Parallel Gaussian Process Regression

1.0.0

Generated by Doxygen 1.8.5

Tue Jul 29 2014 18:27:28

Contents

Chapter 1

Introduction

1. System requirements

Linux/Unix environment 64-bit processors GNU GCC (4.2.1 or above) MPICH 3.0.4 (http://www.mpich.org/); we also test in version 1.5

2. Compile

To compile all the applications, enter the command:

```
make all
```

All applications will be automatically generated in folder demo. In addition, it's also supported to compile each individual application. For example, to compile the application that prepares data (training data, test data etc.) for experiments

make prep

For the demonstration of different Gaussian process regression (GPR), you can use command

```
make fgp
```

to compile the application that demonstrates full Gaussian process regression;

```
make pitc
```

to compile the application that demonstrates PITC GP regression;

```
make ppitc
```

to compile the application that demonstrates parallel PITC GP regression;

```
make pic
```

to compile the application that demonstrates PIC GP regression;

```
make ppic
```

to compile the application that demonstrates PICF-based GP regression. To clean the compilation environment, use the command:

```
make clean
```

3. Demonstrations

A bash script can be used to run all applications, using command

```
cd demo && bash bat_demo.sh
```

Basically, the script first prepares all necessary files (training data, test data, support set, and hyperparameter file) for experiments; Then, different GPR algorithms are run sequentially and output the results (i.e., incurred time, root mean square error (RMSE) and mean negative log probability (MNLP)). For more information about the arguments of applications, please refer to the comments in the bash script.

4. Documentation

To compile the documentation, the documentation generation tool doxygen (http://www.doxygen.org) needs to be installed. Then, enter the home directory of our source code and run the command

```
make doc
```

You can refer to the html version documentation by

```
cd doc/html
```

and use any browser to open index.html; In addition, the pdf version can be accessed by

```
cd doc/latex && make
```

and use any pdf viewer to open refman.pdf.

Chapter 2

Todo List

Member pgpr_data::getRandomBlk ()

the training data is stored in m_sample_{i,j} where i is the i-th observation and j is the j-th bin. I am considering move m_sample out of member attribute

Member pgpr_data::getRandomWalk ()

the training data is stored in m_sample_{i,j} where i is the i-th observation and j is the j-th bin. I am considering move m_sample out of member attribute

Chapter 3

Class Index

3.1 Class List

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

Chapter 4

File Index

4.1 File List

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

Chapter 5

Class Documentation

5.1 pgpr_chol Class Reference

This class provides Cholesky factorization and some related useful functions such as inverse, log-determinant etc.

Public Member Functions

- pgpr_chol (Mdoub &A)

  The constructor does a Cholesky factorization.
- void solve (Vdoub &b, Vdoub &x)

  Solving A∗ x = b using back substitution.
- void **elmult** (Vdoub &y, Vdoub &b)
- void elsolve (Vdoub &b, Vdoub &y)

  solving el ∗ y = b using back substitution.
- void inverse (Mdoub &ainv)

  Compute the inverse of a matrix.
- Doub logdet ()

  Calculate the determinant of square matrix and retrun in log scale.

5.1.1 Detailed Description

This class provides Cholesky factorization and some related useful functions such as inverse, log-determinant etc.

5.1.2 Constructor & Destructor Documentation

5.1.2.1 pgpr_chol::pgpr_chol (**Mdoub** & *A*) [inline]

The constructor does a Cholesky factorization.

The lower triangular factor is stored in el, and upper triangular factor is stored in elt.

Parameters

a	the matrix that we want to do factorization

5.1.3 Member Function Documentation

5.1.3.1 void pgpr_chol::elsolve (Vdoub & *b,* Vdoub & *y*) [inline]

solving el ∗ y = b using back substitution.

b vector b = el ∗ y y vector y stores the result

5.1.3.2 void pgpr_chol::inverse (Mdoub & *ainv*) [inline]

Compute the inverse of a matrix.

Parameters

out	*ainv*	inverted matrix

5.1.3.3 Doub pgpr_chol::logdet ( ) [inline]

Calculate the determinant of square matrix and retrun in log scale.

Returns

Value of log-determinant

5.1.3.4 void pgpr_chol::solve (Vdoub & *b,* Vdoub & *x*) [inline]

Solving A∗ x = b using back substitution.

Parameters

b	vector b = A ∗ x
x	vector which would store the result.

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

- src/pgpr_chol.h

5.2 pgpr_cluster Class Reference

Provides some basic clustering algorithms used in the approximated GP algorithms.

`#include <pgpr_cluster.h>`

Public Member Functions

- **pgpr_cluster** (Int al)
- Int **pic_tset_blk** (Mdoub data[], Vint ds, Mdoub tset, Int ts, Mdoub ∗tset_blk, Vint &ts_blk, Vdoub ∗pmu_blk, Vdoub ∗pvar_blk)
- Int **test_clustering** (Mdoub data[], Vint ds, Mdoub aset, Int as, Mdoub tset, Int ts, Vdoub &pmu, Vdoub &pvar)

5.2.1 Detailed Description

Provides some basic clustering algorithms used in the approximated GP algorithms.

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

- src/pgpr_cluster.h

5.3 pgpr_cov Class Reference

The pgpr_cov class provides the informaiton of covariance.

`#include <pgpr_cov.h>`

Public Member Functions

- pgpr_cov (Char *hypf)

  this functions initialized the class with a hyperparameter file
- **pgpr_cov** (Vdoub h, Int d)
- Doub **se_ard_n** (Doub *x, Doub *y)
- Doub **se_ard** (Doub *x, Doub *y)
- void **se_ard_n** (Mdoub a, Int ss, Mdoub &k)
- void **se_ard** (Mdoub a, Int ss, Mdoub &k)
- void **se_ard** (Mdoub a, Int ssa, Mdoub b, Int ssb, Mdoub &k)

Public Attributes

- Doub nos
- Vdoub lsc
- Doub sig
- Doub mu
- Int dim

5.3.1 Detailed Description

The pgpr_cov class provides the informaiton of covariance.

5.3.2 Constructor & Destructor Documentation

5.3.2.1 pgpr_cov::pgpr_cov (Char * *hypf*) [inline]

this functions initialized the class with a hyperparameter file

5.3.3 Member Data Documentation

5.3.3.1 Int pgpr_cov::dim

the dimension of the features

5.3.3.2 Vdoub pgpr_cov::lsc

vector of the length scale for each dimension

5.3.3.3 Doub pgpr_cov::mu

the mean of the data

5.3.3.4 Doub pgpr_cov::nos

the noise variance of the data

5.3.3.5 Doub pgpr_cov::sig

the signal variance of the data

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

- src/pgpr_cov.h

5.4 pgpr_data Class Reference

This class provides functionalities of preparing train data, test set and support set.

`#include <pgpr_data.h>`

Public Member Functions

- pgpr_data (pgpr_parse cfg, Int an, Int tn)

  Constructor.
- Int getTestSet (Int tsize, Mdoub &test)

  Get a random subset of data for testing purpose.
- Int getTrainSet (Int mode)

  Get a set of training data.
- Int getRandomBlk ()

  Get a set of training data based on a simple clustering scheme.
- Int getRandomWalk ()

  Get a set of training data based on random walk. Note that, it is applicable only if the domain contains topology information.
- Int selMaxVar (Mdoub dset, Int dnum, Int anum, Mdoub &aset)

  Actively select an informative subset of points.

Public Attributes

- pgpr_domain * domain

  Pointer to a domain.
- Mint m_sample

  Matrix of indices used as training data.

5.4.1 Detailed Description

This class provides functionalities of preparing train data, test set and support set.

5.4.2 Constructor & Destructor Documentation

5.4.2.1 pgpr_data::pgpr_data (pgpr_parse *cfg,* Int *an,* Int *tn*) [inline]

Constructor.

Parameters

in	*cfg*	Configure of domain
in	*an*	Number of blocks
in	*tn*	Number of samples that are used for training

5.4.3 Member Function Documentation

5.4.3.1 Int pgpr_data::getRandomBlk ( ) [inline]

Get a set of training data based on a simple clustering scheme.

This function follows steps (see Definition 5, remark 2): 1) randomly select a set of K central points; 2) randomly select an unobserved point; 3) allocate it to the closest non-full bin; 4) loop until all bins are full.

Todo the training data is stored in m_sample_{i,j} where i is the i-th observation and j is the j-th bin. I am considering move m_sample out of member attribute

5.4.3.2 Int pgpr_data::getRandomWalk ( ) [inline]

Get a set of training data based on random walk. Note that, it is applicable only if the domain contains topology information.

Todo the training data is stored in m_sample_{i,j} where i is the i-th observation and j is the j-th bin. I am considering move m_sample out of member attribute

5.4.3.3 Int pgpr_data::getTestSet (Int *tsize,* Mdoub & *test*) [inline]

Get a random subset of data for testing purpose.

Parameters

in	*tsize*	Size of test set
out	*test*	Set of data for test; each row represents a test point.

5.4.3.4 Int pgpr_data::getTrainSet (Int *mode*) [inline]

Get a set of training data.

Parameters

in	*mode*	different selection algorithms: ALGO1 - Simple clustering algorithm; ALGO2 - Random walk algorithm

5.4.3.5 Int pgpr_data::selMaxVar (Mdoub *dset,* Int *dnum,* Int *anum,* Mdoub & *aset*) [inline]

Actively select an informative subset of points.

Parameters

in	*dset*	universal set of points
in	*dnum*	size of the universal set
in	*anum*	size of the subset to be selected
out	*aset*	the informative subset

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

- src/pgpr_data.h

5.5 pgpr_domain Class Reference

This class stores the domain data in a matrix manner.

`#include <pgpr_data.h>`

Public Member Functions

- pgpr_domain (Mint n, Mdoub a)

  Constructor.
- Doub getDist (Int ci, Int cj)

  Compute the Euclidean distance between a pair of inputs.
- Int selRand (t_state *st)

  Select randomly an index of a set, which is not selected.
- Int getAttr (Int s, Doub *a)

  Get the attributes of an input and its corresponding outputs.
- Int getNeighbour (Int s, Vint &c)

  Get a set of indices corresponding to the set of inputs which connected with a specific input.

Public Attributes

- Int dd

  dimension of the domain (dimension of inputs and outputs).
- Int ds

  size of the domain

5.5.1 Detailed Description

This class stores the domain data in a matrix manner.

5.5.2 Constructor & Destructor Documentation

5.5.2.1 pgpr_domain::pgpr_domain (Mint *n*, Mdoub *a*) [inline]

Constructor.

Parameters

in	*n*	Matrix representing the topology of domain inputs

in	*a*	Matrix containing all domain data, both inputs and outputs

5.5.3 Member Function Documentation

5.5.3.1 Int pgpr_domain::getAttr (Int *s*, Doub * *a*) [inline]

Get the attributes of an input and its corresponding outputs.

Parameters

in	*s*	index of an input
out	*a*	pointor to an array that stores the attributes

5.5.3.2 Doub pgpr_domain::getDist (Int *ci*, Int *cj*) [inline]

Compute the Euclidean distance between a pair of inputs.

Parameters

in	*ci*	index of an input
in	*cj*	index of an input

Returns

Value of Euclidean distance

5.5.3.3 Int pgpr_domain::getNeighbour (Int *s*, Vint & *c*) [inline]

Get a set of indices corresponding to the set of inputs which connected with a specific input.

Parameters

in	*s*	Index of a specific input
out	*c*	A list of indices corresponding to the connected inputs

Returns

Number of the connected inputs

5.5.3.4 Int pgpr_domain::selRand (t_state * *st*) [inline]

Select randomly an index of a set, which is not selected.

Parameters

in	*st*	the status indicating if an element is selected or not

Returns

the index of an input

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

- src/pgpr_data.h

5.6 pgpr_fgp Class Reference

This class provides basic regression function using full GP.

`#include <pgpr_fgp.h>`

Public Member Functions

- Int **full_reg** (Mdoub obs, Int dnum, Mdoub xt, Int ts, Vdoub &t_mu, Vdoub &t_var)
- **pgpr_fgp** (Vdoub h, Int dx)
- **pgpr_fgp** (Char *hypf)
- Int **regress** (Char *train, Char *test)
- void **outputRst** (Char *output)

5.6.1 Detailed Description

This class provides basic regression function using full GP.

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

- src/pgpr_fgp.h

5.7 pgpr_matrix< T > Class Template Reference

Matrix class.

Public Types

- typedef T **value_type**

Public Member Functions

- **pgpr_matrix** (int n, int m)
- **pgpr_matrix** (int n, int m, const T &a)
- **pgpr_matrix** (int n, int m, const T *a)
- **pgpr_matrix** (const pgpr_matrix &rhs)
- pgpr_matrix & **operator=** (const pgpr_matrix &rhs)
- T * **operator[]** (const int i)
- const T * **operator[]** (const int i) const
- int **nrows** () const
- int **ncols** () const
- void **resize** (int newn, int newm)
- void **assign** (int newn, int newm, const T &a)

5.7.1 Detailed Description

template<class T>class pgpr_matrix< T >

Matrix class.

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

- src/pgpr_type.h

5.8 pgpr_parse Class Reference

This class parses domain data files, configuration file and commandline of different applications.

`#include <pgpr_parse.h>`

Public Member Functions

- pgpr_parse (Int md, Int argc, Char *argv[])

  Construction function.

Public Attributes

- Vdoub v_param

  Vector of parameters loaded from cfg file.
- Vdoub v_hyp

  Vector of hyperparameters loaded from cfg file.
- Mdoub m_xyz

  Matrix storing domain data where no. of rows indicates the size of domain and no. of columns is the dimension of inputs and outputs.
- Int in_dim

  Dimension of inputs.
- Int out_dim

  Dimension of outputs.
- Mint m_connect

  Connectivity of the domain data: each row stores the indices of a set of data points that connects to the corresponding data points.
- t_command_demo param_demo

  Information extracted from the commandline of a demo application.
- t_command_prep param_prep

  Information extracted from the commandline of experimental data preparing application.

5.8.1 Detailed Description

This class parses domain data files, configuration file and commandline of different applications.

5.8.2 Constructor & Destructor Documentation

5.8.2.1 pgpr_parse::pgpr_parse (Int *md,* Int *argc,* Char * *argv[]*) [inline]

Construction function.

Parameters

in	*md*	Different applications: CFGDEMO - demo application; CFGPREP - data preparing application
in	*argc*	count of arguments
in	*argv*	an array of pointers to the list of arguments

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

- src/pgpr_parse.h

5.9 pgpr_pic Class Reference

This class provides the regression function using PIC Approximation.

`#include <pgpr_pic.h>`

Public Member Functions

- **pgpr_pic** (Char *hypf)
- Int **regress** (Char *train, Char *test, Char *support, Int blocks)
- void **outputRst** (Char *output)

5.9.1 Detailed Description

This class provides the regression function using PIC Approximation.

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

- src/pgpr_pic.h

5.10 pgpr_pitc Class Reference

This class provides the regression function using PITC Approximation.

`#include <pgpr_pitc.h>`

Public Member Functions

- **pgpr_pitc** (Char *hypf)
- Int **regress** (Char *train, Char *test, Char *support, Int blocks)
- void **outputRst** (Char *output)

5.10.1 Detailed Description

This class provides the regression function using PITC Approximation.

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

- src/pgpr_pitc.h

5.11 pgpr_ppic Class Reference

This class provides the regression function using PIC Approximation, but implemented in a paralle manner.

`#include <pgpr_ppic.h>`

Public Member Functions

- **pgpr_ppic** (Char *hypf)
- Int **regress** (Char *train, Char *test, Char *support, Int blocks)
- void **outputRst** (Char *output)

5.11.1 Detailed Description

This class provides the regression function using PIC Approximation, but implemented in a paralle manner.

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

- src/pgpr_ppic.h

5.12 pgpr_ppitc Class Reference

This class provides the regression function using PITC Approximation,implemented in a parallel manner.

`#include <pgpr_ppitc.h>`

Public Member Functions

- **pgpr_ppitc** (Char *hypf)
- Int **regress** (Char *train, Char *test, Char *support, Int blocks)
- void **outputRst** (Char *output)

5.12.1 Detailed Description

This class provides the regression function using PITC Approximation,implemented in a parallel manner.

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

- src/pgpr_ppitc.h

5.13 pgpr_timer Class Reference

This timer class can provide real-time measure (in seconds) incurred by a block of running program.

Public Member Functions

- pgpr_timer ()
  constructor
- void start ()
  Start a timer.
- Doub end ()
  Stop a timer and compute the elapsed time in seconds.

5.13.1 Detailed Description

This timer class can provide real-time measure (in seconds) incurred by a block of running program.

5.13.2 Member Function Documentation

5.13.2.1 Doub pgpr_timer::end ( ) [inline]

Stop a timer and compute the elapsed time in seconds.

Returns

elapsed time

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

- src/pgpr_util.h

5.14 pgpr_vector< T > Class Template Reference

Vector class.

Public Types

- typedef T **value_type**

Public Member Functions

- **pgpr_vector** (int n)
- **pgpr_vector** (int n, const T &a)
- **pgpr_vector** (int n, const T *a)
- **pgpr_vector** (const pgpr_vector &rhs)
- pgpr_vector & **operator=** (const pgpr_vector &rhs)
- T & **operator[]** (const int i)
- const T & **operator[]** (const int i) const
- int **size** () const
- void **resize** (int newn)
- void **assign** (int newn, const T &a)

5.14.1 Detailed Description

template<class T>class pgpr_vector< T >

Vector class.

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

- src/pgpr_type.h

5.15 t_command_demo Struct Reference

Information parsed from commandline of application that demonstrates the regression algorithms.

`#include <pgpr_parse.h>`

Public Attributes

- Int mode

  Running mode (reserved)
- Int blocks

  Number of data blocks (used by PITC/pPITC/PIC/pPIC)
- Int rank

Size of the reduced rank (used by PICF-based GP)

- Char outf [NAMELEN]

  File containing the results.

- Char trainf [NAMELEN]

  File containing the training data.

- Char testf [NAMELEN]

  File containing the test set.

- Char hyperf [NAMELEN]

  File containing the hyperparameters.

- Char supportf [NAMELEN]

  File containing the support set (used by PITC/pPITC/PIC/pPIC)

5.15.1 Detailed Description

Information parsed from commandline of application that demonstrates the regression algorithms.

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

- src/pgpr_parse.h

5.16 t_command_prep Struct Reference

Information parsed from commandline of application that prepares the experimental data.

`#include <pgpr_parse.h>`

Public Attributes

- Int mode

  Running mode of program.

- Int seed

  Random seed.

- Int mach_num

  No. of blocks/machines.

- Int blk_size

  Data size in each block/machine.

- Char trainf [NAMELEN]

  File containing the training data.

- Int percent

  Percentage of domain inputs used for testing.

- Char testf [NAMELEN]

  File containing the test set.

- Char hyperf [NAMELEN]

  File containing the hyperparameters.

- Int support

  Size of support set.

- Char supportf [NAMELEN]

  File containing the support set (used by PITC/pPITC/PIC/pPIC)

5.16.1 Detailed Description

Information parsed from commandline of application that prepares the experimental data.

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

- src/pgpr_parse.h

5.17 t_global_summary Struct Reference

This structure is used for global_summary.

`#include <pgpr_ppic.h>`

Public Attributes

- Mdoub * **gs_kuu**
- Vdoub * **gs_zu**

5.17.1 Detailed Description

This structure is used for global_summary.

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

- src/pgpr_ppic.h

5.18 t_local_summary Struct Reference

This structure is used for local_summary.

`#include <pgpr_ppic.h>`

Public Attributes

- Mdoub * **ls_kuu**
- Vdoub * **ls_zu**

5.18.1 Detailed Description

This structure is used for local_summary.

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

- src/pgpr_ppic.h

5.19 t_state Struct Reference

Status (e.g., modified or not) of a set of elements.

`#include <pgpr_data.h>`

Public Member Functions

- t_state (Int ds)

  Constructor.
- void init (Int ds)

  Initialize the structure.
- Bool getSt (Int s)

  Get the status of a specific element.
- void upSt (Int s)

  Set the status of a specific element.

Public Attributes

- Vbool v_st

  Status of a set of elements.
- Int ss

  Size of elements whose statuses are changed.

5.19.1 Detailed Description

Status (e.g., modified or not) of a set of elements.

5.19.2 Constructor & Destructor Documentation

5.19.2.1 t_state::t_state (Int *ds*) [inline]

Constructor.

Parameters

in	*ds*	size of the set

5.19.3 Member Function Documentation

5.19.3.1 Bool t_state::getSt (Int *s*) [inline]

Get the status of a specific element.

Parameters

in	*s*	Index of an element

Returns

Binary status

5.19.3.2 void t_state::init (Int *ds*) [inline]

Initialize the structure.

Parameters

in	*ds*	size of the set

5.19.3.3 void t_state::upSt (Int *s*) [inline]

Set the status of a specific element.

Parameters

in	*s*	Index of an element

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

- src/pgpr_data.h

Chapter 6

File Documentation

6.1 src/pgpr_chol.h File Reference

This file provides Cholesky factorization and some related useful functions such as inverse, log-determinant etc.

```
#include "pgpr_type.h"
```

Classes

- class pgpr_chol

  This class provides Cholesky factorization and some related useful functions such as inverse, log-determinant etc.

6.1.1 Detailed Description

This file provides Cholesky factorization and some related useful functions such as inverse, log-determinant etc.

Version

1.0

6.2 src/pgpr_cluster.h File Reference

This file provides some basic clustering methods for the GP model.

```
#include "pgpr_type.h"
#include "pgpr_cov.h"
#include "pgpr_chol.h"
#include "pgpr_util.h"
```

Classes

- class pgpr_cluster

  Provides some basic clustering algorithms used in the approximated GP algorithms.

Macros

- #define **CLUSTER_ALGO0** 0

- #define **CLUSTER_ALGO1** 1
- #define **CLUSTER_ALGO2** 2
- #define **CLUSTER_ALGO3** 3
- #define **CLUSTER_ALGO4** 4

6.2.1 Detailed Description

This file provides some basic clustering methods for the GP model.

6.3 src/pgpr_cov.h File Reference

This file provides the covariance class: pgpr_cov, which compute the covariance matrix.

```
#include "pgpr_util.h"
```

Classes

- class pgpr_cov

  The pgpr_cov class provides the informaiton of covariance.

6.3.1 Detailed Description

This file provides the covariance class: pgpr_cov, which compute the covariance matrix.

6.4 src/pgpr_data.h File Reference

This file provides class for maintaining and operating on a domain dataset, and class for training set, test set and support set selection from domain.

```
#include "pgpr_type.h"
#include "pgpr_util.h"
#include "pgpr_parse.h"
#include "pgpr_fgp.h"
```

Classes

- struct t_state

  Status (e.g., modified or not) of a set of elements.
- class pgpr_domain

  This class stores the domain data in a matrix manner.
- class pgpr_data

  This class provides functionalities of preparing train data, test set and support set.

Macros

- #define **ALGO1** 1
- #define **ALGO2** 2
- #define **s2i**(s) (s-1)
- #define **i2s**(i) (i+1)

6.4.1 Detailed Description

This file provides class for maintaining and operating on a domain dataset, and class for training set, test set and support set selection from domain.

Author

CHEN jie, arik.cj@gmail.com

Version

1.0

6.5 src/pgpr_fgp.h File Reference

This file provides the predictor (pgpr_fgp)with full Gaussian Process.

```
#include "pgpr_util.h"
#include "pgpr_cov.h"
#include "pgpr_chol.h"
```

Classes

- class pgpr_fgp

  This class provides basic regression function using full GP.

6.5.1 Detailed Description

This file provides the predictor (pgpr_fgp)with full Gaussian Process.

6.6 src/pgpr_parse.h File Reference

This file provides functionalities to parse commandlines of different applications and domain files.

```
#include "pgpr_type.h"
#include "pgpr_util.h"
```

Classes

- struct t_command_prep

  Information parsed from commandline of application that prepares the experimental data.

- struct t_command_demo

  Information parsed from commandline of application that demonstrates the regression algorithms.

- class pgpr_parse

  This class parses domain data files, configuration file and commandline of different applications.

Macros

- #define **DOM** 0
- #define **INP** 1
- #define **OUT** 2
- #define **MNE** 3
- #define **HYP** 4
- #define **PARANUM** 5
- #define **NAMELEN** 128
- #define **CFGDEMO** 1
- #define **CFGPREP** 2

6.6.1 Detailed Description

This file provides functionalities to parse commandlines of different applications and domain files. In this version, this class support parsing two type of applications: 1. preparing data for experiment; 2. demonstration of regression algorithms. In addition, this class can load and parse domain files which includs a file storing the domain inputs and outputs (features and targets), and a configuration file containing basic information of the domain.

Author

CHEN jie, arik.cj@gmail.com

Version

1.0

6.7 src/pgpr_pic.h File Reference

This file provides the predictor (pgpr_pic)with PIC approximated Gaussian Process.

```
#include "pgpr_type.h"
#include "pgpr_cov.h"
#include "pgpr_chol.h"
#include "pgpr_cluster.h"
#include "pgpr_util.h"
```

Classes

- class pgpr_pic
  This class provides the regression function using PIC Approximation.

6.7.1 Detailed Description

This file provides the predictor (pgpr_pic)with PIC approximated Gaussian Process.

6.8 src/pgpr_pitc.h File Reference

This file provides the predictor with PITC (pgpr_pitc) Gaussian Process.

```
#include "pgpr_util.h"
#include "pgpr_cov.h"
#include "pgpr_chol.h"
```

Classes

- class pgpr_pitc

  This class provides the regression function using PITC Approximation.

6.8.1 Detailed Description

This file provides the predictor with PITC (pgpr_pitc) Gaussian Process.

6.9 src/pgpr_ppic.h File Reference

This file provides the predictor (pgpr_ppic) with parallel PIC Gaussian Process.

```
#include "mpi.h"
#include "pgpr_util.h"
#include "pgpr_cov.h"
#include "pgpr_chol.h"
#include "pgpr_cluster.h"
```

Classes

- struct t_local_summary

  This structure is used for local_summary.
- struct t_global_summary

  This structure is used for global_summary.
- class pgpr_ppic

  This class provides the regression function using PIC Approximation, but implemented in a paralle manner.

6.9.1 Detailed Description

This file provides the predictor (pgpr_ppic) with parallel PIC Gaussian Process.

6.10 src/pgpr_ppitc.h File Reference

This file provides the predictor (pgpr_ppitc) with parallel PITC Gaussian Process.

```
#include "mpi.h"
#include "pgpr_util.h"
#include "pgpr_cov.h"
#include "pgpr_chol.h"
```

Classes

- class pgpr_ppitc

  This class provides the regression function using PITC Approximation,implemented in a parallel manner.

6.10.1 Detailed Description

This file provides the predictor (pgpr_ppitc) with parallel PITC Gaussian Process.

6.11 src/pgpr_type.h File Reference

This file provides important macros, templates, basic data types (e.g., vector, matrix).

```
#include <fstream>
#include <cmath>
#include <complex>
#include <iostream>
#include <iomanip>
#include <vector>
#include <limits>
#include <stdlib.h>
#include <stdio.h>
#include <time.h>
#include <fcntl.h>
#include <string.h>
#include <ctype.h>
#include <cfloat>
#include <stdarg.h>
```

Classes

- class pgpr_vector< T >

  Vector class.
- class pgpr_matrix< T >

  Matrix class.

Macros

- #define **SUCC** 0
- #define **FAIL** -1
- #define **FALSE** 0
- #define **TRUE** 1

Typedefs

- typedef int **Int**
- typedef unsigned int **Uint**
- typedef long long int **Llong**
- typedef unsigned long long int **Ullong**
- typedef char **Char**
- typedef unsigned char **Uchar**
- typedef double **Doub**
- typedef long double **Ldoub**
- typedef bool **Bool**
- typedef const pgpr_vector< Bool > **Vbool_I**
- typedef pgpr_vector< Bool > **Vbool**
- typedef pgpr_vector< Bool > **Vbool_O**
- typedef pgpr_vector< Bool > **Vbool_IO**
- typedef const pgpr_vector< Int > **Vint_I**
- typedef pgpr_vector< Int > **Vint**
- typedef pgpr_vector< Int > **Vint_O**
- typedef pgpr_vector< Int > **Vint_IO**

- typedef const pgpr_vector< Uint > **Vuint_I**
- typedef pgpr_vector< Uint > **Vuint**
- typedef pgpr_vector< Uint > **Vuint_O**
- typedef pgpr_vector< Uint > **Vuint_IO**
- typedef const pgpr_vector< Doub > **Vdoub_I**
- typedef pgpr_vector< Doub > **Vdoub**
- typedef pgpr_vector< Doub > **Vdoub_O**
- typedef pgpr_vector< Doub > **Vdoub_IO**
- typedef const pgpr_matrix< Int > **Mint_I**
- typedef pgpr_matrix< Int > **Mint**
- typedef pgpr_matrix< Int > **Mint_O**
- typedef pgpr_matrix< Int > **Mint_IO**
- typedef const pgpr_matrix< Uint > **Muint_I**
- typedef pgpr_matrix< Uint > **Muint**
- typedef pgpr_matrix< Uint > **Muint_O**
- typedef pgpr_matrix< Uint > **Muint_IO**
- typedef const pgpr_matrix< Doub > **Mdoub_I**
- typedef pgpr_matrix< Doub > **Mdoub**
- typedef pgpr_matrix< Doub > **Mdoub_O**
- typedef pgpr_matrix< Doub > **Mdoub_IO**

6.11.1 Detailed Description

This file provides important macros, templates, basic data types (e.g., vector, matrix).

Author

CHEN jie, arik.cj@gmail.com

Version

1.0

6.12 src/pgpr_util.h File Reference

This file contains a collection of useful functions such as macro-like inline functions, debug functions, exception handling, file I/O, and a real-time timer class.

```
#include <sys/time.h>
#include "pgpr_type.h"
#include "pgpr_chol.h"
```

Classes

- class pgpr_timer

  This timer class can provide real-time measure (in seconds) incurred by a block of running program.

Macros

- #define **LEV_PRG** 0
- #define **LEV_DBG** 5
- #define **LEV_ALL** 10
- #define **CURLEV** LEV_PRG
- #define **RANI**(x) (Int)((Doub)rand()/RAND_MAX∗x)
- #define **SRAND**(x) srand(x); rand()
- #define **LOG**(x) log(x)
- #define **SQRT**(x) sqrt(x)
- #define **EXP**(x) exp(x)
- #define **POW**(x, y) (Int)pow((Doub)(x),(Doub)(y))
- #define **takeSamp**(a, b, d) for(int _i=0; _i<d; _i++) b[_i]=a[_i]
- #define **pcp**(x) pmsg(LEV_DBG,stdout,"CheckPoint> %d\n",x)
- #define **pcpst**(x) pmsg(LEV_DBG,stdout,"[%d] start>\n",x)
- #define **pcpen**(x) pmsg(LEV_DBG,stdout,"[%d] end# \n",x)
- #define **pcp_s**(s, x) pmsg(LEV_DBG,stdout," %s > %.4lf\n",s,(double)x)
- #define **pvec**(vec)
- #define **pvec_r**(vec, vsize)
- #define **pvec_s**(_s, vec)
- #define **pmat_r**(mat, _r)
- #define **pmat_pos**(mat)
- #define **pmat**(mat)
- #define **pmat_s**(_s, mat)
- #define **throw**(message) {printf("ERROR: %s\n in file %s at line %d\n", message,__FILE__,__LINE__); throw(1);}

Functions

- template<class T >
  T **SQR** (const T a)
- template<class T >
  const T & **MAX** (const T &a, const T &b)
- float **MAX** (const double &a, const float &b)
- float **MAX** (const float &a, const double &b)
- template<class T >
  const T & **MIN** (const T &a, const T &b)
- float **MIN** (const double &a, const float &b)
- float **MIN** (const float &a, const double &b)
- template<class T >
  T **SIGN** (const T &a, const T &b)
- float **SIGN** (const float &a, const double &b)
- float **SIGN** (const double &a, const float &b)
- template<class T >
  void **SWAP** (T &a, T &b)
- Int **getLines** (Char ∗file)
- Int **saveData** (Char ∗file, Mdoub m_data)
- Int **loadData** (Char ∗file, Mdoub &m_data)
- Int **saveHyper** (Char ∗file, Vdoub h, Int d)
- void **pmsg** (int level, FILE ∗outfp, const char ∗format,...)
- void **bubble_sort** (Vdoub a, Vint &ai)
- Int **argmaxi** (Vdoub v)
- Doub **getRmse** (Vdoub v1, Vdoub v2)
- Doub **getMnlp** (Vdoub v1, Vdoub v2, Vdoub v3)
- Int **A_invB_C** (Mdoub A, pgpr_chol ∗chol_b, Vdoub C, Vdoub &D)

- Int **A_invB_C** (Mdoub A, pgpr_chol *chol_b, Mdoub C, Mdoub &D)
- Int **A_invB_transC** (Mdoub A, pgpr_chol *chol_b, Mdoub C, Mdoub &D)
- Int **A_invB_C** (Mdoub A, pgpr_chol *chol_b, Mdoub &D)
- Int **trace_A_invB_C** (Mdoub A, pgpr_chol *chol_b, Mdoub C, Vdoub &D)
- Int **trace_A_invB_C** (Mdoub A, pgpr_chol *chol_b, Vdoub &D)
- Int **trace_A_invB_transC** (Mdoub A, pgpr_chol *chol_b, Mdoub C, Vdoub &D)

6.12.1 Detailed Description

This file contains a collection of useful functions such as macro-like inline functions, debug functions, exception handling, file I/O, and a real-time timer class.

Author

CHEN jie, arik.cj@gmail.com

Version

1.0

6.12.2 Macro Definition Documentation

6.12.2.1 #define pmat(*mat*)

Value:

```
for(Int _i=0; _i<mat.nrows(); _i++)\
    {\
        for(Int _j=0; _j<mat.ncols(); _j++)\
        {\
            pmsg(LEV_DBG,stdout,"%.4f ",(double)mat[_i][_j]);\
        }\
        pmsg(LEV_DBG,stdout,"\n");\
    }\
```

6.12.2.2 #define pmat_pos(*mat*)

Value:

```
for(Int _i=0; _i<mat.nrows(); _i++)\
    {\
        for(Int _j=0; _j<mat.ncols(); _j++)\
        {\
            if(mat[_i][_j]<0) break;\
            pmsg(LEV_DBG,stdout,"%.4f ",(double)mat[_i][_j]);\
        }\
        pmsg(LEV_DBG,stdout,"\n");\
    }\
```

6.12.2.3 #define pmat_r(*mat, _r*)

Value:

```
for(Int _i=0; _i<_r; _i++)\
    {\
        for(Int _j=0; _j<_r; _j++)\
        {\
            pmsg(LEV_DBG,stdout,"%.4f ",(double)mat[_i][_j]);\
        }\
        pmsg(LEV_DBG,stdout,"\n");\
    }\
```

6.12.2.4 #define pmat_s(_s, mat)

Value:

```
pmsg(LEV_DBG,stdout,"%s:\n",_s);\
    for(Int _i=0; _i<mat.nrows(); _i++)\
    {\
        for(Int _j=0; _j<mat.ncols(); _j++)\
        {\
            pmsg(LEV_DBG,stdout,"%.4f ",(double)mat[_i][_j]);\
        }\
        pmsg(LEV_DBG,stdout,"\n");\
    }\
```

6.12.2.5 #define pvec(vec)

Value:

```
for(Int _j=0; _j<vec.size(); _j++)\
    {\
        pmsg(LEV_DBG,stdout,"%.4f ",(double)vec[_j]);\
    }\
    pmsg(LEV_DBG,stdout,"\n");
```

6.12.2.6 #define pvec_r(vec, vsize)

Value:

```
for(Int _j=0; _j<vsize; _j++)\
    {\
        pmsg(LEV_DBG,stdout,"%.4f ",(double)vec[_j]);\
    }\
    pmsg(LEV_DBG,stdout,"\n");
```

6.12.2.7 #define pvec_s(_s, vec)

Value:

```
pmsg(LEV_DBG,stdout,"%s:\n",_s);\
    for(Int _j=0; _j<vec.size(); _j++)\
    {\
        pmsg(LEV_DBG,stdout,"%.4f ",(double)vec[_j]);\
    }\
    pmsg(LEV_DBG,stdout,"\n");
```

Index