Package 'VBsparsePCA'

January 30, 2021

Type Package

Title The Variational Bayesian Method for Sparse PCA

Version 0.1.0

Author Bo (Yu-Chien) Ning

Maintainer The package maintainer <bo.ning@upmc.fr>

Description This package contains functions for a variational Bayesian approach for sparse PCA. The algorithm is the PX-CAVI algorithm proposed by Ning (2020) if assuming the jointly row-sparsity assumption for the loadings matrix and the batch PX-CAVI algorithm if otherwise. The outputs of the main function, VBsparsePCA, include the mean and covariance of the loadings matrix, the score functions, the variable selection results, and the estimated variance of the random noise.

Depends R (>= 3.6.0)

License GPL-3

Imports pracma, stats, utils

Encoding UTF-8

LazyData true

RoxygenNote 7.1.1

R topics documented:

`foldednorm.mean` *The function for obtaining the mean of a folded normal distribution*

Description

This function calculates the mean of the folded normal distribution given its location and scale parameters.

Usage

```
foldednorm.mean(mean, var)
```

Arguments

mean	Location parameter of the folded normal distribution.
var	Scale parameter of the folded normal distribution.

Details

The mean of the folded normal distribution with location μ and scale σ^2 is

$$\sigma\sqrt{2/\pi}\exp(-\mu^2/(2\sigma^2)) + \mu(1 - 2\Phi(-\mu/\sigma))$$

.

Value

foldednorm.mean

The mean of the folded normal distribution of iterations to reach convergence.

---

spca.cavi.Laplace *Function for the PX-CAVI algorithm using the Laplace slab*

Description

#' This function employs the PX-CAVI algorithm proposed in Ning (2020). The g in the slab density of the spike and slab prior is chosen to be the Laplace density, i.e., $N(0, \sigma^2/\lambda_1 I_r)$. Details of the model and the prior can be found in the Details section in the description of the 'VBsparsePCA' function. This function is not capable of handling the case when r > 1. In that case, we recommend to use the multivariate distribution instead.

Usage

```
spca.cavi.Laplace(
  x,
  r = 1,
  lambda = 1,
  max.iter = 100,
  eps = 0.001,
  sig2.true = NA,
  threshold = 0.5,
  theta.int = NA,
  theta.var.int = NA,
  kappa.para1 = NA,
  kappa.para2 = NA,
  sigma.a = NA,
  sigma.b = NA
)
```

Arguments

x	Data an $n * p$ matrix.
r	Rank.
lambda	Tuning parameter for the density g.
max.iter	The maximum number of iterations for running the algorithm.
eps	The convergence threshold; the default is 10^{-4}.
sig2.true	The default is false, $sigma^2$ will be estimated; if sig2 is known and its value is given, then $sigma^2$ will not be estimated.
threshold	The threshold to determine whether γ_j is 0 or 1; the default value is 0.5.
theta.int	The initial value of theta mean; if not provided, the algorithm will estimate it using PCA.
theta.var.int	The initial value of theta.var; if not provided, the algorithm will set it to be 1e-3*diag(r).
kappa.para1	The value of α_1 of $\pi(\kappa)$; default is 1.
kappa.para2	The value of α_2 of $\pi(\kappa)$; default is $p + 1$.
sigma.a	The value of σ_a of $\pi(\sigma^2)$; default is 1.
sigma.b	The value of σ_b of $\pi(\sigma^2)$; default is 2.

Value

iter	The number of iterations to reach convergence.
selection	A vector (if $r = 1$ or with the jointly row-sparsity assumption) or a matrix (if otherwise) containing the estimated value for $\boldsymbol{\gamma}$.
theta.mean	The loadings matrix.
theta.var	The covariance of each non-zero rows in the loadings matrix.
sig2	Variance of the noise.
obj.fn	A vector contains the value of the objective function of each iteration. It can be used to check whether the algorithm converges

---

`spca.cavi.mvn` *Function for the PX-CAVI algorithm using the multivariate normal slab*

Description

This function employs the PX-CAVI algorithm proposed in Ning (2020). The g in the slab density of the spike and slab prior is chosen to be the multivariate normal distribution, i.e., $N(0, \sigma^2/\lambda_1 I_r)$. Details of the model and the prior can be found in the Details section in the description of the 'VBsparsePCA' function.

Usage

```
spca.cavi.mvn(
  x,
  r,
  lambda = 1,
  max.iter = 100,
  eps = 1e-04,
  jointly.row.sparse = T,
  sig2.true = NA,
  threshold = 0.5,
  theta.int = NA,
  theta.var.int = NA,
  kappa.para1 = NA,
  kappa.para2 = NA,
  sigma.a = NA,
  sigma.b = NA
)
```

Arguments

x	Data an $n * p$ matrix.
r	Rank.
lambda	Tuning parameter for the density g.
max.iter	The maximum number of iterations for running the algorithm.
eps	The convergence threshold; the default is 10^{-4}.
jointly.row.sparse	The default is true, which means that the jointly row sparsity assumption is used; one could not use this assumptio by changing it to false.
sig2.true	The default is false, $sigma^2$ will be estimated; if sig2 is known and its value is given, then $sigma^2$ will not be estimated.
threshold	The threshold to determine whether γ_j is 0 or 1; the default value is 0.5.
theta.int	The initial value of theta mean; if not provided, the algorithm will estimate it using PCA.
theta.var.int	The initial value of theta.var; if not provided, the algorithm will set it to be 1e-3*diag(r).
kappa.para1	The value of α_1 of $\pi(\kappa)$; default is 1.
kappa.para2	The value of α_2 of $\pi(\kappa)$; default is $p + 1$.
sigma.a	The value of σ_a of $\pi(\sigma^2)$; default is 1.
sigma.b	The value of σ_b of $\pi(\sigma^2)$; default is 2.

Value

iter	The number of iterations to reach convergence.
selection	A vector (if $r = 1$ or with the jointly row-sparsity assumption) or a matrix (if otherwise) containing the estimated value for $\boldsymbol{\gamma}$.
theta.mean	The loadings matrix.
theta.var	The covariance of each non-zero rows in the loadings matrix.

sig2	Variance of the noise.
obj.fn	A vector contains the value of the objective function of each iteration. It can be used to check whether the algorithm converges

---

VBsparsePCA *The main function for the variational Bayesian method for sparse PCA*

Description

This function employs the PX-CAVI algorithm proposed in Ning (2021). The method uses the sparse spiked-covariance model and the spike and slab prior (see below). Two different slab densities can be used: independent Laplace densities and a multivariate normal density. In Ning (2021), it recommends choosing the multivariate normal distribution. The algorithm allows the user to decide whether she/he wants to center and scale their data. The user is also allowed to change the default values of the parameters of each prior.

Usage

```
VBsparsePCA(
  dat,
  r,
  lambda = 1,
  slab.prior = "MVN",
  max.iter = 100,
  eps = 0.001,
  jointly.row.sparse = T,
  center.scale = F,
  sig2.true = NA,
  threshold = 0.5,
  theta.int = NA,
  theta.var.int = NA,
  kappa.para1 = NA,
  kappa.para2 = NA,
  sigma.a = NA,
  sigma.b = NA
)
```

Arguments

dat	Data an $n * p$ matrix.
r	Rank.
lambda	Tuning parameter for the density g.
slab.prior	The density g, the default is "MVN", the multivariate normal distribution. Another choice is "Laplace".
max.iter	The maximum number of iterations for running the algorithm.
eps	The convergence threshold; the default is 10^{-4}.
jointly.row.sparse	The default is true, which means that the jointly row sparsity assumption is used; one could not use this assumptio by changing it to false.

center.scale	The default if false. If true, then the input date will be centered and scaled.
sig2.true	The default is false, $sigma^2$ will be estimated; if sig2 is known and its value is given, then $sigma^2$ will not be estimated.
threshold	The threshold to determine whether γ_j is 0 or 1; the default value is 0.5.
theta.int	The initial value of theta mean; if not provided, the algorithm will estimate it using PCA.
theta.var.int	The initial value of theta.var; if not provided, the algorithm will set it to be 1e-3*diag(r).
kappa.para1	The value of α_1 of $\pi(\kappa)$; default is 1.
kappa.para2	The value of α_2 of $\pi(\kappa)$; default is $p+1$.
sigma.a	The value of σ_a of $\pi(\sigma^2)$; default is 1.
sigma.b	The value of σ_b of $\pi(\sigma^2)$; default is 2.

Details

The model is

$$X_i = \theta w_i + \sigma\epsilon_i, \epsilon \sim N(0, I_p)$$

The spike and slab prior is given by

$$\pi(\theta, \boldsymbol{\gamma}|\lambda_1, r) \propto \prod_{j=1}^{p} \left(\gamma_j \int_{A \in V_{r,r}} g(\theta_j|\lambda_1, A, r)\pi(A)dA + (1-\gamma_j)\delta_0(\theta_j) \right)$$

$$g(\theta_j|\lambda_1, A, r) = C(\lambda_1)^r \exp(-\lambda_1 \|\beta_j\|_q^m)$$

$$\gamma_j|\kappa \sim Bernoulli(\kappa)$$

$$\kappa \sim Beta(\alpha_1, \alpha_2)$$

$$\sigma^2 \sim InvGamma(\sigma_a, \sigma_b)$$

where $V_{r,r} = \{A \in R^{r\times r} : A'A = I_r\}$ and δ_0 is the Dirac measure at zero. The density g can be chosen to be the product of independent Laplace distribution (i.e., $q = 1, m = 1$) or the multivariate normal distribution (i.e., $q = 2, m = 2$).

Value

iter	The number of iterations to reach convergence.
selection	A vector (if $r = 1$ or with the jointly row-sparsity assumption) or a matrix (if otherwise) containing the estimated value for $\boldsymbol{\gamma}$.
loadings	The loadings matrix.
uncertainty	The covariance of each non-zero rows in the loadings matrix.
scores	Score functions for the r principal components.
sig2	Variance of the noise.
obj.fn	A vector contains the value of the objective function of each iteration. It can be used to check whether the algorithm converges

References

Ning, B. (2021). Spike and slab Bayesian sparse principal component analysis. Unpublished manuscript.

Examples

```
## Not run: #In this example, the first 20 rows in the loadings matrix are nonzero, the rank is 2
set.seed(2021)
library(MASS)
library(pracma)
n <- 200
p <- 1000
s <- 20
r <- 2
sig2 <- 0.1
# generate eigenvectors
U.s <- randortho(s, type = c("orthonormal"))
if (r == 1) {
  U <- rep(0, p)
  U[1:s] <- as.vector(U.s[, 1:r])
} else {
  U <- matrix(0, p, r)
  U[1:s, ] <- U.s[, 1:r]
}
s.star <- rep(0, p)
s.star[1:s] <- 1
eigenvalue <- seq(20, 10, length.out = r)
# generate Sigma
if (r == 1) {
  theta.true <- U * sqrt(eigenvalue)
  Sigma <- tcrossprod(theta.true) + sig2*diag(p)
} else {
  theta.true <- U %*% sqrt(diag(eigenvalue))
  Sigma <- tcrossprod(theta.true) + sig2 * diag(p)
}
# generate n*p dataset
X <- t(mvrnorm(n, mu = rep(0, p), Sigma = Sigma))
result <- VBsparsePCA(dat = t(X), r = 2, jointly.row.sparse = T, center.scale = F)
loadings <- result$loadings
scores <- result$scores

## End(Not run)
```

Index