Reference Manual

Generated by Doxygen 1.3.9.1

Thu Oct 11 16:03:32 2007

Contents

1 Class Index 1

1.1 Class List 1

2 File Index 3

2.1 File List 3

3 Class Documentation 5

3.1 InterpolateRBF Class Reference 5

4 File Documentation 13

4.1 InterpolateRBF.h File Reference 13

Chapter 1

Class Index

1.1 Class List

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

InterpolateRBF (This class provides different functions for the approximation using an InterpolationRBF model) 5

Chapter 2

File Index

2.1 File List

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

InterpolateRBF.h 13

Chapter 3

Class Documentation

3.1 InterpolateRBF Class Reference

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

`#include <InterpolateRBF.h>`

Public Member Functions

- **InterpolateRBF** ()

  constructor

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

  This constructor generates an Interpolation RBF model.

- ~**InterpolateRBF** ()

  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>.

- 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 **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 **setWeightMatrix** (Array< double > &targetData)

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

- 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 > **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>*.*

- 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 InterpolationRBF model.

This class is based on the RBF approximation models. 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 InterpolateRBF::InterpolateRBF (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.

3.1.3 Member Function Documentation

3.1.3.1 double InterpolateRBF::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.

3.1.3.2 double InterpolateRBF::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.

3.1.3.3 void InterpolateRBF::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.

3.1.3.4 void InterpolateRBF::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.

3.1.3.5 void InterpolateRBF::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.

3.1.3.6 double InterpolateRBF::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.

3.1.3.7 Array<double> InterpolateRBF::getBaseCentres ()

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

Return values:

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

3.1.3.8 Array<double> InterpolateRBF::getDesignMatrix ()

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

Return values:

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

3.1.3.9 double InterpolateRBF::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.

3.1.3.10 Array<double> InterpolateRBF::getWeightMatrix ()

This function returns the weight matrix of the RBF model.

Return values:

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

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

This function initializes an array to zero.

Parameters:

arr array to be initialized to zero.

3.1.3.12 double InterpolateRBF::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.

3.1.3.13 void InterpolateRBF::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.

3.1.3.14 double InterpolateRBF::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.

3.1.3.15 double InterpolateRBF::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

3.1.3.16 double InterpolateRBF::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.

3.1.3.17 double InterpolateRBF::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.

3.1.3.18 void InterpolateRBF::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.

3.1.3.19 void InterpolateRBF::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.

3.1.3.20 void InterpolateRBF::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.

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

- **InterpolateRBF.h**

Chapter 4

File Documentation

4.1 InterpolateRBF.h File Reference

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

Classes

- class **InterpolateRBF**

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

Typedefs

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

Enumerations

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

4.1.1 Detailed Description

Index

- cauchyFunc
  - InterpolateRBF, 7
- cubicFunc
  - InterpolateRBF, 7

- evaluate
  - InterpolateRBF, 8

- gaussFunc
  - InterpolateRBF, 8
- getBaseCentres
  - InterpolateRBF, 9
- getDesignMatrix
  - InterpolateRBF, 9
- getDistance
  - InterpolateRBF, 9
- getWeightMatrix
  - InterpolateRBF, 9

- initArrayToZero
  - InterpolateRBF, 9
- InterpolateRBF, 5
  - InterpolateRBF, 7
- InterpolateRBF
  - cauchyFunc, 7
  - cubicFunc, 7
  - evaluate, 8
  - gaussFunc, 8
  - getBaseCentres, 9
  - getDesignMatrix, 9
  - getDistance, 9
  - getWeightMatrix, 9
  - initArrayToZero, 9
  - InterpolateRBF, 7
  - inverseMultiquadricFunc, 10
  - KMean, 10
  - linearFunc, 10
  - mse, 11
  - multiquadricFunc, 11
  - setDesignMatrix, 11
  - setWeightMatrix, 12
  - train, 12
- InterpolateRBF.h, 13
- inverseMultiquadricFunc
  - InterpolateRBF, 10

- KMean
  - InterpolateRBF, 10

- linearFunc
  - InterpolateRBF, 10

- mse
  - InterpolateRBF, 11
- multiquadricFunc
  - InterpolateRBF, 11

- setDesignMatrix
  - InterpolateRBF, 11
- setWeightMatrix
  - InterpolateRBF, 12

- train
  - InterpolateRBF, 12