Package ‘binaryGP’

October 12, 2022

Type Package

Title Fit and Predict a Gaussian Process Model with (Time-Series) Binary Response

Version 0.2

Date 2017-09-17

Author Chih-Li Sung

Maintainer Chih-Li Sung <iamdfchile@gmail.com>

Description Allows the estimation and prediction for binary Gaussian process model. The mean function can be assumed to have time-series structure. The estimation methods for the unknown parameters are based on penalized quasi-likelihood/penalized quasi-partial likelihood and restricted maximum likelihood. The predicted probability and its confidence interval are computed by Metropolis-Hastings algorithm. More details can be seen in Sung et al (2017) <arXiv:1705.02511>.

License GPL-2 | GPL-3

LazyData TRUE

Imports Rcpp (>= 0.12.0), lhs (>= 0.10), logitnorm (>= 0.8.29), nloptr (>= 1.0.4), GPfit (>= 1.0-0), stats, graphics, utils, methods

LinkingTo Rcpp, RcppArmadillo

RoxygenNote 5.0.1

Depends R (>= 2.14.1)

NeedsCompilation yes

Repository CRAN

Date/Publication 2017-09-19 08:34:21 UTC

R topics documented:

binaryGP_fit *Binary Gaussian Process (with/without time-series)*

Description

The function fits Gaussian process models with binary response. The models can also include time-series term if a time-series binary response is observed. The estimation methods are based on PQL/PQPL and REML (for mean function and correlation parameter, respectively).

Usage

```
binaryGP_fit(X, Y, R = 0, L = 0, corr = list(type = "exponential", power =
  2), nugget = 1e-10, constantMean = FALSE, orthogonalGP = FALSE,
  converge.tol = 1e-10, maxit = 15, maxit.PQPL = 20, maxit.REML = 100,
  xtol_rel = 1e-10, standardize = FALSE, verbose = TRUE)
```

Arguments

`X`	a design matrix with dimension `n` by `d`.
`Y`	a response matrix with dimension `n` by `T`. The values in the matrix are 0 or 1. If no time-series, `T = 1`. If time-series is included, i.e., `T > 1`, the first column is the response vector of time 1, and second column is the response vector of time 2, and so on.
`R`	a positive integer specifying the order of autoregression only if time-series is included. The default is 1.
`L`	a positive integer specifying the order of interaction between `X` and previous `Y` only if time-series is included. The value couldn't nbe larger than R. The default is 1.
`corr`	a list of parameters for the specifing the correlation to be used. Either exponential correlation function or Matern correlation function can be used. See `corr_matrix` for details.
`nugget`	a positive value to use for the nugget. If `NULL`, a nugget will be estimated. The default is 1e-10.
`constantMean`	logical. `TRUE` indicates that the Gaussian process will have a constant mean, plus time-series structure if `R` or `T` is greater than one; otherwise the mean function will be a linear regression in X, plus an intercept term and time-series structure.
`orthogonalGP`	logical. `TRUE` indicates that the orthogonal Gaussian process will be used. Only available when `corr` is `list(type = "exponential", power = 2)`.
`converge.tol`	convergence tolerance. It converges when relative difference with respect to η_t is smaller than the tolerance. The default is 1e-10.
`maxit`	a positive integer specifying the maximum number of iterations for estimation to be performed before the estimates are convergent. The default is 15.
`maxit.PQPL`	a positive integer specifying the maximum number of iterations for PQL/PQPL estimation to be performed before the estimates are convergent. The default is 20.

`maxit.REML`	a positive integer specifying the maximum number of iterations in `lbfgs` for REML estimation to be performed before the estimates are convergent. The default is 100.
`xtol_rel`	a postive value specifying the convergence tolerance for `lbfgs`. The default is 1e-10.
`standardize`	logical. If `TRUE`, each column of X will be standardized into `[0,1]`. The default is `FALSE`.
`verbose`	logical. If `TRUE`, additional diagnostics are printed. The default is `TRUE`.

Details

Consider the model

$$logit(p_t(x)) = \eta_t(x) = \sum_{r=1}^{R} \varphi_r y_{t-r}(\mathbf{x}) + \alpha_0 + \mathbf{x}'\boldsymbol{\alpha} + \sum_{l=1}^{L} \boldsymbol{\gamma}_l \mathbf{x} y_{t-l}(\mathbf{x}) + Z_t(\mathbf{x}),$$

where $p_t(x) = Pr(y_t(x) = 1)$ and $Z_t(\cdot)$ is a Gaussian process with zero mean, variance σ^2, and correlation function $R_{\boldsymbol{\theta}}(\cdot,\cdot)$ with unknown correlation parameters $\boldsymbol{\theta}$. The power exponential correlation function is used and defined by

$$R_{\boldsymbol{\theta}}(\mathbf{x}_i, \mathbf{x}_j) = \exp\{-\sum_{l=1}^{d} \frac{(x_{il} - x_{jl})^p}{\theta_l}\},$$

where p is given by `power`. The parameters in the mean functions including $\varphi_r, \alpha_0, \boldsymbol{\alpha}, \boldsymbol{\gamma}_l$ are estimated by PQL/PQPL, depending on whether the mean function has the time-series structure. The parameters in the Gaussian process including $\boldsymbol{\theta}$ and σ^2 are estimated by REML. If `orthogonalGP` is `TRUE`, then orthogonal covariance function (Plumlee and Joseph, 2016) is employed. More details can be seen in Sung et al. (unpublished).

Value

`alpha_hat`	a vector of coefficient estimates for the linear relationship with X.
`varphi_hat`	a vector of autoregression coefficient estimates.
`gamma_hat`	a vector of the interaction effect estimates.
`theta_hat`	a vector of correlation parameters.
`sigma_hat`	a value of sigma estimate (standard deviation).
`nugget_hat`	if `nugget` is `NULL`, then return a nugget estimate, otherwise return `nugget`.
`orthogonalGP`	`orthogonalGP`.
`X`	data `X`.
`Y`	data `Y`.
`R`	order of autoregression.
`L`	order of interaction between X and previous Y.
`corr`	a list of parameters for the specifing the correlation to be used.
`Model.mat`	model matrix.

eta_hat	eta_hat.
standardize	standardize.
mean.x	mean of each column of X. Only available when standardize=TRUE, otherwise mean.x=NULL.
scale.x	max(x)-min(x) of each column of X. Only available when standardize=TRUE, otherwise scale.x=NULL.

Author(s)

Chih-Li Sung <iamdfchile@gmail.com>

See Also

predict.binaryGP for prediction of the binary Gaussian process, print.binaryGP for the list of estimation results, and summary.binaryGP for summary of significance results.

Examples

```
library(binaryGP)

#####      Testing function: cos(x1 + x2) * exp(x1*x2) with TT sequences      #####
#####   Thanks to Sonja Surjanovic and Derek Bingham, Simon Fraser University #####
test_function <- function(X, TT)
{
  x1 <- X[,1]
  x2 <- X[,2]

  eta_1 <- cos(x1 + x2) * exp(x1*x2)

  p_1 <- exp(eta_1)/(1+exp(eta_1))
  y_1 <- rep(NA, length(p_1))
  for(i in 1:length(p_1)) y_1[i] <- rbinom(1,1,p_1[i])
  Y <- y_1
  P <- p_1
  if(TT > 1){
    for(tt in 2:TT){
      eta_2 <- 0.3 * y_1 + eta_1
      p_2 <- exp(eta_2)/(1+exp(eta_2))
      y_2 <- rep(NA, length(p_2))
      for(i in 1:length(p_2)) y_2[i] <- rbinom(1,1,p_2[i])
      Y <- cbind(Y, y_2)
      P <- cbind(P, p_2)
      y_1 <- y_2
    }
  }

  return(list(Y = Y, P = P))
}

set.seed(1)
n <- 30
```

```
n.test <- 10
d <- 2
X <- matrix(runif(d * n), ncol = d)

##### without time-series #####
Y <- test_function(X, 1)$Y  ## Y is a vector

binaryGP.model <- binaryGP_fit(X = X, Y = Y)
print(binaryGP.model)   # print estimation results
summary(binaryGP.model) # significance results

##### with time-series, lag 1 #####
Y <- test_function(X, 10)$Y  ## Y is a matrix with 10 columns

binaryGP.model <- binaryGP_fit(X = X, Y = Y, R = 1)
print(binaryGP.model)   # print estimation results
summary(binaryGP.model) # significance results
```

`predict.binaryGP` *Predictions of Binary Gaussian Process*

Description

The function computes the predicted response and its variance as well as its confidence interval.

Usage

```
## S3 method for class 'binaryGP'
predict(object, xnew, conf.level = 0.95,
  sim.number = 101, ...)
```

Arguments

`object`	a class binaryGP object estimated by `binaryGP_fit`.
`xnew`	a testing matrix with dimension `n_new` by `d` in which each row corresponds to a predictive location.
`conf.level`	a value from 0 to 1 specifying the level of confidence interval. The default is 0.95.
`sim.number`	a positive integer specifying the simulation number for Monte-Carlo method. The default is 101.
`...`	for compatibility with generic method `predict`.

Value

- `mean` — a matrix with dimension `n_new` by `T` displaying predicted responses at locations `xnew`.
- `var` — a matrix with dimension `n_new` by `T` displaying predictive variances at locations `xnew`.
- `upper.bound` — a matrix with dimension `n_new` by `T` displaying upper bounds with `conf.level` confidence level.
- `lower.bound` — a matrix with dimension `n_new` by `T` displaying lower bounds with `conf.level` confidence level.
- `y_pred` — a matrix with dimension `n_new` by `T` displaying predicted binary responses at locations `xnew`.

Author(s)

Chih-Li Sung <iamdfchile@gmail.com>

See Also

`binaryGP_fit` for estimation of the binary Gaussian process.

Examples

```
library(binaryGP)

#####      Testing function: cos(x1 + x2) * exp(x1*x2) with TT sequences      #####
#####   Thanks to Sonja Surjanovic and Derek Bingham, Simon Fraser University #####
test_function <- function(X, TT)
{
  x1 <- X[,1]
  x2 <- X[,2]

  eta_1 <- cos(x1 + x2) * exp(x1*x2)

  p_1 <- exp(eta_1)/(1+exp(eta_1))
  y_1 <- rep(NA, length(p_1))
  for(i in 1:length(p_1)) y_1[i] <- rbinom(1,1,p_1[i])
  Y <- y_1
  P <- p_1
  if(TT > 1){
    for(tt in 2:TT){
      eta_2 <- 0.3 * y_1 + eta_1
      p_2 <- exp(eta_2)/(1+exp(eta_2))
      y_2 <- rep(NA, length(p_2))
      for(i in 1:length(p_2)) y_2[i] <- rbinom(1,1,p_2[i])
      Y <- cbind(Y, y_2)
      P <- cbind(P, p_2)
      y_1 <- y_2
    }
  }
```

```
  return(list(Y = Y, P = P))
}

set.seed(1)
n <- 30
n.test <- 10
d <- 2
X <- matrix(runif(d * n), ncol = d)
X.test <- matrix(runif(d * n.test), ncol = d)

##### without time-series #####
Y <- test_function(X, 1)$Y  ## Y is a vector
test.out <- test_function(X.test, 1)
Y.test <- test.out$Y
P.true <- test.out$P

# fitting
binaryGP.model <- binaryGP_fit(X = X, Y = Y)

# prediction
binaryGP.prediction <- predict(binaryGP.model, xnew = X.test)
print(binaryGP.prediction$mean)
print(binaryGP.prediction$var)
print(binaryGP.prediction$upper.bound)
print(binaryGP.prediction$lower.bound)

##### with time-series #####
Y <- test_function(X, 10)$Y  ## Y is a matrix with 10 columns
test.out <- test_function(X.test, 10)
Y.test <- test.out$Y
P.true <- test.out$P

# fitting
binaryGP.model <- binaryGP_fit(X = X, Y = Y, R = 1)

# prediction
binaryGP.prediction <- predict(binaryGP.model, xnew = X.test)
print(binaryGP.prediction$mean)
print(binaryGP.prediction$var)
print(binaryGP.prediction$upper.bound)
print(binaryGP.prediction$lower.bound)
```

`print.binaryGP` *Print Fitted results of Binary Gaussian Process*

Description

The function shows the estimation results by `binaryGP_fit`.

Usage

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

Arguments

`x`	a class binaryGP object estimated by binaryGP_fit.
`...`	for compatibility with generic method `print`.

Value

a list of estimates by binaryGP_fit.

Author(s)

Chih-Li Sung <iamdfchile@gmail.com>

See Also

`binaryGP_fit` for estimation of the binary Gaussian process.

Examples

```
library(binaryGP)

#####      Testing function: cos(x1 + x2) * exp(x1*x2) with TT sequences      #####
#####   Thanks to Sonja Surjanovic and Derek Bingham, Simon Fraser University #####
test_function <- function(X, TT)
{
  x1 <- X[,1]
  x2 <- X[,2]

  eta_1 <- cos(x1 + x2) * exp(x1*x2)

  p_1 <- exp(eta_1)/(1+exp(eta_1))
  y_1 <- rep(NA, length(p_1))
  for(i in 1:length(p_1)) y_1[i] <- rbinom(1,1,p_1[i])
  Y <- y_1
  P <- p_1
  if(TT > 1){
    for(tt in 2:TT){
      eta_2 <- 0.3 * y_1 + eta_1
      p_2 <- exp(eta_2)/(1+exp(eta_2))
      y_2 <- rep(NA, length(p_2))
      for(i in 1:length(p_2)) y_2[i] <- rbinom(1,1,p_2[i])
      Y <- cbind(Y, y_2)
      P <- cbind(P, p_2)
      y_1 <- y_2
    }
  }
```

```
  return(list(Y = Y, P = P))
}

set.seed(1)
n <- 30
n.test <- 10
d <- 2
X <- matrix(runif(d * n), ncol = d)

##### without time-series #####
Y <- test_function(X, 1)$Y  ## Y is a vector

binaryGP.model <- binaryGP_fit(X = X, Y = Y)
print(binaryGP.model)   # print estimation results
summary(binaryGP.model) # significance results

##### with time-series, lag 1 #####
Y <- test_function(X, 10)$Y  ## Y is a matrix with 10 columns

binaryGP.model <- binaryGP_fit(X = X, Y = Y, R = 1)
print(binaryGP.model)   # print estimation results
summary(binaryGP.model) # significance results
```

`summary.binaryGP` *Summary of Fitting a Binary Gaussian Process*

Description

The function summarizes estimation and significance results by `binaryGP_fit`.

Usage

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

Arguments

`object`	a class binaryGP object estimated by `binaryGP_fit`.
`...`	for compatibility with generic method `summary`.

Value

A table including the estimates by `binaryGP_fit`, and the correponding standard deviations, Z-values and p-values.

Author(s)

Chih-Li Sung <iamdfchile@gmail.com>

See Also

binaryGP_fit for estimation of the binary Gaussian process.

Examples

```
library(binaryGP)

#####      Testing function: cos(x1 + x2) * exp(x1*x2) with TT sequences      #####
#####   Thanks to Sonja Surjanovic and Derek Bingham, Simon Fraser University #####
test_function <- function(X, TT)
{
  x1 <- X[,1]
  x2 <- X[,2]

  eta_1 <- cos(x1 + x2) * exp(x1*x2)

  p_1 <- exp(eta_1)/(1+exp(eta_1))
  y_1 <- rep(NA, length(p_1))
  for(i in 1:length(p_1)) y_1[i] <- rbinom(1,1,p_1[i])
  Y <- y_1
  P <- p_1
  if(TT > 1){
    for(tt in 2:TT){
      eta_2 <- 0.3 * y_1 + eta_1
      p_2 <- exp(eta_2)/(1+exp(eta_2))
      y_2 <- rep(NA, length(p_2))
      for(i in 1:length(p_2)) y_2[i] <- rbinom(1,1,p_2[i])
      Y <- cbind(Y, y_2)
      P <- cbind(P, p_2)
      y_1 <- y_2
    }
  }

  return(list(Y = Y, P = P))
}

set.seed(1)
n <- 30
n.test <- 10
d <- 2
X <- matrix(runif(d * n), ncol = d)

##### without time-series #####
Y <- test_function(X, 1)$Y  ## Y is a vector

binaryGP.model <- binaryGP_fit(X = X, Y = Y)
print(binaryGP.model)   # print estimation results
summary(binaryGP.model) # significance results

##### with time-series, lag 1 #####
Y <- test_function(X, 10)$Y  ## Y is a matrix with 10 columns
```

```
binaryGP.model <- binaryGP_fit(X = X, Y = Y, R = 1)
print(binaryGP.model)   # print estimation results
summary(binaryGP.model) # significance results
```

Index