Reference Manual

Generated by Doxygen 1.3.9.1

Tue Sep 25 17:20:14 2007

Contents

Chapter 1

Class Index

1.1 Class List

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

Chapter 2

File Index

2.1 File List

Here is a list of all documented files with brief descriptions:

Chapter 3

Class Documentation

3.1 RBF Class Reference

This class provides different functions for the approximation using an RBF model.

`#include <rbf.h>`

Public Member Functions

- **RBF** ()

  constructor

- **RBF** (unsigned inDim, kernelType aKernel, Array< double > kernelParams)

  This constructor generates an Interpolation RBF model.

- **RBF** (unsigned inDim, kernelType aKernel, Array< double > kernelParams, unsigned nKernel, double regParam, double maxErr, errMethod anErrMethod, unsigned maxKMeanIter, unsigned maxRegIter)

  This constructor generates a Ridge Regression RBF model.

- **RBF** (unsigned inDim, kernelType aKernel, Array< double > kernelParams, unsigned maximumBasis, double maxErr)

  This constructor generates an Orthogonal Least Square Forward Selection RBF model.

- ∼**RBF** ()

  destructor

- void **evaluate** (Individual &offspring)

  This function evaluates the individual <offspring> with the RBF model and stores the result as the fitness value of the offspring.

- void **evaluate** (Population &offsprings)

  This function evaluates the population <offsprings> with the RBF model and stores the results as the fitness value of each member of offsprings.

- void **evaluate** (Array< double > inputData, Array< double > &outputData)

This function evaluates the data in <inputdata> with the RBF model and stores the results in <outputdata>.

- void **Evaluate** (Array< double > inputData, Array< double > &outputData)
- double **mse** (Array< double > Input, Array< double > Target)

  This function calculates the mean square error of the approximation considering the given arrays <input> and <target>.

- double **mse2** (Array< double > Input, Array< double > Target)
- double **mse** (Population offsprings)

  This function calculates the mean square error of the approximation considering a given population <offsprings>.

- void **train** (Array< double > InputData, Array< double > TargetData)

  This function approximates the datas which are stored in the arrays <inputdata> and <targetdata> using the RBF model.

- void **setDesignMatrix** (Array< double > &inputData)

  This function set the design(hidden) matrix of the RBF model based on <inputdata> provided.

- void **setDesignMatrix2** (Array< double > &inputData)
- void **setWeightMatrix** (Array< double > &targetData)

  This function set the weight matrix of the RBF model based on <targetdata> provided.

- void **setWeightMatrix2** (Array< double > &targetData)
- Array< double > **getDesignMatrix** ()

  This function returns the design(hidden) matrix of the RBF model.

- Array< double > **getWeightMatrix** ()

  This function returns the weight matrix of the RBF model.

- Array< double > **getOrthogonalMatrix** ()

  This function returns the orthogonal matrix of the OLSForward (Orthogonal Least Square - Forward Selection) RBF model.

- Array< double > **getOrthogonalWeightMatrix** ()

  This function returns the orthogonal weight matrix of the OLSForward (Orthogonal Least Square - Forward Selection) RBF model.

- Array< double > **getBaseCentres** ()

  This function returns the kernel function centres of the RBF model.

Static Public Member Functions

- double **getDistance** (Array< double > &input, unsigned i, Array< double > &centre, unsigned j)

  This function calculates the distance between -th <input> and j-th <centre>.

- double **gaussFunc** (Array< double > input, unsigned i, Array< double > centre, unsigned j, Array< double > params)

This function calculates the gaussian kernel output between -th <input> and j-th <centre>.

- double **cubicFunc** (Array< double > input, unsigned i, Array< double > centre, unsigned j, Array< double > params)

  This function calculates the cubic spline kernel output between -th <input> and j-th <centre>.

- double **linearFunc** (Array< double > input, unsigned i, Array< double > centre, unsigned j, Array< double > params)

  This function calculates the linear spline kernel output between -th <input> and j-th <centre>.

- double **multiquadricFunc** (Array< double > input, unsigned i, Array< double > centre, unsigned j, Array< double > params)

  This function calculates the multiquadrics kernel output between -th <input> and j-th <centre>.

- double **inverseMultiquadricFunc** (Array< double > input, unsigned i, Array< double > centre, unsigned j, Array< double > params)

  This function calculates the inverse multiquadrics kernel output between -th <input> and j-th <centre>.

- double **cauchyFunc** (Array< double > input, unsigned i, Array< double > centre, unsigned j, Array< double > params)

  This function calculates the cauchy kernel output between -th <input> and j-th <centre>.

- double **YPYFunc** (Array< double > desMatrix, Array< double > output, double regParam, Array< double > &A)

  This function is a helper function to calculate sum squared error analytically. It is used in the regularization parameter optimization.

- double **GCVFunc** (Array< double > desMatrix, Array< double > output, double regParam)

  This function calculates the Generalized Cross-Validation error.

- double **UEVFunc** (Array< double > desMatrix, Array< double > output, double regParam)

  This function calculates the Unbiased Estimate Variance.

- double **FPEFunc** (Array< double > desMatrix, Array< double > output, double regParam)

  This function calculates the Final Prediction Error.

- double **BICFunc** (Array< double > desMatrix, Array< double > output, double regParam)

  This function calculates the Bayesian Information Criterion error.

- double **estRegParamGCV** (double regParam, Array< double > weightMatrix, Array< double > designMatrix, Array< double > predicted)

  This function estimates the new regularization parameter based on the Generalized Cross-Validation error.

- double **estRegParamUEV** (double regParam, Array< double > weightMatrix, Array< double > designMatrix, Array< double > predicted)

This function estimates the new regularization parameter based on the Unbiased Estimate Variance.

- double **estRegParamFPE** (double regParam, Array< double > weightMatrix, Array< double > designMatrix, Array< double > predicted)

  This function estimates the new regularization parameter based on the Final Prediction Error.

- double **estRegParamBIC** (double regParam, Array< double > weightMatrix, Array< double > designMatrix, Array< double > predicted)

  This function estimates the new regularization parameter based on the Bayesian Information Criterion.

- void **KMean** (unsigned nCluster, unsigned maxIter, Array< double > input, Array< double > &centresFound, Array< unsigned > &clustMap)

  This function performs the K-Mean clustering algorithm to provide <ncluster> clusters based on a set of inputs <input>.

- template<class Type> void **initArrayToZero** (Array< Type > &arr)

  This function initializes an array to zero.

3.1.1 Detailed Description

This class provides different functions for the approximation using an RBF model.

This class is based on the RBF approximation models. For using this class in an appropriate way it is required to create an instance of the class <database> or arrays of inputData and targetData. After the known data is stored in the database or the arrays, the functionality of the approximation can be applied. Therefore in the next step the model should be trained and after this the evaluation functions can be used.

3.1.2 Constructor & Destructor Documentation

3.1.2.1 RBF::RBF (unsigned *inDim*, kernelType *aKernel*, Array< double > *kernelParams*)

This constructor generates an Interpolation RBF model.

Parameters:

inDim dimensionality of the problem.

aKernel kernel function used.

kernelParams parameters for the kernel function.

3.1.2.2 RBF::RBF (unsigned *inDim*, kernelType *aKernel*, Array< double > *kernelParams*, unsigned *nKernel*, double *regParam*, double *maxErr*, errMethod *anErrMethod*, unsigned *maxKMeanIter*, unsigned *maxRegIter*)

This constructor generates a Ridge Regression RBF model.

Parameters:

inDim dimensionality of the problem.

aKernel kernel function used.

kernelParams parameters for the kernel function.

nKernel number of kernel function centres used.

regParam starting value of the regularization parameter.

maxErr maximum training MSE allowed.

anErrMethod error calculation method used in optimizing the regularization parameter.

maxKMeanIter maximum K-Mean clustering iteration to determine the centres of kernel functions.

maxRegIter maximum regularization parameter optimization iteration.

3.1.2.3 RBF::RBF (unsigned *inDim*, kernelType *aKernel*, Array< double > *kernelParams*, unsigned *maximumBasis*, double *maxErr*)

This constructor generates an Orthogonal Least Square Forward Selection RBF model.

Parameters:

inDim dimensionality of the problem.

aKernel kernel function used.

kernelParams parameters for the kernel function.

maximumBasis maximum number of kernel functions.

maxErr maximum training MSE allowed.

3.1.3 Member Function Documentation

3.1.3.1 double RBF::BICFunc (Array< double > *desMatrix*, Array< double > *output*, double *regParam*) `[static]`

This function calculates the Bayesian Information Criterion error.

Parameters:

desMatrix array containing the design matrix.

output array containing the desired output value.

regParam regularization parameter.

Return values:

BICError BIC error.

3.1.3.2 double RBF::cauchyFunc (Array< double > *input*, unsigned *i*, Array< double > *centre*, unsigned *j*, Array< double > *params*) `[static]`

This function calculates the cauchy kernel output between *-th <input> and j-th <centre>.*

Parameters:

input array containing the input which are used for training.

i index of the input which distance to be calculated.

centre array containing the centres of the RBF model.

j index of the centre to which the distance from input is to be calculated.

params array containing parameter(s) for the cauchy kernel function.

Return values:

kernelOutput value of kernel output.

3.1.3.3 double RBF::cubicFunc (Array< double > *input*, unsigned *i*, Array< double > *centre*, unsigned *j*, Array< double > *params*) `[static]`

This function calculates the cubic spline kernel output between *-th <input> and j-th <centre>.*

Parameters:

input array containing the input which are used for training.

i index of the input which distance to be calculated.

centre array containing the centres of the RBF model.

j index of the centre to which the distance from input is to be calculated.

params array containing parameter(s) for the cubic spline kernel function.

Return values:

kernelOutput value of kernel output.

3.1.3.4 double RBF::estRegParamBIC (double *regParam*, Array< double > *weightMatrix*, Array< double > *designMatrix*, Array< double > *predicted*) `[static]`

This function estimates the new regularization parameter based on the Bayesian Information Criterion.

Parameters:

regParam regularization parameter.

weightMatrix array containing the weight matrix.

designMatrix array containing the design matrix.

predicted array containing the predicted output.

Return values:

newRegParam new regularization parameter.

3.1.3.5 double RBF::estRegParamFPE (double *regParam*, Array< double > *weightMatrix*, Array< double > *designMatrix*, Array< double > *predicted*) `[static]`

This function estimates the new regularization parameter based on the Final Prediction Error.

Parameters:

regParam regularization parameter.

weightMatrix array containing the weight matrix.

designMatrix array containing the design matrix.

predicted array containing the predicted output.

Return values:

newRegParam new regularization parameter.

3.1.3.6 double RBF::estRegParamGCV (double *regParam*, Array< double > *weightMatrix*, Array< double > *designMatrix*, Array< double > *predicted*) `[static]`

This function estimates the new regularization parameter based on the Generalized Cross-Validation error.

Parameters:

regParam regularization parameter.

weightMatrix array containing the weight matrix.

designMatrix array containing the design matrix.

predicted array containing the predicted output.

Return values:

newRegParam new regularization parameter.

3.1.3.7 double RBF::estRegParamUEV (double *regParam*, Array< double > *weightMatrix*, Array< double > *designMatrix*, Array< double > *predicted*) `[static]`

This function estimates the new regularization parameter based on the Unbiased Estimate Variance.

Parameters:

regParam regularization parameter.

weightMatrix array containing the weight matrix.

designMatrix array containing the design matrix.

predicted array containing the predicted output.

Return values:

newRegParam new regularization parameter.

3.1.3.8 void RBF::evaluate (Array< double > *inputData*, Array< double > & *outputData*)

This function evaluates the data in <inputdata> with the RBF model and stores the results in <outputdata>.

Parameters:

inputData array containing the input parameters.

outputData reference to array containing the evaluation results.

3.1.3.9 void RBF::evaluate (Population & *offsprings*)

This function evaluates the population <offsprings> with the RBF model and stores the results as the fitness value of each member of offsprings.

Parameters:

offsprings population of offspring to be evaluated.

3.1.3.10 void RBF::evaluate (Individual & *offspring*)

This function evaluates the individual <offspring> with the RBF model and stores the result as the fitness value of the offspring.

Parameters:

offspring individual of offspring to be evaluated.

3.1.3.11 double RBF::FPEFunc (Array< double > *desMatrix*, Array< double > *output*, double *regParam*) `[static]`

This function calculates the Final Prediction Error.

Parameters:

desMatrix array containing the design matrix.

output array containing the desired output value.

regParam regularization parameter.

Return values:

FPEError FPE error.

3.1.3.12 double RBF::gaussFunc (Array< double > *input*, unsigned *i*, Array< double > *centre*, unsigned *j*, Array< double > *params*) `[static]`

This function calculates the gaussian kernel output between *-th <input> and j-th <centre>.*

Parameters:

input array containing the input which are used for training.

i index of the input which distance to be calculated.

centre array containing the centres of the RBF model.

j index of the centre to which the distance from input is to be calculated.

params array containing parameter(s) for the gaussian kernel function.

Return values:

kernelOutput value of kernel output.

3.1.3.13 double RBF::GCVFunc (Array< double > *desMatrix*, Array< double > *output*, double *regParam*) [static]

This function calculates the Generalized Cross-Validation error.

Parameters:

desMatrix array containing the design matrix.

output array containing the desired output value.

regParam regularization parameter.

Return values:

GCVError GCV error.

3.1.3.14 Array<double> RBF::getBaseCentres ()

This function returns the kernel function centres of the RBF model.

Return values:

baseCentres array containing the <basecentres> of the RBF model.

3.1.3.15 Array<double> RBF::getDesignMatrix ()

This function returns the design(hidden) matrix of the RBF model.

Return values:

designMatrix array containing the <designmatrix> of the RBF model.

3.1.3.16 double RBF::getDistance (Array< double > & *input*, unsigned *i*, Array< double > & *centre*, unsigned *j*) [static]

This function calculates the distance between *-th <input> and j-th <centre>.*

Parameters:

input array containing the input which are used for training.

i index of the input which distance to be calculated.

centre array containing the centres of the RBF model.

j index of the centre to which the distance from input is to be calculated.

Return values:

distance value of distance.

3.1.3.17 Array<double> RBF::getOrthogonalMatrix ()

This function returns the orthogonal matrix of the OLSForward (Orthogonal Least Square - Forward Selection) RBF model.

Return values:

orthogonalMatrix array containing the <orthogonalmatrix> of the RBF model.

3.1.3.18 Array<double> RBF::getOrthogonalWeightMatrix ()

This function returns the orthogonal weight matrix of the OLSForward (Orthogonal Least Square - Forward Selection) RBF model.

Return values:

orthogonalWeightMatrix array containing the <orthogonalweightmatrix> of the RBF model.

3.1.3.19 Array<double> RBF::getWeightMatrix ()

This function returns the weight matrix of the RBF model.

Return values:

weightMatrix array containing the <weightmatrix> of the RBF model.

3.1.3.20 template<class Type> void RBF::initArrayToZero (Array< Type > & *arr*) `[inline, static]`

This function initializes an array to zero.

Parameters:

arr array to be initialized to zero.

3.1.3.21 double RBF::inverseMultiquadricFunc (Array< double > *input*, unsigned *i*, Array< double > *centre*, unsigned *j*, Array< double > *params*) `[static]`

This function calculates the inverse multiquadrics kernel output between *-th <input> and j-th <centre>.*

Parameters:

input array containing the input which are used for training.

i index of the input which distance to be calculated.

centre array containing the centres of the RBF model.

j index of the centre to which the distance from input is to be calculated.

params array containing parameter(s) for the inverse multiquadrics kernel function.

Return values:

kernelOutput value of kernel output.

3.1.3.22 void RBF::KMean (unsigned *nCluster*, unsigned *maxIter*, Array< double > *input*, Array< double > & *centresFound*, Array< unsigned > & *clustMap*) `[static]`

This function performs the K-Mean clustering algorithm to provide <ncluster> clusters based on a set of inputs <input>.

Parameters:

nCluster number of desired clusters.

maxIter maximum iteration count.

input array containing the inputs.

centresFound reference to array containing the cluster centres found.

clustMap reference to array containing the pair of inputs and cluster numbers to which each of them are assigned.

3.1.3.23 double RBF::linearFunc (Array< double > *input*, unsigned *i*, Array< double > *centre*, unsigned *j*, Array< double > *params*) [static]

This function calculates the linear spline kernel output between *-th <input> and j-th <centre>.*

Parameters:

input array containing the input which are used for training.

i index of the input which distance to be calculated.

centre array containing the centres of the RBF model.

j index of the centre to which the distance from input is to be calculated.

params array containing parameter(s) for the linear spline kernel function.

Return values:

kernelOutput value of kernel output.

3.1.3.24 double RBF::mse (Population *offsprings*)

This function calculates the mean square error of the approximation considering a given population <offsprings>.

Parameters:

offsprings population containing the parameters in the first Chromosome and the fitness value as target value

Return values:

MSE mean square error of the approximation

3.1.3.25 double RBF::mse (Array< double > *Input*, Array< double > *Target*)

This function calculates the mean square error of the approximation considering the given arrays <input> and <target>.

Parameters:

Input array containing the input data.

Target array containing the target data.

Return values:

MSE mean square error of the approximation.

3.1.3.26 double RBF::multiquadricFunc (Array< double > *input*, unsigned *i*, Array< double > *centre*, unsigned *j*, Array< double > *params*) [static]

This function calculates the multiquadrics kernel output between *-th <input> and j-th <centre>.*

Parameters:

input array containing the input which are used for training.

i index of the input which distance to be calculated.

centre array containing the centres of the RBF model.

j index of the centre to which the distance from input is to be calculated.

params array containing parameter(s) for the multiquadrics kernel function.

Return values:

kernelOutput value of kernel output.

3.1.3.27 void RBF::setDesignMatrix (Array< double > & *inputData*)

This function set the design(hidden) matrix of the RBF model based on <inputdata> provided.

Parameters:

inputData array containing the inputdata which are used for approximation.

3.1.3.28 void RBF::setWeightMatrix (Array< double > & *targetData*)

This function set the weight matrix of the RBF model based on <targetdata> provided.

Parameters:

inputData array containing the inputdata which are used for approximation.

targetData array containing the target data used for approximation.

3.1.3.29 void RBF::train (Array< double > *InputData*, Array< double > *TargetData*)

This function approximates the datas which are stored in the arrays <inputdata> and <targetdata> using the RBF model.

Parameters:

InputData array containing the inputdata which are used for approximation.

TargetData array containing the targetdata which are used for approximation.

3.1.3.30 double RBF::UEVFunc (Array< double > *desMatrix*, Array< double > *output*, double *regParam*) [static]

This function calculates the Unbiased Estimate Variance.

Parameters:

desMatrix array containing the design matrix.

output array containing the desired output value.

regParam regularization parameter.

Return values:

UEVError UEV error.

3.1.3.31 double RBF::YPYFunc (Array< double > *desMatrix*, Array< double > *output*, double *regParam*, Array< double > & *A*) `[static]`

This function is a helper function to calculate sum squared error analytically. It is used in the regularization parameter optimization.

Parameters:

desMatrix array containing the design matrix.

output array containing the desired output value.

regParam regularization parameter.

A reference to array containing a helper matrix A.

Return values:

sumSquareError sum squared error

The documentation for this class was generated from the following file:

- **rbf.h**

Chapter 4

File Documentation

4.1 rbf.h File Reference

```
#include "EALib/Population.h"
#include "Array/ArrayOp.h"
```

Classes

- class **RBF**

  This class provides different functions for the approximation using an RBF model.

Typedefs

- typedef enum kernel **kernelType**
- typedef enum lm **learnMethod**
- typedef enum err **errMethod**

Enumerations

- enum **kernel** {
  gaussian = 0, **linear_spline**, **cubic_spline**, **multiquadric**,
  inverse_multiquadric, **cauchy** }
- enum **lm** { **interpolate** = 0, **ridgeRegression**, **OLS_ForwardSelection** }
- enum **err** { **GCV** = 0, **UEV**, **FPE**, **BIC** }

4.1.1 Detailed Description

Index