User's Guide to the R Packages Attached to the lasvdGP Paper

April 14, 2018

`DynamicGP-package` *GP Models for Large-Scale Dynamic Computer Experiments*

Description

This R package provides three functions for emulating dynamic computer experiments. The function `svdGP` fits full SVD-based GP model which is computationally demanding for large-scale dyanmic computer experiments. As is well known, the time complexity of fitting a GP model is $O(N^3)$ where N is the number of training/design points. Since fitting a common GP model for really large N would be computationally burdensome, we fit local SVD-based GP models on a sequentially selected small neighborhood set for every test inputs. The function `knnsvdGP` fits K-nearest neighbor SVD-based GP models which selects neighborhood sets based on the Euclidean distance with repect to the test points. The function `lasvdGP` fits local approximate SVD-based GP model using the new algorithm proposed by Zhang et al. (2017).

The lasvdGP is an extension of the local approximate GP (laGP) model developed by Gramacy and Lee (2015) for the emulation of large-scale scalar valued computer experiments. The neighborhood selection and SVD-based GP model fitting algorithm is suitable for parallelization. We use the R package "parallel" for this task. The parallelization can achieve nearly linear speed since the procedure on each test point is independent and identical.

Author(s)

Ru Zhang <heavenmarshal@gmail.com>,

C. Devon Lin <devon.lin@queensu.ca>,

Pritam Ranjan <pritam.ranjan@gmail.com>

References

Gramacy, R. B. and Apley, D. W. (2015) *Local Gaussian process approximation for large computer experiments*, Journal of Computational and Graphical Statistics 24(2), 561-578.

Zhang, R., Lin, C. D. and Ranjan, P. (2018) *Local Gaussian Process Model for Large-scale Dynamic Computer Experiments*, arXiv:1611.09488.

environ *The Simulation Function of Example 2*

Description

Evaluate the synthesized dynamic computer simulator in Example 2 one design point at a time.

Usage

```
environ(xx,timepoints)
```

Arguments

xx	five dimensional vector of the design input.
timepoints	a vector of time-points with arbitrary length on which to evaluate the simulator.

Value

It returns a vector of time-series outputs with the same length as `timepoints`.

Author(s)

Ru Zhang <heavenmarshal@gmail.com>

Examples

```
library("lhs")
library("simfuncs")
## generate a design matrix of 100 points
design <- maximinLHS(100,5)
## specify the timepoints
timepoints <- seq(0.3,60,0.3)
## evaluate the simulator on the 100 design points
resp <- apply(design,1,environ,timepoints)
```

forretal *The Simulation Function of Example 1*

Description

Evaluate the synthesized dynamic computer simulator in Example 1 one design point at a time.

Usage

```
forretal(xx,timepoints)
```

Arguments

xx	three dimensional vector of the design input.
timepoints	a vector of time-points with arbitrary length on which to evaluate the simulator.

Value

It returns a vector of time-series outputs with the same length as `timepoints`.

Author(s)

Ru Zhang <heavenmarshal@gmail.com>

Examples

```
library("lhs")
library("simfuncs")
## generate a design matrix of 100 points
design <- maximinLHS(100,3)
## specify the timepoints
timepoints <- seq(0,1,len=200)
## evaluate the simulator on the 100 design points
resp <- apply(design,1,forretal,timepoints)
```

---

knnsvdGP *K-nearest neighbor SVD-Based GP model*

Description

Fits a K-nearest neighbour SVD-based GP model on a test set `X0`, training set `design` and response matrix `resp`. The local neighbourhood sets consist of `nn` points which are selected by the Euclidean distance with respect to the test points. See Zhang et al. (2017) for details.

Usage

```
knnsvdGP(design, resp, X0=design, nn=20, nsvd = nn, frac = .9,
         gstart = 0.0001, nstarts = 5,centralize=FALSE, maxit=100, verb=0,
         nthread = 4, clutype="PSOCK")
```

Arguments

`design`	An N by d matrix of N training/design inputs.
`resp`	An L by N response matrix of `design`, where L is the length of the time series outputs, N is the number of design points.
`X0`	An M by d matrix of M test inputs. The localized SVD-based GP models will be fitted on every point (row) of `X0`. The default value of `X0` is `design`.
`nn`	The number of neighborhood points selected by the Euclidean distance. the default value is 20.
`nsvd`	The number of design points closest to the test points on whose response matrix to perform the initial singular value decomposition. The default value is `nn`.
`frac`	The threshold in the cumulative percentage criterion to select the number of SVD bases. The default value is 0.9.
`gstart`	The starting number and upper bound of for estimating the nugget parameter. If `gstart = sqrt(.Machine$double.eps)`, the nugget will be fixed at `sqrt(.Machine$double.eps)`, since the it is the lower bound of the nugget term. The default value is 0.0001.

nstarts	The number of starting points used in the numerical maximization of the posterior density function. The larger nstarts will typically lead to more accurate prediction but longer computational time. The default value is 5.
centralize	If centralize=TRUE the response matrix will be centralized (subtract the mean) before the start of the algorithm. The mean will be added to the predictive mean at the finish of the algorithm. The default value is FALSE.
maxit	Maximum number of iterations in the numerical optimization algorithm for maximizing the posterior density function. The default value is 100.
verb	A nonnegative integer indicates the level of printing on the screen. If verb=0 the function is executed in silence. The default value is 0.
nthread	The number of threads (processes) used in parallel execution of this function. nthread=1 implies no parallelization. The default value is 4.
clutype	The type of cluster in the R package "parallel" to perform parallelization. The default value is "PSOCK". Required only if nthread>1.

Value

pmean	An L by M matrix of posterior predicted mean for the response at the test set X0.
ps2	An L by M matrix of posterior predicted variance for the response at the test set X0.

Author(s)

Ru Zhang <heavenmarshal@gmail.com>,

C. Devon Lin <devon.lin@queensu.ca>,

Pritam Ranjan <pritam.ranjan@gmail.com>

References

Zhang, R., Lin, C. D. and Ranjan, P. (2018) *Local Gaussian Process Model for Large-scale Dynamic Computer Experiments*, arXiv:1611.09488.

See Also

lasvdGP, svdGP.

Examples

```
library("lhs")
forretal <- function(x,t,shift=1)
{
    par1 <- x[1]*6+4
    par2 <- x[2]*16+4
    par3 <- x[3]*6+1
    t <- t+shift
    y <- (par1*t-2)^2*sin(par2*t-par3)
}
timepoints <- seq(0,1,len=200)
design <- lhs::randomLHS(100,3)
test <- lhs::randomLHS(20,3)
## evaluate the response matrix on the design matrix
```

```
resp <- apply(design,1,forretal,timepoints)

nn <- 20
gs <- sqrt(.Machine$double.eps)
## knnsvdGP with mutiple (5) start points for GP model estimation
## It use the R package "parallel" for parallelization
retknnmsp <- knnsvdGP(design,resp,test,nn,frac=.95,gstart=gs,
                      centralize=TRUE,nstarts=5,nthread=2,clutype="PSOCK")

## knnsvdGP with single start point for GP model estimation
## It does not use parallel computation
retknnss <- knnsvdGP(design,resp,test,nn,frac=.95,gstart=gs,
                     centralize=TRUE,nstarts=1,nthread=1)
```

`lasvdGP` *Local Approximate SVD-Based GP Models*

Description

Fits a local approximate SVD-based GP model on a test set `X0`, training/design set `design` and response matrix `resp`. The local neighborhood sets consist of `nn` out of which `n0` points are selected by the Euclidean distance with respect to the test points. The remaining `nn-n0` neighborhood points are selected sequentially by a greedy algorithm proposed by Zhang et al. (2017).

Usage

```
lasvdGP(design, resp, X0=design, n0=10, nn=20,
        nfea = min(1000,nrow(design)),
        nsvd = nn, nadd = 1, frac = .9, gstart = 0.0001,
        resvdThres = min(5, nn-n0), every = min(5,nn-n0),
        nstarts = 5,centralize=FALSE, maxit=100, verb=0,
        nthread = 4, clutype="PSOCK")
```

Arguments

`design`	An N by d matrix of N training/design inputs.
`resp`	An L by N response matrix of `design`, where L is the length of the time series outputs, N is the number of design points.
`X0`	An M by d matrix of M test inputs. The localized SVD-based GP models will be fitted on every point (row) of `X0`. The default value of `X0` is `design`.
`n0`	The number of points in the initial neighborhood set. The initial neighborhood set is selected by the Euclidean distance. The default value is 10.
`nn`	The total number of neighborhood points. The `nn-n0` points are selected sequentially by the proposed algorithm. The default value is 20.
`nfea`	The number of feasible points within which to select the neighborhood points. This function will only consider the `nfea` design points closest to the test point in terms of Euclidean distance when selecting neighborhood points. The default value is the minimum of N and 1000.
`nsvd`	The number of design points closest to the test points on whose response matrix to perform the initial singular value decomposition. The default value is `nn`.

nadd	The number of neighborhood points selected at one iteration. The default value is 1.
frac	The threshold in the cumulative percentage criterion to select the number of SVD bases. The default value is 0.9.
gstart	The starting number and upper bound of for estimating the nugget parameter. If `gstart = sqrt(.Machine$double.eps)`, the nugget will be fixed at `sqrt(.Machine$double.eps)`, since the it is the lower bound of the nugget term. The default value is 0.0001.
resvdThres	The threshold to re-perform SVD. After every `resvdThres` points have been included into the neighborhood set, the SVD of the response matrix will be re-performed and the SVD-based GP model will be refitted. The default value is the minimum of `nn-n0` and 5.
every	The threshold to refit GP models without re-perform SVD. After every `every` points have been included into the neighborhood set, the GP models will be refitted. But the SVD will not be re-performed. It is suggested `every <= resvdThres`. The default value is the minimum of `nn-n0` and 5.
nstarts	The number of starting points used in the numerical maximization of the posterior density function. The larger `nstarts` will typically lead to more accurate prediction but longer computational time. The default value is 5.
centralize	If `centralize=TRUE` the response matrix will be centralized (subtract the mean) before the start of the algorithm. The mean will be added to the predictive mean at the finish of the algorithm. The default value is `FALSE`.
maxit	Maximum number of iterations in the numerical optimization algorithm for maximizing the posterior density function. The default value is 100.
verb	A nonnegative integer indicates the level of printing on the screen. If `verb=0` the function is executed in silence. The default value is 0.
nthread	The number of threads (processes) used in parallel execution of this function. `nthread=1` implies no parallelization. The default value is 4.
clutype	The type of cluster in the R package "parallel" to perform parallelization. The default value is "PSOCK". Required only if `nthread>1`.

Value

pmean	An L by M matrix of posterior predicted mean for the response at the test set `X0`.
ps2	An L by M matrix of posterior predicted variance for the response at the test set `X0`.

Author(s)

Ru Zhang <heavenmarshal@gmail.com>,

C. Devon Lin <devon.lin@queensu.ca>,

Pritam Ranjan <pritam.ranjan@gmail.com>

References

Zhang, R., Lin, C.D. and Ranjan, P. (2018) *Local Gaussian Process Model for Large-scale Dynamic Computer Experiments*, arXiv:1611.09488.

See Also

knnsvdGP, svdGP.

Examples

```
library("lhs")
forretal <- function(x,t,shift=1)
{
    par1 <- x[1]*6+4
    par2 <- x[2]*16+4
    par3 <- x[3]*6+1
    t <- t+shift
    y <- (par1*t-2)^2*sin(par2*t-par3)
}
timepoints <- seq(0,1,len=200)
design <- lhs::randomLHS(100,3)
test <- lhs::randomLHS(20,3)
## evaluate the response matrix on the design matrix
resp <- apply(design,1,forretal,timepoints)

n0 <- 15
nn <- 20
gs <- sqrt(.Machine$double.eps)

## lasvdGP with mutiple (5) start points for GP model estimation,
## It use the R package "parallel" for parallelization
retlamsp <- lasvdGP(design,resp,test,n0,nn,frac=.95,gstart=gs,
                    centralize=TRUE,nstarts=5,nthread=2,clutype="PSOCK")

## lasvdGP with single start point for GP model estimation,
## It does not use parallel computation
retlass <- lasvdGP(design,resp,test,n0,nn,frac=.95,gstart=gs,
                   centralize=TRUE,nstarts=1,nthread=1)
```

`lscalemse` *Evaluating the NMSPE criterion*

Description

Evaluate the NMSPE criterion for the matrix of predictive mean of the time-series valued responses on a test set.

Usage

```
lscalemse(yreal,ypred)
```

Arguments

`yreal`	an $L \times M$ matrix of real responses on the test set, where L is the length of the time series and M is the number of input points in the test set.
`ypred`	an $L \times M$ matrix of predictive mean for the responses on the test set. The size of `ypred` must be equal to that of `yreal`.

Value

It returns a vector of length M contains the values of NMSPE at each test point, where M is the number of test points.

Author(s)

Ru Zhang <heavenmarshal@gmail.com>

Examples

```
library("lhs")
library("lasvdgp")
library("simfuncs")
## generate a design and a test matrix of each 100 points
design <- maximinLHS(100,5)
test <- maximinLHS(100,5)
## specify the timepoints
timepoints <- seq(0.3,60,0.3)
## evaluate the simulator on the design and test set.
resp <- apply(design,1,environ,timepoints)
tres <- apply(test,1,environ,timepoints)
## perform lasvdGP
retg <- lasvdgpWorker(test,design,resp,15,30)
## evaluate the NMSPE
nmse <- lscalemse(tres,retg$pmean)
```

---

readdata *Read the Serialized Arrays from a Serialization File*

Description

It read all the written arrays from a serialization file and return a list of arrays. The length of the list is the same as the number of array names in the header line. The sizes of the arrays in the list are specified by the header line.

Usage

```
readdata(filename)
```

Arguments

filename	a string of the name of the serialization file to be read.

Value

lst	a list of arrays written in the serialization file. The names and the sizes of the arrays are determined by the header line. The readdata function will fill NA for the positions not written in the serialization file.

Author(s)

Ru Zhang <heavenmarshal@gmail.com>

Examples

```
library("serialization")
## we intend to write two matrices into the serialization file
mats <- c("mat1","mat2")
## dim(mat1)=c(2,3), dim(mat2)=c(4,2)
dims <- list(c(2,3),c(4,2))
fcon <- writeheader("mats.txt",mats,dims)
flush(fcon)
## mats.txt has been created the first line is
## headerline:mat1,2,3;mat2,4,2;

## write data
writedata(fcon,"mat1",c(1,1),0)
writedata(fcon,"mat1",c(1,2),1)
writedata(fcon,"mat2",c(1,1),2)
## close connection
close(fcon)
## read the file and obtain the matrices
lst <- readdata("mats.txt")
## show the names of the loaded matrices
names(lst)
print(lst)
```

`scoring` *Evaluating the Proper Scoring Rule*

Description

Evaluate the proper scoring rule for the predictions of the time-series valued responses on a test set.

Usage

```
scoring(yreal,predmean,predsd)
```

Arguments

`yreal`	an $L \times M$ matrix of real responses on the test set, where L is the length of the time series and M is the number of input points in the test set.
`predmean`	an $L \times M$ matrix of predictive mean for the responses on the test set. The size of `predmean` must be equal to that of `yreal`.
`predsd`	an $L \times M$ matrix of predictive standard deviation for the responses on the test set. The size of `predsd` must be equal to that of `yreal`.

Value

It returns a vector of length M contains the values of the proper scoring rule at each test point, where M is the number of test points.

Author(s)

Ru Zhang <heavenmarshal@gmail.com>

Examples

```
library("lhs")
library("lasvdgp")
library("simfuncs")
## generate a design and a test matrix of each 100 points
design <- maximinLHS(100,5)
test <- maximinLHS(100,5)
## specify the timepoints
timepoints <- seq(0.3,60,0.3)
## evaluate the simulator on the design and test set.
resp <- apply(design,1,environ,timepoints)
tres <- apply(test,1,environ,timepoints)
## perform lasvdGP
retg <- lasvdgpWorker(test,design,resp,15,30)
## evaluate the proper scoring rule. Note that lasvdgpWorker returns predictive
## variance, to evaluate proper scoring rule, we must take square root.
score <- scoring(tres,sqrt(retg$ps2))
```

`serialization-package` *A Package for Data Serialization*

Description

This package is designed to record the intermediate results of the simulation study. It writes the results of the simulation coming out from each repetition into a text file. In the meantime or after the simulation, we can load the serialized arrays that recording the results from the serialization file to the memory as a list of arrays.

Author(s)

Ru Zhang <heavenmarshal@gmail.com>

`svdGP` *Full SVD-Based GP Models*

Description

This function fits a full SVD-based GP model with test set `X0`, design set `design` and response matrix `resp`.

Usage

```
svdGP(design,resp,X0=design,nstarts=5,d=NULL,gstart=0.0001,
      frac=.9,centralize=FALSE,nthread=4,clutype="PSOCK")
```

Arguments

design	An N by d matrix of N training/design inputs.
resp	An L by N response matrix of design, where L is the length of the time series outputs, N is the number of design points.
X0	An M by d matrix of M test inputs. The default value of X0 is design.
nstarts	The number of starting points used in the numerical maximization of the posterior density function. The larger nstarts will typically lead to more accurate prediction but longer computational time. The default value is 5.
d	the start value of lengthscale parameter of GP models, a length q vector of positive values. If d=NULL, the start value will be selected automatically. The default value is NULL.
gstart	The starting number and upper bound of for estimating the nugget parameter. If gstart = sqrt(.Machine$double.eps), the nugget will be fixed at sqrt(.Machine$double.eps), since the it is the lower bound of the nugget term. The default value is 0.0001.
frac	The threshold in the cumulative percentage criterion to select the number of SVD bases. The default value is 0.9.
centralize	If centralize=TRUE the response matrix will be centralized (subtract the mean) before the start of the algorithm. The mean will be added to the predictive mean at the finish of the algorithm. The default value is FALSE.
nthread	The number of threads (processes) used in parallel execution of this function. nthread=1 implies no parallelization. The default value is 4.
clutype	The type of cluster in the R package "parallel" to perform parallelization. The default value is "PSOCK". Required only if nthread>1.

Value

pmean	An L by M matrix of posterior predicted mean for the response at the test set X0.
ps2	An L by M matrix of posterior predicted variance for the response at the test set X0.

Author(s)

Ru Zhang <heavenmarshal@gmail.com>,

C. Devon Lin <devon.lin@queensu.ca>,

Pritam Ranjan <pritam.ranjan@gmail.com>

See Also

knnsvdGP, lasvdGP.

Examples

```
library("lhs")
forretal <- function(x,t,shift=1)
{
    par1 <- x[1]*6+4
    par2 <- x[2]*16+4
    par3 <- x[3]*6+1
```

```
    t <- t+shift
    y <- (par1*t-2)^2*sin(par2*t-par3)
}
timepoints <- seq(0,1,len=200)
design <- lhs::randomLHS(50,3)
test <- lhs::randomLHS(50,3)
## evaluate the response matrix on the design matrix
resp <- apply(design,1,forretal,timepoints)

## fit full SVD-based GP model
ret <- svdGP(design,resp,test,frac=.95,
             centralize=TRUE,nthread=2)
```

`writedata` *Write One Entry of an Array into the Serialization File*

Description

This function write one entry of one array to the serialization file, this entry takes up one line in the file. The format is

arrname,dim1,dim2:data

where arrname is the name of the array to be serialized, dim1 and dim2 are the positions of the data in terms of the first and the second dimension of the array, data is value of this entry.

Usage

```
writedata(fcon,itemname,dim,value)
```

Arguments

`fcon`	the file connection object of the serialization file created by the `writeheader` function.
`itemname`	a string of the name of the array (vector or matrix) to be written.
`dim`	a numerical vector indicates the position of the entry to be written.
`value`	the value of the entry to be written, currently only support numerical value.

Author(s)

Ru Zhang <heavenmarshal@gmail.com>

Examples

```
library("serialization")
## we intend to write two matrices into the serialization file
mats <- c("mat1","mat2")
## dim(mat1)=c(2,3), dim(mat2)=c(4,2)
dims <- list(c(2,3),c(4,2))
fcon <- writeheader("mats.txt",mats,dims)
flush(fcon)
## mats.txt has been created the first line is
## headerline:mat1,2,3;mat2,4,2;
```

```
## write data
writedata(fcon,"mat1",c(1,1),0)
writedata(fcon,"mat1",c(1,2),1)
writedata(fcon,"mat2",c(1,1),2)
## close connection
close(fcon)
## the mats.txt becomes
## headerline:mat1,2,3;mat2,4,2;
## mat1,1,1:0
## mat1,1,2:1
## mat2,1,1:2
```

`writeheader` *Write a Header Line to the Serialization File*

Description

Write a header line to the serialization file. The header line indicate the name of the arrays written into this file. The format is

headerline:arr1,row1,col1;arr2,row2,col2;

where arr1 is the name of the first array, row1 and col1 are the numbers of rows and columns of arr1. information for different arrays are separated by semicolon.

Usage

```
writeheader(filename,items,dims)
```

Arguments

`filename`	a string of the name of the file for serialization.
`items`	a vector of strings, each string is a name of an array to be saved in the serialization file.
`dims`	a list of vectors, each vector indicates the size of an array to be saved in the serialization file. The length of `dims` must be equal to the length of `items` or 1. When `dims=1`, all the arrays are the same size.

Value

`fcon`

a connect object to the serialization file.

Author(s)

Ru Zhang <heavenmarshal@gmail.com>

Examples

```
library("serialization")
## we intend to write two matrices into the serialization file
mats <- c("mat1","mat2")
## dim(mat1)=c(2,3), dim(mat2)=c(4,2)
dims <- list(c(2,3),c(4,2))
fcon <- writeheader("mats.txt",mats,dims)
flush(fcon)
## mats.txt has been created the first line is
## headerline:mat1,2,3;mat2,4,2;

## write data
writedata(fcon,"mat1",c(1,1),0)
writedata(fcon,"mat1",c(1,2),1)
writedata(fcon,"mat2",c(1,1),2)
## close connection
close(fcon)
## the mats.txt becomes
## headerline:mat1,2,3;mat2,4,2;
## mat1,1,1:0
## mat1,1,2:1
## mat2,1,1:2
```