Package 'dTBM'

May 8, 2026

Title Multi-Way Spherical Clustering via Degree-Corrected Tensor Block Models

Version 3.0

Date 2023-06-16

Maintainer Jiaxin Hu <jhu267@wisc.edu>

Description Implement weighted higher-order initialization and angle-based iteration for multi-way spherical clustering under degree-corrected tensor block model. See reference Jiaxin Hu and Miaoyan Wang (2023) <doi:10.1109/TIT.2023.3239521>.

Imports WeightedCluster, EnvStats, methods

License GPL (>= 2)

Encoding UTF-8

LazyData true

NeedsCompilation no

Author Jiaxin Hu [aut, cre, cph],
Miaoyan Wang [aut, cph]

RoxygenNote 7.2.3

Depends R (>= 3.5.0)

Repository CRAN

Date/Publication 2023-06-18 22:30:06 UTC

Contents

`angle_iteration` *Angle-based iteration*

Description

Angle-based iteration for multiway spherical clustering under degree-corrected tensor block model. This function takes the tensor/matrix observation, initial clustering assignment, and a logic variable indicating the symmetry as input. Output is the refined clustering assignment.

Usage

```
angle_iteration(Y, z0, max_iter, alpha1 = 0.01, asymm)
```

Arguments

`Y`	array/matrix, order-3 tensor/matrix observation
`z0`	a list of vectors, initial clustering assignment; see "details"
`max_iter`	integer, max number of iterations if update does not converge
`alpha1`	number, substitution of degenerate core tensor; see "details"
`asymm`	logic variable, if "TRUE", assume the clustering assignment differs in different modes; if "FALSE", assume all the modes share the same clustering assignment

Details

`z0` should be a length 2 list for matrix and length 3 list for tensor observation; observations with non-identical dimension on each mode are only applicable with `asymm = T`;

When the estimated core tensor has a degenerate slice, i.e., a slice with all zero elements, randomly pick an entry in the degenerate slice with value `alpha1`.

Value

a list containing the following:

`z` a list of vectors recording the estimated clustering assignment

`s_deg` logic variable, if "TRUE", degenerate estimated core tensor/matrix occurs during the iteration; if "FALSE", otherwise

Examples

```
test_data = sim_dTBM(seed = 1, imat = FALSE, asymm = FALSE, p = c(50,50,50), r = c(3,3,3),
                     core_control = "control", s_min = 0.05, s_max = 1,
                     dist = "normal", sigma = 0.5,
                     theta_dist = "pareto", alpha = 4, beta = 3/4)

initialization <- wkmeans(test_data$Y, r = c(3,3,3), asymm = FALSE)

iteration <- angle_iteration(test_data$Y, initialization$z0, max_iter = 20, asymm = FALSE)
```

---

as.tensor *Tensor Conversion*

Description

Create a Tensor-class object from an array, matrix, or vector.

Usage

```
as.tensor(x, drop = FALSE)
```

Arguments

x	an instance of array, matrix, or vector
drop	whether or not modes of 1 should be dropped

Value

a Tensor-class object

Examples

```
#From vector
vec <- runif(100); vecT <- as.tensor(vec); vecT
#From matrix
mat <- matrix(runif(1000),nrow=100,ncol=10)
matT <- as.tensor(mat); matT
#From array
indices <- c(10,20,30,40)
arr <- array(runif(prod(indices)), dim = indices)
arrT <- as.tensor(arr); arrT
```

dim-methods *Mode Getter for Tensor*

Description

Return the vector of modes from a tensor

Usage

```
## S4 method for signature 'Tensor'
dim(x)
```

Arguments

x	the Tensor instance

Details

`dim(x)`

Value

an integer vector of the modes associated with `x`

Examples

```
tnsr <- rand_tensor()
dim(tnsr)
```

dtbm *Multiway spherical clustering for degree-corrected tensor block model*

Description

Multiway spherical clustering for degree-corrected tensor block model including weighted higher-order initialization and angle-based iteration. Main function in the package. This function takes the tensor/matrix observation, the cluster number, and a logic variable indicating the symmetry as input. Output contains initial and refined clustering assignment.

Usage

```
dtbm(Y, r, max_iter, alpha1 = 0.01, asymm)
```

Arguments

Y	array/matrix, order-3 tensor/matrix observation
r	vector, the cluster number on each mode; see "details"
max_iter	integer, max number of iterations if update does not converge
alpha1	number, substitution of degenerate core tensor; see "details"
asymm	logic variable, if "TRUE", assume the clustering assignment differs in different modes; if "FALSE", assume all the modes share the same clustering assignment

Details

r should be a length 2 vector for matrix and length 3 vector for tensor observation;

all the elements in r should be integer larger than 1;

symmetric case only allow r with the same cluster number on each mode;

observations with non-identical dimension on each mode are only applicable with asymm = T.

When the estimated core tensor has a degenerate slice during iteration, i.e., a slice with all zero elements, randomly pick an entry in the degenerate slice with value alpha1.

Value

a list containing the following:

z a list of vectors recording the refined clustering assignment with initialization z0

s_deg logic variable, if "TRUE", degenerate estimated core tensor/matrix occurs during the iteration; if "FALSE", otherwise

z0 a list of vectors recording the initial clustering assignment

s0 a list of vectors recording the index of degenerate entities with random clustering assignment in initialization

Examples

```
test_data = sim_dTBM(seed = 1, imat = FALSE, asymm = FALSE, p = c(50,50,50), r = c(3,3,3),
                     core_control = "control", s_min = 0.05, s_max = 1,
                     dist = "normal", sigma = 0.5,
                     theta_dist = "pareto", alpha = 4, beta = 3/4)

result = dtbm(test_data$Y, r = c(3,3,3), max_iter = 20, asymm = FALSE)
```

`fold` *General Folding of Matrix*

Description

General folding of a matrix into a Tensor. This is designed to be the inverse function to `unfold-methods`, with the same ordering of the indices. This amounts to following: if we were to unfold a Tensor using a set of `row_idx` and `col_idx`, then we can fold the resulting matrix back into the original Tensor using the same `row_idx` and `col_idx`.

Usage

```
fold(mat, row_idx = NULL, col_idx = NULL, modes = NULL)
```

Arguments

`mat`	matrix to be folded into a Tensor
`row_idx`	the indices of the modes that are mapped onto the row space
`col_idx`	the indices of the modes that are mapped onto the column space
`modes`	the modes of the output Tensor

Details

This function uses `aperm` as the primary workhorse.

Value

Tensor object with modes given by `modes`

References

T. Kolda, B. Bader, "Tensor decomposition and applications". SIAM Applied Mathematics and Applications 2009, Vol. 51, No. 3 (September 2009), pp. 455-500. URL: https://www.jstor.org/stable/25662308.

See Also

`unfold-methods`

Examples

```
tnsr <- new('Tensor',3L,c(3L,4L,5L),data=runif(60))
matT3<-unfold(tnsr,row_idx=2,col_idx=c(3,1))
identical(fold(matT3,row_idx=2,col_idx=c(3,1),modes=c(3,4,5)),tnsr)
```

HCP *HCP data*

Description

The HCP data is obtained by preprocessing the data from Human Connectome Project (HCP); see https://wiki.humanconnectome.org/display/PublicData/.

Usage

```
data(HCP)
```

Format

A list. Includes a 68-68-136 binary array named "tensor" and a 136-573 data frame named "attr".

Details

The array "tensor" is a $68 \times 68 \times 136$ binary tensor consisting of structural connectivity patterns among 68 brain regions for 136 individuals. All the individual images were preprocessed following a standard pipeline (Zhang et al., 2018), and the brain was parcellated to 68 regions-of-interest following the Desikan atlas (Desikan et al., 2006). The tensor entries encode the presence or absence of fiber connections between those 68 brain regions for each of the 136 individuals.

The data frame "attr" is a 136×573 matrix consisting of 573 personal features for 136 individuals. The full list of covariates can be found at: https://wiki.humanconnectome.org/display/PublicData/

kronecker_list *List Kronecker Product*

Description

Returns the Kronecker product from a list of matrices or vectors. Commonly used for n-mode products and various Tensor decompositions.

Usage

```
kronecker_list(L)
```

Arguments

L	list of matrices or vectors

Value

matrix that is the Kronecker product

Examples

```
smalllizt <- list('mat1' = matrix(runif(12),ncol=4),
'mat2' = matrix(runif(12),ncol=4),
'mat3' = matrix(runif(12),ncol=4))
dim(kronecker_list(smalllizt))
```

`peru` *Peru Legislation data*

Description

The Peru Legislation data is obtained by preprocessing the original data in Lee et al., 2017.

Usage

```
data(peru)
```

Format

A list. Includes a 116-2 data frame named "attr_data", a 5844-7 data frame named "laws_data", and a 116-116-116 binary array named "network_data".

Details

The data frame "attr_data" is a 116 x 2 matrix consisting the name and party affiliation of 116 legislators in the top five parties. The legislators IDs are recorded in the row names of the matrix.

The data frame "laws_data" is a 5844 x 7 matrix recording the co-sponsorship of 116 legislators of 802 bills during the first half of 2006-2007 year.

The array "network_data" is a 116 x 116 x 116 binary tensor recording the presence of order-3 co-sponsorship among legislators based on "laws_data". Specfically, the entry (i,j,k) is 1 if the legislators (i,j,k) have sponsored the same bill, and the entry (i,j,k) is 0 otherwise.

`rand_tensor` *Tensor with Random Entries*

Description

Generate a Tensor with specified modes with iid normal(0,1) entries.

Usage

```
rand_tensor(modes = c(3, 4, 5), drop = FALSE)
```

Arguments

modes	the modes of the output Tensor
drop	whether or not modes equal to 1 should be dropped

Value

a Tensor object with modes given by modes

Note

Default rand_tensor() generates a 3-Tensor with modes c(3,4,5).

Examples

```
rand_tensor()
rand_tensor(c(4,4,4))
rand_tensor(c(10,2,1),TRUE)
```

---

select_r *Cluster number selection*

Description

Estimate the cluster number in the degree-corrected tensor block model based on BIC criterion. The choice of BIC aims to balance between the goodness-of-fit for the data and the degree of freedom in the population model. This function is restricted for the Gaussian observation.

Usage

```
select_r(Y, r_range, asymm = FALSE)
```

Arguments

Y	array/matrix, order-3 Gaussian tensor/matrix observation
r_range	matrix, candidates for the cluster number on each row; see "details"
asymm	logic variable, if "TRUE", clustering assignment differs in different modes; if "FALSE", all the modes share the same clustering assignment

Details

r_range should be a two-column matrix for matrix and three-column matrix for tensor observation;

all the elements in r_range should be integer larger than 1;

symmetric case only allow candidates with the same cluster number on each mode;

observations with non-identical dimension on each mode are only applicable with asymm = TRUE.

Value

a list containing the following:

r vector, the cluster number among the candidates with minimal BIC value

bic vector, the BIC value for each candidiate

Examples

```
test_data = sim_dTBM(seed = 1, imat = FALSE, asymm = FALSE, p = c(50,50,50), r = c(3,3,3),
                     core_control = "control", s_min = 0.05, s_max = 1,
                     dist = "normal", sigma = 0.5,
                     theta_dist = "pareto", alpha = 4, beta = 3/4)

r_range = rbind(c(2,2,2), c(3,3,3),c(4,4,4),c(5,5,5))
selection <- select_r(test_data$Y, r_range, asymm = FALSE)
```

---

sim_dTBM *Simulation of degree-corrected tensor block models*

Description

Generate order-3 tensor/matrix observations with degree heterogeneity under degree-corrected tensor block models.

Usage

```
sim_dTBM(
  seed = NA,
  imat = FALSE,
  asymm = FALSE,
  p,
  r,
  core_control = c("random", "control"),
  delta = NULL,
  s_min = NULL,
  s_max = NULL,
  dist = c("normal", "binary"),
  sigma = 1,
  theta_dist = c("abs_normal", "pareto", "non"),
  alpha = NULL,
  beta = NULL
)
```

Arguments

seed	number, random seed for generating data
imat	logic variable, if "TRUE", generate matrix data; if "FALSE", generate order-3 tensor data

asymm	logic variable, if "TRUE", clustering assignment differs in different modes; if "FALSE", all the modes share the same clustering assignment
p	vector, dimension of the tensor/matrix observation
r	vector, cluster number on each mode
core_control	character, the way to control the generation of core tensor/matrix; see "details"
delta	number, Frobenius norm of the slices in core tensor if core_control = "control"
s_min	number, value of off-diagonal elements in original core tensor/matrix if core_control = "control"
s_max	number, value of diagonal elements in original core tensor/matrix if core_control = "control"
dist	character, distribution of tensor/matrix observation; see "details"
sigma	number, standard deviation of Gaussian noise if dist = "normal"
theta_dist	character, distribution of degree heterogeneity; see "details"
alpha	number, shape parameter in pareto distribution if theta_dist = "pareto"
beta	number, scale parameter in pareto distribution if theta_dist = "pareto"

Details

The general tensor observation is generated as

Y = S x1 Theta1 M1 x2 Theta2 M2 x3 Theta3 M3 + E,

where S is the core tensor, Thetak is a diagonal matrix with elements in the k-th vector of theta, Mk is the membership matrix based on the clustering assignment in the k-th vector of z with r[k] clusters, E is the mean-zero noise tensor, and xk refers to the matrix-by-tensor product on the k-th mode, for k = 1,2,3.

If imat = TRUE, Y,S,E degenerate to matrix and Y = Theta1 M1 S M2^T Theta2^T + E.

If asymm = FALSE, Thetak = Theta and Mk = M for all k = 1,2,3.

core_control specifies the way to generate S:

If core_control = "control", first generate S as a diagonal tensor/matrix with diagonal elements s_max and off-diagonal elements s_min; then scale the original core such that Frobenius norm of the slices equal to delta, i.e, delta = sqrt(sum(S[1,,]^2)) or delta = sqrt(sum(S[1,]^2)); ignore the scaling if delta = NULL; option "control" is only applicable for symmetric case asymm = FALSE.

If core_control = "random", generate S with random entries following uniform distribution U(0,1).

dist specifies the distribution of E: "normal" for Gaussian and "binary" for Bernoulli distribution; sigma specifices the standard deviation if dist = "normal".

theta_dist firstly specifies the distribution of theta: "non" for constant 1, "abs_normal" for absoulte normal distribution, "pareto" for pareto distribution; alpha, beta specify the shape and scale parameter if theta_dist = "pareto"; then scale theta to have mean equal to one in each cluster.

Value

a list containing the following:

`Y` array (if `imat = FALSE`)/matrix (if `imat = TRUE`), simulated tensor/matrix observations with dimension `p`

`X` array (if `imat = FALSE`)/matrix (if `imat = TRUE`), mean tensor/matrix of the observation, i.e., the expectation of `Y`

`S` array (if `imat = FALSE`)/matrix (if `imat = TRUE`), core tensor/matrix recording the block effects with dimension `r`

`theta` a list of vectors, degree heterogeneity on each mode

`z` a list of vectors, clustering assignment on each mode

Examples

```
test_data = sim_dTBM(seed = 1, imat = FALSE, asymm = FALSE, p = c(50,50,50), r = c(3,3,3),
                     core_control = "control", s_min = 0.05, s_max = 1,
                     dist = "normal", sigma = 0.5,
                     theta_dist = "pareto", alpha = 4, beta = 3/4)
```

Tensor-class *S4 Class for a Tensor*

Description

An S4 class for a tensor with arbitrary number of modes. The Tensor class extends the base "array" class to include additional tensor manipulation (folding, unfolding, reshaping, subsetting) as well as a formal class definition that enables more explicit tensor algebra.

Slots

num_modes number of modes (integer)

modes vector of modes (integer), aka sizes/extents/dimensions

data actual data of the tensor, which can be 'array' or 'vector'

Note

All of the decompositions and regression models in this package require a Tensor input.

Author(s)

James Li <jamesyili@gmail.com>

References

James Li, Jacob Bien, Martin T. Wells (2018). rTensor: An R Package for Multidimensional Array (Tensor) Unfolding, Multiplication, and Decomposition. Journal of Statistical Software, Vol. 87, No. 10, 1-31. URL: http://www.jstatsoft.org/v087/i10/.

See Also

as.tensor

---

ttl *Tensor Times List*

Description

Contracted (m-Mode) product between a Tensor of arbitrary number of modes and a list of matrices. The result is folded back into Tensor.

Usage

```
ttl(tnsr, list_mat, ms = NULL)
```

Arguments

tnsr	Tensor object with K modes
list_mat	a list of matrices
ms	a vector of modes to contract on (order should match the order of list_mat)

Details

Performs ttm repeated for a single Tensor and a list of matrices on multiple modes. For instance, suppose we want to do multiply a Tensor object tnsr with three matrices mat1, mat2, mat3 on modes 1, 2, and 3. We could do ttm(ttm(ttm(tnsr,mat1,1),mat2,2),3), or we could do ttl(tnsr,list(mat1,mat2,mat3),c(1,2,3)). The order of the matrices in the list should obviously match the order of the modes. This is a common operation for various Tensor decompositions such as CP and Tucker. For the math on the m-Mode Product, see Kolda and Bader (2009).

Value

Tensor object with K modes

Note

The returned Tensor does not drop any modes equal to 1.

References

T. Kolda, B. Bader, "Tensor decomposition and applications". SIAM Applied Mathematics and Applications 2009, Vol. 51, No. 3 (September 2009), pp. 455-500. URL: https://www.jstor.org/stable/25662308

See Also

ttm

Examples

```
tnsr <- new('Tensor',3L,c(3L,4L,5L),data=runif(60))
lizt <- list('mat1' = matrix(runif(30),ncol=3),
'mat2' = matrix(runif(40),ncol=4),
'mat3' = matrix(runif(50),ncol=5))
ttl(tnsr,lizt,ms=c(1,2,3))
```

---

ttm *Tensor Matrix Product (m-Mode Product)*

Description

Contracted (m-Mode) product between a Tensor of arbitrary number of modes and a matrix. The result is folded back into Tensor.

Usage

```
ttm(tnsr, mat, m = NULL)
```

Arguments

`tnsr`	Tensor object with K modes
`mat`	input matrix with same number columns as the `m`th mode of `tnsr`
`m`	the mode to contract on

Details

By definition, the number of columns in `mat` must match the `m`th mode of `tnsr`. For the math on the m-Mode Product, see Kolda and Bader (2009).

Value

a Tensor object with K modes

Note

The `m`th mode of `tnsr` must match the number of columns in `mat`. By default, the returned Tensor does not drop any modes equal to 1.

References

T. Kolda, B. Bader, "Tensor decomposition and applications". SIAM Applied Mathematics and Applications 2009, Vol. 51, No. 3 (September 2009), pp. 455-500. URL: https://www.jstor.org/stable/25662308

See Also

`ttl`

Examples

```
tnsr <- new('Tensor',3L,c(3L,4L,5L),data=runif(60))
mat <- matrix(runif(50),ncol=5)
ttm(tnsr,mat,m=3)
```

`unfold-methods` *Tensor Unfolding*

Description

Unfolds the tensor into a matrix, with the modes in `rs` onto the rows and modes in `cs` onto the columns. Note that `c(rs,cs)` must have the same elements (order doesn't matter) as `x@modes`. Within the rows and columns, the order of the unfolding is determined by the order of the modes. This convention is consistent with Kolda and Bader (2009).

Usage

```
unfold(tnsr, row_idx, col_idx)
```

Arguments

`tnsr`	the Tensor instance
`row_idx`	the indices of the modes to map onto the row space
`col_idx`	the indices of the modes to map onto the column space

Details

```
unfold(tnsr,row_idx=NULL,col_idx=NULL)
```

Value

matrix with `prod(row_idx)` rows and `prod(col_idx)` columns

References

T. Kolda, B. Bader, "Tensor decomposition and applications". SIAM Applied Mathematics and Applications 2009, Vol. 51, No. 3 (September 2009), pp. 455-500. URL: https://www.jstor.org/stable/25662308.

Examples

```
tnsr <- rand_tensor()
matT3<-unfold(tnsr,row_idx=2,col_idx=c(3,1))
```

wkmeans *Weighted higher-order initialization*

Description

Weighted higher-order initialization for multiway spherical clustering under degree-corrected tensor block model. This function takes the tensor/matrix observation, the cluster number, and a logic variable indicating the symmetry as input. Output is the estimated clustering assignment.

Usage

```
wkmeans(Y, r, asymm)
```

Arguments

Y	array/matrix, order-3 tensor/matrix observation
r	vector, the cluster number on each mode; see "details"
asymm	logic variable, if "TRUE", assume the clustering assignment differs in different modes; if "FALSE", assume all the modes share the same clustering assignment

Details

r should be a length 2 vector for matrix and length 3 vector for tensor observation;

all the elements in r should be integer larger than 1;

symmetric case only allow r with the same cluster number on each mode;

observations with non-identical dimension on each mode are only applicable with asymm = T.

Value

a list containing the following:

z0 a list of vectors recording the estimated clustering assignment

s0 a list of vectors recording the index of degenerate entities with random clustering assignment

Examples

```
test_data = sim_dTBM(seed = 1, imat = FALSE, asymm = FALSE, p = c(50,50,50), r = c(3,3,3),
                     core_control = "control", s_min = 0.05, s_max = 1,
                     dist = "normal", sigma = 0.5,
                     theta_dist = "pareto", alpha = 4, beta = 3/4)

initialization <- wkmeans(test_data$Y, r = c(3,3,3), asymm = FALSE)
```

Index