Package ‘mimiSBM’

January 9, 2024

Title Mixture of Multilayer Integrator Stochastic Block Models

Version 0.0.1.3

Description Our approach uses a mixture of multilayer stochastic block models to group co-membership matrices with similar information into components and to partition observations into different clusters. See De Santiago (2023, ISBN: 978-2-87587-088-9).

License GPL-3

Encoding UTF-8

RoxygenNote 7.2.3

Suggests knitr, rmarkdown

VignetteBuilder knitr

Imports blockmodels, stats

NeedsCompilation no

Author Kylliann De Santiago [aut, cre],
Marie Szafranski [aut],
Christophe Ambroise [aut]

Maintainer Kylliann De Santiago <kylliann.desantiago@univ-evry.fr>

Repository CRAN

Date/Publication 2024-01-09 08:40:15 UTC

R topics documented:

`BayesianMixture_SBM_model` *mimiSBM model for fixed K and Q*

Description

mimiSBM model for fixed K and Q

Usage

```
BayesianMixture_SBM_model(
  A,
  K,
  Q,
  beta_0 = rep(1/2, K),
  theta_0 = rep(1/2, Q),
  eta_0 = array(rep(1/2, K * K * Q), c(K, K, Q)),
  xi_0 = array(rep(1/2, K * K * Q), c(K, K, Q)),
  tol = 0.001,
  iter_max = 10,
  n_init = 1,
  alternate = TRUE,
  Verbose = TRUE,
  eps_conv = 1e-04,
  type_init = "SBM",
  nbCores = 2
)
```

Arguments

`A`	an array of dim=c(N,N,V)
`K`	number of clusters
`Q`	number of components
`beta_0`	hyperparameters for beta
`theta_0`	hyperparameters for theta
`eta_0`	hyperparameters for eta
`xi_0`	hyperparameters for xi
`tol`	convergence parameter on ELBO
`iter_max`	maximal number of iteration of mimiSBM
`n_init`	number of initialization of the mimi algorithm.
`alternate`	boolean indicated if we put an M-step after each part of the E-step, after u optimization and after tau optimization. If not, we optimize u and tau and after the M-step is made.
`Verbose`	boolean for information on model fitting
`eps_conv`	parameter of convergence for tau.
`type_init`	select the type of initialization type_init=c("SBM","Kmeans","random")
`nbCores`	the number of cores used to parallelize the calculations

See the vignette for more details.

Value

model with estimation of coefficients.

`CEM` *Clustering Matrix : One hot encoding*

Description

Clustering Matrix : One hot encoding

Usage

```
CEM(Z)
```

Arguments

`Z`	a matrix N x K, with probabilities to belong of a cluster in rows for each observation.

Value

Z a matrix N x K One-Hot-Encoded by rows, where K is the number of clusters.

Examples

```
Z <- matrix(rnorm(12),3,4)
Z_cem <- CEM(Z)
print(Z_cem)
```

---

diag_nulle *Diagonal coefficient to 0 on each slice given the 3rd dimension.*

Description

Diagonal coefficient to 0 on each slice given the 3rd dimension.

Usage

```
diag_nulle(A)
```

Arguments

A	a array of dimension dim=c(N,N,V)

Value

A with 0 on each diagonal given the 3rd dimension.

---

fit_SBM_per_layer *SBM on each layer*

Description

SBM on each layer

Usage

```
fit_SBM_per_layer(A, silent = FALSE, ncores = 2)
```

Arguments

A	an array of dim=c(N,N,V)
silent	Boolean for verbose
ncores	the number of cores used to parallelize the calculations of the various SBMs

Value

a list containing the parameters of each SBM applied to each view

---

`fit_SBM_per_layer_parallel` *SBM on each layer - parallelized*

Description

SBM on each layer - parallelized

Usage

```
fit_SBM_per_layer_parallel(A, nbCores = 2)
```

Arguments

`A`	an array of dim=c(N,N,V)
`nbCores`	the number of cores used to parallelize the calculations of the various SBMs

Value

a list containing the parameters of each SBM applied to each view

---

`initialisation_params_bayesian` *Initialization of mimiSBM parameters*

Description

Initialization of mimiSBM parameters

Usage

```
initialisation_params_bayesian(
  A,
  K,
  Q,
  beta_0 = rep(1/2, K),
  theta_0 = rep(1/2, Q),
  eta_0 = array(rep(1/2, K * K * Q), c(K, K, Q)),
  xi_0 = array(rep(1/2, K * K * Q), c(K, K, Q)),
  type_init = "SBM",
  nbCores = 2
)
```

Arguments

`A`	an array of dim=c(N,N,V)
`K`	Number of clusters
`Q`	Number of components
`beta_0`	hyperparameters for beta
`theta_0`	hyperparameters for theta
`eta_0`	hyperparameters for eta
`xi_0`	hyperparameters for xi
`type_init`	select the type of initialization type_init=c("SBM","Kmeans","random")
`nbCores`	the number of cores used to parallelize the calculations of the various SBMs

Value

a list `params` updated

`lab_switching` *Label Switching*

Description

This function can be used to perturb a clustering vector in order to randomly associate certain individuals with another cluster.

Usage

```
lab_switching(Z, p_out = 0.1)
```

Arguments

`Z`	a clustering vector
`p_out`	a probability of perturbation for the clustering

Value

a perturbed clustering vector

Examples

```
Z <- sample(1:4,100,replace=TRUE)
p = 0.1
Z_pert <- lab_switching(Z,p)
table("Initial clustering" = Z,"Perturbed clustering" = Z_pert)
```

log_Softmax *log softmax of matrices (by row)*

Description

log softmax of matrices (by row)

Usage

```
log_Softmax(log_X)
```

Arguments

`log_X`	a matrix of log(X)

Value

X with log_softmax function applied on each row

Examples

```
set.seed(42)
X <- matrix(rnorm(15,mean=5),5,3)
log_X <- log(X)
X_softmax <-  log_Softmax(X)
```

Loss_BayesianMSBM *mimiSBM Evidence Lower BOund*

Description

mimiSBM Evidence Lower BOund

Usage

```
Loss_BayesianMSBM(params)
```

Arguments

`params`	a list of parameters of the model

Value

computation of the mimiSBM ELBO

Mat_lien_alpha *Create probality-component list for clustering per view component.*

Description

Create probality-component list for clustering per view component.

Usage

```
Mat_lien_alpha(clusters, K_barre, K)
```

Arguments

clusters	list of link between final clustering and clustering per view component.
K_barre	Number of clusters in the final clustering
K	Vector of size Q, indicate the number of clusters in each component.

Value

alpha : probality-component list for clustering per view component.

mimiSBM *Mixture of Multilayer Integrator SBM (mimiSBM)*

Description

Model that allows both clustering of individuals and grouping of views by component. This bayesian model estimates the probability of individuals belonging to each cluster (cluster crossing all views) and the membership component for all views. In addition, the connectivity tensor between classes, conditional on the components, is also estimated.

Usage

```
mimiSBM(
  A,
  Kset,
  Qset,
  beta_0 = 1/2,
  theta_0 = 1/2,
  eta_0 = 1/2,
  xi_0 = 1/2,
  criterion = "ILVB",
  tol = 0.001,
  iter_max = 10,
  n_init = 1,
  alternate = FALSE,
  Verbose = FALSE,
  eps_conv = 1e-04,
  type_init = "SBM"
)
```

Arguments

`A`	an array of dim=c(N,N,V)
`Kset`	Set of number of clusters
`Qset`	Set of number of components
`beta_0`	hyperparameters for beta
`theta_0`	hyperparameters for theta
`eta_0`	hyperparameters for eta
`xi_0`	hyperparameters for xi
`criterion`	model selection criterion, criterion=c("ILVB","ICL_approx","ICL_variationnel","ICL_exact")
`tol`	convergence parameter on ELBO
`iter_max`	maximal number of iteration of mimiSBM
`n_init`	number of initialization of the mimi algorithm.
`alternate`	boolean indicated if we put an M-step after each part of the E-step, after u optimization and after tau optimization. If not, we optimize u and tau and after the M-step is made.
`Verbose`	boolean for information on model fitting
`eps_conv`	parameter of convergence for tau.
`type_init`	select the type of initialization type_init=c("SBM","Kmeans","random")

Value

The best model, conditionnally to the criterion, and its parameters.

Examples

```
set.seed(42)
K = c(2,3); pi_k = rep(1/4,4) ; rho = rep(1/2,2)
res <- rSMB_partition(N = 50,V = 5,K = K ,pi_k = pi_k ,rho = rho,p_switch = 0.1)
A = res$simulation$A ; Kset = 4 ; Qset = 2
model <- mimiSBM(A,Kset,Qset,n_init = 1, Verbose=FALSE)
```

`multinomial_lbeta_function` *Calculation of Log multinomial Beta value.*

Description

Calculation of Log multinomial Beta value.

Usage

```
multinomial_lbeta_function(x)
```

Arguments

`x`	a vector

Value

sum(lgamma(x[j])) - lgamma(sum(x))

`one_hot_errormachine` *One Hot Encoding with Error machine*

Description

One Hot Encoding with Error machine

Usage

```
one_hot_errormachine(Z, size = NULL)
```

Arguments

`Z`	a vector of size N, where Z[i] value indicate the cluster membership of observation i.
`size`	optional parameter, indicating the number of classes (avoid some empty class problems).

Value

Z a matrix N x K One-Hot-Encoded by rows, where K is the number of clusters.

Examples

```
Z <- sample(1:4,10,replace=TRUE)
Z_OHE <- one_hot_errormachine(Z)
print(Z_OHE)
```

`partition_K_barre` *Create a link between final clustering and clustering per view component.*

Description

Create a link between final clustering and clustering per view component.

Usage

```
partition_K_barre(K_barre, K)
```

Arguments

`K_barre`	Number of clusters in the final clustering
`K`	Vector of size Q, indicate the number of clusters in each component. K[q] <= K_barre for all q

Value

cluster : a list of link between final clustering and clustering per view component.

`plot_adjacency` *Plot adjacency matrices*

Description

A function to plot each adjacency matrices defined by the thrid dimension of an array, and plot the sum of all theses matrices.

Usage

```
plot_adjacency(A)
```

Arguments

`A`	an array with dim=c(N,N,V).

Value

None

rMSBM *Simulate data from the mimiSBM generative model.*

Description

Simulate data from the mimiSBM generative model.

Usage

```
rMSBM(N, V, alpha_klq, pi_k, rho, sorted = TRUE, p_switch = NULL)
```

Arguments

N	Number of individuals.
V	Number of views.
alpha_klq	array of component-connection probability (K,K,Q).
pi_k	Vector of proportions of individuals across clusters.
rho	Vector of proportion of views across components.
sorted	Boolean for simulation reordering (clusters and components membership).
p_switch	probability of label-switching, if NULL no perturbation between true clustering and the connectivity of individuals.

Value

list with the parameters of the simulation ($params), and the simulations ($simulation).

rSMB_partition *Simulation of mixture multilayer Stochastick block model*

Description

This simulation process assumes that we have partial information on the clustering within each view component, and that the final clustering of individuals depends on a combination of the clustering on each of the views. In addition, we take into account possible label-switching: we consider that an individual belongs with a certain probability to the wrong class, thus disturbing the adjacency matrices and making the simulation more real and complex.

Usage

```
rSMB_partition(N, V, K, pi_k, rho, sorted = TRUE, p_switch = NULL)
```

Arguments

`N`	Number of observations
`V`	Number of views
`K`	Vector of size Q, indicate the number of clusters in each component.
`pi_k`	Vector of proportions of observations across clusters.
`rho`	Vector of proportion of views across components.
`sorted`	Boolean for simulation reordering (clusters and components membership).
`p_switch`	probability of label-switching, if NULL no perturbation between true clustering and the connectivity of individuals.

Details

See the vignette for more information.

Value

list with the parameters of the simulation ($params), and the simulations ($simulation).

`sort_Z` *Sort the clustering matrix*

Description

Sort the clustering matrix

Usage

```
sort_Z(Z)
```

Arguments

`Z`	a matrix N x K, with probabilities to belong of a cluster in rows for each observation.

Value

a sorted matrix

transpo *Transposition of an array*

Description

Transposition of an array

Usage

```
transpo(A)
```

Arguments

`A`	a array of dim= c(.,.,V)

Value

A_transposed, the transposed array according the third dimension

trig_sup *Upper triangular Matrix/Array*

Description

Upper triangular Matrix/Array

Usage

```
trig_sup(A, transp = FALSE, diag = TRUE)
```

Arguments

`A`	a array or a squared matrix
`transp`	boolean, indicate if we need a transposition or not.
`diag`	boolean, if True, diagonal is not used.

Value

a array or a squared matrix, with only upper-triangular coefficients with non-zero values

---

`update_beta_bayesian` *Update of bayesian parameter beta*

Description

Update of bayesian parameter beta

Usage

```
update_beta_bayesian(params)
```

Arguments

`params`	list of parameters of the model

Value

params with beta updated

`update_eta_bayesian` *Update of bayesian parameter eta*

Description

Update of bayesian parameter eta

Usage

```
update_eta_bayesian(A, params)
```

Arguments

`A`	an array of dim=c(N,N,V)
`params`	list of parameters of the model

Value

params with eta updated

`update_tau_bayesian` *Update of bayesian parameter tau*

Description

Update of bayesian parameter tau

Usage

```
update_tau_bayesian(A, params, eps_conv = 1e-04)
```

Arguments

`A`	an array of dim=c(N,N,V)
`params`	list of parameters of the model
`eps_conv`	parameter of convergence.

Value

params with tau updated

`update_theta_bayesian` *Update of bayesian parameter theta*

Description

Update of bayesian parameter theta

Usage

```
update_theta_bayesian(params)
```

Arguments

`params`	list of parameters of the model

Value

params with theta updated

`update_u_bayesian` *Update of bayesian parameter u*

Description

Update of bayesian parameter u

Usage

```
update_u_bayesian(A, params)
```

Arguments

`A`	an array of dim=c(N,N,V)
`params`	list of parameters of the model

Value

params with u updated

`update_xi_bayesian` *Update of bayesian parameter xi*

Description

Update of bayesian parameter xi

Usage

```
update_xi_bayesian(A, params)
```

Arguments

`A`	an array of dim=c(N,N,V)
`params`	list of parameters of the model

Value

params with xi updated

`VBEM_step` *Variational Bayes Expectation Maximization*

Description

Variational Bayes Expectation Maximization

Usage

```
VBEM_step(A, params, alternate = TRUE, eps_conv = 0.001)
```

Arguments

`A`	an array of dim=c(N,N,V)
`params`	list of parameters of the model
`alternate`	boolean indicated if we put an M-step after each part of the E-step, after u optimization and after tau optimization. If not, we optimize u and tau and after the M-step is made.
`eps_conv`	parameter of convergence for tau.

Value

params with updated parameters.

Index