Package 'GPpenalty'

May 7, 2026

Title Penalized Likelihood in Gaussian Processes

Version 1.0.1

Description Implements maximum likelihood estimation for Gaussian processes, supporting both isotropic and separable models with predictive capabilities. Includes penalized likelihood estimation following Li and Sudjianto (2005, <doi:10.1198/004017004000000671>), with cross-validation guided by decorrelated prediction error (DPE) metric. DPE metric, motivated by Mahalanobis distance, serves as evaluation criteria that accounts for predictive uncertainty in tuning parameter selection (Mutoh, Booth, and Stallrich, 2025, <doi:10.48550/arXiv.2511.18111>). Designed specifically for small datasets.

License MIT + file LICENSE

Encoding UTF-8

RoxygenNote 7.3.3

Depends R (>= 3.5.0)

LinkingTo Rcpp, RcppArmadillo

Imports Rcpp, doParallel, foreach

Suggests testthat (>= 3.0.0)

Config/testthat/edition 3

NeedsCompilation yes

Author Ayumi Mutoh [aut, cre]

Maintainer Ayumi Mutoh <amutoh@ncsu.edu>

Repository CRAN

Date/Publication 2025-11-26 14:20:07 UTC

Contents

GPpenalty-package *GPpenalty*

Description

Implements maximum likelihood estimation for Gaussian processes, supporting both isotropic and separable models with predictive capabilities. Includes penalized likelihood estimation, with cross-validation guided by decorrelated prediction error (DPE) metric. DPE metric, motivated by Mahalanobis distance, serves as evaluation criteria that accounts for predictive uncertainty in tuning parameter selection. Designed specifically for small datasets.

Functions

- `mle_gp`: The function computes maximum likelihood estimates for the lengthscale, scale, mu, and nugget (g) parameters using `optim`, with options to fix or assume zero for certain parameters.
- `predict_gp`: Computes the posterior mean and covariance matrix for a given set of input locations based on a fitted model.
- `gp_cv`: Performs cross-validation to select an optimal tuning parameter for penalized MLE of the lengthscale parameter in Gaussian processes.
- `mle_penalty`: Computes penalized maximum likelihood estimates for the lengthscale parameter using `optim`.
- `score`: Calculates a score value. Higher score values indicate better fits.
- `dpe`: Calculates a decorrelated prediction error value. Lower dpe values indicate better fits.
- `kernel`: Compute the squared exponential kernel defined as $k = \exp(-\theta(x-x')^2)+g$, where θ is the lengthscale parameter and g is a jitter term. Both isotropic and separable kernels are supported.

Examples

```
#### define function ###
f_x <- function(x) {
return(sin(2*pi*x) + x^2)
}

### x and y ###
x <- runif(8, min=0, max=1)
y <- f_x(x)
x.test <- runif(100, min=0, max=1)
y.test <- f_x(x.test)
```

```
### no penalization ###
# fit
fit <- mle_gp(y, x)
# prediction
pred <- predict_gp(fit, x.test)


# obtain kernel function
cov_function <- kernel(x1=x, theta=fit$theta)


# evaluate the predictive performance with score
score_value <- score(y.test, pred$mup, pred$Sigmap)

### penalization ###
# leave-one-out cross validation
loocv.lambda <- gp_cv(y, x)
# fit
fit.loocv <- mle_penalty(loocv.lambda)
# prediction
pred.loocv <- predict_gp(fit.loocv, x.test)

# k-fold cross validation with the dpe metric
kfold.dpe <- gp_cv(y, x, k=4)
# fit
fit.kfold.dpe <- mle_penalty(kfold.dpe)
# prediction
pred.kfold.dpe <- predict_gp(fit.kfold.dpe, x.test)

# k-fold cross validation with the mse metric
kfold.mse <- gp_cv(y, x, k=4, metric="mse")
# fit
fit.kfold.mse <- mle_penalty(kfold.mse)
# prediction
pred.kfold.mse <- predict_gp(fit.kfold.mse, x.test)
```

---

dpe *dpe*

Description

Calculates a decorrelated prediction error (DPE) value. Lower DPE values indicate better fits.

Usage

```
dpe(y, mu, R)
```

Arguments

y	response variable vector
mu	predicted mean vector
R	predicted covariance matrix with the scale parameter removed

Value

a numeric value

Examples

```
### test function ###
f_x <- function(x) {
return(sin(2*pi*x) + x^2)
}

### training data ###
n <- 8
x <- runif(n, 0, 1)
y <- f_x(x)

### testing data ###
n.test <- 100
x.test <- runif(n.test, 0, 1)
y.test <- f_x(x.test)

### get parameter estimates ###
out <- mle_gp(y, x)

### prediction ###
pred <- predict_gp(out, x.test)

### get DPE value ###
DPE_value <- dpe(y.test, pred$mup, pred$R)
```

---

gp_cv *gp_cv*

Description

Performs cross-validation to select an optimal tuning parameter for penalized MLE of the length-scale parameter in Gaussian processes.

Usage

```
gp_cv(
  y,
  x,
  lambda = NULL,
  sep = TRUE,
  mu = FALSE,
  g = FALSE,
  fixed_g = NULL,
  profile = TRUE,
  initialvals = NULL,
  n_init = 10,
  scad = FALSE,
  k = NULL,
  theta_upper = 1000,
  theta_lower = 0.001,
  metric = "dpe",
  ncores = 1
)
```

Arguments

y	A numeric vector of the response variable.
x	A numeric vector or matrix of the input variables.
lambda	A tuning parameter. Default is NULL. Users may specify one or more lambda values to be evaluated. When NULL, 41 lambda values ranging from 0 to 7.389 will be automatically evaluated.
sep	Logical indicator for using a separable kernel function (sep=TRUE) or an isotropic kernel function (sep=FALSE). Default is TRUE.
mu	Logical indicator for assuming zero mean (mu=FALSE) or estimating the mean (mu=TRUE). Default is FALSE (assumes the data is centered beforehand).
g	Logical indicator for fixing the nugget value to a small constant (g=FALSE) or estimating the nugget (g=TRUE). Default is FALSE.
fixed_g	Nugget value to fix when g=FALSE. Default is fixed_g=NULL. If NULL, the nugget is fixed to 1.490116e-08.
profile	Logical indicator for optimizing the profile log-likelihood (profile=TRUE). When TRUE, the log-likelihood is a function of lengthscale and nugget only. Solve the closed forms for scale and mu parameters. When FALSE, the full log-likelihood is optimized (lengthscale, scale, mean, and nugget are estimated together). Default is TRUE.
initialvals	A numeric vector or matrix of initial values for optimization. The length should match the number of parameters to estimate. Default is NULL. If NULL, 10 sets of initial values are randomly generated. The number of sets can be specified by specifying n_init.
n_init	An integer indicating the number of randomly generated initial value sets to evaluate when initialvals is not provided. Default is 10.

scad	Logical indicator for a lasso penalty (scad=FALSE) or SCAD penalty (scad=TRUE) when penalty=TRUE. Default is lasso penalty.
k	The number of folds for k-fold CV. Default is NULL. When NULL, leave-one-out CV using mean squared error metric is performed. To conduct k-fold CV, users must specify a value for k.
theta_upper	Upper bound for theta in optim. Default is 1000.
theta_lower	Lower bound for theta in optim. Default is 0.001.
metric	The evaluation metric used in CV. Default is "dpe". The available metrics are "dpe", "md", "score", and "mse". The dpe, md, and score metrics are only available when k is specified.
ncores	A number of cores for parallel computing with optim. Default is 1 (no parallelization). Make sure your system supports the specified number of cores. Paralleling is recommended to improve performance.

Details

This function supports both leave-one-out and k-fold cross-validation for selecting a suitable tuning parameter value in penalized likelihood estimation. Users can choose among several evaluation metrics, including decorrelated prediction error (dpe), Mahalanobis distance (md), score, and mean squared error (mse), to guide the selection process. For the dpe, md, and score metrics, only k-fold cross-validation is available, as these metrics account for correlation structure. For leave-one-out cross-validation, only the mse metric be used. For dpe, md, and mse metrics, the lambda corresponding to the minimum value across the k folds is selected as optimal. For the score metric, the lambda with the maximum value is selected. The function returns the optimal lambda value along with the lambda selected using the one-standard error rule.

Value

A list includes y, x, selected lambda, and settings:

- y: A copy of y.
- x: A copy of x.
- lambda.dpe.min: Returned when k is specified and metric="dpe"; the lambda value that minimizes the dpe across the folds.
- lambda.dpe.1se: Returned when k is specified and metric="dpe"; the lambda value selected using the one-standard-error rule.
- lambda.min: Returned when k is not specified or metric="mse"; the lambda value that minimizes mean squared error across the folds.
- lambda.1se: Returned when k is not specified or metric="mse"; the lambda value selected using the one-standard-error rule.
- lambda.score.max: Returned when k is specified and metric="score"; the lambda value that maximizes the score across the folds.
- lambda.score.1se: Returned when k is specified and metric="score"; the lambda value selected using the one-standard-error rule.
- lambda.md.min: Returned when k is specified and metric="md"; the lambda value that minimizes the md across the folds.

- lambda.md.1se: Returned when k is specified and metric="md"; the lambda value selected using the one-standard-error rule.
- initialvals: A vector or matrix of initial values used in optim.
- n_init: A copy of n_init: the number of randomly generated initial value sets.
- d: The dimensionality of the lengthscale parameter. If sep=TRUE, d is equal to the number of columns in x. Otherwise it is set to 1 for isotropic kernels.
- profile: A copy of the logical indicator for profile likelihood optimization.
- mu: A copy of the logical indicator for mean estimation.
- g: A copy of the logical indicator for nugget estimation.
- fixed_g: The fixed nugget value used when g = FALSE. If NULL, the nugget is set to 1.490116e-08 in mle_penalty function.
- metric: A copy of the evaluation metric used in CV.
- scad: A copy of the logical indicator for SCAD penalty usage.
- theta_upper: A copy of theta_upper for optimization.
- theta_lower: A copy of theta_lower for optimization.

Examples

```
### training data ###
n <- 8

### test function ###
f_x <- function(x) {
return(sin(2*pi*x) + x^2)
}

### generate x ###
x <- runif(n, 0, 1)
y <- f_x(x)

### k-fold cross validation ###
cv.lambda <- gp_cv(y, x, k=4)

### mse metric ###
cv.lambda <- gp_cv(y, x, k=4, metric="mse")

### leave-one-out cross validation ###
cv.lambda <- gp_cv(y, x)


#' ### specify the number of randomly generated initial value sets to be evaluated. ###
cv.lambda <- gp_cv(y, x, n_init=5)
```

kernel *kernel*

Description

Compute the squared exponential kernel defined as $k = \exp(-\theta(x - x')^2) + g$, where θ is the lengthscale parameter and g is a jitter term. Both isotropic and separable kernels are supported.

Usage

```
kernel(x1, theta, x2 = NULL, g = NULL)
```

Arguments

x1	matrix of input locations
theta	a scalar or vector specifying the lengthscale parameter. If a vector is provided, a separable kernel function is used. If a scalar is provided and x1 has more than one column, an isotropic kernel is assumed.
x2	matrix of second input locations. If NULL, distance is computed between x1 and itself.
g	a jitter term. It is added when x2=NULL for computational stability.

Details

Matrix computations are implemented in C++ for improved performance and computational efficiency.

Value

a matrix representing the evaluated kernel function

Examples

```
### isotropic ###
x <- matrix(seq(0, 10, length=10), ncol=1)
theta <- 5
k <- kernel(x1=x, theta=theta)

### separable ###
x <- matrix(seq(0, 20, length=20), ncol=2)
theta <- c(2, 4)
k <- kernel(x1=x, theta=theta)
```

mle_gp *mle_gp*

Description

The function computes maximum likelihood estimates for the lengthscale, scale, mu, and nugget (g) parameters using `optim`, with options to fix or assume zero for certain parameters.

Usage

```
mle_gp(
  y,
  x,
  sep = TRUE,
  mu = FALSE,
  g = FALSE,
  fixed_g = NULL,
  profile = TRUE,
  initialvals = NULL,
  n_init = 10,
  penalty = FALSE,
  scad = FALSE,
  lambda = 0,
  theta_upper = 1000,
  theta_lower = 0.001,
  ncores = 1
)
```

Arguments

`y`	A numeric vector of the response variable.
`x`	A numeric vector or matrix of the input variables.
`sep`	Logical indicator for using a separable kernel function (`sep=TRUE`) or an isotropic kernel function (`sep=FALSE`). Default is TRUE.
`mu`	Logical indicator for assuming zero mean (`mu=FALSE`) or estimating the mean (`mu=TRUE`). Default is FALSE (assumes the data is centered beforehand).
`g`	Logical indicator for fixing the nugget value to a small constant (`g=FALSE`) or estimating the nugget (`g=TRUE`). Default is FALSE.
`fixed_g`	Nugget value to fix when `g=FALSE`. Default is `fixed_g=NULL`. If NULL, the nugget is fixed to 1.490116e-08.
`profile`	Logical indicator for optimizing the profile log-likelihood (`profile=TRUE`). When TRUE, the log-likelihood is a function of lengthscale and nugget only. Solve the closed forms for scale and mu parameters. When FALSE, the full log-likelihood is optimized (lengthscale, scale, mean, and nugget are estimated together). Default is TRUE.

initialvals	A numeric vector or matrix of initial values for optimization. The length should match the number of parameters to estimate. Default is NULL. If NULL, 10 sets of initial values are randomly generated. The number of sets can be specified by specifying n_init.
n_init	An integer indicating the number of randomly generated initial value sets to evaluate when initialvals is not provided. Default is 10.
penalty	Logical indicator for penalization. Default is penalty=FALSE (returns MLE). When penalty=TRUE and no lambda value is specified, a set of estimated values along with evaluated lambda values is returned.
scad	Logical indicator for a lasso penalty (scad=FALSE) or SCAD penalty (scad=TRUE) when penalty=TRUE. Default is lasso penalty.
lambda	Tuning parameter value. Default is 0 (MLE). The user may specify a custom lambda value.
theta_upper	Upper bound for theta in optim. Default is 1000.
theta_lower	Lower bound for theta in optim. Default is 0.001.
ncores	A number of cores for parallel computing with optim. Default is 1. Make sure your system supports the specified number of cores.

Details

The function uses numerical optimization for lengthscale and nugget parameters as there's no closed-form solutions. In contrast, closed form solutions exist for the scale and mu parameters. Users have options to choose whether to solve them analytically or include them in optimization process. If mu is assumed to be zero (by setting mu=FALSE), the input data should be centered beforehand. The nugget term (g) can also be optimized alongside the lengthscale parameter or fixed to a small constant. When no initial values are provided (initialvals=NULL), the function generates 10 random sets and selects the one that minimizes the negative log-likelihood. The number of sets can be specified by specifying n_init. Additionally, users can apply a penalty to the lengthscale parameter by specifying a tuning parameter, lambda. For guidance on choosing lambda, refer to gp_cv function.

Value

A list of y, x, and hyperparameters:

- y: A copy of y.
- x: A copy of x.
- theta: A matrix of estimated lengthscale parameter.
- sigma2: The estimated scale parameter.
- mu: Returns 0 if mu=FALSE otherwise the estimated mu parameter.
- g: Returns the fixed_g value if g=FALSE otherwise the estimated nugget value.
- penalty: A copy of the penalty indicator.
- lambda: A vector of evaluated lambda values if penalty=TRUE otherwise NULL.
- theta_upper: A copy of theta_upper for optimization.
- theta_lower: A copy of theta_lower for optimization.

Examples

```
### training data ###
n <- 8

### test function ###
f_x <- function(x) {
return(sin(2*pi*x) + x^2)
}

### generate x ###
x <- runif(n, 0, 1)

y <- f_x(x)

### Optimize only the lengthscale parameter and solve for scale. ###
### Assume zero mean and fix g to a small constant. ###
fit <- mle_gp(y, x)

### Include etimation of mu ###
fit <- mle_gp(y, x, mu=TRUE)

### Optimize g as well ###
fit <- mle_gp(y, x, mu=TRUE, g=TRUE)

### Jointly optimize the lengthscale and scale ###
fit <- mle_gp(y, x, profile=FALSE)

### Fix g to a user specified value ###
fit <- mle_gp(y, x, fixed_g=0.0001)

### Set the upper and lower bounds for theta ###
fit <- mle_gp(y, x, theta_upper=100, theta_lower=0.01)
```

`mle_penalty` *mle_penalty*

Description

Computes penalized maximum likelihood estimates for the lengthscale parameter using `optim`.

Usage

```
mle_penalty(object, one.se = FALSE, lambda = NULL, ncores = 1)
```

Arguments

object	A list returned from gp_cv.
one.se	Logical indicator for selecting the lambda value using the one-standard error. Default is FALSE. When FALSE, the lambda value that minimizes mse, dpe, or mahalanobis distance (md), or maximizes the score, is selected. When TRUE, the lambda value is chosen based on the one-standard error rule.
lambda	A user specified tuning parameter. This can be provided directly instead of performing cross-validation.
ncores	A number of cores for parallel computing with optim. Default is 1. Make sure your system supports the specified number of cores.

Details

This function takes the output from gp_cv and computes penalized MLEs for the lengthscale parameter, along with MLEs for other model parameters. Users may choose to apply the one standard error rule for selecting the lambda value. The gp_cv function returns both the optimal lambda and one standard error lambda except for the md metric. See gp_cv for details.

Value

A list of y, x, and hyperparameters:

- y: A copy of y.
- x: A copy of x.
- theta: A matrix of penalized lengthscale estimates.
- sigma2: The estimated scale parameter.
- mu: Returns 0 if mu=FALSE otherwise the estimated mu parameter.
- g: Returns the fixed_g value if g=FALSE otherwise the estimated nugget value.
- lambda: A scalar or vector of lambda values evaluated.
- theta_upper: A copy of the upper bound for theta used in gp_cv function.
- theta_lower: A copy of the lower bound for theta used in gp_cv function.

Examples

```
### training data ###
n <- 8

### test function ###
f_x <- function(x) {
return(sin(2*pi*x) + x^2)
}

### generate x ###
x <- runif(n, 0, 1)
y <- f_x(x)

### k-fold cross validation ###
```

```
cv.lambda <- gp_cv(y, x, k=4)

### fit the model ###
penalized.mle <- mle_penalty(cv.lambda)

#### use the one standard error rule ###
penalized.mle <- mle_penalty(cv.lambda, one.se=TRUE)

### specify lambda ###
penalized.mle <- mle_penalty(cv.lambda, lambda=cv.lambda$lambda.score.max)
```

---

predict_gp *predict_gp*

Description

Computes the posterior mean and covariance matrix for a given set of input locations based on a fitted model.

Usage

```
predict_gp(out, xx)
```

Arguments

out	out from mle_gp or mle_gp.
xx	A numerical vector or matrix of new input locations.

Details

From the model fitted by mle_gp or mle_gp, the posterior mean and covariance matrix are computed.

Value

A list of predictive posterior mean and covariance:

- mup: vector of predicted posterior mean
- Sigmap: predictive posterior covariance matrix
- R: predictive posterior covariance matrix with the scale parameter removed

Examples

```
### test function ###
f_x <- function(x) {
return(sin(2*pi*x) + x^2)
}

### training data ###
n <- 8
x <- runif(n, 0, 1)
y <- f_x(x)

### testing data ###
n.test <- 100
x.test <- runif(n.test, 0, 1)
y.test <- f_x(x.test)

### get parameter estimates ###
out <- mle_gp(y, x)

### prediction ###
pred <- predict_gp(out, x.test)
```

`score` *score*

Description

Calculates a score value. Higher score values indicate better fits.

Usage

```
score(y, mu, sigma, md = FALSE)
```

Arguments

`y`	response variable vector
`mu`	predicted mean vector
`sigma`	predicted covariance matrix
`md`	logical indicating whether to return to a Mahalanobis distance value (`md = TRUE`) and score value or only a score value (`md = FALSE`)

Value

a numerical value

Examples

```
### test function ###
f_x <- function(x) {
return(sin(2*pi*x) + x^2)
}

### training data ###
n <- 8
x <- runif(n, 0, 1)
y <- f_x(x)

### testing data ###
n.test <- 100
x.test <- runif(n.test, 0, 1)
y.test <- f_x(x.test)

### get parameter estimates ###
out <- mle_gp(y, x)

### prediction ###
pred <- predict_gp(out, x.test)

### get score value ###
score_value <- score(y.test, pred$mup, pred$Sigmap)
```

Index