

Package ‘GPfit’

April 12, 2025

Title Gaussian Processes Modeling

Version 1.0-9

Author Blake MacDoanld [aut],
Hugh Chipman [aut, cre],
Chris Campbell [ctb],
Pritam Ranjan [aut]

Maintainer Hugh Chipman <hugh.chipman@acadiau.ca>

Description A computationally stable approach of fitting a Gaussian Process (GP) model to a deterministic simulator.

Imports lhs (>= 0.5), lattice (>= 0.18-8)

Suggests testthat

License GPL-2

NeedsCompilation no

Encoding UTF-8

RoxygenNote 7.3.2

Repository CRAN

Date/Publication 2025-04-12 13:00:06 UTC

Contents

GPfit-package	2
corr_matrix	2
GP_deviance	4
GP_fit	6
plot	9
predict	12
print.GP	14
scale_norm	15
sig_invb	16
summary.GP	17

Index	19
--------------	-----------

GPfit-package

Gaussian Process Modeling

Description

A computationally stable approach of fitting a Gaussian process (GP) model to simulator outputs. It is assumed that the input variables are continuous and the outputs are obtained from scalar valued deterministic computer simulator.

Details

This package implements a slightly modified version of the regularized GP model proposed in Ranjan et al. (2011). For details see MacDonald et al. (2015). A new parameterization of the Gaussian correlation is used for the ease of optimization. This package uses a multi-start gradient based search algorithm for optimizing the deviance (negative $2 \times \log$ -likelihood).

For a complete list of functions, use `library(help="GPfit")`.
The main function for fitting the GP model is `GP_fit`.

Author(s)

Blake MacDoanld, Hugh Chipman, Pritam Ranjan
Maintainer: Hugh Chipman <hugh.chipman@acadiau.ca>

References

MacDonald, K.B., Ranjan, P. and Chipman, H. (2015). GPfit: An R Package for Fitting a Gaussian Process Model to Deterministic Simulator Outputs. *Journal of Statistical Software*, 64(12), 1-23. <https://www.jstatsoft.org/v64/i12/>

Ranjan, P., Haynes, R., and Karsten, R. (2011). A Computationally Stable Approach to Gaussian Process Interpolation of Deterministic Computer Simulation Data, *Technometrics*, 53(4), 366 - 378.

Santner, T.J., Williams, B., and Notz, W. (2003), *The design and analysis of computer experiments*, Springer Verlag, New York.

corr_matrix

Power Exponential or Matern Correlation Matrix

Description

Computes the power exponential or Matern correlation matrix for a set of n design points in d -dimensional input region and a vector of d correlation hyper-parameters (beta).

Usage

```
corr_matrix(X, beta, corr = list(type = "exponential", power = 1.95))
```

Arguments

X the (n x d) design matrix

beta a (d x 1) vector of correlation hyper-parameters in $(-\infty, \infty)$

corr a list that specifies the type of correlation function along with the smoothness parameter. The default corresponds to power exponential correlation with smoothness parameter "power=1.95". One can specify a different power (between 1.0 and 2.0) for the power exponential, or use the Matern correlation function, specified as `corr=list(type = "matern", nu=(2*k+1)/2)`, where $k \in \{0, 1, 2, \dots\}$

Details

The power exponential correlation function is given by

$$R_{ij} = \prod_{k=1}^d \exp(-10^{\beta_k} |x_{ik} - x_{jk}|^{\text{power}}).$$

The Matern correlation function given by Santner, Williams, and Notz (2003) is

$$R_{ij} = \prod_{k=1}^d \frac{1}{\Gamma(\nu)2^{\nu-1}} (2\sqrt{\nu}|x_{ik}-x_{jk}|10^{\beta_k})^\nu \kappa_\nu(2\sqrt{\nu}|x_{ik}-x_{jk}|10^{\beta_k}),$$

where κ_ν is the modified Bessel function of order ν .

Value

The (n x n) correlation matrix, R, for the design matrix (X) and the hyper-parameters (beta).

Note

Both Matern and power exponential correlation functions use the new β parametrization of hyper-parameters given by $\theta_k = 10^{\beta_k}$ for easier likelihood optimization. That is, beta is a log scale parameter (see MacDonald et al. (2015)).

Author(s)

Blake MacDonald, Hugh Chipman, Pritam Ranjan

References

MacDonald, K.B., Ranjan, P. and Chipman, H. (2015). GPfit: An R Package for Fitting a Gaussian Process Model to Deterministic Simulator Outputs. *Journal of Statistical Software*, 64(12), 1-23. <https://www.jstatsoft.org/v64/i12/>

Ranjan, P., Haynes, R., and Karsten, R. (2011). A Computationally Stable Approach to Gaussian Process Interpolation of Deterministic Computer Simulation Data, *Technometrics*, 53(4), 366 - 378.

Santner, T.J., Williams, B., and Notz, W. (2003), The design and analysis of computer experiments, Springer Verlag, New York.

Examples

```
## 1D Example - 1
n = 5
d = 1
set.seed(3)
library(lhs)
x = maximinLHS(n,d)
beta = rnorm(1)
corr_matrix(x,beta)

## 1D Example - 2
beta = rnorm(1)
corr_matrix(x,beta,corr = list(type = "matern"))

## 2D example - 1
n = 10
d = 2
set.seed(2)
library(lhs)
x = maximinLHS(n,d)
beta = rnorm(2)
corr_matrix(x, beta,
            corr = list(type = "exponential", power = 2))

## 2D example - 2
beta = rnorm(2)
R = corr_matrix(x,beta,corr = list(type = "matern", nu = 5/2))
print(R)
```

GP_deviance

Computes the Deviance of a GP model

Description

Evaluates the deviance (negative $2 \cdot \log$ -likelihood), as defined in Ranjan et al. (2011), however the correlation is reparametrized and can be either power exponential or Matern as discussed in [corr_matrix](#).

Usage

```
GP_deviance(
  beta,
  X,
  Y,
  nug_thres = 20,
```

```
  corr = list(type = "exponential", power = 1.95)
)
```

Arguments

beta	a ($d \times 1$) vector of correlation hyper-parameters, as described in corr_matrix
X	the ($n \times d$) design matrix
Y	the ($n \times 1$) vector of simulator outputs
nug_thres	a parameter used in computing the nugget. See GP_fit .
corr	a list of parameters for the specifying the correlation to be used. See corr_matrix .

Value

the deviance (negative $2 * \log$ -likelihood)

Author(s)

Blake MacDonald, Hugh Chipman, Pritam Ranjan

References

Ranjan, P., Haynes, R., and Karsten, R. (2011). A Computationally Stable Approach to Gaussian Process Interpolation of Deterministic Computer Simulation Data, *Technometrics*, 53(4), 366 - 378.

See Also

[corr_matrix](#) for computing the correlation matrix;
[GP_fit](#) for estimating the parameters of the GP model.

Examples

```
## 1D Example 1
n = 5
d = 1
computer_simulator <- function(x) {
  x = 2 * x + 0.5
  y = sin(10 * pi * x)/(2 * x) + (x - 1)^4
  return(y)
}
set.seed(3)
library(lhs)
x = maximinLHS(n,d)
y = computer_simulator(x)
beta = rnorm(1)
GP_deviance(beta,x,y)

## 1D Example 2
n = 7
d = 1
```

```

computer_simulator <- function(x) {
  y <- log(x + 0.1) + sin(5 * pi * x)
  return(y)
}
set.seed(1)
library(lhs)
x = maximinLHS(n, d)
y = computer_simulator(x)
beta = rnorm(1)
GP_deviance(beta, x, y,
  corr = list(type = "matern", nu = 5/2))

## 2D Example: GoldPrice Function
computer_simulator <- function(x) {
  x1 = 4 * x[, 1] - 2
  x2 = 4 * x[, 2] - 2
  t1 = 1 + (x1 + x2 + 1)^2 *
    (19 - 14 * x1 + 3 * x1^2 -
     14 * x2 + 6 * x1 * x2 + 3 * x2^2)
  t2 = 30 + (2 * x1 - 3 * x2)^2 *
    (18 - 32 * x1 + 12 * x1^2 +
     48 * x2 - 36 * x1 * x2 + 27 * x2^2)
  y = t1 * t2
  return(y)
}
n = 10
d = 2
set.seed(1)
library(lhs)
x = maximinLHS(n, d)
y = computer_simulator(x)
beta = rnorm(2)
GP_deviance(beta, x, y)

```

GP_fit

Gaussian Process Model fitting

Description

For an $(n \times d)$ design matrix, X , and the corresponding $(n \times 1)$ simulator output Y , this function fits the GP model and returns the parameter estimates. The optimization routine assumes that the inputs are scaled to the unit hypercube $[0, 1]^d$.

Usage

```

GP_fit(
  X,
  Y,
  control = c(200 * d, 80 * d, 2 * d),

```

```

nug_thres = 20,
trace = FALSE,
maxit = 100,
corr = list(type = "exponential", power = 1.95),
optim_start = NULL
)

```

Arguments

X	the (n x d) design matrix
Y	the (n x 1) vector of simulator outputs.
control	a vector of parameters used in the search for optimal beta (search grid size, percent, number of clusters). See ‘Details’.
nug_thres	a parameter used in computing the nugget. See ‘Details’.
trace	logical, if TRUE, will provide information on the <code>optim</code> runs
maxit	the maximum number of iterations within <code>optim</code> , defaults to 100
corr	a list of parameters for the specifying the correlation to be used. See <code>corr_matrix</code> .
optim_start	a matrix of potentially likely starting values for correlation hyperparameters for the <code>optim</code> runs, i.e., initial guess of the d-vector beta

Details

This function fits the following GP model, $y(x) = \mu + Z(x)$, $x \in [0, 1]^d$, where $Z(x)$ is a GP with mean 0, $Var(Z(x_i)) = \sigma^2$, and $Cov(Z(x_i), Z(x_j)) = \sigma^2 R_{ij}$. Entries in covariance matrix R are determined by `corr` and parameterized by beta, a d-vector of parameters. For computational stability R^{-1} is replaced with $R_{\delta_{lb}}^{-1}$, where $R_{\delta_{lb}} = R + \delta_{lb}I$ and δ_{lb} is the nugget parameter described in Ranjan et al. (2011).

The parameter estimate beta is obtained by minimizing the deviance using a multi-start gradient based search (L-BFGS-B) algorithm. The starting points are selected using the k-means clustering algorithm on a large maximin LHD for values of beta, after discarding beta vectors with high deviance. The control parameter determines the quality of the starting points of the L-BFGS-B algorithm.

`control` is a vector of three tunable parameters used in the deviance optimization algorithm. The default values correspond to choosing $2*d$ clusters (using k-means clustering algorithm) based on $80*d$ best points (smallest deviance, refer to `GP_deviance`) from a $200*d$ - point random maximin LHD in beta. One can change these values to balance the trade-off between computational cost and robustness of likelihood optimization (or prediction accuracy). For details see MacDonald et al. (2015).

The `nug_thres` parameter is outlined in Ranjan et al. (2011) and is used in finding the lower bound on the nugget (δ_{lb}).

Value

an object of class GP containing parameter estimates beta and sig2, nugget parameter delta, the data (X and Y), and a specification of the correlation structure used.

Author(s)

Blake MacDonald, Hugh Chipman, Pritam Ranjan

References

MacDonald, K.B., Ranjan, P. and Chipman, H. (2015). GPfit: An R Package for Fitting a Gaussian Process Model to Deterministic Simulator Outputs. *Journal of Statistical Software*, 64(12), 1-23. <https://www.jstatsoft.org/v64/i12/>

Ranjan, P., Haynes, R., and Karsten, R. (2011). A Computationally Stable Approach to Gaussian Process Interpolation of Deterministic Computer Simulation Data, *Technometrics*, 53(4), 366 - 378.

See Also

[plot](#) for plotting in 1 and 2 dimensions;
[predict](#) for predicting the response and error surfaces;
[optim](#) for information on the L-BFGS-B procedure;
[GP_deviance](#) for computing the deviance.

Examples

```
## 1D Example 1
n = 5
d = 1
computer_simulator <- function(x){
  x = 2 * x + 0.5
  y = sin(10 * pi * x) / (2 * x) + (x - 1)^4
  return(y)
}
set.seed(3)
library(lhs)
x = maximinLHS(n, d)
y = computer_simulator(x)
GPmodel = GP_fit(x, y)
print(GPmodel)
```

```
## 1D Example 2
n = 7
d = 1
computer_simulator <- function(x) {
  y <- log(x + 0.1) + sin(5 * pi * x)
  return(y)
}
set.seed(1)
library(lhs)
x = maximinLHS(n, d)
y = computer_simulator(x)
GPmodel = GP_fit(x, y)
print(GPmodel, digits = 4)
```



```
## 2D Example: GoldPrice Function
computer_simulator <- function(x) {
  x1 = 4 * x[, 1] - 2
  x2 = 4 * x[, 2] - 2
  t1 = 1 + (x1 + x2 + 1)^2 * (19 - 14 * x1 + 3 * x1^2 - 14 * x2 +
    6 * x1 * x2 + 3 * x2^2)
  t2 = 30 + (2 * x1 - 3 * x2)^2 * (18 - 32 * x1 + 12 * x1^2 + 48 * x2 -
    36 * x1 * x2 + 27 * x2^2)
  y = t1 * t2
  return(y)
}
n = 30
d = 2
set.seed(1)
library(lhs)
x = maximinLHS(n, d)
y = computer_simulator(x)
GPmodel = GP_fit(x, y)
print(GPmodel)
```

plot

Plotting GP model fits

Description

Plots the predicted response and mean squared error (MSE) surfaces for simulators with 1 and 2 dimensional inputs (i.e. $d = 1, 2$).

Usage

```
## S3 method for class 'GP'
plot(
  x,
  M = 1,
  range = c(0, 1),
  resolution = 50,
  colors = c("black", "blue", "red"),
  line_type = c(1, 2),
  pch = 20,
  cex = 1,
  legends = FALSE,
  surf_check = FALSE,
  response = TRUE,
  ...
)
```

Arguments

<code>x</code>	a class GP object estimated by <code>GP_fit</code>
<code>M</code>	the number of iterations for use in prediction. See predict.GP
<code>range</code>	the input range for plotting (default set to <code>[0, 1]</code>)
<code>resolution</code>	the number of points along a coordinate in the specified range
<code>colors</code>	a vector of length 3 assigning <code>colors[1]</code> to training design points, <code>colors[2]</code> to model predictions, and <code>colors[3]</code> to the error bounds
<code>line_type</code>	a vector of length 2 assigning <code>line_type[1]</code> to model predictions, and <code>line_type[2]</code> to the error bounds
<code>pch</code>	a parameter defining the plotting character for the training design points, see ‘pch’ for possible options in par
<code>cex</code>	a parameter defining the size of the pch used for plotting the training design points, see ‘cex’ for possible options in par
<code>legends</code>	a parameter that controls the inclusion of a legend ; by default it is ‘FALSE’
<code>surf_check</code>	logical, switch between 3d surface and 2d level/contour plotting, the default of FALSE implies level/contour plotting
<code>response</code>	logical, switch between predicted response and error (MSE) plots, the default of TRUE displays the response surface
<code>...</code>	additional arguments from <code>[lattice::wireframe()]</code> or <code>[lattice::levelplot()]</code>

Methods (by class)

- `plot(GP)`: The plot method creates a **graphics** plot for 1-D fits and **lattice** plot for 2-D fits.

Author(s)

Blake MacDonald, Hugh Chipman, Pritam Ranjan

References

Ranjan, P., Haynes, R., and Karsten, R. (2011). A Computationally Stable Approach to Gaussian Process Interpolation of Deterministic Computer Simulation Data, *Technometrics*, 53(4), 366 - 378.

See Also

[GP_fit](#) for estimating the parameters of the GP model;
[predict.GP](#) for predicting the response and error surfaces;
[par](#) for additional plotting characters and line types for 1 dimensional plots;
[\[lattice::wireframe\(\)\]](#) and [\[lattice::levelplot\(\)\]](#) for additional plotting settings in 2 dimensions.

Examples

```
## 1D Example 1
n <- 5
d <- 1
computer_simulator <- function(x){
```

```

    x <- 2 * x + 0.5
    y <- sin(10 * pi * x) / (2 * x) + (x - 1)^4
    return(y)
}
set.seed(3)
library(lhs)
x <- maximinLHS(n,d)
y <- computer_simulator(x)
GPmodel <- GP_fit(x,y)
plot(GPmodel)

## 1D Example 2
n <- 7
d <- 1
computer_simulator <- function(x) {
  y <- log(x + 0.1) + sin(5 * pi * x)
  return(y)
}
set.seed(1)
library(lhs)
x <- maximinLHS(n,d)
y <- computer_simulator(x)
GPmodel <- GP_fit(x, y)
## Plotting with changes from the default line type and characters
plot(GPmodel, resolution = 100, line_type = c(6,2), pch = 5)

## 2D Example: GoldPrice Function
computer_simulator <- function(x) {
  x1 <- 4 * x[, 1] - 2
  x2 <- 4 * x[, 2] - 2
  t1 <- 1 + (x1 + x2 + 1)^2 * (19 - 14 * x1 + 3 * x1^2 - 14 * x2 +
    6 * x1 * x2 + 3 * x2^2)
  t2 <- 30 + (2 * x1 - 3 * x2)^2 * (18 - 32 * x1 + 12 * x1^2 + 48 * x2 -
    36 * x1 * x2 + 27 * x2^2)
  y <- t1 * t2
  return(y)
}
n <- 30
d <- 2
set.seed(1)
x <- lhs::maximinLHS(n, d)
y <- computer_simulator(x)
GPmodel <- GP_fit(x, y)
## Basic level plot
plot(GPmodel)
## Adding Contours and increasing the number of levels
plot(GPmodel, contour = TRUE, cuts = 50, pretty = TRUE)
## Plotting the Response Surface
plot(GPmodel, surf_check = TRUE)
## Plotting the Error Surface with color
plot(GPmodel, surf_check = TRUE, response = FALSE, shade = TRUE)

```

 predict

Model Predictions from GPfit

Description

Computes the regularized predicted response $\hat{y}_{\delta_{lb},M}(x)$ and the mean squared error $s_{\delta_{lb},M}^2(x)$ for a new set of inputs using the fitted GP model.

The value of M determines the number of iterations (or terms) in approximating $R^{-1} \approx R_{\delta_{lb},M}^{-1}$. The iterative use of the nugget δ_{lb} , as outlined in Ranjan et al. (2011), is used in calculating $\hat{y}_{\delta_{lb},M}(x)$ and $s_{\delta_{lb},M}^2(x)$, where $R_{\delta,M}^{-1} = \sum_{t=1}^M \delta^{t-1} (R + \delta I)^{-t}$.

Usage

```
## S3 method for class 'GP'
predict(object, xnew = object$X, M = 1, ...)

## S3 method for class 'GP'
fitted(object, ...)
```

Arguments

object	a class GP object estimated by GP_fit
xnew	the (n_new x d) design matrix of test points where model predictions and MSEs are desired
M	the number of iterations. See 'Details'
...	for compatibility with generic method predict

Value

Returns a list containing the predicted values (Y_hat), the mean squared errors of the predictions (MSE), and a matrix (complete_data) containing xnew, Y_hat, and MSE

Methods (by class)

- `predict(GP)`: The predict method returns a list of elements Y_hat (fitted values), Y (dependent variable), MSE (residuals), and completed_data (the matrix of independent variables, Y_hat, and MSE).

Functions

- `fitted(GP)`: The fitted method extracts the complete data.

Author(s)

Blake MacDonald, Hugh Chipman, Pritam Ranjan

References

Ranjan, P., Haynes, R., and Karsten, R. (2011). A Computationally Stable Approach to Gaussian Process Interpolation of Deterministic Computer Simulation Data, *Technometrics*, 53(4), 366 - 378.

See Also

[GP_fit](#) for estimating the parameters of the GP model;
[plot](#) for plotting the predicted and error surfaces.

Examples

```
## 1D Example
n <- 5
d <- 1
computer_simulator <- function(x){
  x <- 2*x+0.5
  sin(10*pi*x)/(2*x) + (x-1)^4
}
set.seed(3)
library(lhs)
x <- maximinLHS(n,d)
y <- computer_simulator(x)
xvec <- seq(from = 0, to = 1, length.out = 10)
GPmodel <- GP_fit(x, y)
head(fitted(GPmodel))
lapply(predict(GPmodel, xvec), head)

## 1D Example 2
n <- 7
d <- 1
computer_simulator <- function(x) {
  log(x+0.1)+sin(5*pi*x)
}
set.seed(1)
library(lhs)
x <- maximinLHS(n,d)
y <- computer_simulator(x)
xvec <- seq(from = 0, to = 1, length.out = 10)
GPmodel <- GP_fit(x, y)
head(fitted(GPmodel))
predict(GPmodel, xvec)

## 2D Example: GoldPrice Function
computer_simulator <- function(x) {
  x1 <- 4*x[,1] - 2
  x2 <- 4*x[,2] - 2
  t1 <- 1 + (x1 + x2 + 1)^2*(19 - 14*x1 + 3*x1^2 - 14*x2 +
    6*x1*x2 + 3*x2^2)
  t2 <- 30 + (2*x1 - 3*x2)^2*(18 - 32*x1 + 12*x1^2 + 48*x2 -
    36*x1*x2 + 27*x2^2)
  y <- t1*t2
}
```

```
    return(y)
  }
  n <- 10
  d <- 2
  set.seed(1)
  library(lhs)
  x <- maximinLHS(n,d)
  y <- computer_simulator(x)
  GPmodel <- GP_fit(x,y)
  # fitted values
  head(fitted(GPmodel))
  # new data
  xvector <- seq(from = 0,to = 1, length.out = 10)
  xdf <- expand.grid(x = xvector, y = xvector)
  predict(GPmodel, xdf)
```

print.GP

GP model fit Summary

Description

Prints the summary of a class GP object estimated by `GP_fit`

Usage

```
## S3 method for class 'GP'
print(x, ...)
```

Arguments

`x` a class GP object estimated by `GP_fit`
`...` for compatibility with generic method `print`

Details

Prints the summary of the class GP object. It returns the number of observations, input dimension, parameter estimates of the GP model, lower bound on the nugget, and the nugget threshold parameter (described in `GP_fit`).

Author(s)

Blake MacDonald, Hugh Chipman, Pritam Ranjan

See Also

`GP_fit` for more information on estimating the model;
`print` for more description on the `print` function.

Examples

```

## 1D example
n <- 5
d <- 1
computer_simulator <- function(x){
  x <- 2 * x + 0.5
  y <- sin(10 * pi * x) / (2 * x) + (x - 1)^4
  return(y)
}
set.seed(3)
x <- lhs::maximinLHS(n, d)
y <- computer_simulator(x)
GPmodel <- GP_fit(x, y)
print(GPmodel)

## 2D Example: GoldPrice Function
computer_simulator <- function(x) {
  x1 <- 4*x[,1] - 2
  x2 <- 4*x[,2] - 2
  t1 <- 1 + (x1 + x2 + 1)^2*(19 - 14*x1 + 3*x1^2 - 14*x2 +
    6*x1*x2 + 3*x2^2)
  t2 <- 30 + (2*x1 - 3*x2)^2*(18 - 32*x1 + 12*x1^2 + 48*x2 -
    36*x1*x2 + 27*x2^2)
  y <- t1*t2
  return(y)
}
n <- 30
d <- 2
set.seed(1)
x <- lhs::maximinLHS(n, d)
y <- computer_simulator(x)
GPmodel <- GP_fit(x,y)
print(GPmodel, digits = 3)

```

scale_norm

Scale variable into normal range 0, 1

Description

Perform calculation: $(x - \min(x)) / (\max(x) - \min(x))$

Usage

```
scale_norm(x, range = NULL)
```

Arguments

x numeric vector

range numeric vector additional values for shrinking distribution of values within the 0-1 space, without affecting limits of x

Value

numeric vector

Examples

```
scale_norm(x = c(-1, 4, 10, 182))
# lower bound extended beyond -1
# upper bound still range of data
scale_norm(x = c(-1, 4, 10, 182), range = c(-100, 100))
```

sig_invb

Internal tools

Description

shared utilities between [GP_deviance](#) and [GP_fit](#)

Usage

```
sig_invb(
  X,
  Y,
  beta,
  corr = list(type = "exponential", power = 1.95),
  nug_thres = 20
)
```

Arguments

X the ($n \times d$) design matrix

Y the ($n \times 1$) vector of simulator outputs

beta a ($d \times 1$) vector of correlation hyper-parameters, as described in [corr_matrix](#)

corr a list of parameters for the specifying the correlation to be used. See [corr_matrix](#).

nug_thres a parameter used in computing the nugget. See [GP_fit](#).

Value

list with elements delta, L, mu_hat, Sig_invb

Examples

```
set.seed(3234)
GPfit:::sig_invb(
  X = matrix((0:10) / 10),
  Y = runif(11),
  beta = 1.23)
```

summary.GP

Summary of GP model fit

Description

Prints the summary of a class GP object estimated by GP_fit

Usage

```
## S3 method for class 'GP'
summary(object, ...)
```

Arguments

object a class GP object estimated by GP_fit
... for compatibility with generic method [summary](#)

Details

prints the summary of the GP object (object), by calling [print.GP](#)

Author(s)

Blake MacDonald, Hugh Chipman, Pritam Ranjan

See Also

[print.GP](#) for more description of the output;
[GP_fit](#) for more information on estimating the model;
[summary](#) for more description on the summary function.

Examples

```
## 1D example
n <- 5
d <- 1
computer_simulator <- function(x){
  x <- 2 * x + 0.5
  y <- sin(10 * pi * x) / (2 * x) + (x - 1)^4
  return(y)
}
```

```
set.seed(3)
x <- lhs::maximinLHS(n, d)
y <- computer_simulator(x)
GPmodel <- GP_fit(x, y)
summary(GPmodel)

## 2D Example: GoldPrice Function
computer_simulator <- function(x) {
  x1 = 4*x[, 1] - 2
  x2 = 4*x[, 2] - 2
  t1 = 1 + (x1 + x2 + 1)^2*(19 - 14*x1 + 3*x1^2 - 14*x2 +
    6*x1*x2 + 3*x2^2)
  t2 = 30 + (2*x1 - 3*x2)^2*(18 - 32*x1 + 12*x1^2 + 48*x2 -
    36*x1*x2 + 27*x2^2)
  y = t1*t2
  return(y)
}
n <- 10
d <- 2
set.seed(1)
x <- lhs::maximinLHS(n, d)
y <- computer_simulator(x)
GPmodel <- GP_fit(x, y)
summary(GPmodel)
```

Index

- * **Correlation**
 - corr_matrix, 2
- * **Deviance**
 - GP_deviance, 4
- * **Exponential**
 - corr_matrix, 2
- * **Gaussian**
 - corr_matrix, 2
 - GP_fit, 6
 - predict, 12
- * **Matern**
 - corr_matrix, 2
- * **Model**
 - predict, 12
- * **Power**
 - corr_matrix, 2
- * **Prediction**
 - predict, 12
- * **Process**
 - GP_fit, 6
 - predict, 12

corr_matrix, 2, 4, 5, 7, 16

fitted.GP (predict), 12

GP_deviance, 4, 7, 8, 16

GP_fit, 2, 5, 6, 10, 13, 14, 16, 17

GPfit (GPfit-package), 2

GPfit-package, 2

legend, 10

optim, 7, 8

par, 10

plot, 8, 9, 13

predict, 8, 12, 12

predict.GP, 10

print, 14

print.GP, 14, 17

scale_norm, 15

sig_invb, 16

summary, 17

summary.GP, 17