  [private]
```

Permutations of query points during tree building.

Definition at line 284 of file ra_search.hpp.

```
22.162.4.6  template<typename SortPolicy = NearestNeighborSort, typename MetricType = mpack::metric::SquaredEuclidean↵
Distance, typename TreeType = tree::BinarySpaceTree<bound::HRectBound<2, false>, RAQueryStat<SortPolicy>
>> std::vector<size_t> RASearch< SortPolicy, MetricType, TreeType >::oldFromNewReferences  [private]
```

Permutations of reference points during tree building.

Definition at line 282 of file ra_search.hpp.

```
22.162.4.7  template<typename SortPolicy = NearestNeighborSort, typename MetricType = mpack::metric::SquaredEuclidean↵
Distance, typename TreeType = tree::BinarySpaceTree<bound::HRectBound<2, false>, RAQueryStat<SortPolicy>
>> arma::mat RASearch< SortPolicy, MetricType, TreeType >::queryCopy  [private]
```

Copy of query dataset (if we need it, because tree building modifies it).

Definition at line 256 of file ra_search.hpp.

```
22.162.4.8  template<typename SortPolicy = NearestNeighborSort, typename MetricType = mpack::metric::SquaredEuclidean↵
Distance, typename TreeType = tree::BinarySpaceTree<bound::HRectBound<2, false>, RAQueryStat<SortPolicy>
>> const arma::mat& RASearch< SortPolicy, MetricType, TreeType >::querySet  [private]
```

Query dataset (may not be given).

Definition at line 261 of file ra_search.hpp.

```
22.162.4.9  template<typename SortPolicy = NearestNeighborSort, typename MetricType = mpack::metric::SquaredEuclidean↵
Distance, typename TreeType = tree::BinarySpaceTree<bound::HRectBound<2, false>, RAQueryStat<SortPolicy>
>> TreeType* RASearch< SortPolicy, MetricType, TreeType >::queryTree  [private]
```

Pointer to the root of the query tree (might not exist).

Definition at line 266 of file ra_search.hpp.

```
22.162.4.10  template<typename SortPolicy = NearestNeighborSort, typename MetricType = mpack::metric::SquaredEuclidean↵
Distance, typename TreeType = tree::BinarySpaceTree<bound::HRectBound<2, false>, RAQueryStat<SortPolicy>
>> arma::mat RASearch< SortPolicy, MetricType, TreeType >::referenceCopy  [private]
```

Copy of reference dataset (if we need it, because tree building modifies it).

Definition at line 254 of file ra_search.hpp.

```
22.162.4.11  template<typename SortPolicy = NearestNeighborSort, typename MetricType = mlpack::metric::SquaredEuclidean↵
Distance, typename TreeType = tree::BinarySpaceTree<bound::HRectBound<2, false>, RAQueryStat<SortPolicy>↵
>> const arma::mat& RASearch< SortPolicy, MetricType, TreeType >::referenceSet  [private]
```

Reference dataset.

Definition at line 259 of file ra_search.hpp.

```
22.162.4.12  template<typename SortPolicy = NearestNeighborSort, typename MetricType = mlpack::metric::SquaredEuclidean↵
Distance, typename TreeType = tree::BinarySpaceTree<bound::HRectBound<2, false>, RAQueryStat<SortPolicy>↵
>> TreeType* RASearch< SortPolicy, MetricType, TreeType >::referenceTree  [private]
```

Pointer to the root of the reference tree.

Definition at line 264 of file ra_search.hpp.

```
22.162.4.13  template<typename SortPolicy = NearestNeighborSort, typename MetricType = mlpack::metric::SquaredEuclidean↵
Distance, typename TreeType = tree::BinarySpaceTree<bound::HRectBound<2, false>, RAQueryStat<SortPolicy>↵
>> bool RASearch< SortPolicy, MetricType, TreeType >::singleMode  [private]
```

Indicates if single-tree search is being used (opposed to dual-tree).

Definition at line 276 of file ra_search.hpp.

```
22.162.4.14  template<typename SortPolicy = NearestNeighborSort, typename MetricType = mlpack::metric::SquaredEuclidean↵
Distance, typename TreeType = tree::BinarySpaceTree<bound::HRectBound<2, false>, RAQueryStat<SortPolicy>↵
>> bool RASearch< SortPolicy, MetricType, TreeType >::treeOwner  [private]
```

If true, this object created the trees and is responsible for them.

Definition at line 269 of file ra_search.hpp.

The documentation for this class was generated from the following file:

- src/mlpack/methods/rann/ra_search.hpp

22.163 TraversalInfo< TreeType > Class Template Reference

The **TraversalInfo** (p. 673) class holds traversal information which is used in dual-tree (and single-tree) traversals.

Public Member Functions

- **TraversalInfo** ()
*Create the **TraversalInfo** (p. 673) object and initialize the pointers to NULL.*
- double **LastBaseCase** () const
Get the base case associated with the last node combination.
- double & **LastBaseCase** ()
Modify the base case associated with the last node combination.

- `TreeType * LastQueryNode () const`
Get the last query node.
- `TreeType *& LastQueryNode ()`
Modify the last query node.
- `TreeType * LastReferenceNode () const`
Get the last reference node.
- `TreeType *& LastReferenceNode ()`
Modify the last reference node.
- `double LastScore () const`
Get the score associated with the last query and reference nodes.
- `double & LastScore ()`
Modify the score associated with the last query and reference nodes.

Private Attributes

- `double lastBaseCase`
The last base case.
- `TreeType * lastQueryNode`
The last query node.
- `TreeType * lastReferenceNode`
The last reference node.
- `double lastScore`
The last score.

22.163.1 Detailed Description

```
template<typename TreeType>class TraversalInfo< TreeType >
```

The **TraversalInfo** (p. 673) class holds traversal information which is used in dual-tree (and single-tree) traversals.

A traversal should be updating the members of this class before `Score()` is called. This class should be held as a member of the `RuleType` class and the interface to it should be through a **TraversalInfo()** (p. 675) method.

The information held by this class is the last node combination visited before the current node combination was recursed into, and the score resulting from when `Score()` was called on that combination. However, this information is identical for a query node and a reference node in a particular node combination, so traversals only need to update the **TraversalInfo** (p. 673) object in a query node (and the algorithms should only use the **TraversalInfo** (p. 673) object from a query node).

In general, this auxiliary traversal information is used to try and make a prune without needing to call `BaseCase()` or calculate the distance between nodes. Using this information you can place bounds on the distance between the two nodes quickly.

If the traversal is not updating the members of this class correctly, a likely result is a null pointer dereference. Dual-tree algorithms should assume that the members are set properly and should not need to check for null pointers.

There is one exception, which is the root node combination; the score can be set to 0 and the query and reference nodes can just be set to the root nodes; no algorithm should be able to prune the root combination anyway.

Definition at line 49 of file `traversal_info.hpp`.

22.163.2 Constructor & Destructor Documentation

22.163.2.1 `template<typename TreeType > TraversalInfo< TreeType >::TraversalInfo () [inline]`

Create the **TraversalInfo** (p. 673) object and initialize the pointers to NULL.

Definition at line 55 of file traversal_info.hpp.

22.163.3 Member Function Documentation

22.163.3.1 `template<typename TreeType > double TraversalInfo< TreeType >::LastBaseCase () const [inline]`

Get the base case associated with the last node combination.

Definition at line 77 of file traversal_info.hpp.

References TraversalInfo< TreeType >::lastBaseCase.

22.163.3.2 `template<typename TreeType > double& TraversalInfo< TreeType >::LastBaseCase () [inline]`

Modify the base case associated with the last node combination.

Definition at line 79 of file traversal_info.hpp.

References TraversalInfo< TreeType >::lastBaseCase.

22.163.3.3 `template<typename TreeType > TreeType* TraversalInfo< TreeType >::LastQueryNode () const [inline]`

Get the last query node.

Definition at line 62 of file traversal_info.hpp.

References TraversalInfo< TreeType >::lastQueryNode.

22.163.3.4 `template<typename TreeType > TreeType*& TraversalInfo< TreeType >::LastQueryNode () [inline]`

Modify the last query node.

Definition at line 64 of file traversal_info.hpp.

References TraversalInfo< TreeType >::lastQueryNode.

22.163.3.5 `template<typename TreeType > TreeType* TraversalInfo< TreeType >::LastReferenceNode () const [inline]`

Get the last reference node.

Definition at line 67 of file traversal_info.hpp.

References TraversalInfo< TreeType >::lastReferenceNode.

22.163.3.6 `template<typename TreeType > TreeType*& TraversalInfo< TreeType >::LastReferenceNode () [inline]`

Modify the last reference node.

Definition at line 69 of file traversal_info.hpp.

References TraversalInfo< TreeType >::lastReferenceNode.

22.163.3.7 `template<typename TreeType > double TraversalInfo< TreeType >::LastScore () const [inline]`

Get the score associated with the last query and reference nodes.

Definition at line 72 of file traversal_info.hpp.

References TraversalInfo< TreeType >::lastScore.

22.163.3.8 `template<typename TreeType > double& TraversalInfo< TreeType >::LastScore () [inline]`

Modify the score associated with the last query and reference nodes.

Definition at line 74 of file traversal_info.hpp.

References TraversalInfo< TreeType >::lastScore.

22.163.4 Member Data Documentation

22.163.4.1 `template<typename TreeType > double TraversalInfo< TreeType >::lastBaseCase [private]`

The last base case.

Definition at line 89 of file traversal_info.hpp.

Referenced by TraversalInfo< TreeType >::LastBaseCase().

22.163.4.2 `template<typename TreeType > TreeType* TraversalInfo< TreeType >::lastQueryNode [private]`

The last query node.

Definition at line 83 of file traversal_info.hpp.

Referenced by TraversalInfo< TreeType >::LastQueryNode().

22.163.4.3 `template<typename TreeType > TreeType* TraversalInfo< TreeType >::lastReferenceNode [private]`

The last reference node.

Definition at line 85 of file traversal_info.hpp.

Referenced by TraversalInfo< TreeType >::LastReferenceNode().

22.163.4.4 `template<typename TreeType > double TraversalInfo< TreeType >::lastScore [private]`

The last score.

Definition at line 87 of file traversal_info.hpp.

Referenced by TraversalInfo< TreeType >::LastScore().

The documentation for this class was generated from the following file:

- src/mlpack/core/tree/traversal_info.hpp

Chapter 23

File Documentation

23.1 doc/guide/build.hpp File Reference

23.2 doc/guide/iodoc.hpp File Reference

23.3 doc/guide/matrices.hpp File Reference

23.4 doc/guide/sample.hpp File Reference

23.5 doc/guide/timer.hpp File Reference

23.6 doc/tutorials/amf/amf.txt File Reference

Tutorial for how to use the AMF class.

23.6.1 Detailed Description

Tutorial for how to use the AMF class.

Author

Sumedh Ghaisas

23.7 doc/tutorials/det/det.txt File Reference

Tutorial for how to perform density estimation with Density Estimation Trees (DET).

Functions

- **ff** **\$V** (t)\f \$ is the volume of the node\ f \$t\ f \$ and\ f \$tilde

- see subsection cli_alt_reg_tut Alternate DET **regularization** The usual regularized error $f_{R_alpha}(t)$ of a node f is given by

Variables

- f is the **set** of leaves in the subtree rooted at f For the purposes of density **estimation**
- this option is not available in DET right **now**
- f is the **set** of leaves in the subtree rooted at f For the purposes of density there is a different form of **regularization**
- f is the **set** of leaves in the subtree rooted at f For the purposes of density there is a different form of we penalize the sum of the inverse of the volumes of the leaves With this very small volume nodes are discouraged unless the data actually warrants it **Thus**

23.7.1 Detailed Description

Tutorial for how to perform density estimation with Density Estimation Trees (DET).

Author

Parikshit Ram

23.7.2 Function Documentation

23.7.2.1 $f_{f,V}(t)$

Definition at line 374 of file det.txt.

23.7.2.2 see subsection cli_alt_reg_tut Alternate DET **regularization** The usual regularized error $f_{R_alpha}(t)$

Definition at line 367 of file det.txt.

23.7.3 Variable Documentation

23.7.3.1 f is the **set** of leaves in the subtree rooted at f For the purposes of density estimation

Definition at line 377 of file det.txt.

23.7.3.2 this option is not available in DET right now

Definition at line 364 of file det.txt.

23.7.3.3 f is the **set** of leaves in the subtree rooted at f For the purposes of density there is a different form of we penalize the sum of the inverse of the volumes of the leaves With this regularization

Definition at line 377 of file det.txt.

23.7.3.4 f is the set of leaves in the subtree rooted at f . For the purposes of density there is a different form of we penalize the sum of the inverse of the volumes of the leaves. With this very small volume nodes are discouraged unless the data actually warrants it. Thus

Definition at line 377 of file det.txt.

23.8 doc/tutorials/emst/emst.txt File Reference

Tutorial for the Euclidean Minimum Spanning Tree algorithm.

23.8.1 Detailed Description

Tutorial for the Euclidean Minimum Spanning Tree algorithm.

Author

Bill March

23.9 doc/tutorials/fastmks/fastmks.txt File Reference

Tutorial for how to use FastMKS in mlpack.

23.9.1 Detailed Description

Tutorial for how to use FastMKS in mlpack.

Author

Ryan Curtin

23.10 doc/tutorials/kmeans/kmeans.txt File Reference

Tutorial for how to use k-means in mlpack.

23.10.1 Detailed Description

Tutorial for how to use k-means in mlpack.

Author

Ryan Curtin

23.11 doc/tutorials/linear_regression/linear_regression.txt File Reference

Tutorial for how to use the LinearRegression class.

23.11.1 Detailed Description

Tutorial for how to use the LinearRegression class.

Author

James Cline

23.12 doc/tutorials/neighbor_search/neighbor_search.txt File Reference

Tutorial for how to use the NeighborSearch class.

23.12.1 Detailed Description

Tutorial for how to use the NeighborSearch class.

Author

Ryan Curtin

23.13 doc/tutorials/range_search/range_search.txt File Reference

Tutorial for how to use the RangeSearch class.

23.13.1 Detailed Description

Tutorial for how to use the RangeSearch class.

Author

Ryan Curtin

23.14 doc/tutorials/tutorials.txt File Reference

List of MLPACK tutorials.

23.14.1 Detailed Description

List of MLPACK tutorials.

Author

Ryan Curtin

23.15 src/mlpack/CMakeLists.txt File Reference

Functions

- **include_directories** (..) **set**(MLPACK_SRCS \$

23.15.1 Function Documentation

23.15.1.1 include_directories (..)

Definition at line 1 of file CMakeLists.txt.

23.16 src/mlpack/core/CMakeLists.txt File Reference

Functions

- **add_subdirectory** (\${dir}) endforeach() **set**(MLPACK_SRCS \$
- **set** (DIRS arma_extend data dists kernels math metrics optimizers tree util) foreach(dir \$

23.16.1 Function Documentation

23.16.1.1 add_subdirectory (\${dir})

Definition at line 15 of file CMakeLists.txt.

23.16.1.2 set (DIRS arma_extend data dists kernels math metrics optimizers tree *util*)

Definition at line 2 of file CMakeLists.txt.

23.17 src/mlpack/core/data/CMakeLists.txt File Reference

Functions

- **set** (SOURCES load.hpp load_impl.hpp normalize_labels.hpp normalize_labels_impl.hpp save.hpp save_impl.↵
hpp) **set**(DIR_SRCS) foreach(file \$
- **set** (DIR_SRCS \${DIR_SRCS}\${CMAKE_CURRENT_SOURCE_DIR}/\${file}) endforeach() **set**(MLPACK_SRCS
\$

23.17.1 Function Documentation

23.17.1.1 set (SOURCES load.hpp load_impl.hpp normalize_labels.hpp normalize_labels_impl.hpp save.hpp save_impl. *hpp*)

Definition at line 3 of file CMakeLists.txt.

23.17.1.2 `set (DIR_SRCS ${DIR_SRCS}${CMAKE_CURRENT_SOURCE_DIR}/${file})`

Definition at line 15 of file CMakeLists.txt.

23.18 src/mlpack/core/dists/CMakeLists.txt File Reference

Functions

- `set (SOURCES discrete_distribution.hpp discrete_distribution.cpp gaussian_distribution.hpp gaussian_↵
distribution.cpp laplace_distribution.hpp laplace_distribution.cpp) set(DIR_SRCS) foreach(file $`
- `set (DIR_SRCS ${DIR_SRCS}${CMAKE_CURRENT_SOURCE_DIR}/${file}) endforeach() set(MLPACK_SRCS
$`

23.18.1 Function Documentation

23.18.1.1 `set (SOURCES discrete_distribution.hpp discrete_distribution.cpp gaussian_distribution.hpp gaussian_distribution.cpp
laplace_distribution.hpp laplace_distribution. cpp)`

Definition at line 3 of file CMakeLists.txt.

23.18.1.2 `set (DIR_SRCS ${DIR_SRCS}${CMAKE_CURRENT_SOURCE_DIR}/${file})`

Definition at line 15 of file CMakeLists.txt.

23.19 src/mlpack/core/kernels/CMakeLists.txt File Reference

Functions

- `set (SOURCES cosine_distance.hpp cosine_distance_impl.hpp epanechnikov_kernel.hpp epanechnikov_↵
kernel_impl.hpp epanechnikov_kernel.cpp example_kernel.hpp gaussian_kernel.hpp hyperbolic_tangent_↵
kernel.hpp kernel_traits.hpp laplacian_kernel.hpp linear_kernel.hpp polynomial_kernel.hpp pspectrum_string_↵
_kernel.hpp pspectrum_string_kernel_impl.hpp pspectrum_string_kernel.cpp spherical_kernel.hpp triangular_↵
kernel.hpp) set(DIR_SRCS) foreach(file $`
- `set (DIR_SRCS ${DIR_SRCS}${CMAKE_CURRENT_SOURCE_DIR}/${file}) endforeach() set(MLPACK_SRCS
$`

23.19.1 Function Documentation

23.19.1.1 `set (SOURCES cosine_distance.hpp cosine_distance_impl.hpp epanechnikov_kernel.hpp epanechnikov_kernel_impl.hpp
epanechnikov_kernel.cpp example_kernel.hpp gaussian_kernel.hpp hyperbolic_tangent_kernel.hpp
kernel_traits.hpp laplacian_kernel.hpp linear_kernel.hpp polynomial_kernel.hpp pspectrum_string_kernel.hpp
pspectrum_string_kernel_impl.hpp pspectrum_string_kernel.cpp spherical_kernel.hpp triangular_kernel. hpp)`

Definition at line 3 of file CMakeLists.txt.

23.19.1.2 `set (DIR_SRCS ${DIR_SRCS}${CMAKE_CURRENT_SOURCE_DIR}/${file})`

Definition at line 26 of file CMakeLists.txt.

23.20 src/mlpack/core/math/CMakeLists.txt File Reference

Functions

- `set (SOURCES clamp.hpp lin_alg.hpp lin_alg.cpp random.hpp random.cpp range.hpp range_impl.hpp round.hpp) set(DIR_SRCS) foreach(file $`
- `set (DIR_SRCS ${DIR_SRCS}${CMAKE_CURRENT_SOURCE_DIR}/${file}) endforeach() set(MLPACK_SRCS $`

23.20.1 Function Documentation

23.20.1.1 `set (SOURCES clamp.hpp lin_alg.hpp lin_alg.cpp random.hpp random.cpp range.hpp range_impl.hpp round. hpp)`

Definition at line 3 of file CMakeLists.txt.

23.20.1.2 `set (DIR_SRCS ${DIR_SRCS}${CMAKE_CURRENT_SOURCE_DIR}/${file})`

Definition at line 17 of file CMakeLists.txt.

23.21 src/mlpack/core/metrics/CMakeLists.txt File Reference

Functions

- `set (SOURCES ip_metric.hpp ip_metric_impl.hpp lmetric.hpp lmetric_impl.hpp mahalanobis_distance.hpp mahalanobis_distance_impl.hpp) set(DIR_SRCS) foreach(file $`
- `set (DIR_SRCS ${DIR_SRCS}${CMAKE_CURRENT_SOURCE_DIR}/${file}) endforeach() set(MLPACK_SRCS $`

23.21.1 Function Documentation

23.21.1.1 `set (SOURCES ip_metric.hpp ip_metric_impl.hpp lmetric.hpp lmetric_impl.hpp mahalanobis_distance.hpp mahalanobis_distance_impl. hpp)`

Definition at line 3 of file CMakeLists.txt.

23.21.1.2 `set (DIR_SRCS ${DIR_SRCS}${CMAKE_CURRENT_SOURCE_DIR}/${file})`

Definition at line 15 of file CMakeLists.txt.

23.22 src/mlpack/core/optimizers/aug_lagrangian/CMakeLists.txt File Reference

Functions

- **set** (SOURCES aug_lagrangian.hpp aug_lagrangian_impl.hpp aug_lagrangian_function.hpp aug_lagrangian_function_impl.hpp aug_lagrangian_test_functions.hpp aug_lagrangian_test_functions.cpp) set(DIR_SRCS) foreach(file \$
- **set** (DIR_SRCS \${DIR_SRCS}\${CMAKE_CURRENT_SOURCE_DIR}/\${file}) endforeach() set(MLPACK_SRCS \$

23.22.1 Function Documentation

23.22.1.1 **set** (SOURCES aug_lagrangian.hpp aug_lagrangian_impl.hpp aug_lagrangian_function.hpp
aug_lagrangian_function_impl.hpp aug_lagrangian_test_functions.hpp aug_lagrangian_test_functions. *cpp*)

Definition at line 1 of file CMakeLists.txt.

23.22.1.2 **set** (DIR_SRCS \${DIR_SRCS}\${CMAKE_CURRENT_SOURCE_DIR}/\${file})

Definition at line 12 of file CMakeLists.txt.

23.23 src/mlpack/core/optimizers/CMakeLists.txt File Reference

Functions

- **add_subdirectory** (\${dir}) endforeach() **set**(MLPACK_SRCS \$
- **set** (DIRS aug_lagrangian lbfgs lrsgd sa sgd) foreach(dir \$

23.23.1 Function Documentation

23.23.1.1 **add_subdirectory** (*\${dir}*)

Definition at line 10 of file CMakeLists.txt.

23.23.1.2 **set** (DIRS aug_lagrangian lbfgs lrsgd sa *sgd*)

Definition at line 1 of file CMakeLists.txt.

23.24 src/mlpack/core/optimizers/lbfgs/CMakeLists.txt File Reference

Functions

- **set** (SOURCES lbfgs_impl.hpp lbfgs.hpp test_functions.hpp test_functions.cpp) set(DIR_SRCS) foreach(file \$
- **set** (DIR_SRCS \${DIR_SRCS}\${CMAKE_CURRENT_SOURCE_DIR}/\${file}) endforeach() set(MLPACK_SRCS \$

23.24.1 Function Documentation

23.24.1.1 `set (SOURCES lbfgs_impl.hpp lbfgs.hpp test_functions.hpp test_functions. cpp)`

Definition at line 1 of file CMakeLists.txt.

23.24.1.2 `set (DIR_SRCS ${DIR_SRCS}${CMAKE_CURRENT_SOURCE_DIR}/${file})`

Definition at line 10 of file CMakeLists.txt.

23.25 src/mlpack/core/optimizers/lrsdp/CMakeLists.txt File Reference

Functions

- `set (SOURCES lrsdp.hpp lrsdp.cpp lrsdp_function.hpp lrsdp_function.cpp) set(DIR_SRCS) foreach(file $`
- `set (DIR_SRCS ${DIR_SRCS}${CMAKE_CURRENT_SOURCE_DIR}/${file}) endforeach() set(MLPACK_SRCS $`

23.25.1 Function Documentation

23.25.1.1 `set (SOURCES lrsdp.hpp lrsdp.cpp lrsdp_function.hpp lrsdp_function. cpp)`

Definition at line 1 of file CMakeLists.txt.

23.25.1.2 `set (DIR_SRCS ${DIR_SRCS}${CMAKE_CURRENT_SOURCE_DIR}/${file})`

Definition at line 10 of file CMakeLists.txt.

23.26 src/mlpack/core/optimizers/sa/CMakeLists.txt File Reference

Functions

- `set (SOURCES sa.hpp sa_impl.hpp exponential_schedule.hpp) set(DIR_SRCS) foreach(file $`
- `set (DIR_SRCS ${DIR_SRCS}${CMAKE_CURRENT_SOURCE_DIR}/${file}) endforeach() set(MLPACK_SRCS $`

23.26.1 Function Documentation

23.26.1.1 `set (SOURCES sa.hpp sa_impl.hpp exponential_schedule. hpp)`

Definition at line 1 of file CMakeLists.txt.

23.26.1.2 `set (DIR_SRCS ${DIR_SRCS}${CMAKE_CURRENT_SOURCE_DIR}/${file})`

Definition at line 9 of file CMakeLists.txt.

23.27 src/mlpack/core/optimizers/sgd/CMakeLists.txt File Reference

Functions

- **set** (SOURCES sgd.hpp sgd_impl.hpp test_function.hpp test_function.cpp) set(DIR_SRCS) foreach(file \$
- **set** (DIR_SRCS \${DIR_SRCS}\${CMAKE_CURRENT_SOURCE_DIR}/\${file}) endforeach() set(MLPACK_SRCS \$

23.27.1 Function Documentation

23.27.1.1 **set** (SOURCES sgd.hpp sgd_impl.hpp test_function.hpp test_function. *cpp*)

Definition at line 1 of file CMakeLists.txt.

23.27.1.2 **set** (DIR_SRCS \${DIR_SRCS}\${CMAKE_CURRENT_SOURCE_DIR}/\${file})

Definition at line 10 of file CMakeLists.txt.

23.28 src/mlpack/core/tree/CMakeLists.txt File Reference

Functions

- **set** (SOURCES ballbound.hpp ballbound_impl.hpp binary_space_tree/binary_space_tree.hpp binary_space_↵
_tree/binary_space_tree_impl.hpp binary_space_tree/dual_tree_traverser.hpp binary_space_tree/dual_tree_↵
traverser_impl.hpp binary_space_tree/mean_split.hpp binary_space_tree/mean_split_impl.hpp binary_space_↵
_tree/single_tree_traverser.hpp binary_space_tree/single_tree_traverser_impl.hpp binary_space_tree/traits.↵
hpp bounds.hpp cosine_tree/cosine_tree.hpp cosine_tree/cosine_tree.cpp cover_tree/cover_tree.hpp cover_↵
_tree/cover_tree_impl.hpp cover_tree/first_point_is_root.hpp cover_tree/single_tree_traverser.hpp cover_↵
tree/single_tree_traverser_impl.hpp cover_tree/dual_tree_traverser.hpp cover_tree/dual_tree_traverser_impl.hpp
cover_tree/traits.hpp example_tree.hpp hrectbound.hpp hrectbound_impl.hpp mrkd_statistic.hpp mrkd_statistic_↵
_impl.hpp mrkd_statistic.cpp statistic.hpp traversal_info.hpp tree_traits.hpp) set(DIR_SRCS) foreach(file \$
- **set** (DIR_SRCS \${DIR_SRCS}\${CMAKE_CURRENT_SOURCE_DIR}/\${file}) endforeach() set(MLPACK_SRCS \$

23.28.1 Function Documentation

23.28.1.1 **set** (SOURCES ballbound.hpp ballbound_impl.hpp binary_space_tree/binary_space_tree.hpp binary_space_↵
_space_tree_impl.hpp binary_space_tree/dual_tree_traverser.hpp binary_space_tree/dual_tree_traverser_impl.hpp
binary_space_tree/mean_split.hpp binary_space_tree/mean_split_impl.hpp binary_space_tree/single_tree_traverser.hpp
binary_space_tree/single_tree_traverser_impl.hpp binary_space_tree/traits.hpp bounds.hpp cosine_tree/cosine_tree.hpp
cosine_tree/cosine_tree.cpp cover_tree/cover_tree.hpp cover_tree/cover_tree_impl.hpp cover_tree/first_point_is_root.hpp
cover_tree/single_tree_traverser.hpp cover_tree/single_tree_traverser_impl.hpp cover_tree/dual_tree_traverser.hpp
cover_tree/dual_tree_traverser_impl.hpp cover_tree/traits.hpp example_tree.hpp hrectbound.hpp hrectbound_impl.hpp
mrkd_statistic.hpp mrkd_statistic_impl.hpp mrkd_statistic.cpp statistic.hpp traversal_info.hpp tree_traits. *hpp*)

Definition at line 3 of file CMakeLists.txt.

23.28.1.2 `set (DIR_SRCS ${DIR_SRCS}${CMAKE_CURRENT_SOURCE_DIR}/${file})`

Definition at line 40 of file CMakeLists.txt.

23.29 src/mlpack/core/util/CMakeLists.txt File Reference

Functions

- `set (SOURCES cli.hpp cli.cpp cli_deleter.hpp cli_deleter.cpp cli_impl.hpp log.hpp log.cpp nullostream.hpp option.hpp option.cpp option_impl.hpp prefixedostream.hpp prefixedostream.cpp prefixedostream_impl.↵
hpp save_restore_utility.hpp save_restore_utility.cpp save_restore_utility_impl.hpp sfinae_utility.hpp string_util.↵
hpp string_util.cpp timers.hpp timers.cpp version.hpp version.cpp) set(DIR_SRCS) foreach(file $`
- `set (DIR_SRCS ${DIR_SRCS}${CMAKE_CURRENT_SOURCE_DIR}/${file}) endforeach() set(MLPACK_SRCS
$`

23.29.1 Function Documentation

23.29.1.1 `set (SOURCES cli.hpp cli.cpp cli_deleter.hpp cli_deleter.cpp cli_impl.hpp log.hpp log.cpp nullostream.hpp
option.hpp option.cpp option_impl.hpp prefixedostream.hpp prefixedostream.cpp prefixedostream_impl.hpp
save_restore_utility.hpp save_restore_utility.cpp save_restore_utility_impl.hpp sfinae_utility.hpp string_util.hpp
string_util.cpp timers.hpp timers.cpp version.hpp version. cpp)`

Definition at line 3 of file CMakeLists.txt.

23.29.1.2 `set (DIR_SRCS ${DIR_SRCS}${CMAKE_CURRENT_SOURCE_DIR}/${file})`

Definition at line 33 of file CMakeLists.txt.

23.30 src/mlpack/methods/amf/CMakeLists.txt File Reference

Functions

- `set (SOURCES amf.hpp amf_impl.hpp) set(DIR_SRCS) foreach(file $`
- `set (DIR_SRCS ${DIR_SRCS}${CMAKE_CURRENT_SOURCE_DIR}/${file}) endforeach() set(MLPACK_SRCS
$`

23.30.1 Function Documentation

23.30.1.1 `set (SOURCES amf.hpp amf_impl. hpp)`

Definition at line 3 of file CMakeLists.txt.

23.30.1.2 `set (DIR_SRCS ${DIR_SRCS}${CMAKE_CURRENT_SOURCE_DIR}/${file})`

Definition at line 11 of file CMakeLists.txt.

23.31 src/mlpack/methods/amf/init_rules/CMakeLists.txt File Reference

Functions

- **set** (SOURCES average_init.hpp random_init.hpp random_acol_init.hpp) set(DIR_SRCS) foreach(file \$
- **set** (DIR_SRCS \${DIR_SRCS}\${CMAKE_CURRENT_SOURCE_DIR}/\${file}) endforeach() set(MLPACK_SRCS \$

23.31.1 Function Documentation

23.31.1.1 **set** (SOURCES average_init.hpp random_init.hpp random_acol_init. *hpp*)

Definition at line 3 of file CMakeLists.txt.

23.31.1.2 **set** (DIR_SRCS \${DIR_SRCS}\${CMAKE_CURRENT_SOURCE_DIR}/\${file})

Definition at line 12 of file CMakeLists.txt.

23.32 src/mlpack/methods/amf/termination_policies/CMakeLists.txt File Reference

Functions

- **set** (SOURCES simple_residue_termination.hpp simple_tolerance_termination.hpp validation_rmse_termination.hpp incomplete_incremental_termination.hpp complete_incremental_termination.hpp) set(DIR_SRCS) foreach(file \$
- **set** (DIR_SRCS \${DIR_SRCS}\${CMAKE_CURRENT_SOURCE_DIR}/\${file}) endforeach() set(MLPACK_SRCS \$

23.32.1 Function Documentation

23.32.1.1 **set** (SOURCES simple_residue_termination.hpp simple_tolerance_termination.hpp validation_rmse_termination.hpp incomplete_incremental_termination.hpp complete_incremental_termination. *hpp*)

Definition at line 3 of file CMakeLists.txt.

23.32.1.2 **set** (DIR_SRCS \${DIR_SRCS}\${CMAKE_CURRENT_SOURCE_DIR}/\${file})

Definition at line 14 of file CMakeLists.txt.

23.33 src/mlpack/methods/amf/update_rules/CMakeLists.txt File Reference

Functions

- **set** (SOURCES nmf_als.hpp nmf_mult_dist.hpp nmf_mult_div.hpp svd_batch_learning.hpp svd_incomplete_incremental_learning.hpp svd_complete_incremental_learning.hpp) set(DIR_SRCS) foreach(file \$
- **set** (DIR_SRCS \${DIR_SRCS}\${CMAKE_CURRENT_SOURCE_DIR}/\${file}) endforeach() set(MLPACK_SRCS \$

23.33.1 Function Documentation

23.33.1.1 `set (SOURCES nmf_als.hpp nmf_mult_dist.hpp nmf_mult_div.hpp svd_batch_learning.hpp
svd_incomplete_incremental_learning.hpp svd_complete_incremental_learning. hpp)`

Definition at line 3 of file CMakeLists.txt.

23.33.1.2 `set (DIR_SRCS ${DIR_SRCS}${CMAKE_CURRENT_SOURCE_DIR}/${file})`

Definition at line 15 of file CMakeLists.txt.

23.34 src/mlpack/methods/cf/CMakeLists.txt File Reference

Functions

- `set (SOURCES cf.hpp cf_impl.hpp) set(DIR_SRCS) foreach(file $`
- `set (DIR_SRCS ${DIR_SRCS}${CMAKE_CURRENT_SOURCE_DIR}/${file}) endforeach() set(MLPACK_SRCS
$`

23.34.1 Function Documentation

23.34.1.1 `set (SOURCES cf.hpp cf_impl. hpp)`

Definition at line 3 of file CMakeLists.txt.

23.34.1.2 `set (DIR_SRCS ${DIR_SRCS}${CMAKE_CURRENT_SOURCE_DIR}/${file})`

Definition at line 11 of file CMakeLists.txt.

23.35 src/mlpack/methods/CMakeLists.txt File Reference

Functions

- `add_subdirectory (${dir}) endforeach() set(MLPACK_SRCS $`
- `set (DIRS amf cf decision_stump det emst fastmks gmm hmm kernel_pca kmeans lars linear_regression local_↵
coordinate_coding logistic_regression lsh naive_bayes nca neighbor_search nmf pca perceptron quic_svd radical
range_search rann regularized_svd sparse_autoencoder sparse_coding nystroem_method) foreach(dir $`

23.35.1 Function Documentation

23.35.1.1 `add_subdirectory (${dir})`

Definition at line 38 of file CMakeLists.txt.

23.35.1.2 `set (DIRS amf cf decision_stump det emst fastmks gmm hmm kernel_pca kmeans lars linear_regression
local_coordinate_coding logistic_regression lsh naive_bayes nca neighbor_search nmf pca perceptron quic_svd radical
range_search rann regularized_svd sparse_autoencoder sparse_coding nystroem_method)`

Definition at line 2 of file CMakeLists.txt.

23.36 src/mlpack/methods/decision_stump/CMakeLists.txt File Reference

Functions

- **cmake_minimum_required** (VERSION 2.8) `set(SOURCES decision_stump.hpp decision_stump_impl.hpp)
set(DIR_SRCS) foreach(file $`
- `set (DIR_SRCS ${DIR_SRCS}${CMAKE_CURRENT_SOURCE_DIR}/${file}) endforeach() set(MLPACK_SRCS
$`

23.36.1 Function Documentation

23.36.1.1 `cmake_minimum_required (VERSION 2.8)`

Definition at line 1 of file CMakeLists.txt.

23.36.1.2 `set (DIR_SRCS ${DIR_SRCS}${CMAKE_CURRENT_SOURCE_DIR}/${file})`

Definition at line 13 of file CMakeLists.txt.

23.37 src/mlpack/methods/det/CMakeLists.txt File Reference

Functions

- `set (SOURCES dtree.hpp dtree.cpp dt_utils.hpp dt_utils.cpp) set(DIR_SRCS) foreach(file $`
- `set (DIR_SRCS ${DIR_SRCS}${CMAKE_CURRENT_SOURCE_DIR}/${file}) endforeach() set(MLPACK_SRCS
$`

23.37.1 Function Documentation

23.37.1.1 `set (SOURCES dtree.hpp dtree.cpp dt_utils.hpp dt_utils. cpp)`

Definition at line 4 of file CMakeLists.txt.

23.37.1.2 `set (DIR_SRCS ${DIR_SRCS}${CMAKE_CURRENT_SOURCE_DIR}/${file})`

Definition at line 17 of file CMakeLists.txt.

23.38 src/mlpack/methods/emst/CMakeLists.txt File Reference

Functions

- **set** (SOURCES union_find.hpp dtb.hpp dtb_impl.hpp dtb_rules.hpp dtb_rules_impl.hpp dtb_stat.hpp edge_↔ pair.hpp) set(DIR_SRCS) foreach(file \$
- **set** (DIR_SRCS \${DIR_SRCS}\${CMAKE_CURRENT_SOURCE_DIR}/\${file}) endforeach() set(MLPACK_SRCS \$

23.38.1 Function Documentation

23.38.1.1 **set** (SOURCES union_find.hpp dtb.hpp dtb_impl.hpp dtb_rules.hpp dtb_rules_impl.hpp dtb_stat.hpp edge_pair. *hpp*)

Definition at line 3 of file CMakeLists.txt.

23.38.1.2 **set** (DIR_SRCS \${DIR_SRCS}\${CMAKE_CURRENT_SOURCE_DIR}/\${file})

Definition at line 18 of file CMakeLists.txt.

23.39 src/mlpack/methods/fastmks/CMakeLists.txt File Reference

Functions

- **set** (SOURCES fastmks.hpp fastmks_impl.hpp fastmks_rules.hpp fastmks_rules_impl.hpp) set(DIR_SRCS) foreach(file \$
- **set** (DIR_SRCS \${DIR_SRCS}\${CMAKE_CURRENT_SOURCE_DIR}/\${file}) endforeach() set(MLPACK_SRCS \$

23.39.1 Function Documentation

23.39.1.1 **set** (SOURCES fastmks.hpp fastmks_impl.hpp fastmks_rules.hpp fastmks_rules_impl. *hpp*)

Definition at line 3 of file CMakeLists.txt.

23.39.1.2 **set** (DIR_SRCS \${DIR_SRCS}\${CMAKE_CURRENT_SOURCE_DIR}/\${file})

Definition at line 13 of file CMakeLists.txt.

23.40 src/mlpack/methods/gmm/CMakeLists.txt File Reference

Functions

- **set** (SOURCES gmm.hpp gmm_impl.hpp phi.hpp em_fit.hpp em_fit_impl.hpp no_constraint.hpp positive_↔ definite_constraint.hpp diagonal_constraint.hpp eigenvalue_ratio_constraint.hpp) set(DIR_SRCS) foreach(file \$
- **set** (DIR_SRCS \${DIR_SRCS}\${CMAKE_CURRENT_SOURCE_DIR}/\${file}) endforeach() set(MLPACK_SRCS \$

23.40.1 Function Documentation

23.40.1.1 `set (SOURCES gmm.hpp gmm_impl.hpp phi.hpp em_fit.hpp em_fit_impl.hpp no_constraint.hpp positive_definite_constraint.hpp diagonal_constraint.hpp eigenvalue_ratio_constraint. hpp)`

Definition at line 3 of file CMakeLists.txt.

23.40.1.2 `set (DIR_SRCS ${DIR_SRCS}${CMAKE_CURRENT_SOURCE_DIR}/${file})`

Definition at line 18 of file CMakeLists.txt.

23.41 src/mlpack/methods/hmm/CMakeLists.txt File Reference

Functions

- `set (SOURCES hmm.hpp hmm_impl.hpp hmm_util.hpp hmm_util_impl.hpp) set(DIR_SRCS) foreach(file $`
- `set (DIR_SRCS ${DIR_SRCS}${CMAKE_CURRENT_SOURCE_DIR}/${file}) endforeach() set(MLPACK_SRCS $`

23.41.1 Function Documentation

23.41.1.1 `set (SOURCES hmm.hpp hmm_impl.hpp hmm_util.hpp hmm_util_impl. hpp)`

Definition at line 3 of file CMakeLists.txt.

23.41.1.2 `set (DIR_SRCS ${DIR_SRCS}${CMAKE_CURRENT_SOURCE_DIR}/${file})`

Definition at line 13 of file CMakeLists.txt.

23.42 src/mlpack/methods/kernel_pca/CMakeLists.txt File Reference

Functions

- `set (SOURCES kernel_pca.hpp kernel_pca_impl.hpp) set(DIR_SRCS) foreach(file $`
- `set (DIR_SRCS ${DIR_SRCS}${CMAKE_CURRENT_SOURCE_DIR}/${file}) endforeach() set(MLPACK_SRCS $`

23.42.1 Function Documentation

23.42.1.1 `set (SOURCES kernel_pca.hpp kernel_pca_impl. hpp)`

Definition at line 3 of file CMakeLists.txt.

23.42.1.2 `set (DIR_SRCS ${DIR_SRCS}${CMAKE_CURRENT_SOURCE_DIR}/${file})`

Definition at line 11 of file CMakeLists.txt.

23.43 src/mlpack/methods/kernel_pca/kernel_rules/CMakeLists.txt File Reference

Functions

- **set** (SOURCES nystroem_method.hpp naive_method.hpp) set(DIR_SRCS) foreach(file \$
- **set** (DIR_SRCS \${DIR_SRCS}\${CMAKE_CURRENT_SOURCE_DIR}/\${file}) endforeach() set(MLPACK_SRCS \$

23.43.1 Function Documentation

23.43.1.1 set (SOURCES nystroem_method.hpp naive_method. *hpp*)

Definition at line 3 of file CMakeLists.txt.

23.43.1.2 set (DIR_SRCS \${DIR_SRCS}\${CMAKE_CURRENT_SOURCE_DIR}/\${file})

Definition at line 11 of file CMakeLists.txt.

23.44 src/mlpack/methods/kmeans/CMakeLists.txt File Reference

Functions

- **set** (SOURCES allow_empty_clusters.hpp kmeans.hpp kmeans_impl.hpp max_variance_new_cluster.hpp max_↵
_variance_new_cluster_impl.hpp random_partition.hpp refined_start.hpp refined_start_impl.hpp) set(DIR_SR↵
CS) foreach(file \$
- **set** (DIR_SRCS \${DIR_SRCS}\${CMAKE_CURRENT_SOURCE_DIR}/\${file}) endforeach() set(MLPACK_SRCS \$

23.44.1 Function Documentation

23.44.1.1 set (SOURCES allow_empty_clusters.hpp kmeans.hpp kmeans_impl.hpp max_variance_new_cluster.hpp max_variance_new_cluster_impl.hpp random_partition.hpp refined_start.hpp refined_start_impl. *hpp*)

Definition at line 3 of file CMakeLists.txt.

23.44.1.2 set (DIR_SRCS \${DIR_SRCS}\${CMAKE_CURRENT_SOURCE_DIR}/\${file})

Definition at line 17 of file CMakeLists.txt.

23.45 src/mlpack/methods/lars/CMakeLists.txt File Reference

Functions

- **set** (SOURCES lars.hpp lars.cpp) set(DIR_SRCS) foreach(file \$
- **set** (DIR_SRCS \${DIR_SRCS}\${CMAKE_CURRENT_SOURCE_DIR}/\${file}) endforeach() set(MLPACK_SRCS \$

23.45.1 Function Documentation

23.45.1.1 `set (SOURCES lars.hpp lars. cpp)`

Definition at line 3 of file CMakeLists.txt.

23.45.1.2 `set (DIR_SRCS ${DIR_SRCS}${CMAKE_CURRENT_SOURCE_DIR}/${file})`

Definition at line 11 of file CMakeLists.txt.

23.46 src/mlpack/methods/linear_regression/CMakeLists.txt File Reference

Functions

- `set (SOURCES linear_regression.hpp linear_regression.cpp) set(DIR_SRCS) foreach(file $`
- `set (DIR_SRCS ${DIR_SRCS}${CMAKE_CURRENT_SOURCE_DIR}/${file}) endforeach() set(MLPACK_SRCS $`

23.46.1 Function Documentation

23.46.1.1 `set (SOURCES linear_regression.hpp linear_regression. cpp)`

Definition at line 4 of file CMakeLists.txt.

23.46.1.2 `set (DIR_SRCS ${DIR_SRCS}${CMAKE_CURRENT_SOURCE_DIR}/${file})`

Definition at line 12 of file CMakeLists.txt.

23.47 src/mlpack/methods/local_coordinate_coding/CMakeLists.txt File Reference

Functions

- `set (SOURCES lcc.hpp lcc_impl.hpp) set(DIR_SRCS) foreach(file $`
- `set (DIR_SRCS ${DIR_SRCS}${CMAKE_CURRENT_SOURCE_DIR}/${file}) endforeach() set(MLPACK_SRCS $`

23.47.1 Function Documentation

23.47.1.1 `set (SOURCES lcc.hpp lcc_impl. hpp)`

Definition at line 6 of file CMakeLists.txt.

23.47.1.2 `set (DIR_SRCS ${DIR_SRCS}${CMAKE_CURRENT_SOURCE_DIR}/${file})`

Definition at line 14 of file CMakeLists.txt.

23.48 src/mlpack/methods/logistic_regression/CMakeLists.txt File Reference

Functions

- **set** (SOURCES logistic_regression.hpp logistic_regression_impl.hpp logistic_regression_function.hpp logistic_regression_function.cpp) set(DIR_SRCS) foreach(file \$
- **set** (DIR_SRCS \${DIR_SRCS}\${CMAKE_CURRENT_SOURCE_DIR}/\${file}) endforeach() set(MLPACK_SRCS \$

23.48.1 Function Documentation

23.48.1.1 **set** (SOURCES logistic_regression.hpp logistic_regression_impl.hpp logistic_regression_function.hpp logistic_regression_function. *cpp*)

Definition at line 4 of file CMakeLists.txt.

23.48.1.2 **set** (DIR_SRCS \${DIR_SRCS}\${CMAKE_CURRENT_SOURCE_DIR}/\${file})

Definition at line 14 of file CMakeLists.txt.

23.49 src/mlpack/methods/lsh/CMakeLists.txt File Reference

Functions

- **set** (SOURCES lsh_search.hpp lsh_search_impl.hpp) set(DIR_SRCS) foreach(file \$
- **set** (DIR_SRCS \${DIR_SRCS}\${CMAKE_CURRENT_SOURCE_DIR}/\${file}) endforeach() set(MLPACK_CONTRIB_SRCS \$

23.49.1 Function Documentation

23.49.1.1 **set** (SOURCES lsh_search.hpp lsh_search_impl. *hpp*)

Definition at line 3 of file CMakeLists.txt.

23.49.1.2 **set** (DIR_SRCS \${DIR_SRCS}\${CMAKE_CURRENT_SOURCE_DIR}/\${file})

Definition at line 12 of file CMakeLists.txt.

23.50 src/mlpack/methods/mvu/CMakeLists.txt File Reference

Functions

- **set** (SOURCES mvu.hpp mvu.cpp) set(DIR_SRCS) foreach(file \$
- **set** (DIR_SRCS \${DIR_SRCS}\${CMAKE_CURRENT_SOURCE_DIR}/\${file}) endforeach() set(MLPACK_SRCS \$

23.50.1 Function Documentation

23.50.1.1 `set (SOURCES mvu.hpp mvu. cpp)`

Definition at line 3 of file CMakeLists.txt.

23.50.1.2 `set (DIR_SRCS ${DIR_SRCS}${CMAKE_CURRENT_SOURCE_DIR}/${file})`

Definition at line 11 of file CMakeLists.txt.

23.51 src/mlpack/methods/naive_bayes/CMakeLists.txt File Reference

Functions

- `set (SOURCES naive_bayes_classifier.hpp naive_bayes_classifier_impl.hpp) set(DIR_SRCS) foreach(file $`
- `set (DIR_SRCS ${DIR_SRCS}${CMAKE_CURRENT_SOURCE_DIR}/${file}) endforeach() set(MLPACK_SRCS`
`$`

23.51.1 Function Documentation

23.51.1.1 `set (SOURCES naive_bayes_classifier.hpp naive_bayes_classifier_impl. hpp)`

Definition at line 3 of file CMakeLists.txt.

23.51.1.2 `set (DIR_SRCS ${DIR_SRCS}${CMAKE_CURRENT_SOURCE_DIR}/${file})`

Definition at line 11 of file CMakeLists.txt.

23.52 src/mlpack/methods/nca/CMakeLists.txt File Reference

Functions

- `set (SOURCES nca.hpp nca_impl.hpp nca_softmax_error_function.hpp nca_softmax_error_function_impl.hpp)`
`set(DIR_SRCS) foreach(file $`
- `set (DIR_SRCS ${DIR_SRCS}${CMAKE_CURRENT_SOURCE_DIR}/${file}) endforeach() set(MLPACK_SRCS`
`$`

23.52.1 Function Documentation

23.52.1.1 `set (SOURCES nca.hpp nca_impl.hpp nca_softmax_error_function.hpp nca_softmax_error_function_impl. hpp)`

Definition at line 3 of file CMakeLists.txt.

23.52.1.2 `set (DIR_SRCS ${DIR_SRCS}${CMAKE_CURRENT_SOURCE_DIR}/${file})`

Definition at line 13 of file CMakeLists.txt.

23.53 src/mlpack/methods/neighbor_search/CMakeLists.txt File Reference

Functions

- **set** (SOURCES neighbor_search.hpp neighbor_search_impl.hpp neighbor_search_rules.hpp neighbor_search_rules_impl.hpp neighbor_search_stat.hpp ns_traversal_info.hpp sort_policies/nearest_neighbor_sort.hpp sort_policies/nearest_neighbor_sort.cpp sort_policies/nearest_neighbor_sort_impl.hpp sort_policies/furthest_neighbor_sort.hpp sort_policies/furthest_neighbor_sort.cpp sort_policies/furthest_neighbor_sort_impl.hpp typedef.hpp unmap.hpp unmap.cpp) set(DIR_SRCS) foreach(file \$
- **set** (DIR_SRCS \${DIR_SRCS}\${CMAKE_CURRENT_SOURCE_DIR}/\${file}) endforeach() set(MLPACK_SRCS \$

23.53.1 Function Documentation

23.53.1.1 **set** (SOURCES neighbor_search.hpp neighbor_search_impl.hpp neighbor_search_rules.hpp neighbor_search_rules_impl.hpp neighbor_search_stat.hpp ns_traversal_info.hpp sort_policies/nearest_neighbor_sort.hpp sort_policies/nearest_neighbor_sort.cpp sort_policies/nearest_neighbor_sort_impl.hpp sort_policies/furthest_neighbor_sort.hpp sort_policies/furthest_neighbor_sort.cpp sort_policies/furthest_neighbor_sort_impl.hpp typedef.hpp unmap.hpp unmap.cpp)

Definition at line 3 of file CMakeLists.txt.

23.53.1.2 **set** (DIR_SRCS \${DIR_SRCS}\${CMAKE_CURRENT_SOURCE_DIR}/\${file})

Definition at line 24 of file CMakeLists.txt.

23.54 src/mlpack/methods/nmf/CMakeLists.txt File Reference

23.55 src/mlpack/methods/nystroem_method/CMakeLists.txt File Reference

Functions

- **set** (SOURCES nystroem_method.hpp nystroem_method_impl.hpp ordered_selection.hpp random_selection.hpp kmeans_selection.hpp) set(DIR_SRCS) foreach(file \$
- **set** (DIR_SRCS \${DIR_SRCS}\${CMAKE_CURRENT_SOURCE_DIR}/\${file}) endforeach() set(MLPACK_SRCS \$

23.55.1 Function Documentation

23.55.1.1 **set** (SOURCES nystroem_method.hpp nystroem_method_impl.hpp ordered_selection.hpp random_selection.hpp kmeans_selection.hpp)

Definition at line 3 of file CMakeLists.txt.

23.55.1.2 **set** (DIR_SRCS \${DIR_SRCS}\${CMAKE_CURRENT_SOURCE_DIR}/\${file})

Definition at line 14 of file CMakeLists.txt.

23.56 src/mlpack/methods/pca/CMakeLists.txt File Reference

Functions

- **set** (SOURCES pca.hpp pca.cpp) set(DIR_SRCS) foreach(file \$
- **set** (DIR_SRCS \${DIR_SRCS}\${CMAKE_CURRENT_SOURCE_DIR}/\${file}) endforeach() set(MLPACK_SRCS \$

23.56.1 Function Documentation

23.56.1.1 **set** (SOURCES pca.hpp pca. *cpp*)

Definition at line 3 of file CMakeLists.txt.

23.56.1.2 **set** (DIR_SRCS \${DIR_SRCS}\${CMAKE_CURRENT_SOURCE_DIR}/\${file})

Definition at line 11 of file CMakeLists.txt.

23.57 src/mlpack/methods/perceptron/CMakeLists.txt File Reference

Functions

- **cmake_minimum_required** (VERSION 2.8) **set**(SOURCES perceptron.hpp perceptron_impl.hpp) **set**(DIR_S↵
RCS) foreach(file \$
- **set** (DIR_SRCS \${DIR_SRCS}\${CMAKE_CURRENT_SOURCE_DIR}/\${file}) endforeach() set(MLPACK_SRCS \$

23.57.1 Function Documentation

23.57.1.1 **cmake_minimum_required** (VERSION 2. 8)

Definition at line 1 of file CMakeLists.txt.

23.57.1.2 **set** (DIR_SRCS \${DIR_SRCS}\${CMAKE_CURRENT_SOURCE_DIR}/\${file})

Definition at line 13 of file CMakeLists.txt.

23.58 src/mlpack/methods/perceptron/initialization_methods/CMakeLists.txt File Reference

Functions

- **set** (SOURCES random_init.hpp zero_init.hpp) set(DIR_SRCS) foreach(file \$
- **set** (DIR_SRCS \${DIR_SRCS}\${CMAKE_CURRENT_SOURCE_DIR}/\${file}) endforeach() set(MLPACK_SRCS \$

23.58.1 Function Documentation

23.58.1.1 `set (SOURCES random_init.hpp zero_init. hpp)`

Definition at line 3 of file CMakeLists.txt.

23.58.1.2 `set (DIR_SRCS ${DIR_SRCS}${CMAKE_CURRENT_SOURCE_DIR}/${file})`

Definition at line 11 of file CMakeLists.txt.

23.59 src/mlpack/methods/perceptron/learning_policies/CMakeLists.txt File Reference

Functions

- `set (SOURCES simple_weight_update.hpp) set(DIR_SRCS) foreach(file $`
- `set (DIR_SRCS ${DIR_SRCS}${CMAKE_CURRENT_SOURCE_DIR}/${file}) endforeach() set(MLPACK_SRCS $`

23.59.1 Function Documentation

23.59.1.1 `set (SOURCES simple_weight_update. hpp)`

Definition at line 3 of file CMakeLists.txt.

23.59.1.2 `set (DIR_SRCS ${DIR_SRCS}${CMAKE_CURRENT_SOURCE_DIR}/${file})`

Definition at line 10 of file CMakeLists.txt.

23.60 src/mlpack/methods/quic_svd/CMakeLists.txt File Reference

Functions

- `set (SOURCES quic_svd.hpp quic_svd_impl.hpp) set(DIR_SRCS) foreach(file $`
- `set (DIR_SRCS ${DIR_SRCS}${CMAKE_CURRENT_SOURCE_DIR}/${file}) endforeach() set(MLPACK_SRCS $`

23.60.1 Function Documentation

23.60.1.1 `set (SOURCES quic_svd.hpp quic_svd_impl. hpp)`

Definition at line 3 of file CMakeLists.txt.

23.60.1.2 `set (DIR_SRCS ${DIR_SRCS}${CMAKE_CURRENT_SOURCE_DIR}/${file})`

Definition at line 11 of file CMakeLists.txt.

23.61 src/mlpack/methods/radical/CMakeLists.txt File Reference

Functions

- **set** (SOURCES radical.hpp radical.cpp) set(DIR_SRCS) foreach(file \$
- **set** (DIR_SRCS \${DIR_SRCS})\${CMAKE_CURRENT_SOURCE_DIR}/\${file}) endforeach() set(MLPACK_SRCS \$

23.61.1 Function Documentation

23.61.1.1 **set** (SOURCES radical.hpp radical. *cpp*)

Definition at line 3 of file CMakeLists.txt.

23.61.1.2 **set** (DIR_SRCS \${DIR_SRCS})\${CMAKE_CURRENT_SOURCE_DIR}/\${file})

Definition at line 11 of file CMakeLists.txt.

23.62 src/mlpack/methods/range_search/CMakeLists.txt File Reference

Functions

- **set** (SOURCES range_search.hpp range_search_impl.hpp range_search_rules.hpp range_search_rules_impl.↵
hpp range_search_stat.hpp) set(DIR_SRCS) foreach(file \$
- **set** (DIR_SRCS \${DIR_SRCS})\${CMAKE_CURRENT_SOURCE_DIR}/\${file}) endforeach() set(MLPACK_SRCS \$

23.62.1 Function Documentation

23.62.1.1 **set** (SOURCES range_search.hpp range_search_impl.hpp range_search_rules.hpp range_search_rules_impl.hpp
range_search_stat. *hpp*)

Definition at line 3 of file CMakeLists.txt.

23.62.1.2 **set** (DIR_SRCS \${DIR_SRCS})\${CMAKE_CURRENT_SOURCE_DIR}/\${file})

Definition at line 14 of file CMakeLists.txt.

23.63 src/mlpack/methods/rann/CMakeLists.txt File Reference

Functions

- **set** (SOURCES ra_search.hpp ra_search_impl.hpp ra_search_rules.hpp ra_search_rules_impl.hpp ra_query_↵
stat.hpp ra_typedef.hpp) set(DIR_SRCS) foreach(file \$
- **set** (DIR_SRCS \${DIR_SRCS})\${CMAKE_CURRENT_SOURCE_DIR}/\${file}) endforeach() set(MLPACK_CON↵
TRIB_SRCS \$

23.63.1 Function Documentation

23.63.1.1 `set (SOURCES ra_search.hpp ra_search_impl.hpp ra_search_rules.hpp ra_search_rules_impl.hpp ra_query_stat.hpp
ra_typedef. hpp)`

Definition at line 4 of file CMakeLists.txt.

23.63.1.2 `set (DIR_SRCS ${DIR_SRCS}${CMAKE_CURRENT_SOURCE_DIR}/${file})`

Definition at line 23 of file CMakeLists.txt.

23.64 src/mlpack/methods/regularized_svd/CMakeLists.txt File Reference

Functions

- `set (SOURCES regularized_svd.hpp regularized_svd_impl.hpp regularized_svd_function.hpp regularized_svd_↵
_function.cpp) set(DIR_SRCS) foreach(file $`
- `set (DIR_SRCS ${DIR_SRCS}${CMAKE_CURRENT_SOURCE_DIR}/${file}) endforeach() set(MLPACK_SRCS
$`

23.64.1 Function Documentation

23.64.1.1 `set (SOURCES regularized_svd.hpp regularized_svd_impl.hpp regularized_svd_function.hpp regularized_svd_function.
_function. cpp)`

Definition at line 3 of file CMakeLists.txt.

23.64.1.2 `set (DIR_SRCS ${DIR_SRCS}${CMAKE_CURRENT_SOURCE_DIR}/${file})`

Definition at line 13 of file CMakeLists.txt.

23.65 src/mlpack/methods/sparse_autoencoder/CMakeLists.txt File Reference

Functions

- `set (SOURCES sparse_autoencoder.hpp sparse_autoencoder_impl.hpp sparse_autoencoder_function.hpp
sparse_autoencoder_function.cpp) set(DIR_SRCS) foreach(file $`
- `set (DIR_SRCS ${DIR_SRCS}${CMAKE_CURRENT_SOURCE_DIR}/${file}) endforeach() set(MLPACK_SRCS
$`

23.65.1 Function Documentation

23.65.1.1 `set (SOURCES sparse_autoencoder.hpp sparse_autoencoder_impl.hpp sparse_autoencoder_function.hpp
sparse_autoencoder_function. cpp)`

Definition at line 3 of file CMakeLists.txt.

23.65.1.2 `set (DIR_SRCS ${DIR_SRCS}${CMAKE_CURRENT_SOURCE_DIR}/${file})`

Definition at line 13 of file CMakeLists.txt.

23.66 src/mlpack/methods/sparse_coding/CMakeLists.txt File Reference

Functions

- **set** (SOURCES data_dependent_random_initializer.hpp nothing_initializer.hpp random_initializer.hpp sparse_coding.hpp sparse_coding_impl.hpp) set(DIR_SRCS) foreach(file \$
- **set** (DIR_SRCS \${DIR_SRCS}\${CMAKE_CURRENT_SOURCE_DIR}/\${file}) endforeach() set(MLPACK_SRCS \$

23.66.1 Function Documentation

23.66.1.1 `set (SOURCES data_dependent_random_initializer.hpp nothing_initializer.hpp random_initializer.hpp sparse_coding.hpp sparse_coding_impl. hpp)`

Definition at line 3 of file CMakeLists.txt.

23.66.1.2 `set (DIR_SRCS ${DIR_SRCS}${CMAKE_CURRENT_SOURCE_DIR}/${file})`

Definition at line 14 of file CMakeLists.txt.

23.67 src/mlpack/tests/CMakeLists.txt File Reference

Functions

- **add_custom_command** (TARGET mlpack_test POST_BUILD COMMAND \${CMAKE_COMMAND}-E copy_directory \${CMAKE_CURRENT_SOURCE_DIR}/data/\${PROJECT_BINARY_DIR}) add_custom_command(TARGET mlpack_test POST_BUILD COMMAND \$
- **add_executable** (mlpack_test mlpack_test.cpp allkfn_test.cpp allknn_test.cpp allkrann_search_test.cpp arma_extend_test.cpp aug_lagrangian_test.cpp cf_test.cpp cli_test.cpp cosine_tree_test.cpp decision_stump_test.cpp det_test.cpp distribution_test.cpp emst_test.cpp fastmks_test.cpp gmm_test.cpp hmm_test.cpp kernel_test.cpp kernel_pca_test.cpp kernel_traits_test.cpp kmeans_test.cpp lars_test.cpp lbfgs_test.cpp lin_alg_test.cpp linear_regression_test.cpp load_save_test.cpp local_coordinate_coding_test.cpp logistic_regression_test.cpp lrscp_test.cpp lsh_test.cpp math_test.cpp metric_test.cpp nbc_test.cpp nca_test.cpp nmf_test.cpp pca_test.cpp perceptron_test.cpp quic_svd_test.cpp radical_test.cpp range_search_test.cpp regularized_svd_test.cpp sa_test.cpp save_restore_utility_test.cpp sgd_test.cpp sort_policy_test.cpp sparse_autoencoder_test.cpp sparse_coding_test.cpp to_string_test.cpp tree_test.cpp tree_traits_test.cpp union_find_test.cpp svd_batch_test.cpp svd_incremental_test.cpp nystroem_method_test.cpp) target_link_libraries(mlpack_test mlpack \$

23.67.1 Function Documentation

23.67.1.1 `add_custom_command (TARGET mlpack_test POST_BUILD COMMAND ${CMAKE_COMMAND}-E copy_directory ${CMAKE_CURRENT_SOURCE_DIR}/data/${PROJECT_BINARY_DIR})`

Definition at line 65 of file CMakeLists.txt.

23.67.1.2 `add_executable (mlpack_test mlpack_test.cpp allkfn_test.cpp allknn_test.cpp allkrann_search_test.cpp arma_extend_test.cpp aug_lagrangian_test.cpp cf_test.cpp cli_test.cpp cosine_tree_test.cpp decision_stump_test.cpp det_test.cpp distribution_test.cpp emst_test.cpp fastmks_test.cpp gmm_test.cpp hmm_test.cpp kernel_test.cpp kernel_pca_test.cpp kernel_traits_test.cpp kmeans_test.cpp lars_test.cpp lbfgs_test.cpp lin_alg_test.cpp linear_regression_test.cpp load_save_test.cpp local_coordinate_coding_test.cpp logistic_regression_test.cpp lrmdp_test.cpp lsh_test.cpp math_test.cpp metric_test.cpp nbc_test.cpp nca_test.cpp nmf_test.cpp pca_test.cpp perceptron_test.cpp quic_svd_test.cpp radical_test.cpp range_search_test.cpp regularized_svd_test.cpp sa_test.cpp save_restore_utility_test.cpp sgdp_test.cpp sort_policy_test.cpp sparse_autoencoder_test.cpp sparse_coding_test.cpp to_string_test.cpp tree_test.cpp tree_traits_test.cpp union_find_test.cpp svd_batch_test.cpp svd_incremental_test.cpp nystroem_method_test.cpp)`

Definition at line 2 of file CMakeLists.txt.

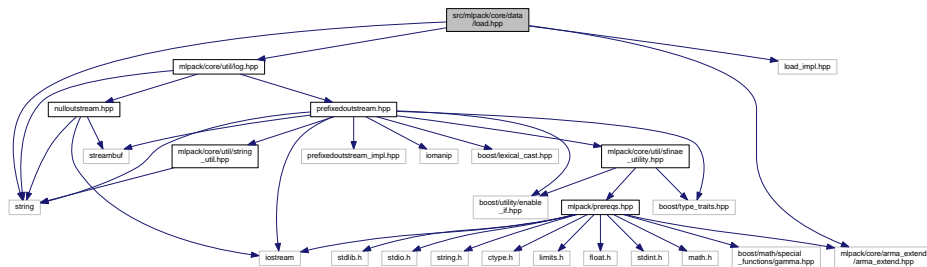
23.68 src/mlpack/core.hpp File Reference

Include dependency graph for core.hpp:



23.69 src/mlpack/core/data/load.hpp File Reference

Include dependency graph for load.hpp:



This graph shows which files directly or indirectly include this file:



Namespaces

- **mlpack**
Linear algebra utility functions, generally performed on matrices or vectors.
- **mlpack::data**
Functions to load and save matrices.

Functions

- `template<typename eT >`
`bool mlpack::data::Load (const std::string &filename, arma::Mat< eT > &matrix, bool fatal=false, bool transpose=true)`

Loads a matrix from file, guessing the filetype from the extension.

23.69.1 Detailed Description

Author

Ryan Curtin

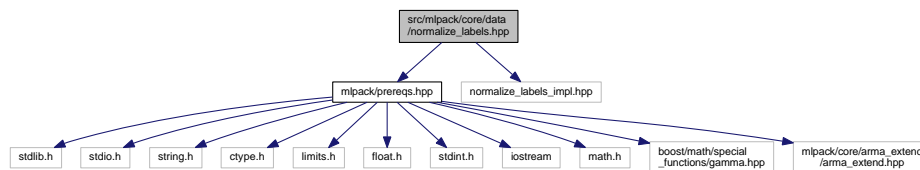
Load an Armadillo matrix from file. This is necessary because Armadillo does not transpose matrices on input, and it allows us to give better error output.

This file is part of mlpack 1.0.12.

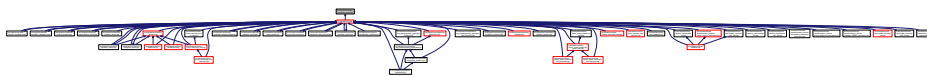
mlpack is free software; you may redistribute it and/or modify it under the terms of the 3-clause BSD license. You should have received a copy of the 3-clause BSD license along with mlpack. If not, see <http://www.opensource.org/licenses/BSD-3-Clause> for more information.

23.70 src/mlpack/core/data/normalize_labels.hpp File Reference

Include dependency graph for `normalize_labels.hpp`:



This graph shows which files directly or indirectly include this file:



Namespaces

- **mlpack**
Linear algebra utility functions, generally performed on matrices or vectors.
- **mlpack::data**
Functions to load and save matrices.

Functions

- `template<typename eT >`
`void mlpack::data::NormalizeLabels (const arma::Col< eT > &labelsIn, arma::Col< size_t > &labels, arma::Col< eT > &mapping)`

- `template<typename eT >`
`void mlpack::data::RevertLabels (const arma::Col< size_t > &labels, const arma::Col< eT > &mapping,`
`arma::Col< eT > &labelsOut)`

23.70.1 Detailed Description

Ryan Curtin

This file is part of mlpack 1.0.12.

23.71 src/mlpack/core/data/save.hpp File Reference

- **mlpack**
Linear algebra utility functions, generally performed on matrices or vectors.
- **mlpack::data**
Functions to load and save matrices.

Functions

- `template<typename eT >`
`bool mlpack::data::Save (const std::string &filename, const arma::Mat< eT > &matrix, bool fatal=false, bool transpose=true)`

Saves a matrix to file, guessing the filetype from the extension.

23.71.1 Detailed Description

Author

Ryan Curtin

Save an Armadillo matrix to file. This is necessary because Armadillo does not transpose matrices upon saving, and it allows us to give better error output.

This file is part of mlpack 1.0.12.

mlpack is free software; you may redistribute it and/or modify it under the terms of the 3-clause BSD license. You should have received a copy of the 3-clause BSD license along with mlpack. If not, see <http://www.opensource.org/licenses/BSD-3-Clause> for more information.

23.72 src/mlpack/core/dists/discrete_distribution.hpp File Reference

Include dependency graph for `discrete_distribution.hpp`:



This graph shows which files directly or indirectly include this file:



Classes

- class **mlpack::distribution::DiscreteDistribution**
A discrete distribution where the only observations are discrete observations.

Namespaces

- **mlpack**
Linear algebra utility functions, generally performed on matrices or vectors.
- **mlpack::distribution**
Probability distributions.

23.72.1 Detailed Description

Author

Ryan Curtin

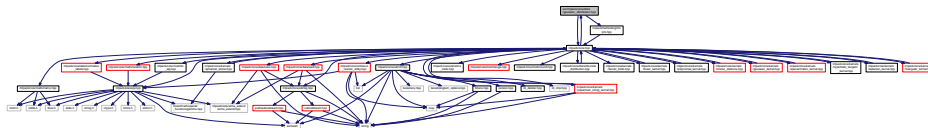
Implementation of the discrete distribution, where each discrete observation has a given probability.

This file is part of mlpack 1.0.12.

mlpack is free software; you may redistribute it and/or modify it under the terms of the 3-clause BSD license. You should have received a copy of the 3-clause BSD license along with mlpack. If not, see <http://www.opensource.org/licenses/BSD-3-Clause> for more information.

23.73 src/mlpack/core/dists/gaussian_distribution.hpp File Reference

Include dependency graph for gaussian_distribution.hpp:



This graph shows which files directly or indirectly include this file:



Classes

- class **mlpack::distribution::GaussianDistribution**

A single multivariate Gaussian distribution.

Namespaces

- **mlpack**

Linear algebra utility functions, generally performed on matrices or vectors.

- **mlpack::distribution**

Probability distributions.

23.73.1 Detailed Description

Author

Ryan Curtin

Implementation of the Gaussian distribution.

This file is part of mlpack 1.0.12.

mlpack is free software; you may redistribute it and/or modify it under the terms of the 3-clause BSD license. You should have received a copy of the 3-clause BSD license along with mlpack. If not, see <http://www.opensource.org/licenses/BSD-3-Clause> for more information.

23.74 src/mlpack/core/dists/laplace_distribution.hpp File Reference

Classes

- class **mlpack::distribution::LaplaceDistribution**
The multivariate Laplace distribution centered at 0 has pdf.

Namespaces

- **mlpack**
Linear algebra utility functions, generally performed on matrices or vectors.
- **mlpack::distribution**
Probability distributions.

23.75 src/mlpack/core/kernels/cosine_distance.hpp File Reference

Include dependency graph for cosine_distance.hpp:



This graph shows which files directly or indirectly include this file:



Classes

- class **mlpack::kernel::CosineDistance**
The cosine distance (or cosine similarity).
- class **mlpack::kernel::KernelTraits< CosineDistance >**
Kernel traits for the cosine distance.

Namespaces

- **mlpack**
Linear algebra utility functions, generally performed on matrices or vectors.
- **mlpack::kernel**
Kernel functions.

23.75.1 Detailed Description

Author

Ryan Curtin

This implements the cosine distance (or cosine similarity) between two vectors, which is a measure of the angle between the two vectors.

This file is part of mlpack 1.0.12.

mlpack is free software; you may redistribute it and/or modify it under the terms of the 3-clause BSD license. You should have received a copy of the 3-clause BSD license along with mlpack. If not, see <http://www.opensource.org/licenses/BSD-3-Clause> for more information.

23.76 src/mlpack/core/kernels/epanechnikov_kernel.hpp File Reference

Include dependency graph for epanechnikov_kernel.hpp:



This graph shows which files directly or indirectly include this file:



Classes

- class **mlpack::kernel::EpanechnikovKernel**
The Epanechnikov kernel, defined as.
- class **mlpack::kernel::KernelTraits<EpanechnikovKernel>**
Kernel traits for the Epanechnikov kernel.

Namespaces

- **mlpack**
Linear algebra utility functions, generally performed on matrices or vectors.
- **mlpack::kernel**
Kernel functions.

23.76.1 Detailed Description

Author

Neil Slagle

Definition of the Epanechnikov kernel.

This file is part of mlpack 1.0.12.

mlpack is free software; you may redistribute it and/or modify it under the terms of the 3-clause BSD license. You should have received a copy of the 3-clause BSD license along with mlpack. If not, see <http://www.opensource.org/licenses/BSD-3-Clause> for more information.

23.77 src/mlpack/core/kernels/example_kernel.hpp File Reference

Include dependency graph for example_kernel.hpp:



Classes

- class **mlpack::kernel::ExampleKernel**

An example kernel function.

Namespaces

- **mlpack**

Linear algebra utility functions, generally performed on matrices or vectors.

- **mlpack::kernel**

Kernel functions.

23.77.1 Detailed Description

Author

Ryan Curtin

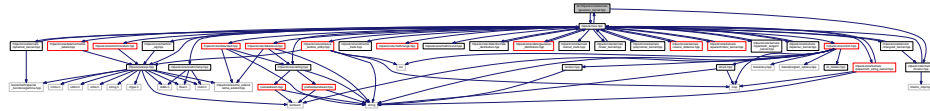
This is an example kernel. If you are making your own kernel, follow the outline specified in this file.

This file is part of mlpack 1.0.12.

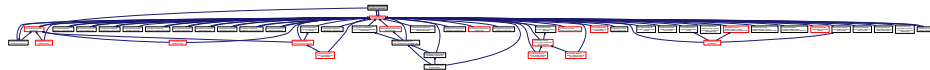
mlpack is free software; you may redistribute it and/or modify it under the terms of the 3-clause BSD license. You should have received a copy of the 3-clause BSD license along with mlpack. If not, see <http://www.opensource.org/licenses/BSD-3-Clause> for more information.

23.78 src/mlpack/core/kernels/gaussian_kernel.hpp File Reference

Include dependency graph for gaussian_kernel.hpp:



This graph shows which files directly or indirectly include this file:



Classes

- class **mlpack::kernel::GaussianKernel**
The standard Gaussian kernel.
- class **mlpack::kernel::KernelTraits**< **GaussianKernel** >
Kernel traits for the Gaussian kernel.

Namespaces

- **mlpack**
Linear algebra utility functions, generally performed on matrices or vectors.
- **mlpack::kernel**
Kernel functions.

23.78.1 Detailed Description

Author

Wei Guan
James Cline
Ryan Curtin

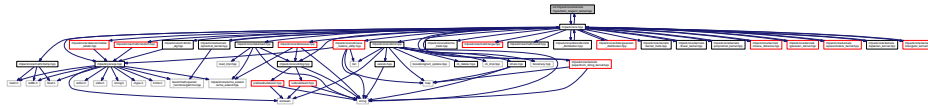
Implementation of the Gaussian kernel (`GaussianKernel`).

This file is part of mlpack 1.0.12.

mlpack is free software; you may redistribute it and/or modify it under the terms of the 3-clause BSD license. You should have received a copy of the 3-clause BSD license along with mlpack. If not, see <http://www.opensource.org/licenses/BSD-3-Clause> for more information.

23.79 src/mlpack/core/kernels/hyperbolic_tangent_kernel.hpp File Reference

Include dependency graph for hyperbolic_tangent_kernel.hpp:



This graph shows which files directly or indirectly include this file:



Classes

- class **mlpack::kernel::HyperbolicTangentKernel**
Hyperbolic tangent kernel.

Namespaces

- **mlpack**
Linear algebra utility functions, generally performed on matrices or vectors.
- **mlpack::kernel**
Kernel functions.

23.79.1 Detailed Description

Author

Ajinkya Kale kaleajinkya@gmail.com

Implementation of the hyperbolic tangent kernel.

This file is part of mlpack 1.0.12.

mlpack is free software; you may redistribute it and/or modify it under the terms of the 3-clause BSD license. You should have received a copy of the 3-clause BSD license along with mlpack. If not, see <http://www.opensource.org/licenses/BSD-3-Clause> for more information.

23.80 src/mlpack/core/kernels/kernel_traits.hpp File Reference

This graph shows which files directly or indirectly include this file:



Classes

- class **mlpack::kernel::KernelTraits**< **KernelType** >

This is a template class that can provide information about various kernels.

Namespaces

- **mlpack**

Linear algebra utility functions, generally performed on matrices or vectors.

- **mlpack::kernel**

Kernel functions.

23.80.1 Detailed Description

Author

Ryan Curtin

This provides the KernelTraits class, a template class to get information about various kernels.

This file is part of mlpack 1.0.12.

mlpack is free software; you may redistribute it and/or modify it under the terms of the 3-clause BSD license. You should have received a copy of the 3-clause BSD license along with mlpack. If not, see <http://www.opensource.org/licenses/BSD-3-Clause> for more information.

23.81 src/mlpack/core/kernels/laplacian_kernel.hpp File Reference

Include dependency graph for laplacian_kernel.hpp:



This graph shows which files directly or indirectly include this file:



Classes

- class **mlpack::kernel::KernelTraits**< **LaplacianKernel** >

Kernel traits of the Laplacian kernel.

- class **mlpack::kernel::LaplacianKernel**

The standard Laplacian kernel.

Namespaces

- **mlpack**

Linear algebra utility functions, generally performed on matrices or vectors.

- **mlpack::kernel**

Kernel functions.

23.81.1 Detailed Description

Author

Ajinkya Kale kaleajinkya@gmail.com

Implementation of the Laplacian kernel (LaplacianKernel).

This file is part of mlpack 1.0.12.

mlpack is free software; you may redistribute it and/or modify it under the terms of the 3-clause BSD license. You should have received a copy of the 3-clause BSD license along with mlpack. If not, see <http://www.opensource.org/licenses/BSD-3-Clause> for more information.

23.82 src/mlpack/core/kernels/linear_kernel.hpp File Reference

Include dependency graph for linear_kernel.hpp:



This graph shows which files directly or indirectly include this file:



Classes

- class **mlpack::kernel::LinearKernel**

The simple linear kernel (dot product).

Namespaces

- **mlpack**

Linear algebra utility functions, generally performed on matrices or vectors.

- **mlpack::kernel**

Kernel functions.

23.82.1 Detailed Description

Author

Wei Guan
James Cline
Ryan Curtin

Implementation of the linear kernel (just the standard dot product).

This file is part of mlpack 1.0.12.

mlpack is free software; you may redistribute it and/or modify it under the terms of the 3-clause BSD license. You should have received a copy of the 3-clause BSD license along with mlpack. If not, see <http://www.opensource.org/licenses/BSD-3-Clause> for more information.

23.83 src/mlpack/core/kernels/polynomial_kernel.hpp File Reference

Include dependency graph for polynomial_kernel.hpp:



This graph shows which files directly or indirectly include this file:



Classes

- class **mlpack::kernel::PolynomialKernel**

The simple polynomial kernel.

Namespaces

- **mlpack**

Linear algebra utility functions, generally performed on matrices or vectors.

- **mlpack::kernel**

Kernel functions.

23.83.1 Detailed Description

Author

Ajinkya Kale kaleajinkya@gmail.com

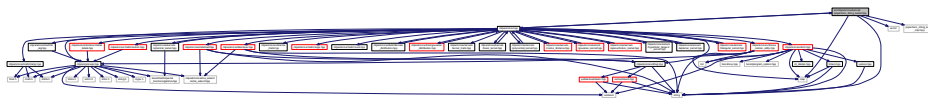
Implementation of the polynomial kernel (just the standard dot product).

This file is part of mlpack 1.0.12.

mlpack is free software; you may redistribute it and/or modify it under the terms of the 3-clause BSD license. You should have received a copy of the 3-clause BSD license along with mlpack. If not, see <http://www.opensource.org/licenses/BSD-3-Clause> for more information.

23.84 src/mlpack/core/kernels/pspectrum_string_kernel.hpp File Reference

Include dependency graph for pspectrum_string_kernel.hpp:



This graph shows which files directly or indirectly include this file:

**Classes**

- class **mlpack::kernel::PSpectrumStringKernel**

The p-spectrum string kernel.

Namespaces

- **mlpack**

Linear algebra utility functions, generally performed on matrices or vectors.

- **mlpack::kernel**

Kernel functions.

23.84.1 Detailed Description

Author

Ryan Curtin

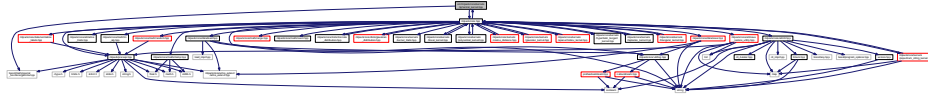
Implementation of the p-spectrum string kernel, created for use with FastMKS. Instead of passing a data matrix to FastMKS which stores the kernels, pass a one-dimensional data matrix (data vector) to FastMKS which stores indices of strings; then, the actual strings are given to the PSpectrumStringKernel at construction time, and the kernel knows to map the indices to actual strings.

This file is part of mlpack 1.0.12.

mlpack is free software; you may redistribute it and/or modify it under the terms of the 3-clause BSD license. You should have received a copy of the 3-clause BSD license along with mlpack. If not, see <http://www.opensource.org/licenses/BSD-3-Clause> for more information.

23.85 src/mlpack/core/kernels/spherical_kernel.hpp File Reference

Include dependency graph for spherical_kernel.hpp:



This graph shows which files directly or indirectly include this file:



Classes

- class **mlpack::kernel::KernelTraits< SphericalKernel >**
Kernel traits for the spherical kernel.
- class **mlpack::kernel::SphericalKernel**

Namespaces

- **mlpack**
Linear algebra utility functions, generally performed on matrices or vectors.
- **mlpack::kernel**
Kernel functions.

23.85.1 Detailed Description

Author

Neil Slagle

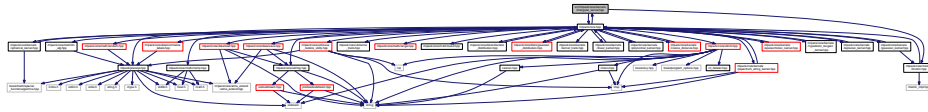
This is an example kernel. If you are making your own kernel, follow the outline specified in this file.

This file is part of mlpack 1.0.12.

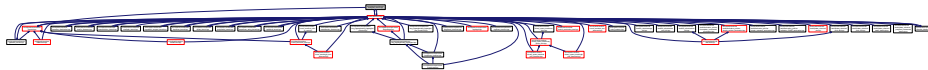
mlpack is free software; you may redistribute it and/or modify it under the terms of the 3-clause BSD license. You should have received a copy of the 3-clause BSD license along with mlpack. If not, see <http://www.opensource.org/licenses/BSD-3-Clause> for more information.

23.86 src/mlpack/core/kernels/triangular_kernel.hpp File Reference

Include dependency graph for triangular_kernel.hpp:



This graph shows which files directly or indirectly include this file:



Classes

- class **mlpack::kernel::KernelTraits**< **TriangularKernel** >
Kernel traits for the triangular kernel.
- class **mlpack::kernel::TriangularKernel**
The trivially simple triangular kernel, defined by.

Namespaces

- **mlpack**
Linear algebra utility functions, generally performed on matrices or vectors.
- **mlpack::kernel**
Kernel functions.

23.86.1 Detailed Description

Author

Ryan Curtin

Definition and implementation of the trivially simple triangular kernel.

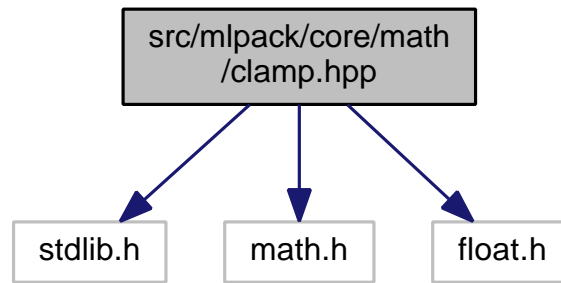
This file is part of mlpack 1.0.12.

mlpack is free software; you may redistribute it and/or modify it under the terms of the 3-clause BSD license. You should have received a copy of the 3-clause BSD license along with mlpack. If not, see <http://www.opensource.org/licenses/BSD-3-Clause> for more information.

23.87 src/mlpack/core/math/clamp.hpp File Reference

Miscellaneous math clamping routines.

Include dependency graph for clamp.hpp:



This graph shows which files directly or indirectly include this file:



Namespaces

- **mlpack**

Linear algebra utility functions, generally performed on matrices or vectors.

- **mlpack::math**

Miscellaneous math routines.

Functions

- double **mlpack::math::ClampNonNegative** (const double d)

Forces a number to be non-negative, turning negative numbers into zero.

- double **mlpack::math::ClampNonPositive** (const double d)

Forces a number to be non-positive, turning positive numbers into zero.

- double **mlpack::math::ClampRange** (double value, const double rangeMin, const double rangeMax)

Clamp a number between a particular range.

23.87.1 Detailed Description

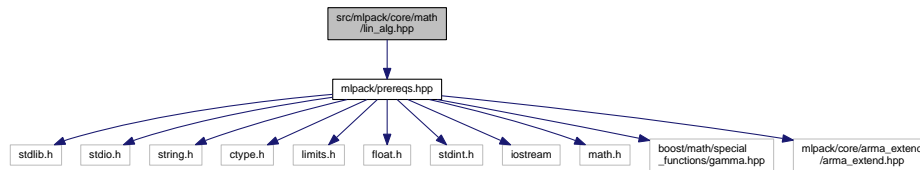
Miscellaneous math clamping routines.

This file is part of mlpack 1.0.12.

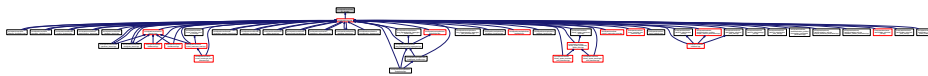
mlpack is free software; you may redistribute it and/or modify it under the terms of the 3-clause BSD license. You should have received a copy of the 3-clause BSD license along with mlpack. If not, see <http://www.opensource.org/licenses/BSD-3-Clause> for more information.

23.88 src/mlpack/core/math/lin_alg.hpp File Reference

Include dependency graph for lin_alg.hpp:



This graph shows which files directly or indirectly include this file:



Namespaces

- **mlpack**
Linear algebra utility functions, generally performed on matrices or vectors.
- **mlpack::math**
Miscellaneous math routines.

Functions

- void **mlpack::math::Center** (const arma::mat &x, arma::mat &xCentered)
Creates a centered matrix, where centering is done by subtracting the sum over the columns (a column vector) from each column of the matrix.
- void **mlpack::math::Orthogonalize** (const arma::mat &x, arma::mat &W)
Orthogonalize x and return the result in W, using eigendecomposition.
- void **mlpack::math::Orthogonalize** (arma::mat &x)
Orthogonalize x in-place.
- void **mlpack::math::RandVector** (arma::vec &v)
Overwrites a dimension-N vector to a random vector on the unit sphere in R^N .
- void **mlpack::math::RemoveRows** (const arma::mat &input, const std::vector< size_t > &rowsToRemove, arma::mat &output)
Remove a certain set of rows in a matrix while copying to a second matrix.
- void **mlpack::math::VectorPower** (arma::vec &vec, const double power)
Auxiliary function to raise vector elements to a specific power.
- void **mlpack::math::WhitenUsingEig** (const arma::mat &x, arma::mat &xWhitened, arma::mat &whitening↔ Matrix)
Whitens a matrix using the eigendecomposition of the covariance matrix.
- void **mlpack::math::WhitenUsingSVD** (const arma::mat &x, arma::mat &xWhitened, arma::mat &whitening↔ Matrix)
Whitens a matrix using the singular value decomposition of the covariance matrix.

23.88.1 Detailed Description

Author

Nishant Mehta

Linear algebra utilities.

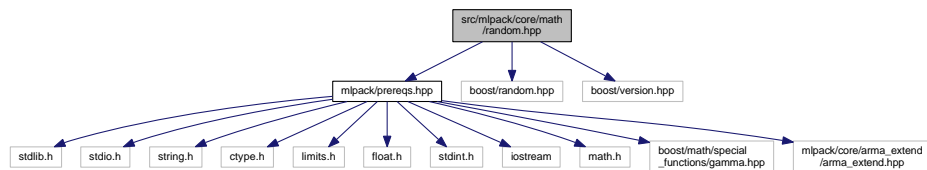
This file is part of mlpack 1.0.12.

mlpack is free software; you may redistribute it and/or modify it under the terms of the 3-clause BSD license. You should have received a copy of the 3-clause BSD license along with mlpack. If not, see <http://www.opensource.org/licenses/BSD-3-Clause> for more information.

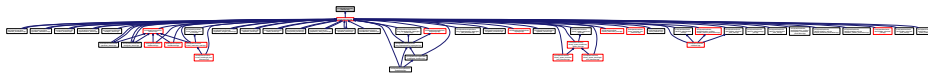
23.89 src/mlpack/core/math/random.hpp File Reference

Miscellaneous math random-related routines.

Include dependency graph for random.hpp:



This graph shows which files directly or indirectly include this file:



Namespaces

- **mlpack**
Linear algebra utility functions, generally performed on matrices or vectors.
- **mlpack::math**
Miscellaneous math routines.

Functions

- **int mlpack::math::RandInt** (const int hiExclusive)
Generates a uniform random integer.
- **int mlpack::math::RandInt** (const int lo, const int hiExclusive)
Generates a uniform random integer.
- **double mlpack::math::RandNormal** ()
Generates a normally distributed random number with mean 0 and variance 1.
- **double mlpack::math::RandNormal** (const double mean, const double variance)
Generates a normally distributed random number with specified mean and variance.

- double **mlpack::math::Random** ()
Generates a uniform random number between 0 and 1.
- double **mlpack::math::Random** (const double lo, const double hi)
Generates a uniform random number in the specified range.
- void **mlpack::math::RandomSeed** (const size_t seed)
*Set the random seed used by the random functions (**Random()** (p. 107) and **RandInt()** (p. 106)).*

Variables

- boost::mt19937 **mlpack::math::randGen**
- boost::normal_distribution **mlpack::math::randNormalDist**
- boost::uniform_01< boost::mt19937, double > **mlpack::math::randUniformDist**

23.89.1 Detailed Description

Miscellaneous math random-related routines.

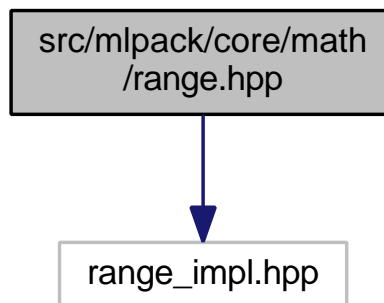
This file is part of mlpack 1.0.12.

mlpack is free software; you may redistribute it and/or modify it under the terms of the 3-clause BSD license. You should have received a copy of the 3-clause BSD license along with mlpack. If not, see <http://www.opensource.org/licenses/BSD-3-Clause> for more information.

23.90 src/mlpack/core/math/range.hpp File Reference

Definition of the Range class, which represents a simple range with a lower and upper bound.

Include dependency graph for range.hpp:



This graph shows which files directly or indirectly include this file:



Classes

- class **mlpack::math::Range**

Simple real-valued range.

Namespaces

- **mlpack**

Linear algebra utility functions, generally performed on matrices or vectors.

- **mlpack::math**

Miscellaneous math routines.

23.90.1 Detailed Description

Definition of the Range class, which represents a simple range with a lower and upper bound.

This file is part of mlpack 1.0.12.

mlpack is free software; you may redistribute it and/or modify it under the terms of the 3-clause BSD license. You should have received a copy of the 3-clause BSD license along with mlpack. If not, see <http://www.opensource.org/licenses/BSD-3-Clause> for more information.

23.91 src/mlpack/core/math/round.hpp File Reference

This graph shows which files directly or indirectly include this file:



23.91.1 Detailed Description

Author

Ryan Curtin

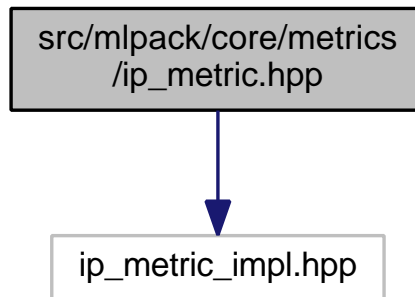
Implementation of round() for use on Visual Studio, where C99 isn't implemented.

This file is part of mlpack 1.0.12.

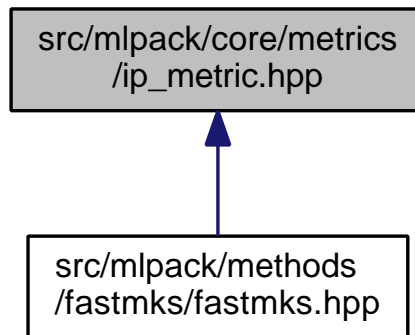
mlpack is free software; you may redistribute it and/or modify it under the terms of the 3-clause BSD license. You should have received a copy of the 3-clause BSD license along with mlpack. If not, see <http://www.opensource.org/licenses/BSD-3-Clause> for more information.

23.92 src/mlpack/core/metrics/ip_metric.hpp File Reference

Include dependency graph for ip_metric.hpp:



This graph shows which files directly or indirectly include this file:



Classes

- class `mlpack::metric::IPMetric< KernelType >`

Namespaces

- `mlpack`
Linear algebra utility functions, generally performed on matrices or vectors.
- `mlpack::metric`

23.92.1 Detailed Description

Author

Ryan Curtin

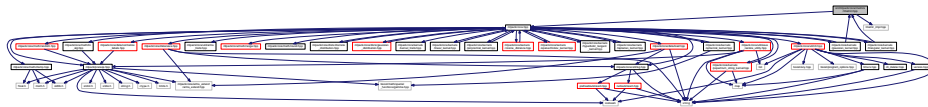
Inner product induced metric. If given a kernel function, this gives the complementary metric.

This file is part of mlpack 1.0.12.

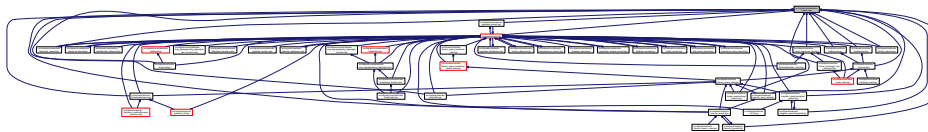
mlpack is free software; you may redistribute it and/or modify it under the terms of the 3-clause BSD license. You should have received a copy of the 3-clause BSD license along with mlpack. If not, see <http://www.opensource.org/licenses/BSD-3-Clause> for more information.

23.93 src/mlpack/core/metrics/lmetric.hpp File Reference

Include dependency graph for lmetric.hpp:



This graph shows which files directly or indirectly include this file:



Classes

- class **mlpack::metric::LMetric**< **Power**, **TakeRoot** >
The L_p metric for arbitrary integer p , with an option to take the root.

Namespaces

- **mlpack**
Linear algebra utility functions, generally performed on matrices or vectors.
- **mlpack::metric**

Typedefs

- typedef LMetric< INT_MAX, false > **mlpack::metric::ChebyshevDistance**
- typedef LMetric< 2, true > **mlpack::metric::EuclideanDistance**
- typedef LMetric< 1, false > **mlpack::metric::ManhattanDistance**
- typedef LMetric< 2, false > **mlpack::metric::SquaredEuclideanDistance**

23.93.1 Detailed Description

Author

Ryan Curtin

Generalized L-metric, allowing both squared distances to be returned as well as non-squared distances. The squared distances are faster to compute.

This also gives several convenience typedefs for commonly used L-metrics.

This file is part of mlpack 1.0.12.

mlpack is free software; you may redistribute it and/or modify it under the terms of the 3-clause BSD license. You should have received a copy of the 3-clause BSD license along with mlpack. If not, see <http://www.opensource.org/licenses/BSD-3-Clause> for more information.

23.94 src/mlpack/core/metrics/mahalanobis_distance.hpp File Reference

Include dependency graph for mahalanobis_distance.hpp:



Classes

- class `mlpack::metric::MahalanobisDistance< TakeRoot >`

The Mahalanobis distance, which is essentially a stretched Euclidean distance.

Namespaces

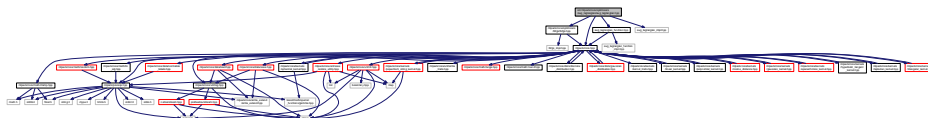
- `mlpack`

Linear algebra utility functions, generally performed on matrices or vectors.

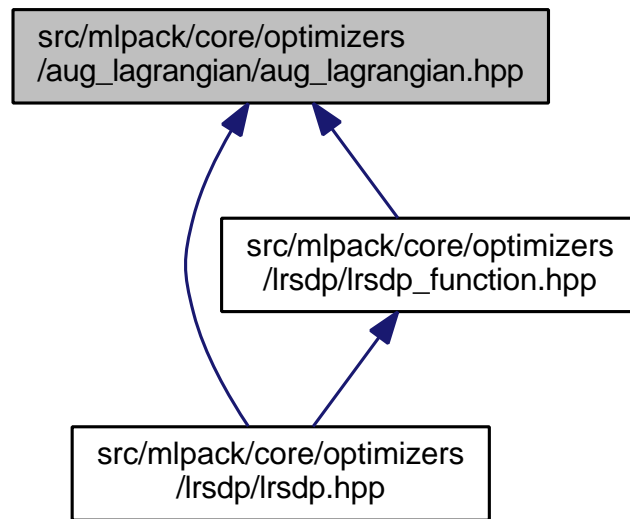
- `mlpack::metric`

23.95 src/mlpack/core/optimizers/aug_lagrangian/aug_lagrangian.hpp File Reference

Include dependency graph for aug_lagrangian.hpp:



This graph shows which files directly or indirectly include this file:



Classes

- class **mlpack::optimization::AugLagrangian**< **LagrangianFunction** >

The **AugLagrangian** (p. 441) class implements the Augmented Lagrangian method of optimization.

Namespaces

- **mlpack**

Linear algebra utility functions, generally performed on matrices or vectors.

- **mlpack::optimization**

23.95.1 Detailed Description

Author

Ryan Curtin

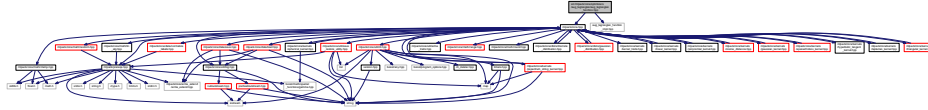
Definition of AugLagrangian class, which implements the Augmented Lagrangian optimization method (also called the 'method of multipliers'. This class uses the L-BFGS optimizer.

This file is part of mlpack 1.0.12.

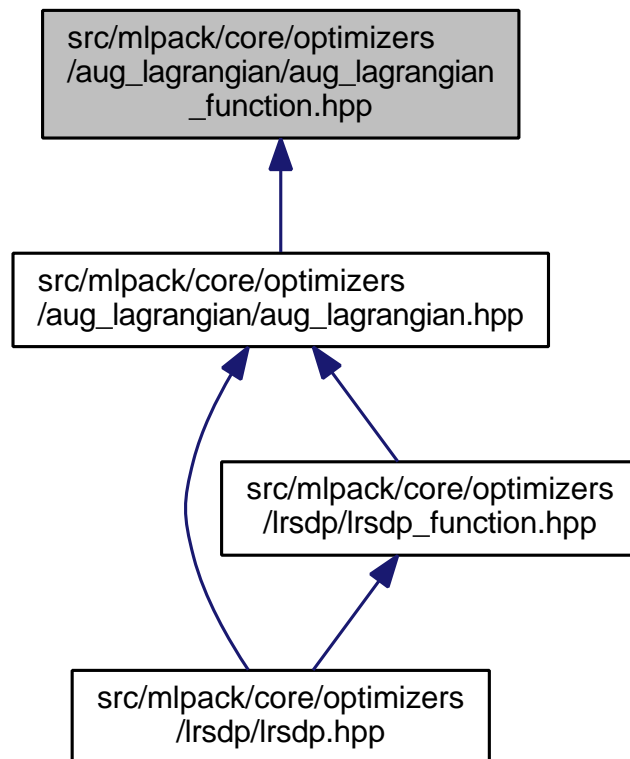
mlpack is free software; you may redistribute it and/or modify it under the terms of the 3-clause BSD license. You should have received a copy of the 3-clause BSD license along with mlpack. If not, see <http://www.opensource.org/licenses/BSD-3-Clause> for more information.

23.96 src/mlpack/core/optimizers/aug_lagrangian/aug_lagrangian_function.hpp File Reference

Include dependency graph for aug_lagrangian_function.hpp:



This graph shows which files directly or indirectly include this file:



Classes

- class **mlpack::optimization::AugLagrangianFunction**< **LagrangianFunction** >

*This is a utility class used by **AugLagrangian** (p. 441), meant to wrap a **LagrangianFunction** into a function usable by a simple optimizer like L-BFGS.*

Namespaces

- **mlpack**

Linear algebra utility functions, generally performed on matrices or vectors.

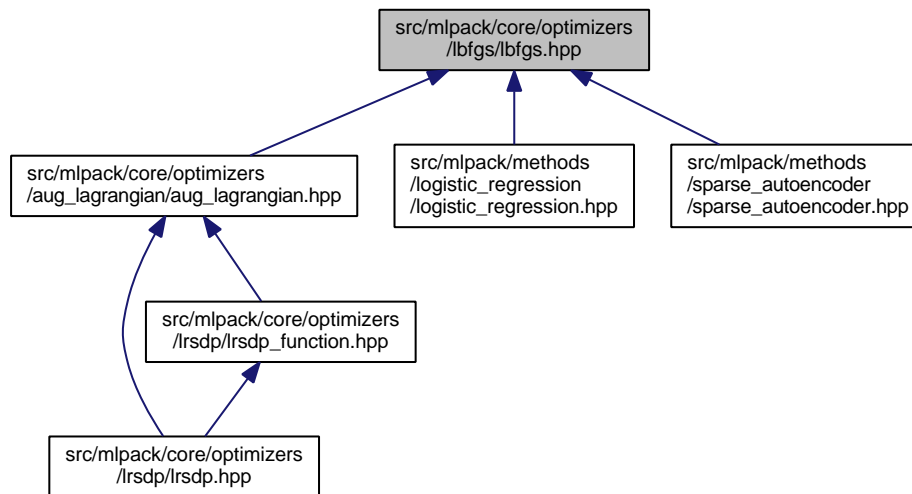
- **mlpack::optimization**

23.98 src/mlpack/core/optimizers/lbfgs/lbfgs.hpp File Reference

Include dependency graph for lbfgs.hpp:



This graph shows which files directly or indirectly include this file:



Classes

- class **mlpack::optimization::L_BFGS**< **FunctionType** >

The generic L-BFGS optimizer, which uses a back-tracking line search algorithm to minimize a function.

Namespaces

- **mlpack**

Linear algebra utility functions, generally performed on matrices or vectors.

- **mlpack::optimization**

23.98.1 Detailed Description

Author

Dongryeol Lee
Ryan Curtin

The generic L-BFGS optimizer.

This file is part of mlpack 1.0.12.

mlpack is free software; you may redistribute it and/or modify it under the terms of the 3-clause BSD license. You should have received a copy of the 3-clause BSD license along with mlpack. If not, see <http://www.opensource.org/licenses/BSD-3-Clause> for more information.

23.99 src/mlpack/core/optimizers/lbfgs/test_functions.hpp File Reference

Include dependency graph for test_functions.hpp:



Classes

- class **mlpack::optimization::test::GeneralizedRosenbrockFunction**
*The Generalized Rosenbrock function in n dimensions, defined by $f(x) = \sum_{i=1}^{n-1} (f(i)(x))$ $f_i(x) = 100 * (x_i^2 - x_{i+1})^2 + (1 - x_{i+1})^2$ $x_0 = [-1.2, 1, -1.2, 1, \dots]$.*
- class **mlpack::optimization::test::RosenbrockFunction**
The Rosenbrock function, defined by $f(x) = f_1(x) + f_2(x)$ $f_1(x) = 100 (x_2 - x_1^2)^2$ $f_2(x) = (1 - x_1)^2$ $x_0 = [-1.2, 1]$.
- class **mlpack::optimization::test::RosenbrockWoodFunction**
The Generalized Rosenbrock function in 4 dimensions with the Wood Function in four dimensions.
- class **mlpack::optimization::test::WoodFunction**
The Wood function, defined by $f(x) = f_1(x) + f_2(x) + f_3(x) + f_4(x) + f_5(x) + f_6(x)$ $f_1(x) = 100 (x_2 - x_1^2)^2$ $f_2(x) = (1 - x_1)^2$ $f_3(x) = 90 (x_4 - x_3^2)^2$ $f_4(x) = (1 - x_3)^2$ $f_5(x) = 10 (x_2 + x_4 - 2)^2$ $f_6(x) = (1 / 10) (x_2 - x_4)^2$ $x_0 = [-3, -1, -3, -1]$.

Namespaces

- **mlpack**
Linear algebra utility functions, generally performed on matrices or vectors.
- **mlpack::optimization**
- **mlpack::optimization::test**

23.99.1 Detailed Description

Author

Ryan Curtin

A collection of functions to test optimizers (in this case, L-BFGS). These come from the following paper:

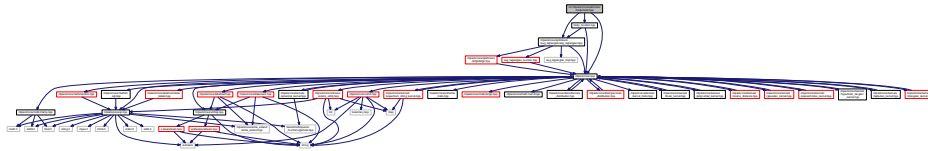
"Testing Unconstrained Optimization Software" Jorge J. Moré, Burton S. Garbow, and Kenneth E. Hillstom. 1981. ACM Trans. Math. Softw. 7, 1 (March 1981), 17-41. <http://portal.acm.org/citation.cfm?id=355934>.

This file is part of mlpack 1.0.12.

mlpack is free software; you may redistribute it and/or modify it under the terms of the 3-clause BSD license. You should have received a copy of the 3-clause BSD license along with mlpack. If not, see <http://www.opensource.org/licenses/BSD-3-Clause> for more information.

23.100 src/mlpack/core/optimizers/lrsdp/lrsdp.hpp File Reference

Include dependency graph for lrsdp.hpp:



Classes

- class **mlpack::optimization::LRSDP**

LRSDP (p. 468) is the implementation of Monteiro and Burer's formulation of low-rank semidefinite programs (LR-SDP).

Namespaces

- **mlpack**

Linear algebra utility functions, generally performed on matrices or vectors.

- **mlpack::optimization**

23.100.1 Detailed Description

Author

Ryan Curtin

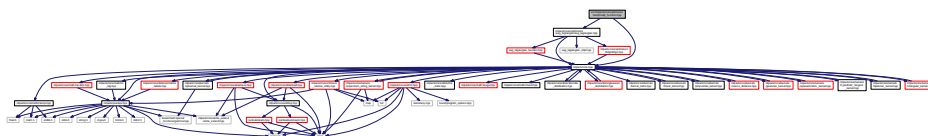
An implementation of Monteiro and Burer's formulation of low-rank semidefinite programs (LR-SDP).

This file is part of mlpack 1.0.12.

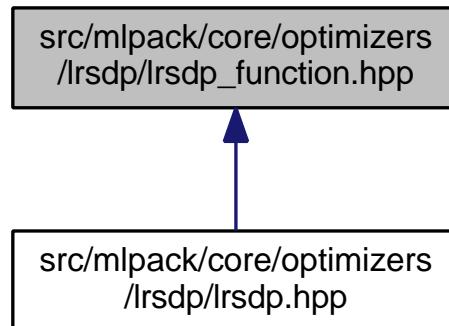
mlpack is free software; you may redistribute it and/or modify it under the terms of the 3-clause BSD license. You should have received a copy of the 3-clause BSD license along with mlpack. If not, see <http://www.opensource.org/licenses/BSD-3-Clause> for more information.

23.101 src/mlpack/core/optimizers/lrsdp/lrsdp_function.hpp File Reference

Include dependency graph for lrsdp_function.hpp:



This graph shows which files directly or indirectly include this file:



Classes

- class **mlpack::optimization::LRSDPFunction**

*The objective function that **LRSDP** (p. 468) is trying to optimize.*

Namespaces

- **mlpack**

Linear algebra utility functions, generally performed on matrices or vectors.

- **mlpack::optimization**

23.101.1 Detailed Description

Author

Ryan Curtin
Abhishek Laddha

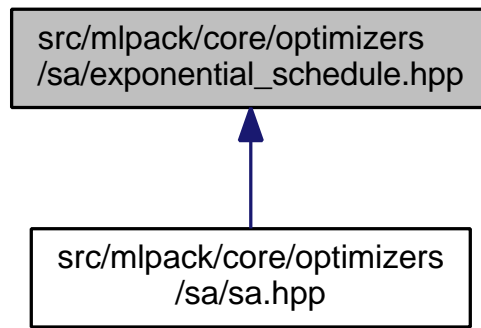
A class that represents the objective function which LRSDP optimizes.

This file is part of mlpack 1.0.12.

mlpack is free software; you may redistribute it and/or modify it under the terms of the 3-clause BSD license. You should have received a copy of the 3-clause BSD license along with mlpack. If not, see <http://www.opensource.org/licenses/BSD-3-Clause> for more information.

23.102 src/mlpack/core/optimizers/sa/exponential_schedule.hpp File Reference

This graph shows which files directly or indirectly include this file:

**Classes**

- class **mlpack::optimization::ExponentialSchedule**

The exponential cooling schedule cools the temperature T at every step according to the equation.

Namespaces

- **mlpack**
Linear algebra utility functions, generally performed on matrices or vectors.
- **mlpack::optimization**

23.102.1 Detailed Description

Author

Zhihao Lou

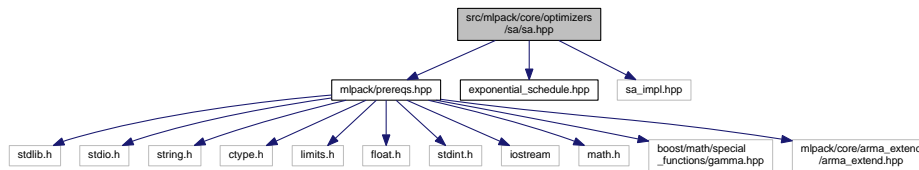
Exponential (geometric) cooling schedule used in SA.

This file is part of mlpack 1.0.12.

mlpack is free software; you may redistribute it and/or modify it under the terms of the 3-clause BSD license. You should have received a copy of the 3-clause BSD license along with mlpack. If not, see <http://www.opensource.org/licenses/BSD-3-Clause> for more information.

23.103 src/mlpack/core/optimizers/sa/sa.hpp File Reference

Include dependency graph for sa.hpp:



Classes

- class **mlpack::optimization::SA**< **FunctionType**, **CoolingScheduleType** >

Simulated Annealing is an stochastic optimization algorithm which is able to deliver near-optimal results quickly without knowing the gradient of the function being optimized.

Namespaces

- mlpack**
Linear algebra utility functions, generally performed on matrices or vectors.
- mlpack::optimization**

23.103.1 Detailed Description

Author

Zhihao Lou

Simulated Annealing (SA).

This file is part of mlpack 1.0.12.

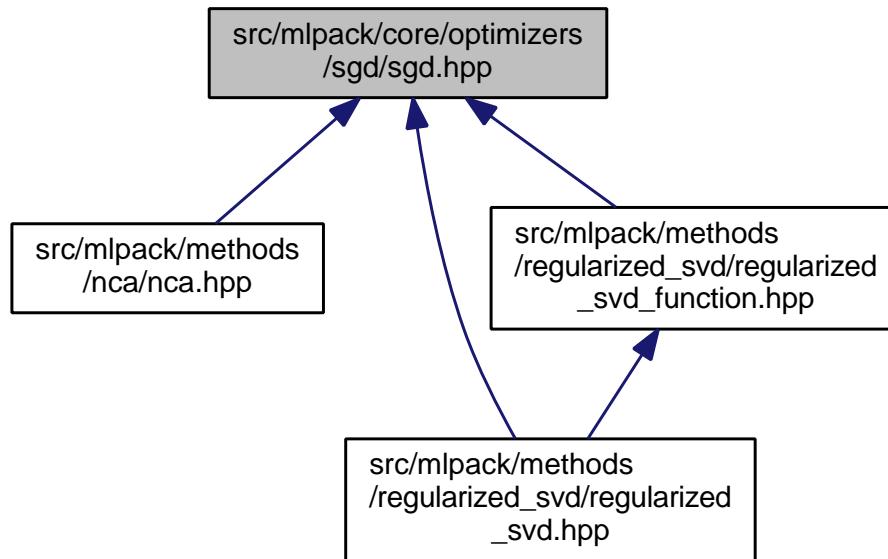
mlpack is free software; you may redistribute it and/or modify it under the terms of the 3-clause BSD license. You should have received a copy of the 3-clause BSD license along with mlpack. If not, see <http://www.opensource.org/licenses/BSD-3-Clause> for more information.

23.104 src/mlpack/core/optimizers/sgd/sgd.hpp File Reference

Include dependency graph for sgd.hpp:



This graph shows which files directly or indirectly include this file:



Classes

- class **mlpack::optimization::SGD**< **DecomposableFunctionType** >

Stochastic Gradient Descent is a technique for minimizing a function which can be expressed as a sum of other functions.

Namespaces

- **mlpack**

Linear algebra utility functions, generally performed on matrices or vectors.

- **mlpack::optimization**

23.104.1 Detailed Description

Author

Ryan Curtin

Stochastic Gradient Descent (SGD).

This file is part of mlpack 1.0.12.

mlpack is free software; you may redistribute it and/or modify it under the terms of the 3-clause BSD license. You should have received a copy of the 3-clause BSD license along with mlpack. If not, see <http://www.opensource.org/licenses/BSD-3-Clause> for more information.

23.105 src/mlpack/core/optimizers/sgd/test_function.hpp File Reference

Include dependency graph for test_function.hpp:



Classes

- class **mlpack::optimization::test::SGDTestFunction**

Very, very simple test function which is the composite of three other functions.

Namespaces

- **mlpack**
Linear algebra utility functions, generally performed on matrices or vectors.
- **mlpack::optimization**
- **mlpack::optimization::test**

23.105.1 Detailed Description

Author

Ryan Curtin

Very simple test function for SGD.

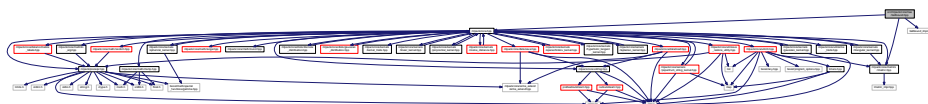
This file is part of mlpack 1.0.12.

mlpack is free software; you may redistribute it and/or modify it under the terms of the 3-clause BSD license. You should have received a copy of the 3-clause BSD license along with mlpack. If not, see <http://www.opensource.org/licenses/BSD-3-Clause> for more information.

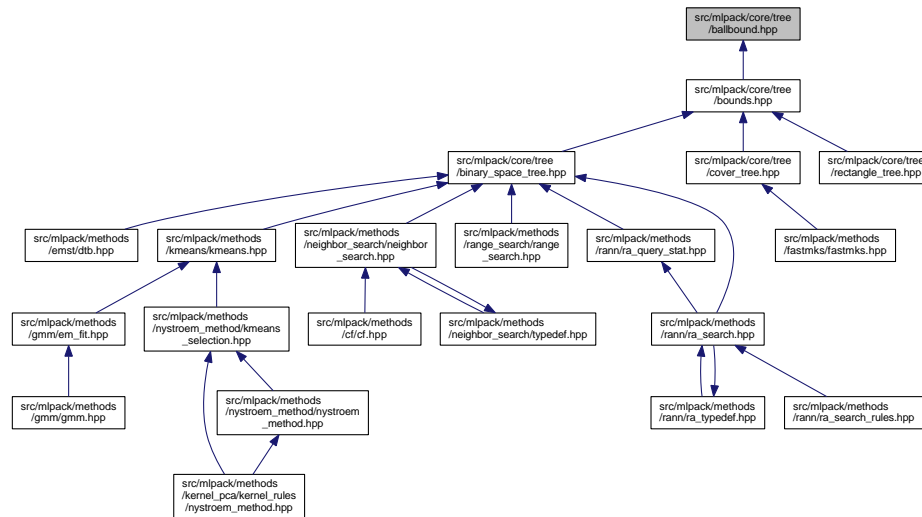
23.106 src/mlpack/core/tree/ballbound.hpp File Reference

Bounds that are useful for binary space partitioning trees.

Include dependency graph for ballbound.hpp:



This graph shows which files directly or indirectly include this file:



Classes

- class **mlpack::bound::BallBound**< **VecType**, **TMetricType** >
Ball bound encloses a set of points at a specific distance (radius) from a specific point (center).

Namespaces

- **mlpack**
Linear algebra utility functions, generally performed on matrices or vectors.
- **mlpack::bound**

23.106.1 Detailed Description

Bounds that are useful for binary space partitioning trees.

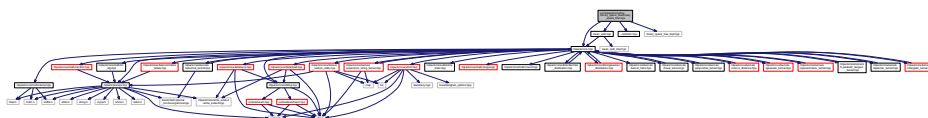
Interface to a ball bound that works in arbitrary metric spaces.

This file is part of mlpack 1.0.12.

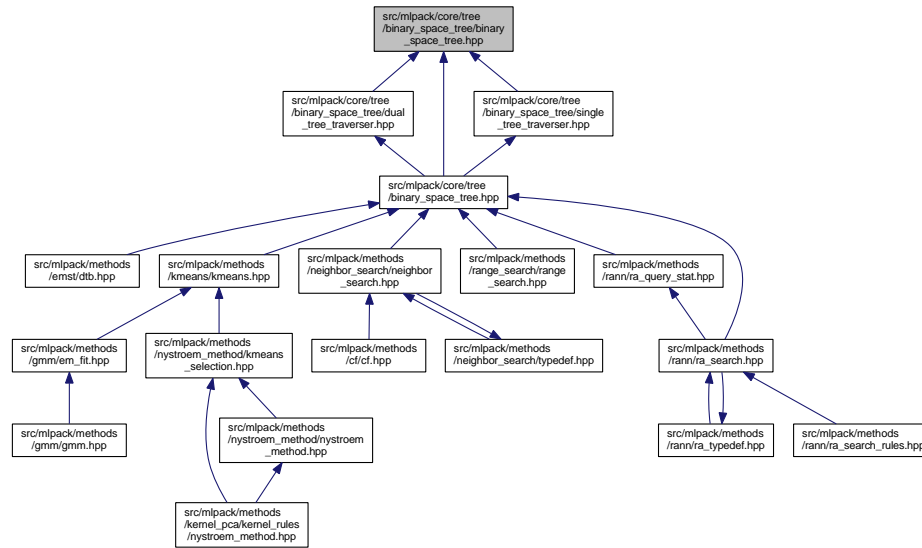
mlpack is free software; you may redistribute it and/or modify it under the terms of the 3-clause BSD license. You should have received a copy of the 3-clause BSD license along with mlpack. If not, see <http://www.opensource.org/licenses/BSD-3-Clause> for more information.

23.107 src/mlpack/core/tree/binary_space_tree/binary_space_tree.hpp File Reference

Include dependency graph for binary_space_tree.hpp:



This graph shows which files directly or indirectly include this file:



Classes

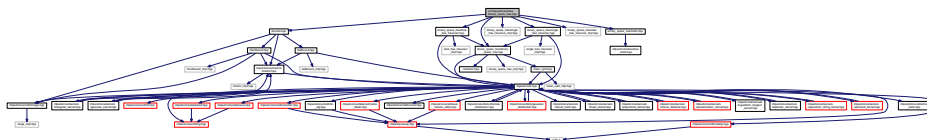
- class **mlpack::tree::BinarySpaceTree**< **BoundType**, **StatisticType**, **MatType**, **SplitType** >
A binary space partitioning tree, such as a KD-tree or a ball tree.
- class **mlpack::tree::BinarySpaceTree**< **BoundType**, **StatisticType**, **MatType**, **SplitType** >::**DualTree**< **Traverser**< **RuleType** >
A dual-tree traverser for binary space trees; see `dual_tree_traverser.hpp`.
- class **mlpack::tree::BinarySpaceTree**< **BoundType**, **StatisticType**, **MatType**, **SplitType** >::**SingleTree**< **Traverser**< **RuleType** >
A single-tree traverser for binary space trees; see `single_tree_traverser.hpp` for implementation.

Namespaces

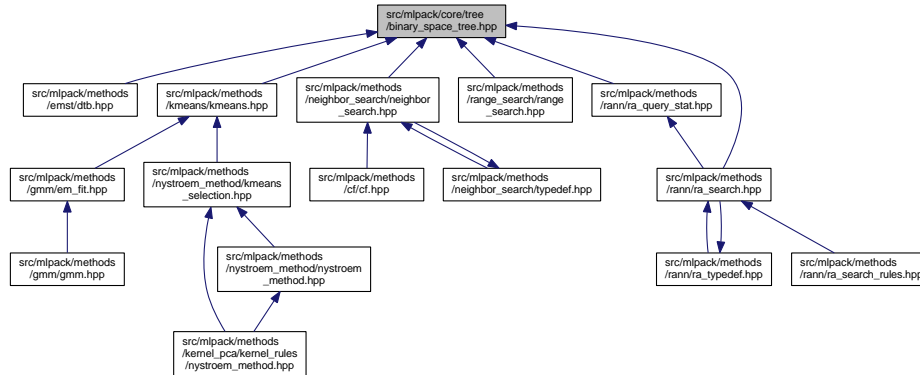
- **mlpack**
Linear algebra utility functions, generally performed on matrices or vectors.
- **mlpack::tree**
Trees and tree-building procedures.

23.108 src/mlpack/core/tree/binary_space_tree.hpp File Reference

Include dependency graph for `binary_space_tree.hpp`:

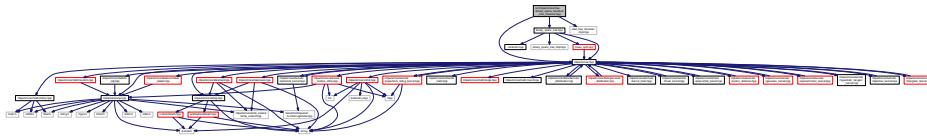


This graph shows which files directly or indirectly include this file:

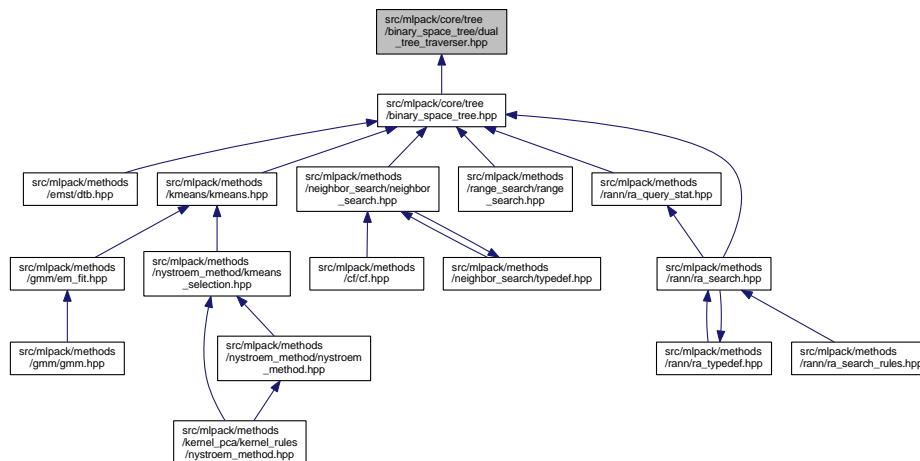


23.109 src/mlpack/core/tree/binary_space_tree/dual_tree_traverser.hpp File Reference

Include dependency graph for dual_tree_traverser.hpp:



This graph shows which files directly or indirectly include this file:



Classes

- class **mlpack::tree::BinarySpaceTree**< **BoundType**, **StatisticType**, **MatType**, **SplitType** >::**DualTree**< **Traverser**< **RuleType** >

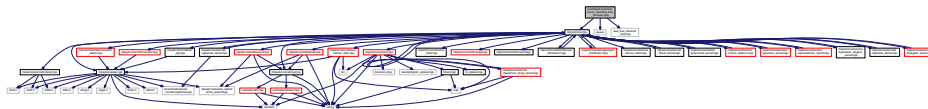
A dual-tree traverser for binary space trees; see dual_tree_traverser.hpp.

Namespaces

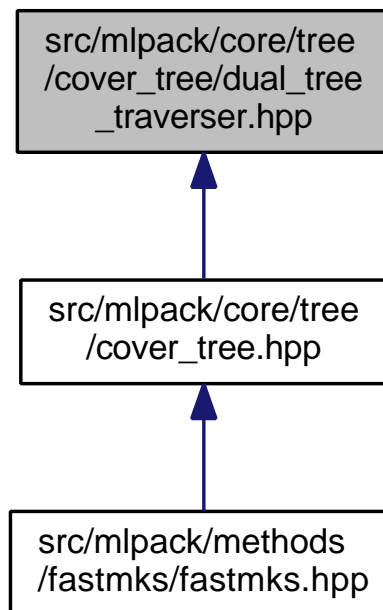
- **mlpack**
Linear algebra utility functions, generally performed on matrices or vectors.
- **mlpack::tree**
Trees and tree-building procedures.

23.110 src/mlpack/core/tree/cover_tree/dual_tree_traverser.hpp File Reference

Include dependency graph for dual_tree_traverser.hpp:



This graph shows which files directly or indirectly include this file:



Classes

- class **mlpack::tree::CoverTree< MetricType, RootPointPolicy, StatisticType >::DualTreeTraverser< RuleType >**
A dual-tree cover tree traverser; see dual_tree_traverser.hpp.
- struct **mlpack::tree::CoverTree< MetricType, RootPointPolicy, StatisticType >::DualTreeTraverser< RuleType >::DualCoverTreeMapEntry**
Struct used for traversal.

Namespaces

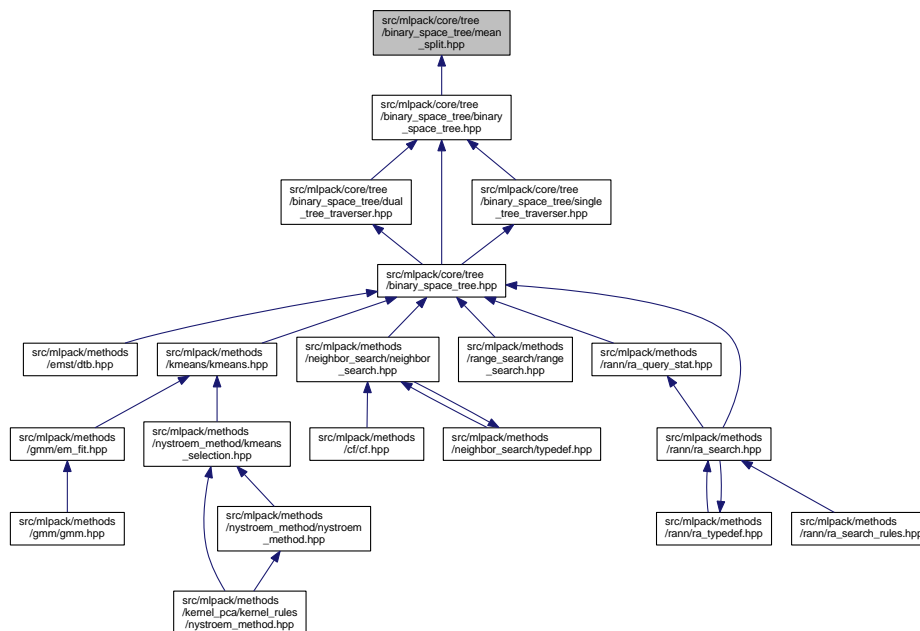
- **mlpack**
Linear algebra utility functions, generally performed on matrices or vectors.
- **mlpack::tree**
Trees and tree-building procedures.

23.111 src/mlpack/core/tree/binary_space_tree/mean_split.hpp File Reference

Include dependency graph for mean_split.hpp:



This graph shows which files directly or indirectly include this file:



Classes

- class **mlpack::tree::MeanSplit**< **BoundType**, **MatType** >
A binary space partitioning tree node is split into its left and right child.

Namespaces

- **mlpack**
Linear algebra utility functions, generally performed on matrices or vectors.

- **mlpack::tree**

Trees and tree-building procedures.

23.111.1 Detailed Description

Author

Yash Vadalía
Ryan Curtin

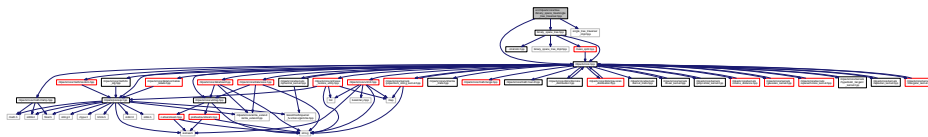
Definition of MeanSplit, a class that splits a binary space partitioning tree node into two parts using the mean of the values in a certain dimension.

This file is part of mlpack 1.0.12.

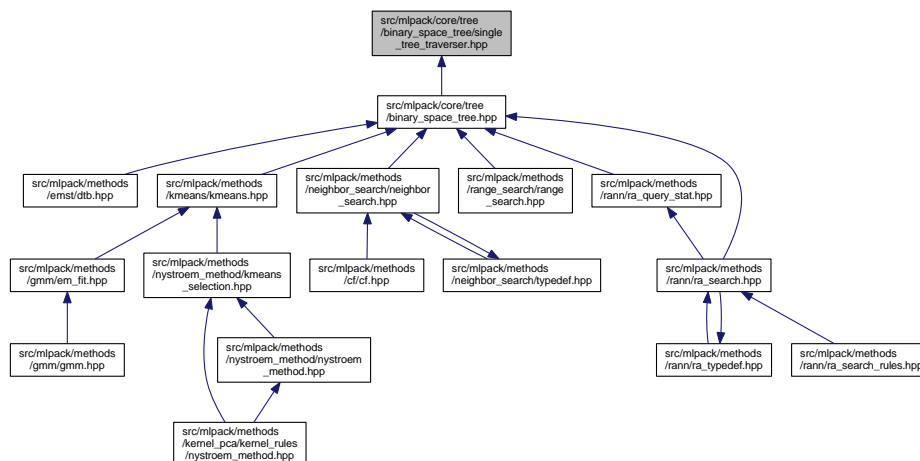
mlpack is free software; you may redistribute it and/or modify it under the terms of the 3-clause BSD license. You should have received a copy of the 3-clause BSD license along with mlpack. If not, see <http://www.opensource.org/licenses/BSD-3-Clause> for more information.

23.112 src/mlpack/core/tree/binary_space_tree/single_tree_traverser.hpp File Reference

Include dependency graph for single_tree_traverser.hpp:



This graph shows which files directly or indirectly include this file:



Classes

- class **mlpack::tree::BinarySpaceTree**< **BoundType**, **StatisticType**, **MatType**, **SplitType** >::**SingleTree**< **Traverser**< **RuleType** >

A single-tree traverser for binary space trees; see `single_tree_traverser.hpp` for implementation.

Namespaces

- **mlpack**

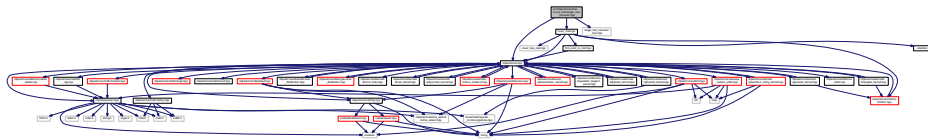
Linear algebra utility functions, generally performed on matrices or vectors.

- **mlpack::tree**

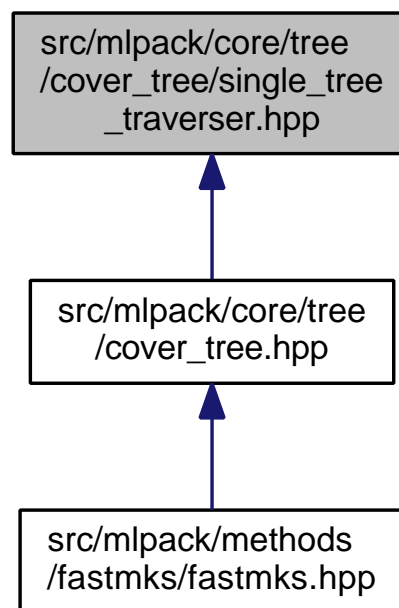
Trees and tree-building procedures.

23.113 src/mlpack/core/tree/cover_tree/single_tree_traverser.hpp File Reference

Include dependency graph for `single_tree_traverser.hpp`:



This graph shows which files directly or indirectly include this file:



Classes

- class **mlpack::tree::CoverTree< MetricType, RootPointPolicy, StatisticType >::SingleTreeTraverser< RuleType >**

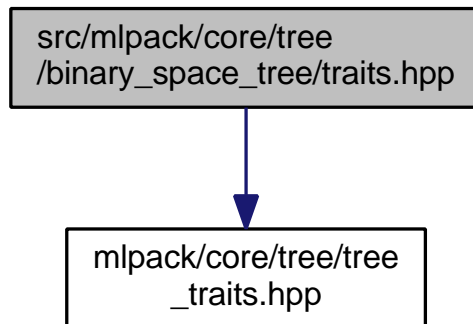
A single-tree cover tree traverser; see `single_tree_traverser.hpp` for implementation.

Namespaces

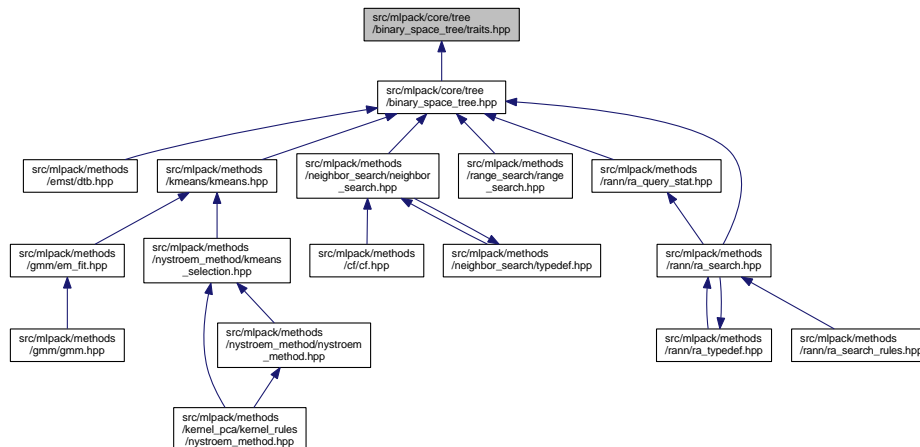
- **mlpack**
Linear algebra utility functions, generally performed on matrices or vectors.
- **mlpack::tree**
Trees and tree-building procedures.

23.114 src/mlpack/core/tree/binary_space_tree/traits.hpp File Reference

Include dependency graph for traits.hpp:



This graph shows which files directly or indirectly include this file:



Classes

- class **mlpack::tree::TreeTraits**< **BinarySpaceTree**< **BoundType**, **StatisticType**, **MatType** > >
*This is a specialization of the `TreeType` class to the **BinarySpaceTree** (p. 566) tree type.*

Namespaces

- **mlpack**

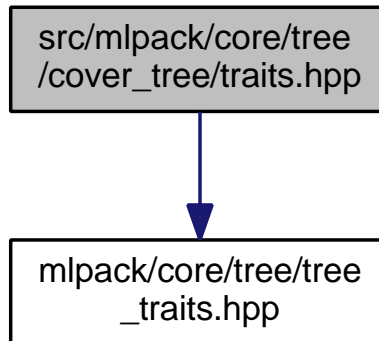
Linear algebra utility functions, generally performed on matrices or vectors.

- **mlpack::tree**

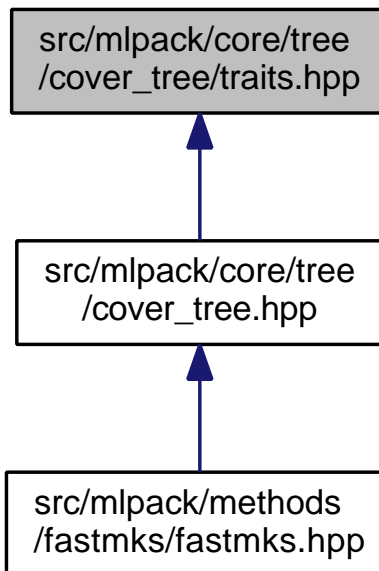
Trees and tree-building procedures.

23.115 src/mlpack/core/tree/cover_tree/traits.hpp File Reference

Include dependency graph for traits.hpp:



This graph shows which files directly or indirectly include this file:



Classes

- class **mlpack::tree::TreeTraits**< **CoverTree**< **MetricType**, **RootPointPolicy**, **StatisticType** > >

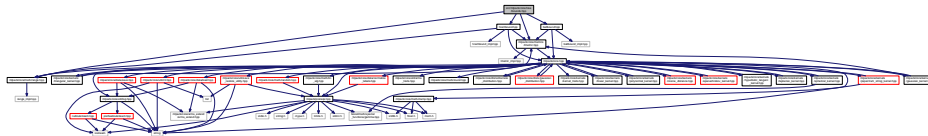
*The specialization of the **TreeTraits** (p. 645) class for the **CoverTree** (p. 603) tree type.*

- **mlpack**

- **mlpack::tree**

23.116 src/mlpack/core/tree/bounds.hpp File Reference

Include dependency graph for bounds.hpp:



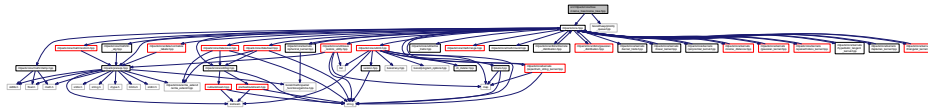
```

graph TD
    Bounds["src/mlpack/core/tree/Bounds.hpp"] --> BinarySpaceTree["src/mlpack/core/tree/binary_space_tree.hpp"]
    Bounds --> CoverTree["src/mlpack/core/tree/cover_tree.hpp"]
    Bounds --> RectangleTree["src/mlpack/core/tree/rectangle_tree.hpp"]
    BinarySpaceTree --> EmstDb["src/mlpack/methods/emst/db.hpp"]
    BinarySpaceTree --> Kmeans["src/mlpack/methods/kmeans/kmeans.hpp"]
    BinarySpaceTree --> NeighborSearch["src/mlpack/methods/neighbor_search/neighbor_search.hpp"]
    BinarySpaceTree --> RangeSearch["src/mlpack/methods/range_search/range_search.hpp"]
    BinarySpaceTree --> RannRaQueryStat["src/mlpack/methods/rann/ra_query_stat.hpp"]
    CoverTree --> Fastmks["src/mlpack/methods/fastmks/fastmks.hpp"]
    CoverTree --> RannRaSearch["src/mlpack/methods/rann/ra_search.hpp"]
    RannRaSearch --> RannRaTypedef["src/mlpack/methods/rann/ra_typedef.hpp"]
    RannRaSearch --> RannRaSearchRules["src/mlpack/methods/rann/ra_search_rules.hpp"]
    NeighborSearch --> IclCl["src/mlpack/methods/icl/cl.hpp"]
    NeighborSearch --> NeighborSearchTypedef["src/mlpack/methods/neighbor_search/typedef.hpp"]
    RannRaSearch --> RannRaTypedef
    RannRaSearch --> RannRaSearchRules
    GmmEmFit["src/mlpack/methods/gmm/em_fit.hpp"] --> Kmeans
    NystroemMethodKmeansSelection["src/mlpack/methods/nystroem_method/kmeans_selection.hpp"] --> Kmeans
    NystroemMethodKmeansSelection --> NystroemMethod["src/mlpack/methods/nystroem_method/nystroem_method.hpp"]
    KernelPcaKernelRulesNystroemMethod["src/mlpack/methods/kernel_pca/kernel_rules/nystroem_method.hpp"] --> NystroemMethod
    Gmm["src/mlpack/methods/gmm/gmm.hpp"] --> GmmEmFit
  
```

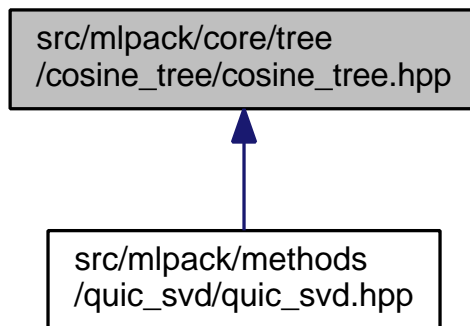
mlpack is free software; you may redistribute it and/or modify it under the terms of the 3-clause BSD license. You should have received a copy of the 3-clause BSD license along with mlpack. If not, see <http://www.opensource.org/licenses/BSD-3-Clause> for more information.

23.117 src/mlpack/core/tree/cosine_tree/cosine_tree.hpp File Reference

Include dependency graph for cosine_tree.hpp:



This graph shows which files directly or indirectly include this file:



Classes

- class **mlpack::tree::CompareCosineNode**
- class **mlpack::tree::CosineTree**

Namespaces

- **mlpack**
Linear algebra utility functions, generally performed on matrices or vectors.
- **mlpack::tree**
Trees and tree-building procedures.

Typedefs

- typedef boost::heap::priority_queue< CosineTree *, boost::heap::compare< CompareCosineNode > > **mlpack::tree::CosineNodeQueue**

23.117.1 Detailed Description

Author

Siddharth Agrawal

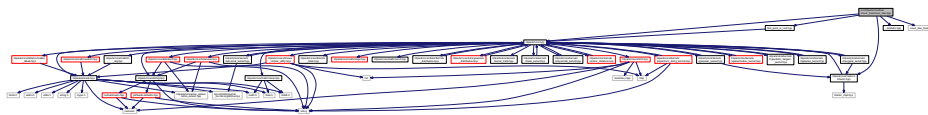
Definition of Cosine Tree.

This file is part of mlpack 1.0.12.

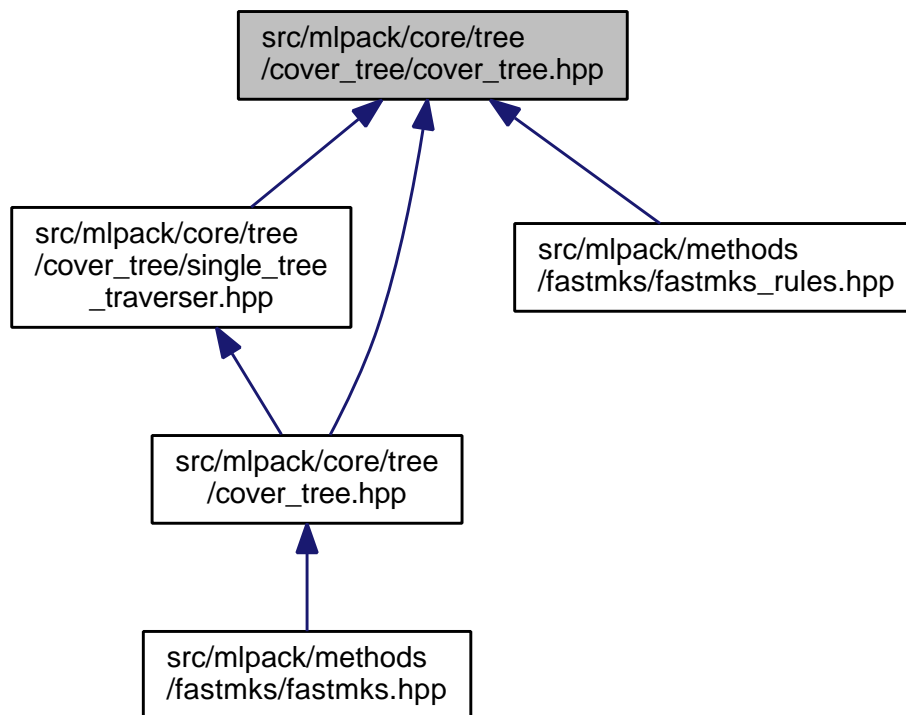
mlpack is free software; you may redistribute it and/or modify it under the terms of the 3-clause BSD license. You should have received a copy of the 3-clause BSD license along with mlpack. If not, see <http://www.opensource.org/licenses/BSD-3-Clause> for more information.

23.118 src/mlpack/core/tree/cover_tree/cover_tree.hpp File Reference

Include dependency graph for cover_tree.hpp:



This graph shows which files directly or indirectly include this file:

**Classes**

- class **mlpack::tree::CoverTree**< **MetricType**, **RootPointPolicy**, **StatisticType** >

A cover tree is a tree specifically designed to speed up nearest-neighbor computation in high-dimensional spaces.

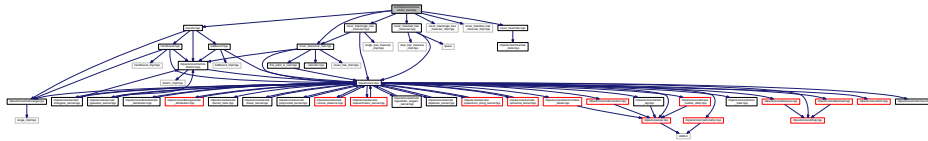
- class **mlpack::tree::CoverTree**< **MetricType**, **RootPointPolicy**, **StatisticType** >::**DualTreeTraverser**< **RuleType** >
A dual-tree cover tree traverser; see *dual_tree_traverser.hpp*.
- class **mlpack::tree::CoverTree**< **MetricType**, **RootPointPolicy**, **StatisticType** >::**SingleTreeTraverser**< **RuleType** >
A single-tree cover tree traverser; see *single_tree_traverser.hpp* for implementation.

Namespaces

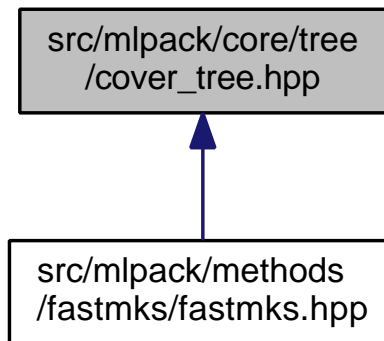
- **mlpack**
Linear algebra utility functions, generally performed on matrices or vectors.
- **mlpack::tree**
Trees and tree-building procedures.

23.119 src/mlpack/core/tree/cover_tree.hpp File Reference

Include dependency graph for *cover_tree.hpp*:



This graph shows which files directly or indirectly include this file:



23.120 src/mlpack/core/tree/cover_tree/first_point_is_root.hpp File Reference

Include dependency graph for *first_point_is_root.hpp*:



```
graph BT; A["src/mlpack/core/tree/cover_tree/first_point_is_root.hpp"] --> B["src/mlpack/core/tree/cover_tree/cover_tree.hpp"]; B --> C["src/mlpack/core/tree/cover_tree/single_tree_traverser.hpp"]; B --> D["src/mlpack/core/tree/cover_tree/cover_tree.hpp"]; B --> E["src/mlpack/methods/fastmks/fastmks_rules.hpp"]; C --> D; D --> D; D --> F["src/mlpack/methods/fastmks/fastmks.hpp"];
```

- class **mlpack::tree::FirstPointsRoot**

Namespaces

- Linear algebra utility functions, generally performed on matrices or vectors.*

- ### *Trees and tree-building procedures.*

Generated on Fri Sep 25 2015 08:06:54 for mlpack by Doxygen

Author

Ryan Curtin

A very simple policy for the cover tree; the first point in the dataset is chosen as the root of the cover tree.

This file is part of mlpack 1.0.12.

mlpack is free software; you may redistribute it and/or modify it under the terms of the 3-clause BSD license. You should have received a copy of the 3-clause BSD license along with mlpack. If not, see <http://www.opensource.org/licenses/BSD-3-Clause> for more information.

23.121 src/mlpack/core/tree/example_tree.hpp File Reference

Classes

- class **mlpack::tree::ExampleTree**< **MetricType**, **StatisticType**, **MatType** >

This is not an actual space tree but instead an example tree that exists to show and document all the functions that mlpack trees must implement.

Namespaces

- **mlpack**

Linear algebra utility functions, generally performed on matrices or vectors.

- **mlpack::tree**

Trees and tree-building procedures.

23.121.1 Detailed Description

Author

Ryan Curtin

An example tree. This contains all the functions that mlpack trees must implement (although the actual implementations here don't make any sense because this is just an example).

This file is part of mlpack 1.0.12.

mlpack is free software; you may redistribute it and/or modify it under the terms of the 3-clause BSD license. You should have received a copy of the 3-clause BSD license along with mlpack. If not, see <http://www.opensource.org/licenses/BSD-3-Clause> for more information.

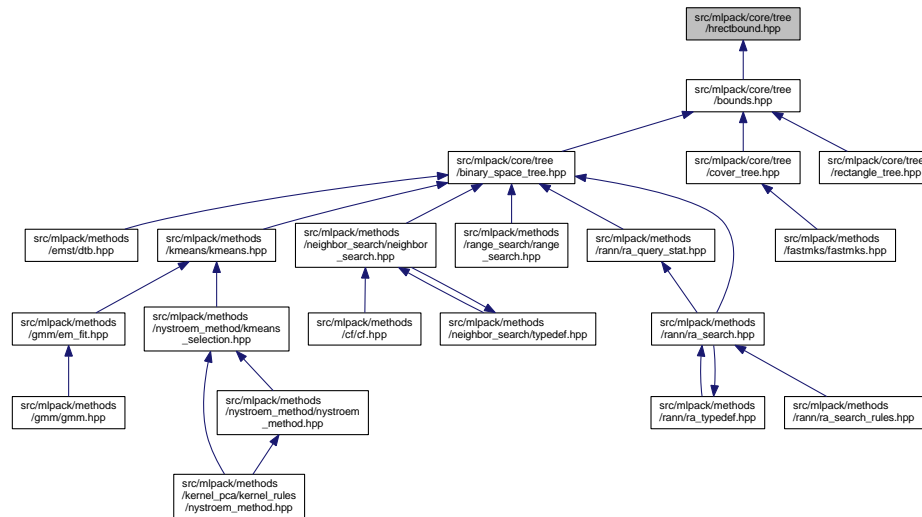
23.122 src/mlpack/core/tree/hrectbound.hpp File Reference

Bounds that are useful for binary space partitioning trees.

Include dependency graph for hrectbound.hpp:



This graph shows which files directly or indirectly include this file:



Classes

- class **mlpack::bound::HRectBound**< **Power, TakeRoot** >
Hyper-rectangle bound for an L-metric.

Namespaces

- **mlpack**
Linear algebra utility functions, generally performed on matrices or vectors.
- **mlpack::bound**

23.122.1 Detailed Description

Bounds that are useful for binary space partitioning trees.

This file describes the interface for the HRectBound class, which implements a hyperrectangle bound.

This file is part of mlpack 1.0.12.

mlpack is free software; you may redistribute it and/or modify it under the terms of the 3-clause BSD license. You should have received a copy of the 3-clause BSD license along with mlpack. If not, see <http://www.opensource.org/licenses/BSD-3-Clause> for more information.

23.123 src/mlpack/core/tree/mrkd_statistic.hpp File Reference

Include dependency graph for mrkd_statistic.hpp:



Classes

- class **mlpack::tree::MRKDStatistic**

Statistic for multi-resolution kd-trees.

Namespaces

- **mlpack**

Linear algebra utility functions, generally performed on matrices or vectors.

- **mlpack::tree**

Trees and tree-building procedures.

23.123.1 Detailed Description

Author

James Cline

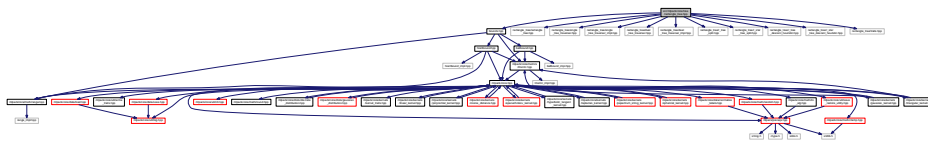
Definition of the statistic for multi-resolution kd-trees.

This file is part of mlpack 1.0.12.

mlpack is free software; you may redistribute it and/or modify it under the terms of the 3-clause BSD license. You should have received a copy of the 3-clause BSD license along with mlpack. If not, see <http://www.opensource.org/licenses/BSD-3-Clause> for more information.

23.124 src/mlpack/core/tree/rectangle_tree.hpp File Reference

Include dependency graph for rectangle_tree.hpp:



23.124.1 Detailed Description

Author

Andrew Wells

Include all the necessary files to use the Rectangle Type Trees (RTree, RStarTree, XTree, and HilbertRTree.)

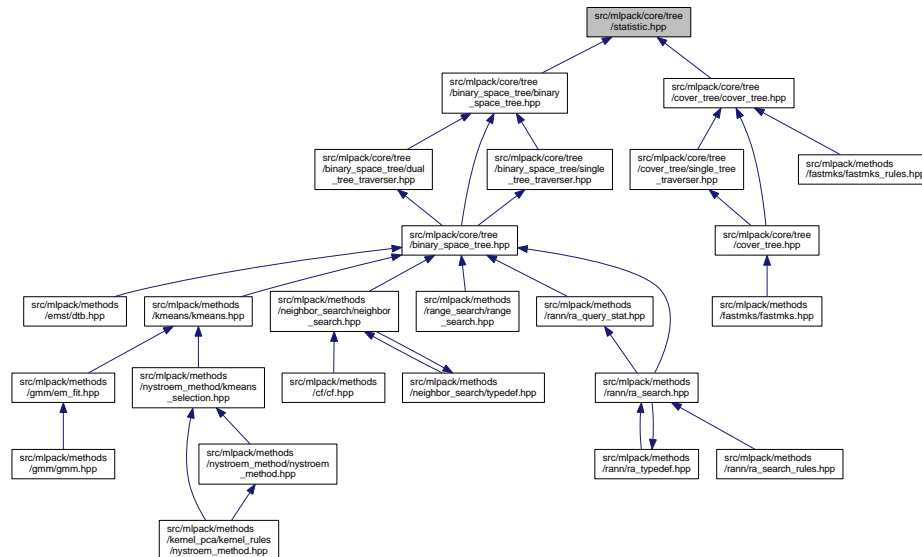
This file is part of mlpack 1.0.12.

mlpack is free software; you may redistribute it and/or modify it under the terms of the 3-clause BSD license. You should have received a copy of the 3-clause BSD license along with mlpack. If not, see <http://www.opensource.org/licenses/BSD-3-Clause> for more information.

23.125 src/mlpack/core/tree/statistic.hpp File Reference

Definition of the policy type for the statistic class.

This graph shows which files directly or indirectly include this file:



Classes

- class **mlpack::tree::EmptyStatistic**

Empty statistic if you are not interested in storing statistics in your tree.

Namespaces

- **mlpack**

Linear algebra utility functions, generally performed on matrices or vectors.

- **mlpack::tree**

Trees and tree-building procedures.

23.125.1 Detailed Description

Definition of the policy type for the statistic class.

You should define your own statistic that looks like EmptyStatistic.

This file is part of mlpack 1.0.12.

mlpack is free software; you may redistribute it and/or modify it under the terms of the 3-clause BSD license. You should have received a copy of the 3-clause BSD license along with mlpack. If not, see <http://www.opensource.org/licenses/BSD-3-Clause> for more information.

23.126 src/mlpack/core/tree/traversal_info.hpp File Reference

Classes

- class **TraversalInfo**< **TreeType** >

*The **TraversalInfo** (p. 673) class holds traversal information which is used in dual-tree (and single-tree) traversals.*

23.126.1 Detailed Description

Author

Ryan Curtin

This class will hold the traversal information for dual-tree traversals. A dual-tree traversal should be updating the members of this class before `Score()` is called.

This file is part of mlpack 1.0.12.

mlpack is free software; you may redistribute it and/or modify it under the terms of the 3-clause BSD license. You should have received a copy of the 3-clause BSD license along with mlpack. If not, see <http://www.opensource.org/licenses/BSD-3-Clause> for more information.

23.127 src/mlpack/core/tree/TREE_EXPLANATION.txt File Reference

Variables

- An attempt to outline how the trees work At the we have a description of how the tree is **set** up in terms of functionality Then we state which files are responsible for each function At the **bottom**
- An attempt to outline how the trees work At the we have a description of how the tree is **set** up in terms of functionality Then we state which files are responsible for each function At the we list the files and explain what each of them does Since dynamic changes are not there are two main activities on a Binary Space the following classes are coded **separately**
- An attempt to outline how the trees work At the we have a description of how the tree is **set** up in terms of functionality Then we state which files are responsible for each function At the we list the files and explain what each of them does Since dynamic changes are not **supported**
- An attempt to outline how the trees work At the **top**
- An attempt to outline how the trees work At the we have a description of how the tree is **set** up in terms of functionality Then we state which files are responsible for each function At the we list the files and explain what each of them does Since dynamic changes are not there are two main activities on a Binary Space **Tree**

23.127.1 Variable Documentation

- 23.127.1.1 An attempt to outline how the trees work At the we have a description of how the tree is **set** up in terms of functionality Then we state which files are responsible for each function At the **bottom**

Definition at line 1 of file TREE_EXPLANATION.txt.

- 23.127.1.2 An attempt to outline how the trees work At the we have a description of how the tree is **set** up in terms of functionality Then we state which files are responsible for each function At the we list the files and explain what each of them does Since dynamic changes are not there are two main activities on a Binary Space the following classes are coded **separately**

Definition at line 1 of file TREE_EXPLANATION.txt.

- 23.127.1.3 An attempt to outline how the trees work At the we have a description of how the tree is set up in terms of functionality Then we state which files are responsible for each function At the we list the files and explain what each of them does Since dynamic changes are not supported

Definition at line 1 of file TREE_EXPLANATION.txt.

- 23.127.1.4 An attempt to outline how the trees work At the top

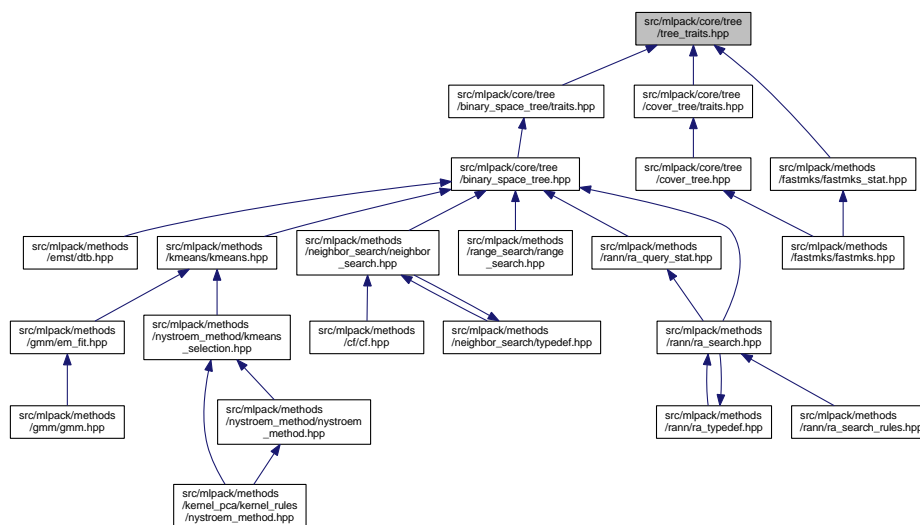
Definition at line 1 of file TREE_EXPLANATION.txt.

- 23.127.1.5 An attempt to outline how the trees work At the we have a description of how the tree is set up in terms of functionality Then we state which files are responsible for each function At the we list the files and explain what each of them does Since dynamic changes are not there are two main activities on a Binary Space Tree

Definition at line 1 of file TREE_EXPLANATION.txt.

23.128 src/mlpack/core/tree/tree_traits.hpp File Reference

This graph shows which files directly or indirectly include this file:



Classes

- class `mlpack::tree::TreeTraits< TreeType >`

The *TreeTraits* (p. 645) class provides compile-time information on the characteristics of a given tree type.

Namespaces

- `mlpack`

Linear algebra utility functions, generally performed on matrices or vectors.

- `mlpack::tree`

Trees and tree-building procedures.

23.128.1 Detailed Description

Author

Ryan Curtin

This file implements the basic, unspecialized TreeTraits class, which provides information about tree types. If you create a tree class, you should specialize this class with the characteristics of your tree.

This file is part of mlpack 1.0.12.

mlpack is free software; you may redistribute it and/or modify it under the terms of the 3-clause BSD license. You should have received a copy of the 3-clause BSD license along with mlpack. If not, see <http://www.opensource.org/licenses/BSD-3-Clause> for more information.

23.129 src/mlpack/core/util/arma_traits.hpp File Reference

This graph shows which files directly or indirectly include this file:



Classes

- struct **IsVector**< **VecType** >
 - If value == true, then VecType is some sort of Armadillo vector or subview.*
- struct **IsVector**< arma::Col< eT > >
- struct **IsVector**< arma::Row< eT > >
- struct **IsVector**< arma::SpCol< eT > >
- struct **IsVector**< arma::SpRow< eT > >
- struct **IsVector**< arma::SpSubview< eT > >
- struct **IsVector**< arma::subview_col< eT > >
- struct **IsVector**< arma::subview_row< eT > >

23.129.1 Detailed Description

Author

Ryan Curtin

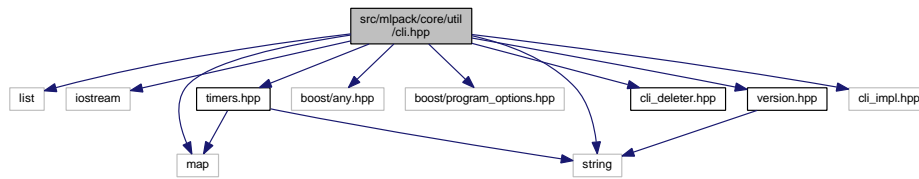
Some traits used for template metaprogramming (SFINAE) with Armadillo types.

This file is part of mlpack 1.0.12.

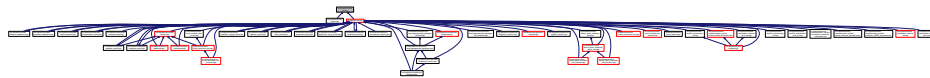
mlpack is free software; you may redistribute it and/or modify it under the terms of the 3-clause BSD license. You should have received a copy of the 3-clause BSD license along with mlpack. If not, see <http://www.opensource.org/licenses/BSD-3-Clause> for more information.

23.130 src/mlpack/core/util/cli.hpp File Reference

Include dependency graph for cli.hpp:



This graph shows which files directly or indirectly include this file:



Classes

- class **mlpack::CLI**
Parses the command line for parameters and holds user-specified parameters.
- struct **mlpack::ParamData**
*Aids in the extensibility of **CLI** (p. 184) by focusing potential changes into one structure.*

Namespaces

- **mlpack**
Linear algebra utility functions, generally performed on matrices or vectors.
- **mlpack::util**

Macros

- **#define PARAM(T, ID, DESC, ALIAS, DEF, REQ)**
Define an input parameter.
- **#define PARAM_DOUBLE(ID, DESC, ALIAS, DEF) PARAM(double, ID, DESC, ALIAS, DEF, false)**
Define a double parameter.
- **#define PARAM_DOUBLE_REQ(ID, DESC, ALIAS)**
Define a required double parameter.
- **#define PARAM_FLAG(ID, DESC, ALIAS) PARAM_FLAG_INTERNAL(ID, DESC, ALIAS);**
Define a flag parameter.
- **#define PARAM_FLOAT(ID, DESC, ALIAS, DEF) PARAM(float, ID, DESC, ALIAS, DEF, false)**
Define a floating-point parameter.
- **#define PARAM_FLOAT_REQ(ID, DESC, ALIAS)**
Define a required floating-point parameter.
- **#define PARAM_INT(ID, DESC, ALIAS, DEF) PARAM(int, ID, DESC, ALIAS, DEF, false)**
Define an integer parameter.
- **#define PARAM_INT_REQ(ID, DESC, ALIAS) PARAM(int, ID, DESC, ALIAS, 0, true)**

Define a required integer parameter.

- `#define PARAM_STRING(ID, DESC, ALIAS, DEF) PARAM(std::string, ID, DESC, ALIAS, DEF, false)`

Define a string parameter.

- `#define PARAM_STRING_REQ(ID, DESC, ALIAS)`

Define a required string parameter.

- `#define PARAM_VECTOR(T, ID, DESC, ALIAS) PARAM(std::vector<T>, ID, DESC, ALIAS, std::vector<T>(), false)`

Define a vector parameter.

- `#define PARAM_VECTOR_REQ(T, ID, DESC, ALIAS)`

Define a required vector parameter.

- `#define PROGRAM_INFO(NAME, DESC)`

Document an executable.

- `#define TYPENAME(x) (std::string(typeid(x).name()))`

The TYPENAME macro is used internally to convert a type into a string.

23.130.1 Detailed Description

Author

Matthew Amidon

This file implements the CLI subsystem which is intended to replace FX. This can be used more or less regardless of context. In the future, it might be expanded to include file I/O.

This file is part of mlpack 1.0.12.

mlpack is free software; you may redistribute it and/or modify it under the terms of the 3-clause BSD license. You should have received a copy of the 3-clause BSD license along with mlpack. If not, see <http://www.opensource.org/licenses/BSD-3-Clause> for more information.

23.130.2 Macro Definition Documentation

23.130.2.1 `#define PARAM(T, ID, DESC, ALIAS, DEF, REQ)`

Value:

```
static mlpack::util::Option<T> JOIN(JOIN(io_option_dummy_object_, __LINE__), opt) (false, DEF, ID, \
DESC, ALIAS, REQ);
```

Define an input parameter.

Don't use this function; use the other ones above that call it. Note that we are using the **LINE** macro for naming these actual parameters when **COUNTER** does not exist, which is a bit of an ugly hack... but this is the preprocessor, after all. We don't have much choice other than ugliness.

Parameters

<i>T</i>	Type of the parameter.
<i>ID</i>	Name of the parameter.

<i>DESC</i>	Description of the parameter (1-2 sentences).
<i>ALIAS</i>	Alias for this parameter (one letter).
<i>DEF</i>	Default value of the parameter.
<i>REQ</i>	Whether or not parameter is required (boolean value).

Definition at line 349 of file cli.hpp.

23.130.2.2 `#define PARAM_DOUBLE(ID, DESC, ALIAS, DEF) PARAM(double, ID, DESC, ALIAS, DEF, false)`

Define a double parameter.

The parameter can then be specified on the command line with `-ID=value`.

Parameters

<i>ID</i>	Name of the parameter.
<i>DESC</i>	Quick description of the parameter (1-2 sentences).
<i>ALIAS</i>	An alias for the parameter (one letter).
<i>DEF</i>	Default value of the parameter.

See also

`mlpack::CLI` (p. 184), `PROGRAM_INFO()` (p. 766)

Bug The **COUNTER** variable is used in most cases to guarantee a unique global identifier for options declared using the `PARAM_*`() macros. However, not all compilers have this support—most notably, `gcc < 4.3`. In that case, the **LINE** macro is used as an attempt to get a unique global identifier, but collisions are still possible, and they produce bizarre error messages. See <http://mlpack.org/trac/ticket/74> for more information.

Definition at line 138 of file cli.hpp.

23.130.2.3 `#define PARAM_DOUBLE_REQ(ID, DESC, ALIAS)`

Value:

```
PARAM(double, ID, DESC, ALIAS, \
    0.0f, true)
```

Define a required double parameter.

The parameter must then be specified on the command line with `-ID=value`.

Parameters

<i>ID</i>	Name of the parameter.
<i>DESC</i>	Quick description of the parameter (1-2 sentences).
<i>ALIAS</i>	An alias for the parameter (one letter).

See also

`mlpack::CLI` (p. 184), `PROGRAM_INFO()` (p. 766)

Bug The **COUNTER** variable is used in most cases to guarantee a unique global identifier for options declared using the `PARAM_*`() macros. However, not all compilers have this support—most notably, `gcc < 4.3`. In that case, the **LINE** macro is used as an attempt to get a unique global identifier, but collisions are still possible, and they produce bizarre error messages. See <http://mlpack.org/trac/ticket/74> for more information.

Definition at line 259 of file cli.hpp.

23.130.2.4 `#define PARAM_FLAG(ID, DESC, ALIAS) PARAM_FLAG_INTERNAL(ID, DESC, ALIAS);`

Define a flag parameter.

Parameters

<i>ID</i>	Name of the parameter.
<i>DESC</i>	Quick description of the parameter (1-2 sentences).
<i>ALIAS</i>	An alias for the parameter (one letter).

See also

`mlpack::CLI` (p. 184), `PROGRAM_INFO()` (p. 766)

Bug The **COUNTER** variable is used in most cases to guarantee a unique global identifier for options declared using the `PARAM_*`() macros. However, not all compilers have this support—most notably, `gcc < 4.3`. In that case, the **LINE** macro is used as an attempt to get a unique global identifier, but collisions are still possible, and they produce bizarre error messages. See <http://mlpack.org/trac/ticket/74> for more information.

Definition at line 66 of file cli.hpp.

23.130.2.5 `#define PARAM_FLOAT(ID, DESC, ALIAS, DEF) PARAM(float, ID, DESC, ALIAS, DEF, false)`

Define a floating-point parameter.

You should use `PARAM_DOUBLE` instead.

The parameter can then be specified on the command line with `-ID=value`.

Parameters

<i>ID</i>	Name of the parameter.
<i>DESC</i>	Quick description of the parameter (1-2 sentences).
<i>ALIAS</i>	An alias for the parameter (one letter).
<i>DEF</i>	Default value of the parameter.

See also

`mlpack::CLI` (p. 184), `PROGRAM_INFO()` (p. 766)

Bug The **COUNTER** variable is used in most cases to guarantee a unique global identifier for options declared using the `PARAM_*`() macros. However, not all compilers have this support—most notably, `gcc < 4.3`. In that case, the **LINE** macro is used as an attempt to get a unique global identifier, but collisions are still possible, and they produce bizarre error messages. See <http://mlpack.org/trac/ticket/74> for more information.

Definition at line 114 of file cli.hpp.

23.130.2.6 `#define PARAM_FLOAT_REQ(ID, DESC, ALIAS)`

Value:

```
PARAM(float, ID, DESC, ALIAS, 0.0f, \
    true)
```

Define a required floating-point parameter.

You should probably use a double instead.

The parameter must then be specified on the command line with `-ID=value`. If `ALIAS` is equal to `DEF_MOD` (which is set using the **PROGRAM_INFO()** (p. 766) macro), the parameter can be specified with just `-ID=value`.

Parameters

<i>ID</i>	Name of the parameter.
<i>DESC</i>	Quick description of the parameter (1-2 sentences).
<i>ALIAS</i>	An alias for the parameter (one letter).

See also

mlpack::CLI (p. 184), **PROGRAM_INFO()** (p. 766)

Bug The **COUNTER** variable is used in most cases to guarantee a unique global identifier for options declared using the `PARAM_*`() macros. However, not all compilers have this support—most notably, `gcc < 4.3`. In that case, the **LINE** macro is used as an attempt to get a unique global identifier, but collisions are still possible, and they produce bizarre error messages. See <http://mlpack.org/trac/ticket/74> for more information.

Definition at line 236 of file cli.hpp.

23.130.2.7 `#define PARAM_INT(ID, DESC, ALIAS, DEF) PARAM(int, ID, DESC, ALIAS, DEF, false)`

Define an integer parameter.

The parameter can then be specified on the command line with `-ID=value`.

Parameters

<i>ID</i>	Name of the parameter.
<i>DESC</i>	Quick description of the parameter (1-2 sentences).
<i>ALIAS</i>	An alias for the parameter (one letter).
<i>DEF</i>	Default value of the parameter.

See also

mlpack::CLI (p. 184), **PROGRAM_INFO()** (p. 766)

Bug The **COUNTER** variable is used in most cases to guarantee a unique global identifier for options declared using the `PARAM_*`() macros. However, not all compilers have this support—most notably, `gcc < 4.3`. In that case, the **LINE** macro is used as an attempt to get a unique global identifier, but collisions are still possible, and they produce bizarre error messages. See <http://mlpack.org/trac/ticket/74> for more information.

Definition at line 90 of file cli.hpp.

23.130.2.8 `#define PARAM_INT_REQ(ID, DESC, ALIAS) PARAM(int, ID, DESC, ALIAS, 0, true)`

Define a required integer parameter.

The parameter must then be specified on the command line with `-ID=value`.

Parameters

<i>ID</i>	Name of the parameter.
<i>DESC</i>	Quick description of the parameter (1-2 sentences).
<i>ALIAS</i>	An alias for the parameter (one letter).

See also

mlpack::CLI (p. 184), **PROGRAM_INFO()** (p. 766)

Bug The **COUNTER** variable is used in most cases to guarantee a unique global identifier for options declared using the **PARAM_*()** macros. However, not all compilers have this support—most notably, gcc < 4.3. In that case, the **LINE** macro is used as an attempt to get a unique global identifier, but collisions are still possible, and they produce bizarre error messages. See <http://mlpack.org/trac/ticket/74> for more information.

Definition at line 212 of file cli.hpp.

```
23.130.2.9 #define PARAM_STRING( ID, DESC, ALIAS, DEF ) PARAM(std::string, ID, DESC, ALIAS, DEF, false)
```

Define a string parameter.

The parameter can then be specified on the command line with **–ID=value**. If **ALIAS** is equal to **DEF_MOD** (which is set using the **PROGRAM_INFO()** (p. 766) macro), the parameter can be specified with just **–ID=value**.

Parameters

<i>ID</i>	Name of the parameter.
<i>DESC</i>	Quick description of the parameter (1-2 sentences).
<i>ALIAS</i>	An alias for the parameter (one letter).
<i>DEF</i>	Default value of the parameter.

See also

mlpack::CLI (p. 184), **PROGRAM_INFO()** (p. 766)

Bug The **COUNTER** variable is used in most cases to guarantee a unique global identifier for options declared using the **PARAM_*()** macros. However, not all compilers have this support—most notably, gcc < 4.3. In that case, the **LINE** macro is used as an attempt to get a unique global identifier, but collisions are still possible, and they produce bizarre error messages. See <http://mlpack.org/trac/ticket/74> for more information.

Definition at line 163 of file cli.hpp.

```
23.130.2.10 #define PARAM_STRING_REQ( ID, DESC, ALIAS )
```

Value:

```
PARAM(std::string, ID, DESC, \
    ALIAS, "", true);
```

Define a required string parameter.

The parameter must then be specified on the command line with **–ID=value**.

Parameters

<i>ID</i>	Name of the parameter.
<i>DESC</i>	Quick description of the parameter (1-2 sentences).
<i>ALIAS</i>	An alias for the parameter (one letter).

See also

mlpack::CLI (p. 184), **PROGRAM_INFO()** (p. 766)

Bug The **COUNTER** variable is used in most cases to guarantee a unique global identifier for options declared using the **PARAM_***() macros. However, not all compilers have this support—most notably, gcc < 4.3. In that case, the **LINE** macro is used as an attempt to get a unique global identifier, but collisions are still possible, and they produce bizarre error messages. See <http://mlpack.org/trac/ticket/74> for more information.

Definition at line 282 of file cli.hpp.

```
23.130.2.11 #define PARAM_VECTOR( T, ID, DESC, ALIAS ) PARAM(std::vector<T>, ID, DESC, ALIAS, std::vector<T>(),
false)
```

Define a vector parameter.

The parameter can then be specified on the command line with **-ID=value**.

Parameters

<i>ID</i>	Name of the parameter.
<i>DESC</i>	Quick description of the parameter (1-2 sentences).
<i>ALIAS</i>	An alias for the parameter (one letter).
<i>DEF</i>	Default value of the parameter.

See also

mlpack::CLI (p. 184), **PROGRAM_INFO()** (p. 766)

Bug The **COUNTER** variable is used in most cases to guarantee a unique global identifier for options declared using the **PARAM_***() macros. However, not all compilers have this support—most notably, gcc < 4.3. In that case, the **LINE** macro is used as an attempt to get a unique global identifier, but collisions are still possible, and they produce bizarre error messages. See <http://mlpack.org/trac/ticket/74> for more information.

Definition at line 187 of file cli.hpp.

```
23.130.2.12 #define PARAM_VECTOR_REQ( T, ID, DESC, ALIAS )
```

Value:

```
PARAM(std::vector<T>, ID, DESC, \
    ALIAS, std::vector<T>(), true);
```

Define a required vector parameter.

The parameter must then be specified on the command line with **-ID=value**.

Parameters

<i>ID</i>	Name of the parameter.
<i>DESC</i>	Quick description of the parameter (1-2 sentences).
<i>ALIAS</i>	An alias for the parameter (one letter).

See also

mlpack::CLI (p. 184), **PROGRAM_INFO()** (p. 766)

Bug The **COUNTER** variable is used in most cases to guarantee a unique global identifier for options declared using the **PARAM_***() macros. However, not all compilers have this support—most notably, gcc < 4.3. In that case, the **LINE** macro is used as an attempt to get a unique global identifier, but collisions are still possible, and they produce bizarre error messages. See <http://mlpack.org/trac/ticket/74> for more information.

Definition at line 305 of file cli.hpp.

23.130.2.13 **#define PROGRAM_INFO(NAME, DESC)**

Value:

```
static mlpack::util::ProgramDoc \
    io_programdoc_dummy_object = mlpack::util::ProgramDoc(NAME, DESC);
```

Document an executable.

Only one instance of this macro should be present in your program! Therefore, use it in the main.cpp (or corresponding executable) in your program.

See also

mlpack::CLI (p. 184), **PARAM_FLAG()** (p. 762), **PARAM_INT()** (p. 763), **PARAM_DOUBLE()** (p. 761), **PARAM_STRING()** (p. 764), **PARAM_VECTOR()** (p. 765), **PARAM_INT_REQ()** (p. 763), **PARAM_DOUBLE_REQ()** (p. 761), **PARAM_STRING_REQ()** (p. 764), **PARAM_VECTOR_REQ()** (p. 765).

Parameters

<i>NAME</i>	Short string representing the name of the program.
<i>DESC</i>	Long string describing what the program does and possibly a simple usage example. Newlines should not be used here; this is taken care of by CLI (however, you can explicitly specify newlines to denote new paragraphs).

Definition at line 46 of file cli.hpp.

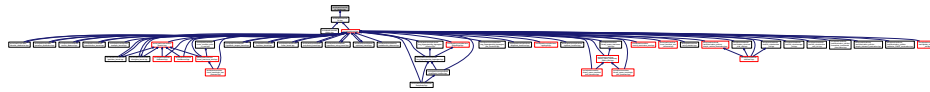
23.130.2.14 **#define TYPENAME(x) (std::string(typeid(x).name()))**

The TYPENAME macro is used internally to convert a type into a string.

Definition at line 364 of file cli.hpp.

23.131 src/mlpack/core/util/cli_deleter.hpp File Reference

This graph shows which files directly or indirectly include this file:



Classes

- class **mlpack::util::CLIDeleter**

*Extremely simple class whose only job is to delete the existing **CLI** (p. 184) object at the end of execution.*

Namespaces

- mlpack**
Linear algebra utility functions, generally performed on matrices or vectors.
- mlpack::util**

Variables

- static CLIDeleter **mlpack::util::cliDeleter**
Declare the deleter.

23.131.1 Detailed Description

Author

Ryan Curtin

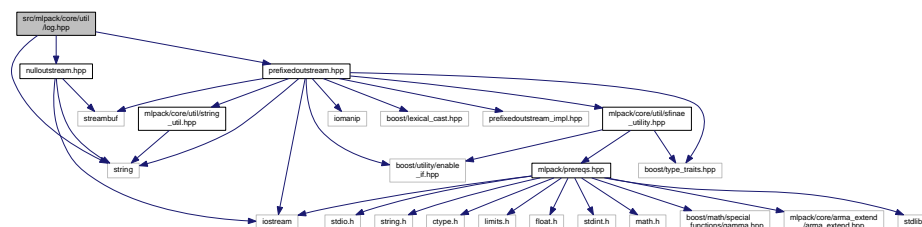
Definition of the CLIDeleter() class.

This file is part of mlpack 1.0.12.

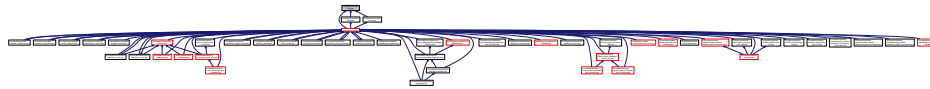
mlpack is free software; you may redistribute it and/or modify it under the terms of the 3-clause BSD license. You should have received a copy of the 3-clause BSD license along with mlpack. If not, see <http://www.opensource.org/licenses/BSD-3-Clause> for more information.

23.132 src/mlpack/core/util/log.hpp File Reference

Include dependency graph for log.hpp:



This graph shows which files directly or indirectly include this file:



Classes

- class **mlpack::Log**
Provides a convenient way to give formatted output.

Namespaces

- **mlpack**
Linear algebra utility functions, generally performed on matrices or vectors.

23.132.1 Detailed Description

Author

Matthew Amidon

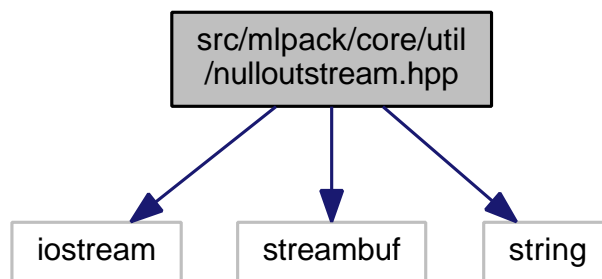
Definition of the Log class.

This file is part of mlpack 1.0.12.

mlpack is free software; you may redistribute it and/or modify it under the terms of the 3-clause BSD license. You should have received a copy of the 3-clause BSD license along with mlpack. If not, see <http://www.opensource.org/licenses/BSD-3-Clause> for more information.

23.133 src/mlpack/core/util/nullostream.hpp File Reference

Include dependency graph for nullostream.hpp:



This graph shows which files directly or indirectly include this file:



Classes

- class **mlpack::util::NullOutputStream**

Used for **Log::Debug** (p. 359) when not compiled with debugging symbols.

Namespaces

- mlpack**
Linear algebra utility functions, generally performed on matrices or vectors.
- mlpack::util**

23.133.1 Detailed Description

Author

Ryan Curtin
Matthew Amidon

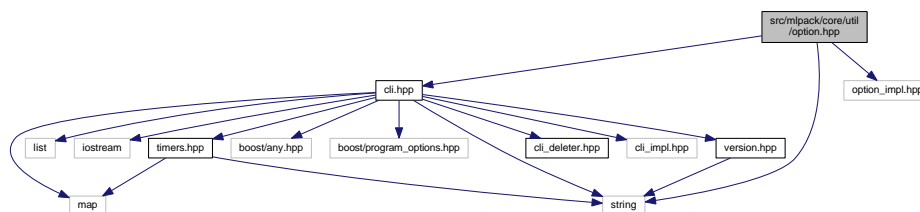
Definition of the NullOutputStream class.

This file is part of mlpack 1.0.12.

mlpack is free software; you may redistribute it and/or modify it under the terms of the 3-clause BSD license. You should have received a copy of the 3-clause BSD license along with mlpack. If not, see <http://www.opensource.org/licenses/BSD-3-Clause> for more information.

23.134 src/mlpack/core/util/option.hpp File Reference

Include dependency graph for option.hpp:



Classes

- class **mlpack::util::Option< N >**
A static object whose constructor registers a parameter with the **CLI** (p. 184) class.
- class **mlpack::util::ProgramDoc**
A static object whose constructor registers program documentation with the **CLI** (p. 184) class.

Namespaces

- mlpack**
Linear algebra utility functions, generally performed on matrices or vectors.
- mlpack::util**

Author

Ryan Curtin
Matthew Amidon

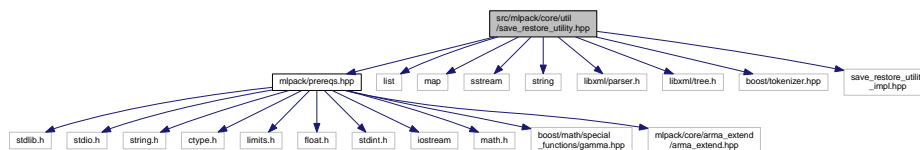
Declaration of the PrefixedOutputStream class.

This file is part of mlpack 1.0.12.

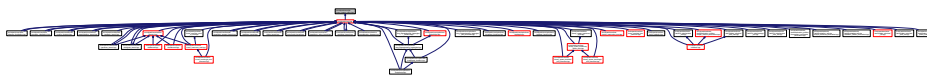
mlpack is free software; you may redistribute it and/or modify it under the terms of the 3-clause BSD license. You should have received a copy of the 3-clause BSD license along with mlpack. If not, see <http://www.opensource.org/licenses/BSD-3-Clause> for more information.

23.136 src/mlpack/core/util/save_restore_utility.hpp File Reference

Include dependency graph for save_restore_utility.hpp:



This graph shows which files directly or indirectly include this file:



Classes

- class **mlpack::util::SaveRestoreUtility**

Namespaces

- **mlpack**
Linear algebra utility functions, generally performed on matrices or vectors.
- **mlpack::util**

23.136.1 Detailed Description

Author

Neil Slagle

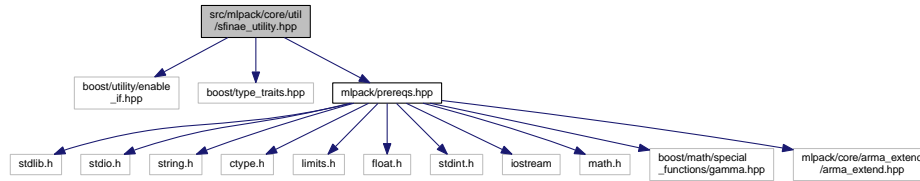
The SaveRestoreUtility provides helper functions in saving and restoring models. The current output file type is XML.

This file is part of mlpack 1.0.12.

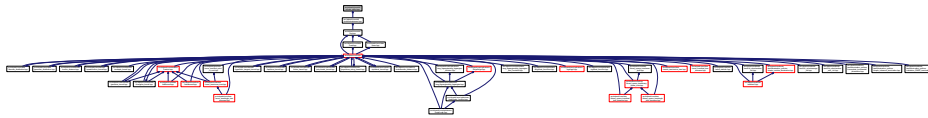
mlpack is free software; you may redistribute it and/or modify it under the terms of the 3-clause BSD license. You should have received a copy of the 3-clause BSD license along with mlpack. If not, see <http://www.opensource.org/licenses/BSD-3-Clause> for more information.

23.137 src/mlpack/core/util/sfinae_utility.hpp File Reference

Include dependency graph for sfinae_utility.hpp:



This graph shows which files directly or indirectly include this file:



Macros

- `#define HAS_MEM_FUNC(FUNC, NAME)`

23.137.1 Detailed Description

Author

Trironk Kiatkungwanglai

This file contains macro utilities for the SFINAE Paradigm. These utilities determine if classes passed in as template parameters contain members at compile time, which is useful for changing functionality depending on what operations an object is capable of performing.

This file is part of mlpack 1.0.12.

mlpack is free software; you may redistribute it and/or modify it under the terms of the 3-clause BSD license. You should have received a copy of the 3-clause BSD license along with mlpack. If not, see <http://www.opensource.org/licenses/BSD-3-Clause> for more information.

23.137.2 Macro Definition Documentation

23.137.2.1 `#define HAS_MEM_FUNC(FUNC, NAME)`

Value:

```

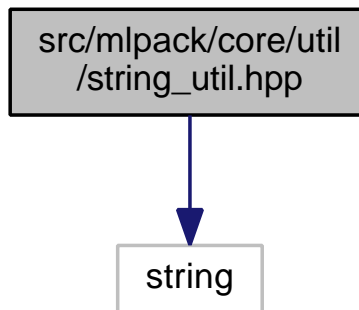
template<typename T, typename sig>
struct NAME {
    typedef char yes[1];
    typedef char no [2];
    template<typename U, U> struct type_check;
    template<typename _1> static yes &chk(type_check<sig, &_1::FUNC> *);
    template<typename _> static no  &chk(...);
    static bool const value = sizeof(chk<T>(0)) == sizeof(yes);
};

```

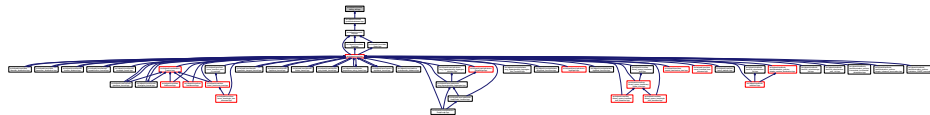
Definition at line 43 of file sfinae_utility.hpp.

23.138 src/mlpack/core/util/string_util.hpp File Reference

Include dependency graph for string_util.hpp:



This graph shows which files directly or indirectly include this file:



Namespaces

- **mlpack**
Linear algebra utility functions, generally performed on matrices or vectors.
- **mlpack::util**

Functions

- `std::string mlpack::util::Indent` (`std::string` input, `const size_t howManyTabs=1`)
A utility function that replaces all all newlines with a number of spaces depending on the indentation level.

23.138.1 Detailed Description

Author

Trironk Kiatkungwanglai
Ryan Birmingham

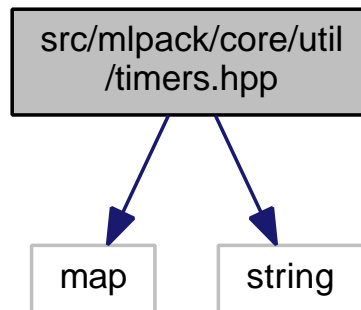
Declares methods that are useful for writing formatting output.

This file is part of mlpack 1.0.12.

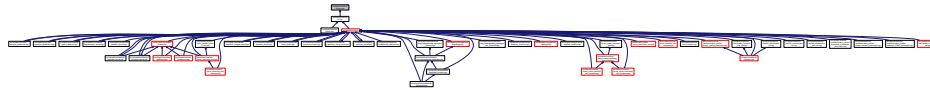
mlpack is free software; you may redistribute it and/or modify it under the terms of the 3-clause BSD license. You should have received a copy of the 3-clause BSD license along with mlpack. If not, see <http://www.opensource.org/licenses/BSD-3-Clause> for more information.

23.139 src/mlpack/core/util/timers.hpp File Reference

Include dependency graph for timers.hpp:



This graph shows which files directly or indirectly include this file:



Classes

- class **mlpack::Timer**
The timer class provides a way for MLPACK methods to be timed.
- class **mlpack::Timers**

Namespaces

- **mlpack**
Linear algebra utility functions, generally performed on matrices or vectors.

23.139.1 Detailed Description

Author

Matthew Amidon
Marcus Edel

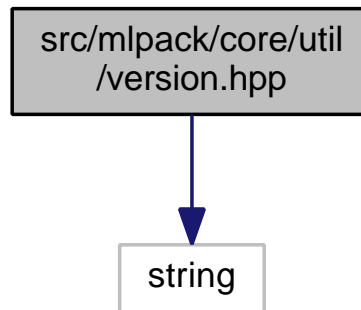
Timers for MLPACK.

This file is part of mlpack 1.0.12.

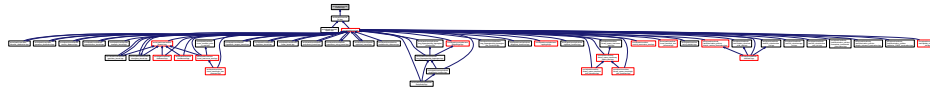
mlpack is free software; you may redistribute it and/or modify it under the terms of the 3-clause BSD license. You should have received a copy of the 3-clause BSD license along with mlpack. If not, see <http://www.opensource.org/licenses/BSD-3-Clause> for more information.

23.140 src/mlpack/core/util/version.hpp File Reference

Include dependency graph for version.hpp:



This graph shows which files directly or indirectly include this file:



Namespaces

- **mlpack**
Linear algebra utility functions, generally performed on matrices or vectors.
- **mlpack::util**

Macros

- `#define __MLPACK_VERSION_MAJOR 1`
- `#define __MLPACK_VERSION_MINOR 0`
- `#define __MLPACK_VERSION_PATCH 12`

Functions

- `std::string mlpack::util::GetVersion ()`
This will return either "mlpack x.y.z" or "mlpack trunk-rXXXXX" depending on whether or not this is a stable version of mlpack or an svn revision.

23.140.1 Macro Definition Documentation

23.140.1.1 `#define __MLPACK_VERSION_MAJOR 1`

Definition at line 21 of file `version.hpp`.

23.140.1.2 `#define __MLPACK_VERSION_MINOR 0`

Definition at line 22 of file version.hpp.

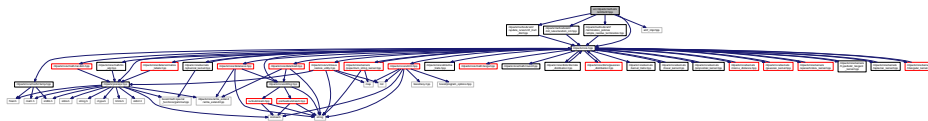
23.140.1.3 `#define __MLPACK_VERSION_PATCH 12`

Definition at line 23 of file version.hpp.

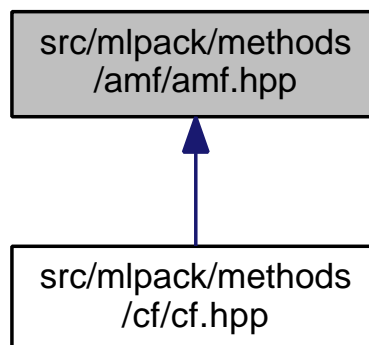
23.141 doc/guide/version.hpp File Reference

23.142 src/mlpack/methods/amf/amf.hpp File Reference

Include dependency graph for amf.hpp:



This graph shows which files directly or indirectly include this file:



Classes

- class **mlpack::amf::AMF**< **TerminationPolicyType**, **InitializationRuleType**, **UpdateRuleType** >
*This class implements **AMF** (p. 123) (alternating matrix factorization) on the given matrix V .*

Namespaces

- **mlpack**
Linear algebra utility functions, generally performed on matrices or vectors.
- **mlpack::amf**
Alternating Matrix Factorization.

23.142.1 Detailed Description

Author

Sumedh Ghaisas
Mohan Rajendran
Ryan Curtin

Alternating Matrix Factorization

The AMF (alternating matrix factorization) class, from which more commonly known techniques such as incremental SVD, NMF, and batch-learning SVD can be derived.

This file is part of mlpack 1.0.12.

mlpack is free software; you may redistribute it and/or modify it under the terms of the 3-clause BSD license. You should have received a copy of the 3-clause BSD license along with mlpack. If not, see <http://www.opensource.org/licenses/BSD-3-Clause> for more information.

23.143 src/mlpack/methods/amf/init_rules/average_init.hpp File Reference

Include dependency graph for average_init.hpp:



Classes

- class **mlpack::amf::AverageInitialization**

This initialization rule initializes matrix W and H to root of average of V with uniform noise.

Namespaces

- **mlpack**

Linear algebra utility functions, generally performed on matrices or vectors.

- **mlpack::amf**

Alternating Matrix Factorization.

23.144 src/mlpack/methods/amf/init_rules/random_acol_init.hpp File Reference

Include dependency graph for random_acol_init.hpp:



Classes

- class **mlpack::amf::RandomAcolInitialization**< **p** >

*This class initializes the W matrix of the **AMF** (p. 123) algorithm by averaging p randomly chosen columns of V .*

Namespaces

- **mlpack**

Linear algebra utility functions, generally performed on matrices or vectors.

- **mlpack::amf**

Alternating Matrix Factorization.

23.144.1 Detailed Description

Author

Mohan Rajendran

Initialization rule for Alternating Matrix Factorization.

This file is part of mlpack 1.0.12.

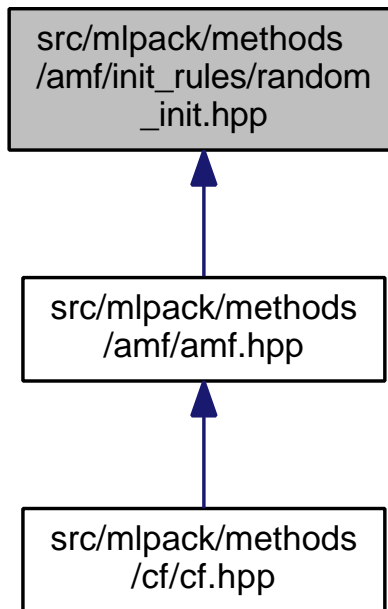
mlpack is free software; you may redistribute it and/or modify it under the terms of the 3-clause BSD license. You should have received a copy of the 3-clause BSD license along with mlpack. If not, see <http://www.opensource.org/licenses/BSD-3-Clause> for more information.

23.145 src/mlpack/methods/amf/init_rules/random_init.hpp File Reference

Include dependency graph for random_init.hpp:



This graph shows which files directly or indirectly include this file:



Classes

- class **mlpack::amf::RandomInitialization**

Namespaces

- **mlpack**

Linear algebra utility functions, generally performed on matrices or vectors.

- **mlpack::amf**

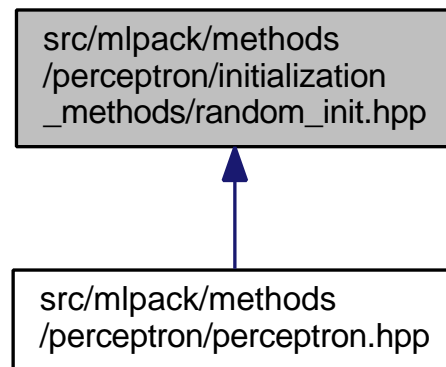
Alternating Matrix Factorization.

23.146 src/mlpack/methods/perceptron/initialization_methods/random_init.hpp File Reference

Include dependency graph for random_init.hpp:



This graph shows which files directly or indirectly include this file:



Classes

- class **mlpack::perceptron::RandomInitialization**

This class is used to initialize weights for the weightVectors matrix in a random manner.

Namespaces

- **mlpack**

Linear algebra utility functions, generally performed on matrices or vectors.

- **mlpack::perceptron**

23.147 src/mlpack/methods/amf/termination_policies/complete_incremental_termination.hpp File Reference

Classes

- class **mlpack::amf::CompleteIncrementalTermination**< **TerminationPolicy** >

Namespaces

- **mlpack**

Linear algebra utility functions, generally performed on matrices or vectors.

- **mlpack::amf**

Alternating Matrix Factorization.

23.148 src/mlpack/methods/amf/termination_policies/incomplete_incremental_termination.hpp File Reference

Include dependency graph for incomplete_incremental_termination.hpp:



Classes

- class `mlpack::amf::IncompleteIncrementalTermination`< `TerminationPolicy` >

Namespaces

- `mlpack`
Linear algebra utility functions, generally performed on matrices or vectors.
- `mlpack::amf`
Alternating Matrix Factorization.

23.148.1 Detailed Description

Author

Sumedh Ghaisas

This file is part of mlpack 1.0.12.

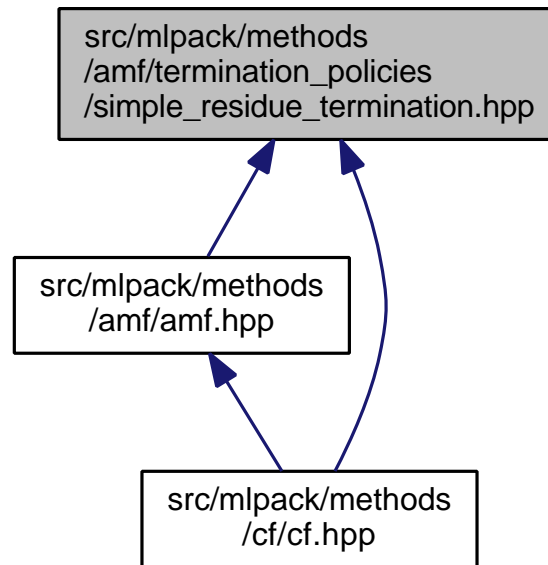
mlpack is free software; you may redistribute it and/or modify it under the terms of the 3-clause BSD license. You should have received a copy of the 3-clause BSD license along with mlpack. If not, see <http://www.opensource.org/licenses/BSD-3-Clause> for more information.

23.149 src/mlpack/methods/amf/termination_policies/simple_residue_termination.hpp File Reference

Include dependency graph for simple_residue_termination.hpp:



This graph shows which files directly or indirectly include this file:



Classes

- class **mlpack::amf::SimpleResidueTermination**

This class implements a simple residue-based termination policy.

Namespaces

- **mlpack**

Linear algebra utility functions, generally performed on matrices or vectors.

- **mlpack::amf**

Alternating Matrix Factorization.

23.149.1 Detailed Description

Author

Sumedh Ghaisas

Termination policy used in AMF (Alternating Matrix Factorization).

This file is part of mlpack 1.0.12.

mlpack is free software; you may redistribute it and/or modify it under the terms of the 3-clause BSD license. You should have received a copy of the 3-clause BSD license along with mlpack. If not, see <http://www.opensource.org/licenses/BSD-3-Clause> for more information.

Author

Sumedh Ghaisas

This file is part of mlpack 1.0.12.

mlpack is free software; you may redistribute it and/or modify it under the terms of the 3-clause BSD license. You should have received a copy of the 3-clause BSD license along with mlpack. If not, see <http://www.opensource.org/licenses/BSD-3-Clause> for more information.

23.150 src/mlpack/methods/amf/termination_policies/simple_tolerance_termination.hpp File Reference

Include dependency graph for simple_tolerance_termination.hpp:

**Classes**

- class **mlpack::amf::SimpleToleranceTermination**< **MatType** >

This class implements residue tolerance termination policy.

Namespaces

- **mlpack**
Linear algebra utility functions, generally performed on matrices or vectors.
- **mlpack::amf**
Alternating Matrix Factorization.

23.150.1 Detailed Description

Author

Sumedh Ghaisas

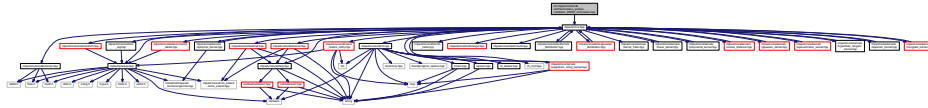
Termination policy used in AMF (Alternating Matrix Factorization).

This file is part of mlpack 1.0.12.

mlpack is free software; you may redistribute it and/or modify it under the terms of the 3-clause BSD license. You should have received a copy of the 3-clause BSD license along with mlpack. If not, see <http://www.opensource.org/licenses/BSD-3-Clause> for more information.

23.151 src/mlpack/methods/amf/termination_policies/validation_RMSE_termination.hpp File Reference

Include dependency graph for validation_RMSE_termination.hpp:



Classes

- class `mlpack::amf::ValidationRMSETermination< MatType >`

Namespaces

- `mlpack`
Linear algebra utility functions, generally performed on matrices or vectors.
- `mlpack::amf`
Alternating Matrix Factorization.

23.151.1 Detailed Description

Author

Sumedh Ghaisas

Termination policy that checks validation RMSE.

This file is part of mlpack 1.0.12.

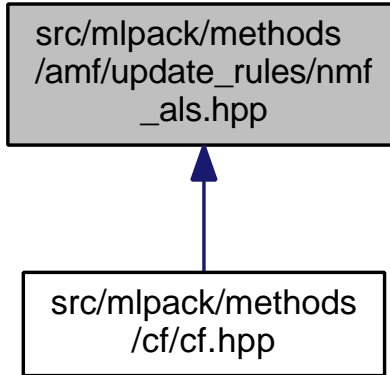
mlpack is free software; you may redistribute it and/or modify it under the terms of the 3-clause BSD license. You should have received a copy of the 3-clause BSD license along with mlpack. If not, see <http://www.opensource.org/licenses/BSD-3-Clause> for more information.

23.152 src/mlpack/methods/amf/update_rules/nmf_als.hpp File Reference

Include dependency graph for nmf_als.hpp:



This graph shows which files directly or indirectly include this file:



Classes

- class **mlpack::amf::NMFALSUpdate**

This class implements a method titled 'Alternating Least Squares' described in the paper 'Positive Matrix Factorization: A Non-negative Factor Model with Optimal Utilization of Error Estimates of Data Values' by P Paatero and U Tapper.

Namespaces

- **mlpack**

Linear algebra utility functions, generally performed on matrices or vectors.

- **mlpack::amf**

Alternating Matrix Factorization.

23.152.1 Detailed Description

Author

Mohan Rajendran

Update rules for the Non-negative Matrix Factorization.

This file is part of mlpack 1.0.12.

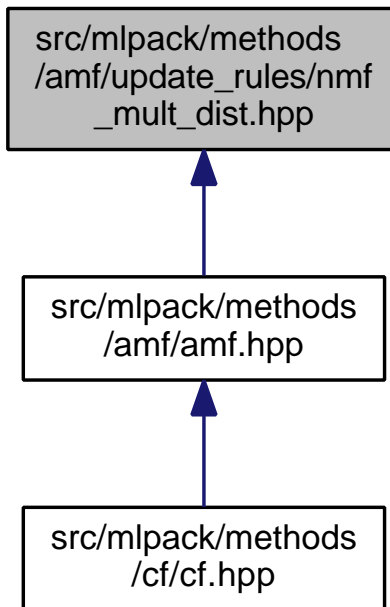
mlpack is free software; you may redistribute it and/or modify it under the terms of the 3-clause BSD license. You should have received a copy of the 3-clause BSD license along with mlpack. If not, see <http://www.opensource.org/licenses/BSD-3-Clause> for more information.

23.153 src/mlpack/methods/amf/update_rules/nmf_mult_dist.hpp File Reference

Include dependency graph for nmf_mult_dist.hpp:



This graph shows which files directly or indirectly include this file:



Classes

- class **mlpack::amf::NMFMultiplicativeDistanceUpdate**
The multiplicative distance update rules for matrices W and H .

Namespaces

- **mlpack**
Linear algebra utility functions, generally performed on matrices or vectors.
- **mlpack::amf**
Alternating Matrix Factorization.

23.153.1 Detailed Description

Author

Mohan Rajendran

Update rules for the Non-negative Matrix Factorization.

This file is part of mlpack 1.0.12.

mlpack is free software; you may redistribute it and/or modify it under the terms of the 3-clause BSD license. You should have received a copy of the 3-clause BSD license along with mlpack. If not, see <http://www.opensource.org/licenses/BSD-3-Clause> for more information.

23.154 src/mlpack/methods/amf/update_rules/nmf_mult_div.hpp File Reference

Include dependency graph for nmf_mult_div.hpp:



Classes

- class **mlpack::amf::NMFMultiplicativeDivergenceUpdate**

This follows a method described in the paper 'Algorithms for Non-negative Matrix Factorization' by D.

Namespaces

- **mlpack**

Linear algebra utility functions, generally performed on matrices or vectors.

- **mlpack::amf**

Alternating Matrix Factorization.

23.155 src/mlpack/methods/amf/update_rules/svd_batch_learning.hpp File Reference

Include dependency graph for svd_batch_learning.hpp:



Classes

- class **mlpack::amf::SVDBatchLearning**

This class implements SVD batch learning with momentum.

Namespaces

- **mlpack**

Linear algebra utility functions, generally performed on matrices or vectors.

- **mlpack::amf**

Alternating Matrix Factorization.

Functions

- `template<>`
`void mlpack::amf::SVDBatchLearning::HUpdate< arma::sp_mat > (const arma::sp_mat &V, const arma::mat &W, arma::mat &H)`
- `template<>`
`void mlpack::amf::SVDBatchLearning::WUpdate< arma::sp_mat > (const arma::sp_mat &V, arma::mat &W, const arma::mat &H)`

TODO : Merge this template specialized function for sparse matrix using common row_col_iterator.

23.156 src/mlpack/methods/amf/update_rules/svd_complete_incremental_learning.hpp File Reference

Include dependency graph for `svd_complete_incremental_learning.hpp`:



Classes

- class `mlpack::amf::SVDCompleteIncrementalLearning< MatType >`
- class `mlpack::amf::SVDCompleteIncrementalLearning< arma::sp_mat >`

Namespaces

- `mlpack`
Linear algebra utility functions, generally performed on matrices or vectors.
- `mlpack::amf`
Alternating Matrix Factorization.

23.157 src/mlpack/methods/amf/update_rules/svd_incomplete_incremental_learning.hpp File Reference

Classes

- class `mlpack::amf::SVDIncompleteIncrementalLearning`

Namespaces

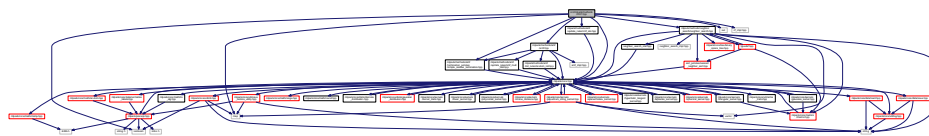
- `mlpack`
Linear algebra utility functions, generally performed on matrices or vectors.
- `mlpack::amf`
Alternating Matrix Factorization.

Functions

- `template<>`
`void mlpack::amf::SVDIncompleteIncrementalLearning::HUpdate< arma::sp_mat > (const arma::sp_mat &V, const arma::mat &W, arma::mat &H)`
- `template<>`
`void mlpack::amf::SVDIncompleteIncrementalLearning::WUpdate< arma::sp_mat > (const arma::sp_mat &V, arma::mat &W, const arma::mat &H)`

23.158 src/mlpack/methods/cf/cf.hpp File Reference

Include dependency graph for cf.hpp:



Classes

- class `mlpack::cf::CF< FactorizerType >`
This class implements Collaborative Filtering (CF (p. 177)).

Namespaces

- `mlpack`
Linear algebra utility functions, generally performed on matrices or vectors.
- `mlpack::cf`
Collaborative filtering.

23.158.1 Detailed Description

Author

Mudit Raj Gupta

Collaborative filtering.

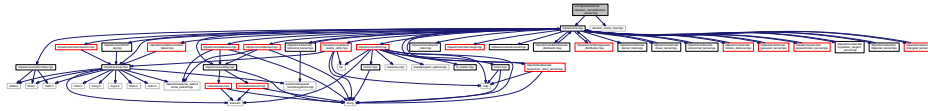
Defines the CF class to perform collaborative filtering on the specified data set using alternating least squares (ALS).

This file is part of mlpack 1.0.12.

mlpack is free software; you may redistribute it and/or modify it under the terms of the 3-clause BSD license. You should have received a copy of the 3-clause BSD license along with mlpack. If not, see <http://www.opensource.org/licenses/BSD-3-Clause> for more information.

23.159 src/mlpack/methods/decision_stump/decision_stump.hpp File Reference

Include dependency graph for decision_stump.hpp:



Classes

- class **mlpack::decision_stump::DecisionStump**< **MatType** >
This class implements a decision stump.

Namespaces

- **mlpack**
Linear algebra utility functions, generally performed on matrices or vectors.
- **mlpack::decision_stump**

23.159.1 Detailed Description

Author

Udit Saxena

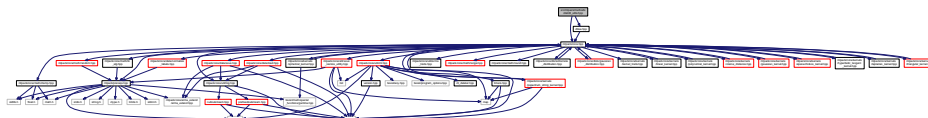
Definition of decision stumps.

This file is part of mlpack 1.0.12.

mlpack is free software; you may redistribute it and/or modify it under the terms of the 3-clause BSD license. You should have received a copy of the 3-clause BSD license along with mlpack. If not, see <http://www.opensource.org/licenses/BSD-3-Clause> for more information.

23.160 src/mlpack/methods/det/dt_utils.hpp File Reference

Include dependency graph for dt_utils.hpp:



Namespaces

- **mlpack**
Linear algebra utility functions, generally performed on matrices or vectors.
- **mlpack::det**
Density Estimation Trees.

Functions

- void **mlpack::det::PrintLeafMembership** (DTree *dtree, const arma::mat &data, const arma::Mat< size_t > &labels, const size_t numClasses, const std::string leafClassMembershipFile="")
Print the membership of leaves of a density estimation tree given the labels and number of classes.
- void **mlpack::det::PrintVariableImportance** (const DTree *dtree, const std::string viFile="")
Print the variable importance of each dimension of a density estimation tree.
- DTree * **mlpack::det::Trainer** (arma::mat &dataset, const size_t folds, const bool useVolumeReg=false, const size_t maxLeafSize=10, const size_t minLeafSize=5, const std::string unprunedTreeOutput="")
Train the optimal decision tree using cross-validation with the given number of folds.

23.160.1 Detailed Description

Author

Parikshit Ram (pram@cc.gatech.edu)

This file implements functions to perform different tasks with the Density Tree class.

This file is part of mlpack 1.0.12.

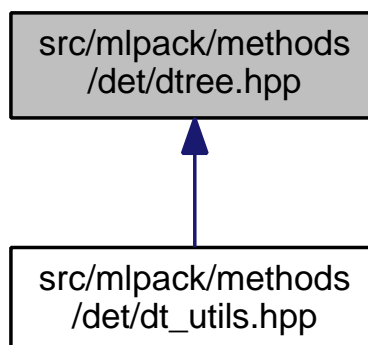
mlpack is free software; you may redistribute it and/or modify it under the terms of the 3-clause BSD license. You should have received a copy of the 3-clause BSD license along with mlpack. If not, see <http://www.opensource.org/licenses/BSD-3-Clause> for more information.

23.161 src/mlpack/methods/det/dtree.hpp File Reference

Include dependency graph for dtree.hpp:



This graph shows which files directly or indirectly include this file:



Classes

- class **mlpack::det::DTree**

A density estimation tree is similar to both a decision tree and a space partitioning tree (like a kd-tree).

Namespaces

- **mlpack**

Linear algebra utility functions, generally performed on matrices or vectors.

- **mlpack::det**

Density Estimation Trees.

23.161.1 Detailed Description

Author

Parikshit Ram (pram@cc.gatech.edu)

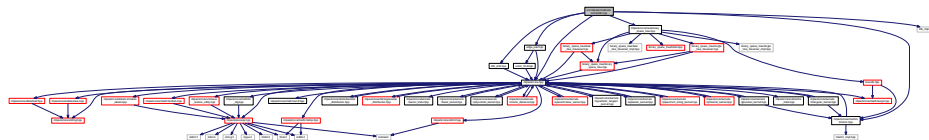
Density Estimation Tree class

This file is part of mlpack 1.0.12.

mlpack is free software; you may redistribute it and/or modify it under the terms of the 3-clause BSD license. You should have received a copy of the 3-clause BSD license along with mlpack. If not, see <http://www.opensource.org/licenses/BSD-3-Clause> for more information.

23.162 src/mlpack/methods/emst/dtb.hpp File Reference

Include dependency graph for dtb.hpp:



Classes

- class **mlpack::emst::DualTreeBoruvka< MetricType, TreeType >**
Performs the MST calculation using the Dual-Tree Boruvka algorithm, using any type of tree.
- struct **mlpack::emst::DualTreeBoruvka< MetricType, TreeType >::SortEdgesHelper**
For sorting the edge list after the computation.

Namespaces

- **mlpack**

Linear algebra utility functions, generally performed on matrices or vectors.

- **mlpack::emst**

Euclidean Minimum Spanning Trees.

23.162.1 Detailed Description

Author

Bill March (march@gatech.edu)

Contains an implementation of the DualTreeBoruvka algorithm for finding a Euclidean Minimum Spanning Tree using the kd-tree data structure.

```
@inproceedings{
  author = {March, W.B., Ram, P., and Gray, A.G.},
  title = {{Fast Euclidean Minimum Spanning Tree: Algorithm, Analysis,
    Applications.}},
  booktitle = {Proceedings of the 16th ACM SIGKDD International Conference
    on Knowledge Discovery and Data Mining}
  series = {KDD 2010},
  year = {2010}
}
```

This file is part of mlpack 1.0.12.

mlpack is free software; you may redistribute it and/or modify it under the terms of the 3-clause BSD license. You should have received a copy of the 3-clause BSD license along with mlpack. If not, see <http://www.opensource.org/licenses/BSD-3-Clause> for more information.

Author

Bill March (march@gatech.edu)

Tree traverser rules for the DualTreeBoruvka algorithm.

This file is part of mlpack 1.0.12.

mlpack is free software; you may redistribute it and/or modify it under the terms of the 3-clause BSD license. You should have received a copy of the 3-clause BSD license along with mlpack. If not, see <http://www.opensource.org/licenses/BSD-3-Clause> for more information.

Author

Bill March (march@gatech.edu)

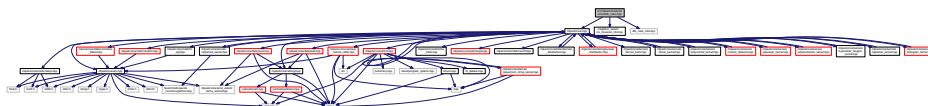
DTBStat is the StatisticType used by trees when performing EMST.

This file is part of mlpack 1.0.12.

mlpack is free software; you may redistribute it and/or modify it under the terms of the 3-clause BSD license. You should have received a copy of the 3-clause BSD license along with mlpack. If not, see <http://www.opensource.org/licenses/BSD-3-Clause> for more information.

23.163 src/mlpack/methods/emst/dtb_rules.hpp File Reference

Include dependency graph for dtb_rules.hpp:



Classes

- class **mlpack::emst::DTBRules**< **MetricType**, **TreeType** >

Namespaces

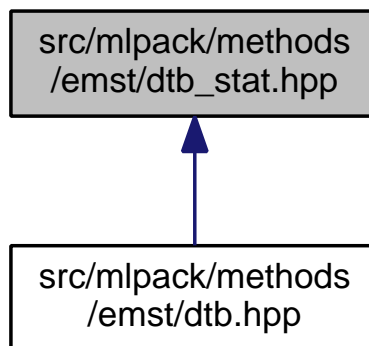
- **mlpack**
Linear algebra utility functions, generally performed on matrices or vectors.
- **mlpack::emst**
Euclidean Minimum Spanning Trees.

23.164 src/mlpack/methods/emst/dtb_stat.hpp File Reference

Include dependency graph for dtb_stat.hpp:



This graph shows which files directly or indirectly include this file:



Classes

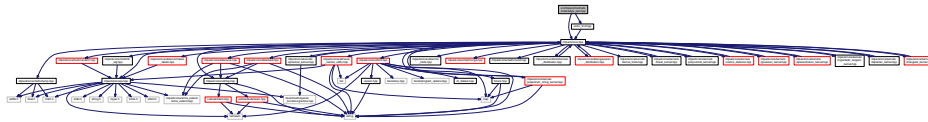
- class **mlpack::emst::DTBStat**
A statistic for use with MLPACK trees, which stores the upper bound on distance to nearest neighbors and the component which this node belongs to.

Namespaces

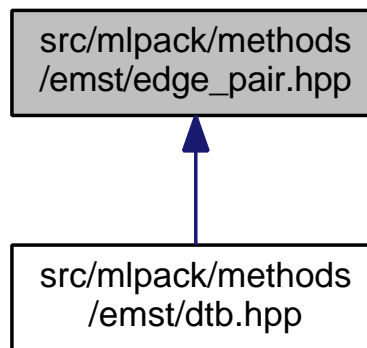
- **mlpack**
Linear algebra utility functions, generally performed on matrices or vectors.
- **mlpack::emst**
Euclidean Minimum Spanning Trees.

23.165 src/mlpack/methods/emst/edge_pair.hpp File Reference

Include dependency graph for edge_pair.hpp:



This graph shows which files directly or indirectly include this file:



Classes

- class **mlpack::emst::EdgePair**
An edge pair is simply two indices and a distance.

Namespaces

- **mlpack**
Linear algebra utility functions, generally performed on matrices or vectors.
- **mlpack::emst**
Euclidean Minimum Spanning Trees.

23.165.1 Detailed Description

Author

Bill March (march@gatech.edu)

This file contains utilities necessary for all of the minimum spanning tree algorithms.

This file is part of mlpack 1.0.12.

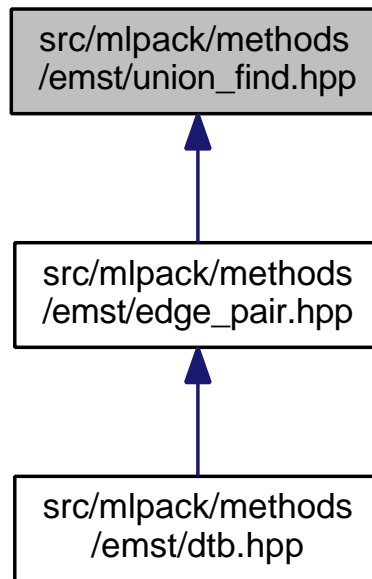
mlpack is free software; you may redistribute it and/or modify it under the terms of the 3-clause BSD license. You should have received a copy of the 3-clause BSD license along with mlpack. If not, see <http://www.opensource.org/licenses/BSD-3-Clause> for more information.

23.166 src/mlpack/methods/emst/union_find.hpp File Reference

Include dependency graph for union_find.hpp:



This graph shows which files directly or indirectly include this file:



Classes

- class **mlpack::emst::UnionFind**
A Union-Find data structure.

Namespaces

- **mlpack**
Linear algebra utility functions, generally performed on matrices or vectors.
- **mlpack::emst**
Euclidean Minimum Spanning Trees.

23.166.1 Detailed Description

Author

Bill March (march@gatech.edu)

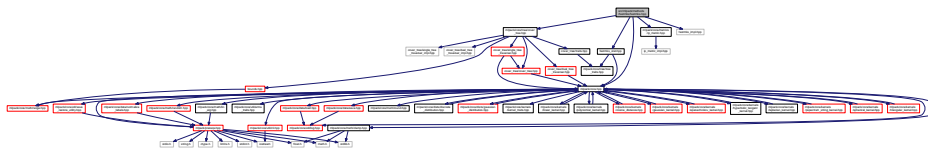
Implements a union-find data structure. This structure tracks the components of a graph. Each point in the graph is initially in its own component. Calling `unionfind.Union(x, y)` unites the components indexed by `x` and `y`. `unionfind.Find(x)` returns the index of the component containing point `x`.

This file is part of mlpack 1.0.12.

mlpack is free software; you may redistribute it and/or modify it under the terms of the 3-clause BSD license. You should have received a copy of the 3-clause BSD license along with mlpack. If not, see <http://www.opensource.org/licenses/BSD-3-Clause> for more information.

23.167 src/mlpack/methods/fastmks/fastmks.hpp File Reference

Include dependency graph for fastmks.hpp:



Classes

- `class mlpack::fastmks::FastMKS< KernelType, TreeType >`

An implementation of fast exact max-kernel search.

Namespaces

- **mlpack**

Linear algebra utility functions, generally performed on matrices or vectors.

- **mlpack::fastmks**

Fast max-kernel search.

23.167.1 Detailed Description

Author

Ryan Curtin

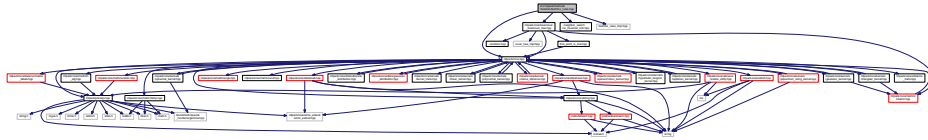
Definition of the FastMKS class, which implements fast exact max-kernel search.

This file is part of mlpack 1.0.12.

mlpack is free software; you may redistribute it and/or modify it under the terms of the 3-clause BSD license. You should have received a copy of the 3-clause BSD license along with mlpack. If not, see <http://www.opensource.org/licenses/BSD-3-Clause> for more information.

23.168 src/mlpack/methods/fastmks/fastmks_rules.hpp File Reference

Include dependency graph for fastmks_rules.hpp:



Classes

- class **mlpack::fastmks::FastMKSRules**< **KernelType**, **TreeType** >

*The base case and pruning rules for **FastMKS** (p. 247) (fast max-kernel search).*

Namespaces

- **mlpack**

Linear algebra utility functions, generally performed on matrices or vectors.

- **mlpack::fastmks**

Fast max-kernel search.

23.168.1 Detailed Description

Author

Ryan Curtin

Rules for the single or dual tree traversal for fast max-kernel search.

This file is part of mlpack 1.0.12.

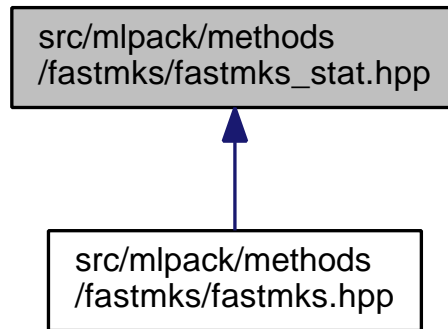
mlpack is free software; you may redistribute it and/or modify it under the terms of the 3-clause BSD license. You should have received a copy of the 3-clause BSD license along with mlpack. If not, see <http://www.opensource.org/licenses/BSD-3-Clause> for more information.

23.169 src/mlpack/methods/fastmks/fastmks_stat.hpp File Reference

Include dependency graph for fastmks_stat.hpp:



This graph shows which files directly or indirectly include this file:



Classes

- class **mlpack::fastmks::FastMKStat**

The statistic used in trees with **FastMKS** (p. 247).

Namespaces

- **mlpack**

Linear algebra utility functions, generally performed on matrices or vectors.

- **mlpack::fastmks**

Fast max-kernel search.

23.169.1 Detailed Description

Author

Ryan Curtin

The statistic used in trees with FastMKS.

This file is part of mlpack 1.0.12.

mlpack is free software; you may redistribute it and/or modify it under the terms of the 3-clause BSD license. You should have received a copy of the 3-clause BSD license along with mlpack. If not, see <http://www.opensource.org/licenses/BSD-3-Clause> for more information.

23.170 src/mlpack/methods/gmm/diagonal_constraint.hpp File Reference

Include dependency graph for `diagonal_constraint.hpp`:



Classes

- class **mlpack::gmm::DiagonalConstraint**

Force a covariance matrix to be diagonal.

Namespaces

- **mlpack**

Linear algebra utility functions, generally performed on matrices or vectors.

- **mlpack::gmm**

Gaussian Mixture Models.

23.170.1 Detailed Description

Author

Ryan Curtin

Constrain a covariance matrix to be diagonal.

This file is part of mlpack 1.0.12.

mlpack is free software; you may redistribute it and/or modify it under the terms of the 3-clause BSD license. You should have received a copy of the 3-clause BSD license along with mlpack. If not, see <http://www.opensource.org/licenses/BSD-3-Clause> for more information.

23.171 src/mlpack/methods/gmm/eigenvalue_ratio_constraint.hpp File Reference

Include dependency graph for eigenvalue_ratio_constraint.hpp:



Classes

- class **mlpack::gmm::EigenvalueRatioConstraint**

Given a vector of eigenvalue ratios, ensure that the covariance matrix always has those eigenvalue ratios.

Namespaces

- **mlpack**

Linear algebra utility functions, generally performed on matrices or vectors.

- **mlpack::gmm**

Gaussian Mixture Models.

23.171.1 Detailed Description

Author

Ryan Curtin

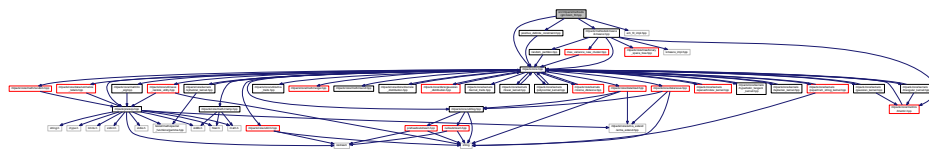
Constrain a covariance matrix to have a certain ratio of eigenvalues.

This file is part of mlpack 1.0.12.

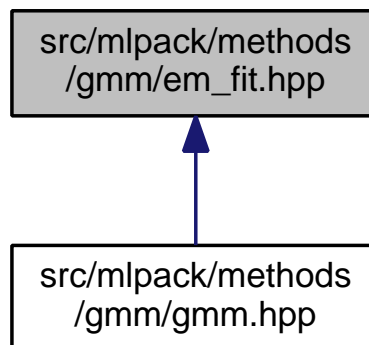
mlpack is free software; you may redistribute it and/or modify it under the terms of the 3-clause BSD license. You should have received a copy of the 3-clause BSD license along with mlpack. If not, see <http://www.opensource.org/licenses/BSD-3-Clause> for more information.

23.172 src/mlpack/methods/gmm/em_fit.hpp File Reference

Include dependency graph for em_fit.hpp:



This graph shows which files directly or indirectly include this file:



Classes

- class **mlpack::gmm::EMFit**< **InitialClusteringType**, **CovarianceConstraintPolicy** >
*This class contains methods which can fit a **GMM** (p. 271) to observations using the EM algorithm.*

Namespaces

- **mlpack**
Linear algebra utility functions, generally performed on matrices or vectors.
- **mlpack::gmm**
Gaussian Mixture Models.

23.172.1 Detailed Description

Author

Ryan Curtin

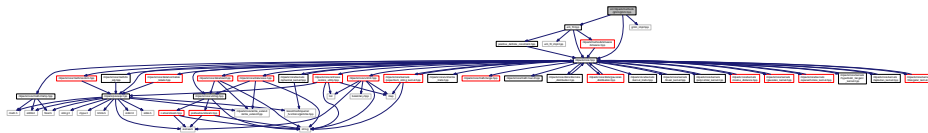
Utility class to fit a GMM using the EM algorithm. Used by `GMM::Estimate<>()`.

This file is part of mlpack 1.0.12.

mlpack is free software; you may redistribute it and/or modify it under the terms of the 3-clause BSD license. You should have received a copy of the 3-clause BSD license along with mlpack. If not, see <http://www.opensource.org/licenses/BSD-3-Clause> for more information.

23.173 src/mlpack/methods/gmm/gmm.hpp File Reference

Include dependency graph for gmm.hpp:



Classes

- class **mlpack::gmm::GMM< FittingType >**
A Gaussian Mixture Model (*GMM* (p. 271)).

Namespaces

- **mlpack**
Linear algebra utility functions, generally performed on matrices or vectors.
- **mlpack::gmm**
Gaussian Mixture Models.

23.173.1 Detailed Description

Author

Parikshit Ram (pram@cc.gatech.edu)

Defines a Gaussian Mixture model and estimates the parameters of the model

This file is part of mlpack 1.0.12.

mlpack is free software; you may redistribute it and/or modify it under the terms of the 3-clause BSD license. You should have received a copy of the 3-clause BSD license along with mlpack. If not, see <http://www.opensource.org/licenses/BSD-3-Clause> for more information.

23.174 src/mlpack/methods/gmm/no_constraint.hpp File Reference

Include dependency graph for no_constraint.hpp:



Classes

- class **mlpack::gmm::NoConstraint**
This class enforces no constraint on the covariance matrix.

Namespaces

- **mlpack**
Linear algebra utility functions, generally performed on matrices or vectors.
- **mlpack::gmm**
Gaussian Mixture Models.

23.174.1 Detailed Description

Author

Ryan Curtin

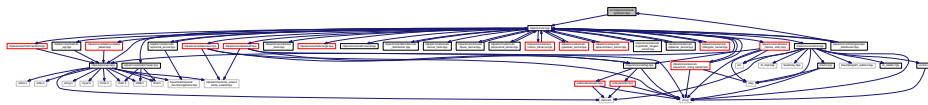
No constraint on the covariance matrix.

This file is part of mlpack 1.0.12.

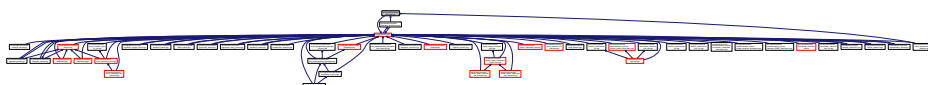
mlpack is free software; you may redistribute it and/or modify it under the terms of the 3-clause BSD license. You should have received a copy of the 3-clause BSD license along with mlpack. If not, see <http://www.opensource.org/licenses/BSD-3-Clause> for more information.

23.175 src/mlpack/methods/gmm/phi.hpp File Reference

Include dependency graph for phi.hpp:



This graph shows which files directly or indirectly include this file:



Namespaces

- **mlpack**

Linear algebra utility functions, generally performed on matrices or vectors.

- **mlpack::gmm**

Gaussian Mixture Models.

Functions

- double **mlpack::gmm::phi** (const double x, const double mean, const double var)

Calculates the univariate Gaussian probability density function.

- double **mlpack::gmm::phi** (const arma::vec &x, const arma::vec &mean, const arma::mat &cov)

Calculates the multivariate Gaussian probability density function.

- double **mlpack::gmm::phi** (const arma::vec &x, const arma::vec &mean, const arma::mat &cov, const std::vector< arma::mat > &d_cov, arma::vec &g_mean, arma::vec &g_cov)

Calculates the multivariate Gaussian probability density function and also the gradients with respect to the mean and the variance.

- void **mlpack::gmm::phi** (const arma::mat &x, const arma::vec &mean, const arma::mat &cov, arma::vec &probabilities)

Calculates the multivariate Gaussian probability density function for each data point (column) in the given matrix, with respect to the given mean and variance.

23.175.1 Detailed Description

Author

Parikshit Ram (pram@cc.gatech.edu)

This file computes the Gaussian probability density function

This file is part of mlpack 1.0.12.

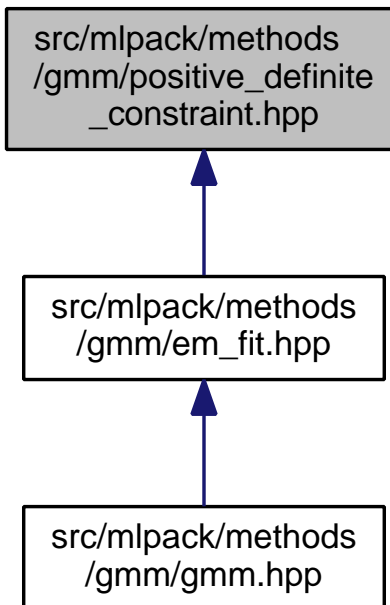
mlpack is free software; you may redistribute it and/or modify it under the terms of the 3-clause BSD license. You should have received a copy of the 3-clause BSD license along with mlpack. If not, see <http://www.opensource.org/licenses/BSD-3-Clause> for more information.

23.176 src/mlpack/methods/gmm/positive_definite_constraint.hpp File Reference

Include dependency graph for positive_definite_constraint.hpp:



This graph shows which files directly or indirectly include this file:



Classes

- class **mlpack::gmm::PositiveDefiniteConstraint**
Given a covariance matrix, force the matrix to be positive definite.

Namespaces

- **mlpack**
Linear algebra utility functions, generally performed on matrices or vectors.
- **mlpack::gmm**
Gaussian Mixture Models.

23.176.1 Detailed Description

Author

Ryan Curtin

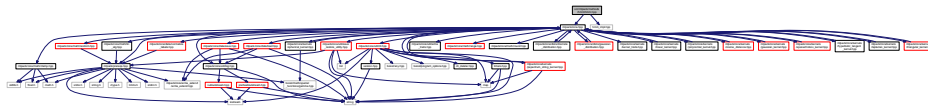
Restricts a covariance matrix to being positive definite.

This file is part of mlpack 1.0.12.

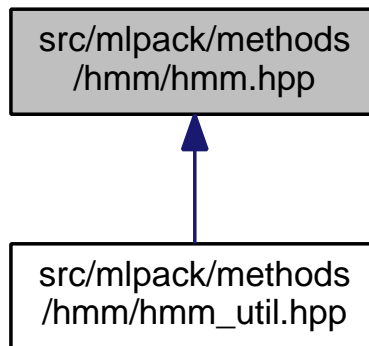
mlpack is free software; you may redistribute it and/or modify it under the terms of the 3-clause BSD license. You should have received a copy of the 3-clause BSD license along with mlpack. If not, see <http://www.opensource.org/licenses/BSD-3-Clause> for more information.

23.177 src/mlpack/methods/hmm/hmm.hpp File Reference

Include dependency graph for hmm.hpp:



This graph shows which files directly or indirectly include this file:



Classes

- class **mlpack::hmm::HMM**< **Distribution** >
A class that represents a Hidden Markov Model with an arbitrary type of emission distribution.

Namespaces

- **mlpack**
Linear algebra utility functions, generally performed on matrices or vectors.
- **mlpack::hmm**
Hidden Markov Models.

23.177.1 Detailed Description

Author

Ryan Curtin
 Tran Quoc Long

Definition of HMM class.

This file is part of mlpack 1.0.12.

mlpack is free software; you may redistribute it and/or modify it under the terms of the 3-clause BSD license. You should have received a copy of the 3-clause BSD license along with mlpack. If not, see <http://www.opensource.org/licenses/BSD-3-Clause> for more information.

23.178 src/mlpack/methods/hmm/hmm_util.hpp File Reference

Include dependency graph for hmm_util.hpp:



Namespaces

- **mlpack**
Linear algebra utility functions, generally performed on matrices or vectors.
- **mlpack::hmm**
Hidden Markov Models.

Functions

- `template<typename Distribution >`
`void mlpack::hmm::LoadHMM (HMM< Distribution > &hmm, util::SaveRestoreUtility &sr)`
*Load an **HMM** (p. 284) from file.*
- `template<typename Distribution >`
`void mlpack::hmm::SaveHMM (const HMM< Distribution > &hmm, util::SaveRestoreUtility &sr)`
*Save an **HMM** (p. 284) to file.*

23.178.1 Detailed Description

Author

Ryan Curtin

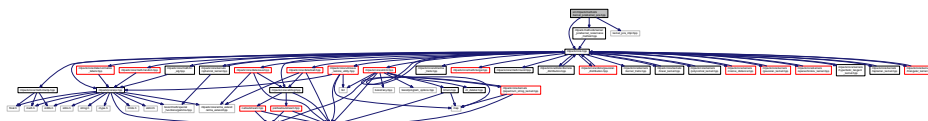
Save/load utilities for HMMs. This should be eventually merged into the HMM class itself.

This file is part of mlpack 1.0.12.

mlpack is free software; you may redistribute it and/or modify it under the terms of the 3-clause BSD license. You should have received a copy of the 3-clause BSD license along with mlpack. If not, see <http://www.opensource.org/licenses/BSD-3-Clause> for more information.

23.179 src/mlpack/methods/kernel_pca/kernel_pca.hpp File Reference

Include dependency graph for kernel_pca.hpp:



Classes

- class **mlpack::kpca::KernelPCA**< **KernelType**, **KernelRule** >

This class performs kernel principal components analysis (Kernel PCA), for a given kernel.

Namespaces

- **mlpack**

Linear algebra utility functions, generally performed on matrices or vectors.

- **mlpack::kpca**

23.179.1 Detailed Description

Author

Ajinkya Kale
Marcus Edel

Defines the KernelPCA class to perform Kernel Principal Components Analysis on the specified data set.

This file is part of mlpack 1.0.12.

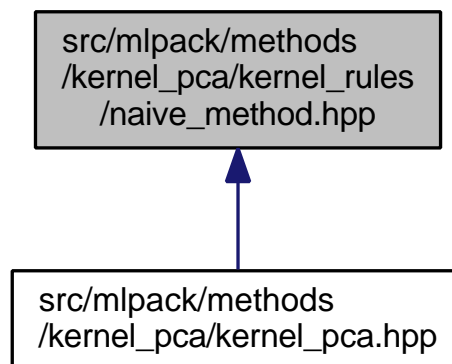
mlpack is free software; you may redistribute it and/or modify it under the terms of the 3-clause BSD license. You should have received a copy of the 3-clause BSD license along with mlpack. If not, see <http://www.opensource.org/licenses/BSD-3-Clause> for more information.

23.180 src/mlpack/methods/kernel_pca/kernel_rules/naive_method.hpp File Reference

Include dependency graph for naive_method.hpp:



This graph shows which files directly or indirectly include this file:



Classes

- class `mlpack::kpca::NaiveKernelRule< KernelType >`

Namespaces

- `mlpack`
Linear algebra utility functions, generally performed on matrices or vectors.
- `mlpack::kpca`

23.180.1 Detailed Description

Author

Ajinkya Kale

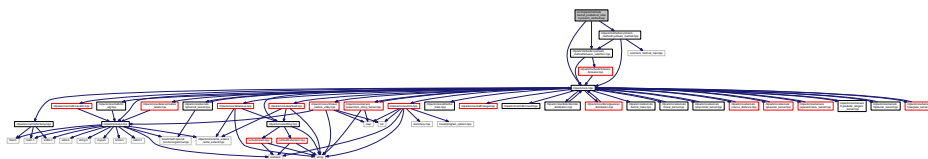
Use the naive method to construct the kernel matrix.

This file is part of mlpack 1.0.12.

mlpack is free software; you may redistribute it and/or modify it under the terms of the 3-clause BSD license. You should have received a copy of the 3-clause BSD license along with mlpack. If not, see <http://www.opensource.org/licenses/BSD-3-Clause> for more information.

23.181 src/mlpack/methods/kernel_pca/kernel_rules/nystroem_method.hpp File Reference

Include dependency graph for `nystroem_method.hpp`:



Classes

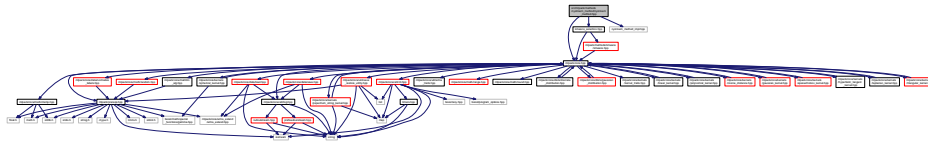
- class `mlpack::kpca::NystroemKernelRule< KernelType, PointSelectionPolicy >`

Namespaces

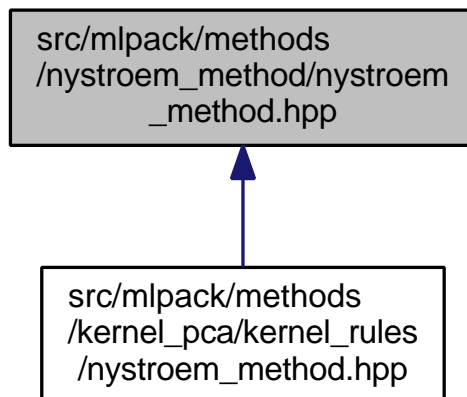
- `mlpack`
Linear algebra utility functions, generally performed on matrices or vectors.
- `mlpack::kpca`

23.182 src/mlpack/methods/nystroem_method/nystroem_method.hpp File Reference

Include dependency graph for nystroem_method.hpp:



This graph shows which files directly or indirectly include this file:



Classes

- class **mlpack::kernel::NystroemMethod**< **KernelType**, **PointSelectionPolicy** >

Namespaces

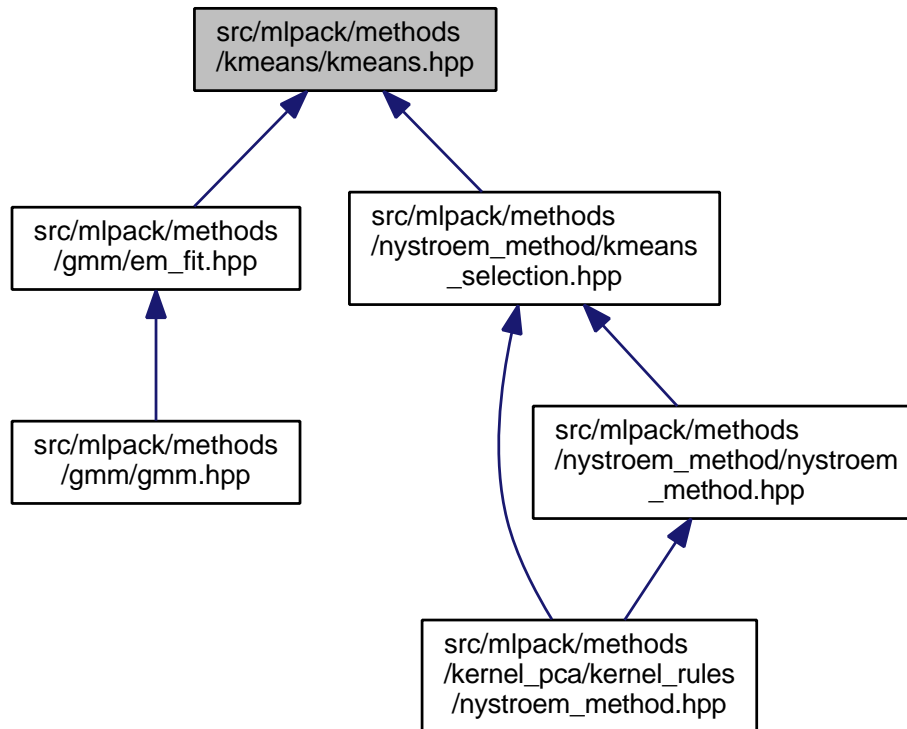
- **mlpack**
Linear algebra utility functions, generally performed on matrices or vectors.
- **mlpack::kernel**
Kernel functions.

23.183 src/mlpack/methods/kmeans/allow_empty_clusters.hpp File Reference

Include dependency graph for allow_empty_clusters.hpp:



This graph shows which files directly or indirectly include this file:



Classes

- class **mlpack::kmeans::KMeans**< **MetricType**, **InitialPartitionPolicy**, **EmptyClusterPolicy** >
This class implements K-Means clustering.

Namespaces

- **mlpack**
Linear algebra utility functions, generally performed on matrices or vectors.
- **mlpack::kmeans**
K-Means clustering.

23.184.1 Detailed Description

Author

Parikshit Ram (pram@cc.gatech.edu)

K-Means clustering.

This file is part of mlpack 1.0.12.

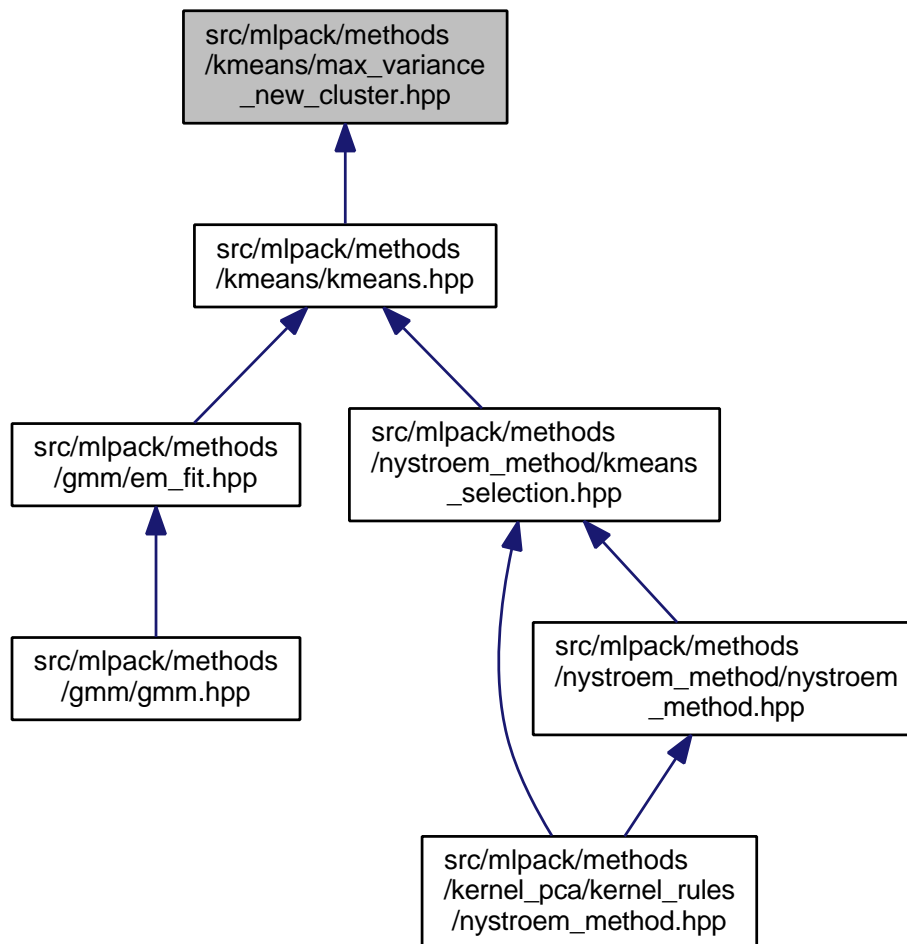
mlpack is free software; you may redistribute it and/or modify it under the terms of the 3-clause BSD license. You should have received a copy of the 3-clause BSD license along with mlpack. If not, see <http://www.opensource.org/licenses/BSD-3-Clause> for more information.

23.185 src/mlpack/methods/kmeans/max_variance_new_cluster.hpp File Reference

Include dependency graph for max_variance_new_cluster.hpp:



This graph shows which files directly or indirectly include this file:



Classes

- class **mlpack::kmeans::MaxVarianceNewCluster**

When an empty cluster is detected, this class takes the point furthest from the centroid of the cluster with maximum variance as a new cluster.

Namespaces

- **mlpack**

Linear algebra utility functions, generally performed on matrices or vectors.

- **mlpack::kmeans**

K-Means clustering.

23.185.1 Detailed Description

Author

Ryan Curtin

An implementation of the EmptyClusterPolicy policy class for K-Means. When an empty cluster is detected, the point furthest from the centroid of the cluster with maximum variance is taken to be a new cluster.

This file is part of mlpack 1.0.12.

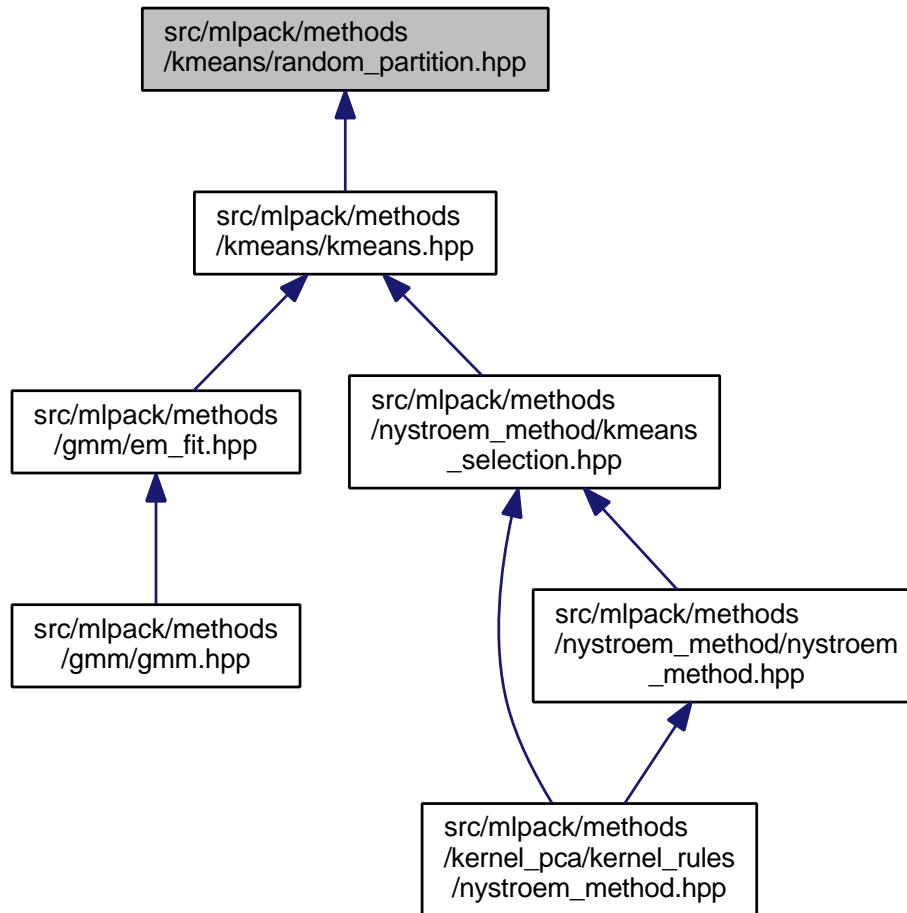
mlpack is free software; you may redistribute it and/or modify it under the terms of the 3-clause BSD license. You should have received a copy of the 3-clause BSD license along with mlpack. If not, see <http://www.opensource.org/licenses/BSD-3-Clause> for more information.

23.186 src/mlpack/methods/kmeans/random_partition.hpp File Reference

Include dependency graph for random_partition.hpp:



This graph shows which files directly or indirectly include this file:



Classes

- class **mlpack::kmeans::RandomPartition**

A very simple partitioner which partitions the data randomly into the number of desired clusters.

Namespaces

- **mlpack**

Linear algebra utility functions, generally performed on matrices or vectors.

- **mlpack::kmeans**

K-Means clustering.

23.186.1 Detailed Description

Author

Ryan Curtin

Very simple partitioner which partitions the data randomly into the number of desired clusters. Used as the default InitialPartitionPolicy for KMeans.

This file is part of mlpack 1.0.12.

mlpack is free software; you may redistribute it and/or modify it under the terms of the 3-clause BSD license. You should have received a copy of the 3-clause BSD license along with mlpack. If not, see <http://www.opensource.org/licenses/BSD-3-Clause> for more information.

23.187 src/mlpack/methods/kmeans/refined_start.hpp File Reference

Include dependency graph for refined_start.hpp:

**Classes**

- class **mlpack::kmeans::RefinedStart**

A refined approach for choosing initial points for k-means clustering.

Namespaces

- **mlpack**

Linear algebra utility functions, generally performed on matrices or vectors.

- **mlpack::kmeans**

K-Means clustering.

23.187.1 Detailed Description

Author

Ryan Curtin

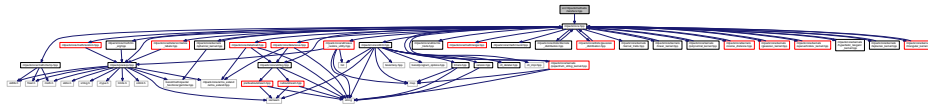
An implementation of Bradley and Fayyad's "Refining Initial Points for K-Means clustering". This class is meant to provide better initial points for the k-means algorithm.

This file is part of mlpack 1.0.12.

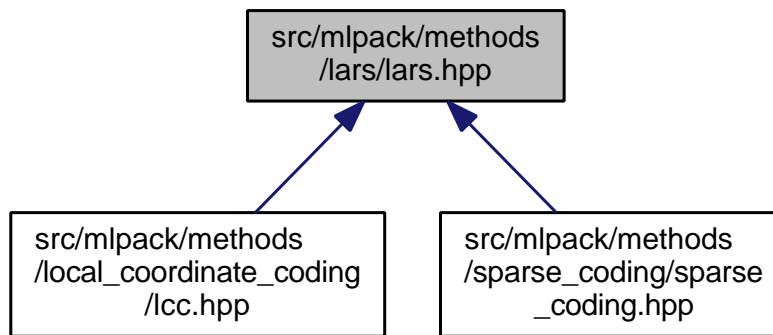
mlpack is free software; you may redistribute it and/or modify it under the terms of the 3-clause BSD license. You should have received a copy of the 3-clause BSD license along with mlpack. If not, see <http://www.opensource.org/licenses/BSD-3-Clause> for more information.

23.188 src/mlpack/methods/lars/lars.hpp File Reference

Include dependency graph for lars.hpp:



This graph shows which files directly or indirectly include this file:



Classes

- class **mlpack::regression::LARS**

An implementation of **LARS** (p. 525), a stage-wise homotopy-based algorithm for l_1 -regularized linear regression (LASSO) and l_1+l_2 regularized linear regression (Elastic Net).

Namespaces

- **mlpack**

Linear algebra utility functions, generally performed on matrices or vectors.

- **mlpack::regression**

Regression methods.

23.188.1 Detailed Description

Author

Nishant Mehta (niche)

Definition of the LARS class, which performs Least Angle Regression and the LASSO.

Only minor modifications of LARS are necessary to handle the constrained version of the problem:

$$\min_{\beta} 0.5 \|X\beta - y\|_2^2 + 0.5\lambda_2 \|\beta\|_2^2$$

subject to $\|\beta\|_1 \leq \tau$

Although this option currently is not implemented, it will be implemented very soon.

This file is part of mlpack 1.0.12.

mlpack is free software; you may redistribute it and/or modify it under the terms of the 3-clause BSD license. You should have received a copy of the 3-clause BSD license along with mlpack. If not, see <http://www.opensource.org/licenses/BSD-3-Clause> for more information.

23.189 src/mlpack/methods/linear_regression/linear_regression.hpp File Reference

Include dependency graph for linear_regression.hpp:



Classes

- class **mlpack::regression::LinearRegression**

A simple linear regression algorithm using ordinary least squares.

Namespaces

- **mlpack**

Linear algebra utility functions, generally performed on matrices or vectors.

- **mlpack::regression**

Regression methods.

23.189.1 Detailed Description

Author

James Cline
Michael Fox

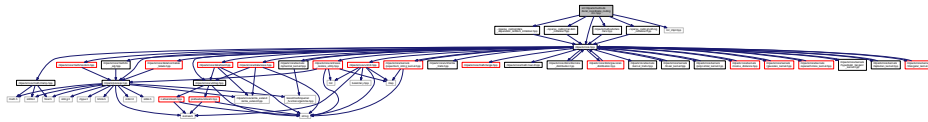
Simple least-squares linear regression.

This file is part of mlpack 1.0.12.

mlpack is free software; you may redistribute it and/or modify it under the terms of the 3-clause BSD license. You should have received a copy of the 3-clause BSD license along with mlpack. If not, see <http://www.opensource.org/licenses/BSD-3-Clause> for more information.

23.190 src/mlpack/methods/local_coordinate_coding/lcc.hpp File Reference

Include dependency graph for lcc.hpp:



Classes

- class **mlpack::lcc::LocalCoordinateCoding**< **DictionaryInitializer** >

An implementation of Local Coordinate Coding (LCC) that codes data which approximately lives on a manifold using a variation of l_1 -norm regularized sparse coding; in LCC, the penalty on the absolute value of each point's coefficient for each atom is weighted by the squared distance of that point to that atom.

Namespaces

- **mlpack**
Linear algebra utility functions, generally performed on matrices or vectors.
- **mlpack::lcc**

23.190.1 Detailed Description

Author

Nishant Mehta

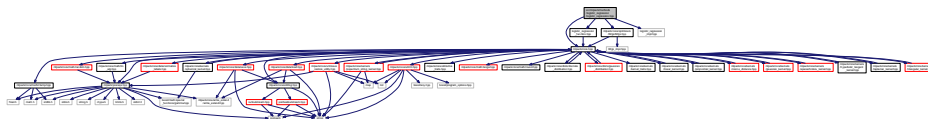
Definition of the LocalCoordinateCoding class, which performs the Local Coordinate Coding algorithm.

This file is part of mlpack 1.0.12.

mlpack is free software; you may redistribute it and/or modify it under the terms of the 3-clause BSD license. You should have received a copy of the 3-clause BSD license along with mlpack. If not, see <http://www.opensource.org/licenses/BSD-3-Clause> for more information.

23.191 src/mlpack/methods/logistic_regression/logistic_regression.hpp File Reference

Include dependency graph for logistic_regression.hpp:



Classes

- class **mlpack::regression::LogisticRegression**< **OptimizerType** >

Namespaces

- **mlpack**

Linear algebra utility functions, generally performed on matrices or vectors.

- **mlpack::regression**

Regression methods.

23.191.1 Detailed Description

Author

Sumedh Ghaisas

The LogisticRegression class, which implements logistic regression. This implements supports L2-regularization.

This file is part of mlpack 1.0.12.

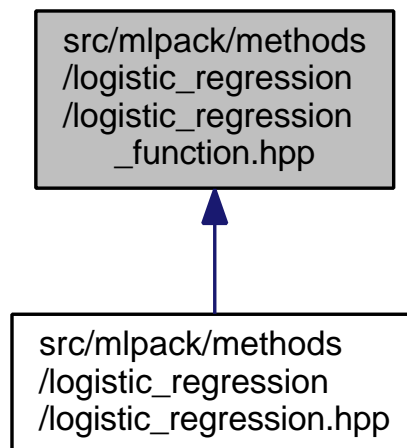
mlpack is free software; you may redistribute it and/or modify it under the terms of the 3-clause BSD license. You should have received a copy of the 3-clause BSD license along with mlpack. If not, see <http://www.opensource.org/licenses/BSD-3-Clause> for more information.

23.192 src/mlpack/methods/logistic_regression/logistic_regression_function.hpp File Reference

Include dependency graph for logistic_regression_function.hpp:



This graph shows which files directly or indirectly include this file:



Classes

- class **mlpack::regression::LogisticRegressionFunction**
The log-likelihood function for the logistic regression objective function.

Namespaces

- **mlpack**
Linear algebra utility functions, generally performed on matrices or vectors.
- **mlpack::regression**
Regression methods.

23.192.1 Detailed Description

Author

Sumedh Ghaisas

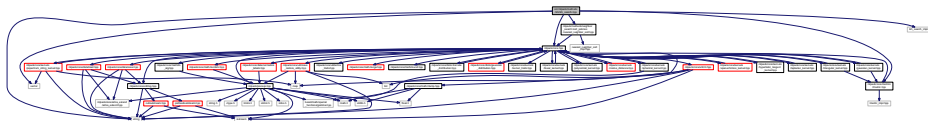
Implementation of the logistic regression function, which is meant to be optimized by a separate optimizer class that takes LogisticRegressionFunction as its FunctionType class.

This file is part of mlpack 1.0.12.

mlpack is free software; you may redistribute it and/or modify it under the terms of the 3-clause BSD license. You should have received a copy of the 3-clause BSD license along with mlpack. If not, see <http://www.opensource.org/licenses/BSD-3-Clause> for more information.

23.193 src/mlpack/methods/lsh/lsh_search.hpp File Reference

Include dependency graph for lsh_search.hpp:



Classes

- class **mlpack::neighbor::LSHSearch< SortPolicy >**
The **LSHSearch** (p. 388) class – This class builds a hash on the reference set and uses this hash to compute the distance-approximate nearest-neighbors of the given queries.

Namespaces

- **mlpack**
Linear algebra utility functions, generally performed on matrices or vectors.
- **mlpack::neighbor**
Neighbor-search routines.

23.193.1 Detailed Description

Author

Parikshit Ram

Defines the LSHSearch class, which performs an approximate nearest neighbor search for a queries in a query set over a given dataset using Locality-sensitive hashing with 2-stable distributions.

The details of this method can be found in the following paper:

{datar2004locality, title={Locality-sensitive hashing scheme based on p-stable distributions}, author={Datar, M. and Immorlica, N. and Indyk, P. and Mirrokni, V.S.}, booktitle= {Proceedings of the 12th Annual Symposium on Computational Geometry}, pages={253–262}, year={2004}, organization={ACM} }

This file is part of mlpack 1.0.12.

mlpack is free software; you may redistribute it and/or modify it under the terms of the 3-clause BSD license. You should have received a copy of the 3-clause BSD license along with mlpack. If not, see <http://www.opensource.org/licenses/BSD-3-Clause> for more information.

23.194 src/mlpack/methods/mvu/mvu.hpp File Reference

Include dependency graph for mvu.hpp:



Classes

- class **mlpack::mvu::MVU**

*The **MVU** (p. 372) class is meant to provide a good abstraction for users.*

Namespaces

- **mlpack**

Linear algebra utility functions, generally performed on matrices or vectors.

- **mlpack::mvu**

23.194.1 Detailed Description

Author

Ryan Curtin

An implementation of Maximum Variance Unfolding. This file defines an MVU class as well as a class representing the objective function (a semidefinite program) which MVU seeks to minimize. Minimization is performed by the Augmented Lagrangian optimizer (which in turn uses the L-BFGS optimizer).

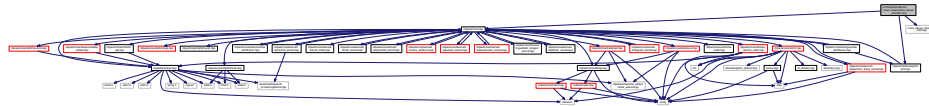
Note: this implementation of MVU does not work. See #189.

This file is part of mlpack 1.0.12.

mlpack is free software; you may redistribute it and/or modify it under the terms of the 3-clause BSD license. You should have received a copy of the 3-clause BSD license along with mlpack. If not, see <http://www.opensource.org/licenses/BSD-3-Clause> for more information.

23.195 src/mlpack/methods/naive_bayes/naive_bayes_classifier.hpp File Reference

Include dependency graph for naive_bayes_classifier.hpp:



Classes

- class **mlpack::naive_bayes::NaiveBayesClassifier**< **MatType** >
The simple Naive Bayes classifier.

Namespaces

- **mlpack**
Linear algebra utility functions, generally performed on matrices or vectors.
- **mlpack::naive_bayes**
The Naive Bayes Classifier.

23.195.1 Detailed Description

Author

Parikshit Ram (pram@cc.gatech.edu)

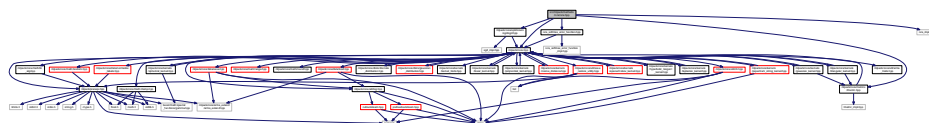
A Naive Bayes Classifier which parametrically estimates the distribution of the features. It is assumed that the features have been sampled from a Gaussian PDF.

This file is part of mlpack 1.0.12.

mlpack is free software; you may redistribute it and/or modify it under the terms of the 3-clause BSD license. You should have received a copy of the 3-clause BSD license along with mlpack. If not, see <http://www.opensource.org/licenses/BSD-3-Clause> for more information.

23.196 src/mlpack/methods/nca/nca.hpp File Reference

Include dependency graph for nca.hpp:



Classes

- class **mlpack::nca::NCA** < **MetricType**, **OptimizerType** >
An implementation of Neighborhood Components Analysis, both a linear dimensionality reduction technique and a distance learning technique.

Namespaces

- **mlpack**
Linear algebra utility functions, generally performed on matrices or vectors.
- **mlpack::nca**
Neighborhood Components Analysis.

23.196.1 Detailed Description

Author

Ryan Curtin

Declaration of NCA class (Neighborhood Components Analysis).

This file is part of mlpack 1.0.12.

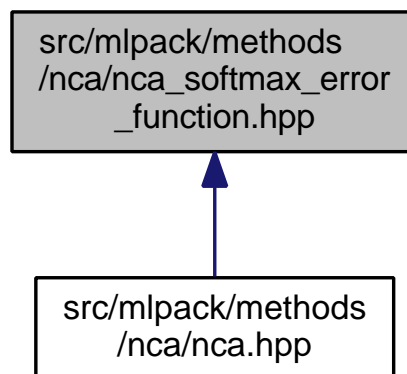
mlpack is free software; you may redistribute it and/or modify it under the terms of the 3-clause BSD license. You should have received a copy of the 3-clause BSD license along with mlpack. If not, see <http://www.opensource.org/licenses/BSD-3-Clause> for more information.

23.197 src/mlpack/methods/nca/nca_softmax_error_function.hpp File Reference

Include dependency graph for nca_softmax_error_function.hpp:



This graph shows which files directly or indirectly include this file:



Classes

- class **mlpack::nca::SoftmaxErrorFunction**< **MetricType** >

The "softmax" stochastic neighbor assignment probability function.

Namespaces

- **mlpack**
Linear algebra utility functions, generally performed on matrices or vectors.
- **mlpack::nca**
Neighborhood Components Analysis.

23.197.1 Detailed Description

Author

Ryan Curtin

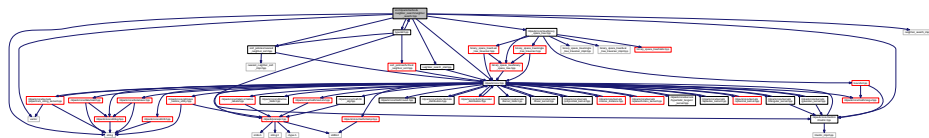
Implementation of the stochastic neighbor assignment probability error function (the "softmax error").

This file is part of mlpack 1.0.12.

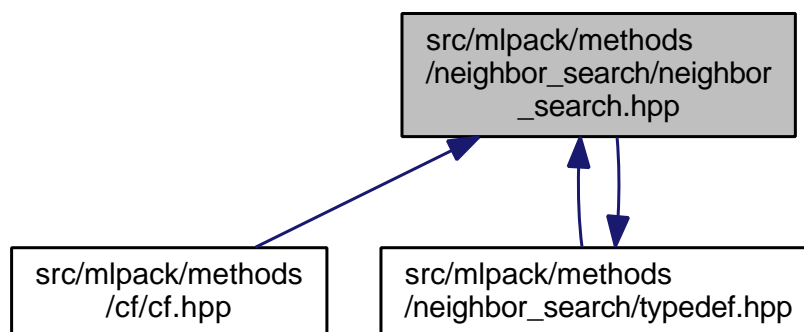
mlpack is free software; you may redistribute it and/or modify it under the terms of the 3-clause BSD license. You should have received a copy of the 3-clause BSD license along with mlpack. If not, see <http://www.opensource.org/licenses/BSD-3-Clause> for more information.

23.198 src/mlpack/methods/neighbor_search/neighbor_search.hpp File Reference

Include dependency graph for neighbor_search.hpp:



This graph shows which files directly or indirectly include this file:



Classes

- class **mlpack::neighbor::NeighborSearch**< **SortPolicy**, **MetricType**, **TreeType** >

The *NeighborSearch* (p. 399) class is a template class for performing distance-based neighbor searches.

Namespaces

- **mlpack**

Linear algebra utility functions, generally performed on matrices or vectors.

- **mlpack::neighbor**

Neighbor-search routines.

23.198.1 Detailed Description

Author

Ryan Curtin

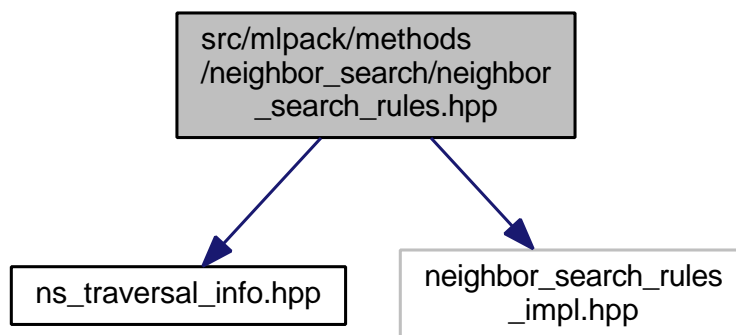
Defines the NeighborSearch class, which performs an abstract nearest-neighbor-like query on two datasets.

This file is part of mlpack 1.0.12.

mlpack is free software; you may redistribute it and/or modify it under the terms of the 3-clause BSD license. You should have received a copy of the 3-clause BSD license along with mlpack. If not, see <http://www.opensource.org/licenses/BSD-3-Clause> for more information.

23.199 src/mlpack/methods/neighbor_search/neighbor_search_rules.hpp File Reference

Include dependency graph for neighbor_search_rules.hpp:



Classes

- class **mlpack::neighbor::NeighborSearchRules**< **SortPolicy**, **MetricType**, **TreeType** >

Namespaces

- **mlpack**
Linear algebra utility functions, generally performed on matrices or vectors.
- **mlpack::neighbor**
Neighbor-search routines.

23.199.1 Detailed Description

Author

Ryan Curtin

Defines the pruning rules and base case rules necessary to perform a tree-based search (with an arbitrary tree) for the NeighborSearch class.

This file is part of mlpack 1.0.12.

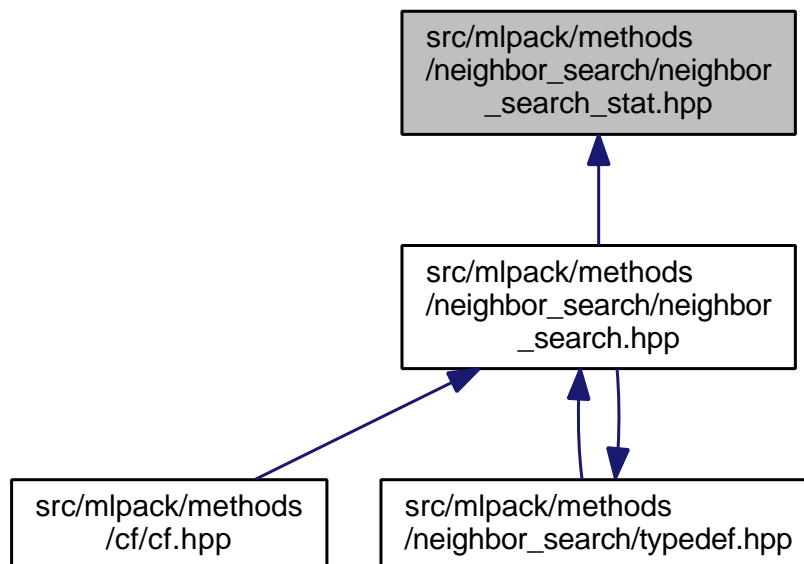
mlpack is free software; you may redistribute it and/or modify it under the terms of the 3-clause BSD license. You should have received a copy of the 3-clause BSD license along with mlpack. If not, see <http://www.opensource.org/licenses/BSD-3-Clause> for more information.

23.200 src/mlpack/methods/neighbor_search/neighbor_search_stat.hpp File Reference

Include dependency graph for neighbor_search_stat.hpp:



This graph shows which files directly or indirectly include this file:



Classes

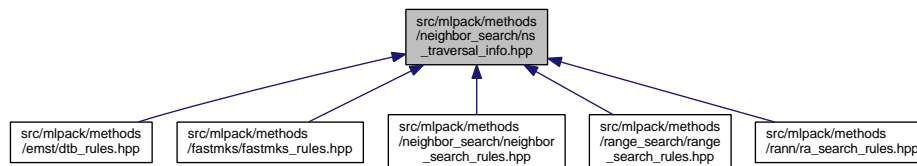
- class **mlpack::neighbor::NeighborSearchStat**< **SortPolicy** >
Extra data for each node in the tree.

Namespaces

- **mlpack**
Linear algebra utility functions, generally performed on matrices or vectors.
- **mlpack::neighbor**
Neighbor-search routines.

23.201 src/mlpack/methods/neighbor_search/ns_traversal_info.hpp File Reference

This graph shows which files directly or indirectly include this file:



Classes

- class **mlpack::neighbor::NeighborSearchTraversalInfo**< **TreeType** >
*Traversal information for **NeighborSearch** (p. 399).*

Namespaces

- **mlpack**
Linear algebra utility functions, generally performed on matrices or vectors.
- **mlpack::neighbor**
Neighbor-search routines.

23.201.1 Detailed Description

Author

Ryan Curtin

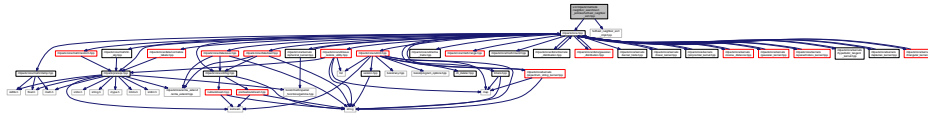
This class holds traversal information for dual-tree traversals that are using the NeighborSearchRules RuleType.

This file is part of mlpack 1.0.12.

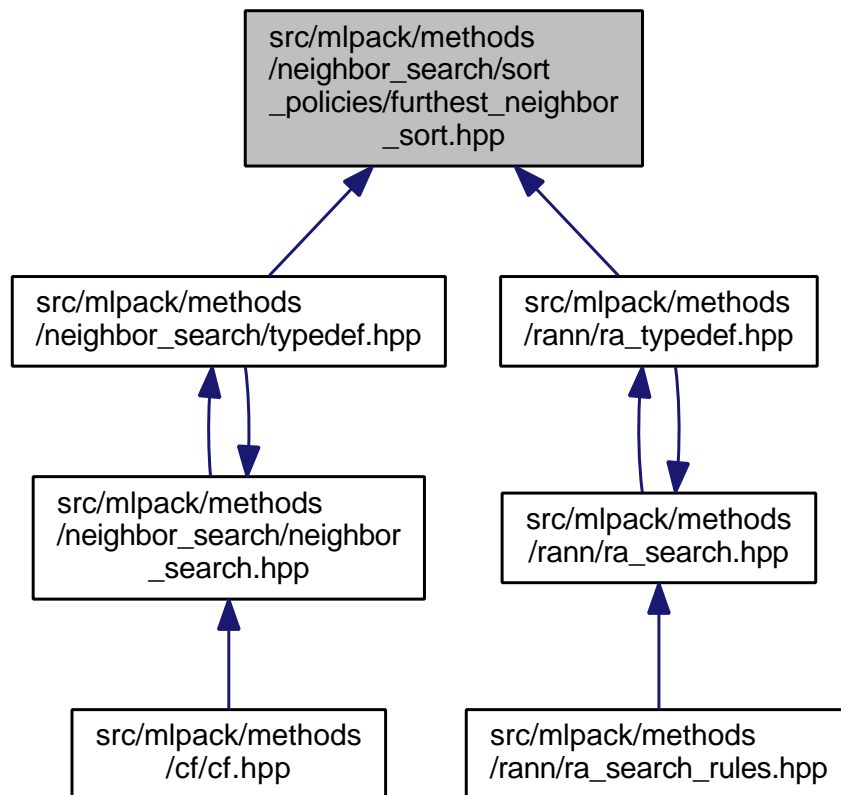
mlpack is free software; you may redistribute it and/or modify it under the terms of the 3-clause BSD license. You should have received a copy of the 3-clause BSD license along with mlpack. If not, see <http://www.opensource.org/licenses/BSD-3-Clause> for more information.

23.202 src/mlpack/methods/neighbor_search/sort_policies/furthest_neighbor_sort.hpp File Reference

Include dependency graph for furthest_neighbor_sort.hpp:



This graph shows which files directly or indirectly include this file:



Classes

- class **mlpack::neighbor::FurthestNeighborSort**

*This class implements the necessary methods for the SortPolicy template parameter of the **NeighborSearch** (p. 399) class.*

Namespaces

- **mlpack**

Linear algebra utility functions, generally performed on matrices or vectors.

- **mlpack::neighbor**

Neighbor-search routines.

23.202.1 Detailed Description

Author

Ryan Curtin

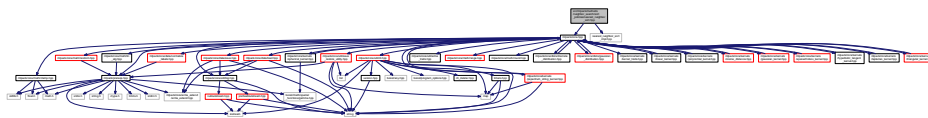
Implementation of the SortPolicy class for NeighborSearch; in this case, the furthest neighbors are those that are most important.

This file is part of mlpack 1.0.12.

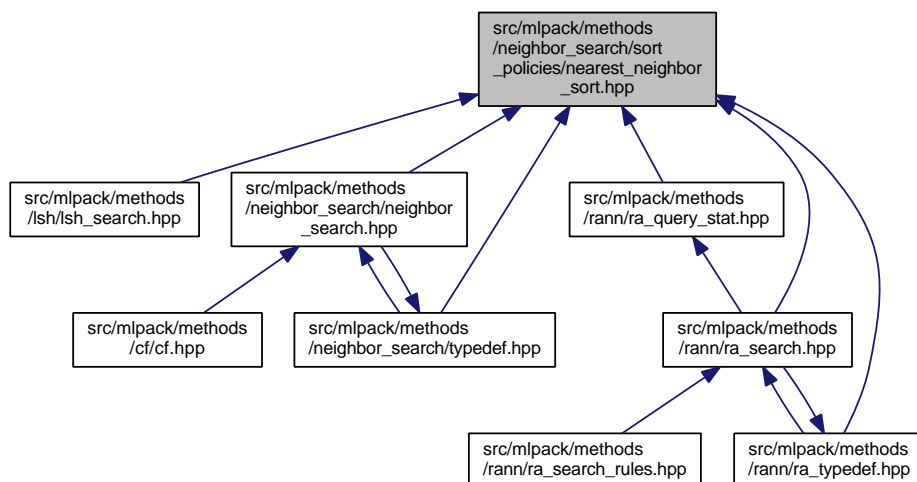
mlpack is free software; you may redistribute it and/or modify it under the terms of the 3-clause BSD license. You should have received a copy of the 3-clause BSD license along with mlpack. If not, see <http://www.opensource.org/licenses/BSD-3-Clause> for more information.

23.203 src/mlpack/methods/neighbor_search/sort_policies/nearest_neighbor_sort.hpp File Reference

Include dependency graph for nearest_neighbor_sort.hpp:



This graph shows which files directly or indirectly include this file:



Classes

- class **mlpack::neighbor::NearestNeighborSort**

*This class implements the necessary methods for the SortPolicy template parameter of the **NeighborSearch** (p. 399) class.*

Namespaces

- **mlpack**

Linear algebra utility functions, generally performed on matrices or vectors.

- **mlpack::neighbor**

Neighbor-search routines.

23.203.1 Detailed Description

Author

Ryan Curtin

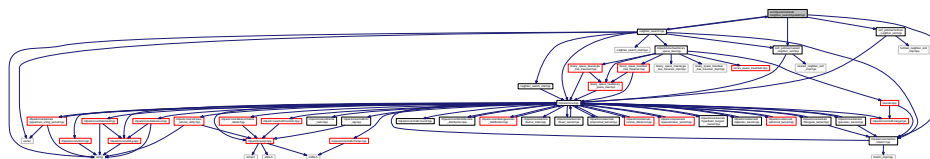
Implementation of the SortPolicy class for NeighborSearch; in this case, the nearest neighbors are those that are most important.

This file is part of mlpack 1.0.12.

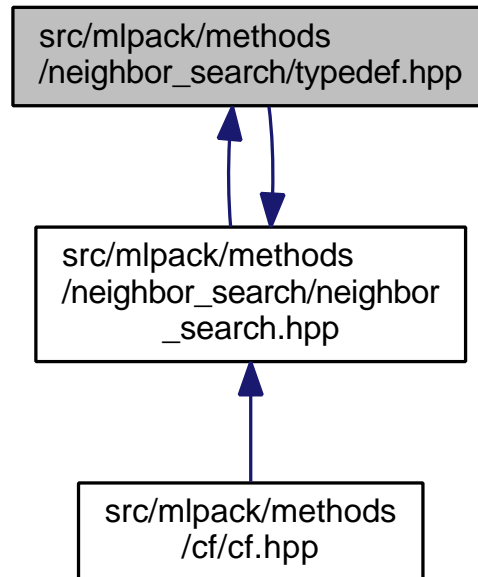
mlpack is free software; you may redistribute it and/or modify it under the terms of the 3-clause BSD license. You should have received a copy of the 3-clause BSD license along with mlpack. If not, see <http://www.opensource.org/licenses/BSD-3-Clause> for more information.

23.204 src/mlpack/methods/neighbor_search/typedef.hpp File Reference

Include dependency graph for typedef.hpp:



This graph shows which files directly or indirectly include this file:



Namespaces

- **mlpack**

Linear algebra utility functions, generally performed on matrices or vectors.

- **mlpack::neighbor**

Neighbor-search routines.

Typedefs

- `typedef NeighborSearch< FurthestNeighborSort, metric::EuclideanDistance > mlpack::neighbor::AllkFN`

The AllkFN class is the all-k-furthest-neighbors method.

- `typedef NeighborSearch< NearestNeighborSort, metric::EuclideanDistance > mlpack::neighbor::AllkNN`

The AllkNN class is the all-k-nearest-neighbors method.

23.204.1 Detailed Description

Author

Ryan Curtin

Simple typedefs describing template instantiations of the NeighborSearch class which are commonly used. This is meant to be included by neighbor_search.h but is a separate file for simplicity.

This file is part of mlpack 1.0.12.

mlpack is free software; you may redistribute it and/or modify it under the terms of the 3-clause BSD license. You should have received a copy of the 3-clause BSD license along with mlpack. If not, see <http://www.opensource.org/licenses/BSD-3-Clause> for more information.

23.205 src/mlpack/methods/neighbor_search/unmap.hpp File Reference

Include dependency graph for unmap.hpp:



Namespaces

- **mlpack**

Linear algebra utility functions, generally performed on matrices or vectors.

- **mlpack::neighbor**

Neighbor-search routines.

Functions

- void **mlpack::neighbor::Unmap** (const arma::Mat< size_t > &neighbors, const arma::mat &distances, const std::vector< size_t > &referenceMap, const std::vector< size_t > &queryMap, arma::Mat< size_t > &neighborsOut, arma::mat &distancesOut, const bool squareRoot=false)

Assuming that the datasets have been mapped using the referenceMap and the queryMap (such as during kd-tree construction), unmap the columns of the distances and neighbors matrices into neighborsOut and distancesOut, and also unmap the entries in each row of neighbors.

- void **mlpack::neighbor::Unmap** (const arma::Mat< size_t > &neighbors, const arma::mat &distances, const std::vector< size_t > &referenceMap, arma::Mat< size_t > &neighborsOut, arma::mat &distancesOut, const bool squareRoot=false)

Assuming that the datasets have been mapped using referenceMap (such as during kd-tree construction), unmap the columns of the distances and neighbors matrices into neighborsOut and distancesOut, and also unmap the entries in each row of neighbors.

23.205.1 Detailed Description

Author

Ryan Curtin

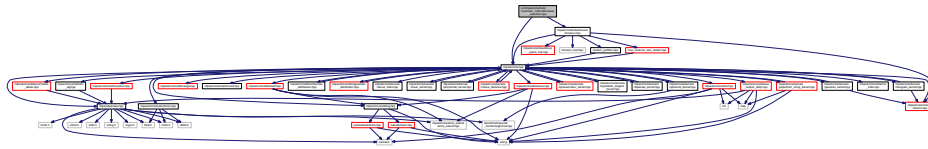
Convenience methods to unmap results.

This file is part of mlpack 1.0.12.

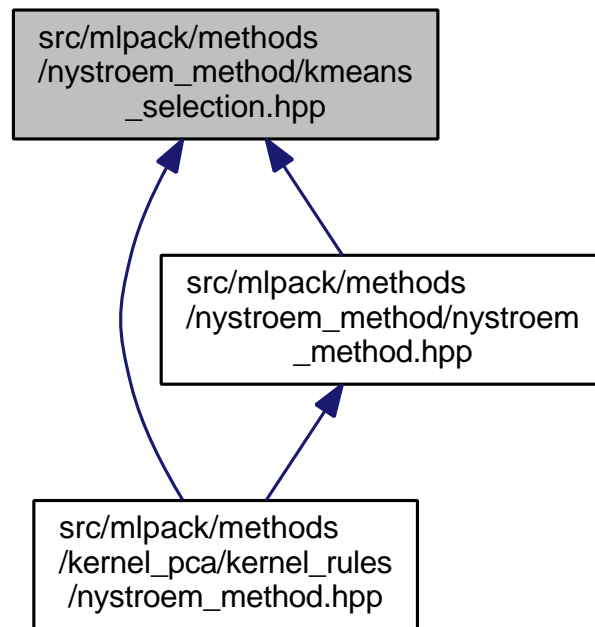
mlpack is free software; you may redistribute it and/or modify it under the terms of the 3-clause BSD license. You should have received a copy of the 3-clause BSD license along with mlpack. If not, see <http://www.opensource.org/licenses/BSD-3-Clause> for more information.

23.206 src/mlpack/methods/nystroem_method/kmeans_selection.hpp File Reference

Include dependency graph for kmeans_selection.hpp:



This graph shows which files directly or indirectly include this file:



Classes

- class **mlpack::kernel::KMeansSelection**< **ClusteringType** >

Namespaces

- **mlpack**
Linear algebra utility functions, generally performed on matrices or vectors.
- **mlpack::kernel**
Kernel functions.

23.206.1 Detailed Description

Author

Marcus Edel

Use the centroids of the K-Means clustering method for use in the Nystroem method of kernel matrix approximation.

This file is part of mlpack 1.0.12.

mlpack is free software; you may redistribute it and/or modify it under the terms of the 3-clause BSD license. You should have received a copy of the 3-clause BSD license along with mlpack. If not, see <http://www.opensource.org/licenses/BSD-3-Clause> for more information.

23.207 src/mlpack/methods/nystroem_method/ordered_selection.hpp File Reference

Include dependency graph for ordered_selection.hpp:

**Classes**

- class **mlpack::kernel::OrderedSelection**

Namespaces

- **mlpack**
Linear algebra utility functions, generally performed on matrices or vectors.
- **mlpack::kernel**
Kernel functions.

23.207.1 Detailed Description

Author

Ryan Curtin

Select the first points of the dataset for use in the Nystroem method of kernel matrix approximation. This is mostly for testing, but might have other uses.

This file is part of mlpack 1.0.12.

mlpack is free software; you may redistribute it and/or modify it under the terms of the 3-clause BSD license. You should have received a copy of the 3-clause BSD license along with mlpack. If not, see <http://www.opensource.org/licenses/BSD-3-Clause> for more information.

23.208 src/mlpack/methods/nystroem_method/random_selection.hpp File Reference

Include dependency graph for random_selection.hpp:



Classes

- class **mlpack::kernel::RandomSelection**

Namespaces

- **mlpack**
Linear algebra utility functions, generally performed on matrices or vectors.
- **mlpack::kernel**
Kernel functions.

23.208.1 Detailed Description

Author

Ryan Curtin

Randomly select some points (with replacement) to use for the Nystroem method. Replacement is suboptimal, but for rank \ll number of points, this is unlikely.

This file is part of mlpack 1.0.12.

mlpack is free software; you may redistribute it and/or modify it under the terms of the 3-clause BSD license. You should have received a copy of the 3-clause BSD license along with mlpack. If not, see <http://www.opensource.org/licenses/BSD-3-Clause> for more information.

23.209 src/mlpack/methods/pca/pca.hpp File Reference

Include dependency graph for pca.hpp:



Classes

- class **mlpack::pca::PCA**
*This class implements principal components analysis (**PCA** (p. 498)).*

Namespaces

- **mlpack**
Linear algebra utility functions, generally performed on matrices or vectors.
- **mlpack::pca**

23.209.1 Detailed Description

Author

Ajinkya Kale

Defines the PCA class to perform Principal Components Analysis on the specified data set.

This file is part of mlpack 1.0.12.

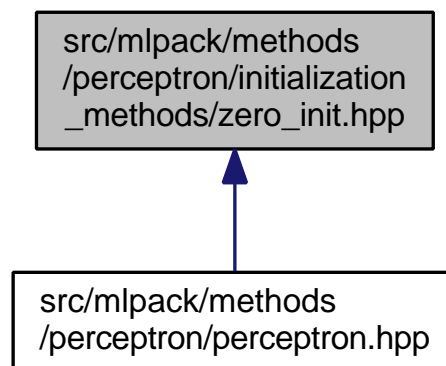
mlpack is free software; you may redistribute it and/or modify it under the terms of the 3-clause BSD license. You should have received a copy of the 3-clause BSD license along with mlpack. If not, see <http://www.opensource.org/licenses/BSD-3-Clause> for more information.

23.210 src/mlpack/methods/perceptron/initialization_methods/zero_init.hpp File Reference

Include dependency graph for zero_init.hpp:



This graph shows which files directly or indirectly include this file:



Classes

- class **mlpack::perceptron::ZeroInitialization**
This class is used to initialize the matrix weightVectors to zero.

Namespaces

- **mlpack**
Linear algebra utility functions, generally performed on matrices or vectors.
- **mlpack::perceptron**

23.210.1 Detailed Description

Author

Udit Saxena

Implementation of ZeroInitialization policy for perceptrons.

This file is part of mlpack 1.0.12.

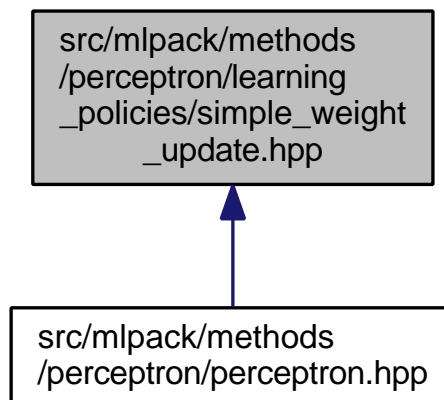
mlpack is free software; you may redistribute it and/or modify it under the terms of the 3-clause BSD license. You should have received a copy of the 3-clause BSD license along with mlpack. If not, see <http://www.opensource.org/licenses/BSD-3-Clause> for more information.

23.211 src/mlpack/methods/perceptron/learning_policies/simple_weight_update.hpp File Reference

Include dependency graph for simple_weight_update.hpp:



This graph shows which files directly or indirectly include this file:



Classes

- class **mlpack::perceptron::SimpleWeightUpdate**

Author

Udit Saxena

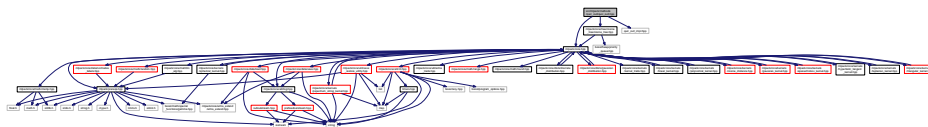
Definition of Perceptron class.

This file is part of mlpack 1.0.12.

mlpack is free software; you may redistribute it and/or modify it under the terms of the 3-clause BSD license. You should have received a copy of the 3-clause BSD license along with mlpack. If not, see <http://www.opensource.org/licenses/BSD-3-Clause> for more information.

23.213 src/mlpack/methods/quic_svd/quic_svd.hpp File Reference

Include dependency graph for quic_svd.hpp:

**Classes**

- class **mlpack::svd::QUIC_SVD**

Namespaces

- **mlpack**
Linear algebra utility functions, generally performed on matrices or vectors.
- **mlpack::svd**

23.213.1 Detailed Description

Author

Siddharth Agrawal

An implementation of QUIC-SVD.

This file is part of mlpack 1.0.12.

mlpack is free software; you may redistribute it and/or modify it under the terms of the 3-clause BSD license. You should have received a copy of the 3-clause BSD license along with mlpack. If not, see <http://www.opensource.org/licenses/BSD-3-Clause> for more information.

23.214 src/mlpack/methods/radical/radical.hpp File Reference

Include dependency graph for radical.hpp:



Classes

- class **mlpack::radical::Radical**

An implementation of RADICAL, an algorithm for independent component analysis (ICA).

Namespaces

- **mlpack**

Linear algebra utility functions, generally performed on matrices or vectors.

- **mlpack::radical**

Functions

- void **mlpack::radical::WhitenFeatureMajorMatrix** (const arma::mat &matX, arma::mat &matXWhitened, arma::mat &matWhitening)

23.214.1 Detailed Description

Author

Nishant Mehta

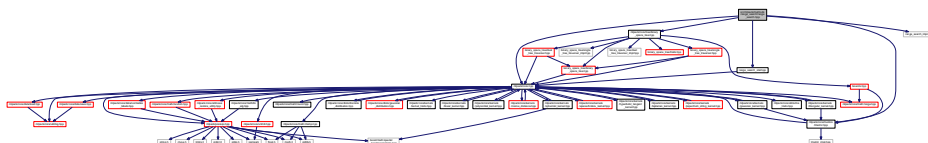
Declaration of Radical class (RADICAL is Robust, Accurate, Direct ICA aLgorithm).

This file is part of mlpack 1.0.12.

mlpack is free software; you may redistribute it and/or modify it under the terms of the 3-clause BSD license. You should have received a copy of the 3-clause BSD license along with mlpack. If not, see <http://www.opensource.org/licenses/BSD-3-Clause> for more information.

23.215 src/mlpack/methods/range_search/range_search.hpp File Reference

Include dependency graph for range_search.hpp:



Classes

- class **mlpack::range::RangeSearch**< **MetricType**, **TreeType** >

The *RangeSearch* (p. 512) class is a template class for performing range searches.

Namespaces

- **mlpack**

Linear algebra utility functions, generally performed on matrices or vectors.

- **mlpack::range**

Range-search routines.

23.215.1 Detailed Description

Author

Ryan Curtin

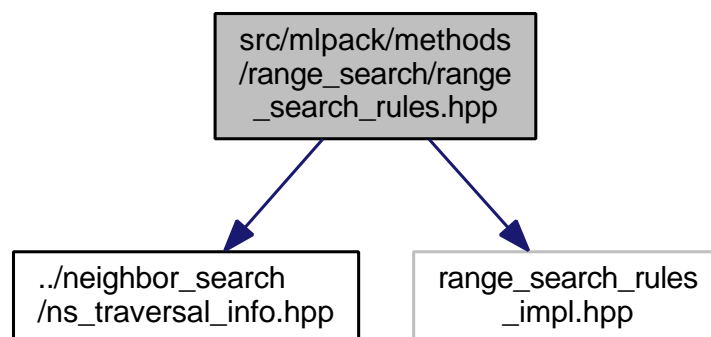
Defines the RangeSearch class, which performs a generalized range search on points.

This file is part of mlpack 1.0.12.

mlpack is free software; you may redistribute it and/or modify it under the terms of the 3-clause BSD license. You should have received a copy of the 3-clause BSD license along with mlpack. If not, see <http://www.opensource.org/licenses/BSD-3-Clause> for more information.

23.216 src/mlpack/methods/range_search/range_search_rules.hpp File Reference

Include dependency graph for range_search_rules.hpp:



Classes

- class **mlpack::range::RangeSearchRules**< **MetricType**, **TreeType** >

Namespaces

- **mlpack**
Linear algebra utility functions, generally performed on matrices or vectors.
- **mlpack::range**
Range-search routines.

23.216.1 Detailed Description

Author

Ryan Curtin

Rules for range search, so that it can be done with arbitrary tree types.

This file is part of mlpack 1.0.12.

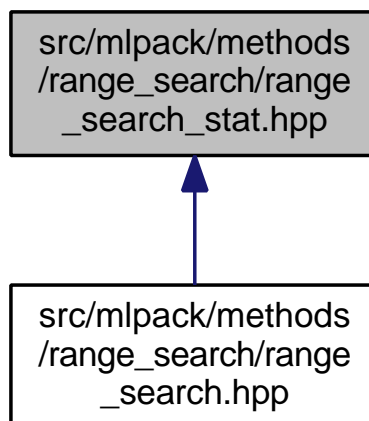
mlpack is free software; you may redistribute it and/or modify it under the terms of the 3-clause BSD license. You should have received a copy of the 3-clause BSD license along with mlpack. If not, see <http://www.opensource.org/licenses/BSD-3-Clause> for more information.

23.217 src/mlpack/methods/range_search/range_search_stat.hpp File Reference

Include dependency graph for range_search_stat.hpp:



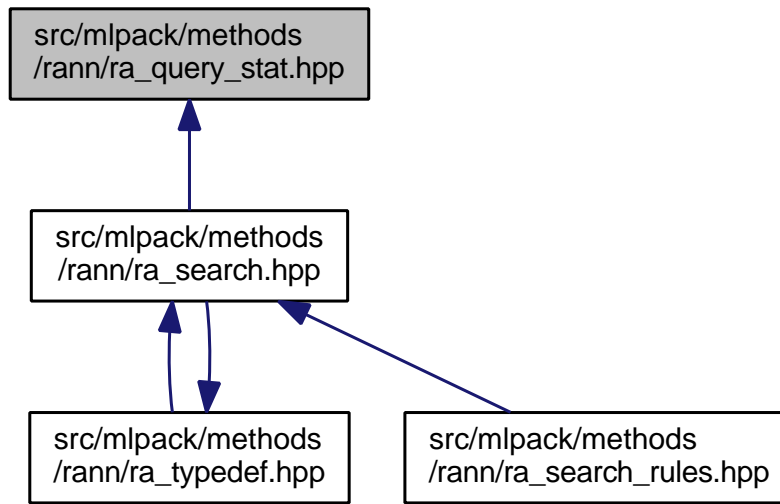
This graph shows which files directly or indirectly include this file:



Classes

- class **mlpack::range::RangeSearchStat**

This graph shows which files directly or indirectly include this file:



23.218.1 Detailed Description

Author

Parikshit Ram

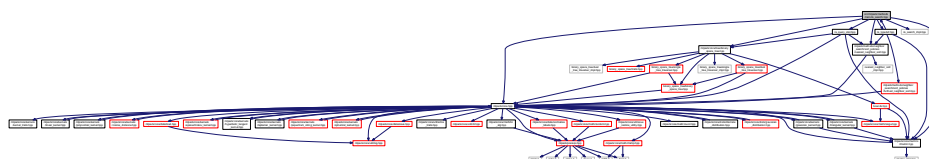
Defines the `RAQueryStat` class, which is the statistic used for rank-approximate nearest neighbor search (**RASearch** (p. 666)).

This file is part of mlpack 1.0.12.

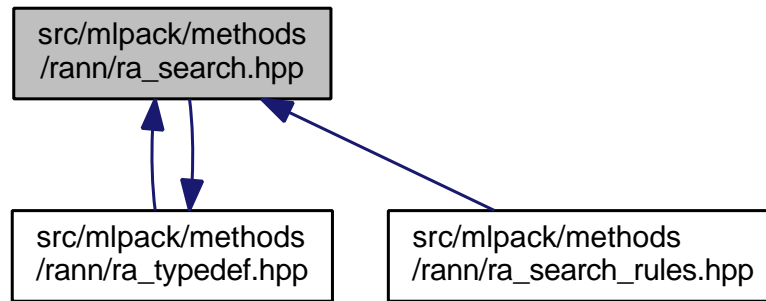
mlpack is free software; you may redistribute it and/or modify it under the terms of the 3-clause BSD license. You should have received a copy of the 3-clause BSD license along with mlpack. If not, see <http://www.opensource.org/licenses/BSD-3-Clause> for more information.

23.219 src/mlpack/methods/rann/ra_search.hpp File Reference

Include dependency graph for `ra_search.hpp`:



This graph shows which files directly or indirectly include this file:



Classes

- class **RASearch**< **SortPolicy**, **MetricType**, **TreeType** >

The **RASearch** (p. 666) class: This class provides a generic manner to perform rank-approximate search via random-sampling.

23.219.1 Detailed Description

Author

Parikshit Ram

Defines the **RASearch** (p. 666) class, which performs an abstract rank-approximate nearest/farthest neighbor query on two datasets.

The details of this method can be found in the following paper:

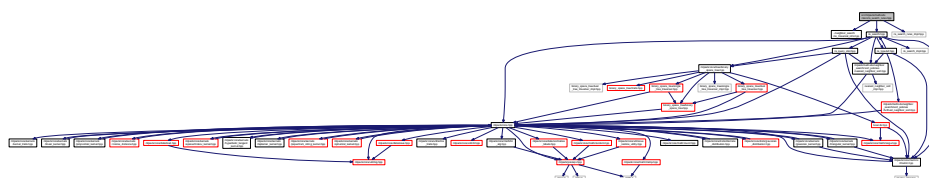
{ram2009rank, title={{Rank-Approximate Nearest Neighbor Search: Retaining Meaning and Speed in High Dimensions}}, author={{Ram, P. and Lee, D. and Ouyang, H. and Gray, A. G.}}, booktitle={{Advances of Neural Information Processing Systems}}, year={2009} }

This file is part of mlpack 1.0.12.

mlpack is free software; you may redistribute it and/or modify it under the terms of the 3-clause BSD license. You should have received a copy of the 3-clause BSD license along with mlpack. If not, see <http://www.opensource.org/licenses/BSD-3-Clause> for more information.

23.220 src/mlpack/methods/rann/ra_search_rules.hpp File Reference

Include dependency graph for ra_search_rules.hpp:



Classes

- class `mlpack::neighbor::RASearchRules`< `SortPolicy`, `MetricType`, `TreeType` >

Namespaces

- `mlpack`

Linear algebra utility functions, generally performed on matrices or vectors.

- `mlpack::neighbor`

Neighbor-search routines.

23.220.1 Detailed Description

Author

Parikshit Ram

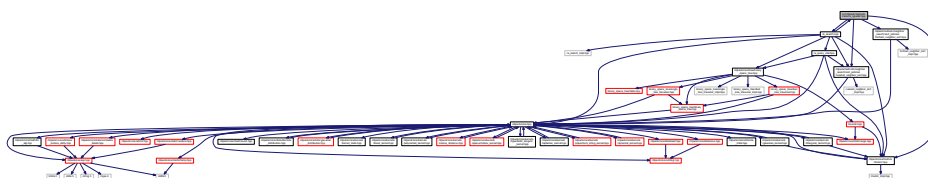
Defines the pruning rules and base case rules necessary to perform a tree-based rank-approximate search (with an arbitrary tree) for the **RASearch** (p. 666) class.

This file is part of mlpack 1.0.12.

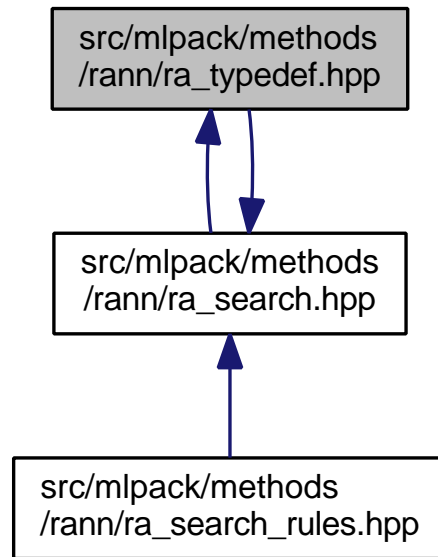
mlpack is free software; you may redistribute it and/or modify it under the terms of the 3-clause BSD license. You should have received a copy of the 3-clause BSD license along with mlpack. If not, see <http://www.opensource.org/licenses/BSD-3-Clause> for more information.

23.221 src/mlpack/methods/rann/ra_typedef.hpp File Reference

Include dependency graph for `ra_typedef.hpp`:



This graph shows which files directly or indirectly include this file:



Namespaces

- **mlpack**

Linear algebra utility functions, generally performed on matrices or vectors.

- **mlpack::neighbor**

Neighbor-search routines.

Typedefs

- typedef **RASearch**< FurthestNeighborSort > **mlpack::neighbor::AllkRAFN**

The AllkRAFN class is the all-k-rank-approximate-farthest-neighbors method.

- typedef **RASearch** **mlpack::neighbor::AllkRANN**

The AllkRANN class is the all-k-rank-approximate-nearest-neighbors method.

23.221.1 Detailed Description

Author

Parikshit Ram

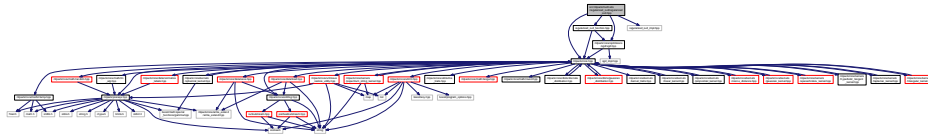
Simple typedefs describing template instantiations of the **RASearch** (p. 666) class which are commonly used.

This file is part of mlpack 1.0.12.

mlpack is free software; you may redistribute it and/or modify it under the terms of the 3-clause BSD license. You should have received a copy of the 3-clause BSD license along with mlpack. If not, see <http://www.opensource.org/licenses/BSD-3-Clause> for more information.

23.222 src/mlpack/methods/regularized_svd/regularized_svd.hpp File Reference

Include dependency graph for regularized_svd.hpp:



Classes

- class **mlpack::svd::RegularizedSVD**< **OptimizerType** >

Namespaces

- **mlpack**
Linear algebra utility functions, generally performed on matrices or vectors.
- **mlpack::svd**

23.222.1 Detailed Description

Author

Siddharth Agrawal

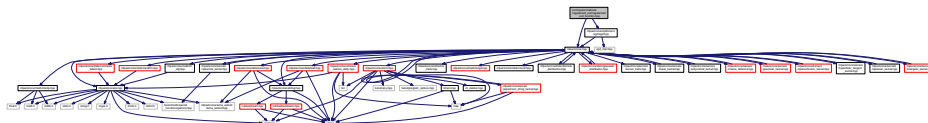
An implementation of Regularized SVD.

This file is part of mlpack 1.0.12.

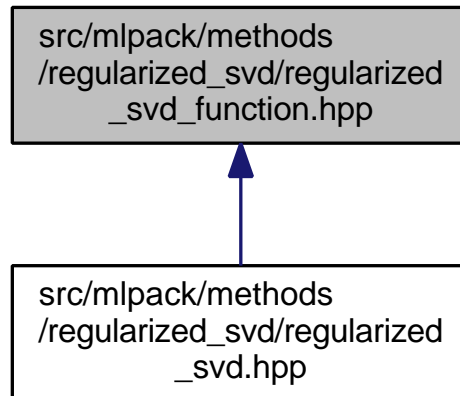
mlpack is free software; you may redistribute it and/or modify it under the terms of the 3-clause BSD license. You should have received a copy of the 3-clause BSD license along with mlpack. If not, see <http://www.opensource.org/licenses/BSD-3-Clause> for more information.

23.223 src/mlpack/methods/regularized_svd/regularized_svd_function.hpp File Reference

Include dependency graph for regularized_svd_function.hpp:



This graph shows which files directly or indirectly include this file:



Classes

- class **mlpack::svd::RegularizedSVDFunction**

Namespaces

- **mlpack**
Linear algebra utility functions, generally performed on matrices or vectors.
- **mlpack::optimization**
- **mlpack::svd**

23.223.1 Detailed Description

Author

Siddharth Agrawal

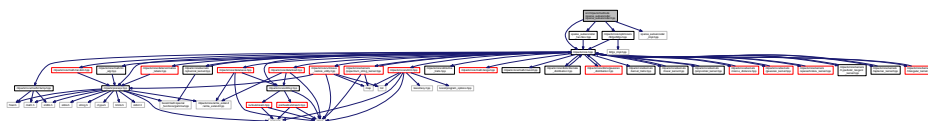
An implementation of the RegularizedSVDFunction class.

This file is part of mlpack 1.0.12.

mlpack is free software; you may redistribute it and/or modify it under the terms of the 3-clause BSD license. You should have received a copy of the 3-clause BSD license along with mlpack. If not, see <http://www.opensource.org/licenses/BSD-3-Clause> for more information.

23.224 src/mlpack/methods/sparse_autoencoder/sparse_autoencoder.hpp File Reference

Include dependency graph for sparse_autoencoder.hpp:



Classes

- class **mlpack::nn::SparseAutoencoder**< **OptimizerType** >

A sparse autoencoder is a neural network whose aim to learn compressed representations of the data, typically for dimensionality reduction, with a constraint on the activity of the neurons in the network.

Namespaces

- **mlpack**
Linear algebra utility functions, generally performed on matrices or vectors.
- **mlpack::nn**

23.224.1 Detailed Description

Author

Siddharth Agrawal

An implementation of sparse autoencoders.

This file is part of mlpack 1.0.12.

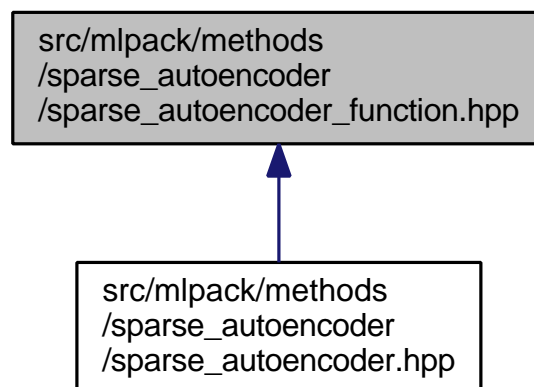
mlpack is free software; you may redistribute it and/or modify it under the terms of the 3-clause BSD license. You should have received a copy of the 3-clause BSD license along with mlpack. If not, see <http://www.opensource.org/licenses/BSD-3-Clause> for more information.

23.225 src/mlpack/methods/sparse_autoencoder/sparse_autoencoder_function.hpp File Reference

Include dependency graph for sparse_autoencoder_function.hpp:



This graph shows which files directly or indirectly include this file:



Classes

- class **mlpack::nn::SparseAutoencoderFunction**

This is a class for the sparse autoencoder objective function.

Namespaces

- **mlpack**

Linear algebra utility functions, generally performed on matrices or vectors.

- **mlpack::nn**

23.225.1 Detailed Description

Author

Siddharth Agrawal

The function to be optimized for sparse autoencoders. Any mlpack optimizer can be used.

This file is part of mlpack 1.0.12.

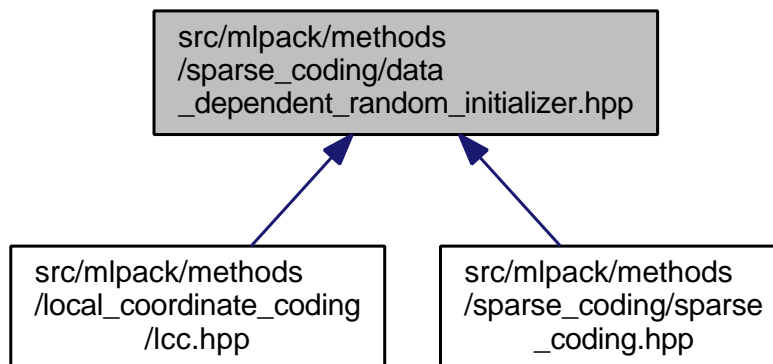
mlpack is free software; you may redistribute it and/or modify it under the terms of the 3-clause BSD license. You should have received a copy of the 3-clause BSD license along with mlpack. If not, see <http://www.opensource.org/licenses/BSD-3-Clause> for more information.

23.226 src/mlpack/methods/sparse_coding/data_dependent_random_initializer.hpp File Reference

Include dependency graph for data_dependent_random_initializer.hpp:



This graph shows which files directly or indirectly include this file:



Classes

- class **mlpack::sparse_coding::DataDependentRandomInitializer**
A data-dependent random dictionary initializer for *SparseCoding* (p. 547).

Namespaces

- **mlpack**
Linear algebra utility functions, generally performed on matrices or vectors.
- **mlpack::sparse_coding**

23.226.1 Detailed Description

Author

Nishant Mehta

A sensible heuristic for initializing dictionaries for sparse coding.

This file is part of mlpack 1.0.12.

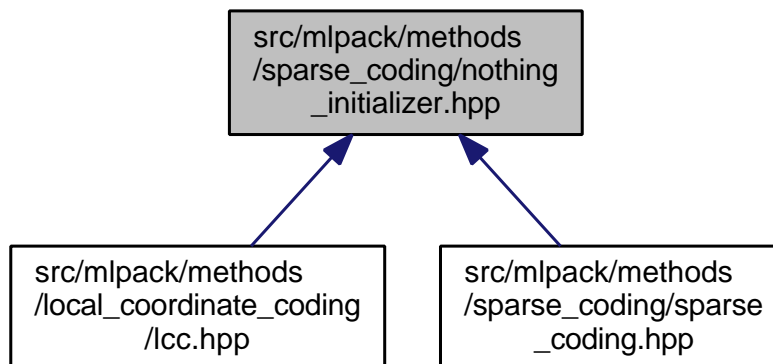
mlpack is free software; you may redistribute it and/or modify it under the terms of the 3-clause BSD license. You should have received a copy of the 3-clause BSD license along with mlpack. If not, see <http://www.opensource.org/licenses/BSD-3-Clause> for more information.

23.227 src/mlpack/methods/sparse_coding/nothing_initializer.hpp File Reference

Include dependency graph for nothing_initializer.hpp:



This graph shows which files directly or indirectly include this file:



Classes

- class **mlpack::sparse_coding::NothingInitializer**

A *DictionaryInitializer* for **SparseCoding** (p. 547) which does not initialize anything; it is useful for when the dictionary is already known and will be set with **SparseCoding::Dictionary()** (p. 550).

Namespaces

- **mlpack**
Linear algebra utility functions, generally performed on matrices or vectors.
- **mlpack::sparse_coding**

23.227.1 Detailed Description

Author

Ryan Curtin

An initializer for SparseCoding which does precisely nothing. It is useful for when you have an already defined dictionary and you plan on setting it with SparseCoding::Dictionary().

This file is part of mlpack 1.0.12.

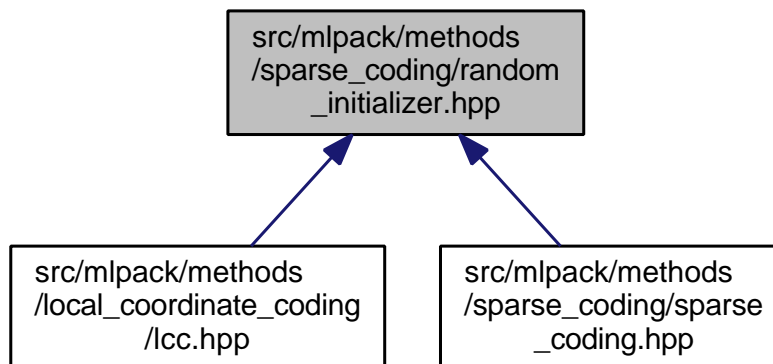
mlpack is free software; you may redistribute it and/or modify it under the terms of the 3-clause BSD license. You should have received a copy of the 3-clause BSD license along with mlpack. If not, see <http://www.opensource.org/licenses/BSD-3-Clause> for more information.

23.228 src/mlpack/methods/sparse_coding/random_initializer.hpp File Reference

Include dependency graph for random_initializer.hpp:



This graph shows which files directly or indirectly include this file:



Classes

- class **mlpack::sparse_coding::RandomInitializer**
A *DictionaryInitializer* for use with the *SparseCoding* (p. 547) class.

Namespaces

- **mlpack**
Linear algebra utility functions, generally performed on matrices or vectors.
- **mlpack::sparse_coding**

23.228.1 Detailed Description

Author

Nishant Mehta

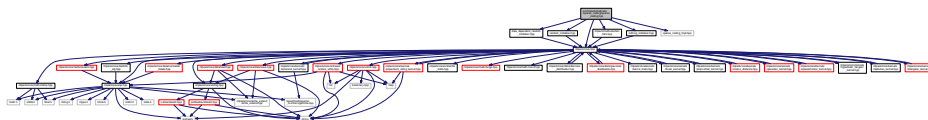
A very simple random dictionary initializer for SparseCoding; it is probably not a very good choice.

This file is part of mlpack 1.0.12.

mlpack is free software; you may redistribute it and/or modify it under the terms of the 3-clause BSD license. You should have received a copy of the 3-clause BSD license along with mlpack. If not, see <http://www.opensource.org/licenses/BSD-3-Clause> for more information.

23.229 src/mlpack/methods/sparse_coding/sparse_coding.hpp File Reference

Include dependency graph for sparse_coding.hpp:



Classes

- class **mlpack::sparse_coding::SparseCoding< DictionaryInitializer >**
An implementation of Sparse Coding with Dictionary Learning that achieves sparsity via an l_1 -norm regularizer on the codes (LASSO) or an (l_1+l_2) -norm regularizer on the codes (the Elastic Net).

Namespaces

- **mlpack**
Linear algebra utility functions, generally performed on matrices or vectors.
- **mlpack::sparse_coding**

23.229.1 Detailed Description

Author

Nishant Mehta

Definition of the SparseCoding class, which performs L1 (LASSO) or L1+L2 (Elastic Net)-regularized sparse coding with dictionary learning

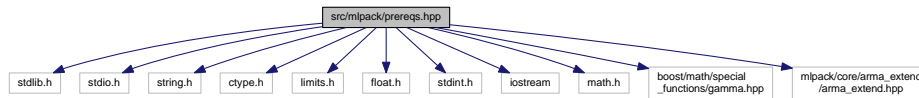
This file is part of mlpack 1.0.12.

mlpack is free software; you may redistribute it and/or modify it under the terms of the 3-clause BSD license. You should have received a copy of the 3-clause BSD license along with mlpack. If not, see <http://www.opensource.org/licenses/BSD-3-Clause> for more information.

23.230 src/mlpack/prereqs.hpp File Reference

The core includes that mlpack expects; standard C++ includes and Armadillo.

Include dependency graph for prereqs.hpp:



This graph shows which files directly or indirectly include this file:



Macros

- `#define _USE_MATH_DEFINES`
- `#define force_inline`
- `#define M_PI 3.141592653589793238462643383279`

23.230.1 Detailed Description

The core includes that mlpack expects; standard C++ includes and Armadillo.

This file is part of mlpack 1.0.12.

mlpack is free software; you may redistribute it and/or modify it under the terms of the 3-clause BSD license. You should have received a copy of the 3-clause BSD license along with mlpack. If not, see <http://www.opensource.org/licenses/BSD-3-Clause> for more information.

23.230.2 Macro Definition Documentation

23.230.2.1 `#define _USE_MATH_DEFINES`

Definition at line 34 of file `prereqs.hpp`.

23.230.2.2 `#define force_inline`

Definition at line 46 of file `prereqs.hpp`.

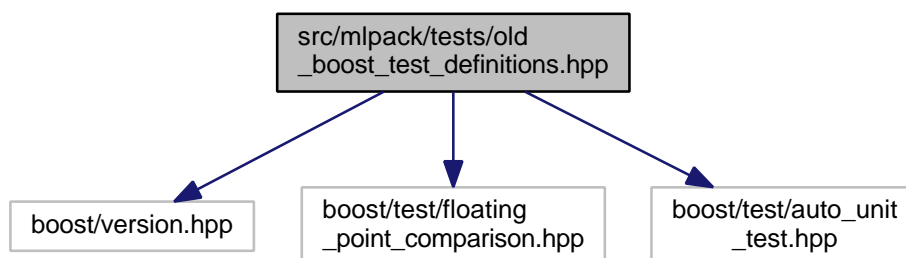
23.230.2.3 `#define M_PI 3.141592653589793238462643383279`

Definition at line 42 of file `prereqs.hpp`.

Referenced by `mlpack::kernel::SphericalKernel::Normalizer()`, `mlpack::kernel::GaussianKernel::Normalizer()`, and `mlpack::gmm::phi()`.

23.231 `src/mlpack/tests/data/data_3d_ind.txt` File Reference23.232 `src/mlpack/tests/data/data_3d_mixed.txt` File Reference23.233 `src/mlpack/tests/data/iris.txt` File Reference23.234 `src/mlpack/tests/data/iris_labels.txt` File Reference23.235 `src/mlpack/tests/old_boost_test_definitions.hpp` File Reference

Include dependency graph for `old_boost_test_definitions.hpp`:



Macros

- `#define BOOST_REQUIRE_GE(L, R) BOOST_REQUIRE_EQUAL((L >= R), true)`
- `#define BOOST_REQUIRE_GT(L, R) BOOST_REQUIRE_EQUAL((L > R), true)`
- `#define BOOST_REQUIRE_LE(L, R) BOOST_REQUIRE_EQUAL((L <= R), true)`
- `#define BOOST_REQUIRE_LT(L, R) BOOST_REQUIRE_EQUAL((L < R), true)`
- `#define BOOST_REQUIRE_NE(L, R) BOOST_REQUIRE_EQUAL((L != R), true)`

23.235.1 Detailed Description

Author

Ryan Curtin

Ancient Boost.Test versions don't act how we expect. This file includes the things we need to fix that.

This file is part of mpack 1.0.12.

mpack is free software; you may redistribute it and/or modify it under the terms of the 3-clause BSD license. You should have received a copy of the 3-clause BSD license along with mpack. If not, see <http://www.opensource.org/licenses/BSD-3-Clause> for more information.

23.235.2 Macro Definition Documentation

23.235.2.1 `#define BOOST_REQUIRE_GE(L, R) BOOST_REQUIRE_EQUAL((L >= R), true)`

Definition at line 28 of file `old_boost_test_definitions.hpp`.

23.235.2.2 `#define BOOST_REQUIRE_GT(L, R) BOOST_REQUIRE_EQUAL((L > R), true)`

Definition at line 40 of file `old_boost_test_definitions.hpp`.

23.235.2.3 `#define BOOST_REQUIRE_LE(L, R) BOOST_REQUIRE_EQUAL((L <= R), true)`

Definition at line 34 of file `old_boost_test_definitions.hpp`.

23.235.2.4 `#define BOOST_REQUIRE_LT(L, R) BOOST_REQUIRE_EQUAL((L < R), true)`

Definition at line 37 of file `old_boost_test_definitions.hpp`.

23.235.2.5 `#define BOOST_REQUIRE_NE(L, R) BOOST_REQUIRE_EQUAL((L != R), true)`

Definition at line 31 of file `old_boost_test_definitions.hpp`.

Index

\$V

det.txt, 678

__USE_MATH_DEFINES

prereqs.hpp, 856

__MLPACK_VERSION_MAJOR

src/mlpack/core/util/version.hpp, 775

__MLPACK_VERSION_MINOR

src/mlpack/core/util/version.hpp, 775

__MLPACK_VERSION_PATCH

src/mlpack/core/util/version.hpp, 776

~BallBound

mlpack::bound::BallBound, 166

~BinarySpaceTree

mlpack::tree::BinarySpaceTree, 574

~CLI

mlpack::CLI, 188

~CLIDeleter

mlpack::util::CLIDeleter, 650

~CosineTree

mlpack::tree::CosineTree, 597

~CoverTree

mlpack::tree::CoverTree, 610

~DTree

mlpack::det::DTree, 207

~DualTreeBoruvka

mlpack::emst::DualTreeBoruvka, 239

~EmptyStatistic

mlpack::tree::EmptyStatistic, 629

~FastMKS

mlpack::fastmks::FastMKS, 251

~HRectBound

mlpack::bound::HRectBound, 173

~IPMetric

mlpack::metric::IPMetric, 366

~NeighborSearch

mlpack::neighbor::NeighborSearch, 402

~RASearch

RASearch, 670

~RangeSearch

mlpack::range::RangeSearch, 515

~SVDCompleteIncrementalLearning

mlpack::amf::SVDCompleteIncrementalLearning<
arma::sp_mat >, 155

~SaveRestoreUtility

mlpack::util::SaveRestoreUtility, 664

~UnionFind

mlpack::emst::UnionFind, 246

A

mlpack::optimization::LRSDP, 469

mlpack::optimization::LRSDPFunction, 473

a

mlpack::optimization::LRSDPFunction, 475

AMF

mlpack::amf::AMF, 124

AModes

mlpack::optimization::LRSDP, 470

mlpack::optimization::LRSDPFunction, 473

aModes

mlpack::optimization::LRSDPFunction, 475

Activate

mlpack::regression::LARS, 528

ActiveSet

mlpack::regression::LARS, 528

activeSet

mlpack::regression::LARS, 530

Add

mlpack::CLI, 190

add_custom_command

tests/CMakeLists.txt, 702

add_executable

tests/CMakeLists.txt, 702

add_subdirectory

core/CMakeLists.txt, 681

core/optimizers/CMakeLists.txt, 684

methods/CMakeLists.txt, 689

AddAlias

mlpack::CLI, 190

AddAllEdges

mlpack::emst::DualTreeBoruvka, 239

AddEdge

mlpack::emst::DualTreeBoruvka, 239

AddFlag

mlpack::CLI, 190

AddResult

mlpack::range::RangeSearchRules, 520

AliasReverseLookup

mlpack::CLI, 192

aliasValues

mlpack::CLI, 195

AllkFN

- mlpack::neighbor, 111
- AllkNN
 - mlpack::neighbor, 111
- AllkRAFN
 - mlpack::neighbor, 112
- AllkRANN
 - mlpack::neighbor, 112
- AllowEmptyClusters
 - mlpack::kmeans::AllowEmptyClusters, 333
- alpha
 - det.txt, 678
 - mlpack::svd::RegularizedSVD, 556
- AlphaUpper
 - mlpack::det::DTree, 207
- alphaUpper
 - mlpack::det::DTree, 212
- amap_t
 - mlpack::CLI, 188
- Angles
 - mlpack::radical::Radical, 509
- angles
 - mlpack::radical::Radical, 511
- Apply
 - mlpack::amf::AMF, 125
 - mlpack::kernel::NystroemMethod, 318
 - mlpack::kpca::KernelPCA, 348
 - mlpack::pca::PCA, 499, 500
- ApplyConstraint
 - mlpack::gmm::DiagonalConstraint, 264
 - mlpack::gmm::EigenvalueRatioConstraint, 265
 - mlpack::gmm::NoConstraint, 283
 - mlpack::gmm::PositiveDefiniteConstraint, 284
- ApplyKernelMatrix
 - mlpack::kpca::NaiveKernelRule, 350
 - mlpack::kpca::NystroemKernelRule, 351
- ArmijoConstant
 - mlpack::optimization::L_BFGS, 458
- armijoConstant
 - mlpack::optimization::L_BFGS, 463
- Assert
 - mlpack::Log, 358
- atoms
 - mlpack::lcc::LocalCoordinateCoding, 356
 - mlpack::sparse_coding::SparseCoding, 552
- AugLag
 - mlpack::optimization::LRSDP, 470
- augLag
 - mlpack::optimization::LRSDP, 471
- AugLagrangian
 - mlpack::optimization::AugLagrangian, 443
- AugLagrangianFunction
 - mlpack::optimization::AugLagrangianFunction, 447, 448
- AugLagrangianTestFunction
 - mlpack::optimization::AugLagrangianTestFunction, 451
- augfunc
 - mlpack::optimization::AugLagrangian, 445
- AverageInitialization
 - mlpack::amf::AverageInitialization, 127
- B
 - mlpack::optimization::LRSDP, 470
 - mlpack::optimization::LRSDPFunction, 473, 474
- b
 - mlpack::optimization::LRSDPFunction, 475
- BOOST_REQUIRE_GE
 - old_boost_test_definitions.hpp, 858
- BOOST_REQUIRE_GT
 - old_boost_test_definitions.hpp, 858
- BOOST_REQUIRE_LE
 - old_boost_test_definitions.hpp, 858
- BOOST_REQUIRE_LT
 - old_boost_test_definitions.hpp, 858
- BOOST_REQUIRE_NE
 - old_boost_test_definitions.hpp, 858
- Backward
 - mlpack::hmm::HMM, 287
- BallBound
 - mlpack::bound::BallBound, 166
- Bandwidth
 - mlpack::kernel::GaussianKernel, 300, 301
 - mlpack::kernel::LaplacianKernel, 314
 - mlpack::kernel::TriangularKernel, 330
- bandwidth
 - mlpack::kernel::EpanechnikovKernel, 296
 - mlpack::kernel::GaussianKernel, 303
 - mlpack::kernel::LaplacianKernel, 315
 - mlpack::kernel::SphericalKernel, 329
 - mlpack::kernel::TriangularKernel, 332
- bandwidthSquared
 - mlpack::kernel::SphericalKernel, 329
- Base
 - mlpack::tree::CoverTree, 610
- base
 - mlpack::tree::CoverTree, 619
- BaseCase
 - mlpack::emst::DTBRules, 227
 - mlpack::fastmks::FastMKSRules, 256
 - mlpack::neighbor::LSHSearch, 390
 - mlpack::neighbor::NeighborSearchRules, 407
 - mlpack::neighbor::RASearchRules, 422
 - mlpack::range::RangeSearchRules, 520
- baseCase
 - mlpack::tree::CoverTree::DualTreeTraverser::Dual↔CoverTreeMapEntry, 626
- BaseCases
 - mlpack::emst::DTBRules, 227

- mlpack::fastmks::FastMKSRules, 256
- mlpack::neighbor::NeighborSearchRules, 407, 408
- baseCases
 - mlpack::emst::DTBRules, 230
 - mlpack::fastmks::FastMKSRules, 258
 - mlpack::neighbor::NeighborSearchRules, 410
- BaseLogic
 - mlpack::util::PrefixedOutputStream, 659
- basis
 - mlpack::svd::QUIC_SVD, 555
 - mlpack::tree::CosineTree, 600
- BasisVector
 - mlpack::tree::CosineTree, 597
- basisVector
 - mlpack::tree::CosineTree, 600
- Begin
 - mlpack::tree::BinarySpaceTree, 574
 - mlpack::tree::MRKDStatistic, 642
- begin
 - mlpack::tree::BinarySpaceTree, 584
 - mlpack::tree::MRKDStatistic, 644
- BestDistance
 - mlpack::neighbor::FurthestNeighborSort, 386
 - mlpack::neighbor::NearestNeighborSort, 396
- BestNodeToNodeDistance
 - mlpack::neighbor::FurthestNeighborSort, 386
 - mlpack::neighbor::NearestNeighborSort, 396, 397
- BestPointToNodeDistance
 - mlpack::neighbor::FurthestNeighborSort, 386
 - mlpack::neighbor::NearestNeighborSort, 397
- Beta
 - mlpack::nn::SparseAutoencoder, 432, 433
 - mlpack::nn::SparseAutoencoderFunction, 437
- beta
 - mlpack::nn::SparseAutoencoder, 435
 - mlpack::nn::SparseAutoencoderFunction, 440
- BetaPath
 - mlpack::regression::LARS, 528
- betaPath
 - mlpack::regression::LARS, 530
- BinLabels
 - mlpack::decision_stump::DecisionStump, 198
- binLabels
 - mlpack::decision_stump::DecisionStump, 201
- BinarySearch
 - mlpack::tree::CosineTree, 597
- BinarySpaceTree
 - mlpack::tree::BinarySpaceTree, 571–574
- bottom
 - TREE_EXPLANATION.txt, 756
- Bound
 - mlpack::emst::DTBStat, 234
 - mlpack::fastmks::FastMKSSStat, 262
 - mlpack::neighbor::NeighborSearchStat, 413, 414
 - mlpack::tree::BinarySpaceTree, 574, 575
- bound
 - mlpack::emst::DTBStat, 235
 - mlpack::fastmks::FastMKSSStat, 263
 - mlpack::neighbor::NeighborSearchStat, 415
 - mlpack::tree::BinarySpaceTree, 584
- bounds
 - mlpack::bound::HRectBound, 177
- bucketContentSize
 - mlpack::neighbor::LSHSearch, 393
- bucketRowInHashTable
 - mlpack::neighbor::LSHSearch, 393
- bucketSize
 - mlpack::decision_stump::DecisionStump, 201
 - mlpack::neighbor::LSHSearch, 393
- bucketTag
 - mlpack::det::DTree, 212
- BuildHash
 - mlpack::neighbor::LSHSearch, 392
- C
 - mlpack::optimization::LRSDP, 470
 - mlpack::optimization::LRSDPFunction, 474
- c
 - mlpack::optimization::LRSDPFunction, 475
- c_index
 - mlpack::amf::SimpleToleranceTermination, 146
 - mlpack::amf::ValidationRMSETermination, 161
- c_indexOld
 - mlpack::amf::SimpleToleranceTermination, 146
 - mlpack::amf::ValidationRMSETermination, 161
- CF
 - mlpack::cf::CF, 179
- CLI
 - mlpack::CLI, 188, 190
- CLIDeleter
 - mlpack::util::CLIDeleter, 650
- CMakeLists.txt
 - include_directories, 681
- CalculateBound
 - mlpack::emst::DTBRules, 228
 - mlpack::fastmks::FastMKSRules, 256
 - mlpack::neighbor::NeighborSearchRules, 408
- CalculateCentroid
 - mlpack::tree::CosineTree, 597
- CalculateCosines
 - mlpack::tree::CosineTree, 597
- CalculateEntropy
 - mlpack::decision_stump::DecisionStump, 199
- CallBaseLogic
 - mlpack::util::PrefixedOutputStream, 659
- candidate
 - mlpack::radical::Radical, 511
- carriageReturned

- mlpack::util::PrefixedOutputStream, 661
- Center
 - mlpack::bound::BallBound, 166
 - mlpack::math, 105
- center
 - mlpack::bound::BallBound, 169
- CenterOfMass
 - mlpack::tree::MRKDStatistic, 643
- centerOfMass
 - mlpack::tree::MRKDStatistic, 644
- CenterTransformedData
 - mlpack::kpca::KernelPCA, 349
- centerTransformedData
 - mlpack::kpca::KernelPCA, 349
- Centroid
 - mlpack::bound::BallBound, 167
 - mlpack::bound::HRectBound, 173
 - mlpack::tree::BinarySpaceTree, 575
 - mlpack::tree::CosineTree, 598
 - mlpack::tree::CoverTree, 610
 - mlpack::tree::ExampleTree, 632
- centroid
 - mlpack::tree::CosineTree, 601
- ChebyshevDistance
 - mlpack::metric, 109
- Child
 - mlpack::tree::BinarySpaceTree, 575
 - mlpack::tree::CoverTree, 611
 - mlpack::tree::ExampleTree, 634
- Children
 - mlpack::tree::CoverTree, 611
- children
 - mlpack::tree::CoverTree, 619
- CholeskyDelete
 - mlpack::regression::LARS, 529
- CholeskyInsert
 - mlpack::regression::LARS, 529
- ChooseRoot
 - mlpack::tree::FirstPointsRoot, 638
- ChooseScalingFactor
 - mlpack::optimization::L_BFGS, 459
- ClampNonNegative
 - mlpack::math, 105
- ClampNonPositive
 - mlpack::math, 105
- ClampRange
 - mlpack::math, 106
- classLabels
 - mlpack::perceptron::Perceptron, 503
- Classify
 - mlpack::decision_stump::DecisionStump, 199
 - mlpack::gmm::GMM, 276
 - mlpack::naive_bayes::NaiveBayesClassifier, 374
 - mlpack::perceptron::Perceptron, 503
- CleanData
 - mlpack::cf::CF, 179
- CleanedData
 - mlpack::cf::CF, 179
- cleanedData
 - mlpack::cf::CF, 182
- Cleanup
 - mlpack::emst::DualTreeBoruvka, 239
- CleanupHelper
 - mlpack::emst::DualTreeBoruvka, 239
- Clear
 - mlpack::bound::HRectBound, 174
- cli.hpp
 - PARAM, 760
 - PARAM_DOUBLE, 761
 - PARAM_DOUBLE_REQ, 761
 - PARAM_FLAG, 762
 - PARAM_FLOAT, 762
 - PARAM_FLOAT_REQ, 762
 - PARAM_INT, 763
 - PARAM_INT_REQ, 763
 - PARAM_STRING, 764
 - PARAM_STRING_REQ, 764
 - PARAM_VECTOR, 765
 - PARAM_VECTOR_REQ, 765
 - PROGRAM_INFO, 766
 - TYPENAME, 766
- cliDeleter
 - mlpack::util, 118
- Cluster
 - mlpack::kmeans::KMeans, 336
 - mlpack::kmeans::RandomPartition, 343
 - mlpack::kmeans::RefinedStart, 344
- Clusterer
 - mlpack::gmm::EMFit, 268
- clusterer
 - mlpack::gmm::EMFit, 270
- cmake_minimum_required
 - methods/decision_stump/CMakeLists.txt, 690
 - methods/perceptron/CMakeLists.txt, 698
- Codes
 - mlpack::lcc::LocalCoordinateCoding, 354
 - mlpack::sparse_coding::SparseCoding, 550
- codes
 - mlpack::lcc::LocalCoordinateCoding, 356
 - mlpack::sparse_coding::SparseCoding, 553
- ColumnSampleLS
 - mlpack::tree::CosineTree, 598
- ColumnSamplesLS
 - mlpack::tree::CosineTree, 598
- CombineBest
 - mlpack::neighbor::FurthestNeighborSort, 387
 - mlpack::neighbor::NearestNeighborSort, 397
- CombineWorst

- mlpack::neighbor::FurthestNeighborSort, 387
- mlpack::neighbor::NearestNeighborSort, 398
- CompleteIncrementalTermination
 - mlpack::amf::CompleteIncrementalTermination, 128
- ComponentMembership
 - mlpack::emst::DTBStat, 234
- componentMembership
 - mlpack::emst::DTBStat, 235
- ComputeAccuracy
 - mlpack::regression::LogisticRegression, 538
- ComputeDistances
 - mlpack::tree::CoverTree, 611
- ComputeError
 - mlpack::regression::LinearRegression, 534
 - mlpack::regression::LogisticRegression, 539
- ComputeMST
 - mlpack::emst::DualTreeBoruvka, 239
- ComputeValue
 - mlpack::det::DTree, 207
- ComputeVariableImportance
 - mlpack::det::DTree, 207
- ComputeYHatDirection
 - mlpack::regression::LARS, 529
- connections
 - mlpack::emst::DTBRules, 230
 - mlpack::emst::DualTreeBoruvka, 240
- Constraint
 - mlpack::gmm::EMFit, 268
- constraint
 - mlpack::gmm::EMFit, 271
- ConstructBasis
 - mlpack::tree::CosineTree, 598
- Contains
 - mlpack::bound::BallBound, 167
 - mlpack::bound::HRectBound, 174
 - mlpack::math::Range, 361
- ConvolutionIntegral
 - mlpack::kernel::EpanechnikovKernel, 295
 - mlpack::kernel::ExampleKernel, 298
 - mlpack::kernel::GaussianKernel, 301
 - mlpack::kernel::SphericalKernel, 328
- coolingSchedule
 - mlpack::optimization::SA, 483
- CopyAndPerturb
 - mlpack::radical::Radical, 509
- CopyMe
 - mlpack::tree::BinarySpaceTree, 575
- core/CMakeLists.txt
 - add_subdirectory, 681
 - set, 681
- core/data/CMakeLists.txt
 - set, 681
- core/dists/CMakeLists.txt
 - set, 682
- core/kernels/CMakeLists.txt
 - set, 682
- core/math/CMakeLists.txt
 - set, 683
- core/metrics/CMakeLists.txt
 - set, 683
- core/optimizers/CMakeLists.txt
 - add_subdirectory, 684
 - set, 684
- core/optimizers/aug_lagrangian/CMakeLists.txt
 - set, 684
- core/optimizers/lbfgs/CMakeLists.txt
 - set, 685
- core/optimizers/lrsdp/CMakeLists.txt
 - set, 685
- core/optimizers/sa/CMakeLists.txt
 - set, 685
- core/optimizers/sgd/CMakeLists.txt
 - set, 686
- core/tree/CMakeLists.txt
 - set, 686
- core/util/CMakeLists.txt
 - set, 687
- CosineNodeQueue
 - mlpack::tree, 117
- CosineNodeSplit
 - mlpack::tree::CosineTree, 598
- CosineTree
 - mlpack::tree::CosineTree, 596
- Count
 - mlpack::tree::BinarySpaceTree, 575
 - mlpack::tree::MRKDStatistic, 643
- count
 - mlpack::tree::BinarySpaceTree, 584
 - mlpack::tree::MRKDStatistic, 644
- CountMostFreq
 - mlpack::decision_stump::DecisionStump, 199
- Counts
 - mlpack::kernel::PSpectrumStringKernel, 324
- counts
 - mlpack::kernel::PSpectrumStringKernel, 325
- cout
 - mlpack::Log, 359
- Covariance
 - mlpack::distribution::GaussianDistribution, 220
 - mlpack::metric::MahalanobisDistance, 371
- covariance
 - mlpack::distribution::GaussianDistribution, 221
 - mlpack::metric::MahalanobisDistance, 372
- Covariances
 - mlpack::gmm::GMM, 276
- covariances
 - mlpack::gmm::GMM, 282
- CoverTree

- mlpack::tree::CoverTree, 608–610
- CreateChildren
 - mlpack::tree::CoverTree, 612
- currentItemIndex
 - mlpack::amf::SVDCompleteIncrementalLearning, 153
- currentUserIndex
 - mlpack::amf::SVDCompleteIncrementalLearning, 153
 - mlpack::amf::SVDIncompleteIncrementalLearning, 158
- DTBRules
 - mlpack::emst::DTBRules, 227
- DTBStat
 - mlpack::emst::DTBStat, 232
- DTree
 - mlpack::det::DTree, 205, 207
- Data
 - mlpack::cf::CF, 180
 - mlpack::lcc::LocalCoordinateCoding, 354
 - mlpack::sparse_coding::SparseCoding, 550
- data
 - mlpack::cf::CF, 182
 - mlpack::emst::DualTreeBoruvka, 240
 - mlpack::kernel::NystroemMethod, 318
 - mlpack::lcc::LocalCoordinateCoding, 356
 - mlpack::mvu::MVU, 373
 - mlpack::nn::SparseAutoencoderFunction, 440
 - mlpack::sparse_coding::SparseCoding, 553
 - mlpack::svd::RegularizedSVD, 557
 - mlpack::svd::RegularizedSVDFunction, 561
- dataCopy
 - mlpack::emst::DualTreeBoruvka, 240
- dataSet
 - mlpack::emst::DTBRules, 230
- Dataset
 - mlpack::nca::NCA, 378
 - mlpack::svd::RegularizedSVDFunction, 559
 - mlpack::tree::BinarySpaceTree, 576
 - mlpack::tree::CoverTree, 612
- dataset
 - mlpack::nca::NCA, 379
 - mlpack::nca::SoftmaxErrorFunction, 383
 - mlpack::svd::QUIC_SVD, 555
 - mlpack::tree::BinarySpaceTree, 584
 - mlpack::tree::CosineTree, 601
 - mlpack::tree::CoverTree, 619
 - mlpack::tree::MRKDStatistic, 644
- datasets
 - mlpack::kernel::PSpectrumStringKernel, 325
- Deactivate
 - mlpack::regression::LARS, 529
- Debug
 - mlpack::Log, 359
- DecisionStump
 - mlpack::decision_stump::DecisionStump, 198
- DefaultMessages
 - mlpack::CLI, 192
- Degree
 - mlpack::kernel::PolynomialKernel, 321
- degree
 - mlpack::kernel::PolynomialKernel, 322
- delta
 - mlpack::svd::QUIC_SVD, 555
 - mlpack::tree::CosineTree, 601
- denominators
 - mlpack::nca::SoftmaxErrorFunction, 383
- desc
 - mlpack::CLI, 195
 - mlpack::ParamData, 497
- Descendant
 - mlpack::tree::BinarySpaceTree, 576
 - mlpack::tree::CoverTree, 612
 - mlpack::tree::ExampleTree, 634
- destination
 - mlpack::util::PrefixedOutputStream, 661
- Destroy
 - mlpack::CLI, 192
- det.txt
 - \$V, 678
 - alpha, 678
 - estimation, 678
 - now, 678
 - regularization, 678
 - Thus, 678
- Diameter
 - mlpack::bound::BallBound, 167
 - mlpack::bound::HRectBound, 174
- Dictionary
 - mlpack::lcc::LocalCoordinateCoding, 354
 - mlpack::sparse_coding::SparseCoding, 550
- dictionary
 - mlpack::lcc::LocalCoordinateCoding, 357
 - mlpack::sparse_coding::SparseCoding, 553
- didParse
 - mlpack::CLI, 195
- Dim
 - mlpack::bound::BallBound, 167
 - mlpack::bound::HRectBound, 174
- dim
 - mlpack::bound::HRectBound, 177
- Dimensionality
 - mlpack::distribution::DiscreteDistribution, 217
 - mlpack::distribution::GaussianDistribution, 220
 - mlpack::distribution::LaplaceDistribution, 223
 - mlpack::gmm::GMM, 276, 277
 - mlpack::hmm::HMM, 288

- dimensionality
 - mlpack::gmm::GMM, 282
 - mlpack::hmm::HMM, 292
- DiscreteDistribution
 - mlpack::distribution::DiscreteDistribution, 215
- Distance
 - mlpack::emst::EdgePair, 244
- distance
 - mlpack::emst::EdgePair, 245
- DistanceComps
 - mlpack::tree::CoverTree, 612
- distanceComps
 - mlpack::tree::CoverTree, 619
- distancePtr
 - mlpack::neighbor::LSHSearch, 393
- distances
 - mlpack::neighbor::NeighborSearchRules, 410
 - mlpack::neighbor::RASearchRules, 428
 - mlpack::range::RangeSearchRules, 522
- DoRadical
 - mlpack::radical::Radical, 509
- DoRadical2D
 - mlpack::radical::Radical, 509
- doc
 - mlpack::CLI, 195
- doc/guide/build.hpp, 677
- doc/guide/iodoc.hpp, 677
- doc/guide/matrices.hpp, 677
- doc/guide/sample.hpp, 677
- doc/guide/timer.hpp, 677
- doc/guide/version.hpp, 776
- doc/tutorials/amf/amf.txt, 677
- doc/tutorials/det/det.txt, 677
- doc/tutorials/emst/emst.txt, 679
- doc/tutorials/fastmks/fastmks.txt, 679
- doc/tutorials/kmeans/kmeans.txt, 679
- doc/tutorials/linear_regression/linear_regression.txt, 679
- doc/tutorials/neighbor_search/neighbor_search.txt, 680
- doc/tutorials/range_search/range_search.txt, 680
- doc/tutorials/tutorials.txt, 680
- documentation
 - mlpack::util::ProgramDoc, 663
- DominatingCentroid
 - mlpack::tree::MRKDStatistic, 643
- dominatingCentroid
 - mlpack::tree::MRKDStatistic, 644
- DualTreeBoruvka
 - mlpack::emst::DualTreeBoruvka, 238
- DualTreeTraverser
 - mlpack::tree::BinarySpaceTree::DualTreeTraverser, 587
 - mlpack::tree::CoverTree::DualTreeTraverser, 622
- dummy
 - mlpack::amf::SVDCompleteIncrementalLearning<arma::sp_mat>, 156
- EMFit
 - mlpack::gmm::EMFit, 267
- EdgePair
 - mlpack::emst::EdgePair, 244
- Edges
 - mlpack::optimization::LovaszThetaSDP, 467
- edges
 - mlpack::emst::DualTreeBoruvka, 240
 - mlpack::optimization::LovaszThetaSDP, 467
- EigenvalueRatioConstraint
 - mlpack::gmm::EigenvalueRatioConstraint, 265
- elasticNet
 - mlpack::regression::LARS, 530
- Emission
 - mlpack::hmm::HMM, 288
- emission
 - mlpack::hmm::HMM, 292
- EmitResults
 - mlpack::emst::DualTreeBoruvka, 240
- EmptyCluster
 - mlpack::kmeans::AllowEmptyClusters, 333
 - mlpack::kmeans::MaxVarianceNewCluster, 340
- EmptyClusterAction
 - mlpack::kmeans::KMeans, 337
- emptyClusterAction
 - mlpack::kmeans::KMeans, 339
- EmptyStatistic
 - mlpack::tree::EmptyStatistic, 629, 630
- Encode
 - mlpack::lcc::LocalCoordinateCoding, 354
 - mlpack::sparse_coding::SparseCoding, 550
- End
 - mlpack::det::DTree, 208
 - mlpack::tree::BinarySpaceTree, 576
- end
 - mlpack::det::DTree, 212
- EpanechnikovKernel
 - mlpack::kernel::EpanechnikovKernel, 295
- epsilon
 - mlpack::svd::QUIC_SVD, 555
 - mlpack::tree::CosineTree, 601
- errorFunction
 - mlpack::nca::NCA, 379
- Estimate
 - mlpack::distribution::DiscreteDistribution, 217
 - mlpack::distribution::GaussianDistribution, 220
 - mlpack::distribution::LaplaceDistribution, 223, 224
 - mlpack::gmm::EMFit, 268, 269
 - mlpack::gmm::GMM, 277
 - mlpack::hmm::HMM, 288, 289
- estimation

- det.txt, 678
- EuclideanDistance
 - mlpack::metric, 109
- Evaluate
 - mlpack::kernel::CosineDistance, 294
 - mlpack::kernel::EpanechnikovKernel, 295, 296
 - mlpack::kernel::ExampleKernel, 298
 - mlpack::kernel::GaussianKernel, 301
 - mlpack::kernel::HyperbolicTangentKernel, 305
 - mlpack::kernel::LaplacianKernel, 314
 - mlpack::kernel::LinearKernel, 316
 - mlpack::kernel::PSpectrumStringKernel, 324
 - mlpack::kernel::PolynomialKernel, 321
 - mlpack::kernel::SphericalKernel, 328
 - mlpack::kernel::TriangularKernel, 330
 - mlpack::metric::IPMetric, 366
 - mlpack::metric::LMetric, 368
 - mlpack::metric::MahalanobisDistance, 371
 - mlpack::nca::SoftmaxErrorFunction, 382
 - mlpack::nn::SparseAutoencoderFunction, 437
 - mlpack::optimization::AugLagrangianFunction, 448
 - mlpack::optimization::AugLagrangianTestFunction, 451
 - mlpack::optimization::GockenbachFunction, 454
 - mlpack::optimization::L_BFGS, 459
 - mlpack::optimization::LRSDPFunction, 474
 - mlpack::optimization::LovaszThetaSDP, 467
 - mlpack::optimization::test::GeneralizedRosenbrockFunction, 492
 - mlpack::optimization::test::RosenbrockFunction, 493
 - mlpack::optimization::test::RosenbrockWoodFunction, 494
 - mlpack::optimization::test::SGDTestFunction, 495
 - mlpack::optimization::test::WoodFunction, 496
 - mlpack::regression::LogisticRegressionFunction, 542
 - mlpack::svd::RegularizedSVDFunction, 559
- EvaluateConstraint
 - mlpack::optimization::AugLagrangianTestFunction, 451
 - mlpack::optimization::GockenbachFunction, 454
 - mlpack::optimization::LRSDPFunction, 474
 - mlpack::optimization::LovaszThetaSDP, 467
- ExampleKernel
 - mlpack::kernel::ExampleKernel, 297
- ExampleTree
 - mlpack::tree::ExampleTree, 632
- ExponentialSchedule
 - mlpack::optimization::ExponentialSchedule, 452
- ExtendTree
 - mlpack::tree::BinarySpaceTree, 576
- ExtractSVD
 - mlpack::svd::QUIC_SVD, 554
- Factorizer
 - mlpack::cf::CF, 180
- factorizer
 - mlpack::cf::CF, 182
- FastMKS
 - mlpack::fastmks::FastMKS, 249, 250
- FastMKSRules
 - mlpack::fastmks::FastMKSRules, 256
- FastMKSSStat
 - mlpack::fastmks::FastMKSSStat, 261
- Fatal
 - mlpack::Log, 359
- fatal
 - mlpack::util::PrefixedOutputStream, 661
- FileTimeToTimeVal
 - mlpack::Timers, 564
- Find
 - mlpack::emst::UnionFind, 246
- FindBucket
 - mlpack::det::DTree, 208
- FindByBeginCount
 - mlpack::tree::BinarySpaceTree, 576, 577
- FindSplit
 - mlpack::det::DTree, 208
- FirstBound
 - mlpack::neighbor::NeighborSearchStat, 414
- firstBound
 - mlpack::neighbor::NeighborSearchStat, 415
- firstLeafExact
 - mlpack::neighbor::RASearchRules, 428
- FirstPointIsCentroid
 - mlpack::tree::TreeTraits, 647
 - mlpack::tree::TreeTraits< BinarySpaceTree< BoundType, StatisticType, MatType > >, 648
 - mlpack::tree::TreeTraits< CoverTree< MetricType, RootPointPolicy, StatisticType > >, 649
- Fitter
 - mlpack::gmm::GMM, 279
- fitter
 - mlpack::gmm::GMM, 282
- force_inline
 - prereqs.hpp, 857
- Forward
 - mlpack::hmm::HMM, 289
- FrobNormSquared
 - mlpack::tree::CosineTree, 598
- frobNormSquared
 - mlpack::tree::CosineTree, 601
- Function
 - mlpack::optimization::AugLagrangian, 443
 - mlpack::optimization::AugLagrangianFunction, 448
 - mlpack::optimization::L_BFGS, 459
 - mlpack::optimization::LRSDP, 471
 - mlpack::optimization::SA, 479
 - mlpack::optimization::SGD, 488

- function
 - mlpack::optimization::AugLagrangian, 445
 - mlpack::optimization::AugLagrangianFunction, 450
 - mlpack::optimization::L_BFGS, 463
 - mlpack::optimization::LRSDP, 471
 - mlpack::optimization::SA, 484
 - mlpack::optimization::SGD, 490
- FurthestDescendantDistance
 - mlpack::tree::BinarySpaceTree, 577
 - mlpack::tree::CoverTree, 612, 613
 - mlpack::tree::ExampleTree, 634
- furthestDescendantDistance
 - mlpack::tree::BinarySpaceTree, 584
 - mlpack::tree::CoverTree, 619
- FurthestPointDistance
 - mlpack::tree::BinarySpaceTree, 577
 - mlpack::tree::CoverTree, 613
- GMM
 - mlpack::gmm::GMM, 274–276
- Gain
 - mlpack::optimization::SA, 479
- gain
 - mlpack::optimization::SA, 484
- Gamma
 - mlpack::kernel::GaussianKernel, 303
- gamma
 - mlpack::kernel::GaussianKernel, 303
- GaussianDistribution
 - mlpack::distribution::GaussianDistribution, 219
- GaussianKernel
 - mlpack::kernel::GaussianKernel, 300
- Gaussians
 - mlpack::gmm::GMM, 279
- gaussians
 - mlpack::gmm::GMM, 282
- GeneralizedRosenbrockFunction
 - mlpack::optimization::test::GeneralizedRosenbrock↵Function, 492
- Generate
 - mlpack::hmm::HMM, 289
- GenerateMove
 - mlpack::optimization::SA, 479
- Get
 - mlpack::Timer, 563
- GetAllTimers
 - mlpack::Timers, 564
- GetDataset
 - mlpack::tree::CosineTree, 598
- GetDescription
 - mlpack::CLI, 192
- GetFinalBasis
 - mlpack::tree::CosineTree, 599
- GetInitialPoint
 - mlpack::nca::SoftmaxErrorFunction, 382
 - mlpack::nn::SparseAutoencoderFunction, 438
 - mlpack::optimization::AugLagrangianFunction, 448
 - mlpack::optimization::AugLagrangianTestFunction, 451
 - mlpack::optimization::GockenbachFunction, 454
 - mlpack::optimization::LRSDPFunction, 474
 - mlpack::optimization::LovaszThetaSDP, 467
 - mlpack::optimization::test::GeneralizedRosenbrock↵Function, 492
 - mlpack::optimization::test::RosenbrockFunction, 493
 - mlpack::optimization::test::RosenbrockWood↵Function, 494
 - mlpack::optimization::test::SGDTestFunction, 495
 - mlpack::optimization::test::WoodFunction, 496
 - mlpack::regression::LogisticRegressionFunction, 543
 - mlpack::svd::RegularizedSVDFunction, 559
- GetKernelMatrix
 - mlpack::kernel::NystroemMethod, 318
- GetNewFeatures
 - mlpack::nn::SparseAutoencoder, 433
- GetParam
 - mlpack::CLI, 192
- GetRecommendations
 - mlpack::cf::CF, 180
- GetSingleton
 - mlpack::CLI, 193
- GetSplitDimension
 - mlpack::tree::BinarySpaceTree, 577
- GetTime
 - mlpack::Timers, 564
- GetTimer
 - mlpack::Timers, 565
- GetVersion
 - mlpack::util, 118
- GivensRotate
 - mlpack::regression::LARS, 529
- globalValues
 - mlpack::CLI, 195
- gmap_t
 - mlpack::CLI, 188
- GockenbachFunction
 - mlpack::optimization::GockenbachFunction, 454
- Gradient
 - mlpack::nca::SoftmaxErrorFunction, 382, 383
 - mlpack::nn::SparseAutoencoderFunction, 438
 - mlpack::optimization::AugLagrangianFunction, 449
 - mlpack::optimization::AugLagrangianTestFunction, 451
 - mlpack::optimization::GockenbachFunction, 454
 - mlpack::optimization::LRSDPFunction, 474
 - mlpack::optimization::LovaszThetaSDP, 467
 - mlpack::optimization::test::GeneralizedRosenbrock↵Function, 492

- mlpack::optimization::test::RosenbrockFunction, 493
- mlpack::optimization::test::RosenbrockWood↵
Function, 494
- mlpack::optimization::test::SGDTestFunction, 495
- mlpack::optimization::test::WoodFunction, 496
- mlpack::regression::LogisticRegressionFunction, 543
- mlpack::svd::RegularizedSVDFunction, 559
- GradientConstraint
 - mlpack::optimization::AugLagrangianTestFunction,
451
 - mlpack::optimization::GockenbachFunction, 454
 - mlpack::optimization::LRSDPFunction, 474
 - mlpack::optimization::LovaszThetaSDP, 467
- GradientNormTooSmall
 - mlpack::optimization::L_BFGS, 459
- Greater
 - mlpack::emst::EdgePair, 244
- greater
 - mlpack::emst::EdgePair, 245
- Grow
 - mlpack::det::DTree, 208
- H
 - mlpack::amf::SimpleToleranceTermination, 146
 - mlpack::amf::ValidationRMSETermination, 161
 - mlpack::cf::CF, 181
- h
 - mlpack::cf::CF, 183
- HAS_MEM_FUNC
 - sfinae_utility.hpp, 772
- HMM
 - mlpack::hmm::HMM, 286, 287
- HRectBound
 - mlpack::bound::HRectBound, 173
- HUpdate
 - mlpack::amf::NMFALSUpdate, 133
 - mlpack::amf::NMFMultiplicativeDistanceUpdate, 135
 - mlpack::amf::NMFMultiplicativeDivergenceUpdate,
137
 - mlpack::amf::SVDBatchLearning, 149
 - mlpack::amf::SVDCompleteIncrementalLearning,
152
 - mlpack::amf::SVDCompleteIncrementalLearning<
arma::sp_mat >, 155
 - mlpack::amf::SVDIncompleteIncrementalLearning,
158
- HasOverlappingChildren
 - mlpack::tree::TreeTraits, 647
 - mlpack::tree::TreeTraits< BinarySpaceTree<
BoundType, StatisticType, MatType > >, 648
 - mlpack::tree::TreeTraits< CoverTree< MetricType,
RootPointPolicy, StatisticType > >, 649
- HasParam
 - mlpack::CLI, 193
- hasQuerySet
 - mlpack::neighbor::NeighborSearch, 403
 - mlpack::range::RangeSearch, 516
 - RASearch, 671
- HasSelfChildren
 - mlpack::tree::BinarySpaceTree, 578
 - mlpack::tree::CoverTree, 613
 - mlpack::tree::TreeTraits, 647
 - mlpack::tree::TreeTraits< BinarySpaceTree<
BoundType, StatisticType, MatType > >, 648
 - mlpack::tree::TreeTraits< CoverTree< MetricType,
RootPointPolicy, StatisticType > >, 649
- hashWidth
 - mlpack::neighbor::LSHSearch, 393
- Hi
 - mlpack::math::Range, 362
- hi
 - mlpack::math::Range, 364
- HiddenSize
 - mlpack::nn::SparseAutoencoder, 433
 - mlpack::nn::SparseAutoencoderFunction, 438
- hiddenSize
 - mlpack::nn::SparseAutoencoder, 435
 - mlpack::nn::SparseAutoencoderFunction, 440
- HyperbolicTangentKernel
 - mlpack::kernel::HyperbolicTangentKernel, 305
- HyphenateString
 - mlpack::CLI, 193
- IPMetric
 - mlpack::metric::IPMetric, 365
- Ignore
 - mlpack::regression::LARS, 529
- ignoreInput
 - mlpack::util::PrefixedOutputStream, 661
- ignoreSet
 - mlpack::regression::LARS, 530
- include_directories
 - CMakeLists.txt, 681
- IncompleteIncrementalTermination
 - mlpack::amf::IncompleteIncrementalTermination, 131
- incrementalIndex
 - mlpack::amf::CompleteIncrementalTermination, 130
 - mlpack::amf::IncompleteIncrementalTermination, 132
- Indent
 - mlpack::util, 118
- Index
 - mlpack::amf::CompleteIncrementalTermination, 129
 - mlpack::amf::IncompleteIncrementalTermination, 131
 - mlpack::amf::SimpleResidueTermination, 140
 - mlpack::amf::SimpleToleranceTermination, 144
 - mlpack::amf::ValidationRMSETermination, 160
- indices
 - mlpack::fastmks::FastMKSRules, 259

- mlpack::tree::CosineTree, 601
- Info
 - mlpack::Log, 359
- InitMoves
 - mlpack::optimization::SA, 480
- initMoves
 - mlpack::optimization::SA, 484
- Initial
 - mlpack::hmm::HMM, 290
- initial
 - mlpack::hmm::HMM, 292
- InitialClustering
 - mlpack::gmm::EMFit, 269
- InitialPoint
 - mlpack::regression::LogisticRegressionFunction, 543
- initialPoint
 - mlpack::nn::SparseAutoencoderFunction, 440
 - mlpack::optimization::AugLagrangianTestFunction, 451
 - mlpack::optimization::GockenbachFunction, 455
 - mlpack::optimization::LRSDPFunction, 475
 - mlpack::optimization::LovaszThetaSDP, 467
 - mlpack::optimization::test::GeneralizedRosenbrockFunction, 492
 - mlpack::optimization::test::RosenbrockFunction, 493
 - mlpack::optimization::test::RosenbrockWoodFunction, 494
 - mlpack::optimization::test::WoodFunction, 496
 - mlpack::regression::LogisticRegressionFunction, 544
 - mlpack::svd::RegularizedSVDFunction, 561
- initializationRule
 - mlpack::amf::AMF, 126
- Initialize
 - mlpack::amf::AverageInitialization, 127
 - mlpack::amf::CompleteIncrementalTermination, 129
 - mlpack::amf::IncompleteIncrementalTermination, 131
 - mlpack::amf::NMFALSUpdate, 134
 - mlpack::amf::NMFMultiplicativeDistanceUpdate, 135
 - mlpack::amf::NMFMultiplicativeDivergenceUpdate, 137
 - mlpack::amf::RandomAcolInitialization, 138
 - mlpack::amf::RandomInitialization, 139
 - mlpack::amf::SVDBatchLearning, 150
 - mlpack::amf::SVDCompleteIncrementalLearning, 152
 - mlpack::amf::SVDCompleteIncrementalLearning< arma::sp_mat >, 155
 - mlpack::amf::SVDIncompleteIncrementalLearning, 158
 - mlpack::amf::SimpleResidueTermination, 140
 - mlpack::amf::SimpleToleranceTermination, 144
 - mlpack::amf::ValidationRMSETermination, 160
 - mlpack::perceptron::RandomInitialization, 504
 - mlpack::perceptron::ZeroInitialization, 506
 - mlpack::sparse_coding::DataDependentRandomInitializer, 545
 - mlpack::sparse_coding::NothingInitializer, 546
 - mlpack::sparse_coding::RandomInitializer, 547
- InitializeRule
 - mlpack::amf::AMF, 125
- InitializeWeights
 - mlpack::nn::SparseAutoencoderFunction, 438
- InsertNeighbor
 - mlpack::cf::CF, 181
 - mlpack::fastmks::FastMKS, 251
 - mlpack::fastmks::FastMKSRules, 257
 - mlpack::neighbor::LSHSearch, 392
 - mlpack::neighbor::NeighborSearchRules, 408
 - mlpack::neighbor::RASearchRules, 422
- intercept
 - mlpack::regression::LinearRegression, 536
- InterpolateBeta
 - mlpack::regression::LARS, 529
- inverseBandwidthSquared
 - mlpack::kernel::EpanechnikovKernel, 296
- isActive
 - mlpack::regression::LARS, 530
- IsBetter
 - mlpack::neighbor::FurthestNeighborSort, 387
 - mlpack::neighbor::NearestNeighborSort, 398
- IsConverged
 - mlpack::amf::CompleteIncrementalTermination, 129
 - mlpack::amf::IncompleteIncrementalTermination, 131
 - mlpack::amf::SimpleResidueTermination, 141
 - mlpack::amf::SimpleToleranceTermination, 145
 - mlpack::amf::ValidationRMSETermination, 160
- isCopy
 - mlpack::amf::SimpleToleranceTermination, 146
 - mlpack::amf::ValidationRMSETermination, 161
- IsDistinct
 - mlpack::decision_stump::DecisionStump, 199
- isFlag
 - mlpack::ParamData, 497
- isIgnored
 - mlpack::regression::LARS, 530
- IsLeaf
 - mlpack::tree::BinarySpaceTree, 578
 - mlpack::tree::CoverTree, 613
- IsNormalized
 - mlpack::kernel::KernelTraits, 307
 - mlpack::kernel::KernelTraits< CosineDistance >, 307
 - mlpack::kernel::KernelTraits< EpanechnikovKernel >, 308
 - mlpack::kernel::KernelTraits< GaussianKernel >, 309
 - mlpack::kernel::KernelTraits< LaplacianKernel >, 309

- mlpack::kernel::KernelTraits< SphericalKernel >, 310
- mlpack::kernel::KernelTraits< TriangularKernel >, 311
- isStart
 - mlpack::amf::SVDCompleteIncrementalLearning< arma::sp_mat >, 156
- IsVector
 - value, 119
- IsVector< arma::Col< eT > >, 120
 - value, 120
- IsVector< arma::Row< eT > >, 120
 - value, 120
- IsVector< arma::SpCol< eT > >, 120
 - value, 121
- IsVector< arma::SpRow< eT > >, 121
 - value, 121
- IsVector< arma::SpSubview< eT > >, 121
 - value, 122
- IsVector< arma::SubviewCol< eT > >, 122
 - value, 122
- IsVector< arma::SubviewRow< eT > >, 122
 - value, 123
- IsVector< VecType >, 119
- isWhitelistValid
 - mlpack::tree::MRKDStatistic, 645
- it
 - mlpack::amf::SVDCompleteIncrementalLearning< arma::sp_mat >, 156
- iter
 - mlpack::perceptron::Perceptron, 503
- Iteration
 - mlpack::amf::CompleteIncrementalTermination, 129
 - mlpack::amf::IncompleteIncrementalTermination, 131
 - mlpack::amf::SimpleResidueTermination, 141
 - mlpack::amf::SimpleToleranceTermination, 145
 - mlpack::amf::ValidationRMSETermination, 161
- iteration
 - mlpack::amf::CompleteIncrementalTermination, 130
 - mlpack::amf::IncompleteIncrementalTermination, 132
 - mlpack::amf::SimpleResidueTermination, 142
 - mlpack::amf::SimpleToleranceTermination, 146
 - mlpack::amf::ValidationRMSETermination, 162
- iterations
 - mlpack::svd::RegularizedSVD, 557
- KMeans
 - mlpack::kmeans::KMeans, 335
- Kernel
 - mlpack::kpca::KernelPCA, 349
 - mlpack::metric::IPMetric, 366
- kernel
 - mlpack::fastmks::FastMKSRules, 259
 - mlpack::kernel::NystroemMethod, 318
 - mlpack::kpca::KernelPCA, 350
 - mlpack::metric::IPMetric, 366
 - KernelPCA
 - mlpack::kpca::KernelPCA, 346
 - kh
 - mlpack::amf::SVDBatchLearning, 150
 - mlpack::amf::SVDCompleteIncrementalLearning, 153
 - mlpack::amf::SVDCompleteIncrementalLearning< arma::sp_mat >, 156
 - mlpack::amf::SVDIncompleteIncrementalLearning, 158
 - kw
 - mlpack::amf::SVDBatchLearning, 150
 - mlpack::amf::SVDCompleteIncrementalLearning, 153
 - mlpack::amf::SVDCompleteIncrementalLearning< arma::sp_mat >, 156
 - mlpack::amf::SVDIncompleteIncrementalLearning, 159
 - L2Error
 - mlpack::tree::CosineTree, 599
 - l2Error
 - mlpack::tree::CosineTree, 601
 - l2NormsSquared
 - mlpack::tree::CosineTree, 602
 - L_BFGS
 - mlpack::optimization::L_BFGS, 458
 - L_BFGSType
 - mlpack::optimization::AugLagrangian, 443
 - LARS
 - mlpack::regression::LARS, 528
 - LBFGS
 - mlpack::optimization::AugLagrangian, 444
 - LMetric
 - mlpack::metric::LMetric, 368
 - LRSDP
 - mlpack::optimization::LRSDP, 469
 - LRSDPFunction
 - mlpack::optimization::LRSDPFunction, 473
 - LSHSearch
 - mlpack::neighbor::LSHSearch, 390
 - Labels
 - mlpack::nca::NCA, 378
 - labels
 - mlpack::nca::NCA, 379
 - mlpack::nca::SoftmaxErrorFunction, 384
 - Lambda
 - mlpack::nn::SparseAutoencoder, 433
 - mlpack::nn::SparseAutoencoderFunction, 438, 439
 - mlpack::optimization::AugLagrangian, 444
 - mlpack::optimization::AugLagrangianFunction, 449
 - mlpack::optimization::ExponentialSchedule, 453

- mlpack::regression::LinearRegression, 534
- mlpack::regression::LogisticRegression, 539
- mlpack::regression::LogisticRegressionFunction, 543, 544
- mlpack::svd::RegularizedSVDFunction, 561
- lambda
 - mlpack::lcc::LocalCoordinateCoding, 357
 - mlpack::nn::SparseAutoencoder, 435
 - mlpack::nn::SparseAutoencoderFunction, 440
 - mlpack::optimization::AugLagrangianFunction, 450
 - mlpack::optimization::ExponentialSchedule, 453
 - mlpack::regression::LinearRegression, 536
 - mlpack::regression::LogisticRegression, 540
 - mlpack::regression::LogisticRegressionFunction, 544
 - mlpack::svd::RegularizedSVD, 557
 - mlpack::svd::RegularizedSVDFunction, 562
- lambda1
 - mlpack::regression::LARS, 531
 - mlpack::sparse_coding::SparseCoding, 553
- lambda2
 - mlpack::regression::LARS, 531
 - mlpack::sparse_coding::SparseCoding, 553
- LambdaPath
 - mlpack::regression::LARS, 529
- lambdaPath
 - mlpack::regression::LARS, 531
- LaplaceDistribution
 - mlpack::distribution::LaplaceDistribution, 223
- LaplacianKernel
 - mlpack::kernel::LaplacianKernel, 312
- lasso
 - mlpack::regression::LARS, 531
- LastBaseCase
 - mlpack::neighbor::NeighborSearchTraversallInfo, 417
 - TraversallInfo, 675
- lastBaseCase
 - mlpack::neighbor::NeighborSearchRules, 410
 - mlpack::neighbor::NeighborSearchTraversallInfo, 419
 - TraversallInfo, 676
- lastCoordinates
 - mlpack::nca::SoftmaxErrorFunction, 384
- LastDistance
 - mlpack::neighbor::NeighborSearchStat, 414
 - mlpack::range::RangeSearchStat, 524
- lastDistance
 - mlpack::neighbor::NeighborSearchStat, 415
 - mlpack::range::RangeSearchStat, 525
- LastDistanceNode
 - mlpack::neighbor::NeighborSearchStat, 414, 415
 - mlpack::range::RangeSearchStat, 524, 525
- lastDistanceNode
 - mlpack::neighbor::NeighborSearchStat, 416
 - mlpack::range::RangeSearchStat, 525
- LastKernel
 - mlpack::fastmks::FastMKSSStat, 262
- lastKernel
 - mlpack::fastmks::FastMKSRules, 259
 - mlpack::fastmks::FastMKSSStat, 263
- LastKernelNode
 - mlpack::fastmks::FastMKSSStat, 262
- lastKernelNode
 - mlpack::fastmks::FastMKSSStat, 263
- lastQueryIndex
 - mlpack::fastmks::FastMKSRules, 259
 - mlpack::neighbor::NeighborSearchRules, 411
 - mlpack::range::RangeSearchRules, 522
- LastQueryNode
 - mlpack::neighbor::NeighborSearchTraversallInfo, 418
 - TraversallInfo, 675
- lastQueryNode
 - mlpack::neighbor::NeighborSearchTraversallInfo, 419
 - TraversallInfo, 676
- lastReferenceIndex
 - mlpack::fastmks::FastMKSRules, 259
 - mlpack::neighbor::NeighborSearchRules, 411
 - mlpack::range::RangeSearchRules, 522
- LastReferenceNode
 - mlpack::neighbor::NeighborSearchTraversallInfo, 418
 - TraversallInfo, 675
- lastReferenceNode
 - mlpack::neighbor::NeighborSearchTraversallInfo, 419
 - TraversallInfo, 676
- LastScore
 - mlpack::neighbor::NeighborSearchTraversallInfo, 418
 - TraversallInfo, 676
- lastScore
 - mlpack::neighbor::NeighborSearchTraversallInfo, 419
 - TraversallInfo, 676
- lbfgs
 - mlpack::optimization::AugLagrangian, 445
- lbfgsInternal
 - mlpack::optimization::AugLagrangian, 445
- LearnDistance
 - mlpack::nca::NCA, 378
- Left
 - mlpack::det::DTree, 208
 - mlpack::tree::BinarySpaceTree, 578
 - mlpack::tree::CosineTree, 599
- left
 - mlpack::det::DTree, 212
 - mlpack::tree::BinarySpaceTree, 584
 - mlpack::tree::CosineTree, 602
- leftStat
 - mlpack::tree::MRKDStatistic, 645
- Lesser
 - mlpack::emst::EdgePair, 244
- lesser
 - mlpack::emst::EdgePair, 245

- LineSearch
 - mlpack::optimization::L_BFGS, 459
- LinearKernel
 - mlpack::kernel::LinearKernel, 316
- LinearRegression
 - mlpack::regression::LinearRegression, 533
- Lo
 - mlpack::math::Range, 362
- lo
 - mlpack::math::Range, 364
- Load
 - mlpack::data, 93
 - mlpack::gmm::GMM, 279
- LoadHMM
 - mlpack::hmm, 101
- LoadParameter
 - mlpack::util::SaveRestoreUtility, 664, 665
- LocalCoordinateCoding
 - mlpack::lcc::LocalCoordinateCoding, 353
- localFitter
 - mlpack::gmm::GMM, 282
- localKernel
 - mlpack::metric::IPMetric, 366
- localMetric
 - mlpack::tree::CoverTree, 620
- LogLikelihood
 - mlpack::gmm::EMFit, 269
 - mlpack::gmm::GMM, 280
 - mlpack::hmm::HMM, 290
- LogNegError
 - mlpack::det::DTree, 209
- logNegError
 - mlpack::det::DTree, 212
- LogNegativeError
 - mlpack::det::DTree, 208
- LogVolume
 - mlpack::det::DTree, 209
- logVolume
 - mlpack::det::DTree, 212
- LogisticRegression
 - mlpack::regression::LogisticRegression, 537, 538
- LogisticRegressionFunction
 - mlpack::regression::LogisticRegressionFunction, 542
- LovaszThetaSDP
 - mlpack::optimization::LovaszThetaSDP, 466
- m
 - mlpack::amf::SVDCompleteIncrementalLearning, 154
 - mlpack::amf::SVDCompleteIncrementalLearning<arma::sp_mat>, 156
 - mlpack::amf::SVDIncompleteIncrementalLearning, 159
 - mlpack::radical::Radical, 511
- M_PI
 - prereqs.hpp, 857
- mH
 - mlpack::amf::SVDBatchLearning, 150
- MRKDStatistic
 - mlpack::tree::MRKDStatistic, 642
- MVU
 - mlpack::mvu::MVU, 372
- mW
 - mlpack::amf::SVDBatchLearning, 151
- MahalanobisDistance
 - mlpack::metric::MahalanobisDistance, 369, 371
- ManhattanDistance
 - mlpack::metric, 109
- Mat
 - mlpack::tree::BinarySpaceTree, 571
 - mlpack::tree::CoverTree, 608
- matGram
 - mlpack::regression::LARS, 531
- matGramInternal
 - mlpack::regression::LARS, 531
- MatUtriCholFactor
 - mlpack::regression::LARS, 529
- matUtriCholFactor
 - mlpack::regression::LARS, 531
- max
 - mlpack::amf::SVDBatchLearning, 150
- MaxDistance
 - mlpack::bound::BallBound, 167
 - mlpack::bound::HRectBound, 174
 - mlpack::tree::BinarySpaceTree, 578
 - mlpack::tree::CoverTree, 613, 614
 - mlpack::tree::ExampleTree, 634
- MaxIterations
 - mlpack::amf::CompleteIncrementalTermination, 129
 - mlpack::amf::IncompleteIncrementalTermination, 132
 - mlpack::amf::SimpleResidueTermination, 141
 - mlpack::amf::SimpleToleranceTermination, 145
 - mlpack::amf::ValidationRMSETermination, 161
 - mlpack::gmm::EMFit, 270
 - mlpack::kmeans::KMeans, 337
 - mlpack::optimization::L_BFGS, 460
 - mlpack::optimization::SA, 480
 - mlpack::optimization::SGD, 488
- maxIterations
 - mlpack::amf::SimpleResidueTermination, 142
 - mlpack::amf::SimpleToleranceTermination, 147
 - mlpack::amf::ValidationRMSETermination, 162
 - mlpack::gmm::EMFit, 271
 - mlpack::kmeans::KMeans, 339
 - mlpack::optimization::L_BFGS, 464
 - mlpack::optimization::SA, 484
 - mlpack::optimization::SGD, 490
- MaxLeafSize

- mlpack::tree::BinarySpaceTree, 579
- maxLeafSize
 - mlpack::tree::BinarySpaceTree, 585
- MaxLineSearchTrials
 - mlpack::optimization::L_BFGS, 460
- maxLineSearchTrials
 - mlpack::optimization::L_BFGS, 464
- MaxMove
 - mlpack::optimization::SA, 480, 481
- maxMove
 - mlpack::optimization::SA, 484
- MaxNeighborDistance
 - mlpack::emst::DTBStat, 234
- maxNeighborDistance
 - mlpack::emst::DTBStat, 235
- MaxStep
 - mlpack::optimization::L_BFGS, 460
- maxStep
 - mlpack::optimization::L_BFGS, 464
- MaxToleranceSweep
 - mlpack::optimization::SA, 481
- maxToleranceSweep
 - mlpack::optimization::SA, 484
- MaxVals
 - mlpack::det::DTree, 209
- maxVals
 - mlpack::det::DTree, 212
- MaxVarianceNewCluster
 - mlpack::kmeans::MaxVarianceNewCluster, 340
- Mean
 - mlpack::distribution::GaussianDistribution, 220
 - mlpack::distribution::LaplaceDistribution, 224
- mean
 - mlpack::distribution::GaussianDistribution, 221
 - mlpack::distribution::LaplaceDistribution, 225
- Means
 - mlpack::gmm::GMM, 280
 - mlpack::naive_bayes::NaiveBayesClassifier, 375
- means
 - mlpack::gmm::GMM, 282
 - mlpack::naive_bayes::NaiveBayesClassifier, 376
- MergeRanges
 - mlpack::decision_stump::DecisionStump, 199
- methods/CMakeLists.txt
 - add_subdirectory, 689
 - set, 689
- methods/amf/CMakeLists.txt
 - set, 687
- methods/amf/init_rules/CMakeLists.txt
 - set, 688
- methods/amf/termination_policies/CMakeLists.txt
 - set, 688
- methods/amf/update_rules/CMakeLists.txt
 - set, 689
- methods/cf/CMakeLists.txt
 - set, 689
- methods/decision_stump/CMakeLists.txt
 - cmake_minimum_required, 690
 - set, 690
- methods/det/CMakeLists.txt
 - set, 690
- methods/emst/CMakeLists.txt
 - set, 691
- methods/fastmks/CMakeLists.txt
 - set, 691
- methods/gmm/CMakeLists.txt
 - set, 692
- methods/hmm/CMakeLists.txt
 - set, 692
- methods/kernel_pca/CMakeLists.txt
 - set, 692
- methods/kernel_pca/kernel_rules/CMakeLists.txt
 - set, 693
- methods/kmeans/CMakeLists.txt
 - set, 693
- methods/lars/CMakeLists.txt
 - set, 694
- methods/linear_regression/CMakeLists.txt
 - set, 694
- methods/local_coordinate_coding/CMakeLists.txt
 - set, 694
- methods/logistic_regression/CMakeLists.txt
 - set, 695
- methods/lsh/CMakeLists.txt
 - set, 695
- methods/mvu/CMakeLists.txt
 - set, 696
- methods/naive_bayes/CMakeLists.txt
 - set, 696
- methods/nca/CMakeLists.txt
 - set, 696
- methods/neighbor_search/CMakeLists.txt
 - set, 697
- methods/nystroem_method/CMakeLists.txt
 - set, 697
- methods/pca/CMakeLists.txt
 - set, 698
- methods/perceptron/CMakeLists.txt
 - cmake_minimum_required, 698
 - set, 698
- methods/perceptron/initialization_methods/CMakeLists.txt
 - set, 699
- methods/perceptron/learning_policies/CMakeLists.txt
 - set, 699
- methods/quic_svd/CMakeLists.txt
 - set, 699
- methods/radical/CMakeLists.txt
 - set, 700

- methods/range_search/CMakeLists.txt
 - set, 700
- methods/rann/CMakeLists.txt
 - set, 701
- methods/regularized_svd/CMakeLists.txt
 - set, 701
- methods/sparse_autoencoder/CMakeLists.txt
 - set, 701
- methods/sparse_coding/CMakeLists.txt
 - set, 702
- Metric
 - mlpack::bound::BallBound, 167
 - mlpack::bound::HRectBound, 174
 - mlpack::fastmks::FastMKS, 251
 - mlpack::kmeans::KMeans, 337, 338
 - mlpack::tree::BinarySpaceTree, 579
 - mlpack::tree::CoverTree, 614
 - mlpack::tree::ExampleTree, 635
- metric
 - mlpack::bound::BallBound, 170
 - mlpack::emst::DTBRules, 230
 - mlpack::emst::DualTreeBoruvka, 240
 - mlpack::fastmks::FastMKS, 253
 - mlpack::kmeans::KMeans, 339
 - mlpack::nca::NCA, 379
 - mlpack::nca::SoftmaxErrorFunction, 384
 - mlpack::neighbor::LSHSearch, 394
 - mlpack::neighbor::NeighborSearch, 403
 - mlpack::neighbor::NeighborSearchRules, 411
 - mlpack::neighbor::RASearchRules, 428
 - mlpack::range::RangeSearch, 516
 - mlpack::range::RangeSearchRules, 522
 - mlpack::tree::CoverTree, 620
 - mlpack::tree::ExampleTree, 637
 - RASearch, 671
- MetricType
 - mlpack::bound::BallBound, 165
 - mlpack::bound::HRectBound, 173
- Mid
 - mlpack::math::Range, 362
- min
 - mlpack::amf::SVDBatchLearning, 151
- MinDistance
 - mlpack::bound::BallBound, 168
 - mlpack::bound::HRectBound, 175
 - mlpack::tree::BinarySpaceTree, 579
 - mlpack::tree::CoverTree, 614
 - mlpack::tree::ExampleTree, 635
- MinGradientNorm
 - mlpack::optimization::L_BFGS, 461
- minGradientNorm
 - mlpack::optimization::L_BFGS, 464
- MinNeighborDistance
 - mlpack::emst::DTBStat, 234, 235
- minNeighborDistance
 - mlpack::emst::DTBStat, 235
- MinPointIterate
 - mlpack::optimization::L_BFGS, 461
- minPointIterate
 - mlpack::optimization::L_BFGS, 464
- MinResidue
 - mlpack::amf::SimpleResidueTermination, 141
- minResidue
 - mlpack::amf::SimpleResidueTermination, 142
- MinStep
 - mlpack::optimization::L_BFGS, 461
- minStep
 - mlpack::optimization::L_BFGS, 464
- MinVals
 - mlpack::det::DTree, 209
- minVals
 - mlpack::det::DTree, 213
- MinWidth
 - mlpack::bound::BallBound, 168
 - mlpack::bound::HRectBound, 175
- minWidth
 - mlpack::bound::HRectBound, 177
- MinimumBoundDistance
 - mlpack::tree::BinarySpaceTree, 580
 - mlpack::tree::CoverTree, 614
- minimumBoundDistance
 - mlpack::tree::BinarySpaceTree, 585
- MinimumSamplesReqd
 - mlpack::neighbor::RASearchRules, 422
- mlpack, 89
- mlpack::CLI, 184
 - ~CLI, 188
 - Add, 190
 - AddAlias, 190
 - AddFlag, 190
 - AliasReverseLookup, 192
 - aliasValues, 195
 - amap_t, 188
 - CLI, 188, 190
 - DefaultMessages, 192
 - desc, 195
 - Destroy, 192
 - didParse, 195
 - doc, 195
 - GetDescription, 192
 - GetParam, 192
 - GetSingleton, 193
 - globalValues, 195
 - gmap_t, 188
 - HasParam, 193
 - HyphenateString, 193
 - ParseCommandLine, 193
 - ParseStream, 193

- Print, 194
- PrintHelp, 194
- programName, 195
- RegisterProgramDoc, 194
- RemoveDuplicateFlags, 194
- RequiredOptions, 194
- requiredOptions, 195
- SanitizeString, 194
- singleton, 195
- Timer, 195
- timer, 196
- UpdateGmap, 194
- vmap, 196
- mlpack::Log, 357
 - Assert, 358
 - cout, 359
 - Debug, 359
 - Fatal, 359
 - Info, 359
 - Warn, 359
- mlpack::ParamData, 497
 - desc, 497
 - isFlag, 497
 - name, 497
 - tname, 498
 - value, 498
 - wasPassed, 498
- mlpack::Timer, 562
 - Get, 563
 - Start, 563
 - Stop, 563
- mlpack::Timers, 564
 - FileTimeToTimeVal, 564
 - GetAllTimers, 564
 - GetTime, 564
 - GetTimer, 565
 - PrintTimer, 566
 - StartTimer, 566
 - StopTimer, 566
 - Timers, 564
 - timers, 566
- mlpack::amf, 90
 - SVDBatchLearning::HUpdate< arma::sp_mat >, 92
 - SVDBatchLearning::WUpdate< arma::sp_mat >, 92
 - SVDIncompleteIncrementalLearning::HUpdate< arma::sp_mat >, 92
 - SVDIncompleteIncrementalLearning::WUpdate< arma::sp_mat >, 92
- mlpack::amf::AMF
 - AMF, 124
 - Apply, 125
 - initializationRule, 126
 - InitializeRule, 125
 - TerminationPolicy, 125, 126
 - terminationPolicy, 126
 - Update, 126
 - update, 127
- mlpack::amf::AMF< TerminationPolicyType, Initialization← RuleType, UpdateRuleType >, 123
- mlpack::amf::AverageInitialization, 127
 - AverageInitialization, 127
 - Initialize, 127
- mlpack::amf::CompleteIncrementalTermination
 - CompleteIncrementalTermination, 128
 - incrementalIndex, 130
 - Index, 129
 - Initialize, 129
 - IsConverged, 129
 - Iteration, 129
 - iteration, 130
 - MaxIterations, 129
 - t_policy, 130
- mlpack::amf::CompleteIncrementalTermination< Termination← Policy >, 128
- mlpack::amf::IncompleteIncrementalTermination
 - IncompleteIncrementalTermination, 131
 - incrementalIndex, 132
 - Index, 131
 - Initialize, 131
 - IsConverged, 131
 - Iteration, 131
 - iteration, 132
 - MaxIterations, 132
 - t_policy, 132
- mlpack::amf::IncompleteIncrementalTermination< Termination← Policy >, 130
- mlpack::amf::NMFALSUpdate, 132
 - HUpdate, 133
 - Initialize, 134
 - NMFALSUpdate, 133
 - WUpdate, 134
- mlpack::amf::NMFMultiplicativeDistanceUpdate, 134
 - HUpdate, 135
 - Initialize, 135
 - NMFMultiplicativeDistanceUpdate, 135
 - WUpdate, 135
- mlpack::amf::NMFMultiplicativeDivergenceUpdate, 136
 - HUpdate, 137
 - Initialize, 137
 - NMFMultiplicativeDivergenceUpdate, 136
 - WUpdate, 137
- mlpack::amf::RandomAcolInitialization
 - Initialize, 138
 - RandomAcolInitialization, 138
- mlpack::amf::RandomAcolInitialization< p >, 137
- mlpack::amf::RandomInitialization, 138
 - Initialize, 139
 - RandomInitialization, 139

- mlpack::amf::SVDBatchLearning, 148
 - HUpdate, 149
 - Initialize, 150
 - kh, 150
 - kw, 150
 - mH, 150
 - mW, 151
 - max, 150
 - min, 151
 - momentum, 151
 - SVDBatchLearning, 149
 - u, 151
 - WUpdate, 150
- mlpack::amf::SVDCompleteIncrementalLearning
 - currentItemIndex, 153
 - currentUserIndex, 153
 - HUpdate, 152
 - Initialize, 152
 - kh, 153
 - kw, 153
 - m, 154
 - n, 154
 - SVDCompleteIncrementalLearning, 152
 - u, 154
 - WUpdate, 152
- mlpack::amf::SVDCompleteIncrementalLearning< arma←
::sp_mat >, 154
 - ~SVDCompleteIncrementalLearning, 155
 - dummy, 156
 - HUpdate, 155
 - Initialize, 155
 - isStart, 156
 - it, 156
 - kh, 156
 - kw, 156
 - m, 156
 - n, 157
 - SVDCompleteIncrementalLearning, 155
 - u, 157
 - WUpdate, 156
- mlpack::amf::SVDCompleteIncrementalLearning< Mat←
Type >, 151
- mlpack::amf::SVDIncompleteIncrementalLearning, 157
 - currentUserIndex, 158
 - HUpdate, 158
 - Initialize, 158
 - kh, 158
 - kw, 159
 - m, 159
 - n, 159
 - SVDIncompleteIncrementalLearning, 157
 - u, 159
 - WUpdate, 158
- mlpack::amf::SimpleResidueTermination, 139
 - Index, 140
 - Initialize, 140
 - IsConverged, 141
 - Iteration, 141
 - iteration, 142
 - MaxIterations, 141
 - maxIterations, 142
 - MinResidue, 141
 - minResidue, 142
 - nm, 142
 - normOld, 142
 - residue, 142
 - SimpleResidueTermination, 140
- mlpack::amf::SimpleToleranceTermination
 - c_index, 146
 - c_indexOld, 146
 - H, 146
 - Index, 144
 - Initialize, 144
 - IsConverged, 145
 - isCopy, 146
 - Iteration, 145
 - iteration, 146
 - MaxIterations, 145
 - maxIterations, 147
 - normOld, 147
 - residue, 147
 - residueOld, 147
 - reverseStepCount, 147
 - reverseStepTolerance, 147
 - SimpleToleranceTermination, 144
 - Tolerance, 145, 146
 - tolerance, 148
 - V, 148
 - W, 148
- mlpack::amf::SimpleToleranceTermination< MatType >, 143
- mlpack::amf::ValidationRMSETermination
 - c_index, 161
 - c_indexOld, 161
 - H, 161
 - Index, 160
 - Initialize, 160
 - IsConverged, 160
 - isCopy, 161
 - Iteration, 161
 - iteration, 162
 - MaxIterations, 161
 - maxIterations, 162
 - num_test_points, 162
 - reverseStepCount, 162
 - reverseStepTolerance, 162
 - rmse, 162
 - rmseOld, 163

- test_points, 163
- tolerance, 163
- ValidationRMSETermination, 160
- W, 163
- mlpack::amf::ValidationRMSETermination< MatType >, 159
- mlpack::bound, 92
- mlpack::bound::BallBound
 - ~BallBound, 166
 - BallBound, 166
 - Center, 166
 - center, 169
 - Centroid, 167
 - Contains, 167
 - Diameter, 167
 - Dim, 167
 - MaxDistance, 167
 - Metric, 167
 - metric, 170
 - MetricType, 165
 - MinDistance, 168
 - MinWidth, 168
 - operator=, 168
 - operator[], 168
 - operator|, 168
 - ownsMetric, 170
 - Radius, 169
 - radius, 170
 - RangeDistance, 169
 - ToString, 169
 - Vec, 165
- mlpack::bound::BallBound< VecType, TMetricType >, 163
- mlpack::bound::HRectBound
 - ~HRectBound, 173
 - bounds, 177
 - Centroid, 173
 - Clear, 174
 - Contains, 174
 - Diameter, 174
 - Dim, 174
 - dim, 177
 - HRectBound, 173
 - MaxDistance, 174
 - Metric, 174
 - MetricType, 173
 - MinDistance, 175
 - MinWidth, 175
 - minWidth, 177
 - operator=, 175
 - operator[], 175, 176
 - operator|, 176
 - RangeDistance, 176
 - ToString, 176
- mlpack::bound::HRectBound< Power, TakeRoot >, 170
- mlpack::cf, 92
- mlpack::cf::CF
 - CF, 179
 - CleanData, 179
 - CleanedData, 179
 - cleanedData, 182
 - Data, 180
 - data, 182
 - Factorizer, 180
 - factorizer, 182
 - GetRecommendations, 180
 - H, 181
 - h, 183
 - InsertNeighbor, 181
 - NumUsersForSimilarity, 181
 - numUsersForSimilarity, 183
 - Rank, 181
 - rank, 183
 - Rating, 182
 - rating, 183
 - ToString, 182
 - W, 182
 - w, 183
- mlpack::cf::CF< FactorizerType >, 177
- mlpack::data, 93
 - Load, 93
 - NormalizeLabels, 94
 - RevertLabels, 94
 - Save, 94
- mlpack::decision_stump, 95
- mlpack::decision_stump::DecisionStump
 - BinLabels, 198
 - binLabels, 201
 - bucketSize, 201
 - CalculateEntropy, 199
 - Classify, 199
 - CountMostFreq, 199
 - DecisionStump, 198
 - IsDistinct, 199
 - MergeRanges, 199
 - numClass, 201
 - SetupSplitAttribute, 199
 - Split, 200
 - split, 201
 - SplitAttribute, 200
 - splitAttribute, 201
 - Train, 200
 - TrainOnAtt, 201
- mlpack::decision_stump::DecisionStump< MatType >, 196
- mlpack::det, 95
 - PrintLeafMembership, 96
 - PrintVariableImportance, 96
 - Trainer, 96

- mlpack::det::DTree, 202
 - ~DTree, 207
 - AlphaUpper, 207
 - alphaUpper, 212
 - bucketTag, 212
 - ComputeValue, 207
 - ComputeVariableImportance, 207
 - DTree, 205, 207
 - End, 208
 - end, 212
 - FindBucket, 208
 - FindSplit, 208
 - Grow, 208
 - Left, 208
 - left, 212
 - LogNegError, 209
 - logNegError, 212
 - LogNegativeError, 208
 - LogVolume, 209
 - logVolume, 212
 - MaxVals, 209
 - maxVals, 212
 - MinVals, 209
 - minVals, 213
 - PruneAndUpdate, 209
 - Ratio, 210
 - ratio, 213
 - Right, 210
 - right, 213
 - Root, 210
 - root, 213
 - SplitData, 210
 - SplitDim, 210
 - splitDim, 213
 - SplitValue, 210
 - splitValue, 213
 - Start, 211
 - start, 213
 - SubtreeLeaves, 211
 - subtreeLeaves, 214
 - SubtreeLeavesLogNegError, 211
 - subtreeLeavesLogNegError, 214
 - TagTree, 211
 - ToString, 211
 - WithinRange, 211
 - WriteTree, 211
- mlpack::distribution, 97
- mlpack::distribution::DiscreteDistribution, 214
 - Dimensionality, 217
 - DiscreteDistribution, 215
 - Estimate, 217
 - Probabilities, 217
 - probabilities, 218
 - Probability, 217
 - Random, 218
 - ToString, 218
- mlpack::distribution::GaussianDistribution, 218
 - Covariance, 220
 - covariance, 221
 - Dimensionality, 220
 - Estimate, 220
 - GaussianDistribution, 219
 - Mean, 220
 - mean, 221
 - Probability, 221
 - Random, 221
 - ToString, 221
- mlpack::distribution::LaplaceDistribution, 221
 - Dimensionality, 223
 - Estimate, 223, 224
 - LaplaceDistribution, 223
 - Mean, 224
 - mean, 225
 - Probability, 224
 - Random, 224
 - Scale, 224
 - scale, 225
 - ToString, 225
- mlpack::emst, 97
- mlpack::emst::DTBRules
 - BaseCase, 227
 - BaseCases, 227
 - baseCases, 230
 - CalculateBound, 228
 - connections, 230
 - DTBRules, 227
 - dataSet, 230
 - metric, 230
 - neighborsDistances, 230
 - neighborsInComponent, 231
 - neighborsOutComponent, 231
 - Rescore, 228
 - Score, 228, 229
 - Scores, 229
 - scores, 231
 - TraversalInfo, 230
 - traversalInfo, 231
 - TraversalInfoType, 227
- mlpack::emst::DTBRules< MetricType, TreeType >, 225
- mlpack::emst::DTBStat, 231
 - Bound, 234
 - bound, 235
 - ComponentMembership, 234
 - componentMembership, 235
 - DTBStat, 232
 - MaxNeighborDistance, 234
 - maxNeighborDistance, 235
 - MinNeighborDistance, 234, 235

- minNeighborDistance, 235
- mlpack::emst::DualTreeBoruvka
 - ~DualTreeBoruvka, 239
 - AddAllEdges, 239
 - AddEdge, 239
 - Cleanup, 239
 - CleanupHelper, 239
 - ComputeMST, 239
 - connections, 240
 - data, 240
 - dataCopy, 240
 - DualTreeBoruvka, 238
 - edges, 240
 - EmitResults, 240
 - metric, 240
 - naive, 241
 - neighborsDistances, 241
 - neighborsInComponent, 241
 - neighborsOutComponent, 241
 - oldFromNew, 241
 - ownTree, 241
 - SortFun, 242
 - ToString, 240
 - totalDist, 242
 - tree, 242
- mlpack::emst::DualTreeBoruvka< MetricType, TreeType >, 236
- mlpack::emst::DualTreeBoruvka< MetricType, TreeType >::SortEdgesHelper, 242
- mlpack::emst::DualTreeBoruvka::SortEdgesHelper
 - operator(), 242
- mlpack::emst::EdgePair, 243
 - Distance, 244
 - distance, 245
 - EdgePair, 244
 - Greater, 244
 - greater, 245
 - Lesser, 244
 - lesser, 245
- mlpack::emst::UnionFind, 245
 - ~UnionFind, 246
 - Find, 246
 - parent, 247
 - rank, 247
 - Union, 246
 - UnionFind, 246
- mlpack::fastmks, 98
- mlpack::fastmks::FastMKS
 - ~FastMKS, 251
 - FastMKS, 249, 250
 - InsertNeighbor, 251
 - Metric, 251
 - metric, 253
 - naive, 253
 - querySet, 253
 - queryTree, 253
 - referenceSet, 253
 - referenceTree, 254
 - Search, 251
 - single, 254
 - ToString, 253
 - treeOwner, 254
- mlpack::fastmks::FastMKS< KernelType, TreeType >, 247
- mlpack::fastmks::FastMKSRules
 - BaseCase, 256
 - BaseCases, 256
 - baseCases, 258
 - CalculateBound, 256
 - FastMKSRules, 256
 - indices, 259
 - InsertNeighbor, 257
 - kernel, 259
 - lastKernel, 259
 - lastQueryIndex, 259
 - lastReferenceIndex, 259
 - products, 259
 - queryKernels, 259
 - querySet, 260
 - referenceKernels, 260
 - referenceSet, 260
 - Rescore, 257
 - Score, 257, 258
 - Scores, 258
 - scores, 260
 - TraversalInfo, 258
 - traversalInfo, 260
 - TraversalInfoType, 256
- mlpack::fastmks::FastMKSRules< KernelType, TreeType >, 254
- mlpack::fastmks::FastMKSSStat, 260
 - Bound, 262
 - bound, 263
 - FastMKSSStat, 261
 - LastKernel, 262
 - lastKernel, 263
 - LastKernelNode, 262
 - lastKernelNode, 263
 - SelfKernel, 263
 - selfKernel, 263
- mlpack::gmm, 98
 - phi, 99, 100
- mlpack::gmm::DiagonalConstraint, 264
 - ApplyConstraint, 264
- mlpack::gmm::EMFit
 - Clusterer, 268
 - clusterer, 270
 - Constraint, 268
 - constraint, 271

- EMFit, 267
- Estimate, 268, 269
- InitialClustering, 269
- LogLikelihood, 269
- MaxIterations, 270
- maxIterations, 271
- Tolerance, 270
- tolerance, 271
- mlpack::gmm::EMFit< InitialClusteringType, Covariance↔
ConstraintPolicy >, 266
- mlpack::gmm::EigenvalueRatioConstraint, 264
 - ApplyConstraint, 265
 - EigenvalueRatioConstraint, 265
 - ratios, 265
- mlpack::gmm::GMM
 - Classify, 276
 - Covariances, 276
 - covariances, 282
 - Dimensionality, 276, 277
 - dimensionality, 282
 - Estimate, 277
 - Fitter, 279
 - fitter, 282
 - GMM, 274–276
 - Gaussians, 279
 - gaussians, 282
 - Load, 279
 - localFitter, 282
 - LogLikelihood, 280
 - Means, 280
 - means, 282
 - operator=, 280
 - Probability, 280, 281
 - Random, 281
 - Save, 281
 - ToString, 281
 - Weights, 281
 - weights, 282
- mlpack::gmm::GMM< FittingType >, 271
- mlpack::gmm::NoConstraint, 283
 - ApplyConstraint, 283
- mlpack::gmm::PositiveDefiniteConstraint, 283
 - ApplyConstraint, 284
- mlpack::hmm, 100
 - LoadHMM, 101
 - SaveHMM, 101
- mlpack::hmm::HMM
 - Backward, 287
 - Dimensionality, 288
 - dimensionality, 292
 - Emission, 288
 - emission, 292
 - Estimate, 288, 289
 - Forward, 289
 - Generate, 289
 - HMM, 286, 287
 - Initial, 290
 - initial, 292
 - LogLikelihood, 290
 - Predict, 290
 - ToString, 291
 - Tolerance, 290, 291
 - tolerance, 293
 - Train, 291
 - Transition, 292
 - transition, 293
- mlpack::hmm::HMM< Distribution >, 284
- mlpack::kernel, 101
- mlpack::kernel::CosineDistance, 293
 - Evaluate, 294
 - ToString, 294
- mlpack::kernel::EpanechnikovKernel, 294
 - bandwidth, 296
 - ConvolutionIntegral, 295
 - EpanechnikovKernel, 295
 - Evaluate, 295, 296
 - inverseBandwidthSquared, 296
 - Normalizer, 296
 - ToString, 296
- mlpack::kernel::ExampleKernel, 296
 - ConvolutionIntegral, 298
 - Evaluate, 298
 - ExampleKernel, 297
 - Normalizer, 298
 - ToString, 299
- mlpack::kernel::GaussianKernel, 299
 - Bandwidth, 300, 301
 - bandwidth, 303
 - ConvolutionIntegral, 301
 - Evaluate, 301
 - Gamma, 303
 - gamma, 303
 - GaussianKernel, 300
 - Normalizer, 303
 - ToString, 303
- mlpack::kernel::HyperbolicTangentKernel, 304
 - Evaluate, 305
 - HyperbolicTangentKernel, 305
 - Offset, 305
 - offset, 306
 - Scale, 306
 - scale, 306
 - ToString, 306
- mlpack::kernel::KMeansSelection
 - Select, 311
- mlpack::kernel::KMeansSelection< ClusteringType >, 311
- mlpack::kernel::KernelTraits

- IsNormalized, 307
- mlpack::kernel::KernelTraits< CosineDistance >, 307
 - IsNormalized, 307
- mlpack::kernel::KernelTraits< EpanechnikovKernel >, 308
 - IsNormalized, 308
- mlpack::kernel::KernelTraits< GaussianKernel >, 308
 - IsNormalized, 309
- mlpack::kernel::KernelTraits< KernelType >, 306
- mlpack::kernel::KernelTraits< LaplacianKernel >, 309
 - IsNormalized, 309
- mlpack::kernel::KernelTraits< SphericalKernel >, 310
 - IsNormalized, 310
- mlpack::kernel::KernelTraits< TriangularKernel >, 310
 - IsNormalized, 311
- mlpack::kernel::LaplacianKernel, 312
 - Bandwidth, 314
 - bandwidth, 315
 - Evaluate, 314
 - LaplacianKernel, 312
 - ToString, 315
- mlpack::kernel::LinearKernel, 315
 - Evaluate, 316
 - LinearKernel, 316
 - ToString, 316
- mlpack::kernel::NystroemMethod
 - Apply, 318
 - data, 318
 - GetKernelMatrix, 318
 - kernel, 318
 - NystroemMethod, 317
 - rank, 319
- mlpack::kernel::NystroemMethod< KernelType, Point← SelectionPolicy >, 317
- mlpack::kernel::OrderedSelection, 319
 - Select, 319
- mlpack::kernel::PSpectrumStringKernel, 322
 - Counts, 324
 - counts, 325
 - datasets, 325
 - Evaluate, 324
 - P, 324
 - p, 325
 - PSpectrumStringKernel, 323
 - ToString, 324
- mlpack::kernel::PolynomialKernel, 319
 - Degree, 321
 - degree, 322
 - Evaluate, 321
 - Offset, 321
 - offset, 322
 - PolynomialKernel, 320
 - ToString, 322
- mlpack::kernel::RandomSelection, 325
 - Select, 326
- mlpack::kernel::SphericalKernel, 327
 - bandwidth, 329
 - bandwidthSquared, 329
 - ConvolutionIntegral, 328
 - Evaluate, 328
 - Normalizer, 328
 - SphericalKernel, 327
 - ToString, 328
- mlpack::kernel::TriangularKernel, 329
 - Bandwidth, 330
 - bandwidth, 332
 - Evaluate, 330
 - ToString, 332
 - TriangularKernel, 330
- mlpack::kmeans, 103
- mlpack::kmeans::AllowEmptyClusters, 332
 - AllowEmptyClusters, 333
 - EmptyCluster, 333
- mlpack::kmeans::KMeans
 - Cluster, 336
 - EmptyClusterAction, 337
 - emptyClusterAction, 339
 - KMeans, 335
 - MaxIterations, 337
 - maxIterations, 339
 - Metric, 337, 338
 - metric, 339
 - OverclusteringFactor, 338
 - overclusteringFactor, 339
 - Partitioner, 338
 - partitioner, 339
 - ToString, 339
- mlpack::kmeans::KMeans< MetricType, InitialPartition← Policy, EmptyClusterPolicy >, 333
- mlpack::kmeans::MaxVarianceNewCluster, 340
 - EmptyCluster, 340
 - MaxVarianceNewCluster, 340
- mlpack::kmeans::RandomPartition, 342
 - Cluster, 343
 - RandomPartition, 342
- mlpack::kmeans::RefinedStart, 343
 - Cluster, 344
 - Percentage, 344
 - percentage, 345
 - RefinedStart, 344
 - Samplings, 345
 - samplings, 345
- mlpack::kpca, 103
- mlpack::kpca::KernelPCA
 - Apply, 348
 - CenterTransformedData, 349
 - centerTransformedData, 349
 - Kernel, 349

- kernel, 350
- KernelPCA, 346
- ToString, 349
- mlpack::kpca::KernelPCA< KernelType, KernelRule >, 345
- mlpack::kpca::NaiveKernelRule
 - ApplyKernelMatrix, 350
- mlpack::kpca::NaiveKernelRule< KernelType >, 350
- mlpack::kpca::NystroemKernelRule
 - ApplyKernelMatrix, 351
- mlpack::kpca::NystroemKernelRule< KernelType, Point↔ SelectionPolicy >, 351
- mlpack::lcc, 104
- mlpack::lcc::LocalCoordinateCoding
 - atoms, 356
 - Codes, 354
 - codes, 356
 - Data, 354
 - data, 356
 - Dictionary, 354
 - dictionary, 357
 - Encode, 354
 - lambda, 357
 - LocalCoordinateCoding, 353
 - Objective, 356
 - OptimizeCode, 356
 - OptimizeDictionary, 356
 - ToString, 356
- mlpack::lcc::LocalCoordinateCoding< DictionaryInitializer >, 352
- mlpack::math, 104
 - Center, 105
 - ClampNonNegative, 105
 - ClampNonPositive, 105
 - ClampRange, 106
 - Orthogonalize, 106
 - randGen, 108
 - RandInt, 106
 - RandNormal, 107
 - randNormalDist, 108
 - randUniformDist, 108
 - RandVector, 108
 - Random, 107
 - RandomSeed, 107
 - RemoveRows, 108
 - VectorPower, 108
 - WhitenUsingEig, 108
 - WhitenUsingSVD, 108
- mlpack::math::Range, 359
 - Contains, 361
 - Hi, 362
 - hi, 364
 - Lo, 362
 - lo, 364
 - Mid, 362
 - operator!=, 362
 - operator<, 363
 - operator>, 363
 - operator*, 363, 364
 - operator*=: 363
 - operator==, 363
 - operator&, 362
 - operator&=: 363
 - operator|, 364
 - operator|=, 364
 - Range, 361
 - ToString, 364
 - Width, 364
- mlpack::metric, 109
 - ChebyshevDistance, 109
 - EuclideanDistance, 109
 - ManhattanDistance, 109
 - SquaredEuclideanDistance, 109
- mlpack::metric::IPMetric
 - ~IPMetric, 366
 - Evaluate, 366
 - IPMetric, 365
 - Kernel, 366
 - kernel, 366
 - localKernel, 366
 - ToString, 366
- mlpack::metric::IPMetric< KernelType >, 365
- mlpack::metric::LMetric
 - Evaluate, 368
 - LMetric, 368
 - ToString, 368
- mlpack::metric::LMetric< Power, TakeRoot >, 367
- mlpack::metric::MahalanobisDistance
 - Covariance, 371
 - covariance, 372
 - Evaluate, 371
 - MahalanobisDistance, 369, 371
 - ToString, 371
- mlpack::metric::MahalanobisDistance< TakeRoot >, 368
- mlpack::mvu, 109
- mlpack::mvu::MVU, 372
 - data, 373
 - MVU, 372
 - Unfold, 373
- mlpack::naive_bayes, 109
- mlpack::naive_bayes::NaiveBayesClassifier
 - Classify, 374
 - Means, 375
 - means, 376
 - NaiveBayesClassifier, 374
 - Probabilities, 375
 - probabilities, 376
 - Variances, 375

- variances, 376
- mlpack::naive_bayes::NaiveBayesClassifier< MatType >, 373
- mlpack::nca, 110
- mlpack::nca::NCA
 - Dataset, 378
 - dataset, 379
 - errorFunction, 379
 - Labels, 378
 - labels, 379
 - LearnDistance, 378
 - metric, 379
 - NCA, 377
 - Optimizer, 379
 - optimizer, 380
 - ToString, 379
- mlpack::nca::NCA< MetricType, OptimizerType >, 376
- mlpack::nca::SoftmaxErrorFunction
 - dataset, 383
 - denominators, 383
 - Evaluate, 382
 - GetInitialPoint, 382
 - Gradient, 382, 383
 - labels, 384
 - lastCoordinates, 384
 - metric, 384
 - NumFunctions, 383
 - p, 384
 - Precalculate, 383
 - precalculated, 384
 - SoftmaxErrorFunction, 381
 - stretchedDataset, 384
 - ToString, 383
- mlpack::nca::SoftmaxErrorFunction< MetricType >, 380
- mlpack::neighbor, 110
 - AllkFN, 111
 - AllkNN, 111
 - AllkRAFN, 112
 - AllkRANN, 112
 - Unmap, 112, 113
- mlpack::neighbor::FurthestNeighborSort, 385
 - BestDistance, 386
 - BestNodeToNodeDistance, 386
 - BestPointToNodeDistance, 386
 - CombineBest, 387
 - CombineWorst, 387
 - IsBetter, 387
 - SortDistance, 387
 - WorstDistance, 388
- mlpack::neighbor::LSHSearch
 - BaseCase, 390
 - bucketContentSize, 393
 - bucketRowInHashTable, 393
 - bucketSize, 393
 - BuildHash, 392
 - distancePtr, 393
 - hashWidth, 393
 - InsertNeighbor, 392
 - LSHSearch, 390
 - metric, 394
 - neighborPtr, 394
 - numProj, 394
 - numTables, 394
 - offsets, 394
 - projections, 394
 - querySet, 394
 - referenceSet, 394
 - ReturnIndicesFromTable, 392
 - Search, 392
 - secondHashSize, 395
 - secondHash Table, 395
 - secondHashWeights, 395
 - ToString, 393
- mlpack::neighbor::LSHSearch< SortPolicy >, 388
- mlpack::neighbor::NearestNeighborSort, 395
 - BestDistance, 396
 - BestNodeToNodeDistance, 396, 397
 - BestPointToNodeDistance, 397
 - CombineBest, 397
 - CombineWorst, 398
 - IsBetter, 398
 - SortDistance, 398
 - WorstDistance, 398
- mlpack::neighbor::NeighborSearch
 - ~NeighborSearch, 402
 - hasQuerySet, 403
 - metric, 403
 - naive, 403
 - NeighborSearch, 400–402
 - oldFromNewQueries, 403
 - oldFromNewReferences, 404
 - queryCopy, 404
 - querySet, 404
 - queryTree, 404
 - referenceCopy, 404
 - referenceSet, 404
 - referenceTree, 405
 - Search, 403
 - singleMode, 405
 - ToString, 403
 - treeOwner, 405
- mlpack::neighbor::NeighborSearch< SortPolicy, MetricType, TreeType >, 399
- mlpack::neighbor::NeighborSearchRules
 - BaseCase, 407
 - BaseCases, 407, 408
 - baseCases, 410
 - CalculateBound, 408

- distances, 410
- InsertNeighbor, 408
- lastBaseCase, 410
- lastQueryIndex, 411
- lastReferenceIndex, 411
- metric, 411
- NeighborSearchRules, 407
- neighbors, 411
- querySet, 411
- referenceSet, 411
- Rescore, 408, 409
- Score, 409
- Scores, 409, 410
- scores, 411
- TraversallInfo, 410
- traversallInfo, 412
- TraversallInfoType, 407
- mlpack::neighbor::NeighborSearchRules< SortPolicy, MetricType, TreeType >, 405
- mlpack::neighbor::NeighborSearchStat
 - Bound, 413, 414
 - bound, 415
 - FirstBound, 414
 - firstBound, 415
 - LastDistance, 414
 - lastDistance, 415
 - LastDistanceNode, 414, 415
 - lastDistanceNode, 416
 - NeighborSearchStat, 413
 - SecondBound, 415
 - secondBound, 416
- mlpack::neighbor::NeighborSearchStat< SortPolicy >, 412
- mlpack::neighbor::NeighborSearchTraversallInfo
 - LastBaseCase, 417
 - lastBaseCase, 419
 - LastQueryNode, 418
 - lastQueryNode, 419
 - LastReferenceNode, 418
 - lastReferenceNode, 419
 - LastScore, 418
 - lastScore, 419
 - NeighborSearchTraversallInfo, 417
- mlpack::neighbor::NeighborSearchTraversallInfo< TreeType >, 416
- mlpack::neighbor::RASearchRules
 - BaseCase, 422
 - distances, 428
 - firstLeafExact, 428
 - InsertNeighbor, 422
 - metric, 428
 - MinimumSamplesReqd, 422
 - neighbors, 428
 - NumDistComputations, 422
 - numDistComputations, 429
 - NumEffectiveSamples, 422
 - numSamplesMade, 429
 - numSamplesReqd, 429
 - ObtainDistinctSamples, 423
 - querySet, 429
 - RASearch< SortPolicy, MetricType, TreeType >, 428
 - RASearchRules, 422
 - referenceSet, 429
 - Rescore, 423
 - sampleAtLeaves, 429
 - samplingRatio, 429
 - Score, 425, 426
 - singleSampleLimit, 430
 - SuccessProbability, 426
 - TraversallInfo, 428
 - traversallInfo, 430
 - TraversallInfoType, 421
- mlpack::neighbor::RASearchRules< SortPolicy, MetricType, TreeType >, 419
- mlpack::nn, 113
- mlpack::nn::SparseAutoencoder
 - Beta, 432, 433
 - beta, 435
 - GetNewFeatures, 433
 - HiddenSize, 433
 - hiddenSize, 435
 - Lambda, 433
 - lambda, 435
 - parameters, 435
 - Rho, 434
 - rho, 435
 - Sigmoid, 434
 - SparseAutoencoder, 432
 - VisibleSize, 434
 - visibleSize, 435
- mlpack::nn::SparseAutoencoder< OptimizerType >, 430
- mlpack::nn::SparseAutoencoderFunction, 436
 - Beta, 437
 - beta, 440
 - data, 440
 - Evaluate, 437
 - GetInitialPoint, 438
 - Gradient, 438
 - HiddenSize, 438
 - hiddenSize, 440
 - initialPoint, 440
 - InitializeWeights, 438
 - Lambda, 438, 439
 - lambda, 440
 - Rho, 439
 - rho, 440
 - Sigmoid, 439
 - SparseAutoencoderFunction, 437

- VisibleSize, 439
- visibleSize, 440
- mlpack::optimization, 113
- mlpack::optimization::AugLagrangian
 - AugLagrangian, 443
 - augfunc, 445
 - Function, 443
 - function, 445
 - L_BFGSType, 443
 - LBFGS, 444
 - Lambda, 444
 - lbfgs, 445
 - lbfgsInternal, 445
 - Optimize, 444
 - Sigma, 445
 - ToString, 445
- mlpack::optimization::AugLagrangian< Lagrangian↔
Function >, 441
- mlpack::optimization::AugLagrangianFunction
 - AugLagrangianFunction, 447, 448
 - Evaluate, 448
 - Function, 448
 - function, 450
 - GetInitialPoint, 448
 - Gradient, 449
 - Lambda, 449
 - lambda, 450
 - Sigma, 449
 - sigma, 450
 - ToString, 449
- mlpack::optimization::AugLagrangianFunction< Lagrangian↔
Function >, 446
- mlpack::optimization::AugLagrangianTestFunction, 450
 - AugLagrangianTestFunction, 451
 - Evaluate, 451
 - EvaluateConstraint, 451
 - GetInitialPoint, 451
 - Gradient, 451
 - GradientConstraint, 451
 - initialPoint, 451
 - NumConstraints, 451
 - ToString, 451
- mlpack::optimization::ExponentialSchedule, 452
 - ExponentialSchedule, 452
 - Lambda, 453
 - lambda, 453
 - NextTemperature, 453
- mlpack::optimization::GockenbachFunction, 453
 - Evaluate, 454
 - EvaluateConstraint, 454
 - GetInitialPoint, 454
 - GockenbachFunction, 454
 - Gradient, 454
 - GradientConstraint, 454
- initialPoint, 455
- NumConstraints, 455
- mlpack::optimization::L_BFGS
 - ArmijoConstant, 458
 - armijoConstant, 463
 - ChooseScalingFactor, 459
 - Evaluate, 459
 - Function, 459
 - function, 463
 - GradientNormTooSmall, 459
 - L_BFGS, 458
 - LineSearch, 459
 - MaxIterations, 460
 - maxIterations, 464
 - MaxLineSearchTrials, 460
 - maxLineSearchTrials, 464
 - MaxStep, 460
 - maxStep, 464
 - MinGradientNorm, 461
 - minGradientNorm, 464
 - MinPointIterate, 461
 - minPointIterate, 464
 - MinStep, 461
 - minStep, 464
 - newIterateTmp, 465
 - NumBasis, 461
 - numBasis, 465
 - Optimize, 462
 - s, 465
 - SearchDirection, 462
 - ToString, 463
 - UpdateBasisSet, 463
 - Wolfe, 463
 - wolfe, 465
 - y, 465
- mlpack::optimization::L_BFGS< FunctionType >, 455
- mlpack::optimization::LRSDP, 468
 - A, 469
 - AModes, 470
 - AugLag, 470
 - augLag, 471
 - B, 470
 - C, 470
 - Function, 471
 - function, 471
 - LRSDP, 469
 - Optimize, 471
 - ToString, 471
- mlpack::optimization::LRSDPFunction, 472
 - A, 473
 - a, 475
 - AModes, 473
 - aModes, 475
 - B, 473, 474

- b, 475
- C, 474
- c, 475
- Evaluate, 474
- EvaluateConstraint, 474
- GetInitialPoint, 474
- Gradient, 474
- GradientConstraint, 474
- initialPoint, 475
- LRSDPFunction, 473
- NumConstraints, 475
- ToString, 475
- mlpack::optimization::LovaszThetaSDP, 466
 - Edges, 467
 - edges, 467
 - Evaluate, 467
 - EvaluateConstraint, 467
 - GetInitialPoint, 467
 - Gradient, 467
 - GradientConstraint, 467
 - initialPoint, 467
 - LovaszThetaSDP, 466
 - NumConstraints, 467
 - vertices, 467
- mlpack::optimization::SA
 - coolingSchedule, 483
 - Function, 479
 - function, 484
 - Gain, 479
 - gain, 484
 - GenerateMove, 479
 - InitMoves, 480
 - initMoves, 484
 - MaxIterations, 480
 - maxIterations, 484
 - MaxMove, 480, 481
 - maxMove, 484
 - MaxToleranceSweep, 481
 - maxToleranceSweep, 484
 - MoveControl, 481
 - MoveCtrlSweep, 482
 - moveCtrlSweep, 484
 - MoveSize, 482
 - moveSize, 485
 - Optimize, 482
 - SA, 478
 - Temperature, 483
 - temperature, 485
 - ToString, 483
 - Tolerance, 483
 - tolerance, 485
- mlpack::optimization::SA< FunctionType, Cooling↔ ScheduleType >, 476
- mlpack::optimization::SGD
 - Function, 488
 - function, 490
 - MaxIterations, 488
 - maxIterations, 490
 - Optimize, 488, 489
 - SGD, 488
 - Shuffle, 489
 - shuffle, 490
 - StepSize, 489
 - stepSize, 490
 - ToString, 490
 - Tolerance, 489, 490
 - tolerance, 490
- mlpack::optimization::SGD< DecomposableFunctionType >, 485
- mlpack::optimization::test, 114
- mlpack::optimization::test::GeneralizedRosenbrock↔ Function, 491
 - Evaluate, 492
 - GeneralizedRosenbrockFunction, 492
 - GetInitialPoint, 492
 - Gradient, 492
 - initialPoint, 492
 - n, 492
 - NumFunctions, 492
- mlpack::optimization::test::RosenbrockFunction, 492
 - Evaluate, 493
 - GetInitialPoint, 493
 - Gradient, 493
 - initialPoint, 493
 - RosenbrockFunction, 493
- mlpack::optimization::test::RosenbrockWoodFunction, 493
 - Evaluate, 494
 - GetInitialPoint, 494
 - Gradient, 494
 - initialPoint, 494
 - rf, 494
 - RosenbrockWoodFunction, 494
 - wf, 494
- mlpack::optimization::test::SGDTestFunction, 495
 - Evaluate, 495
 - GetInitialPoint, 495
 - Gradient, 495
 - NumFunctions, 495
 - SGDTestFunction, 495
- mlpack::optimization::test::WoodFunction, 496
 - Evaluate, 496
 - GetInitialPoint, 496
 - Gradient, 496
 - initialPoint, 496
 - WoodFunction, 496
- mlpack::pca, 114
- mlpack::pca::PCA, 498
 - Apply, 499, 500

- PCA, 499
- ScaleData, 500, 501
- scaleData, 501
- ToString, 501
- mlpack::perceptron, 115
- mlpack::perceptron::Perceptron
 - classLabels, 503
 - Classify, 503
 - iter, 503
 - Perceptron, 502
 - Train, 503
 - trainData, 503
 - weightVectors, 504
- mlpack::perceptron::Perceptron< LearnPolicy, Weight←
InitializationPolicy, MatType >, 501
- mlpack::perceptron::RandomInitialization, 504
 - Initialize, 504
 - RandomInitialization, 504
- mlpack::perceptron::SimpleWeightUpdate, 505
 - UpdateWeights, 505
- mlpack::perceptron::ZeroInitialization, 505
 - Initialize, 506
 - ZeroInitialization, 506
- mlpack::radical, 115
 - WhitenFeatureMajorMatrix, 115
- mlpack::radical::Radical, 506
 - Angles, 509
 - angles, 511
 - candidate, 511
 - CopyAndPerturb, 509
 - DoRadical, 509
 - DoRadical2D, 509
 - m, 511
 - NoiseStdDev, 509, 510
 - noiseStdDev, 511
 - perturbed, 511
 - Radical, 508
 - Replicates, 510
 - replicates, 511
 - Sweeps, 510
 - sweeps, 511
 - ToString, 510
 - Vasicek, 510
- mlpack::range, 115
- mlpack::range::RangeSearch
 - ~RangeSearch, 515
 - hasQuerySet, 516
 - metric, 516
 - naive, 516
 - numPrunes, 516
 - oldFromNewQueries, 516
 - oldFromNewReferences, 517
 - queryCopy, 517
 - querySet, 517
 - queryTree, 517
 - RangeSearch, 513, 514
 - referenceCopy, 517
 - referenceSet, 517
 - referenceTree, 517
 - Search, 515
 - singleMode, 518
 - ToString, 516
 - treeOwner, 518
- mlpack::range::RangeSearch< MetricType, TreeType >, 512
- mlpack::range::RangeSearchRules
 - AddResult, 520
 - BaseCase, 520
 - distances, 522
 - lastQueryIndex, 522
 - lastReferenceIndex, 522
 - metric, 522
 - neighbors, 522
 - querySet, 522
 - range, 523
 - RangeSearchRules, 520
 - referenceSet, 523
 - Rescore, 520, 521
 - Score, 521
 - TraversalInfo, 521, 522
 - traversalInfo, 523
 - TraversalInfoType, 520
- mlpack::range::RangeSearchRules< MetricType, Tree←
Type >, 518
- mlpack::range::RangeSearchStat, 523
 - LastDistance, 524
 - lastDistance, 525
 - LastDistanceNode, 524, 525
 - lastDistanceNode, 525
 - RangeSearchStat, 524
- mlpack::regression, 116
- mlpack::regression::LARS, 525
 - Activate, 528
 - ActiveSet, 528
 - activeSet, 530
 - BetaPath, 528
 - betaPath, 530
 - CholeskyDelete, 529
 - CholeskyInsert, 529
 - ComputeYHatDirection, 529
 - Deactivate, 529
 - elasticNet, 530
 - GivensRotate, 529
 - Ignore, 529
 - ignoreSet, 530
 - InterpolateBeta, 529
 - isActive, 530
 - isIgnored, 530

- LARS, 528
- lambda1, 531
- lambda2, 531
- LambdaPath, 529
- lambdaPath, 531
- lasso, 531
- matGram, 531
- matGramInternal, 531
- MatUtriCholFactor, 529
- matUtriCholFactor, 531
- Regress, 529
- ToString, 530
- tolerance, 531
- useCholesky, 532
- mlpack::regression::LinearRegression, 532
 - ComputeError, 534
 - intercept, 536
 - Lambda, 534
 - lambda, 536
 - LinearRegression, 533
 - Parameters, 534
 - parameters, 536
 - Predict, 534
 - ToString, 536
- mlpack::regression::LogisticRegression
 - ComputeAccuracy, 538
 - ComputeError, 539
 - Lambda, 539
 - lambda, 540
 - LogisticRegression, 537, 538
 - Parameters, 539, 540
 - parameters, 540
 - Predict, 540
 - ToString, 540
- mlpack::regression::LogisticRegression< OptimizerType >, 536
- mlpack::regression::LogisticRegressionFunction, 541
 - Evaluate, 542
 - GetInitialPoint, 543
 - Gradient, 543
 - InitialPoint, 543
 - initialPoint, 544
 - Lambda, 543, 544
 - lambda, 544
 - LogisticRegressionFunction, 542
 - NumFunctions, 544
 - Predictors, 544
 - predictors, 544
 - Responses, 544
 - responses, 545
- mlpack::sparse_coding, 116
- mlpack::sparse_coding::DataDependentRandomInitializer, 545
 - Initialize, 545
- mlpack::sparse_coding::NothingInitializer, 546
 - Initialize, 546
- mlpack::sparse_coding::RandomInitializer, 546
 - Initialize, 547
- mlpack::sparse_coding::SparseCoding
 - atoms, 552
 - Codes, 550
 - codes, 553
 - Data, 550
 - data, 553
 - Dictionary, 550
 - dictionary, 553
 - Encode, 550
 - lambda1, 553
 - lambda2, 553
 - Objective, 552
 - OptimizeCode, 552
 - OptimizeDictionary, 552
 - ProjectDictionary, 552
 - SparseCoding, 549
 - ToString, 552
- mlpack::sparse_coding::SparseCoding< Dictionary< Initializer >, 547
- mlpack::svd, 116
- mlpack::svd::QUIC_SVD, 553
 - basis, 555
 - dataset, 555
 - delta, 555
 - epsilon, 555
 - ExtractSVD, 554
 - QUIC_SVD, 554
- mlpack::svd::RegularizedSVD
 - alpha, 556
 - data, 557
 - iterations, 557
 - lambda, 557
 - optimizer, 557
 - rSVDFunc, 557
 - rank, 557
 - RegularizedSVD, 556
- mlpack::svd::RegularizedSVD< OptimizerType >, 555
- mlpack::svd::RegularizedSVDFunction, 557
 - data, 561
 - Dataset, 559
 - Evaluate, 559
 - GetInitialPoint, 559
 - Gradient, 559
 - initialPoint, 561
 - Lambda, 561
 - lambda, 562
 - NumFunctions, 561
 - NumItems, 561
 - numItems, 562
 - NumUsers, 561

- numUsers, 562
- Rank, 561
- rank, 562
- RegularizedSVDFunction, 559
- mlpack::tree, 116
 - CosineNodeQueue, 117
- mlpack::tree::BinarySpaceTree
 - ~BinarySpaceTree, 574
 - Begin, 574
 - begin, 584
 - BinarySpaceTree, 571–574
 - Bound, 574, 575
 - bound, 584
 - Centroid, 575
 - Child, 575
 - CopyMe, 575
 - Count, 575
 - count, 584
 - Dataset, 576
 - dataset, 584
 - Descendant, 576
 - End, 576
 - ExtendTree, 576
 - FindByBeginCount, 576, 577
 - FurthestDescendantDistance, 577
 - furthestDescendantDistance, 584
 - FurthestPointDistance, 577
 - GetSplitDimension, 577
 - HasSelfChildren, 578
 - IsLeaf, 578
 - Left, 578
 - left, 584
 - Mat, 571
 - MaxDistance, 578
 - MaxLeafSize, 579
 - maxLeafSize, 585
 - Metric, 579
 - MinDistance, 579
 - MinimumBoundDistance, 580
 - minimumBoundDistance, 585
 - NumChildren, 580
 - NumDescendants, 580
 - NumPoints, 580
 - Parent, 580
 - parent, 585
 - ParentDistance, 580, 581
 - parentDistance, 585
 - Point, 581
 - RangeDistance, 581
 - Right, 581, 582
 - right, 585
 - SplitDimension, 582
 - splitDimension, 585
 - SplitNode, 582
 - Stat, 583
 - stat, 586
 - ToString, 583
 - TreeDepth, 583
 - TreeSize, 583
- mlpack::tree::BinarySpaceTree< BoundType, Statistic←Type, MatType, SplitType >, 566
- mlpack::tree::BinarySpaceTree< BoundType, Statistic←Type, MatType, SplitType >::DualTree←Traverser< RuleType >, 586
- mlpack::tree::BinarySpaceTree< BoundType, Statistic←Type, MatType, SplitType >::SingleTree←Traverser< RuleType >, 590
- mlpack::tree::BinarySpaceTree::DualTreeTraverser
 - DualTreeTraverser, 587
 - NumBaseCases, 587
 - numBaseCases, 589
 - NumPrunes, 588
 - numPrunes, 589
 - NumScores, 588
 - numScores, 589
 - NumVisited, 588
 - numVisited, 589
 - rule, 590
 - traversalInfo, 590
 - Traverse, 589
- mlpack::tree::BinarySpaceTree::SingleTreeTraverser
 - NumPrunes, 591
 - numPrunes, 593
 - rule, 593
 - SingleTreeTraverser, 591
 - Traverse, 591
- mlpack::tree::CompareCosineNode, 593
 - operator(), 593
- mlpack::tree::CosineTree, 594
 - ~CosineTree, 597
 - basis, 600
 - BasisVector, 597
 - basisVector, 600
 - BinarySearch, 597
 - CalculateCentroid, 597
 - CalculateCosines, 597
 - Centroid, 598
 - centroid, 601
 - ColumnSampleLS, 598
 - ColumnSamplesLS, 598
 - ConstructBasis, 598
 - CosineNodeSplit, 598
 - CosineTree, 596
 - dataset, 601
 - delta, 601
 - epsilon, 601
 - FrobNormSquared, 598
 - frobNormSquared, 601

- GetDataset, 598
- GetFinalBasis, 599
- indices, 601
- L2Error, 599
- l2Error, 601
- l2NormsSquared, 602
- Left, 599
- left, 602
- ModifiedGramSchmidt, 599
- MonteCarloError, 599
- NumColumns, 600
- numColumns, 602
- parent, 602
- Right, 600
- right, 602
- SplitPointIndex, 600
- splitPointIndex, 602
- VectorIndices, 600
- mlpack::tree::CoverTree
 - ~CoverTree, 610
 - Base, 610
 - base, 619
 - Centroid, 610
 - Child, 611
 - Children, 611
 - children, 619
 - ComputeDistances, 611
 - CoverTree, 608–610
 - CreateChildren, 612
 - Dataset, 612
 - dataset, 619
 - Descendant, 612
 - DistanceComps, 612
 - distanceComps, 619
 - FurthestDescendantDistance, 612, 613
 - furthestDescendantDistance, 619
 - FurthestPointDistance, 613
 - HasSelfChildren, 613
 - IsLeaf, 613
 - localMetric, 620
 - Mat, 608
 - MaxDistance, 613, 614
 - Metric, 614
 - metric, 620
 - MinDistance, 614
 - MinimumBoundDistance, 614
 - MoveToUsedSet, 615
 - NumChildren, 615
 - NumDescendants, 615
 - numDescendants, 620
 - NumPoints, 615
 - Parent, 615
 - parent, 620
 - ParentDistance, 616
 - parentDistance, 620
 - Point, 616
 - point, 620
 - PruneFarSet, 616
 - RangeDistance, 616, 617
 - RemoveNewImplicitNodes, 617
 - Scale, 617
 - scale, 621
 - SortPointSet, 617
 - SplitNearFar, 618
 - Stat, 618
 - stat, 621
 - ToString, 619
- mlpack::tree::CoverTree< MetricType, RootPointPolicy, StatisticType >, 603
- mlpack::tree::CoverTree< MetricType, RootPointPolicy, StatisticType >::DualTreeTraverser< RuleType >, 621
- mlpack::tree::CoverTree< MetricType, RootPointPolicy, StatisticType >::DualTreeTraverser< RuleType >::DualCoverTreeMapEntry, 625
- mlpack::tree::CoverTree< MetricType, RootPointPolicy, StatisticType >::SingleTreeTraverser< RuleType >, 627
- mlpack::tree::CoverTree::DualTreeTraverser
 - DualTreeTraverser, 622
 - NumBaseCases, 623
 - NumPrunes, 623
 - numPrunes, 625
 - NumScores, 623
 - NumVisited, 623
 - PruneMap, 623
 - ReferenceRecursion, 623
 - rule, 625
 - Traverse, 623, 625
- mlpack::tree::CoverTree::DualTreeTraverser::DualCoverTreeMapEntry
 - baseCase, 626
 - operator<, 626
 - referenceNode, 626
 - score, 627
 - traversalInfo, 627
- mlpack::tree::CoverTree::SingleTreeTraverser
 - NumPrunes, 628
 - numPrunes, 628
 - rule, 629
 - SingleTreeTraverser, 628
 - Traverse, 628
- mlpack::tree::EmptyStatistic, 629
 - ~EmptyStatistic, 629
 - EmptyStatistic, 629, 630
 - ToString, 630
- mlpack::tree::ExampleTree
 - Centroid, 632

- Child, 634
- Descendant, 634
- ExampleTree, 632
- FurthestDescendantDistance, 634
- MaxDistance, 634
- Metric, 635
- metric, 637
- MinDistance, 635
- NumChildren, 635
- NumDescendants, 636
- NumPoints, 636
- Parent, 636
- ParentDistance, 636
- Point, 636
- RangeDistance, 636, 637
- Stat, 637
- stat, 637
- mlpack::tree::ExampleTree< MetricType, StatisticType, MatType >, 630
- mlpack::tree::FirstPointIsRoot, 638
 - ChooseRoot, 638
- mlpack::tree::MRKDStatistic, 640
 - Begin, 642
 - begin, 644
 - CenterOfMass, 643
 - centerOfMass, 644
 - Count, 643
 - count, 644
 - dataset, 644
 - DominatingCentroid, 643
 - dominatingCentroid, 644
 - isWhitelistValid, 645
 - leftStat, 645
 - MRKDStatistic, 642
 - parentStat, 645
 - rightStat, 645
 - sumOfSquaredNorms, 645
 - ToString, 643
 - Whitelist, 644
 - whitelist, 645
- mlpack::tree::MeanSplit
 - PerformSplit, 639
 - SplitNode, 640
- mlpack::tree::MeanSplit< BoundType, MatType >, 638
- mlpack::tree::TreeTraits
 - FirstPointIsCentroid, 647
 - HasOverlappingChildren, 647
 - HasSelfChildren, 647
 - RearrangesDataset, 647
- mlpack::tree::TreeTraits< BinarySpaceTree< BoundType, StatisticType, MatType > >, 647
 - FirstPointIsCentroid, 648
 - HasOverlappingChildren, 648
 - HasSelfChildren, 648
 - RearrangesDataset, 648
- mlpack::tree::TreeTraits< CoverTree< MetricType, Root←PointPolicy, StatisticType > >, 649
 - FirstPointIsCentroid, 649
 - HasOverlappingChildren, 649
 - HasSelfChildren, 649
 - RearrangesDataset, 650
- mlpack::tree::TreeTraits< TreeType >, 645
- mlpack::util, 117
 - cliDeleter, 118
 - GetVersion, 118
 - Indent, 118
- mlpack::util::CLIDeleter, 650
 - ~CLIDeleter, 650
 - CLIDeleter, 650
- mlpack::util::NullOutputStream, 651
 - NullOutputStream, 652
 - operator<<, 652–654
- mlpack::util::Option
 - Option, 655
- mlpack::util::Option< N >, 654
- mlpack::util::PrefixedOutputStream, 655
 - BaseLogic, 659
 - CallBaseLogic, 659
 - carriageReturned, 661
 - destination, 661
 - fatal, 661
 - ignoreInput, 661
 - operator<<, 659–661
 - prefix, 661
 - PrefixIfNeeded, 661
 - PrefixedOutputStream, 658
- mlpack::util::ProgramDoc, 662
 - documentation, 663
 - ProgramDoc, 662
 - programName, 663
- mlpack::util::SaveRestoreUtility, 663
 - ~SaveRestoreUtility, 664
 - LoadParameter, 664, 665
 - parameters, 666
 - ReadFile, 665
 - RecurseOnNodes, 665
 - SaveParameter, 665
 - SaveRestoreUtility, 664
 - WriteFile, 665
- ModifiedGramSchmidt
 - mlpack::tree::CosineTree, 599
- momentum
 - mlpack::amf::SVDBatchLearning, 151
- MonteCarloError
 - mlpack::tree::CosineTree, 599
- MoveControl
 - mlpack::optimization::SA, 481
- MoveCtrlSweep

- mlpack::optimization::SA, 482
- moveCtrlSweep
 - mlpack::optimization::SA, 484
- MoveSize
 - mlpack::optimization::SA, 482
- moveSize
 - mlpack::optimization::SA, 485
- MoveToUsedSet
 - mlpack::tree::CoverTree, 615
- n
 - mlpack::amf::SVDCompleteIncrementalLearning, 154
 - mlpack::amf::SVDCompleteIncrementalLearning<arma::sp_mat>, 157
 - mlpack::amf::SVDIncompleteIncrementalLearning, 159
 - mlpack::optimization::test::GeneralizedRosenbrock↵Function, 492
- NCA
 - mlpack::nca::NCA, 377
- NMFALSUpdate
 - mlpack::amf::NMFALSUpdate, 133
- NMFMultiplicativeDistanceUpdate
 - mlpack::amf::NMFMultiplicativeDistanceUpdate, 135
- NMFMultiplicativeDivergenceUpdate
 - mlpack::amf::NMFMultiplicativeDivergenceUpdate, 136
- naive
 - mlpack::emst::DualTreeBoruvka, 241
 - mlpack::fastmks::FastMKS, 253
 - mlpack::neighbor::NeighborSearch, 403
 - mlpack::range::RangeSearch, 516
 - RASearch, 671
- NaiveBayesClassifier
 - mlpack::naive_bayes::NaiveBayesClassifier, 374
- name
 - mlpack::ParamData, 497
- neighborPtr
 - mlpack::neighbor::LSHSearch, 394
- NeighborSearch
 - mlpack::neighbor::NeighborSearch, 400–402
- NeighborSearchRules
 - mlpack::neighbor::NeighborSearchRules, 407
- NeighborSearchStat
 - mlpack::neighbor::NeighborSearchStat, 413
- NeighborSearchTraversalInfo
 - mlpack::neighbor::NeighborSearchTraversalInfo, 417
- neighbors
 - mlpack::neighbor::NeighborSearchRules, 411
 - mlpack::neighbor::RASearchRules, 428
 - mlpack::range::RangeSearchRules, 522
- neighborsDistances
 - mlpack::emst::DTBRules, 230
- mlpack::emst::DualTreeBoruvka, 241
- neighborsInComponent
 - mlpack::emst::DTBRules, 231
 - mlpack::emst::DualTreeBoruvka, 241
- neighborsOutComponent
 - mlpack::emst::DTBRules, 231
 - mlpack::emst::DualTreeBoruvka, 241
- newIterateTmp
 - mlpack::optimization::L_BFGS, 465
- NextTemperature
 - mlpack::optimization::ExponentialSchedule, 453
- nm
 - mlpack::amf::SimpleResidueTermination, 142
- NoiseStdDev
 - mlpack::radical::Radical, 509, 510
- noiseStdDev
 - mlpack::radical::Radical, 511
- normOld
 - mlpack::amf::SimpleResidueTermination, 142
 - mlpack::amf::SimpleToleranceTermination, 147
- NormalizeLabels
 - mlpack::data, 94
- Normalizer
 - mlpack::kernel::EpanechnikovKernel, 296
 - mlpack::kernel::ExampleKernel, 298
 - mlpack::kernel::GaussianKernel, 303
 - mlpack::kernel::SphericalKernel, 328
- now
 - det.txt, 678
- NullOutputStream
 - mlpack::util::NullOutputStream, 652
- num_test_points
 - mlpack::amf::ValidationRMSETermination, 162
- NumBaseCases
 - mlpack::tree::BinarySpaceTree::DualTreeTraverser, 587
 - mlpack::tree::CoverTree::DualTreeTraverser, 623
- numBaseCases
 - mlpack::tree::BinarySpaceTree::DualTreeTraverser, 589
- NumBasis
 - mlpack::optimization::L_BFGS, 461
- numBasis
 - mlpack::optimization::L_BFGS, 465
- NumChildren
 - mlpack::tree::BinarySpaceTree, 580
 - mlpack::tree::CoverTree, 615
 - mlpack::tree::ExampleTree, 635
- numClass
 - mlpack::decision_stump::DecisionStump, 201
- NumColumns
 - mlpack::tree::CosineTree, 600
- numColumns
 - mlpack::tree::CosineTree, 602

- NumConstraints
 - mlpack::optimization::AugLagrangianTestFunction, 451
 - mlpack::optimization::GockenbachFunction, 455
 - mlpack::optimization::LRSDPFunction, 475
 - mlpack::optimization::LovaszThetaSDP, 467
- NumDescendants
 - mlpack::tree::BinarySpaceTree, 580
 - mlpack::tree::CoverTree, 615
 - mlpack::tree::ExampleTree, 636
- numDescendants
 - mlpack::tree::CoverTree, 620
- NumDistComputations
 - mlpack::neighbor::RASearchRules, 422
- numDistComputations
 - mlpack::neighbor::RASearchRules, 429
- NumEffectiveSamples
 - mlpack::neighbor::RASearchRules, 422
- NumFunctions
 - mlpack::nca::SoftmaxErrorFunction, 383
 - mlpack::optimization::test::GeneralizedRosenbrockFunction, 492
 - mlpack::optimization::test::SGDTestFunction, 495
 - mlpack::regression::LogisticRegressionFunction, 544
 - mlpack::svd::RegularizedSVDFunction, 561
- NumItems
 - mlpack::svd::RegularizedSVDFunction, 561
- numItems
 - mlpack::svd::RegularizedSVDFunction, 562
- NumPoints
 - mlpack::tree::BinarySpaceTree, 580
 - mlpack::tree::CoverTree, 615
 - mlpack::tree::ExampleTree, 636
- numProj
 - mlpack::neighbor::LSHSearch, 394
- NumPrunes
 - mlpack::tree::BinarySpaceTree::DualTreeTraverser, 588
 - mlpack::tree::BinarySpaceTree::SingleTreeTraverser, 591
 - mlpack::tree::CoverTree::DualTreeTraverser, 623
 - mlpack::tree::CoverTree::SingleTreeTraverser, 628
- numPrunes
 - mlpack::range::RangeSearch, 516
 - mlpack::tree::BinarySpaceTree::DualTreeTraverser, 589
 - mlpack::tree::BinarySpaceTree::SingleTreeTraverser, 593
 - mlpack::tree::CoverTree::DualTreeTraverser, 625
 - mlpack::tree::CoverTree::SingleTreeTraverser, 628
- numSamplesMade
 - mlpack::neighbor::RASearchRules, 429
- numSamplesReqd
 - mlpack::neighbor::RASearchRules, 429
- NumScores
 - mlpack::tree::BinarySpaceTree::DualTreeTraverser, 588
 - mlpack::tree::CoverTree::DualTreeTraverser, 623
- numScores
 - mlpack::tree::BinarySpaceTree::DualTreeTraverser, 589
- numTables
 - mlpack::neighbor::LSHSearch, 394
- NumUsers
 - mlpack::svd::RegularizedSVDFunction, 561
- numUsers
 - mlpack::svd::RegularizedSVDFunction, 562
- NumUsersForSimilarity
 - mlpack::cf::CF, 181
- numUsersForSimilarity
 - mlpack::cf::CF, 183
- NumVisited
 - mlpack::tree::BinarySpaceTree::DualTreeTraverser, 588
 - mlpack::tree::CoverTree::DualTreeTraverser, 623
- numVisited
 - mlpack::tree::BinarySpaceTree::DualTreeTraverser, 589
- numberOfPrunes
 - RASearch, 671
- NystroemMethod
 - mlpack::kernel::NystroemMethod, 317
- Objective
 - mlpack::lcc::LocalCoordinateCoding, 356
 - mlpack::sparse_coding::SparseCoding, 552
- ObtainDistinctSamples
 - mlpack::neighbor::RASearchRules, 423
- Offset
 - mlpack::kernel::HyperbolicTangentKernel, 305
 - mlpack::kernel::PolynomialKernel, 321
- offset
 - mlpack::kernel::HyperbolicTangentKernel, 306
 - mlpack::kernel::PolynomialKernel, 322
- offsets
 - mlpack::neighbor::LSHSearch, 394
- old_boost_test_definitions.hpp
 - BOOST_REQUIRE_GE, 858
 - BOOST_REQUIRE_GT, 858
 - BOOST_REQUIRE_LE, 858
 - BOOST_REQUIRE_LT, 858
 - BOOST_REQUIRE_NE, 858
- oldFromNew
 - mlpack::emst::DualTreeBoruvka, 241
- oldFromNewQueries
 - mlpack::neighbor::NeighborSearch, 403
 - mlpack::range::RangeSearch, 516
 - RASearch, 672

- oldFromNewReferences
 - mlpack::neighbor::NeighborSearch, 404
 - mlpack::range::RangeSearch, 517
 - RASearch, 672
- operator!=
 - mlpack::math::Range, 362
- operator<
 - mlpack::math::Range, 363
 - mlpack::tree::CoverTree::DualTreeTraverser::Dual↔
CoverTreeMapEntry, 626
- operator<<
 - mlpack::util::NullOutputStream, 652–654
 - mlpack::util::PrefixedOutputStream, 659–661
- operator>
 - mlpack::math::Range, 363
- operator*
 - mlpack::math::Range, 363, 364
- operator*=
 - mlpack::math::Range, 363
- operator()
 - mlpack::emst::DualTreeBoruvka::SortEdgesHelper,
242
 - mlpack::tree::CompareCosineNode, 593
- operator=
 - mlpack::bound::BallBound, 168
 - mlpack::bound::HRectBound, 175
 - mlpack::gmm::GMM, 280
- operator==
 - mlpack::math::Range, 363
- operator[]
 - mlpack::bound::BallBound, 168
 - mlpack::bound::HRectBound, 175, 176
- operator&
 - mlpack::math::Range, 362
- operator&=
 - mlpack::math::Range, 363
- operator |
 - mlpack::math::Range, 364
- operator | =
 - mlpack::bound::BallBound, 168
 - mlpack::bound::HRectBound, 176
 - mlpack::math::Range, 364
- Optimize
 - mlpack::optimization::AugLagrangian, 444
 - mlpack::optimization::L_BFGS, 462
 - mlpack::optimization::LRSDP, 471
 - mlpack::optimization::SA, 482
 - mlpack::optimization::SGD, 488, 489
- OptimizeCode
 - mlpack::lcc::LocalCoordinateCoding, 356
 - mlpack::sparse_coding::SparseCoding, 552
- OptimizeDictionary
 - mlpack::lcc::LocalCoordinateCoding, 356
 - mlpack::sparse_coding::SparseCoding, 552
- Optimizer
 - mlpack::nca::NCA, 379
- optimizer
 - mlpack::nca::NCA, 380
 - mlpack::svd::RegularizedSVD, 557
- Option
 - mlpack::util::Option, 655
- Orthogonalize
 - mlpack::math, 106
- OverclusteringFactor
 - mlpack::kmeans::KMeans, 338
- overclusteringFactor
 - mlpack::kmeans::KMeans, 339
- ownTree
 - mlpack::emst::DualTreeBoruvka, 241
- ownsMetric
 - mlpack::bound::BallBound, 170
- P
 - mlpack::kernel::PSpectrumStringKernel, 324
- p
 - mlpack::kernel::PSpectrumStringKernel, 325
 - mlpack::nca::SoftmaxErrorFunction, 384
- PARAM
 - cli.hpp, 760
- PARAM_DOUBLE
 - cli.hpp, 761
- PARAM_DOUBLE_REQ
 - cli.hpp, 761
- PARAM_FLAG
 - cli.hpp, 762
- PARAM_FLOAT
 - cli.hpp, 762
- PARAM_FLOAT_REQ
 - cli.hpp, 762
- PARAM_INT
 - cli.hpp, 763
- PARAM_INT_REQ
 - cli.hpp, 763
- PARAM_STRING
 - cli.hpp, 764
- PARAM_STRING_REQ
 - cli.hpp, 764
- PARAM_VECTOR
 - cli.hpp, 765
- PARAM_VECTOR_REQ
 - cli.hpp, 765
- PCA
 - mlpack::pca::PCA, 499
- PROGRAM_INFO
 - cli.hpp, 766
- PSpectrumStringKernel
 - mlpack::kernel::PSpectrumStringKernel, 323
- Parameters

- mlpack::regression::LinearRegression, 534
- mlpack::regression::LogisticRegression, 539, 540
- parameters
 - mlpack::nn::SparseAutoencoder, 435
 - mlpack::regression::LinearRegression, 536
 - mlpack::regression::LogisticRegression, 540
 - mlpack::util::SaveRestoreUtility, 666
- Parent
 - mlpack::tree::BinarySpaceTree, 580
 - mlpack::tree::CoverTree, 615
 - mlpack::tree::ExampleTree, 636
- parent
 - mlpack::emst::UnionFind, 247
 - mlpack::tree::BinarySpaceTree, 585
 - mlpack::tree::CosineTree, 602
 - mlpack::tree::CoverTree, 620
- ParentDistance
 - mlpack::tree::BinarySpaceTree, 580, 581
 - mlpack::tree::CoverTree, 616
 - mlpack::tree::ExampleTree, 636
- parentDistance
 - mlpack::tree::BinarySpaceTree, 585
 - mlpack::tree::CoverTree, 620
- parentStat
 - mlpack::tree::MRKDStatistic, 645
- ParseCommandLine
 - mlpack::CLI, 193
- ParseStream
 - mlpack::CLI, 193
- Partitioner
 - mlpack::kmeans::KMeans, 338
- partitioner
 - mlpack::kmeans::KMeans, 339
- Percentage
 - mlpack::kmeans::RefinedStart, 344
- percentage
 - mlpack::kmeans::RefinedStart, 345
- Perceptron
 - mlpack::perceptron::Perceptron, 502
- PerformSplit
 - mlpack::tree::MeanSplit, 639
- perturbed
 - mlpack::radical::Radical, 511
- phi
 - mlpack::gmm, 99, 100
- Point
 - mlpack::tree::BinarySpaceTree, 581
 - mlpack::tree::CoverTree, 616
 - mlpack::tree::ExampleTree, 636
- point
 - mlpack::tree::CoverTree, 620
- PolynomialKernel
 - mlpack::kernel::PolynomialKernel, 320
- Precalculate
 - mlpack::nca::SoftmaxErrorFunction, 383
- precalculated
 - mlpack::nca::SoftmaxErrorFunction, 384
- Predict
 - mlpack::hmm::HMM, 290
 - mlpack::regression::LinearRegression, 534
 - mlpack::regression::LogisticRegression, 540
- Predictors
 - mlpack::regression::LogisticRegressionFunction, 544
- predictors
 - mlpack::regression::LogisticRegressionFunction, 544
- prefix
 - mlpack::util::PrefixedOutputStream, 661
- PrefixIfNeeded
 - mlpack::util::PrefixedOutputStream, 661
- PrefixedOutputStream
 - mlpack::util::PrefixedOutputStream, 658
- prereqs.hpp
 - _USE_MATH_DEFINES, 856
 - force_inline, 857
 - M_PI, 857
- Print
 - mlpack::CLI, 194
- PrintHelp
 - mlpack::CLI, 194
- PrintLeafMembership
 - mlpack::det, 96
- PrintTimer
 - mlpack::Timers, 566
- PrintVariableImportance
 - mlpack::det, 96
- Probabilities
 - mlpack::distribution::DiscreteDistribution, 217
 - mlpack::naive_bayes::NaiveBayesClassifier, 375
- probabilities
 - mlpack::distribution::DiscreteDistribution, 218
 - mlpack::naive_bayes::NaiveBayesClassifier, 376
- Probability
 - mlpack::distribution::DiscreteDistribution, 217
 - mlpack::distribution::GaussianDistribution, 221
 - mlpack::distribution::LaplaceDistribution, 224
 - mlpack::gmm::GMM, 280, 281
- products
 - mlpack::fastmks::FastMKSRules, 259
- ProgramDoc
 - mlpack::util::ProgramDoc, 662
- programName
 - mlpack::CLI, 195
 - mlpack::util::ProgramDoc, 663
- ProjectDictionary
 - mlpack::sparse_coding::SparseCoding, 552
- projections
 - mlpack::neighbor::LSHSearch, 394
- PruneAndUpdate

- mlpack::det::DTree, 209
- PruneFarSet
 - mlpack::tree::CoverTree, 616
- PruneMap
 - mlpack::tree::CoverTree::DualTreeTraverser, 623
- QUIC_SVD
 - mlpack::svd::QUIC_SVD, 554
- queryCopy
 - mlpack::neighbor::NeighborSearch, 404
 - mlpack::range::RangeSearch, 517
 - RASearch, 672
- queryKernels
 - mlpack::fastmks::FastMKSRules, 259
- querySet
 - mlpack::fastmks::FastMKS, 253
 - mlpack::fastmks::FastMKSRules, 260
 - mlpack::neighbor::LSHSearch, 394
 - mlpack::neighbor::NeighborSearch, 404
 - mlpack::neighbor::NeighborSearchRules, 411
 - mlpack::neighbor::RASearchRules, 429
 - mlpack::range::RangeSearch, 517
 - mlpack::range::RangeSearchRules, 522
 - RASearch, 672
- queryTree
 - mlpack::fastmks::FastMKS, 253
 - mlpack::neighbor::NeighborSearch, 404
 - mlpack::range::RangeSearch, 517
 - RASearch, 672
- RASearch
 - ~RASearch, 670
 - hasQuerySet, 671
 - metric, 671
 - naive, 671
 - numberOfPrunes, 671
 - oldFromNewQueries, 672
 - oldFromNewReferences, 672
 - queryCopy, 672
 - querySet, 672
 - queryTree, 672
 - RASearch, 668, 669
 - referenceCopy, 672
 - referenceSet, 673
 - referenceTree, 673
 - ResetQueryTree, 670
 - ResetRAQueryStat, 670
 - Search, 670
 - singleMode, 673
 - ToString, 671
 - treeOwner, 673
- RASearch< SortPolicy, MetricType, TreeType >, 666
 - mlpack::neighbor::RASearchRules, 428
- RASearchRules
 - mlpack::neighbor::RASearchRules, 422
- rSVDFunc
 - mlpack::svd::RegularizedSVD, 557
- Radical
 - mlpack::radical::Radical, 508
- Radius
 - mlpack::bound::BallBound, 169
- radius
 - mlpack::bound::BallBound, 170
- randGen
 - mlpack::math, 108
- RandInt
 - mlpack::math, 106
- RandNormal
 - mlpack::math, 107
- randNormalDist
 - mlpack::math, 108
- randUniformDist
 - mlpack::math, 108
- RandVector
 - mlpack::math, 108
- Random
 - mlpack::distribution::DiscreteDistribution, 218
 - mlpack::distribution::GaussianDistribution, 221
 - mlpack::distribution::LaplaceDistribution, 224
 - mlpack::gmm::GMM, 281
 - mlpack::math, 107
- RandomAcolInitialization
 - mlpack::amf::RandomAcolInitialization, 138
- RandomInitialization
 - mlpack::amf::RandomInitialization, 139
 - mlpack::perceptron::RandomInitialization, 504
- RandomPartition
 - mlpack::kmeans::RandomPartition, 342
- RandomSeed
 - mlpack::math, 107
- Range
 - mlpack::math::Range, 361
- range
 - mlpack::range::RangeSearchRules, 523
- RangeDistance
 - mlpack::bound::BallBound, 169
 - mlpack::bound::HRectBound, 176
 - mlpack::tree::BinarySpaceTree, 581
 - mlpack::tree::CoverTree, 616, 617
 - mlpack::tree::ExampleTree, 636, 637
- RangeSearch
 - mlpack::range::RangeSearch, 513, 514
- RangeSearchRules
 - mlpack::range::RangeSearchRules, 520
- RangeSearchStat
 - mlpack::range::RangeSearchStat, 524
- Rank
 - mlpack::cf::CF, 181
 - mlpack::svd::RegularizedSVDFunction, 561

- rank
 - mlpack::cf::CF, 183
 - mlpack::emst::UnionFind, 247
 - mlpack::kernel::NystroemMethod, 319
 - mlpack::svd::RegularizedSVD, 557
 - mlpack::svd::RegularizedSVDFunction, 562
- Rating
 - mlpack::cf::CF, 182
- rating
 - mlpack::cf::CF, 183
- Ratio
 - mlpack::det::DTree, 210
- ratio
 - mlpack::det::DTree, 213
- ratios
 - mlpack::gmm::EigenvalueRatioConstraint, 265
- ReadFile
 - mlpack::util::SaveRestoreUtility, 665
- RearrangesDataset
 - mlpack::tree::TreeTraits, 647
 - mlpack::tree::TreeTraits< BinarySpaceTree< BoundType, StatisticType, MatType > >, 648
 - mlpack::tree::TreeTraits< CoverTree< MetricType, RootPointPolicy, StatisticType > >, 650
- RecurseOnNodes
 - mlpack::util::SaveRestoreUtility, 665
- referenceCopy
 - mlpack::neighbor::NeighborSearch, 404
 - mlpack::range::RangeSearch, 517
 - RASearch, 672
- referenceKernels
 - mlpack::fastmks::FastMKSRules, 260
- referenceNode
 - mlpack::tree::CoverTree::DualTreeTraverser::Dual< CoverTreeMapEntry, 626
- ReferenceRecursion
 - mlpack::tree::CoverTree::DualTreeTraverser, 623
- referenceSet
 - mlpack::fastmks::FastMKS, 253
 - mlpack::fastmks::FastMKSRules, 260
 - mlpack::neighbor::LSHSearch, 394
 - mlpack::neighbor::NeighborSearch, 404
 - mlpack::neighbor::NeighborSearchRules, 411
 - mlpack::neighbor::RASearchRules, 429
 - mlpack::range::RangeSearch, 517
 - mlpack::range::RangeSearchRules, 523
 - RASearch, 673
- referenceTree
 - mlpack::fastmks::FastMKS, 254
 - mlpack::neighbor::NeighborSearch, 405
 - mlpack::range::RangeSearch, 517
 - RASearch, 673
- RefinedStart
 - mlpack::kmeans::RefinedStart, 344
- RegisterProgramDoc
 - mlpack::CLI, 194
- Regress
 - mlpack::regression::LARS, 529
- regularization
 - det.txt, 678
- RegularizedSVD
 - mlpack::svd::RegularizedSVD, 556
- RegularizedSVDFunction
 - mlpack::svd::RegularizedSVDFunction, 559
- RemoveDuplicateFlags
 - mlpack::CLI, 194
- RemoveNewImplicitNodes
 - mlpack::tree::CoverTree, 617
- RemoveRows
 - mlpack::math, 108
- Replicates
 - mlpack::radical::Radical, 510
- replicates
 - mlpack::radical::Radical, 511
- RequiredOptions
 - mlpack::CLI, 194
- requiredOptions
 - mlpack::CLI, 195
- Rescore
 - mlpack::emst::DTBRules, 228
 - mlpack::fastmks::FastMKSRules, 257
 - mlpack::neighbor::NeighborSearchRules, 408, 409
 - mlpack::neighbor::RASearchRules, 423
 - mlpack::range::RangeSearchRules, 520, 521
- ResetQueryTree
 - RASearch, 670
- ResetRAQueryStat
 - RASearch, 670
- residue
 - mlpack::amf::SimpleResidueTermination, 142
 - mlpack::amf::SimpleToleranceTermination, 147
- residueOld
 - mlpack::amf::SimpleToleranceTermination, 147
- Responses
 - mlpack::regression::LogisticRegressionFunction, 544
- responses
 - mlpack::regression::LogisticRegressionFunction, 545
- ReturnIndicesFromTable
 - mlpack::neighbor::LSHSearch, 392
- reverseStepCount
 - mlpack::amf::SimpleToleranceTermination, 147
 - mlpack::amf::ValidationRMSETermination, 162
- reverseStepTolerance
 - mlpack::amf::SimpleToleranceTermination, 147
 - mlpack::amf::ValidationRMSETermination, 162
- RevertLabels
 - mlpack::data, 94
- rf

- mlpack::optimization::test::RosenbrockWood↵
Function, 494
- Rho
 - mlpack::nn::SparseAutoencoder, 434
 - mlpack::nn::SparseAutoencoderFunction, 439
- rho
 - mlpack::nn::SparseAutoencoder, 435
 - mlpack::nn::SparseAutoencoderFunction, 440
- Right
 - mlpack::det::DTree, 210
 - mlpack::tree::BinarySpaceTree, 581, 582
 - mlpack::tree::CosineTree, 600
- right
 - mlpack::det::DTree, 213
 - mlpack::tree::BinarySpaceTree, 585
 - mlpack::tree::CosineTree, 602
- rightStat
 - mlpack::tree::MRKDStatistic, 645
- rmse
 - mlpack::amf::ValidationRMSETermination, 162
- rmseOld
 - mlpack::amf::ValidationRMSETermination, 163
- Root
 - mlpack::det::DTree, 210
- root
 - mlpack::det::DTree, 213
- RosenbrockFunction
 - mlpack::optimization::test::RosenbrockFunction, 493
- RosenbrockWoodFunction
 - mlpack::optimization::test::RosenbrockWood↵
Function, 494
- rule
 - mlpack::tree::BinarySpaceTree::DualTreeTraverser,
590
 - mlpack::tree::BinarySpaceTree::SingleTreeTraverser,
593
 - mlpack::tree::CoverTree::DualTreeTraverser, 625
 - mlpack::tree::CoverTree::SingleTreeTraverser, 629
- s
 - mlpack::optimization::L_BFGS, 465
- SA
 - mlpack::optimization::SA, 478
- SGD
 - mlpack::optimization::SGD, 488
- SGDTestFunction
 - mlpack::optimization::test::SGDTestFunction, 495
- SVDBatchLearning
 - mlpack::amf::SVDBatchLearning, 149
- SVDBatchLearning::HUpdate< arma::sp_mat >
 - mlpack::amf, 92
- SVDBatchLearning::WUpdate< arma::sp_mat >
 - mlpack::amf, 92
- SVDCompleteIncrementalLearning
 - mlpack::amf::SVDCompleteIncrementalLearning,
152
 - mlpack::amf::SVDCompleteIncrementalLearning<
arma::sp_mat >, 155
- SVDIncompleteIncrementalLearning
 - mlpack::amf::SVDIncompleteIncrementalLearning,
157
- SVDIncompleteIncrementalLearning::HUpdate< arma↵
::sp_mat >
 - mlpack::amf, 92
- SVDIncompleteIncrementalLearning::WUpdate< arma↵
::sp_mat >
 - mlpack::amf, 92
- sampleAtLeaves
 - mlpack::neighbor::RASearchRules, 429
- samplingRatio
 - mlpack::neighbor::RASearchRules, 429
- Samplings
 - mlpack::kmeans::RefinedStart, 345
- samplings
 - mlpack::kmeans::RefinedStart, 345
- SanitizeString
 - mlpack::CLI, 194
- Save
 - mlpack::data, 94
 - mlpack::gmm::GMM, 281
- SaveHMM
 - mlpack::hmm, 101
- SaveParameter
 - mlpack::util::SaveRestoreUtility, 665
- SaveRestoreUtility
 - mlpack::util::SaveRestoreUtility, 664
- Scale
 - mlpack::distribution::LaplaceDistribution, 224
 - mlpack::kernel::HyperbolicTangentKernel, 306
 - mlpack::tree::CoverTree, 617
- scale
 - mlpack::distribution::LaplaceDistribution, 225
 - mlpack::kernel::HyperbolicTangentKernel, 306
 - mlpack::tree::CoverTree, 621
- ScaleData
 - mlpack::pca::PCA, 500, 501
- scaleData
 - mlpack::pca::PCA, 501
- Score
 - mlpack::emst::DTBRules, 228, 229
 - mlpack::fastmks::FastMKSRules, 257, 258
 - mlpack::neighbor::NeighborSearchRules, 409
 - mlpack::neighbor::RASearchRules, 425, 426
 - mlpack::range::RangeSearchRules, 521
- score
 - mlpack::tree::CoverTree::DualTreeTraverser::Dual↵
CoverTreeMapEntry, 627
- Scores

- mlpack::emst::DTBRules, 229
- mlpack::fastmks::FastMKSRules, 258
- mlpack::neighbor::NeighborSearchRules, 409, 410
- scores
 - mlpack::emst::DTBRules, 231
 - mlpack::fastmks::FastMKSRules, 260
 - mlpack::neighbor::NeighborSearchRules, 411
- Search
 - mlpack::fastmks::FastMKS, 251
 - mlpack::neighbor::LSHSearch, 392
 - mlpack::neighbor::NeighborSearch, 403
 - mlpack::range::RangeSearch, 515
 - RASearch, 670
- SearchDirection
 - mlpack::optimization::L_BFGS, 462
- SecondBound
 - mlpack::neighbor::NeighborSearchStat, 415
- secondBound
 - mlpack::neighbor::NeighborSearchStat, 416
- secondHashSize
 - mlpack::neighbor::LSHSearch, 395
- secondHashTable
 - mlpack::neighbor::LSHSearch, 395
- secondHashWeights
 - mlpack::neighbor::LSHSearch, 395
- Select
 - mlpack::kernel::KMeansSelection, 311
 - mlpack::kernel::OrderedSelection, 319
 - mlpack::kernel::RandomSelection, 326
- SelfKernel
 - mlpack::fastmks::FastMKSSStat, 263
- selfKernel
 - mlpack::fastmks::FastMKSSStat, 263
- separately
 - TREE_EXPLANATION.txt, 756
- set
 - core/CMakeLists.txt, 681
 - core/data/CMakeLists.txt, 681
 - core/dists/CMakeLists.txt, 682
 - core/kernels/CMakeLists.txt, 682
 - core/math/CMakeLists.txt, 683
 - core/metrics/CMakeLists.txt, 683
 - core/optimizers/CMakeLists.txt, 684
 - core/optimizers/aug_lagrangian/CMakeLists.txt, 684
 - core/optimizers/lbfgs/CMakeLists.txt, 685
 - core/optimizers/lrsdp/CMakeLists.txt, 685
 - core/optimizers/sa/CMakeLists.txt, 685
 - core/optimizers/sgd/CMakeLists.txt, 686
 - core/tree/CMakeLists.txt, 686
 - core/util/CMakeLists.txt, 687
 - methods/CMakeLists.txt, 689
 - methods/amf/CMakeLists.txt, 687
 - methods/amf/init_rules/CMakeLists.txt, 688
 - methods/amf/termination_policies/CMakeLists.txt, 688
 - methods/amf/update_rules/CMakeLists.txt, 689
 - methods/cf/CMakeLists.txt, 689
 - methods/decision_stump/CMakeLists.txt, 690
 - methods/det/CMakeLists.txt, 690
 - methods/emst/CMakeLists.txt, 691
 - methods/fastmks/CMakeLists.txt, 691
 - methods/gmm/CMakeLists.txt, 692
 - methods/hmm/CMakeLists.txt, 692
 - methods/kernel_pca/CMakeLists.txt, 692
 - methods/kernel_pca/kernel_rules/CMakeLists.txt, 693
 - methods/kmeans/CMakeLists.txt, 693
 - methods/lars/CMakeLists.txt, 694
 - methods/linear_regression/CMakeLists.txt, 694
 - methods/local_coordinate_coding/CMakeLists.txt, 694
 - methods/logistic_regression/CMakeLists.txt, 695
 - methods/lsh/CMakeLists.txt, 695
 - methods/mvu/CMakeLists.txt, 696
 - methods/naive_bayes/CMakeLists.txt, 696
 - methods/nca/CMakeLists.txt, 696
 - methods/neighbor_search/CMakeLists.txt, 697
 - methods/nystroem_method/CMakeLists.txt, 697
 - methods/pca/CMakeLists.txt, 698
 - methods/perceptron/CMakeLists.txt, 698
 - methods/perceptron/initialization_methods/CMakeLists.txt, 699
 - methods/perceptron/learning_policies/CMakeLists.txt, 699
 - methods/quic_svd/CMakeLists.txt, 699
 - methods/radical/CMakeLists.txt, 700
 - methods/range_search/CMakeLists.txt, 700
 - methods/rann/CMakeLists.txt, 701
 - methods/regularized_svd/CMakeLists.txt, 701
 - methods/sparse_autoencoder/CMakeLists.txt, 701
 - methods/sparse_coding/CMakeLists.txt, 702
- SetupSplitAttribute
 - mlpack::decision_stump::DecisionStump, 199
- sfnai_utility.hpp
 - HAS_MEM_FUNC, 772
- Shuffle
 - mlpack::optimization::SGD, 489
- shuffle
 - mlpack::optimization::SGD, 490
- Sigma
 - mlpack::optimization::AugLagrangian, 445
 - mlpack::optimization::AugLagrangianFunction, 449
- sigma
 - mlpack::optimization::AugLagrangianFunction, 450
- Sigmoid
 - mlpack::nn::SparseAutoencoder, 434
 - mlpack::nn::SparseAutoencoderFunction, 439

- SimpleResidueTermination
 - mlpack::amf::SimpleResidueTermination, 140
- SimpleToleranceTermination
 - mlpack::amf::SimpleToleranceTermination, 144
- single
 - mlpack::fastmks::FastMKS, 254
- singleMode
 - mlpack::neighbor::NeighborSearch, 405
 - mlpack::range::RangeSearch, 518
 - RASearch, 673
- singleSampleLimit
 - mlpack::neighbor::RASearchRules, 430
- SingleTreeTraverser
 - mlpack::tree::BinarySpaceTree::SingleTreeTraverser, 591
 - mlpack::tree::CoverTree::SingleTreeTraverser, 628
- singleton
 - mlpack::CLI, 195
- SoftmaxErrorFunction
 - mlpack::nca::SoftmaxErrorFunction, 381
- SortDistance
 - mlpack::neighbor::FurthestNeighborSort, 387
 - mlpack::neighbor::NearestNeighborSort, 398
- SortFun
 - mlpack::emst::DualTreeBoruvka, 242
- SortPointSet
 - mlpack::tree::CoverTree, 617
- SparseAutoencoder
 - mlpack::nn::SparseAutoencoder, 432
- SparseAutoencoderFunction
 - mlpack::nn::SparseAutoencoderFunction, 437
- SparseCoding
 - mlpack::sparse_coding::SparseCoding, 549
- SphericalKernel
 - mlpack::kernel::SphericalKernel, 327
- Split
 - mlpack::decision_stump::DecisionStump, 200
- split
 - mlpack::decision_stump::DecisionStump, 201
- SplitAttribute
 - mlpack::decision_stump::DecisionStump, 200
- splitAttribute
 - mlpack::decision_stump::DecisionStump, 201
- SplitData
 - mlpack::det::DTree, 210
- SplitDim
 - mlpack::det::DTree, 210
- splitDim
 - mlpack::det::DTree, 213
- SplitDimension
 - mlpack::tree::BinarySpaceTree, 582
- splitDimension
 - mlpack::tree::BinarySpaceTree, 585
- SplitNearFar
 - mlpack::tree::CoverTree, 618
- SplitNode
 - mlpack::tree::BinarySpaceTree, 582
 - mlpack::tree::MeanSplit, 640
- SplitPointIndex
 - mlpack::tree::CosineTree, 600
- splitPointIndex
 - mlpack::tree::CosineTree, 602
- SplitValue
 - mlpack::det::DTree, 210
- splitValue
 - mlpack::det::DTree, 213
- SquaredEuclideanDistance
 - mlpack::metric, 109
- src/mlpack/CMakeLists.txt, 681
- src/mlpack/core.hpp, 703
- src/mlpack/core/CMakeLists.txt, 681
- src/mlpack/core/data/CMakeLists.txt, 681
- src/mlpack/core/data/load.hpp, 703
- src/mlpack/core/data/normalize_labels.hpp, 704
- src/mlpack/core/data/save.hpp, 705
- src/mlpack/core/dists/CMakeLists.txt, 682
- src/mlpack/core/dists/discrete_distribution.hpp, 706
- src/mlpack/core/dists/gaussian_distribution.hpp, 707
- src/mlpack/core/dists/laplace_distribution.hpp, 708
- src/mlpack/core/kernels/CMakeLists.txt, 682
- src/mlpack/core/kernels/cosine_distance.hpp, 708
- src/mlpack/core/kernels/epanetchnikov_kernel.hpp, 709
- src/mlpack/core/kernels/example_kernel.hpp, 710
- src/mlpack/core/kernels/gaussian_kernel.hpp, 711
- src/mlpack/core/kernels/hyperbolic_tangent_kernel.hpp, 712
- src/mlpack/core/kernels/kernel_traits.hpp, 712
- src/mlpack/core/kernels/laplacian_kernel.hpp, 713
- src/mlpack/core/kernels/linear_kernel.hpp, 714
- src/mlpack/core/kernels/polynomial_kernel.hpp, 715
- src/mlpack/core/kernels/pspectrum_string_kernel.hpp, 716
- src/mlpack/core/kernels/spherical_kernel.hpp, 717
- src/mlpack/core/kernels/triangular_kernel.hpp, 718
- src/mlpack/core/math/CMakeLists.txt, 683
- src/mlpack/core/math/clamp.hpp, 718
- src/mlpack/core/math/lin_alg.hpp, 720
- src/mlpack/core/math/random.hpp, 721
- src/mlpack/core/math/range.hpp, 722
- src/mlpack/core/math/round.hpp, 723
- src/mlpack/core/metrics/CMakeLists.txt, 683
- src/mlpack/core/metrics/ip_metric.hpp, 724
- src/mlpack/core/metrics/lmetric.hpp, 725
- src/mlpack/core/metrics/mahalanobis_distance.hpp, 726
- src/mlpack/core/optimizers/CMakeLists.txt, 684
- src/mlpack/core/optimizers/aug_lagrangian/CMakeLists.txt, 684

src/mlpack/core/optimizers/aug_lagrangian/aug_↵
lagrangian.hpp, 726

src/mlpack/core/optimizers/aug_lagrangian/aug_↵
lagrangian_function.hpp, 728

src/mlpack/core/optimizers/aug_lagrangian/aug_↵
lagrangian_test_functions.hpp, 729

src/mlpack/core/optimizers/lbfgs/CMakeLists.txt, 684

src/mlpack/core/optimizers/lbfgs/lbfgs.hpp, 730

src/mlpack/core/optimizers/lbfgs/test_functions.hpp, 731

src/mlpack/core/optimizers/lrsdp/CMakeLists.txt, 685

src/mlpack/core/optimizers/lrsdp/lrsdp.hpp, 732

src/mlpack/core/optimizers/lrsdp/lrsdp_function.hpp, 732

src/mlpack/core/optimizers/sa/CMakeLists.txt, 685

src/mlpack/core/optimizers/sa/exponential_schedule.hpp, 734

src/mlpack/core/optimizers/sa/sa.hpp, 735

src/mlpack/core/optimizers/sgd/CMakeLists.txt, 686

src/mlpack/core/optimizers/sgd/sgd.hpp, 735

src/mlpack/core/optimizers/sgd/test_function.hpp, 737

src/mlpack/core/tree/CMakeLists.txt, 686

src/mlpack/core/tree/TREE_EXPLANATION.txt, 756

src/mlpack/core/tree/ballbound.hpp, 737

src/mlpack/core/tree/binary_space_tree.hpp, 739

src/mlpack/core/tree/binary_space_tree/binary_space_↵
tree.hpp, 738

src/mlpack/core/tree/binary_space_tree/dual_tree_↵
traverser.hpp, 740

src/mlpack/core/tree/binary_space_tree/mean_split.hpp, 742

src/mlpack/core/tree/binary_space_tree/single_tree_↵
traverser.hpp, 743

src/mlpack/core/tree/binary_space_tree/traits.hpp, 745

src/mlpack/core/tree/bounds.hpp, 747

src/mlpack/core/tree/cosine_tree/cosine_tree.hpp, 748

src/mlpack/core/tree/cover_tree.hpp, 750

src/mlpack/core/tree/cover_tree/cover_tree.hpp, 749

src/mlpack/core/tree/cover_tree/dual_tree_traverser.hpp, 741

src/mlpack/core/tree/cover_tree/first_point_is_root.hpp, 750

src/mlpack/core/tree/cover_tree/single_tree_traverser.↵
hpp, 744

src/mlpack/core/tree/cover_tree/traits.hpp, 746

src/mlpack/core/tree/example_tree.hpp, 752

src/mlpack/core/tree/hrectbound.hpp, 752

src/mlpack/core/tree/mrkd_statistic.hpp, 753

src/mlpack/core/tree/rectangle_tree.hpp, 754

src/mlpack/core/tree/statistic.hpp, 755

src/mlpack/core/tree/traversal_info.hpp, 755

src/mlpack/core/tree/tree_traits.hpp, 757

src/mlpack/core/util/CMakeLists.txt, 687

src/mlpack/core/util/arma_traits.hpp, 758

src/mlpack/core/util/cli.hpp, 759

src/mlpack/core/util/cli_deleter.hpp, 767

src/mlpack/core/util/log.hpp, 767

src/mlpack/core/util/nullostream.hpp, 768

src/mlpack/core/util/option.hpp, 769

src/mlpack/core/util/prefixedostream.hpp, 770

src/mlpack/core/util/save_restore_utility.hpp, 771

src/mlpack/core/util/sfinae_utility.hpp, 772

src/mlpack/core/util/string_util.hpp, 773

src/mlpack/core/util/timers.hpp, 774

src/mlpack/core/util/version.hpp, 775

 __MLPACK_VERSION_MAJOR, 775

 __MLPACK_VERSION_MINOR, 775

 __MLPACK_VERSION_PATCH, 776

src/mlpack/methods/CMakeLists.txt, 689

src/mlpack/methods/amf/CMakeLists.txt, 687

src/mlpack/methods/amf/amf.hpp, 776

src/mlpack/methods/amf/init_rules/CMakeLists.txt, 688

src/mlpack/methods/amf/init_rules/average_init.hpp, 777

src/mlpack/methods/amf/init_rules/random_acol_init.hpp, 777

src/mlpack/methods/amf/init_rules/random_init.hpp, 778

src/mlpack/methods/amf/termination_policies/CMake↵
Lists.txt, 688

src/mlpack/methods/amf/termination_policies/complete↵
_incremental_termination.hpp, 780

src/mlpack/methods/amf/termination_policies/incomplete↵
_incremental_termination.hpp, 781

src/mlpack/methods/amf/termination_policies/simple_↵
residue_termination.hpp, 781

src/mlpack/methods/amf/termination_policies/simple_↵
tolerance_termination.hpp, 783

src/mlpack/methods/amf/termination_policies/validation↵
_RMSE_termination.hpp, 784

src/mlpack/methods/amf/update_rules/CMakeLists.txt, 688

src/mlpack/methods/amf/update_rules/nmf_als.hpp, 784

src/mlpack/methods/amf/update_rules/nmf_mult_dist.hpp, 785

src/mlpack/methods/amf/update_rules/nmf_mult_div.hpp, 787

src/mlpack/methods/amf/update_rules/svd_batch_↵
learning.hpp, 787

src/mlpack/methods/amf/update_rules/svd_complete_↵
incremental_learning.hpp, 788

src/mlpack/methods/amf/update_rules/svd_incomplete↵
_incremental_learning.hpp, 788

src/mlpack/methods/cf/CMakeLists.txt, 689

src/mlpack/methods/cf/cf.hpp, 789

src/mlpack/methods/decision_stump/CMakeLists.txt, 690

src/mlpack/methods/decision_stump/decision_stump.hpp, 790

src/mlpack/methods/det/CMakeLists.txt, 690

src/mlpack/methods/det/dt_utils.hpp, 790

src/mlpack/methods/det/dtree.hpp, 791

src/mlpack/methods/emst/CMakeLists.txt, 691

src/mlpack/methods/emst/dtb.hpp, 792
 src/mlpack/methods/emst/dtb_rules.hpp, 793
 src/mlpack/methods/emst/dtb_stat.hpp, 794
 src/mlpack/methods/emst/edge_pair.hpp, 795
 src/mlpack/methods/emst/union_find.hpp, 796
 src/mlpack/methods/fastmks/CMakeLists.txt, 691
 src/mlpack/methods/fastmks/fastmks.hpp, 797
 src/mlpack/methods/fastmks/fastmks_rules.hpp, 798
 src/mlpack/methods/fastmks/fastmks_stat.hpp, 798
 src/mlpack/methods/gmm/CMakeLists.txt, 691
 src/mlpack/methods/gmm/diagonal_constraint.hpp, 799
 src/mlpack/methods/gmm/eigenvalue_ratio_constraint.↵
 hpp, 800
 src/mlpack/methods/gmm/em_fit.hpp, 801
 src/mlpack/methods/gmm/gmm.hpp, 802
 src/mlpack/methods/gmm/no_constraint.hpp, 803
 src/mlpack/methods/gmm/phi.hpp, 803
 src/mlpack/methods/gmm/positive_definite_constraint.↵
 hpp, 804
 src/mlpack/methods/hmm/CMakeLists.txt, 692
 src/mlpack/methods/hmm/hmm.hpp, 806
 src/mlpack/methods/hmm/hmm_util.hpp, 807
 src/mlpack/methods/kernel_pca/CMakeLists.txt, 692
 src/mlpack/methods/kernel_pca/kernel_pca.hpp, 807
 src/mlpack/methods/kernel_pca/kernel_rules/CMake↵
 Lists.txt, 693
 src/mlpack/methods/kernel_pca/kernel_rules/naive_↵
 method.hpp, 808
 src/mlpack/methods/kernel_pca/kernel_rules/nystroem↵
 _method.hpp, 809
 src/mlpack/methods/kmeans/CMakeLists.txt, 693
 src/mlpack/methods/kmeans/allow_empty_clusters.hpp,
 810
 src/mlpack/methods/kmeans/kmeans.hpp, 811
 src/mlpack/methods/kmeans/max_variance_new_↵
 cluster.hpp, 813
 src/mlpack/methods/kmeans/random_partition.hpp, 814
 src/mlpack/methods/kmeans/refined_start.hpp, 816
 src/mlpack/methods/lars/CMakeLists.txt, 693
 src/mlpack/methods/lars/lars.hpp, 817
 src/mlpack/methods/linear_regression/CMakeLists.txt,
 694
 src/mlpack/methods/linear_regression/linear_regression.↵
 hpp, 818
 src/mlpack/methods/local_coordinate_coding/CMake↵
 Lists.txt, 694
 src/mlpack/methods/local_coordinate_coding/lcc.hpp, 819
 src/mlpack/methods/logistic_regression/CMakeLists.txt,
 695
 src/mlpack/methods/logistic_regression/logistic_regression.↵
 hpp, 819
 src/mlpack/methods/logistic_regression/logistic_regression↵
 _function.hpp, 820
 src/mlpack/methods/lsh/CMakeLists.txt, 695
 src/mlpack/methods/lsh/lsh_search.hpp, 821
 src/mlpack/methods/mvu/CMakeLists.txt, 695
 src/mlpack/methods/mvu/mvu.hpp, 822
 src/mlpack/methods/naive_bayes/CMakeLists.txt, 696
 src/mlpack/methods/naive_bayes/naive_bayes_classifier.↵
 hpp, 823
 src/mlpack/methods/nca/CMakeLists.txt, 696
 src/mlpack/methods/nca/nca.hpp, 823
 src/mlpack/methods/nca/nca_softmax_error_function.hpp,
 824
 src/mlpack/methods/neighbor_search/CMakeLists.txt, 697
 src/mlpack/methods/neighbor_search/neighbor_search.↵
 hpp, 825
 src/mlpack/methods/neighbor_search/neighbor_search↵
 _rules.hpp, 826
 src/mlpack/methods/neighbor_search/neighbor_search↵
 _stat.hpp, 827
 src/mlpack/methods/neighbor_search/ns_traversal_info.↵
 hpp, 828
 src/mlpack/methods/neighbor_search/sort_policies/furthest↵
 _neighbor_sort.hpp, 829
 src/mlpack/methods/neighbor_search/sort_policies/nearest↵
 _neighbor_sort.hpp, 830
 src/mlpack/methods/neighbor_search/typedef.hpp, 831
 src/mlpack/methods/neighbor_search/unmap.hpp, 833
 src/mlpack/methods/nmf/CMakeLists.txt, 697
 src/mlpack/methods/nystroem_method/CMakeLists.txt,
 697
 src/mlpack/methods/nystroem_method/kmeans_↵
 selection.hpp, 834
 src/mlpack/methods/nystroem_method/nystroem_↵
 method.hpp, 810
 src/mlpack/methods/nystroem_method/ordered_↵
 selection.hpp, 835
 src/mlpack/methods/nystroem_method/random_selection.↵
 hpp, 836
 src/mlpack/methods/pca/CMakeLists.txt, 698
 src/mlpack/methods/pca/pca.hpp, 836
 src/mlpack/methods/perceptron/CMakeLists.txt, 698
 src/mlpack/methods/perceptron/initialization_methods/↵
 CMakeLists.txt, 698
 src/mlpack/methods/perceptron/initialization_methods/random↵
 _init.hpp, 779
 src/mlpack/methods/perceptron/initialization_methods/zero↵
 _init.hpp, 837
 src/mlpack/methods/perceptron/learning_policies/C↵
 MakeLists.txt, 699
 src/mlpack/methods/perceptron/learning_policies/simple↵
 _weight_update.hpp, 838
 src/mlpack/methods/perceptron/perceptron.hpp, 839
 src/mlpack/methods/quic_svd/CMakeLists.txt, 699
 src/mlpack/methods/quic_svd/quic_svd.hpp, 840
 src/mlpack/methods/radical/CMakeLists.txt, 700
 src/mlpack/methods/radical/radical.hpp, 841

- src/mlpack/methods/range_search/CMakeLists.txt, 700
- src/mlpack/methods/range_search/range_search.hpp, 841
- src/mlpack/methods/range_search/range_search_rules.↵
hpp, 842
- src/mlpack/methods/range_search/range_search_stat.↵
hpp, 843
- src/mlpack/methods/rann/CMakeLists.txt, 700
- src/mlpack/methods/rann/ra_query_stat.hpp, 844
- src/mlpack/methods/rann/ra_search.hpp, 845
- src/mlpack/methods/rann/ra_search_rules.hpp, 846
- src/mlpack/methods/rann/ra_typedef.hpp, 847
- src/mlpack/methods/regularized_svd/CMakeLists.txt, 701
- src/mlpack/methods/regularized_svd/regularized_svd.↵
hpp, 849
- src/mlpack/methods/regularized_svd/regularized_svd_↵
function.hpp, 849
- src/mlpack/methods/sparse_autoencoder/CMakeLists.txt, 701
- src/mlpack/methods/sparse_autoencoder/sparse_↵
autoencoder.hpp, 850
- src/mlpack/methods/sparse_autoencoder/sparse_↵
autoencoder_function.hpp, 851
- src/mlpack/methods/sparse_coding/CMakeLists.txt, 702
- src/mlpack/methods/sparse_coding/data_dependent_↵
random_initializer.hpp, 852
- src/mlpack/methods/sparse_coding/nothing_initializer.↵
hpp, 853
- src/mlpack/methods/sparse_coding/random_initializer.↵
hpp, 854
- src/mlpack/methods/sparse_coding/sparse_coding.hpp, 855
- src/mlpack/prereqs.hpp, 856
- src/mlpack/tests/CMakeLists.txt, 702
- src/mlpack/tests/data/data_3d_ind.txt, 857
- src/mlpack/tests/data/data_3d_mixed.txt, 857
- src/mlpack/tests/data/iris.txt, 857
- src/mlpack/tests/data/iris_labels.txt, 857
- src/mlpack/tests/old_boost_test_definitions.hpp, 857
- Start
 - mlpack::Timer, 563
 - mlpack::det::DTree, 211
- start
 - mlpack::det::DTree, 213
- StartTimer
 - mlpack::Timers, 566
- Stat
 - mlpack::tree::BinarySpaceTree, 583
 - mlpack::tree::CoverTree, 618
 - mlpack::tree::ExampleTree, 637
- stat
 - mlpack::tree::BinarySpaceTree, 586
 - mlpack::tree::CoverTree, 621
 - mlpack::tree::ExampleTree, 637
- StepSize
 - mlpack::optimization::SGD, 489
- stepSize
 - mlpack::optimization::SGD, 490
- Stop
 - mlpack::Timer, 563
- StopTimer
 - mlpack::Timers, 566
- stretchedDataset
 - mlpack::nca::SoftmaxErrorFunction, 384
- SubtreeLeaves
 - mlpack::det::DTree, 211
- subtreeLeaves
 - mlpack::det::DTree, 214
- SubtreeLeavesLogNegError
 - mlpack::det::DTree, 211
- subtreeLeavesLogNegError
 - mlpack::det::DTree, 214
- SuccessProbability
 - mlpack::neighbor::RASearchRules, 426
- sumOfSquaredNorms
 - mlpack::tree::MRKDStatistic, 645
- supported
 - TREE_EXPLANATION.txt, 756
- Sweeps
 - mlpack::radical::Radical, 510
- sweeps
 - mlpack::radical::Radical, 511
- t_policy
 - mlpack::amf::CompleteIncrementalTermination, 130
 - mlpack::amf::IncompleteIncrementalTermination, 132
- TREE_EXPLANATION.txt
 - bottom, 756
 - separately, 756
 - supported, 756
 - top, 757
 - Tree, 757
- TYPENAME
 - cli.hpp, 766
- TagTree
 - mlpack::det::DTree, 211
- Temperature
 - mlpack::optimization::SA, 483
- temperature
 - mlpack::optimization::SA, 485
- TerminationPolicy
 - mlpack::amf::AMF, 125, 126
- terminationPolicy
 - mlpack::amf::AMF, 126
- test_points
 - mlpack::amf::ValidationRMSETermination, 163
- tests/CMakeLists.txt
 - add_custom_command, 702

- add_executable, 702
- Thus
 - det.txt, 678
- Timer
 - mlpack::CLI, 195
- timer
 - mlpack::CLI, 196
- Timers
 - mlpack::Timers, 564
- timers
 - mlpack::Timers, 566
- tname
 - mlpack::ParamData, 498
- ToString
 - mlpack::bound::BallBound, 169
 - mlpack::bound::HRectBound, 176
 - mlpack::cf::CF, 182
 - mlpack::det::DTree, 211
 - mlpack::distribution::DiscreteDistribution, 218
 - mlpack::distribution::GaussianDistribution, 221
 - mlpack::distribution::LaplaceDistribution, 225
 - mlpack::emst::DualTreeBoruvka, 240
 - mlpack::fastmks::FastMKS, 253
 - mlpack::gmm::GMM, 281
 - mlpack::hmm::HMM, 291
 - mlpack::kernel::CosineDistance, 294
 - mlpack::kernel::EpanechnikovKernel, 296
 - mlpack::kernel::ExampleKernel, 299
 - mlpack::kernel::GaussianKernel, 303
 - mlpack::kernel::HyperbolicTangentKernel, 306
 - mlpack::kernel::LaplacianKernel, 315
 - mlpack::kernel::LinearKernel, 316
 - mlpack::kernel::PSpectrumStringKernel, 324
 - mlpack::kernel::PolynomialKernel, 322
 - mlpack::kernel::SphericalKernel, 328
 - mlpack::kernel::TriangularKernel, 332
 - mlpack::kmeans::KMeans, 339
 - mlpack::kpca::KernelPCA, 349
 - mlpack::lcc::LocalCoordinateCoding, 356
 - mlpack::math::Range, 364
 - mlpack::metric::IPMetric, 366
 - mlpack::metric::LMetric, 368
 - mlpack::metric::MahalanobisDistance, 371
 - mlpack::nca::NCA, 379
 - mlpack::nca::SoftmaxErrorFunction, 383
 - mlpack::neighbor::LSHSearch, 393
 - mlpack::neighbor::NeighborSearch, 403
 - mlpack::optimization::AugLagrangian, 445
 - mlpack::optimization::AugLagrangianFunction, 449
 - mlpack::optimization::AugLagrangianTestFunction, 451
 - mlpack::optimization::L_BFGS, 463
 - mlpack::optimization::LRSDP, 471
 - mlpack::optimization::LRSDPFunction, 475
 - mlpack::optimization::SA, 483
 - mlpack::optimization::SGD, 490
 - mlpack::pca::PCA, 501
 - mlpack::radical::Radical, 510
 - mlpack::range::RangeSearch, 516
 - mlpack::regression::LARS, 530
 - mlpack::regression::LinearRegression, 536
 - mlpack::regression::LogisticRegression, 540
 - mlpack::sparse_coding::SparseCoding, 552
 - mlpack::tree::BinarySpaceTree, 583
 - mlpack::tree::CoverTree, 619
 - mlpack::tree::EmptyStatistic, 630
 - mlpack::tree::MRKDStatistic, 643
 - RASearch, 671
- Tolerance
 - mlpack::amf::SimpleToleranceTermination, 145, 146
 - mlpack::gmm::EMFit, 270
 - mlpack::hmm::HMM, 290, 291
 - mlpack::optimization::SA, 483
 - mlpack::optimization::SGD, 489, 490
- tolerance
 - mlpack::amf::SimpleToleranceTermination, 148
 - mlpack::amf::ValidationRMSETermination, 163
 - mlpack::gmm::EMFit, 271
 - mlpack::hmm::HMM, 293
 - mlpack::optimization::SA, 485
 - mlpack::optimization::SGD, 490
 - mlpack::regression::LARS, 531
- top
 - TREE_EXPLANATION.txt, 757
- totalDist
 - mlpack::emst::DualTreeBoruvka, 242
- Train
 - mlpack::decision_stump::DecisionStump, 200
 - mlpack::hmm::HMM, 291
 - mlpack::perceptron::Perceptron, 503
- trainData
 - mlpack::perceptron::Perceptron, 503
- TrainOnAtt
 - mlpack::decision_stump::DecisionStump, 201
- Trainer
 - mlpack::det, 96
- Transition
 - mlpack::hmm::HMM, 292
- transition
 - mlpack::hmm::HMM, 293
- TraversalInfo
 - LastBaseCase, 675
 - lastBaseCase, 676
 - LastQueryNode, 675
 - lastQueryNode, 676
 - LastReferenceNode, 675
 - lastReferenceNode, 676
 - LastScore, 676

- lastScore, 676
- mlpack::emst::DTBRules, 230
- mlpack::fastmks::FastMKSRules, 258
- mlpack::neighbor::NeighborSearchRules, 410
- mlpack::neighbor::RASearchRules, 428
- mlpack::range::RangeSearchRules, 521, 522
- TraversallInfo, 675
- traversallInfo
 - mlpack::emst::DTBRules, 231
 - mlpack::fastmks::FastMKSRules, 260
 - mlpack::neighbor::NeighborSearchRules, 412
 - mlpack::neighbor::RASearchRules, 430
 - mlpack::range::RangeSearchRules, 523
 - mlpack::tree::BinarySpaceTree::DualTreeTraverser, 590
 - mlpack::tree::CoverTree::DualTreeTraverser::Dual←CoverTreeMapEntry, 627
- TraversallInfo< TreeType >, 673
- TraversallInfoType
 - mlpack::emst::DTBRules, 227
 - mlpack::fastmks::FastMKSRules, 256
 - mlpack::neighbor::NeighborSearchRules, 407
 - mlpack::neighbor::RASearchRules, 421
 - mlpack::range::RangeSearchRules, 520
- Traverse
 - mlpack::tree::BinarySpaceTree::DualTreeTraverser, 589
 - mlpack::tree::BinarySpaceTree::SingleTreeTraverser, 591
 - mlpack::tree::CoverTree::DualTreeTraverser, 623, 625
 - mlpack::tree::CoverTree::SingleTreeTraverser, 628
- Tree
 - TREE_EXPLANATION.txt, 757
- tree
 - mlpack::emst::DualTreeBoruvka, 242
- TreeDepth
 - mlpack::tree::BinarySpaceTree, 583
- treeOwner
 - mlpack::fastmks::FastMKS, 254
 - mlpack::neighbor::NeighborSearch, 405
 - mlpack::range::RangeSearch, 518
 - RASearch, 673
- TreeSize
 - mlpack::tree::BinarySpaceTree, 583
- TriangularKernel
 - mlpack::kernel::TriangularKernel, 330
- u
 - mlpack::amf::SVDBatchLearning, 151
 - mlpack::amf::SVDCompleteIncrementalLearning, 154
 - mlpack::amf::SVDCompleteIncrementalLearning< arma::sp_mat >, 157
 - mlpack::amf::SVDIncompleteIncrementalLearning, 159
- Unfold
 - mlpack::mvu::MVU, 373
- Union
 - mlpack::emst::UnionFind, 246
- UnionFind
 - mlpack::emst::UnionFind, 246
- Unmap
 - mlpack::neighbor, 112, 113
- Update
 - mlpack::amf::AMF, 126
- update
 - mlpack::amf::AMF, 127
- UpdateBasisSet
 - mlpack::optimization::L_BFGS, 463
- UpdateGmap
 - mlpack::CLI, 194
- UpdateWeights
 - mlpack::perceptron::SimpleWeightUpdate, 505
- useCholesky
 - mlpack::regression::LARS, 532
- V
 - mlpack::amf::SimpleToleranceTermination, 148
- ValidationRMSETermination
 - mlpack::amf::ValidationRMSETermination, 160
- value
 - IsVector, 119
 - IsVector< arma::Col< eT > >, 120
 - IsVector< arma::Row< eT > >, 120
 - IsVector< arma::SpCol< eT > >, 121
 - IsVector< arma::SpRow< eT > >, 121
 - IsVector< arma::SpSubview< eT > >, 122
 - IsVector< arma::subview_col< eT > >, 122
 - IsVector< arma::subview_row< eT > >, 123
 - mlpack::ParamData, 498
- Variances
 - mlpack::naive_bayes::NaiveBayesClassifier, 375
- variances
 - mlpack::naive_bayes::NaiveBayesClassifier, 376
- Vasicek
 - mlpack::radical::Radical, 510
- Vec
 - mlpack::bound::BallBound, 165
- VectorIndices
 - mlpack::tree::CosineTree, 600
- VectorPower
 - mlpack::math, 108
- vertices
 - mlpack::optimization::LovaszThetaSDP, 467
- VisibleSize
 - mlpack::nn::SparseAutoencoder, 434
 - mlpack::nn::SparseAutoencoderFunction, 439

- visibleSize
 - mlpack::nn::SparseAutoencoder, 435
 - mlpack::nn::SparseAutoencoderFunction, 440
- vmap
 - mlpack::CLI, 196
- W
 - mlpack::amf::SimpleToleranceTermination, 148
 - mlpack::amf::ValidationRMSETermination, 163
 - mlpack::cf::CF, 182
- w
 - mlpack::cf::CF, 183
- WUpdate
 - mlpack::amf::NMFALSUpdate, 134
 - mlpack::amf::NMFMultiplicativeDistanceUpdate, 135
 - mlpack::amf::NMFMultiplicativeDivergenceUpdate, 137
 - mlpack::amf::SVDBatchLearning, 150
 - mlpack::amf::SVDCompleteIncrementalLearning, 152
 - mlpack::amf::SVDCompleteIncrementalLearning< arma::sp_mat >, 156
 - mlpack::amf::SVDIncompleteIncrementalLearning, 158
- Warn
 - mlpack::Log, 359
- wasPassed
 - mlpack::ParamData, 498
- weightVectors
 - mlpack::perceptron::Perceptron, 504
- Weights
 - mlpack::gmm::GMM, 281
- weights
 - mlpack::gmm::GMM, 282
- wf
 - mlpack::optimization::test::RosenbrockWood↵Function, 494
- Whitelist
 - mlpack::tree::MRKDStatistic, 644
- whitelist
 - mlpack::tree::MRKDStatistic, 645
- WhitenFeatureMajorMatrix
 - mlpack::radical, 115
- WhitenUsingEig
 - mlpack::math, 108
- WhitenUsingSVD
 - mlpack::math, 108
- Width
 - mlpack::math::Range, 364
- WithinRange
 - mlpack::det::DTree, 211
- Wolfe
 - mlpack::optimization::L_BFGS, 463
- wolfe
 - mlpack::optimization::L_BFGS, 465
- WoodFunction
 - mlpack::optimization::test::WoodFunction, 496
- WorstDistance
 - mlpack::neighbor::FurthestNeighborSort, 388
 - mlpack::neighbor::NearestNeighborSort, 398
- WriteFile
 - mlpack::util::SaveRestoreUtility, 665
- WriteTree
 - mlpack::det::DTree, 211
- y
 - mlpack::optimization::L_BFGS, 465
- Zeroinitialization
 - mlpack::perceptron::Zeroinitialization, 506