Package ‘nnTensor’

November 15, 2022

Type Package

Title Non-Negative Tensor Decomposition

Version 1.1.10

Date 2022-11-12

Suggests testthat

Depends R (>= 3.4.0)

Imports methods, MASS, fields, rTensor, plot3D, tagcloud, ggplot2

Description Some functions for performing non-negative matrix factorization, non-negative CANDECOMP/PARAFAC (CP) decomposition, non-negative Tucker decomposition, and generating toy model data. See Andrzej Cichock et al (2009) and the reference section of GitHub README.md <`https://github.com/rikenbit/nnTensor`>, for details of the methods.

License MIT + file LICENSE

URL `https://github.com/rikenbit/nnTensor`

NeedsCompilation no

Author Koki Tsuyuzaki [aut, cre],
Manabu Ishii [aut],
Itoshi Nikaido [aut]

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

Repository CRAN

Date/Publication 2022-11-15 22:40:05 UTC

R topics documented:

`nnTensor-package` *Non-Negative Tensor Decomposition*

Description

Some functions for performing non-negative matrix factorization, non-negative CANDECOMP/PARAFAC (CP) decomposition, non-negative Tucker decomposition, and generating toy model data. See Andrzej Cichock et al (2009) and the reference section of GitHub README.md <https://github.com/rikenbit/nnTensor>, for details of the methods.

Details

The DESCRIPTION file:

Package:	nnTensor
Type:	Package
Title:	Non-Negative Tensor Decomposition
Version:	1.1.10
Date:	2022-11-12
Authors@R:	c(person("Koki", "Tsuyuzaki", role = c("aut", "cre"), email = "k.t.the-answer@hotmail.co.jp"), person("Manab
Suggests:	testthat
Depends:	R (>= 3.4.0)
Imports:	methods, MASS, fields, rTensor, plot3D, tagcloud, ggplot2
Description:	Some functions for performing non-negative matrix factorization, non-negative CANDECOMP/PARAFAC (C
License:	MIT + file LICENSE
URL:	https://github.com/rikenbit/nnTensor
Author:	Koki Tsuyuzaki [aut, cre], Manabu Ishii [aut], Itoshi Nikaido [aut]
Maintainer:	Koki Tsuyuzaki <k.t.the-answer@hotmail.co.jp>

Index of help topics:

```
GabrielNMF              Gabriel-type Bi-Cross-Validation for
                        Non-negative Matrix Factorization
NMF                     Non-negative Matrix Factorization Algorithms
                        (NMF)
NMTF                    Non-negative Matrix Tri-Factorization
                        Algorithms (NMTF)
```

`NTD`	`Non-negative Tucker Decomposition Algorithms (NTD)`
`NTF`	`Non-negative CP Decomposition Algorithms (NTF)`
`jNMF`	`Joint Non-negative Matrix Factorization Algorithms (jNMF)`
`kFoldMaskTensor`	`Mask tensors generator to perform k-fold cross validation`
`nnTensor-package`	`Non-Negative Tensor Decomposition`
`plot.NMF`	`Plot function of the result of NMF function`
`plotTensor2D`	`Plot function for visualization of matrix data structure`
`plotTensor3D`	`Plot function for visualization of tensor data structure`
`recTensor`	`Tensor Reconstruction from core tensor (S) and factor matrices (A)`
`siNMF`	`Simultaneous Non-negative Matrix Factorization Algorithms (siNMF)`
`toyModel`	`Toy model data for using NMF, NTF, and NTD`

Author(s)

NA

Maintainer: NA

References

Andrzej CICHOCK, et. al., (2009). Nonnegative Matrix and Tensor Factorizations. *John Wiley & Sons, Ltd*

Keigo Kimura, (2017). A Study on Efficient Algorithms for Nonnegative Matrix/Tensor Factorization. *Hokkaido University Collection of Scholarly and Academic Papers*

Andrzej CICHOCKI et. al., (2007). Non-negative Tensor Factorization using Alpha and Beta Divergence. *IEEE ICASSP 2007*

Anh Huy PHAN et. al., (2008). Multi-way Nonnegative Tensor Factorization Using Fast Hierarchical Alternating Least Squares Algorithm (HALS). *NOLTA2008*

Andrzej CICHOCKI et. al., (2008). Fast Local Algorithms for Large Scale Nonnegative Matrix and Tensor Factorizations. *IEICE Transactions on Fundamentals of Electronics, Communications and Computer Sciences*

Yong-Deok Kim et. al., (2007). Nonnegative Tucker Decomposition. *IEEE Conference on Computer Vision and Pattern Recognition*

Yong-Deok Kim et. al., (2008). Nonneegative Tucker Decomposition With Alpha-Divergence. *IEEE International Conference on Acoustics, Speech and Signal Processing*

Anh Huy Phan, (2008). Fast and efficient algorithms for nonnegative Tucker decomposition. *Advances in Neural Networks - ISNN2008*

Anh Hyu Phan et. al. (2011). Extended HALS algorithm for nonnegative Tucker decomposition and its applications for multiway analysis and classification. *Neurocomputing*

Jean-Philippe Brunet. et. al., (2004). Metagenes and molecular pattern discovery using matrix factorization. *PNAS*

Xiaoxu Han. (2007). CANCER MOLECULAR PATTERN DISCOVERY BY SUBSPACE CONSENSUS KERNEL CLASSIFICATION

Attila Frigyesi. et. al., (2008). Non-Negative Matrix Factorization for the Analysis of Complex Gene Expression Data: Identification of Clinically Relevant Tumor Subtypes. *Cancer Informatics*

Haesun Park. et. al., (2019). Lecture 3: Nonnegative Matrix Factorization: Algorithms and Applications. *SIAM Gene Golub Summer School, Aussois France, June 18, 2019*

Chunxuan Shao. et. al., (2017). Robust classification of single-cell transcriptome data by nonnegative matrix factorization. *Bioinformatics*

Paul Fogel (2013). Permuted NMF: A Simple Algorithm Intended to Minimize the Volume of the Score Matrix

Philip M. Kim. et. al., (2003). Subsystem Identification Through Dimensionality Reduction of Large-Scale Gene Expression Data. *Genome Research*

Lucie N. Hutchins. et. al., (2008). Position-dependent motif characterization using non-negative matrix factorization. *Bioinformatics*

Patrik O. Hoyer (2004). Non-negative Matrix Factorization with Sparseness Constraints. *Journal of Machine Learning 5*

N. Fujita et al., (2018) Biomarker discovery by integrated joint non-negative matrix factorization and pathway signature analyses, *Scientific Report*

Art B. Owen et. al., (2009). Bi-Cross-Validation of the SVD and the Nonnegative Matrix Factorization. *The Annals of Applied Statistics*

See Also

`toyModel`,`NMF`,`NTF`,`NTD`,`recTensor`,`plotTensor3D`

Examples

```
ls("package:nnTensor")
```

`GabrielNMF` *Gabriel-type Bi-Cross-Validation for Non-negative Matrix Factorization*

Description

The input data is assumed to be non-negative matrix. GabrielNMF devides the input file into four matrices (A, B, C, and D) and perform cross validation by the prediction of A from the matrices B, C, and D.

Usage

```
GabrielNMF(X, J = 3, nx = 5, ny = 5, ...)
```

Arguments

`X`	The input matrix which has N-rows and M-columns.
`J`	The number of low-dimension (J < N, M).
`nx`	The number of hold-out in row-wise direction (2 < nx < N).
`ny`	The number of hold-out in row-wise direction (2 < ny < M).
`...`	Other parameters for NMF function.

Value

TestRecError : The reconstruction error calculated by Gabriel-style Bi-Cross Validation.

Author(s)

Koki Tsuyuzaki

References

Art B. Owen et. al., (2009). Bi-Cross-Validation of the SVD and the Nonnegative Matrix Factorization. *The Annals of Applied Statistics*

Examples

```
if(interactive()){
  # Test data
  matdata <- toyModel(model = "NMF")

  # Bi-Cross-Validation
  BCV <- rep(0, length=5)
  names(BCV) <- 2:6
  for(j in seq(BCV)){
    print(j+1)
    BCV[j] <- mean(GabrielNMF(matdata, J=j+1, nx=2, ny=2)$TestRecError)
  }
  proper.rank <- as.numeric(names(BCV)[which(BCV == min(BCV))])

  # NMF
  out <- NMF(matdata, J=proper.rank)
}
```

`jNMF` *Joint Non-negative Matrix Factorization Algorithms (jNMF)*

Description

The input data objects are assumed to be non-negative matrices. jNMF decompose the matrices to two low-dimensional factor matices simultaneously.

Usage

```
jNMF(X, M=NULL, pseudocount=.Machine$double.eps,
  initW=NULL, initV=NULL, initH=NULL, fixW=FALSE, fixV=FALSE,
  fixH=FALSE,
  L1_W=1e-10, L1_V=1e-10, L1_H=1e-10,
  L2_W=1e-10, L2_V=1e-10, L2_H=1e-10,
  J = 3, w=NULL, algorithm = c("Frobenius", "KL", "IS", "PLTF"),
  p=1, thr = 1e-10, num.iter = 100, viz = FALSE,
  figdir = NULL, verbose = FALSE)
```

Arguments

`X`	A list containing input matrices (X_k, <N*Mk>, k=1..K).
`M`	A list containing the mask matrices (X_k, <N*Mk>, k=1..K). If the input matrix has missing values, specify the element as 0 (otherwise 1).
`pseudocount`	The pseudo count to avoid zero division, when the element is zero (Default: Machine Epsilon).
`initW`	The initial values of factor matrix W, which has N-rows and J-columns (Default: NULL).
`initV`	A list containing the initial values of multiple factor matrices (V_k, <N*J>, k=1..K, Default: NULL).
`initH`	A list containing the initial values of multiple factor matrices (H_k, <Mk*J>, k=1..K, Default: NULL).
`fixW`	Whether the factor matrix W is updated in each iteration step (Default: FALSE).
`fixV`	Whether the factor matrices Vk are updated in each iteration step (Default: FALSE).
`fixH`	Whether the factor matrices Hk are updated in each iteration step (Default: FALSE).
`L1_W`	Paramter for L1 regularitation (Default: 1e-10). This also works as small positive constant to prevent division by zero, so should be set as 0.
`L1_V`	Paramter for L1 regularitation (Default: 1e-10). This also works as small positive constant to prevent division by zero, so should be set as 0.
`L1_H`	Paramter for L1 regularitation (Default: 1e-10). This also works as small positive constant to prevent division by zero, so should be set as 0.
`L2_W`	Paramter for L2 regularitation (Default: 1e-10).
`L2_V`	Paramter for L2 regularitation (Default: 1e-10).
`L2_H`	Paramter for L2 regularitation (Default: 1e-10).
`J`	Number of low-dimension (J < N, Mk).
`w`	Weight vector (Default: NULL)
`algorithm`	Divergence between X and X_bar. "Frobenius", "KL", and "IS" are available (Default: "KL").
`p`	The parameter of Probabilistic Latent Tensor Factorization (p=0: Frobenius, p=1: KL, p=2: IS)

`thr`	When error change rate is lower than thr, the iteration is terminated (Default: 1E-10).
`num.iter`	The number of interation step (Default: 100).
`viz`	If viz == TRUE, internal reconstructed matrix can be visualized.
`figdir`	the directory for saving the figure, when viz == TRUE.
`verbose`	If verbose == TRUE, Error change rate is generated in console windos.

Value

W : A matrix which has N-rows and J-columns (J < N, Mk). V : A list which has multiple elements containing N-rows and J-columns (J < N, Mk). H : A list which has multiple elements containing Mk-rows and J-columns matrix (J < N, Mk). RecError : The reconstruction error between data matrix and reconstructed matrix from W and H. TrainRecError : The reconstruction error calculated by training set (observed values specified by M). TestRecError : The reconstruction error calculated by test set (missing values specified by M). RelChange : The relative change of the error.

Author(s)

Koki Tsuyuzaki

References

Liviu Badea, (2008) Extracting Gene Expression Profiles Common to Colon and Pancreatic Adenocarcinoma using Simultaneous nonnegative matrix factorization. *Pacific Symposium on Biocomputing* 13:279-290

Shihua Zhang, et al. (2012) Discovery of multi-dimensional modules by integrative analysis of cancer genomic data. *Nucleic Acids Research* 40(19), 9379-9391

Zi Yang, et al. (2016) A non-negative matrix factorization method for detecting modules in heterogeneous omics multi-modal data, *Bioinformatics* 32(1), 1-8

Y. Kenan Yilmaz et al., (2010) Probabilistic Latent Tensor Factorization, *International Conference on Latent Variable Analysis and Signal Separation* 346-353

N. Fujita et al., (2018) Biomarker discovery by integrated joint non-negative matrix factorization and pathway signature analyses, *Scientific Report*

Examples

```
matdata <- toyModel(model = "siNMF_Hard")
out <- jNMF(matdata, J=2, num.iter=2)
```

`kFoldMaskTensor` *Mask tensors generator to perform k-fold cross validation*

Description

The output multiple mask tensors can be immediately specified as the argument M for NTF() or NTD().

Usage

```
kFoldMaskTensor(X, k=5, avoid.zero=TRUE, seeds=123)
```

Arguments

`X`	An rTensor object.
`k`	Number of split for k-fold cross validation.
`avoid.zero`	If TRUE, only non-zero elements are splitted (Default: TRUE).
`seeds`	Random seed to use for set.seed().

Author(s)

Koki Tsuyuzaki

Examples

```
tensordata <- toyModel(model = "CP")

Ms <- kFoldMaskTensor(tensordata, k=5, avoid.zero=TRUE, seeds=123)
```

`NMF` *Non-negative Matrix Factorization Algorithms (NMF)*

Description

The input data is assumed to be non-negative matrix. NMF decompose the matrix to two low-dimensional factor matices. This function is also used as initialization step of tensor decomposition (see also NTF and NTD).

Usage

```
NMF(X, M=NULL, pseudocount=.Machine$double.eps, initU=NULL, initV=NULL,
  fixU=FALSE, fixV=FALSE,
  L1_U=1e-10, L1_V=1e-10, L2_U=1e-10, L2_V=1e-10, J = 3,
  rank.method=c("all", "ccc", "dispersion", "rss", "evar", "residuals",
    "sparseness.basis", "sparseness.coef", "sparseness2.basis",
    "sparseness2.coef", "norm.info.gain.basis", "norm.info.gain.coef",
    "singular",  "volume", "condition"), runtime=30,
  algorithm = c("Frobenius", "KL", "IS", "Pearson", "Hellinger", "Neyman",
    "Alpha", "Beta", "ALS", "PGD", "HALS", "GCD", "Projected", "NHR", "DTPP",
    "Orthogonal", "OrthReg"), Alpha = 1, Beta = 2,
  eta = 1e-04, thr1 = 1e-10, thr2 = 1e-10, tol = 1e-04,
  num.iter = 100, viz = FALSE, figdir = NULL, verbose = FALSE)
```

Arguments

`X`	The input matrix which has N-rows and M-columns.
`M`	The mask matrix which has N-rows and M-columns. If the input matrix has missing values, specify the element as 0 (otherwise 1).
`pseudocount`	The pseudo count to avoid zero division, when the element is zero (Default: Machine Epsilon).
`initU`	The initial values of factor matrix U, which has N-rows and J-columns (Default: NULL).
`initV`	The initial values of factor matrix V, which has M-rows and J-columns (Default: NULL).
`fixU`	Whether the factor matrix U is updated in each iteration step (Default: FALSE).
`fixV`	Whether the factor matrix V is updated in each iteration step (Default: FALSE).
`L1_U`	Paramter for L1 regularitation (Default: 1e-10). This also works as small positive constant to prevent division by zero, so should be set as 0.
`L1_V`	Paramter for L1 regularitation (Default: 1e-10). This also works as small positive constant to prevent division by zero, so should be set as 0.
`L2_U`	Paramter for L2 regularitation (Default: 1e-10).
`L2_V`	Paramter for L2 regularitation (Default: 1e-10).
`J`	The number of low-dimension (J < N, M). If a numerical vector is specified (e.g. 2:6), the appropriate rank is estimated.
`rank.method`	The rank estimation method (Default: "all"). Only if the J option is specified as a numerical vector longer than two, this option will be active.
`runtime`	The number of trials to estimate rank (Default: 10).
`algorithm`	NMF algorithms. "Frobenius", "KL", "IS", "Pearson", "Hellinger", "Neyman", "Alpha", "Beta", "ALS", "PGD", "HALS", "GCD", "Projected", "NHR", "DTPP", "Orthogonal", and "OrthReg" are available (Default: "Frobenius").
`Alpha`	The parameter of Alpha-divergence.
`Beta`	The parameter of Beta-divergence.

`eta`	The stepsize for PGD algorithm (Default: 0.0001).
`thr1`	When error change rate is lower than thr1, the iteration is terminated (Default: 1E-10).
`thr2`	If the minus-value is generated, replaced as thr2 (Default: 1E-10). This value is used within the internal function .positive().
`tol`	The tolerance parameter used in GCD algorithm.
`num.iter`	The number of interation step (Default: 100).
`viz`	If viz == TRUE, internal reconstructed matrix can be visualized.
`figdir`	The directory for saving the figure, when viz == TRUE.
`verbose`	If verbose == TRUE, Error change rate is generated in console window.

Value

U : A matrix which has N-rows and J-columns (J < N, M). V : A matrix which has M-rows and J-columns (J < N, M). J : The number of dimension (J < N, M). RecError : The reconstruction error between data tensor and reconstructed tensor from U and V. TrainRecError : The reconstruction error calculated by training set (observed values specified by M). TestRecError : The reconstruction error calculated by test set (missing values specified by M). RelChange : The relative change of the error. Trial : All the results of the trials to estimate the rank. Runtime : The number of the trials to estimate the rank. RankMethod : The rank estimation method.

Author(s)

Koki Tsuyuzaki

References

Andrzej CICHOCK, et. al., (2009). Nonnegative Matrix and Tensor Factorizations. *John Wiley & Sons, Ltd*

Keigo Kimura, (2017). A Study on Efficient Algorithms for Nonnegative Matrix/ Tensor Factorization. *Hokkaido University Collection of Scholarly and Academic Papers*

Examples

```
if(interactive()){
  # Test data
  matdata <- toyModel(model = "NMF")

  # Simple usage
  out <- NMF(matdata, J=5)

  # Rank estimation mode (single method)
  out2 <- NMF(matdata, J=2:10, rank.method="ccc", runtime=3)
  plot(out2)

  # Rank estimation mode (all method)
  out3 <- NMF(matdata, J=2:10, rank.method="all", runtime=10)
  plot(out3)
}
```

NMTF *Non-negative Matrix Tri-Factorization Algorithms (NMTF)*

Description

The input data is assumed to be non-negative matrix. NMTF decompose the matrix to three low-dimensional factor matices.

Usage

```
NMTF(X, pseudocount=.Machine$double.eps,
    initU=NULL, initS=NULL, initV=NULL,
    fixU=FALSE, fixS=FALSE, fixV=FALSE,
    L1_U=1e-10, L1_S=1e-10, L1_V=1e-10,
    L2_U=1e-10, L2_S=1e-10, L2_V=1e-10,
    orthU=FALSE, orthV=FALSE,
    rank = c(3, 4),
    algorithm = c("Frobenius", "KL", "IS", "ALS", "PG", "COD", "Beta"),
    Beta = 2, root = FALSE, thr = 1e-10, num.iter = 100,
    viz = FALSE, figdir = NULL, verbose = FALSE)
```

Arguments

`X`	The input matrix which has N-rows and M-columns.
`pseudocount`	The pseudo count to avoid zero division, when the element is zero (Default: Machine Epsilon).
`initU`	The initial values of factor matrix U, which has N-rows and J1-columns (Default: NULL).
`initS`	The initial values of factor matrix S, which has J1-rows and J2-columns (Default: NULL).
`initV`	The initial values of factor matrix V, which has M-rows and J2-columns (Default: NULL).
`fixU`	Whether the factor matrix U is updated in each iteration step (Default: FALSE).
`fixS`	Whether the factor matrix S is updated in each iteration step (Default: FALSE).
`fixV`	Whether the factor matrix V is updated in each iteration step (Default: FALSE).
`L1_U`	Paramter for L1 regularitation (Default: 1e-10).
`L1_S`	Paramter for L1 regularitation (Default: 1e-10).
`L1_V`	Paramter for L1 regularitation (Default: 1e-10).
`L2_U`	Paramter for L2 regularitation (Default: 1e-10).
`L2_S`	Paramter for L2 regularitation (Default: 1e-10).
`L2_V`	Paramter for L2 regularitation (Default: 1e-10).
`orthU`	Whether the column vectors of matrix U are orthogonalized (Default: FALSE).

`orthV`	Whether the column vectors of matrix V are orthogonalized (Default: FALSE).
`rank`	The number of low-dimension (J1 (< N) and J2 (< M)) (Default: c(3,4)).
`algorithm`	NMTF algorithms. "Frobenius", "KL", "IS", "ALS", "PG", "COD", and "Beta" are available (Default: "Frobenius").
`Beta`	The parameter of Beta-divergence (Default: 2, which means "Frobenius").
`root`	Whether square root is calculed in each iteration (Default: FALSE).
`thr`	When error change rate is lower than thr, the iteration is terminated (Default: 1E-10).
`num.iter`	The number of interation step (Default: 100).
`viz`	If viz == TRUE, internal reconstructed matrix can be visualized.
`figdir`	The directory for saving the figure, when viz == TRUE.
`verbose`	If verbose == TRUE, Error change rate is generated in console window.

Value

U : A matrix which has N-rows and J1-columns (J1 < N). S : A matrix which has J1-rows and J2-columns. V : A matrix which has M-rows and J2-columns (J2 < M). RecError : The reconstruction error between data tensor and reconstructed tensor from U and V. RelChange : The relative change of the error. algorithm: algorithm specified.

Author(s)

Koki Tsuyuzaki

References

Fast Optimization of Non-Negative Matrix Tri-Factorization: Supporting Information, Andrej Copar, et. al., *PLOS ONE*, 14(6), e0217994, 2019

Co-clustering by Block Value Decomposition, Bo Long et al., *SIGKDD'05*, 2005

Orthogonal Nonnegative Matrix Tri-Factorizations for Clustering, Chris Ding et. al., *12th ACM SIGKDD*, 2006

Examples

```
if(interactive()){
  # Test data
  matdata <- toyModel(model = "NMF")

  # Simple usage
  out <- NMTF(matdata, rank=c(4,4))
}
```

NTD *Non-negative Tucker Decomposition Algorithms (NTD)*

Description

The input data is assumed to be non-negative tensor. NTD decompose the tensor to the dense core tensor (S) and low-dimensional factor matices (A).

Usage

```
NTD(X, M=NULL, pseudocount=.Machine$double.eps, initS=NULL, initA=NULL,
  fixS=FALSE, fixA=FALSE, L1_A=1e-10, L2_A=1e-10, rank = c(3, 3, 3),
  modes = 1:3,
  algorithm = c("Frobenius", "KL", "IS", "Pearson", "Hellinger", "Neyman",
    "HALS", "Alpha", "Beta", "NMF"), init = c("NMF", "ALS", "Random"),
  nmf.algorithm = c("Frobenius", "KL", "IS", "Pearson", "Hellinger", "Neyman",
    "Alpha", "Beta", "ALS", "PGD", "HALS", "GCD", "Projected", "NHR", "DTPP",
    "Orthogonal", "OrthReg"),
  Alpha = 1,
  Beta = 2, thr = 1e-10, num.iter = 100, num.iter2 = 10, viz = FALSE,
  figdir = NULL, verbose = FALSE)
```

Arguments

`X`	The input tensor which has I1, I2, and I3 dimensions.
`M`	The mask tensor which has I1, I2, and I3 dimensions. If the mask tensor has missing values, specify the element as 0 (otherwise 1).
`pseudocount`	The pseudo count to avoid zero division, when the element is zero (Default: Machine Epsilon).
`initS`	The initial values of core tensor which has J1, J2, and J3 dimensions (Default: NULL).
`initA`	A list containing the initial values of multiple factor matrices (A_k, <Ik*Jk>, k=1..K, Default: NULL).
`fixS`	Whether the core tensor S is updated in each iteration step (Default: FALSE).
`fixA`	Whether the factor matrices Ak are updated in each iteration step (Default: FALSE).
`L1_A`	Paramter for L1 regularitation (Default: 1e-10). This also works as small positive constant to prevent division by zero, so should be set as 0.
`L2_A`	Paramter for L2 regularitation (Default: 1e-10).
`rank`	The number of low-dimension in each mode (J1, J2, J3, J1<I1, J2<I2, J3 < I3) (Default: c(3,3,3)).
`modes`	The vector of the modes on whih to perform the decomposition (Default: 1:3 <all modes>).

`algorithm`	NTD algorithms. "Frobenius", "KL", "IS", "Pearson", "Hellinger", "Neyman", "HALS", "Alpha", "Beta", "NMF" are available (Default: "Frobenius").
`nmf.algorithm`	NMF algorithms, when the algorithm is "NMF". "Frobenius", "KL", "IS", "Pearson", "Hellinger", "Neyman", "Alpha", "Beta", "ALS", "PGD", "HALS", "GCD", "Projected", "NHR", "DTPP", "Orthogonal", and "OrthReg" are available (Default: "Frobenius").
`init`	The initialization algorithms. "NMF", "ALS", and "Random" are available (Default: "NMF").
`Alpha`	The parameter of Alpha-divergence.
`Beta`	The parameter of Beta-divergence.
`thr`	When error change rate is lower than thr1, the iteration is terminated (Default: 1E-10).
`num.iter`	The number of interation step (Default: 100).
`num.iter2`	The number of NMF interation step, when the algorithm is "NMF" (Default: 10).
`viz`	If viz == TRUE, internal reconstructed tensor can be visualized.
`figdir`	the directory for saving the figure, when viz == TRUE (Default: NULL).
`verbose`	If verbose == TRUE, Error change rate is generated in console windos.

Value

S : Tensor object, which is defined as S4 class of rTensor package. A : A list containing three factor matrices. RecError : The reconstruction error between data tensor and reconstructed tensor from S and A. TrainRecError : The reconstruction error calculated by training set (observed values specified by M). TestRecError : The reconstruction error calculated by test set (missing values specified by M). RelChange : The relative change of the error.

Author(s)

Koki Tsuyuzaki

References

Yong-Deok Kim et. al., (2007). Nonnegative Tucker Decomposition. *IEEE Conference on Computer Vision and Pattern Recognition*

Yong-Deok Kim et. al., (2008). Nonneegative Tucker Decomposition With Alpha-Divergence. *IEEE International Conference on Acoustics, Speech and Signal Processing*

Anh Huy Phan, (2008). Fast and efficient algorithms for nonnegative Tucker decomposition. *Advances in Neural Networks - ISNN2008*

Anh Hyu Phan et. al. (2011). Extended HALS algorithm for nonnegative Tucker decomposition and its applications for multiway analysis and classification. *Neurocomputing*

See Also

`plotTensor3D`

Examples

```
tensordata <- toyModel(model = "Tucker")
out <- NTD(tensordata, rank=c(2,2,2), algorithm="Frobenius",
  init="Random", num.iter=2)
```

NTF *Non-negative CP Decomposition Algorithms (NTF)*

Description

The input data is assumed to be non-negative tensor. NTF decompose the tensor to the diagonal core tensor (S) and low-dimensional factor matices (A).

Usage

```
NTF(X, M=NULL, pseudocount=.Machine$double.eps, initA=NULL,
  fixA=FALSE, L1_A=1e-10, L2_A=1e-10, rank = 3,
  algorithm = c("Frobenius", "KL", "IS", "Pearson", "Hellinger", "Neyman",
    "HALS", "Alpha-HALS", "Beta-HALS", "Alpha", "Beta"),
  init = c("NMF", "ALS", "Random"), Alpha = 1,
  Beta = 2, thr = 1e-10, num.iter = 100, viz = FALSE,
  figdir = NULL, verbose = FALSE)
```

Arguments

`X`	The input tensor which has I1, I2, and I3 dimensions.
`M`	The mask tensor which has I1, I2, and I3 dimensions. If the mask tensor has missing values, specify the element as 0 (otherwise 1).
`pseudocount`	The pseudo count to avoid zero division, when the element is zero (Default: Machine Epsilon).
`initA`	A list containing the initial values of multiple factor matrices (A_k, <Ik*Jk>, k=1..K, Default: NULL).
`fixA`	Whether the factor matrices Ak are updated in each iteration step (Default: FALSE).
`L1_A`	Paramter for L1 regularitation (Default: 1e-10). This also works as small positive constant to prevent division by zero, so should be set as 0.
`L2_A`	Paramter for L2 regularitation (Default: 1e-10).
`rank`	The number of low-dimension in each mode (J1=J2=J3, J1<I1, J2<I2, J3 < I3) (Default: 3).
`algorithm`	NTF algorithms. "Frobenius", "KL", "IS", "Pearson", "Hellinger", "Neyman", "HALS", "Alpha-HALS", "Beta-HALS", "Alpha", and "Beta" are available (Default: "Frobenius").
`init`	The initialization algorithms. "NMF", "ALS", and "Random" are available (Default: "NMF").

`Alpha`	The parameter of Alpha-divergence.
`Beta`	The parameter of Beta-divergence.
`thr`	When error change rate is lower than thr1, the iteration is terminated (Default: 1E-10).
`num.iter`	The number of interation step (Default: 100).
`viz`	If viz == TRUE, internal reconstructed tensor can be visualized.
`figdir`	the directory for saving the figure, when viz == TRUE (Default: NULL).
`verbose`	If verbose == TRUE, Error change rate is generated in console windos.

Value

S : Tensor object, which is defined as S4 class of rTensor package. A : A list containing three factor matrices. RecError : The reconstruction error between data tensor and reconstructed tensor from S and A. TrainRecError : The reconstruction error calculated by training set (observed values specified by M). TestRecError : The reconstruction error calculated by test set (missing values specified by M). RelChange : The relative change of the error.

Author(s)

Koki Tsuyuzaki

References

Andrzej CICHOCKI et. al., (2007). Non-negative Tensor Factorization using Alpha and Beta Divergence. *IEEE ICASSP 2007*

Anh Huy PHAN et. al., (2008). Multi-way Nonnegative Tensor Factorization Using Fast Hierarchical Alternating Least Squares Algorithm (HALS). *NOLTA2008*

Andrzej CICHOCKI et. al., (2008). Fast Local Algorithms for Large Scale Nonnegative Matrix and Tensor Factorizations. *IEICE Transactions on Fundamentals of Electronics, Communications and Computer Sciences*

See Also

`plotTensor3D`

Examples

```
tensordata <- toyModel(model = "CP")
out <- NTF(tensordata, rank=3, algorithm="Beta-HALS", num.iter=2)
```

`plot.NMF` *Plot function of the result of NMF function*

Description

Only if J is specified as a vector longer than 1, this function will be active.

Usage

```
plot(x, ...)
```

Arguments

`x`	The result of NMF function (NMF class).
`...`	Optional parameter for plot.

Value

A ggplot will be gerarated.

Author(s)

Koki Tsuyuzaki

References

Jean-Philippe Brunet. et. al., (2004). Metagenes and molecular pattern discovery using matrix factorization. *PNAS*

Xiaoxu Han. (2007). CANCER MOLECULAR PATTERN DISCOVERY BY SUBSPACE CONSENSUS KERNEL CLASSIFICATION

Attila Frigyesi. et. al., (2008). Non-Negative Matrix Factorization for the Analysis of Complex Gene Expression Data: Identification of Clinically Relevant Tumor Subtypes. *Cancer Informatics*

Haesun Park. et. al., (2019). Lecture 3: Nonnegative Matrix Factorization: Algorithms and Applications. *SIAM Gene Golub Summer School, Aussois France, June 18, 2019*

Chunxuan Shao. et. al., (2017). Robust classification of single-cell transcriptome data by nonnegative matrix factorization. *Bioinformatics*

Paul Fogel (2013). Permuted NMF: A Simple Algorithm Intended to Minimize the Volume of the Score Matrix

Philip M. Kim. et. al., (2003). Subsystem Identification Through Dimensionality Reduction of Large-Scale Gene Expression Data. *Genome Research*

Lucie N. Hutchins. et. al., (2008). Position-dependent motif characterization using non-negative matrix factorization. *Bioinformatics*

Patrik O. Hoyer (2004). Non-negative Matrix Factorization with Sparseness Constraints. *Journal of Machine Learning 5*

Examples

```
plot.NMF
```

`plotTensor2D` *Plot function for visualization of matrix data structure*

Description

Combined with recTensor function and the result of NTF or NTD, the reconstructed tensor structure can be visullized.

Usage

```
plotTensor2D(X = NULL, method=c("sd", "mad"),
    sign=c("positive", "negative", "both"), thr=2)
```

Arguments

`X`	Matrix object.
`method`	Cutoff method to focus on large/small value in the tensor data (Default: "sd").
`sign`	Direction to cutoff the large/small value in the tensor data (Default: "positive").
`thr`	Threshold of cutoff method (Default: 2).

Author(s)

Koki Tsuyuzaki

Examples

```
tensordata <- toyModel(model = "CP")

out <- NTF(tensordata, rank=3, num.iter=2)

tmp <- tempdir()

png(filename=paste0(tmp, "/NTF.png"))
plotTensor2D(out$A[[1]])
dev.off()
```

`plotTensor3D` *Plot function for visualization of tensor data structure*

Description

Combined with recTensor function and the result of NTF or NTD, the reconstructed tensor structure can be visullized.

Usage

```
plotTensor3D(X = NULL, method=c("sd", "mad"),
  sign=c("positive", "negative", "both"), thr=2)
```

Arguments

`X`	Tensor object, which is defined as S4 class of rTensor package.
`method`	Cutoff method to focus on large/small value in the tensor data (Default: "sd").
`sign`	Direction to cutoff the large/small value in the tensor data (Default: "positive").
`thr`	Threshold of cutoff method (Default: 2).

Author(s)

Koki Tsuyuzaki

Examples

```
tensordata <- toyModel(model = "CP")

out <- NTF(tensordata, rank=3, algorithm="Beta-HALS", num.iter=2)

tmp <- tempdir()

png(filename=paste0(tmp, "/NTF.png"))
plotTensor3D(recTensor(out$S, out$A))
dev.off()
```

`recTensor` *Tensor Reconstruction from core tensor (S) and factor matrices (A)*

Description

Combined with plotTensor3D function and the result of NTF or NTD, the reconstructed tesor structure can be visullized.

Usage

```
recTensor(S = NULL, A = NULL, idx = 1:3, reverse = FALSE)
```

Arguments

`S`	Tensor object, which is defined as S4 class of rTensor package.
`A`	A list containing three factor matrices.
`idx`	The direction of mode-n muliplication (Default: 1:3). For example idx=1 is defined. S x_1 A is calculated (x_1 : mode-1 multiplication).
`reverse`	If reverse = TRUE, t(A[[n]]) is multiplicated to S (Default: FALSE).

Value

Tensor object, which is defined as S4 class of rTensor package.

Author(s)

Koki Tsuyuzaki

See Also

`Tensor-class`, `NTF`, `NTD`

Examples

```
tensordata <- toyModel(model = "CP")
out <- NTF(tensordata, rank=3, algorithm="Beta-HALS", num.iter=2)
rec <- recTensor(out$S, out$A)
```

`siNMF` *Simultaneous Non-negative Matrix Factorization Algorithms (siNMF)*

Description

The input data objects are assumed to be non-negative matrices. siNMF decompose the matrices to two low-dimensional factor matices simultaneously.

Usage

```
siNMF(X, M=NULL, pseudocount=.Machine$double.eps, initW=NULL, initH=NULL,
  fixW=FALSE, fixH=FALSE,
  L1_W=1e-10, L1_H=1e-10, L2_W=1e-10, L2_H=1e-10, J = 3,
  w=NULL, algorithm = c("Frobenius", "KL", "IS", "PLTF"), p=1,
    thr = 1e-10, num.iter = 100,
    viz = FALSE, figdir = NULL, verbose = FALSE)
```

Arguments

`X`	A list containing the input matrices (X_k, <N*Mk>, k=1..K).
`M`	A list containing the mask matrices (X_k, <N*Mk>, k=1..K). If the input matrix has missing values, specify the element as 0 (otherwise 1).
`pseudocount`	The pseudo count to avoid zero division, when the element is zero (Default: Machine Epsilon).
`initW`	The initial values of factor matrix W, which has N-rows and J-columns (Default: NULL).
`initH`	A list containing the initial values of multiple factor matrices (H_k, <Mk*J>, k=1..K, Default: NULL).
`fixW`	Whether the factor matrix W is updated in each iteration step (Default: FALSE).
`fixH`	Whether the factor matrices Hk are updated in each iteration step (Default: FALSE).
`L1_W`	Paramter for L1 regularitation (Default: 1e-10). This also works as small positive constant to prevent division by zero, so should be set as 0.
`L1_H`	Paramter for L1 regularitation (Default: 1e-10). This also works as small positive constant to prevent division by zero, so should be set as 0.
`L2_W`	Paramter for L2 regularitation (Default: 1e-10).
`L2_H`	Paramter for L2 regularitation (Default: 1e-10).
`J`	Number of low-dimension (J < N, Mk).
`w`	Weight vector (Default: NULL)
`algorithm`	Divergence between X and X_bar. "Frobenius", "KL", and "IS" are available (Default: "KL").
`p`	The parameter of Probabilistic Latent Tensor Factorization (p=0: Frobenius, p=1: KL, p=2: IS)
`thr`	When error change rate is lower than thr, the iteration is terminated (Default: 1E-10).
`num.iter`	The number of interation step (Default: 100).
`viz`	If viz == TRUE, internal reconstructed matrix can be visualized.
`figdir`	the directory for saving the figure, when viz == TRUE.
`verbose`	If verbose == TRUE, Error change rate is generated in console windos.

Value

W : A matrix which has N-rows and J-columns (J < N, Mk). H : A list which has multiple elements containing Mk-rows and J-columns matrix (J < N, Mk). RecError : The reconstruction error between data matrix and reconstructed matrix from W and H. TrainRecError : The reconstruction error calculated by training set (observed values specified by M). TestRecError : The reconstruction error calculated by test set (missing values specified by M). RelChange : The relative change of the error.

Author(s)

Koki Tsuyuzaki

References

Liviu Badea, (2008) Extracting Gene Expression Profiles Common to Colon and Pancreatic Adenocarcinoma using Simultaneous nonnegative matrix factorization. *Pacific Symposium on Biocomputing* 13:279-290

Shihua Zhang, et al. (2012) Discovery of multi-dimensional modules by integrative analysis of cancer genomic data. *Nucleic Acids Research* 40(19), 9379-9391

Zi Yang, et al. (2016) A non-negative matrix factorization method for detecting modules in heterogeneous omics multi-modal data, *Bioinformatics* 32(1), 1-8

Y. Kenan Yilmaz et al., (2010) Probabilistic Latent Tensor Factorization, *International Conference on Latent Variable Analysis and Signal Separation* 346-353

N. Fujita et al., (2018) Biomarker discovery by integrated joint non-negative matrix factorization and pathway signature analyses, *Scientific Report*

Examples

```
matdata <- toyModel(model = "siNMF_Easy")
out <- siNMF(matdata, J=2, num.iter=2)
```

`toyModel` *Toy model data for using NMF, NTF, and NTD*

Description

The data is used for confirming the algorithm are properly working.

Usage

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

Arguments

`model`	Single character string is specified. "NMF", "CP", and "Tucker" are available (Default: "CP").
`seeds`	Random number for setting set.seeds in the function (Default: 123).

Value

If model is specified as "NMF", a matrix is generated. Otherwise, a tensor is generated.

Author(s)

Koki Tsuyuzaki

See Also

`NMF`, `NTF`, `NTD`

Examples

```
matdata <- toyModel(model = "NMF", seeds=123)
tensordata1 <- toyModel(model = "CP", seeds=123)
tensordata2 <- toyModel(model = "Tucker", seeds=123)
```

Index