Package 'BSPBSS'

September 2, 2022

Title Bayesian Spatial Blind Source Separation

Version 1.0.2

Description Gibbs sampling for Bayesian spatial blind source separation (BSP-BSS). BSP-BSS is designed for spatially dependent signals in high dimensional and large-scale data, such as neuroimaging. The method assumes the expectation of the observed images as a linear mixture of multiple sparse and piece-wise smooth latent source signals, and constructs a Bayesian nonparametric prior by thresholding Gaussian processes. Details can be found in our working paper: Ben et al. (2022+) ``Bayesian Spatial Blind Source Separation via the Thresholded Gaussian Process".

Depends R (>= 3.4.0), movMF

License GPL (>= 3)

Encoding UTF-8

RoxygenNote 7.2.0

LinkingTo Rcpp, RcppArmadillo

Imports rstiefel, Rcpp, ica, glmnet, gplots, BayesGPfit, svd, RandomFieldsUtils, neurobase, oro.nifti, gridExtra, ggplot2, gtools

SystemRequirements GNU make

Suggests knitr, rmarkdown

VignetteBuilder knitr

NeedsCompilation yes

Author Ben Wu [aut, cre], Ying Guo [aut], Jian Kang [aut]

Maintainer Ben Wu <wuben@ruc.edu.cn>

Repository CRAN

Date/Publication 2022-09-02 07:50:14 UTC

R topics documented:

`init_bspbss` *Initial values*

Description

Generate initial values, set up priors and perform kernel decomposition for the MCMC algorithm.

Usage

```
init_bspbss(
  X,
  coords,
  standardize = TRUE,
  q = 2,
  dens = 0.5,
  ker_par = c(0.05, 20),
  num_eigen = 500,
  noise = 0
)
```

Arguments

`X`	Data matrix with n rows (sample) and p columns (voxel).
`coords`	Cordinate matrix with p rows (voxel) and d columns (dimension).
`standardize`	If TRUE, standarize each row of X.
`q`	Number of latent sources.
`dens`	The initial density level (between 0 and 1) of the latent sources.
`ker_par`	2-dimensional vector (a,b) with a>0, b>0, specifing the parameters in the modified exponetial squared kernel.
`num_eigen`	Number of eigen functions.
`noise`	Gaussian noise added to the initial latent sources, with mean 0 and standard deviation being noise * sd(S0), where sd(S0) is the standard deviation of the initial latent sources.

Value

List containing initial values, priors and eigen functions/eigen values of the kernel of the Gaussian process.

Examples

```
sim = sim_2Dimage(length = 30, sigma = 5e-4, n = 30, smooth = 6)
ini = init_bspbss(sim$X, sim$coords, q = 3, ker_par = c(0.1,50), num_eigen = 50)
```

---

levelplot2D *levelplot for 2D images.*

Description

The function plots 2D images for a data matrix.

Usage

```
levelplot2D(
  S,
  coords,
  lim = c(min(S), max(S)),
  xlim = c(0, max(coords[, 1])),
  ylim = c(0, max(coords[, 2])),
  color = bluered(100),
  layout = c(1, nrow(S)),
  file = NULL
)
```

Arguments

S	Data matrix with q rows (sample) and p colums (pixel).
coords	Coordinates matrix with p rows (pixel) and 2 columns (dimension), specifying the coordinates of the data points.
lim	2-dimensional numeric vector, specifying the limits for the data.
xlim	2-dimensional numeric vector, specifying the lower and upper limits of x.
ylim	2-dimensional numeric vector, specifying the lower and upper limits of y.
color	Colorbar.
layout	2-dimensional numeric vector, specifying the number of rows and number of columns for the layout of components.
file	Name of the file to be saved.

Value

No return value.

Examples

```
sim = sim_2Dimage(length = 30, sigma = 5e-4, n = 30, smooth = 6)
levelplot2D(sim$S,lim = c(-0.04,0.04), sim$coords)
```

`mcmc_bspbss` *MCMC algorithm for Bayesian spatial blind source separation with the thresholded Gaussian Process prior.*

Description

Performan MCMC algorithm to draw samples from a Bayesian spatial blind source separation model.

Usage

```
mcmc_bspbss(
  X,
  init,
  prior,
  kernel,
  n.iter,
  n.burn_in,
  thin = 1,
  show_step,
  ep = 0.01,
  lr = 0.01,
  decay = 0.01,
  subsample_n = 0.5,
  subsample_p = 0.5
)
```

Arguments

`X`	Data matrix with n rows (sample) and p columns (voxel).
`init`	List of initial values, see `init_bspbss`.
`prior`	List of priors, see `init_bspbss`.
`kernel`	List including eigenvalues and eigenfunctions of the kernel, see `init_bspbss`.
`n.iter`	Total iterations in MCMC.
`n.burn_in`	Number of burn-in.
`thin`	Thining interval.

show_step	Frequency for printing the current number of iterations.
ep	Approximation parameter.
lr	Per-batch learning rate in SGHMC.
decay	Decay parameter in SGHMC.
subsample_n	Mini-batch size of samples.
subsample_p	Mini-batch size of voxels.

Value

List containing MCMC samples of: A, b, sigma, and zeta.

Examples

```
sim = sim_2Dimage(length = 30, sigma = 5e-4, n = 30, smooth = 6)
ini = init_bspbss(sim$X, sim$coords, q = 3, ker_par = c(0.1,50), num_eigen = 50)
res = mcmc_bspbss(ini$X,ini$init,ini$prior,ini$kernel,n.iter=200,n.burn_in=100,thin=10,show_step=50)
```

---

output_nii *Write a NIfTI file.*

Description

This function saves a data matrix into a NIfTI file.

Usage

```
output_nii(X, nii, xgrid, file = NULL, std = TRUE, thres = 0)
```

Arguments

X	Data matrix with n rows (sample) and p colums (pixel).
nii	a reference NIfTI-class object, representing a image with p voxels.
xgrid	Cordinate matrix with p rows (voxel) and d columns (dimension).
file	The name of the file to be saved.
std	If TRUE, standarize each row of X.
thres	Quantile to threshold each row of X.

Value

NIfTI-class object.

pre_nii *Transforms NIfTI to matrix*

Description

This function transforms a NIfTI-class object into a matrix.

Usage

```
pre_nii(nii, mask)
```

Arguments

nii	4D NIfTI-class object with dimensions x,y,z and t. Can be read from NIfTI file with `readNIfTI` function from the package `oro.nifti`.
mask	Mask variable, also in NIfTI format.

Value

List containing the data matrix with t rows and x*y*z colums (voxels), and the coordinates of the voxels.

sim_2Dimage *Simulate image data using ICA*

Description

The function simulates image data using a probabilistic ICA model whose latent components have specific spatial patterns.

Usage

```
sim_2Dimage(length = 20, n = 50, sigma = 0.002, smooth = 6)
```

Arguments

length	The length of the image.
n	sample size.
sigma	variance of the noise.
smooth	smoothness of the latent components.

Details

The observations are generated using probabilistic ICA:

$$X_i(v) = \sum_{j=1}^{q} A_{i,j} S_j(v) + \epsilon_i(v),$$

where $S_j, j = 1, ..., q$ are the latent components, $A_{i,j}$ is the mixing coeffecient and ϵ_i is the noise term. Specifically, the number of components in this function is $q = 3$, with each of them being a specific geometric shape. The mixing coefficient matrix is generated with a von Mises-Fisher distribution with the concentration parameter being zero, which means it is uniformly distributed on the sphere. ϵ_i is a i.i.d. Gaussian noise term with 0 mean and user-specified variance.

Value

List that contains the following terms:

X Data matrix with n rows (sample) and p columns (pixel).

coords Cordinate matrix with p rows (pixel) and d columns (dimension)

S Latent components.

A Mixing coefficent matrix.

snr Signal-to-noise ratio.

Examples

```
sim = sim_2Dimage(length = 30, sigma = 5e-4, n = 30, smooth = 6)
```

---

sum_mcmc_bspbss *Summarization of the MCMC result.*

Description

The function summarizes the MCMC results obtained from `mcmc_bspbss`.

Usage

```
sum_mcmc_bspbss(res, X, kernel, start = 1, end = 100, select_prob = 0.8)
```

Arguments

`res`	List including MCMC samples, which can be obtained from function `mcmc_bspbss`
`X`	Original data matrix.
`kernel`	List including eigenvalues and eigenfunctions of the kernel, see `init_bspbss`.
`start`	Start point of the iterations being summarized.
`end`	End point of the iterations being summarized.
`select_prob`	Lower bound of the posterior inclusion probability required when summarizing the samples of latent sources.

Value

List that contains the following terms:

S Estimated latent sources.

pip Voxel-wise posterior inclusion probability for the latent sources.

A Estimated mixing coefficent matrix.

zeta Estimated zeta.

sigma Estimated sigma.

logLik Trace of log-likelihood.

Slist MCMC samples of S.

Examples

```
sim = sim_2Dimage(length = 30, sigma = 5e-4, n = 30, smooth = 6)
ini = init_bspbss(sim$X, sim$coords, q = 3, ker_par = c(0.1,50), num_eigen = 50)
res = mcmc_bspbss(ini$X,ini$init,ini$prior,ini$kernel,n.iter=200,n.burn_in=100,thin=10,show_step=50)
res_sum = sum_mcmc_bspbss(res, ini$X, ini$kernel, start = 11, end = 20, select_p = 0.5)
```

Index