Package ‘tensorsparse’

March 22, 2020

Type Package
Title Multiway Clustering via Tensor Block Models
Version 2.0
Date 2020-03-20
Author Miaoyan Wang, Yuchen Zeng
Maintainer Yuchen Zeng <yzeng58@wisc.edu>
Imports fields, glmnet, rgl, reshape, mvtnorm, HDCI, parallel,
        RColorBrewer, viridis, methods, stats, utils
Suggests cluster
Description Implements the multiway sparse clustering approach of M. Wang and Y. Zeng, "Multi-
        way clustering via tensor block models". Advances in Neural Information Processing Sys-
License GPL (>= 2)
NeedsCompilation yes
Repository CRAN
Date/Publication 2020-03-22 07:10:02 UTC
RoxygenNote 7.0.0
Encoding UTF-8

R topics documented:

  choosekrl_bic .......................................................... 2
  chooseLambda ........................................................... 3
  get_data ................................................................. 3
  get_data4 .............................................................. 5
  tbm_clustering ......................................................... 6

Index 8
choosekrl_bic

Select the clustering size for sparse tensor clustering via BIC

Description

Select the clustering size for three-way clustering. The function searches over a range of clustering sizes and outputs the one that minimizes BIC. The clustering size \((d_1, d_2, d_3)\) is a length-3 vector consisting of the number of clusters in each mode.

Usage

```r
choosekrl_bic(
  x,
  k,
  r,
  l,
  lambda = 0,
  sim.times = 1,
  method = "L0",
  n.cores = NULL
)
```

Arguments

- **x**: a three-dimensional array
- **k**: a vector, the possible numbers of clusters at mode 1
- **r**: a vector, the possible numbers of clusters at mode 2
- **l**: a vector, the possible numbers of clusters at mode 3
- **lambda**: a numeric value, regularization coefficient
- **sim.times**: the number of simulation replicates when performing clustering
- **method**: two options: "L0", "L1". "L0" indicates L0 penalty, and "L1" indicates Lasso penalty
- **n.cores**: the number of cores in parallel implementation

Value

- **estimated_krl**: a 1*3 matrix consisting of the estimated clustering size
- **BIC**: a vector consisting of the BIC value for all combinations of clustering sizes
chooseLambda

Select the regularization coefficient for sparse tensor clustering via BIC

Description

Select the regularization coefficient for three-way clustering. The clustering size is assumed to be known. The function searches over a range of regularization sizes and outputs the one that minimizes the BIC.

Usage

chooseLambda(x, k, r, l, lambda = NULL, method = "L0")

Arguments

- x: a three-dimensional array
- k: an positive integer, the numbers of clusters at mode 1
- r: an positive integer, the numbers of clusters at mode 2
- l: an positive integer, the numbers of clusters at mode 3
- lambda: a vector of possible lambda, eg: lambda = c(0,50,100,200)
- method: two options: "L0", "L1". "L0" indicates L0 penalty, and "L1" indicates Lasso penalty

Value

- lambda: the lambda with lowest BIC
- BIC: the BIC for each lambda in the given range
- nonzeromus: the number of clusters with non-zero means

get_data

Generate a random order-3 tensor

Description

Generate an order-3 random tensor based on tensor block model.
Usage

get_data(
  n,
  p,
  q,
  k = NULL,
  r = NULL,
  l = NULL,
  error = 3,
  sort = TRUE,
  sparse.percent = 0,
  center = FALSE,
  seed = NULL,
  mumin = -3,
  mumax = 3
)

Arguments

n the dimension at mode 1
p the dimension at mode 2
q the dimension at mode 3
k an positive integer, the numbers of clusters at mode 1
r an positive integer, the numbers of clusters at mode 2
l an positive integer, the numbers of clusters at mode 3
error a positive numeric value, noise level
sort if TRUE, the tensor entries belonging to the same cluster would be assumed together
sparse.percent the proportion of zero entries based on the Gaussian tensor block model
center if True, the data tensor would be centered to zero-mean before clustering
seed a positive integer, used to specify the random seed
mumin a numeric value, the lower bound of the block mean
mumax a numeric value, the upper bound of the block mean

Value

a list
  x the tensor
  truthX the underlying signal tensor following block model
  truthCs true cluster label assignment at mode 1
  truthDs true cluster label assignment at mode 2
  truthEs true cluster label assignment at mode 3
  mus the block means
  binaryX the 0-1 tensor (0:the mean signal = 0; 1:the mean signal != 0)
get_data4

Examples

get_data(20,20,20,2,2,2)$x

get_data4

Generate a random order-4 tensor

Description

Generate a random order-4 tensor based on tensor block model.

Usage

get_data4(
  n,
  p,
  q,
  s,
  k = NULL,
  r = NULL,
  l = NULL,
  m = NULL,
  error = 3,
  sort = TRUE,
  sparse.percent = 0,
  center = FALSE,
  seed = NULL,
  mumin = -3,
  mumax = 3
)

Arguments

n the dimension at mode 1
p the dimension at mode 2
q the dimension at mode 3
s the dimension at mode 4
k an positive integer, the numbers of clusters at mode 1
r an positive integer, the numbers of clusters at mode 2
l an positive integer, the numbers of clusters at mode 3
m an positive integer, the numbers of clusters at mode 4
error a positive numeric value, noise level
sort if TRUE, the tensor entries belonging to the same cluster would be assumed together
sparse.percent the proportion of zero entries based on the Gaussian tensor block model
center if True, the data tensor would be centered to zero-mean before clustering
seed a positive integer, used to specify the random seed
mumin a numeric value, the lower bound of the block mean
mumax a numeric value, the upper bound of the block mean

Value

a list
x the tensor
truthX the underlying signal tensor following block model
truthCs true cluster label assignment at mode 1
truthDs true cluster label assignment at mode 2
truthEs true cluster label assignment at mode 3
truthFs true cluster label assignment at mode 4
mus the block means
binaryX the 0-1 tensor (0:the mean signal = 0; 1:the mean signal != 0)

---

**tbm_clustering**  
Perform tensor clustering via tensor block model (TBM)

**Description**

Perform tensor clustering via tensor block model (TBM) method.

**Usage**

```r
tbm_clustering(
  x,  
k,  
r,  
l,  
lambda = 0,  
max.iter = 1000,  
threshold = 1e-10,  
sim.times = 1,  
trace = FALSE,  
Cs.init = NULL,  
Ds.init = NULL,  
Es.init = NULL,  
method = “L0”  
)
```
Arguments

- **x**: an order-3 data tensor
- **k**: an positive integer, the numbers of clusters at mode 1
- **r**: an positive integer, the numbers of clusters at mode 2
- **l**: an positive integer, the numbers of clusters at mode 3
- **lambda**: a numeric value, regularization coefficient
- **max.iter**: a positive integer, the maximum numbers of iteration
- **threshold**: a positive small numeric value for convergence threshold
- **sim.times**: the number of simulation replicates when performing clustering
- **trace**: logic value, print result per each iteration if TRUE
- **Cs.init**: vector or NULL, initial cluster label assignment at mode 1
- **Ds.init**: vector or NULL, initial cluster label assignment at mode 2
- **Es.init**: vector or NULL, initial cluster label assignment at mode 3
- **method**: two options: "L0", "L1". "L0" indicates L0 penalty, and "L1" indicates Lasso penalty

Value

- a list
  - **judgeX**: estimated underlying signal tensor
  - **Cs**: clustering result at mode 1
  - **Ds**: clustering result at mode 2
  - **Es**: clustering result at mode 3
  - **mus**: estimated block means

Author(s)

Yuchen Zeng <yzeng58@wisc.edu>

References


Examples

```r
x = get_data(20,20,20,2,2,2)$x
tbm_clustering(x,2,2,2)
```
Index

choosekrl_bic, 2
chooseLambda, 3

get_data, 3
get_data4, 5

tbm_clustering, 6