Package 'liGP'

October 13, 2022

Title Locally Induced Gaussian Process Regression

Version 1.0.1

Date 2021-06-29

Maintainer D. Austin Cole <austin.cole8@vt.edu>

Description

Performs locally induced approximate GP regression for large computer experiments and spatial datasets following Cole D.A., Christianson, R., Gramacy, R.B. (2021) Statistics and Computing, 31(3), 1-21, <arXiv:2008.12857>. The approximation is based on small local designs combined with a set of inducing points (latent design points) for predictions at particular inputs. Parallelization is supported for generating predictions over an immense out-of-sample testing set. Local optimization of the inducing points design is provided based on variance-based criteria. Inducing point template schemes, including scaling of space-filling designs, are also provided.

Depends R (>= 3.4)

Imports hetGP, laGP, doParallel, foreach

Suggests lhs

License LGPL

Encoding UTF-8

RoxygenNote 7.1.1

NeedsCompilation yes

Author D. Austin Cole [aut, cre],
Ryan B Christianson [cph],
Robert B. Gramacy [cph]

Repository CRAN

Date/Publication 2021-07-17 06:00:02 UTC

R topics documented:

`borehole` *Borehole equation data generator*

Description

Generates a vector of outputs from the borehole function, a common testing function for computer experiments. The function models water flow through a borehole.

Usage

```
borehole(X)
```

Arguments

`X`	a `matrix` containing the full (large) design matrix of input locations in [0,1]^8

Details

For more details, see Worley, B. A. (1987). Deterministic uncertainty analysis (No. CONF-871101-30). Oak Ridge National Lab., TN (USA)

Value

a vector of evaluations of the borehole function, length = `nrow(X)`

References

Harper, W. V., & Gupta, S. K. (1983). *Sensitivity/uncertainty analysis of a borehole scenario comparing Latin Hypercube Sampling and deterministic sensitivity approaches* (No. BMI/ONWI-516). Battelle Memorial Inst., Columbus, OH (USA). Office of Nuclear Waste Isolation.

Worley, B. A. (1987). *Deterministic uncertainty analysis* (No. CONF-871101-30). Oak Ridge National Lab., TN (USA).

Examples

```
X <- matrix(runif(800), ncol=8)
borehole(X)
```

`build_gauss_measure_ipTemplate` *Inducing point template design for a Gaussian measure built through sequential optimization*

Description

Constructs a design of inducing points around a Gaussian measure whose mean is the center of the design matrix and its local neighborhood. The output is an inducing point design centered at the origin that can be used as a template for predictions anywhere in the design space (with a local neighborhood of the same size). The inducing points are sequentially selected by optimizing "`wimse`", weighted Integrated Mean Squared Error.

Usage

```
build_gauss_measure_ipTemplate(X = NULL, Y = NULL, M, N, gauss_sd,
                               theta = NULL, g = 1e-4, seq_length=20,
                               ip_bounds = NULL, integral_bounds = NULL,
                               num_multistart = 20,
                               epsK = sqrt(.Machine$double.eps), epsQ = 1e-5,
                               reps = FALSE, verbose = TRUE)
```

Arguments

`X`	a `matrix` containing the full (large) design matrix of input locations. If using a list for `reps`, this entry is not used
`Y`	a vector of responses/dependent values with `length(Y)=nrow(X)`. If using a list for `reps`, this entry is not used
`M`	a positive integer number of inducing points; `M` should be less than `N`
`N`	the positive integer number of Nearest Neighbor (NN) locations used to build a local neighborhood
`gauss_sd`	a vector of standard deviations for the Gaussian measure with `length(gauss_sd)=nrow(X)`. Note: at this time, the Gaussian measure must only have one nonzero standard deviation (i.e. the Gaussian measure is a slice)
`theta`	the lengthscale parameter (positive number) in a Gaussian correlation function; a (default) `NULL` value sets the lengthscale at the square of the 10th percentile of pairwise distances between neighborhood points (similar to `darg` in `laGP` package)
`g`	the nugget parameter (positive number) in a covariance
`seq_length`	a positive integer used to build sequences of this length in the nondegenerate dimensions for the purpose of building a local neighbhorhood.

ip_bounds	a 2 by d matrix of the bounds used in the optimization of inducing points; the first row contains minimum values, the second row the maximum values; if not provided, the bounds of the center's local neighborhood are used
integral_bounds	a 2 by d matrix of the bounds used in the calculation of wimse; the first row contains minimum values, the second row the maximum values; only relevant when method="wimse"; if not provided, defaults to the range of each column of X
num_multistart	a scalar positive integer indicating the number of multistart points used to optimize each inducing point
epsK	a small positive number added to the diagonal of the correlation matrix of inducing points for numerically stability for inversion
epsQ	a small positive number added to the diagonal of the Q matrix (see Cole (2021)) for numerically stability for inversion
reps	a notification of replicate design locations in the data set. If TRUE, the unique design locations are used for the calculations along with the average response for each unique design location. Alternatively, reps can be a list from find_reps in the hetGP package. In this case, X and Y are not used.
verbose	when TRUE, prints the current number of inducing points selected during the sequential optimization process

Details

This function is built to deal with the special class of problems where liGP is used to predict and integrate over a degenerate Gaussian measure where only one dimension has a nonzero standard deviation. To build the wIMSE inducing point design, the function optIP.wIMSE is called with the reference point being the median of the design matrix.

For each inducing point design, the first inducing point is placed at the predictive location (i.e. the origin).

Value

The output is a list with the following components.

Xm.t	a matrix of M inducing points centered at the origin
Xn	a matrix of the local neighborhood at the center of the design
Xc	a matrix of the center of the design used to build the local neighborhood and inducing point template
gauss_sd	the gauss_sd used to generate the local neighborhood
time	a scalar giving the passage of wall-clock time elapsed for (substantive parts of) the calculation

Author(s)

D. Austin Cole <austin.cole8@vt.edu>

References

D.A. Cole, R.B. Christianson, and R.B. Gramacy (2021). *Locally Induced Gaussian Processes for Large-Scale Simulation Experiments* Statistics and Computing, 31(3), 1-21; preprint on arXiv:2008.12857; https://arxiv.org/abs/2008.12857

See Also

optIP.wIMSE

Examples

```
## "2D Toy Problem"
## Herbie's Tooth function used in Cole et al (2020);
## thanks to Lee, Gramacy, Taddy, and others who have used it before

## build up a design with N=~40K locations
x <- seq(-2, 2, by=0.02)
X <- as.matrix(expand.grid(x, x))
Y <- herbtooth(X)
X_center <- apply(X, 2, median)
gauss_sd <- c(0, .05)

## build a inducing point template, first with original weighted Integrated Mean-Square Error
int_bounds <- rbind(c(-2,-2), c(2,2))
wimse.out <- build_ipTemplate(X, Y, N=100, M=10, method='wimse',
                              integral_bounds=int_bounds)
Xm.t_wimse <- wimse.out$Xm.t
Xn <- wimse.out$Xn


wimse_gauss.out <- build_gauss_measure_ipTemplate(X, Y, N=100, M=10,
                                                  gauss_sd = gauss_sd,
                                                  integral_bounds=int_bounds)
Xm.t_wimse_gauss <- wimse_gauss.out$Xm.t
Xn_gauss <- wimse_gauss.out$Xn

## plot locally optimized inducing point templates
ylim <- range(Xn_gauss[,2]) + c(-.03, .05)
plot(Xn, pch=16, cex=.5, col='grey',
     xlab = 'x1', ylab = 'x2', ylim = ylim,
     main='Locally optimized IP template based on Gaussian measure')
points(Xn_gauss, cex=.7)
points(X_center[1], X_center[2], pch=16, cex=1.5)
points(Xm.t_wimse, pch=2, lwd=2, col=3)
points(Xm.t_wimse_gauss, pch=6, lwd=2, col=2)
legend('topleft', pch = c(16, 1, 2, 3), col = c('grey', 1, 3, 2),
       legend=c('Local neighborhood (wIMSE)',
                'Local neighborhood (Gauss measure)',
                'wIMSE ip design',
                'Gaussian measure ip design'))
```

`build_ipTemplate` *Inducing point template design built through sequential optimization*

Description

Constructs a design of inducing points around the center of the design matrix and its local neighborhood. The output is an inducing point design centered at the origin that can be used as a template for predictions anywhere in the design space (with a local neighborhood of the same size). Different criteria are available to optimize the inducing points. The methods "`wimse`" and "`alc`" use weighted Integrated Mean Squared Error and Active Learning Cohn respectively to sequentially select inducing points.

Usage

```
build_ipTemplate(X = NULL, Y = NULL, M, N, theta = NULL, g = 1e-4,
                 method = c('wimse','alc'), ip_bounds = NULL,
                 integral_bounds = NULL, num_thread = 1, num_multistart = 20,
                 w_var = NULL, epsK = sqrt(.Machine$double.eps), epsQ = 1e-5,
                 reps = FALSE, verbose = TRUE)
```

Arguments

`X`	a `matrix` containing the full (large) design matrix of input locations. If using a list for `reps`, this entry is not used.
`Y`	a vector of responses/dependent values with `length(Y)=nrow(X)`. If using a list for `reps`, this entry is not used.
`M`	a positive integer number of inducing points; `M` should be less than `N`
`N`	a positive integer number of Nearest Neighbor (NN) locations used to build a local neighborhood
`theta`	the lengthscale parameter (positive number) in a Gaussian correlation function; a (default) `NULL` value sets the lengthscale at the square of the 10th percentile of pairwise distances between neighborhood points (similar to `darg` in `laGP` package)
`g`	the nugget parameter (positive number) in a covariance
`method`	specifies the method by which the inducing point template is built. In brief, wIMSE (`"wimse"`, default) minimizes the weighted integrated predictive variance and ALC (`"alc"`) minimizes predictive variance
`ip_bounds`	a 2 by d `matrix` of the bounds used in the optimization of inducing points; the first row contains minimum values, the second row the maximum values; if not provided, the bounds of the center's local neighborhood are used
`integral_bounds`	a 2 by d `matrix` of the bounds used in the calculation of `wimse`; the first row contains minimum values, the second row the maximum values; only relevant when `method="wimse"`; if not provided, defaults to the range of each column of `X`

num_thread	a scalar positive integer indicating the number of GPUs available for calculating ALC; only relevant when method="alc"
num_multistart	a scalar positive integer indicating the number of multistart points used to optimize each inducing point with wIMSE or ALC
w_var	a scalar positive number used as the variance for the Gaussian weight in wIMSE. If NULL, theta is used.
epsK	a small positive number added to the diagonal of the correlation matrix of inducing points for numerically stability for inversion
epsQ	a small positive number added to the diagonal of the Q matrix (see Cole (2021)) for numerically stability for inversion
reps	a notification of replicate design locations in the data set. If TRUE, the unique design locations are used for the calculations along with the average response for each unique design location. Alternatively, reps can be a list from find_reps in the hetGP package. In this case, X and Y are not used.
verbose	when TRUE, prints the current number of inducing points selected during the sequential optimization process

Details

This function calls separate subroutines for certain methods. For method=wimse, the function optIP.wIMSE is called with the reference point being the median of the design matrix. If method=alc, optIP.ALC is called with the predictive variance being minimized at the median of the design matrix. For any inducing point design, the first inducing point is placed at the predictive location (i.e. the origin).

Value

The output is a list with the following components.

Xm.t	a matrix of M inducing points centered at the origin
Xn	a matrix of the local neighborhood at the center of the design
time	a scalar giving the passage of wall-clock time elapsed for (substantive parts of) the calculation

Author(s)

D. Austin Cole <austin.cole8@vt.edu>

References

D.A. Cole, R.B. Christianson, and R.B. Gramacy (2021). *Locally Induced Gaussian Processes for Large-Scale Simulation Experiments* Statistics and Computing, 31(3), 1-21; preprint on arXiv:2008.12857; https://arxiv.org/abs/2008.12857

See Also

optIP.wIMSE,optIP.ALC

Examples

```
## "1D Toy Problem"
## Test function from Forrester et al (2008);
library(hetGP)
X <- as.matrix(seq(0, 1, length=1000))
Y <- f1d(X)
int_bounds <- matrix(c(0, 1))

## Center of design space used to build inducing point templates
X_center <- median(X)


## Optimize inducing points with weighted Integrated Mean-Square Error
wimse.out <- build_ipTemplate(X, Y, N=100, M=10, method="wimse", integral_bounds=int_bounds)
Xm.t_wimse <- wimse.out$Xm.t

## now optimize inducing points using Active Learning Cohn
alc.out <- build_ipTemplate(X, Y, N=100, M=10, method="alc", integral_bounds=int_bounds)
Xm.t_alc <- alc.out$Xm.t
Xn <- alc.out$Xn ## X_center neighborhood

## plot locally optimized inducing point templates
plot(X, Y, pch=16, cex=.5, col='grey')
points(Xn, f1d(Xn), col=2)
points(Xm.t_wimse + X_center, rep(-4, 10), pch=2, col=3)
points(Xm.t_alc + X_center, rep(-5, 10), pch=3, col=4)
legend('topleft', pch = c(16, 16, 2, 3), col = c('grey', 2, 3, 4),
       legend=c('Data', 'Local neighborhood', 'wIMSE inducing point design',
                'ALC inducing point design'))


## "2D Toy Problem"
## Herbie's Tooth function used in Cole et al (2020);
## thanks to Lee, Gramacy, Taddy, and others who have used it before

## build up a design with N=~40K locations
x <- seq(-2, 2, by=0.02)
X <- as.matrix(expand.grid(x, x))
Y <- herbtooth(X)
X_center <- apply(X, 2, median)

## build a inducing point template, first with weighted Integrated Mean-Square Error
int_bounds <- rbind(c(-2,-2), c(2,2))
wimse.out <- build_ipTemplate(X, Y, N=100, M=10, method="wimse", integral_bounds=int_bounds)
Xm.t_wimse <- wimse.out$Xm.t

## now optimize inducing points using Active Learning Cohn
alc.out <- build_ipTemplate(X, Y, N=100, M=10, method="alc", integral_bounds=int_bounds)
Xm.t_alc <- alc.out$Xm.t
Xn <- alc.out$Xn
```

```
## plot locally optimized inducing point templates
plot(Xn, pch=16, cex=.5, col='grey',
     xlab = 'x1', ylab = 'x2', main='Locally optimized IP templates')
points(X_center[1], X_center[2], pch=16, cex=1.5)
points(Xm.t_wimse, pch=2, lwd=2, col=3)
points(Xm.t_alc, pch =3, lwd=2, col=4)
legend('topleft', pch = c(16, 2, 3), col = c('grey', 3, 4),
       legend=c('Local neighborhood', 'wIMSE inducing point design',
                'ALC inducing point design'))
```

`build_neighborhood` *Nearest Neighbor (NN) data subset given a center*

Description

Constructs a neighborhood of points that are a subset of the data for a given center (i.e. predictive) location.

Usage

```
build_neighborhood(N, xx = NULL, X = NULL, Y = NULL, reps_list = NULL)
```

Arguments

`N`	the positive integer number of Nearest Neighbor (NN) locations used to build a local neighborhood
`xx`	a row matrix of the location of the neighborhood's center. If `NULL`, the center (median) of the data is used
`X`	a `matrix` containing the full (large) design matrix of input locations. If `reps_list` is supplied, this entry is not used.
`Y`	a vector of responses/dependent values with `length(Y)=nrow(X)`. If `reps_list` is supplied, this entry is not used.
`reps_list`	a list from `find_reps` in the `hetGP` package, that includes the entries `X0` and `Z0`. In this case, `X` and `Y` are not used.

Details

This function builds a local neighborhood around the center `xx`. If `X` is supplied, the `N` NN points are found and chosen. If `reps_list` is supplied, `N` unique data locations `X0` are supplied, along with their averaged responses (`Z0`) and original responses (`Zlist`).

Value

The output is a `list` with the following components:

`xx`	a row `matrix` of the neighborhood's center

If `reps_list=NULL`,

`Xn`	a `matrix` of the local neighborhood's design points.
`Yn`	a `vector` of the local neighborhood's responses. Only provided when `Y` is provided.

If `reps_list` is provided,

`Xn0`	a `matrix` of the local neighborhood's unique design locations.
`Yn0`	a `vector` of averages observations at `Xn0`.
`mult`	a vector of the number of replicates at `Xn0`.
`Yn_list`	a `list` where each element corresponds to observations at a design in `Xn0`.

Author(s)

D. Austin Cole <austin.cole8@vt.edu>

References

D.A. Cole, R.B. Christianson, and R.B. Gramacy (2021). *Locally Induced Gaussian Processes for Large-Scale Simulation Experiments* Statistics and Computing, 31(3), 1-21; preprint on arXiv:2008.12857; https://arxiv.org/abs/2008.12857

Examples

```
## "2D Toy Problem"
## Herbie's Tooth function used in Cole et al (2021);
## thanks to Lee, Gramacy, Taddy, and others who have used it before
library(hetGP)

## Build data with replicates
x <- seq(-2, 2, by=0.05)
X <- as.matrix(expand.grid(x, x))
X <- rbind(X, X)
Y <- herbtooth(X) + rnorm(nrow(X), sd = .02)
reps_list <- find_reps(X, Y)
xx <- matrix(c(-0.12, 1.53), nrow=1)

## Build neighborhoods
neighborhood1 <- build_neighborhood(N=100, xx=xx, X=X, Y=Y)
neighborhood2 <- build_neighborhood(N=100, xx=xx, reps_list=reps_list)

## Compare neighborhood sizes
Xn0_range <- apply(neighborhood2$Xn0, 2, range)
plot(X, xlim = Xn0_range[,1] + c(-.15, .15), ylim = Xn0_range[,2] + c(-.1, .25),
     pch=3)
```

```
points(neighborhood2$Xn0, pch=16, col='grey')
points(neighborhood1$Xn, col=2, lwd=2)
points(xx, pch=17, col=3, cex=1.5)
legend('topleft', ncol=2, pch=c(3, 17, 16, 1), col=c(1, 3, 'grey', 2),
       legend=c('Design locations', 'Neighborhood center',
                'Xn based on unique locations', 'Xn ignoring unique locations'))
```

`calc_IMSE` *Integrated Mean-Square Error Given a New Inducing Point*

Description

Calculates the Integrated Mean-Square Error (IMSE) given a set of data points, inducing point design, and new proposed inducing point location.

Usage

```
calc_IMSE(xm1, Xm = NULL, X, theta = NULL, g = 1e-4,
          integral_bounds = NULL, epsK = sqrt(.Machine$double.eps),
          epsQ = 1e-5, mult = NULL)
```

Arguments

`xm1`	a vector containg the location of a proposed inducing point
`Xm`	optional design `matrix` of existing inducing points; `ncol(Xm) = length(xm1)`
`X`	the design `matrix` of input locations; `ncol(X) = length(xm1)`
`theta`	the lengthscale parameter (positive number) in a Gaussian correlation function; a (default) `NULL` value sets the lengthscale at the square of the 10th percentile of pairwise distances between input locations `X` (similar to `darg` in `laGP` package)
`g`	the nugget parameter (positive number) in the covariance
`integral_bounds`	a 2 by d `matrix` containing the domain bounds for the data; first row contains minimum values for each dimension, second row contains maximum values; if `integral_bounds` is `NULL`, defaults to range of the input locations `X`
`epsK`	a small positive number added to the diagonal of the correlation matrix, of inducing points, K, for numerically stability for inversion
`epsQ`	a small positive number added to the diagonal of the Q `matrix` (see Cole (2021)) for numerically stability for inversion
`mult`	an optional vector of length `nrow(X)` that contains the number of replicates for each design location in `X`

Details

The function calculates the integrated mean-square error over the provided domain (`integral_bounds`). The IMSE is calculated in closed-form.

Value

the integrated mean-sqaure error

Author(s)

D. Austin Cole <austin.cole8@vt.edu>

References

D.A. Cole, R.B. Christianson, and R.B. Gramacy (2021). *Locally Induced Gaussian Processes for Large-Scale Simulation Experiments* Statistics and Computing, 31(3), 1-21; preprint on arXiv:2008.12857; https://arxiv.org/abs/2008.12857

Examples

```
## Build a set of input locations and existing inducing points
X = matrix(runif(100), ncol=2)
Xm = matrix(runif(10), ncol=2)

integral_bounds <- rbind(c(0,0), c(1,1))
xm1_new <- c(.4, .2)

## Calculate the integrated mean-square error
calc_IMSE(xm1=xm1_new, Xm=Xm, X=X,
          integral_bounds=integral_bounds)

## without an existing inducing point design
calc_IMSE(xm1=xm1_new, Xm=NULL, X=X,
          integral_bounds=integral_bounds)
```

---

calc_wIMSE *Weighted Integrated Mean-Square Error Given a New Inducing Point*

Description

Calculates the Weighted Integrated Mean-Square Error (wIMSE) given a prediction location, local neighborhood, design of inducing points, and new proposed inducing point location.

Usage

```
calc_wIMSE(xm1, Xm = NULL, Xn, theta = NULL, g = 1e-4,
           w_mean, w_var = NULL, integral_bounds = NULL,
           epsK = sqrt(.Machine$double.eps),
           epsQ = 1e-5, mult = NULL)
```

Arguments

xm1	a vector containg the location of a proposed inducing point
Xm	a design matrix of existing inducing points; ncol(Xm) = length(xm1)
Xn	a matrix of the local neighborhood; ncol(Xn) = length(xm1)
theta	the lengthscale parameter (positive number) in a Gaussian correlation function; a (default) NULL value sets the lengthscale at the square of the 10th percentile of pairwise distances between neighborhood points (similar to darg in laGP package)
g	the nugget parameter (positive number) in the covariance
w_mean	a vector of the mean (center) of the Gaussian weight; length(w_mean) should equal ncol(Xn)
w_var	a positive number or vector of positive numbers (length equal to ncol(Xn)) denoting the variance(s) in the Gaussian weight. If NULL (default), theta is used.
integral_bounds	a 2 by d matrix containing the domain bounds for the data; first row contains minimum values for each dimension, second row contains maximum values; if integral_bounds=NULL, defaults to range of the local neighborhood Xn
epsK	a small positive number added to the diagonal of the correlation matrix, of inducing points, K, for numerically stability for inversion
epsQ	a small positive number added to the diagonal of the Q matrix (see Cole (2021)) for numerically stability for inversion
mult	a vector of length nrow(X) that contains the number of replicates for each design location in X

Details

The function calculates the integrated mean-square error with a Gaussian weight with mean w_mean (i.e. predictive location) and variance w_var. By using a Gaussian weight along with a Gaussian kernel for the GP, the wIMSE is calculated in closed-form.

Value

the weighted integrated mean-sqaure error

Author(s)

D. Austin Cole <austin.cole8@vt.edu>

References

D.A. Cole, R.B. Christianson, and R.B. Gramacy (2021). *Locally Induced Gaussian Processes for Large-Scale Simulation Experiments* Statistics and Computing, 31(3), 1-21; preprint on arXiv:2008.12857; https://arxiv.org/abs/2008.12857

Examples

```
## Build a "local neighborhood" and existing inducing point design
X_center <- c(.5, .5)
Xn <- matrix(runif(100), ncol=2)
Xm <- matrix(runif(10), ncol=2)

integral_bounds <- rbind(c(0,0), c(1,1))
xm1_new <- c(.4, .2)

## Calculate the weighted integrated mean-square error
calc_wIMSE(xm1=xm1_new,  Xm=Xm, Xn=Xn, w_mean=X_center,
           integral_bounds=integral_bounds)

## Define weight's variance
calc_wIMSE(xm1=xm1_new, Xm=Xm, Xn=Xn, w_mean=X_center,
           w_var=c(.1, .2), integral_bounds=integral_bounds)

## Without an exisiting inducing point design
calc_wIMSE(xm1=xm1_new, Xm=NULL, Xn=Xn, w_mean=X_center,
           integral_bounds=integral_bounds)
```

`giGP` *Global Inducing Point Approximate GP Regression For Many Predictive Locations*

Description

Facilitates Gaussian process inference and prediction at a set of predictive locations through the implementation of an inducing point design. Optimizes hyperparameters and returns the moments of the predictive equations.

Usage

```
giGP(XX, X = NULL, Y = NULL, Xm, g = 1e-6, theta = NULL, nu = NULL,
     epsK = sqrt(.Machine$double.eps), epsQ = 1e-5, tol = .01, reps = FALSE)
```

Arguments

`XX`	a `matrix` of out-of-sample predictive locations with `ncol(XX)=ncol(X)`
`X`	a `matrix` containing the full (large) design matrix of all input locations. If using a list for `reps`, this entry is not used.
`Y`	a vector of all responses/dependent values with `length(Y)=nrow(X)`. If using a list for `reps`, this entry is not used.
`Xm`	a `matrix` containing the inducing points design with `ncol(Xm)=ncol(X)`.
`g`	an initial setting or fixed value for the nugget parameter. In order to optimize the nugget, a list can be provided that includes: • `start` – starting value to initialize the nugget

- min – minimum value in the allowable range for the nugget
- max – maximum value in the allowable range for the nugget
- ab – shape and rate parameters specifying a Gamma prior for the nugget

If ab is not provided, a prior is not placed with the likelihood for optimization. If min and max aren't provided, the nugget is not optimized. If a single positive scalar is provided, the nugget is fixed for all predictions. If NULL, an initial setting is based on garg in the laGP package.

theta — an initial setting or fixed value for the lengthscale parameter. A (default) NULL value generates an initial setting based on darg in the laGP package. Similarly, a list can be provided that includes:

- start – starting value to initialize the lengthscale
- min – minimum value in the allowable range for the lengthscale
- max – maximum value in the allowable range for the lengthscale
- ab – shape and rate parameters specifying a Gamma prior for the lengthscale

If ab is not provided, a prior is not placed with the likelihood for optimization. If min and max aren't provided, the lengthscale is not optimized. If a single positive scalar is provided, the lengthscale is fixed for all predictions.

nu — a positive number used to set the scale parameter; default (NULL) calculates the maximum likelihood estimator

epsK — a small positive number added to the diagonal of the correlation matrix of inducing points for numerically stability for inversion. The value is automatically increased if needed.

epsQ — a small positive number added to the diagonal of the Q matrix (see Cole (2021)) for numerically stability for inversion. The value is automatically increased if needed.

tol — a positive number to serve as the tolerance level for covergence of the log-likelihood when optimizing the hyperparameter(s) theta, g

reps — a notification of replicate design locations in the data set. If TRUE, the unique design locations are used for the calculations along with the average response for each unique design location. Alternatively, reps can be a list from find_reps in the hetGP package. In this case, X and Y are not used.

Details

The function uses the likelihood and predictive equations derived in Snelson and Ghahramani (2006) to fit a induced Gaussian Process for predictions. All the data {X,Y} and inducing points Xm are used for each prediction.

Value

The output is a list with the following components.

mean — a vector of predictive means of length nrow(XX)

var — a vector of predictive variances of length nrow(XX)

nu	a vector of values of the scale parameter of length nrow(XX)
g	a full version of the g argument
theta	a full version of the theta argument
mle	if g and/or theta is optimized, a matrix containing the values found for these parameters and the number of required iterations, for each predictive location in XX
eps	a vector of the jitter values used on the correlation matrix and Q matrix
time	a scalar giving the passage of wall-clock time elapsed for (substantive parts of) the calculation

Author(s)

D. Austin Cole <austin.cole8@vt.edu>

References

D.A. Cole, R.B. Christianson, and R.B. Gramacy (2021). *Locally Induced Gaussian Processes for Large-Scale Simulation Experiments* Statistics and Computing, 31(3), 1-21; preprint on arXiv:2008.12857; https://arxiv.org/abs/2008.12857

E. Snelson Z. and Ghahramani. (2006). *Sparse Gaussian Processes using Pseudo-inputs* Advances in Neural Information Processing Systems 18 , 1257-1264.

Examples

```
## "1D Toy Problem"
## Test function from Forrester et al (2008);
library(hetGP); library(lhs)
X <- matrix(seq(0, 1, length=1000))
Y <- f1d(X)
XX <- matrix(seq(0, 1, length=100))
YY <- f1d(XX)
Xm <- randomLHS(10,1)

out <- giGP(XX=XX, X=X, Y=Y, Xm=Xm, theta=.1)
par(mfrow=c(1,2))
plot(X, Y, type='l', lwd=4, ylim=c(-8, 16))
lines(XX, out$mean, lwd=3, lty=2, col=2)
points(Xm, rep(-8, 10), lwd=2, pch=3, col=3)
legend('topleft', legend=c('Test Function', 'Predicted mean', 'Inducing Points'),
       lty=c(1, 2, NA), col=1:3, pch=c(NA, NA, 3), lwd=2)

plot(XX, YY - out$mean, ylab='Error', type = 'l')

##------------------------------------------------------------------------------##
## a "computer experiment"

## Simple 2-d Herbie's Tooth function used in Cole et al (2020);
## thanks to Lee, Gramacy, Taddy, and others who have used it before
library(lhs)
```

```
## Build up a design with N=~40K locations
x <- seq(-2, 2, by=0.05)
X <- as.matrix(expand.grid(x, x))
Y <- herbtooth(X)

## Build a inducing point template centered at origin
Xm <- 4*randomLHS(30, 2) - 2

## Predictive grid with N'=400 locations,
xx <- seq(-1.975, 1.975, length=20)
XX <- as.matrix(expand.grid(xx, xx))
YY <- herbtooth(XX)


## Get the predictive equations, first with fixed lengthscale and nugget
out <- giGP(XX=XX, X=X, Y=Y, Xm=Xm, theta=.1)
## RMSE
sqrt(mean((out$mean - YY)^2))


## Refine with optimizing the lengthscale
theta_list <- list(start = .1, min = .05, max = 5)
out2 <- giGP(XX=XX, X=X, Y=Y, Xm=Xm, theta=theta_list)
## RMSE
sqrt(mean((out2$mean - YY)^2))

## Visualize the results
orig_par <- par()
par(mfrow=c(1,3))
image(xx, xx, matrix(out2$mean, nrow=length(xx)), col=heat.colors(128),
      xlab="x1", ylab="x2", main="Predicted mean")
image(xx, xx, matrix(out2$mean-YY, nrow=length(xx)), col=heat.colors(128),
      xlab="x1", ylab="x2", main="Bias")
image(xx, xx, sqrt(out2$nu) *matrix(sqrt(out$var), nrow=length(xx)),
      col=heat.colors(128), xlab="x1", ylab="x2", main="Stand. Dev.")
points(Xm, pch=3, col=3, lwd=2)
par(orig_par)
```

---

herbtooth *Herbie's Tooth function*

Description

Two-dimensional function whose surface resembles a molar, with multiple local minima/maxima

Arguments

X	a `matrix` or `data.frame` containing the full (large) design matrix of input locations in [-2,2]^2

Details

A non-stationary function with many local minima/maxima that is difficult to model with a global model.

Value

a vector of evaluations of the Herbie's tooth function, length = `nrow(X)`

Author(s)

D. Austin Cole <austin.cole8@vt.edu>

References

H.K.H. Lee, R.B. Gramacy, C. Linkletter, and G. Gray. 2011. *Optimization Subject to Hidden Constraints via Statistical Emulation* Pacific Journal of Optimization 7 (3): 467-78.

Examples

```
X <- matrix(runif(200, min = -2, max = 2), ncol=2)
herbtooth(X)
```

`liGP` *Localized Inducing Point Approximate GP Regression For Many Predictive Locations*

Description

Facilitates locally induced Gaussian process inference and prediction at a large set of predictive locations by: building local neighborhoods, shifting an inducing point template, optimizing hyperparameters, and calculating GP predictive equations.

Usage

```
liGP(XX, X = NULL, Y = NULL, Xm.t, N, g = 1e-6, theta = NULL,
     nu = NULL, num_thread = 1, epsK = sqrt(.Machine$double.eps), epsQ = 1e-5,
     tol = .01, reps = FALSE, Xni.return = FALSE)
```

Arguments

`XX`	a `matrix` of out-of-sample predictive locations with `ncol(XX)` = `ncol(X)`
`X`	a `matrix` containing the full (large) design matrix of all input locations. If `reps` is a list, this entry is not used.
`Y`	a vector of all responses/dependent values with `length(Y)=nrow(X)`. If `reps` is a list, this entry is not used.
`Xm.t`	a `matrix` containing the `M` inducing points template with `ncol(Xm.t)` = `ncol(X)`. See 'Note' for more.

N — the positive integer number of nearest neighbor (NN) locations used to build a local neighborhood; N should be greater than M. See 'Note' for more.

g — an initial setting or fixed value for the nugget parameter. In order to optimize g, a list can be provided that includes:

- start – starting value to initialize the nugget
- min – minimum value in the allowable range for the nugget
- max – maximum value in the allowable range for the nugget
- ab – shape and rate parameters specifying a Gamma prior for the nugget

If ab is not provided, a prior is not placed with the likelihood for optimization. If min and max aren't provided, the nugget is not optimized. A NULL value generates a list based on garg in the **laGP** package. If a single positive scalar is provided, the nugget is fixed for all predictions. Alternatively, a vector of nuggets whose length equals nrow(XX) can be provided to fix distinct nuggets for each prediction.

theta — an initial setting or fixed value for the lengthscale parameter. A (default) NULL value generates an initial setting based on darg in the **laGP** package. Similarly, a list can be provided that includes:

- start – starting value to initialize the lengthscale
- min – minimum value in the allowable range for the lengthscale
- max – maximum value in the allowable range for the lengthscale
- ab – shape and rate parameters specifying a Gamma prior for the lengthscale

If ab is not provided, a prior is not placed with the likelihood for optimization. If min and max aren't provided, the lengthscale is not optimized. If a single positive scalar is provided, the lengthscale is fixed for all predictions. Alternatively, a vector of lengthscales whose length equals nrow(XX) can be provided to fix distinct lengthscales for each prediction.

nu — a positive number used to set the scale parameter; default (NULL) calculates the maximum likelihood estimator

num_thread — a scalar positive integer indicating the number of threads to use for parallel processing

epsK — a small positive number added to the diagonal of the correlation matrix of inducing points for numerically stability for inversion. It is automatically increased if neccessary for each prediction.

epsQ — a small positive number added to the diagonal of the Q matrix (see Cole (2021)) of inducing points for numerically stability for inversion. It is automatically increased if neccessary for each prediction.

tol — a positive number to serve as the tolerance level for covergence of the log-likelihood when optimizing the hyperparameter(s) theta, g

reps — a notification of replicate design locations in the data set. If TRUE, the unique design locations are used for the calculations along with the average response for each unique design location. Alternatively, reps can be a list from find_reps in the **hetGP** package. In this case, X and Y are not used.

Xni.return — A scalar logical indicating whether or not a vector of indices into X (or X0 if a reps list is supplied), specifying the chosen sub-design, should be returned on output.

Details

When num_threads > 1, the predictions are performed in parallel using foreach with clusters created by **parallel**.

Value

The output is a list with the following components:

mean	a vector of predictive means of length nrow(XX)
var	a vector of predictive variances of length nrow(XX)
nu	a vector of values of the scale parameter of length nrow(XX)
g	a full version of the g argument
theta	a full version of the theta argument
Xm.t	the input for Xm.t
eps	a matrix of epsK and epsQ (jitter) values used for each prediction, nrow(eps)=nrow(XX)
mle	if g and/or theta is optimized, a matrix containing the values found for these parameters and the number of required iterations, for each predictive location in XX
Xni	when Xni.return = TRUE, this field contains a vector of indices of length N into X (or X0) indicating the sub-design (neighborhood) chosen. If nrow(XX)>1, a matrix is returned with each row matched with the corresponding row of XX
time	a scalar giving the passage of wall-clock time elapsed for (substantive parts of) the calculation

Note

When selecting the neighborhood size (N) and number of inducing points in Xm.t, there is no general rule that works for all problems. However, for lower dimensions (dim<9) the following values seem to perform well: N = 100 + 10*dim, M = 10*dim

Author(s)

D. Austin Cole <austin.cole8@vt.edu>

References

D.A. Cole, R.B. Christianson, and R.B. Gramacy (2021). *Locally Induced Gaussian Processes for Large-Scale Simulation Experiments* Statistics and Computing, 31(3), 1-21; preprint on arXiv:2008.12857; https://arxiv.org/abs/2008.12857

See Also

darg, garg, find_reps, makeCluster, clusterApply

Examples

```
## "1D Toy Problem"
## Test function from Forrester et al (2008);
library(hetGP); library(lhs)
X <- matrix(seq(0, 1, length=1000))
Y <- f1d(X)
XX <- matrix(seq(0, 1, length=100))
YY <- f1d(XX)

## Create inducing point template
lhs_design <- randomLHS(9,1)
n <- 80
Xmt <- scale_ipTemplate(X, n, space_fill_design=lhs_design, method='qnorm')$Xm.t

out <- liGP(XX=XX, X=X, Y=Y, Xm=Xmt, N=n, theta=.1)

## Plot predicted mean and error
orig_par <- par()
par(mfrow=c(1,2))
plot(X, Y, type='l', lwd=4, ylim=c(-8, 16),
     main='LIGP fit to Test Function')
lines(XX, out$mean, lwd=3, lty=2, col=2)
legend('topleft', legend=c('Test Function', 'Predicted mean'),
       lty=1:2, col=1:2, lwd=2)

plot(XX, YY - out$mean, xlab='X', ylab='Error', type = 'l',
     main='Predicted Error')
par(orig_par)

##
## Generate new data from function with same mean and non-constant noise
fY <- function(x) { f1d(x) + rnorm(length(x), sd=(1.1 + sin(2*pi*x))) }
Y2 <- fY(X)

## Estimate lengthscale and nugget in predictions
library(laGP)
theta_prior <- darg(NULL, X)
g_prior <- garg(list(mle=TRUE), Y2)
out2 <- liGP(XX=XX, X=X, Y=Y2, Xm=Xmt, N=n, theta=theta_prior,
             g=g_prior, epsK=1e-5)

## Plot predictived mean and confidence intervals
plot(X, Y2, col='grey', cex=.5, pch=16,
     main='LIGP fit to heteroskedastic data', ylab='Y')
lines(X, Y, lwd=2)
lines(XX, out2$mean, lwd=2, lty=2, col=2)
lines(XX, out2$mean + 1.96*sqrt(out2$nu*out2$var), lwd=1, lty=4, col=2)
lines(XX, out2$mean - 1.96*sqrt(out2$nu*out2$var), lwd=1, lty=4, col=2)
legend('topleft', legend=c('Noisy data','Function mean', 'Predicted mean',
                           'Predicted 95 percent confidence interval'),
       lwd=2, lty=c(NA,1,2,3), pch=c(16,NA,NA,NA), col=c('grey',1,2,2))
```

```
## View mean and variance errors
par(mfrow=c(1,2))
plot(XX, YY - out2$mean, xlab='X', ylab='Mean Error', type = 'l')
plot(XX, (1.1 + sin(2*pi*XX))^2 - (out2$nu*out2$var),
     xlab='X', ylab='Variance Error', type = 'l')
par(orig_par)

##
## Generate new data with replicates
mults <- sample(2:10, nrow(X), replace=TRUE)
X.reps <- X[rep(1:nrow(X), mults),]
Y.reps <- fY(X.reps)
g_prior <- garg(list(mle=TRUE), Y.reps)

## Generate rep list from hetGP
rep_list <- find_reps(X.reps, Y.reps)

out3 <- liGP(XX=XX, Xm=Xmt, N=n, theta=theta_prior,
             g=g_prior, epsK=1e-5, reps = rep_list)


## Plot predictived mean and confidence intervals
plot(X.reps, Y.reps, col='grey', cex=.5, pch=16,
     main='LIGP fit to data with replicates', xlab='X', ylab='Y')
lines(X, Y, lwd=2)
lines(XX, out3$mean, lwd=2, lty=2, col=2)
lines(XX, out3$mean + 1.96*sqrt(out3$nu*out3$var), lwd=1, lty=4, col=2)
lines(XX, out3$mean - 1.96*sqrt(out3$nu*out3$var), lwd=1, lty=4, col=2)
legend('topleft', legend=c('Noisy data','Function mean', 'Predicted mean',
                           'Predicted 95 percent confidence interval'),
       lwd=2, lty=c(NA,1,2,3), pch=c(16,NA,NA,NA), col=c('grey',1,2,2))


##------------------------------------------------------------------------------##
## a "computer experiment"

## Simple 2-d Herbie's Tooth function used in Cole et al (2021);
## thanks to Lee, Gramacy, Taddy, and others who have used it before

## Build up a design with N=~40K locations
x <- seq(-2, 2, by=0.02)
X <- as.matrix(expand.grid(x, x))
Y <- herbtooth(X)

## Build a inducing point template centered at origin
Xm <- matrix(runif(20), ncol=2)
Xmt <- scale_ipTemplate(X=X, N=100, method="chr", space_fill_design = Xm)$Xm.t

## Predictive grid with N'=400 locations
xx <- seq(-1.975, 1.975, length=20)
XX <- as.matrix(expand.grid(xx, xx))
YY <- herbtooth(XX)
```

```
## Get the predictive equations, first with fixed lengthscale and nugget
out <- liGP(XX=XX, X=X, Y=Y, Xm.t=Xmt, N=100, Xni.return=TRUE)
## RMSE
sqrt(mean((out$mean - YY)^2))

## View one local neighborhood
xylim <- apply(X[out$Xni[33,],], 2, range)
plot(X[,1], X[,2], pch=16, col='grey', cex=.5,
     xlim=xylim[,1] + c(-.05, .05), ylim=xylim[,2] + c(-.05, .05),
     xlab='X1', ylab='X2')
points(X[out$Xni[33,],1], X[out$Xni[33,],2], pch=16)
points(XX[33,1], XX[33,2], col=3, pch=17, cex=1.5)
points(sweep(Xmt, 2, XX[33,,drop=FALSE], '+'), pch=18, col=2)
legend('topleft', legend=c('Predictive location', 'Data not in neighborhoood',
                           'Neighborhood', 'Inducing points'),
       pch=c(17, 16, 16, 18), col=c(3, 'grey',1,2), cex=1.3)



##
## Refine with optimizing the lengthscale
theta_list <- darg(NULL, X)
out2 <- liGP(XX=XX, X=X, Y=Y, Xm.t=Xmt, N=100, theta=theta_prior)

## RMSE
sqrt(mean((out2$mean - YY)^2))

## Visualize the results
par(mfrow=c(1,3))
image(xx, xx, matrix(out2$mean, nrow=length(xx)), col=heat.colors(128),
      xlab="x1", ylab="x2", main="Predictive Mean")
image(xx, xx, matrix(out2$mean-YY, nrow=length(xx)), col=heat.colors(128),
      xlab="x1", ylab="x2", main="Bias")
image(xx, xx, matrix(sqrt(out2$nu*out$var), nrow=length(xx)),
      col=heat.colors(128), xlab="x1", ylab="x2", main="Stand. Dev.")
par(orig_par)

##
## Predictions from noisy training data with replicates
Xreps <- X[rep(1:nrow(X), 5),]
Ynoisy <- herbtooth(Xreps) + rnorm(nrow(Xreps), sd=.02)

library(hetGP)
reps_list <- find_reps(Xreps, Ynoisy)

## Priors for theta and g
theta_prior <- darg(NULL, Xreps)
g_prior <- garg(list(mle=TRUE), Ynoisy)

## Predictions with estimated nugget
out_noisydata <- liGP(XX, Xm.t = Xmt, N = 100, g=g_prior, theta=theta_prior,
                      reps = reps_list)
```

```
## RMSE
estimated_noise <- sqrt(out_noisydata$mle[,1]*out_noisydata$nu)
sqrt(mean((estimated_noise - .02)^2))


## Visualize the results
par(mfrow=c(1,3))
image(xx, xx, matrix(out_noisydata$mean, nrow=length(xx)), col=heat.colors(128),
      xlab="x1", ylab="x2", main="Predictive Mean")
image(xx, xx, matrix(out_noisydata$mean-YY, nrow=length(xx)), col=heat.colors(128),
      xlab="x1", ylab="x2", main="Bias")
image(xx, xx, matrix(sqrt(out_noisydata$nu*out_noisydata$var) - .02, nrow=length(xx)),
      col=heat.colors(128), xlab="x1", ylab="x2", main="Stand. Dev. Error")
par(orig_par)
```

`liGP.forloop` *Localized Inducing Point Approximate GP Regression For Many Predictive Locations*

Description

Facilitates locally induced Gaussian process inference and prediction at a large set of predictive locations by: building local neighborhoods, shifting an inducing point template, optimizing hyperparameters, and calculating GP predictive equations.

Usage

```
liGP.forloop(XX, X = NULL, Y = NULL, Xm.t, N, g = 1e-6, theta = NULL,
     nu = NULL, epsK = sqrt(.Machine$double.eps), epsQ = 1e-5,
     tol = .01, reps = FALSE, Xni.return = FALSE)
```

Arguments

`XX`	a `matrix` of out-of-sample predictive locations with `ncol(XX)` = `ncol(X)`
`X`	a `matrix` containing the full (large) design matrix of all input locations. If `reps` is a list, this entry is not used.
`Y`	a vector of all responses/dependent values with `length(Y)=nrow(X)`. If `reps` is a list, this entry is not used.
`Xm.t`	a `matrix` containing the `M` inducing points template with `ncol(Xm.t)` = `ncol(X)`. See 'Note' for more.
`N`	the positive integer number of nearest neighbor (NN) locations used to build a local neighborhood; `N` should be greater than `M`. See 'Note' for more.
`g`	an initial setting or fixed value for the nugget parameter. In order to optimize g, a list can be provided that includes: • `start` – starting value to initialize the nugget

- `min` – minimum value in the allowable range for the nugget
- `max` – maximum value in the allowable range for the nugget
- `ab` – shape and rate parameters specifying a Gamma prior for the nugget

If `ab` is not provided, a prior is not placed with the likelihood for optimization. If `min` and `max` aren't provided, the nugget is not optimized. A `NULL` value generates a list based on `garg` in the **laGP** package. If a single positive scalar is provided, the nugget is fixed for all predictions. Alternatively, a vector of nuggets whose length equals `nrow(XX)` can be provided to fix distinct nuggets for each prediction.

`theta` an initial setting or fixed value for the lengthscale parameter. A (default) `NULL` value generates an initial setting based on `darg` in the **laGP** package. Similarly, a list can be provided that includes:

- `start` – starting value to initialize the lengthscale
- `min` – minimum value in the allowable range for the lengthscale
- `max` – maximum value in the allowable range for the lengthscale
- `ab` – shape and rate parameters specifying a Gamma prior for the lengthscale

If `ab` is not provided, a prior is not placed with the likelihood for optimization. If `min` and `max` aren't provided, the lengthscale is not optimized. If a single positive scalar is provided, the lengthscale is fixed for all predictions. Alternatively, a vector of lengthscales whose length equals `nrow(XX)` can be provided to fix distinct lengthscales for each prediction.

`nu` a positive number used to set the scale parameter; default (`NULL`) calculates the maximum likelihood estimator

`epsK` a small positive number added to the diagonal of the correlation `matrix` of inducing points for numerically stability for inversion. It is automatically increased if neccessary for each prediction.

`epsQ` a small positive number added to the diagonal of the Q `matrix` (see Cole (2021)) of inducing points for numerically stability for inversion. It is automatically increased if neccessary for each prediction.

`tol` a positive number to serve as the tolerance level for covergence of the log-likelihood when optimizing the hyperparameter(s) theta, g

`reps` a notification of replicate design locations in the data set. If `TRUE`, the unique design locations are used for the calculations along with the average response for each unique design location. Alternatively, `reps` can be a list from `find_reps` in the **hetGP** package. In this case, `X` and `Y` are not used.

`Xni.return` A scalar logical indicating whether or not a vector of indices into `X` (or `X0` if a reps list is supplied), specifying the chosen sub-design, should be returned on output.

Value

The output is a `list` with the following components:

`mean` a vector of predictive means of length `nrow(XX)`

var	a vector of predictive variances of length nrow(XX)
nu	a vector of values of the scale parameter of length nrow(XX)
g	a full version of the g argument
theta	a full version of the theta argument
Xm.t	the input for Xm.t
eps	a matrix of epsK and epsQ (jitter) values used for each prediction, nrow(eps)=nrow(XX)
mle	if g and/or theta is optimized, a matrix containing the values found for these parameters and the number of required iterations, for each predictive location in XX
Xni	when Xni.return = TRUE, this field contains a vector of indices of length N into X (or X0) indicating the sub-design (neighborhood) chosen. If nrow(XX)>1, a matrix is returned with each row matched with the corresponding row of XX
time	a scalar giving the passage of wall-clock time elapsed for (substantive parts of) the calculation

Note

When selecting the neighborhood size (N) and number of inducing points in Xm.t, there is no general rule that works for all problems. However, for lower dimensions (dim<9) the following values seem to perform well: N = 100 + 10*dim, M = 10*dim

Author(s)

D. Austin Cole <austin.cole8@vt.edu>

References

D.A. Cole, R.B. Christianson, and R.B. Gramacy (2021). *Locally Induced Gaussian Processes for Large-Scale Simulation Experiments* Statistics and Computing, 31(3), 1-21; preprint on arXiv:2008.12857; https://arxiv.org/abs/2008.12857

See Also

darg, garg, find_reps, makeCluster, clusterApply

Examples

```
## See LIGP examples
```

`liGP_gauss_measure` *Localized Inducing Point Approximate GP Regression For a Gaussian Measure*

Description

Facilitates locally induced Gaussian process inference and prediction across a Gaussian measure by: building one local neighborhood around the measure, shifting an inducing point template, optimizing hyperparameters, calculating GP mean predictions, and estimating the integral through a discrete set or quadrature.

Usage

```
liGP_gauss_measure(xstar, X, Y, Xm.t, N, gauss_sd, measure_bounds = c(-Inf, Inf),
      g = 1e-6, epsi = NULL, epsK = 1e-6, epsQ = 1e-5, seq_length = 20)
```

Arguments

`xstar`	a one-row `matrix` of the mean of the Gaussian measure.
`X`	a `matrix` containing the full (large) design matrix of all input locations.
`Y`	a vector of all responses/dependent values with `length(Y)=nrow(X)`.
`Xm.t`	a `matrix` containing the `M` inducing points template with `ncol(Xmt) = ncol(X)`.
`N`	the positive integer number of nearest neighbor (NN) locations used to build a local neighborhood; `N` should be greater than `M`
`gauss_sd`	a vector of standard deviations for the Gaussian measure with with `length(gauss_sd)=nrow(X)`. Note: at this time, the Gaussian measure must only have one nonzero standard deviation (i.e. the Gaussian measure is a slice).
`measure_bounds`	a vector of the bounds of the Gaussian measure for the single dimension with a nonzero standard deviation. This is only used if `epsi` is `NULL`.
`g`	a fixed value for the nugget parameter.
`epsi`	an optional vector of Gaussian noise drawn from gauss_sd used with `xstar` to generate a set of predictive locations for estimating the integral. If not provided, the `integrate` function is called to perform to estimate the integral.
`epsK`	a small positive number added to the diagonal of the correlation `matrix` of inducing points for numerically stability for inversion
`epsQ`	a small positive number added to the diagonal of the Q `matrix` (see Cole (2021)) of inducing points for numerically stability for inversion
`seq_length`	a positive integer used to build sequences of this length in the nondegenerate dimension for the purpose of building a local neighbhorhood. This sequence is not used in prediction.

Details

This function is built to deal with the special class of problems where liGP is used to predict and integrate over a degenerate Gaussian measure where only one dimension has a nonzero standard deviation.

Value

the pointwise estimate for the mean prediction over the Gaussian measure

Author(s)

D. Austin Cole <austin.cole8@vt.edu>

References

D.A. Cole, R.B. Christianson, and R.B. Gramacy (2021). *Locally Induced Gaussian Processes for Large-Scale Simulation Experiments* Statistics and Computing, 31(3), 1-21; preprint on arXiv:2008.12857; https://arxiv.org/abs/2008.12857

See Also

darg, integrate

Examples

```
##-----------------------------------------------------------------------------##
## a "computer experiment"

## Simple 2-d Herbie's Tooth function used in Cole et al (2020);
## thanks to Lee, Gramacy, Taddy, and others who have used it before

## Build up a design with N=~40K locations
x <- seq(-2, 2, by=0.02)
X <- as.matrix(expand.grid(x, x))
Y <- herbtooth(X)

## Build a inducing point template centered at origin
X_m <- matrix(runif(20), ncol=2)
Xmt <- scale_ipTemplate(X, N=100, space_fill_design=X_m, method="qnorm")$Xm.t


## predictive center
xx <- matrix(c(.5, .5), ncol=2)

## Standard deviation of gaussian measure with random draws
gauss_sd <- c(0, .1)
epsi <- rnorm(30, sd = gauss_sd[2])

## Get the predictive equations, first with fixed lengthscale and nugget
out <- liGP_gauss_measure(xx, X=X, Y=Y, Xm.t=Xmt, N=100,
                          gauss_sd=gauss_sd, epsi=epsi)


## Refine with using integrate function
out2 <- liGP_gauss_measure(xx, X=X, Y=Y, Xm.t=Xmt, N=100, gauss_sd=gauss_sd)
```

loiGP *Locally Optimized Inducing Point Approximate GP Regression For Many Predictive Locations*

Description

Facilitates localized Gaussian process inference and prediction at a large set of predictive locations, by opimizing a local set of inducing points for each predictive location's local neighborhood and then calling giGP.

Usage

```
loiGP(XX, X = NULL, Y = NULL, M, N, g = 1e-6, theta = NULL, nu = NULL,
      method = c('wimse','alc'), integral_bounds = NULL, num_thread = 1,
      epsK = sqrt(.Machine$double.eps), epsQ = 1e-5, tol = .01, reps = FALSE)
```

Arguments

XX	a matrix of out-of-sample predictive locations with ncol(XX) = ncol(X); loiGP calls giGP for each row of XX, independently
X	a matrix containing the full (large) design matrix of all input locations. If reps is a list, this entry is not used.
Y	a vector of all responses/dependent values with length(Y)=nrow(X). If reps is a list, this entry is not used.
M	the positive integer number of inducing points placed for each local neighborhood; M should be less than N
N	the positive integer number of Nearest Neighbor (NN) locations used to build a local neighborhood
g	an initial setting or fixed value for the nugget parameter. In order to optimize g, a list can be provided that includes: • start – starting value to initialize the nugget • min – minimum value in the allowable range for the nugget • max – maximum value in the allowable range for the nugget • ab – shape and rate parameters specifying a Gamma prior for the nugget If ab is not provided, a prior is not placed with the likelihood for optimization. If min and max aren't provided, the nugget is not optimized. A NULL value generates an initial setting based on garg in the laGP package. If a single positive scalar is provided, the nugget is fixed for all predictions. Alternatively, a vector of nuggets whose length equals nrow(XX) can be provided to fix distinct nuggets for each prediction.
theta	an initial setting or fixed value for the lengthscale parameter. A (default) NULL value generates an initial setting based on darg in the laGP package. Similarly, a list can be provided that includes: • start – starting value to initialize the lengthscale

- min – minimum value in the allowable range for the lengthscale
- max – maximum value in the allowable range for the lengthscale
- ab – shape and rate parameters specifying a Gamma prior for the lengthscale

If ab is not provided, a prior is not placed with the likelihood for optimization. If min and max aren't provided, the lengthscale is not optimized. If a single positive scalar is provided, the lengthscale is fixed for all predictions. Alternatively, a vector of lengthscales whose length equals nrow(XX) can be provided to fix distinct lengthscales for each prediction.

nu — a positive number used to set the scale parameter; default (NULL) calculates the maximum likelihood estimator

method — specifies the method by which the inducing point template is built. In brief, wIMSE ("wimse", default) minimizes the weighted integrated mean-sqaure error, and ALC ("alc") minimizes predictive variance at the preditive location.

integral_bounds — a 2 by d matrix of the domain bounds of the data (used in the calculation of wimse); the first row contains minimum values, the second row the maximum values; only relevant when method="wimse"; if not provided, defaults to the range of each column of X

num_thread — a scalar positive integer indicating the number of threads to use for parallel processing

epsK — a small positive number added to the diagonal of the correlation matrix, of inducing points, K, for numerically stability for inversion. It is automatically increased if neccessary for each prediction.

epsQ — a small positive number added to the diagonal of the Q matrix (see Cole (2021)) for numerically stability for inversion. It is automatically increased if neccessary for each prediction.

tol — a positive number to serve as the tolerance level for covergence of the log-likelihood when optimizing the hyperparameter(s) theta and/or g

reps — a notification of replicate design locations in the data set. If TRUE, the unique design locations are used for the calculations along with the average response for each unique design location. Alternatively, reps can be a list from find_reps in the hetGP package. In this case, X and Y are not used.

Details

This function builds a unique inducing point design to accompany the local neighborhood for each preditive location in XX. It then invokes giGP for each row of XX with X=Xn, Y=Yn from the corresponding local neighborhood and locally optimial inducing point design. For further information, see giGP.

Value

The output is a list with the following components:

mean — a vector of predictive means of length nrow(XX)

var	a vector of predictive variances of length nrow(XX)
nu	a vector of values of the scale parameter of length nrow(XX)
g	a full version of the g argument
theta	a full version of the theta argument
Xm	a list of inducing point designs; each entry in the list is a matrix containing M locally optimized inducing points; length(Xm)=nrow(XX)
eps	a matrix of epsK and epsQ (jitter) values used for each prediction, nrow(eps)=nrow(XX)
mle	if g and/or theta is optimized, a matrix containing the values found for these parameters and the number of required iterations, for each predictive location in XX
time	a scalar giving the passage of wall-clock time elapsed for (substantive parts of) the calculation

Author(s)

D. Austin Cole <austin.cole8@vt.edu>

References

D.A. Cole, R.B. Christianson, and R.B. Gramacy (2021). *Locally Induced Gaussian Processes for Large-Scale Simulation Experiments* Statistics and Computing, 31(3), 1-21; preprint on arXiv:2008.12857; https://arxiv.org/abs/2008.12857

Examples

```
library(hetGP); library(lhs)
X <- matrix(seq(0, 1, length=1000))
Y <- f1d(X)
XX <- matrix(seq(.01, .99, length=50))
YY <- f1d(XX)


n <- 50
m <- 7
int_bounds <- matrix(c(0,1))

out <- loiGP(XX=XX, X=X, Y=Y, M=m, N=n, method='wimse',
             integral_bounds=int_bounds)

## Plot predicted mean and error
orig_par <- par()
par(mfrow=c(1,2))
plot(X, Y, type='l', lwd=4, ylim=c(-8, 16))
lines(XX, out$mean, lwd=3, lty=2, col=2)
legend('topleft', legend=c('Test Function', 'Predicted mean'),
       lty=1:2, col=1:2, lwd=2)

plot(XX, YY - out$mean, xlab='X', ylab='Error', type = 'l')
par(orig_par)
```

`optIP.ALC` *Sequential Selection of an Inducing Point Design by Optimizing Active Learning Cohn*

Description

Optimizes the ALC surface to sequentially select inducing points for a given predictive location and local neighborhood. ALC can be based solely on the predictive location or an additional set of reference locations.

Usage

```
optIP.ALC(Xc, Xref = NULL, M, Xn, Yn, theta = NULL, g = 1e-4,
          ip_bounds = NULL, num_thread = 1, num_multistart = 15,
          epsK = sqrt(.Machine$double.eps), epsQ = 1e-5, tol = .01,
          rep_list = NULL, verbose = TRUE)
```

Arguments

`Xc`	a vector containing the predictive location used as the center of the design/neighborhood
`Xref`	a `matrix` containing other reference locations used in the predictive variance sum that is minimized
`M`	the positive integer number of inducing points placed for each local neighborhood; `M` should be less than `N`
`Xn`	a `matrix` of the local neighborhood of `N` nearest neighbors to `Xc`
`Yn`	a vector of the corresponding responses to `Xn`; `length(Yn)=nrow(Xn)`
`theta`	the lengthscale parameter (positive number) in a Gaussian correlation function; a (default) `NULL` value sets the lengthscale at the square of the 10th percentile of pairwise distances between neighborhood points (see `darg` in `laGP` package)
`g`	the nugget parameter (positive number) in the covariance
`ip_bounds`	a 2 by d `matrix` containing the range of possible values for inducing points; first row contains minimum values for each dimension, second row contains maximum values; if `ip_bounds` is `NULL`, defaults to range of the local neighborhood `Xn`
`num_thread`	a scalar positive integer indicating the number of threads to use for parallel processing for the multistart search: `num_thread<=num_multistart`
`num_multistart`	a positive integer indicating the number of starting locations used to optimize the ALC and find the global minimum
`epsK`	a small positive number added to the diagonal of the correlation matrix, of inducing points, K, for numerically stability for inversion
`epsQ`	a small positive number added to the diagonal of the Q `matrix` (see Cole (2021)) for numerically stability for inversion
`tol`	a positive number to serve as the tolerance level for covergence of ALC when optimizing the location of the next inducing point

rep_list	a list from find_reps in the hetGP package that details the replicates in Xn and their associated Yn
verbose	if TRUE, outputs the progress of the number of inducing points optimally placed

Details

The function sequentially places M inducing points around the local neighborhood (Xn) of Xc. The first inducing point is placed at Xc. The remaining points and placed based on the minimum in the ALC surface using rbind(Xc, Xref) as a reference set for the predictive variance. Hyperparameters theta,g are fixed.

Value

The output is a list with the following components.

Xm	a matrix of the locally optimal inducing point locations
alc	a vector of the ALC progress at each inducing point selection step
time	a scalar giving the passage of wall-clock time elapsed for (substantive parts of) the calculation

Author(s)

D. Austin Cole <austin.cole8@vt.edu>

References

D.A. Cole, R.B. Christianson, and R.B. Gramacy (2021). *Locally Induced Gaussian Processes for Large-Scale Simulation Experiments* Statistics and Computing, 31(3), 1-21; preprint on arXiv:2008.12857; https://arxiv.org/abs/2008.12857

S. Seo, M. Wallat, T. Graepel, and K. Obermayer (2000). *Gaussian Process Regression: Active Data Selection and Test Point Rejection* In Mustererkennung 2000, 27-34. New York, NY: Springer-Verlag.

Examples

```
## a "computer experiment"

## Simple 2-d Herbie's Tooth function used in Cole et al (2020);
## thanks to Lee, Gramacy, Taddy, and others who have used it before

## Build up a design with N=~40K locations
x <- seq(-2, 2, by=0.02)
X <- as.matrix(expand.grid(x, x))
Y <- herbtooth(X)


library(laGP)

## Build a local neighborhood
Xstar <- matrix(c(.4, -1.1), nrow=1)
```

```
n <- 100; m <- 10
Xstar_to_X_dists <- distance(Xstar, X)
quant_dists <- quantile(Xstar_to_X_dists, n/nrow(X))
Xn <- X[Xstar_to_X_dists < quant_dists,]
Yn <- Y[Xstar_to_X_dists < quant_dists]

theta <- darg(NULL, Xn)$start

## Optimize inducing point locations
Xm.alc <- optIP.ALC(Xstar, Xref=NULL, M=m, Xn=Xn, Yn=Yn, theta=theta)

## Define reference locations for ALC sum
Xref <- as.matrix(expand.grid(Xstar[1]+c(-.05, 0, .05), Xstar[2]+c(-.05, 0, .05)))
Xm.alc_with_Xref <- optIP.ALC(Xstar, Xref=Xref, M=m, Xn=Xn, Yn=Yn, theta=theta)

## Plot inducing point design and neighborhood
ranges <- apply(Xn, 2, range)
plot(Xn, pch = 16, cex=.5, col='grey',
     xlim=ranges[,1]+c(-.02, .02), ylim=ranges[,2]+c(-.02, .15),
     xlab='x1', ylab = 'x2',
     main='ALC-optimal Inducing Point Design')
points(Xstar[1], Xstar[2], pch=16)
points(Xm.alc$Xm, pch=2, col=3, lwd=2)
points(Xm.alc_with_Xref$Xm, pch=3, col=4, lwd=2)
legend('topleft', col=c(1,'grey',3,4), pch=c(16,16,2,3), lty=NA, lwd=2, ncol=2,
       legend=c('Xstar','Neighborhood','Xm based on Xstar','Xm based on Xref'))

## View  ALC progress
plot(1:m, Xm.alc$alc, type='l', xlab='Inducing point number',
     ylab='ALC',main='ALC optimization progress')
```

---

optIP.wIMSE *Sequential Selection of an Inducing Point Design by Optimizing Weighted Integrates Mean-Sqaure Error*

Description

Optimizes the weighted integrated mean-square error (wIMSE) surface to sequentially select inducing points for a given predictive location and local neighborhood.

Usage

```
optIP.wIMSE(Xn, M, theta = NULL, g = 1e-4, w_mean, w_var = NULL,
            ip_bounds = NULL, integral_bounds = NULL, num_multistart = 15,
            fix_xm1 = TRUE, epsK = sqrt(.Machine$double.eps), epsQ = 1e-5,
            mult = NULL, verbose = TRUE)
```

Arguments

Xn	a matrix of the local neighborhood; nrow(Xn)=N
M	the positive integer number of inducing points placed for each local neighborhood; M should be less than N
theta	the lengthscale parameter (positive number) in a Gaussian correlation function; a (default) NULL value sets the lengthscale at the square of the 10th percentile of pairwise distances between neighborhood points (see darg in laGP package)
g	the nugget parameter (positive number) in the covariance
w_mean	a vector of the mean (center) of the Gaussian weight; length(w_mean) should equal ncol(Xn)
w_var	a positive number or vector of positive numbers (length equal to ncol(Xn)) denoting the variance(s) in the Gaussian weight. If NULL (default), theta is used.
ip_bounds	a 2 by d matrix containing the range of possible values for inducing points; first row contains minimum values for each dimension, second row contains maximum values; if ip_bounds is NULL, defaults to range of the local neighborhood Xn
integral_bounds	a 2 by d matrix containing the domain bounds for the data; first row contains minimum values for each dimension, second row contains maximum values; if integral_bounds=NULL, defaults to range of the local neighborhood Xn
num_multistart	a positive integer indicating the number of starting locations used to optimize wIMSE for each inducing point
fix_xm1	an indicator of whether or not the first inducing point should be fixed at w_mean (TRUE, default) or optimized (FALSE)
epsK	a small positive number added to the diagonal of the correlation matrix, of inducing points, K, for numerically stability for inversion
epsQ	a small positive number added to the diagonal of the Q matrix (see Cole (2021)) for numerically stability for inversion
mult	a vector of length nrow(X) that contains the number of replicates for each design location in X
verbose	if TRUE, outputs the progress of the number of inducing points optimally placed

Details

The function sequentially places M inducing points around the local neighborhood (Xn). Inducing points are placed based on the minimum in the wIMSE surface integrating over integral_bounds. Hyperparameters theta, g are fixed.

Value

The output is a list with the following components:

Xm	a matrix of the locally optimal inducing point locations
wimse	a vector of the wIMSE progress at each inducing point selection step
time	a scalar giving the passage of wall-clock time elapsed for (substantive parts of) the calculation

Author(s)

D. Austin Cole <austin.cole8@vt.edu>

References

D.A. Cole, R.B. Christianson, and R.B. Gramacy (2021). *Locally Induced Gaussian Processes for Large-Scale Simulation Experiments* Statistics and Computing, 31(3), 1-21; preprint on arXiv:2008.12857; https://arxiv.org/abs/2008.12857

Examples

```
## a "computer experiment"

## Simple 2-d Herbie's Tooth function used in Cole et al (2020);
## thanks to Lee, Gramacy, Taddy, and others who have used it before

## Build up a design with N=~40K locations
x <- seq(-2, 2, by=0.02)
X <- as.matrix(expand.grid(x, x))
Y <- herbtooth(X)


library(laGP)

## Build a local neighborhood
Xstar <- matrix(c(.4, -1.1), nrow=1)
n <- 100; m <- 10
Xstar_to_X_dists <- distance(Xstar, X)
quant_dists <- quantile(Xstar_to_X_dists, n/nrow(X))
Xn <- X[Xstar_to_X_dists < quant_dists,]
Yn <- Y[Xstar_to_X_dists < quant_dists]

theta <- darg(NULL, Xn)$start
integral_bounds <- rbind(c(-2,-2), c(2,2))

## Optimize inducing point locations
Xm.wimse1 <- optIP.wIMSE(Xn, M=m, Xn=, theta=theta, w_mean=Xstar,
                         integral_bounds=integral_bounds)

## Use a different variance for weight
Xm.wimse2 <- optIP.wIMSE(Xn, M=m, Xn=, theta=theta, w_mean=Xstar,
                         w_var = c(theta/2, 3*theta),
                         integral_bounds=integral_bounds)

## Plot inducing point design and neighborhood
ranges <- apply(Xn, 2, range)
plot(Xn, pch = 16, cex=.5, col='grey',
     xlim=ranges[,1]+c(-.02, .02), ylim=ranges[,2]+c(-.02, .15),
     xlab='x1', ylab = 'x2',
     main='ALC-optimal Inducing Point Design')
points(Xstar[1], Xstar[2], pch=16)
points(Xm.wimse1$Xm, pch=2, col=3, lwd=2)
```

```
points(Xm.wimse2$Xm, pch=3, col=4, lwd=2)
legend('topleft', legend=c('Xstar','Neighborhood','Xm with w_var=theta',
                           'Xm with nonisotropic weight'),
       col=c(1,'grey',3,4), pch=c(16,16,2,3), lty=NA, lwd=2, ncol=2)

## View wIMSE progress
plot(1:m, log(Xm.wimse1$wimse), type='l', xlab='inducing point number',
     ylab='log wIMSE',main='wIMSE optimization progress')
```

---

qnormscale *Scaling of Inducing Point Design based on Inverse Gaussian CDF*

Description

Scales a set of proposed inducing point locations in [0,1]^d to center around a reference location, returning the scaled design

Usage

```
qnormscale(X, mean, sd)
```

Arguments

X	a matrix or containing a proposed inducing point design in [0,1]^d
mean	a vector representing the reference location to act as the center of the scaling; length(mean) = ncol(X)
sd	a scalar or vector determining the standard deviation for each dimension of the Gaussian CDF

Details

This function scales a set of proposed inducing points in [0,1]^d to be centered and concentrated around a reference location. The proposed inducing points are interpreted as quantiles of one-dimensional Gaussian distributions centered at the reference location with the standard deviation provided by the user. For each dimension qnorm is invoked to rescale the inducing points.

Value

a matrix of the scaled set of inducing points

Author(s)

D. Austin Cole <austin.cole8@vt.edu>

References

D.A. Cole, R.B. Christianson, and R.B. Gramacy (2021). *Locally Induced Gaussian Processes for Large-Scale Simulation Experiments* Statistics and Computing, 31(3), 1-21; preprint on arXiv:2008.12857; https://arxiv.org/abs/2008.12857

See Also

qnorm

Examples

```
## Generate data and define xmean
X <- matrix(runif(30), ncol=2)
xmean <- c(0.3, 0.4) # doesn't need to be in [0,1]^2

## Scale centered at xmean with different standard deviations
X_scaled1 <- qnormscale(X, mean=xmean, sd=.1)
X_scaled2 <- qnormscale(X, mean=xmean, sd=c(.05,.15))

## View scaled X
plot(X, xlab='X1', ylab='X2')
points(xmean[1], xmean[2], pch=16)
points(X_scaled1, pch=2, col=3, lwd=2)
points(X_scaled2, pch=3, col=4, lwd=2)
legend('topright',legend = c('Original X','xmean', 'Xscaled1', 'Xscaled2'),
       pch = c(1,16,2,3), col= c(1,1,3,4), lwd=2, lty=NA)
```

---

scale_gauss_measure_ipTemplate

Inducing points design scaling for a Gaussian measure local neighborhood template

---

Description

Scales a design of inducing points around a Gaussian measure whose mean is the center of the design matrix and its local neighborhood. The output is an inducing points design centered at the origin that can be used as a template for predictions anywhere in the design space (with a local neighborhood of the same size). Method include scaling by a circumscribed hyperrectangle (chr) and an inverse Gaussian CDF (qnorm).

Usage

```
scale_gauss_measure_ipTemplate(X, N, gauss_sd, space_fill_design,
                               method = c('qnorm','chr'), seq_length=20)
```

Arguments

X	a matrix containing the full (large) design matrix of input locations
N	the positive integer number of Nearest Neighbor (NN) locations used to build a local neighborhood
gauss_sd	a vector of standard deviations for the Gaussian measure with with length(gauss_sd)=nrow(X). Note: at this time, the Gaussian measure must only have one nonzero standard deviation (i.e. the Gaussian measure is a slice).
space_fill_design	a matrix in [0,1]^d with M rows and number of columns = ncol(X) that is scaled and centered to create the inducing points template
method	the method by which the inducing point template is scaled. In brief, cHR ("chr") scales space_fill_design to circumscribe the neighborhood and qNorm ("qnorm") scales space_fill_design by the inverse Gaussian CDF.
seq_length	an integer that defines the sequence length used to represent the gaussian measure when building the neighbhorhood.

Details

This function is built to deal with the special class of problems where liGP is used to predict and integrate over a degenerate Gaussian measure where only one dimension has a nonzero standard deviation. Separate subroutines are called for different methods. When method=qnorm, qnormscale is called. The mean of the Gaussian distribution is the median of the design matrix. The standard deviation of the Gaussian distribution is one-third of the maximum distance from the median of the design matrix to the neighborhood points for each dimension.

For each inducing point design, the origin (i.e. predictive location) is appended to the scaled inducing point design. Thus, the resulting design contains M+1 inducing points.

Value

The output is a list with the following components.

Xm.t	a matrix of M+1 inducing points centered at the origin
Xn	a matrix of the local neighborhood at the center of the design
Xc	a matrix of the center of the design used to build the local neighborhood and inducing point template
gauss_sd	the gauss_sd used to generate the local neighborhood
time	a scalar giving the passage of wall-clock time elapsed for (substantive parts of) the calculation

Author(s)

D. Austin Cole <austin.cole8@vt.edu>

References

D.A. Cole, R.B. Christianson, and R.B. Gramacy (2021). *Locally Induced Gaussian Processes for Large-Scale Simulation Experiments* Statistics and Computing, 31(3), 1-21; preprint on arXiv:2008.12857; https://arxiv.org/abs/2008.12857

See Also

qnormscale

Examples

```
## Build up a design with N=~40K locations
x <- seq(-2, 2, by=0.02)
X <- as.matrix(expand.grid(x, x))
X_center <- apply(X, 2, median)
## Create inducing point template, first with
## circumscribed hyperrectangle (cHR) scaling
M = 10
Xm <- matrix(runif(2*M), ncol=2)

## Create template with Inverse Gaussian CDF scaling
qnorm_temp <- scale_ipTemplate(X, N=100, space_fill_design=Xm, method="qnorm")
Xm.t_qnorm <- qnorm_temp$Xm.t
Xn <- qnorm_temp$Xn


## Create template with Inverse Gaussian CDF scaling
gauss_sd <-  c(0, .05)
qnorm_temp_gauss <- scale_gauss_measure_ipTemplate(X, N=100, gauss_sd=gauss_sd,
                                                   space_fill_design=Xm,
                                                   method="qnorm")
Xm.t_qnorm_gauss <- qnorm_temp_gauss$Xm.t
Xn_gauss <- qnorm_temp_gauss$Xn



## View different scaled template designs
ylim <- range(Xn_gauss[,2]) + c(-.03, .05)
plot(Xn, pch=16, cex=.5, col='grey',
     xlab = 'x1', ylab = 'x2', ylim = ylim,
     main='Locally optimized IP template based on Gaussian measure')
points(Xn_gauss, cex=.7)
points(X_center[1], X_center[2], pch=16, cex=1.5)
points(Xm.t_qnorm, pch=2, lwd=2, col=3)
points(Xm.t_qnorm_gauss, pch=6, lwd=2, col=2)
legend('topleft', pch = c(16, 1, 2, 3), col = c('grey', 1, 3, 2),
       legend=c('Local neighborhood (qNorm)',
                'Local neighborhood (Gauss measure)',
                'qnorm ip design',
                'Gaussian measure ip design'))
```

---

scale_ipTemplate *Inducing points design scaling for a local neighborhood template*

Description

Scales a design of inducing points around the center of the design matrix and its local neighborhood. The output is an inducing points design centered at the origin that can be used as a template for predictions anywhere in the design space (with a local neighborhood of the same size). Method include scaling by a circumscribed hyperrectangle (`chr`) and an inverse Gaussian CDF (`qnorm`).

Usage

```
scale_ipTemplate(X, N, space_fill_design, method = c('qnorm','chr'))
```

Arguments

`X`	a `matrix` containing the full (large) design matrix of input locations
`N`	the positive integer number of Nearest Neighbor (NN) locations used to build a local neighborhood
`space_fill_design`	a `matrix` in [0,1]^d with `M` rows and number of columns = `ncol(X)` that is scaled and centered to create the inducing points template
`method`	the method by which the inducing point template is scaled. In brief, cHR (`"chr"`) scales `space_fill_design` to circumscribe the neighborhood and qNorm (`"qnorm"`) scales `space_fill_design` by the inverse Gaussian CDF.

Details

This function calls separate subroutines for certain methods. When `method=qnorm`, `qnormscale` is called. The mean of the Gaussian distribution is the median of the design matrix. The standard deviation of the Gaussian distribution is one-third of the maximum distance from the median of the design matrix to the neighborhood points for each dimension.

For each inducing point design, the origin (i.e. predictive location) is appended to the scaled inducing point design. Thus, the resulting design contains `M+1` inducing points.

Value

The output is a `list` with the following components:

`Xm.t`	a `matrix` of `M+1` inducing points centered at the origin
`Xn`	a `matrix` of the local neighborhood at the center of the design
`time`	a scalar giving the passage of wall-clock time elapsed for (substantive parts of) the calculation

Author(s)

D. Austin Cole <austin.cole8@vt.edu>

References

D.A. Cole, R.B. Christianson, and R.B. Gramacy (2021). *Locally Induced Gaussian Processes for Large-Scale Simulation Experiments* Statistics and Computing, 31(3), 1-21; preprint on arXiv:2008.12857; https://arxiv.org/abs/2008.12857

See Also

qnormscale

Examples

```
## Build up a design with N=~40K locations
x <- seq(-2, 2, by=0.02)
X <- as.matrix(expand.grid(x, x))

## Create inducing point template, first with
## circumscribed hyperrectangle (cHR) scaling
M = 10
Xm <- matrix(runif(2*M), ncol=2)
chr_temp <- scale_ipTemplate(X, N=100, space_fill_design=Xm, method="chr")
Xm.t_chr <- chr_temp$Xm.t
Xn <- chr_temp$Xn

## Now create template with Inverse Gaussian CDF scaling
qnorm_temp <- scale_ipTemplate(X, N=100, space_fill_design=Xm, method="qnorm")
Xm.t_qnorm <- qnorm_temp$Xm.t


## View different scaled template designs
X_center <- apply(X, 2, median)
ranges <- apply(Xn, 2, range)
plot(Xn[,1], Xn[,2], pch=16, cex=.5, col='grey',
     xlim=ranges[,1], ylim=ranges[,2]+c(0,.1),
     xlab = 'x1', ylab = 'x2', main='Scaled Inducing Point templates')
points(X_center[1],X_center[2], pch=16)
points(Xm.t_chr, col=3, pch=2, lwd=2)
points(Xm.t_qnorm, col=4, pch=3, lwd=2)
legend('topleft', pch=c(16,16,2,3), lty=NA, lwd=2, col=c(1,'grey',3,4), ncol=2,
       legend=c('Xcenter', 'Neighborhood','cHR IP template','qNorm IP template'))
```

Index