Package 'iTensor'

April 28, 2023

Type Package

Title ICA-Based Matrix/Tensor Decomposition

Version 1.0.2

Description
Some functions for performing ICA, MICA, Group ICA, and Multilinear ICA are implemented. ICA, MICA/Group ICA, and Multilinear ICA extract statistically independent components from single matrix, multiple matrices, and single tensor, respectively.
For the details of these methods, see the reference section of GitHub README.md <https://github.com/rikenbit/iTensor>.

License MIT + file LICENSE

Encoding UTF-8

Depends R (>= 4.1.0)

URL https://github.com/rikenbit/iTensor

RoxygenNote 7.2.3

Suggests nnTensor, knitr, rmarkdown, testthat (>= 3.0.0)

Config/testthat/edition 3

Imports MASS, methods, graphics, utils, stats, rTensor, jointDiag, mgcv, einsum, geigen, mixOmics, groupICA

VignetteBuilder knitr

NeedsCompilation no

Author Koki Tsuyuzaki [aut, cre]

Maintainer Koki Tsuyuzaki <k.t.the-answer@hotmail.co.jp>

Repository CRAN

Date/Publication 2023-04-28 08:00:02 UTC

R topics documented:

`CorrIndex` *CorrIndex*

Description

Calculate the CorrIndex of the cross-correlation matrix of S_true and estimated S. The closer the value is to 0, the closer estimated S is to S_true.

Usage

```
CorrIndex(cross_correlation_matrix)
```

Arguments

`cross_correlation_matrix`
: Cross-correlation matrix

Value

CorrIndex, which means the closeness between S and S_true , is returned.

Examples

```
S_true <- matrix(runif(5*5), nrow=5, ncol=5)
S <- matrix(runif(5*5), nrow=5, ncol=5)
CorrIndex(cor(S_true, S))
```

`GroupICA` *Group Independent Component Analysis (GroupICA)*

Description

The input data is assumed to be a list containing multiple matrices, which share common column.

Usage

```
GroupICA(
  Xs,
  J1,
  J2 = J1,
  algorithm = c("pooled", "Calhoun2009", "Pfister2018"),
 ica.algorithm = c("FastICA", "InfoMax", "ExtInfoMax", "JADE", "AuxICA1", "AuxICA2",
    "IPCA", "SIMBEC", "AMUSE", "SOBI", "FOBI", "ProDenICA", "RICA"),
  num.iter = 30,
  thr = 1e-10,
  verbose = FALSE
)
```

Arguments

Xs	A list containing multiple matrices
J1	Rank parameter to decompose
J2	Rank parameter used in Calhoun2009
algorithm	Pool algorithm to merge multiple ICA results (Default: pooled)
ica.algorithm	The decomposition algorithm (Default: "FastICA")
num.iter	The number of iterations
thr	The threshold to terminate the iteration (Default: 1E-10)
verbose	Verbose option

Value

A list containing the result of the decomposition

Examples

```
X1 <- matrix(runif(100*200), nrow=100, ncol=200)
X2 <- matrix(runif(150*200), nrow=150, ncol=200)
Xs <- list(X1=X1, X2=X2)
out <- GroupICA(Xs, J1=5)
```

ICA *Independent Component Analysis (Classic Methods)*

Description

The input data is assumed to be a matrix. ICA decomposes the matrix and extract the components that are statistically independent each other.

Usage

```
ICA(
  X,
  J,
  algorithm = c("FastICA", "InfoMax", "ExtInfoMax"),
  num.iter = 100,
  thr = 1e-10,
  nonlinear_func = c("tanh", "exp", "kurtosis"),
  learning_rate = 1,
  verbose = FALSE
)
```

Arguments

X	A matrix
J	Rank parameter to decompose
algorithm	The decomposition algorithm (Default: "FastICA")
num.iter	The number of iteration
thr	The threshold to terminate the iteration (Default: 1E-10)
nonlinear_func	The function used in FastICA (Default: "tanh")
learning_rate	The learning rate used in InfoMax or ExtInfoMax
verbose	Verbose option

Value

A list containing the result of the decomposition

Examples

```
X <- matrix(runif(100*200), nrow=100, ncol=200)
J <- 5
out.FastICA <- ICA(X, J=J, algorithm="FastICA")
out.InfoMax <- ICA(X, J=J, algorithm="InfoMax")
out.ExtInfoMax <- ICA(X, J=J, algorithm="ExtInfoMax")
```

ICA2 *Independent Component Analysis (Modern Methods)*

Description

The input data is assumed to be a matrix. ICA decomposes the matrix and extract the components that are statistically independent each other.

Usage

```
ICA2(
  X,
  J,
 algorithm = c("JADE", "AuxICA1", "AuxICA2", "IPCA", "SIMBEC", "AMUSE", "SOBI", "FOBI",
    "ProDenICA", "RICA"),
  num.iter = NULL,
  thr = 1e-10,
  r_list = NULL,
  omega_for_each_r = NULL,
  a_r_for_each_r = NULL,
  tau_list = NULL,
  num_bins = NULL,
  alpha = NULL,
  num_epoch = NULL,
  verbose = FALSE
)
```

Arguments

X	A matrix
J	Rank parameter to decompose
algorithm	The decomposition algorithm (Default: "JADE")
num.iter	The number of iteration
thr	The threshold to terminate the iteration (Default: 1E-10)
r_list	List of r-th order cumulants used in SIMBEC (Default: NULL)
omega_for_each_r	Weight vector of r_list used in SIMBEC (Default: NULL)
a_r_for_each_r	Parameter vector to specify the shape of partial activation function in SIMBEC (Default: NULL)
tau_list	List of lags to consider the auto-correlation used in AMUSE and SOBI (Default: NULL)
num_bins	Number of bins for histgram in ProDenICA (Default: NULL)
alpha	Learning rate used for gradient descent in RICA (Default: NULL)
num_epoch	Number of epoch used for gradient descent in RICA (Default: NULL)
verbose	Verbose option

Value

A list containing the result of the decomposition

Examples

```
ICA2
```

MICA *Multimodal independent component analysis*

Description

The input datasets are assumed to be two matrices sharing the column space. MICA decomposes the matrices simutanously and extracts the components that maximizes the mutual information between the components.

Usage

```
MICA(
  X,
  Y,
  J,
  eta = 1000 * 1e-04,
  verbose = FALSE,
  mu = 50 * 1e-04,
  gamma_ts = 1
)
```

Arguments

X	A matrix sharing the column space with Y (??? x N)
Y	A matrix sharing the column space with X (??? x N)
J	The rank parameter to decompose the matrices
eta	A learning rate parameter of stochastic gradient descent
verbose	Verbose option
mu	A learning rate parameter of stochastic gradient descent
gamma_ts	Weighting factor for dependence on independence

Value

A list containing the result of the decomposition

Examples

```
X <- array(runif(10*20), dim=c(10,20))
Y <- array(runif(15*20), dim=c(15,20))
J <- 20
out <- MICA(X, Y, J=J)
```

`MultilinearICA` *Multilinear independent component analysis*

Description

#' The input object is assumed to be a Tensor object defined by rTensor package. In MultilinearICA, ICA function is performed in each mode of the tensor.

Usage

```
MultilinearICA(
  X,
  Js = c(3, 3, 3),
  modes = 1:3,
 algorithm = c("FastICA", "InfoMax", "ExtInfoMax", "JADE", "AuxICA1", "AuxICA2", "IPCA",
    "SIMBEC", "AMUSE", "SOBI", "FOBI", "ProDenICA", "RICA")
)
```

Arguments

`X`	An rTensor object
`Js`	A vector to specify the rank in each mode (Default: c(3,3,3))
`modes`	A vector to specify which modes are decomposed (Default: 1:3)
`algorithm`	The algorithm to decompose the input tensor in each mode (Default: "FastICA")

Value

A list containing the result of the decomposition

Examples

```
library("rTensor")
arrX <- array(runif(10*20*30), dim=c(10,20,30))
X <- as.tensor(arrX)
Js <- c(2,3,4)
out <- MultilinearICA(X, Js=Js)
```

toyModel	*Toy model data for using ICA, MICA, and GroupICA There are 7 types of simulation: ICA_Type1: Time-independent sub-gaussian data ICA_Type2: Time-independent super-gaussian data ICA_Type3: Data mixed with signals having no time dependence and different kurtosis ICA_Type4: Time-dependent data ICA_Type5: Toydata to model IPCA in N < P systems MICA: Two time-serices data to model MICA GroupICA: Toydata to model GroupICA*

Description

Toy model data for using ICA, MICA, and GroupICA There are 7 types of simulation: ICA_Type1: Time-independent sub-gaussian data ICA_Type2: Time-independent super-gaussian data ICA_Type3: Data mixed with signals having no time dependence and different kurtosis ICA_Type4: Time-dependent data ICA_Type5: Toydata to model IPCA in N < P systems MICA: Two time-serices data to model MICA GroupICA: Toydata to model GroupICA

Usage

```
toyModel(model = "ICA_Type1", seeds = 123)
```

Arguments

model	"ICA_Type1", "ICA_Type2", "ICA_Type3", "ICA_Type4", and "ICA_Type5", "MICA", and "GrouICA" are available (Default: "ICA_Type1").
seeds	Random number for setting set.seeds in the function (Default: 123).

Value

A list containing simulation data sets.

Examples

```
data1 <- toyModel("ICA_Type1")
data2 <- toyModel("ICA_Type2")
data3 <- toyModel("ICA_Type3")
data4 <- toyModel("ICA_Type4")
data5 <- toyModel("ICA_Type5")
data6 <- toyModel("MICA")
data7 <- toyModel("GroupICA")
```

Index