Package 'SLGP'

February 12, 2024

Type Package

Title Spatial Logistic Gaussian Process for Field Density Estimation

Version 0.0.0.9000

Maintainer Athénaïs Gautier <athenais.gautier@unibe.ch>

Description Spatial Logistic Gaussian Process for field density estimation with emphasis on Bayesian spectral methods.

License GPL (>= 3)

Encoding UTF-8

Roxygen list(markdown = TRUE)

RoxygenNote 7.3.0

Biarch true

Depends ggplot2, R (>= 3.4.0), stats

Imports DiceDesign, dplyr, methods, mvnfast, Rcpp (>= 0.12.0), RcppParallel (>= 5.0.1), rstan (>= 2.18.1), rstantools (>= 2.3.1.1), tidyr, truncnorm

LinkingTo BH (>= 1.66.0), Rcpp (>= 0.12.0), RcppEigen (>= 0.3.3.3.0), RcppParallel (>= 5.0.1), rstan (>= 2.18.1), StanHeaders (>= 2.21.0)

SystemRequirements GNU make

R topics documented:

`SLGP-package` *SLGP: A package for performing sample-based estimation of spatially dependent probability distributions.*

Description

SLGP: A package for performing sample-based estimation of spatially dependent probability distributions.

SLGP functions

The SLGP functions ...

References

Gautier, Athénaïs. (2023). *Modelling and predicting distribution-valued fields with applications to inversion under uncertainty*. PhD Thesis, Institute of Mathematical Statistics and Actuarial Science, University of Bern.

`check_basisfun_opts` *Check basis function parameters*

Description

This function checks the parameters specified type of basis function, and whether they are consistent. If some values are missing, it fills them with defaults.

Usage

```
check_basisfun_opts(basisFunctionsUsed, dimension, opts_BasisFun = list())
```

Arguments

`basisFunctionsUsed`
: Character. The type of basis function to use. Possible values: "inducing points", "RFF", "Discrete FF", "filling FF", "custom cosines".

`dimension`
: Numeric. The dimension of the index $[\mathbf{x}, t]$.

`opts_BasisFun` List. Options specific to the chosen basis function. If the type is "custom cosines", the basis functions considered are: coef $\cdot \cos(\text{freq}^\top [x, t] + \text{offset})$ and the user must provide three vectors: `opts_BasisFun$freq`, `opts_BasisFun$offset` and `opts_BasisFun$coef`. Users can refer to the documentation of specific basis function initialization functions (e.g., `initialize_basisfun_inducingpt`, `initialize_basisfun_RFF`, `initialize_basisfun_fillingRFF`, `initialize_basisfun_discre` etc.) for details on the available options.

Value

A list containing the initialized parameters necessary for evaluating the specified basis function.

`crossdist` *Computes the Euclidean distance between rows of two matrices*

Description

Computes the Euclidean distance between rows of two matrices

Usage

```
crossdist(x, y)
```

Arguments

`x`	First matrix
`y`	Second matrix

Value

Euclidean distance between rows of `x` and `y`

`evaluate_basis_functions`
Evaluate a Basis of Functions at Given Locations

Description

Evaluate a basis of functions at given locations.

Usage

```
evaluate_basis_functions(parameters, X, lengthscale)
```

Arguments

`parameters`	A list containing outputs of initialize_basisfun.
`X`	A design matrix containing locations where we want to evaluate the function.
`lengthscale`	Numeric vector containing the lengthscales to use for the kernel.

Value

A matrix with the evaluated basis functions.

`heuristic_find_variance` *Heuristic function to find the range of values of the unconditioned SLGP.*

Description

A Heuristic function to find the empirical range of the unconditioned SLGP

Usage

```
heuristic_find_variance(
  parameters,
  nsimu = 1000,
  grid_size = 101,
  plot = FALSE
)
```

Arguments

`parameters`	A list containing outputs of initialize_basisfun.
`nsimu`	An integer giving the number of unconditional simulations to use.
`grid_size`	An integer giving the number of nodes in each dimension of the grid to consider.
`plot`	A boolean indicating whether a ngraphical output is produced.

Value

A matrix with the evaluated basis functions.

`initialize_basisfun` *Initialize basis functions parameters*

Description

This function initializes the basis function's parameters based on the specified type of basis function.

Usage

```
initialize_basisfun(basisFunctionsUsed, dimension, opts_BasisFun = list())
```

Arguments

basisFunctionsUsed	Character. The type of basis function to use. Possible values: "inducing points", "RFF", "Discrete FF", "filling FF", "custom cosines".
dimension	Numeric. The dimension of the index $[\mathbf{x}, t]$.
opts_BasisFun	List. Optional. Additional options specific to the chosen basis function. If the type is "custom cosines", the basis functions considered are $coef\cos(freq^\top[x,t]+offset)$ and the user must provide three vectors: `opts_BasisFun$freq`, `opts_BasisFun$offset` and `opts_BasisFun$coef`. Users can refer to the documentation of specific basis function initialization functions (e.g., `initialize_basisfun_inducingpt`, `initialize_basisfun_RFF`, `initialize_basisfun_fillingRFF`, `initialize_basisfun_discre` etc.) for details on the available options. #'

Value

List. A list containing the initialized parameters necessary for evaluating the specified basis function.

---

initialize_basisfun_discreteFF

Initialize parameters for basis functions based on discrete Fourier Features.

Description

This function initializes parameters for basis functions based on discrete Fourier Features.

Usage

```
initialize_basisfun_discreteFF(dimension, maxOrdert, maxOrderx)
```

Arguments

dimension	Numeric. The dimension of the index $[\mathbf{x}, t]$.
maxOrdert	Numeric. Maximum frequency in t.
maxOrderx	Numeric. Maximum frequency in x.

Value

List. A list containing the initialized parameters necessary for evaluating the specified basis function.

Examples

```
1+1
```

`initialize_basisfun_fillingRFF` *Initialize parameters for basis functions based on space-filling Random Fourier Features.*

Description

This function initializes parameters for basis functions based on space-filling Random Fourier Features (for Matérn kernels only).

Usage

```
initialize_basisfun_fillingRFF(
  dimension,
  nFreq,
  MatParam = 5/2,
  lengthscale,
  seed = 0
)
```

Arguments

`dimension`	Numeric. The dimension of the index $[\mathbf{x}, t]$.
`nFreq`	Numeric. Number of frequencies to sample.
`MatParam`	Numeric, specifying the parameter of the Matérn kernel considered (default = 5/2).
`lengthscale`	Numeric vector containing the lengthscales to use for the kernel.

Value

List. A list containing the initialized parameters necessary for evaluating the specified basis function.

Examples

```
1+1
```

`initialize_basisfun_inducingpt` *Initialize parameters for Inducing points based functions.*

Description

This function initializes parameters for basis functions based on inducing points.

Usage

```
initialize_basisfun_inducingpt(
  dimension,
  kernel = "Mat52",
  lengthscale,
  pointscoord = NULL,
  numberPoints = NULL
)
```

Arguments

dimension	Numeric. The dimension of the index $[\mathbf{x}, t]$.
kernel	Character, specifying the kernel to be consider among "Gaussian", "Exp", "Mat32" and "Mat52" (default "Mat52").
lengthscale	Numeric vector containing the lengthscales to use for the kernel.
pointscoord	Optional matrix with the coordinates of the inducing points. If none is provided, we sample them uniformly in the unit hypercube.
numberPoints	Optional numerical value specifying the number of inducing points to sample (ignored if pointscoord is specified)

Value

List. A list containing the upper triangular factor of the Cholesky decomposition of the kernel matrix as well as its inverse.

Examples

```
1+1
```

---

initialize_basisfun_RFF

Initialize parameters basis functions based on Random Fourier Features.

Description

This function initializes parameters for basis functions based on Random Fourier Features (for Matérn kernels only).

Usage

```
initialize_basisfun_RFF(dimension, nFreq, MatParam = 5/2, lengthscale)
```

Arguments

dimension	Numeric. The dimension of the index $[\mathbf{x}, t]$.
nFreq	Numeric. Number of frequencies to sample.
MatParam	Numeric, specifying the parameter of the Matérn kernel considered (default = 5/2).
lengthscale	Numeric vector containing the lengthscales to use for the kernel.

Value

List. A list containing the initialized parameters necessary for evaluating the specified basis function.

Examples

```
1+1
```

`normalize_data` *Normalize Data to the Range 0, 1*

Description

This function takes a dataframe, a vector of predictor names, a response name, and optional range information, and normalizes the data to the range 0, 1. If range information is not provided, the observed values in the data are used to determine the range.

Usage

```
normalize_data(
  data,
  predictorNames,
  responseName,
  predictorsUpper = NULL,
  predictorsLower = NULL,
  responseRange = NULL
)
```

Arguments

`data`	A dataframe containing the dataset.
`predictorNames`	A character vector specifying the names of the predictor variables.
`responseName`	A character string specifying the name of the response variable.
`predictorsUpper`	A numeric vector representing the upper range for the predictors (optional).
`predictorsLower`	A numeric vector representing the lower range for the predictors (optional).
`responseRange`	A numeric vector representing the upper and lower range for the response (optional).

Value

A dataframe with the normalized values.

Examples

```
data <- data.frame(x1 = c(1, 2, 3), x2 = c(4, 5, 6), y = c(10, 20, 30))
normalized_data <- normalize_data(data, c("x1", "x2"), "y")
```

predictSLGP_cdf *Perform prediction at candidate points of the cdf(s) in a SLGP model.*

Description

Perform prediction at candidate points of the cdf(s) in a SLGP model.

Usage

```
predictSLGP_cdf(
  SLGPmodel,
  newNodes,
  interpolateBasisFun = "WNN",
  nIntegral = 101,
  nDiscret = 101
)
```

Arguments

SLGPmodel	An object of class SLGP.
newNodes	A data frame containing the new points at which we want the SLGP(s) evaluated
interpolateBasisFun	String specifying whether the basis functions are evaluated on all points ("nothing"), on the closest neighbour of a regular grid ("NN" - default) or with a weighted inverse distance to the closest neighbours ("WID").
nIntegral	Number of points used to approximate the integral.
nDiscret	Integer, optional, discretization step used if "interpolateBasisFun" is "NN" or "WNN".

Value

A list containing the results of the SLGP regression.

predictSLGP_newNode *Perform prediction at candidate points of a slgp model.*

Description

Perform prediction at candidate points of a slgp model.

Usage

```
predictSLGP_newNode(
  SLGPmodel,
  newNodes,
  interpolateBasisFun = "WNN",
  nIntegral = 101,
  nDiscret = 101
)
```

Arguments

SLGPmodel	An object of class SLGP.
newNodes	A data frame containing the new points at which we want the SLGP(s) evaluated
interpolateBasisFun	String specifying whether the basis functions are evaluated on all points ("nothing"), on the closest neighbour of a regular grid ("NN" - default) or with a weighted inverse distance to the closest neighbours ("WID").
nIntegral	Number of points used to approximate the integral.
nDiscret	Integer, optional, discretization step used if "interpolateBasisFun" is "NN" or "WNN".

Value

A list containing the results of the SLGP regression.

---

pre_comput_NN *Compute intermediate quantities for basis functions evaluation (when a Nearest neighbour approximation is done)*

Description

This function takes normalized data, predictor names, response name, and computes intermediate quantities useful for later evaluation of basis functions.

Usage

```
pre_comput_NN(
  normalizedData,
  predictorNames,
  responseName,
  nIntegral = 101,
  nDiscret = 51
)
```

Arguments

normalizedData	A dataframe containing the normalized dataset.
predictorNames	A character vector specifying the names of the predictor variables.
responseName	A character string specifying the name of the response variable.
nIntegral	Number of points used to approximate the integral.
nDiscret	Number of points used to discretize the predictors' domain.

Value

A list containing intermediate quantities for basis function evaluation.

Examples

```
data <- data.frame(x1 = c(0.1, 0.5, 0.9), x2 = c(0.2, 0.6, 1.0), y = c(0.3, 0.7, 1.0))
normalized_data <- normalize_data(data, c("x1", "x2"), "y")
intermediate_quantities <- pre_comput_NN(normalized_data, c("x1", "x2"), "y")
```

---

pre_comput_nothing *Compute intermediate quantities for basis functions evaluation (when no interpolation is done)*

Description

This function takes normalized data, predictor names, response name, and computes intermediate quantities useful for later evaluation of basis functions.

Usage

```
pre_comput_nothing(
  normalizedData,
  predictorNames,
  responseName,
  nIntegral = 51
)
```

Arguments

normalizedData	A dataframe containing the normalized dataset.
predictorNames	A character vector specifying the names of the predictor variables.
responseName	A character string specifying the name of the response variable.
nIntegral	Number of points used to approximate the integral.

Value

A list containing intermediate quantities for basis function evaluation.

Examples

```
data <- data.frame(x1 = c(0.1, 0.5, 0.9), x2 = c(0.2, 0.6, 1.0), y = c(0.3, 0.7, 1.0))
ndata <- normalize_data(data, c("x1", "x2"), "y")
intermediate_quantities <- pre_comput_nothing(ndata, c("x1", "x2"), "y")
```

`pre_comput_WNN` *Compute intermediate quantities for basis functions evaluation (when a weighted nearest neighbours interpolation is done)*

Description

This function takes normalized data, predictor names, response name, and computes intermediate quantities useful for later evaluation of basis functions.

Usage

```
pre_comput_WNN(
  normalizedData,
  predictorNames,
  responseName,
  nIntegral = 101,
  nDiscret = 51
)
```

Arguments

`normalizedData`	A dataframe containing the normalized dataset.
`predictorNames`	A character vector specifying the names of the predictor variables.
`responseName`	A character string specifying the name of the response variable.
`nIntegral`	Number of points used to approximate the integral.
`nDiscret`	Number of points used to discretize the predictors' domain.

Value

A list containing intermediate quantities for basis function evaluation.

Examples

```
data <- data.frame(x1 = c(0.1, 0.5, 0.9), x2 = c(0.2, 0.6, 1.0), y = c(0.3, 0.7, 1.0))
normalized_data <- normalize_data(data, c("x1", "x2"), "y")
intermediate_quantities <- pre_comput_WNN(normalized_data, c("x1", "x2"), "y")
```

`retrainSLGP` *Retrain a SLGP by changing the data and/or method.*

Description

Creates a `slgp` object and performs the training using either a Bayesian MCMC estimation, a MAP estimation or a Laplace approximation (i.e. MAP + Laplace).

Usage

```
retrainSLGP(
  SLGPmodel,
  newdata = NULL,
  epsilonStart = NULL,
  method,
  interpolateBasisFun = "NN",
  nIntegral = 51,
  nDiscret = 51,
  hyperparams = NULL,
  sigmaEstimationMethod = "none",
  seed = NULL,
  opts = list()
)
```

Arguments

SLGPmodel	A SLGP.
newdata	An optional data frame containing the variables in the formula.
epsilonStart	An optional numeric vector, the starting weights in the finite-rank GP: $Z(x,t) = \sum_{i=1}^{p} \epsilon_i f_i(x,t)$
method	The method to be used among "MCMC", "MAP", "Laplace".
interpolateBasisFun	String specifying whether the basis functions are evaluated on all points ("nothing"), on the closest neighbour of a regular grid ("NN" - default) or with a weighted inverse distance to the closest neighbours ("WID").
nIntegral	Number of points used to approximate the integral.
nDiscret	Integer, optional, discretization step used if "interpolateBasisFun" is "NN" or "WNN".
hyperparams	Optional hyper-parameter values. It should be a list with sigma and a vector for lengthscale.
sigmaEstimationMethod	Method for estimating sigma2 ("none" (default) or "heuristic").
opts	Optional list of extra parameters, typically for the MCMC or optimisation.

Value

A list containing the results of the SLGP regression.

---

rosenblatt_transform_multivarStudent

Auxiliary function: performs the Rosenblatt transform from the multivariate uniform distribution to the multivariate student distribution that is a Matérn's kernel spectral density

---

Description

Auxiliary function: performs the Rosenblatt transform from the multivariate uniform distribution to the multivariate student distribution that is a Matérn's kernel spectral density

Usage

```
rosenblatt_transform_multivarStudent(x, dimension, MatParam = 5/2)
```

Arguments

x	vector or matrix, the points to be transformed.
dimension	Integer. The dimension of the problem.
MatParam	Numeric, specifying the parameter of the Matérn kernel considered (default = 5/2).

Value

The transformed coordinates of x.

Examples

```
data <- matrix(c(0, 0, 0.1, 0.9, 0.5, 0.5, 0.1, 0.1), ncol=2, byrow=TRUE)
rosenblatt_transform_multivarStudent(x=data, dimension = 2)
```

---

sampleSLGP *Draw new samples from a SLGP model*

Description

Draw new samples from a SLGP model

Usage

```
sampleSLGP(
  SLGPmodel,
  newX,
  n,
  interpolateBasisFun = "NN",
  nIntegral = 51,
  nDiscret = 51,
  seed = NULL
)
```

Arguments

SLGPmodel	An object of class SLGP.
newX	A data frame containing the new points at which we want draws from a SLGP
n	An integer, (or vector of integers with length matching the number of rows in newPredictors) specifying the number of samples to be drawn.
interpolateBasisFun	String specifying whether the basis functions are evaluated on all points ("nothing"), on the closest neighbour of a regular grid ("NN" - default) or with a weighted inverse distance to the closest neighbours ("WID").
nIntegral	Number of points used to approximate the integral.

nDiscret	Integer, optional, discretization step used if "interpolateBasisFun" is "NN" or "WNN".
seed	Optional, to specify a seed

Value

A list containing the results of the SLGP regression.

`sample_spectral_Matern` *Draw Random Frequencies from the Spectral Density of Matérn Kernel*

Description

Sample frequencies from the Spectral density of a Matérn GP.

Usage

```
sample_spectral_Matern(dimension, order)
```

Arguments

dimension	The dimension of the space for the index $[\mathbf{x}, t]$.
order	Number of frequencies.

Value

A matrix of frequencies with `order` rows and `dimension` columns.

Examples

```
w <- sample_spectral_Matern(dimension = 1, order = 10000)
plot(density(w)); rug(w)
w <- sample_spectral_Matern(dimension = 2, order = 100)
```

`slgp` *Perform SLGP estimation using method of choice.*

Description

Creates a `slgp` object and performs the training using either a Bayesian MCMC estimation, a MAP estimation or a Laplace approximation (i.e. MAP + Laplace).

Usage

```
slgp(
  formula,
  data,
  epsilonStart = NULL,
  method,
  basisFunctionsUsed,
  interpolateBasisFun = "NN",
  nIntegral = 51,
  nDiscret = 51,
  hyperparams = NULL,
  predictorsUpper = NULL,
  predictorsLower = NULL,
  responseRange = NULL,
  sigmaEstimationMethod = "none",
  seed = NULL,
  opts_BasisFun = list(),
  BasisFunParam = NULL,
  opts = list()
)
```

Arguments

formula	A formula specifying the model.
data	A data frame containing the variables in the formula.
epsilonStart	An optional numeric vector, the starting weights in the finite-rank GP: $Z(x,t) = \sum_{i=1}^{p} \epsilon_i f_i(x,t)$
method	The method to be used among "MCMC", "MAP", "Laplace".
basisFunctionsUsed	String specifying the basis functions ("inducing points", "RFF", "Discrete FF", "filling FF", "custom cosines").
interpolateBasisFun	String specifying whether the basis functions are evaluated on all points ("nothing"), on the closest neighbour of a regular grid ("NN" - default) or with a weighted inverse distance to the closest neighbours ("WNN").
nIntegral	Number of points used to approximate the integral.
nDiscret	Integer, optional, discretization step used if "interpolateBasisFun" is "NN" or "WNN".
hyperparams	Optional hyper-parameter values. It should be a list with sigma and a vector for lengthscale.
predictorsUpper	An optional vector with the response upper range and lower range.
predictorsLower	An optional vector with the response upper range and lower range.
responseRange	An optional vector with the response upper range and lower range.
sigmaEstimationMethod	Method for estimating sigma2 ("none" (default) or "heuristic").
opts_BasisFun	List of extra parameters for the basis functions.
BasisFunParam	List to specify the basis functions
opts	Optional list of extra parameters, typically for the MCMC or optimisation.

Value

A list containing the results of the SLGP regression.

References

Gautier, Athénaïs (2023). "Modelling and Predicting Distribution-Valued Fields with Applications to Inversion Under Uncertainty." Thesis, Universität Bern, Bern. https://boristheses.unibe.ch/4377/ This thesis discusses modeling and predicting distribution-valued fields with Spatial Logistic Gaussian Processes.

`SLGP-class` *Spatial Logistic Gaussian Process Class*

Description

Spatial Logistic Gaussian Process Class

Slots

`formula` Formula specifying the covariates.

`data` A data frame containing the data to train the SLGP.

`responseName` A character, specifying the name of the response.

`covariateName` A character vector, specifying the names of the covariates

`responseRange` A vector with the response upper range and lower range.

`predictorsRange` A list containing the vector 'predictorsLower' specifying the covariate's lower range and the vector 'predictorsUpper' specifying the covariate's upper range

`method` The method used to train the SLGP among "MCMC", "MAP", "Laplace", "none".

`p` Number of basis functions.

`basisFunctionsUsed` String specifying the basis functions ("inducing points", "RFF", "Discrete FF", "filling FF", "custom cosines").

`opts_BasisFun` List of extra parameters for the basis functions.

`coefficients` Matrix of epsilon's values for the finite-rank GP: $Z(x,t) = \sum_{i=1}^{p} \epsilon_i f_i(x,t)$

`hyperparams` Hyper-parameter values.It should be a list with a numeric 'sigma' and a vector 'lengthscale'.

Index