

Package ‘dcTensor’

August 25, 2025

Type Package

Title Discrete Matrix/Tensor Decomposition

Version 1.3.1

Date 2025-08-25

Depends R (>= 3.4.0)

Imports methods, MASS, fields, rTensor, nnTensor

Suggests knitr, rmarkdown, testthat

Description Semi-Binary and Semi-Ternary Matrix Decomposition are performed based on Non-negative Matrix Factorization (NMF) and Singular Value Decomposition (SVD). For the details of the methods, see the reference section of GitHub README.md <<https://github.com/rikenbit/dcTensor>>.

License MIT + file LICENSE

URL <https://github.com/rikenbit/dcTensor>

VignetteBuilder knitr

NeedsCompilation no

Author Koki Tsuyuzaki [aut, cre]

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

Repository CRAN

Date/Publication 2025-08-25 02:50:09 UTC

Contents

dcTensor-package	2
djNMF	3
dNMF	6
dNMTF	7
dNTD	10
dNTF	12
dPLS	14
dsiNMF	15
dSVD	17
toyModel	19

Index**20**

dcTensor-package

*Discrete Matrix/Tensor Decomposition***Description**

Semi-Binary and Semi-Ternary Matrix Decomposition are performed based on Non-negative Matrix Factorization (NMF) and Singular Value Decomposition (SVD). For the details of the methods, see the reference section of GitHub README.md <<https://github.com/rikenbit/dcTensor>>.

Details

The DESCRIPTION file:

```
Package:      dcTensor
Type:         Package
Title:        Discrete Matrix/Tensor Decomposition
Version:      1.3.1
Date:         2025-08-25
Authors@R:    c(person("Koki", "Tsuyuzaki", role = c("aut", "cre"), email = "k.t.the-answer@hotmail.co.jp"))
Depends:      R (>= 3.4.0)
Imports:      methods, MASS, fields, rTensor, nnTensor
Suggests:     knitr, rmarkdown, testthat
Description:   Semi-Binary and Semi-Ternary Matrix Decomposition are performed based on Non-negative Matrix Factorization
License:      MIT + file LICENSE
URL:          https://github.com/rikenbit/dcTensor
VignetteBuilder: knitr
Author:       Koki Tsuyuzaki [aut, cre]
Maintainer:   Koki Tsuyuzaki <k.t.the-answer@hotmail.co.jp>
```

Index of help topics:

dcTensor-package	Discrete Matrix/Tensor Decomposition
djNMF	Discretized Joint Non-negative Matrix Factorization Algorithms (djNMF)
dNMF	Discretized Non-negative Matrix Factorization Algorithms (dNMF)
dNMTF	Discretized Non-negative Matrix Tri-Factorization Algorithms (dNMTF)
dNTD	Discretized Non-negative Tucker Decomposition Algorithms (dNTD)
dNTF	Discretized Non-negative CP Decomposition Algorithms (dNTF)
dPLS	Discretized Partial Least Squares (dPLS)
dsiNMF	Discretized Simultaneous Non-negative Matrix Factorization Algorithms (dsiNMF)

	Factorization Algorithms (dsinMF)
dSVD	Discretized Singular Value Decomposition (dSVD)
toyModel	Toy model data for using dNMF, dSVD, dsinMF, djNMF, dPLS, dNTF, and dNTD

Author(s)

Koki Tsuyuzaki [aut, cre]

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

References

Z. Zhang, T. Li, C. Ding and X. Zhang, (2007). Binary Matrix Factorization with Applications, *Seventh IEEE International Conference on Data Mining (ICDM 2007)*, 391-400

See Also[toyModel, dNMF, dSVD](#)**Examples**

```
ls("package:dcTensor")
```

djNMF	<i>Discretized Joint Non-negative Matrix Factorization Algorithms (djNMF)</i>
-------	---

Description

This function is the discretized version of nnTensor::jNMF. The input data objects are assumed to be a list containing multiple non-negative matrices ($X_1, X_2, \dots, X_K$), and decomposed to multiple matrix products $((W + V_1) H_1', (W + V_2) H_2', \dots, (W + V_K) H_K')$, where W is common across all the data matrices but each V_k or H_k ($k=1..K$) is specific in each X_k . Unlike regular jNMF, in djNMF, W , V_k , and H_k are estimated by adding binary regularization so that the values are 0 or 1 as much as possible. Likewise, W , V_k , and H_k are estimated by adding ternary regularization so that the values are 0, 1, or 2 as much as possible.

Usage

```
djNMF(X, M=NULL, pseudocount=.Machine$double.eps,
      initW=NULL, initV=NULL, initH=NULL,
      fixW=FALSE, fixV=FALSE, fixH=FALSE,
      Bin_W=1e-10, Bin_V=rep(1e-10, length=length(X)), Bin_H=rep(1e-10, length=length(X)),
      Ter_W=1e-10, Ter_V=rep(1e-10, length=length(X)), Ter_H=rep(1e-10, length=length(X)),
      L1_W=1e-10, L1_V=rep(1e-10, length=length(X)), L1_H=rep(1e-10, length=length(X)),
      L2_W=1e-10, L2_V=rep(1e-10, length=length(X)), L2_H=rep(1e-10, length=length(X)),
      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 , $\langle N \times M_k \rangle$, $k=1..K$).
M	A list containing the mask matrices (X_k , $\langle N \times M_k \rangle$, $k=1..K$). 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 , $\langle N \times J \rangle$, $k=1..K$, Default: NULL).
initH	A list containing the initial values of multiple factor matrices (H_k , $\langle M_k \times J \rangle$, $k=1..K$, Default: NULL).
fixW	Whether the factor matrix W is updated in each iteration step (Default: FALSE).
fixV	Whether the factor matrices V_k are updated in each iteration step (Default: FALSE).
fixH	Whether the factor matrices H_k are updated in each iteration step (Default: FALSE).
Bin_W	Paramter for binary (0,1) regularitation (Default: 1e-10).
Bin_V	A K-length vector containing the paramters for binary (0,1) regularitation (Default: rep(1e-10, length=length(dim(X)))).
Bin_H	A K-length vector containing the paramters for binary (0,1) regularitation (Default: rep(1e-10, length=length(dim(X)))).
Ter_W	Paramter for terary (0,1,2) regularitation (Default: 1e-10).
Ter_V	A K-length vector containing the paramters for terary (0,1,2) regularitation (Default: rep(1e-10, length=length(dim(X)))).
Ter_H	A K-length vector containing the paramters for terary (0,1,2) regularitation (Default: rep(1e-10, length=length(dim(X)))).
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	A K-length vector containing the paramters for L1 regularitation (Default: rep(1e-10, length=length(dim(X)))). This also works as small positive constant to prevent division by zero, so should be set as 0.
L1_H	A K-length vector containing the paramters for L1 regularitation (Default: rep(1e-10, length=length(dim(X)))). 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	A K-length vector containing the paramters for L2 regularitation (Default: rep(1e-10, length=length(dim(X)))).
L2_H	A K-length vector containing the paramters for L2 regularitation (Default: rep(1e-10, length=length(dim(X)))).
J	Number of low-dimension ($J < N$, M_k).

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 iteration 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 = "dsiNMF_Hard")
out <- djNMF(matdata, J=2, num.iter=2)
```

Description

This function is the discretized version of `nnTensor::NMF`. The input data X is assumed to be a non-negative matrix and decomposed to a matrix product $U V'$. Unlike regular NMF, in dNMF, U and V are estimated by adding binary regularization so that the values are 0 or 1 as much as possible. Likewise, U and V are estimated by adding ternary regularization so that the values are 0, 1, or 2 as much as possible.

Usage

```
dNMF(X, M=NULL, pseudocount=.Machine$double.eps,
      initU=NULL, initV=NULL, fixU=FALSE, fixV=FALSE,
      Bin_U=1e-10, Bin_V=1e-10, Ter_U=1e-10, Ter_V=1e-10,
      L1_U=1e-10, L1_V=1e-10, L2_U=1e-10, L2_V=1e-10, J = 3,
      algorithm = c("Frobenius", "KL", "IS", "Beta"), Beta = 2,
      thr = 1e-10, num.iter = 100,
      viz = FALSE, figdir = NULL, verbose = FALSE)
```

Arguments

<code>X</code>	The input matrix which has N-rows and M-columns.
<code>M</code>	The mask matrix which has N-rows and M-columns. If the input matrix has missing values, specify the element as 0 (otherwise 1).
<code>pseudocount</code>	The pseudo count to avoid zero division, when the element is zero (Default: Machine Epsilon).
<code>initU</code>	The initial values of factor matrix U , which has N-rows and J-columns (Default: NULL).
<code>initV</code>	The initial values of factor matrix V , which has M-rows and J-columns (Default: NULL).
<code>fixU</code>	Whether the factor matrix U is updated in each iteration step (Default: FALSE).
<code>fixV</code>	Whether the factor matrix V is updated in each iteration step (Default: FALSE).
<code>Bin_U</code>	Paramter for binary (0,1) regularitation (Default: 1e-10).
<code>Bin_V</code>	Paramter for binary (0,1) regularitation (Default: 1e-10).
<code>Ter_U</code>	Paramter for terary (0,1,2) regularitation (Default: 1e-10).
<code>Ter_V</code>	Paramter for terary (0,1,2) regularitation (Default: 1e-10).
<code>L1_U</code>	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.
<code>L1_V</code>	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.
<code>L2_U</code>	Paramter for L2 regularitation (Default: 1e-10).

L2_V	Paramter for L2 regularitation (Default: 1e-10).
J	The number of low-dimension ($J < \{N, M\}$, Default: 3)
algorithm	dNMF algorithms. "Frobenius", "KL", "IS", and "Beta" are available (Default: "Frobenius").
Beta	The parameter of Beta-divergence.
thr	When error change rate is lower than thr, the iteration is terminated (Default: 1E-10).
num.iter	The number of iteration 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 ($J < \{N, M\}$). 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.

Author(s)

Koki Tsuyuzaki

References

Z. Zhang, T. Li, C. Ding and X. Zhang, (2007). Binary Matrix Factorization with Applications, *Seventh IEEE International Conference on Data Mining (ICDM 2007)*, 391-400

Examples

```
# Test data
matdata <- toyModel(model = "dNMF")

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

dNMTF	<i>Discretized Non-negative Matrix Tri-Factorization Algorithms (dN-MTF)</i>
-------	--

Description

This function is the discretized version of nnTensor::NMTF. The input data is assumed to be non-negative matrix. dNMTF decompose the matrix to three low-dimensional factor matices.

Usage

```
dNMTF(X, M=NULL, pseudocount=.Machine$double.eps,
      initU=NULL, initS=NULL, initV=NULL,
      fixU=FALSE, fixS=FALSE, fixV=FALSE,
      Bin_U=1e-10, Bin_S=1e-10, Bin_V=1e-10,
      Ter_U=1e-10, Ter_S=1e-10, Ter_V=1e-10,
      L1_U=1e-10, L1_S=1e-10, L1_V=1e-10,
      L2_U=1e-10, L2_S=1e-10, L2_V=1e-10,
      rank = c(3, 4),
      algorithm = c("Frobenius", "KL", "IS", "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.
M	The mask matrix which has N-rows and M-columns. If the input matrix has missing values, specify the elements 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 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).
Bin_U	Paramter for binary (0,1) regularitation (Default: 1e-10).
Bin_S	Paramter for binary (0,1) regularitation (Default: 1e-10).
Bin_V	Paramter for binary (0,1) regularitation (Default: 1e-10).
Ter_U	Paramter for terary (0,1,2) regularitation (Default: 1e-10).
Ter_S	Paramter for terary (0,1,2) regularitation (Default: 1e-10).
Ter_V	Paramter for terary (0,1,2) regularitation (Default: 1e-10).
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).
rank	The number of low-dimension (J1 (< N) and J2 (< M)) (Default: c(3,4)).

algorithm	dNMTF algorithms. "Frobenius", "KL", "IS", and "Beta" are available (Default: "Frobenius").
Beta	The parameter of Beta-divergence (Default: 2, which means "Frobenius").
root	Whether square root is calculated 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 iteration 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 ($J2 < M$). rank : The number of low-dimension ($J1 < N$ and $J2 < M$). 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. 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 = "dNMF")

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

Description

This function is the discretized version of `nnTensor::NTD`. The input data X is assumed to be a non-negative tensor and decomposed to a product of a dense core tensor (S) and low-dimensional factor matrices (A_k , $k=1..K$). Unlike regular NTD, in dNTD, each A_k is estimated by adding binary regularization so that the values are 0 or 1 as much as possible. Likewise, each A_k are estimated by adding ternary regularization so that the values are 0, 1, or 2 as much as possible.

Usage

```
dNTD(X, M=NULL, pseudocount=.Machine$double.eps,
      initS=NULL, initA=NULL, fixS=FALSE, fixA=FALSE,
      Bin_A=rep(1e-10, length=length(dim(X))),
      Ter_A=rep(1e-10, length=length(dim(X))),
      L1_A=rep(1e-10, length=length(dim(X))),
      L2_A=rep(1e-10, length=length(dim(X))),
      rank = rep(3, length=length(dim(X))),
      modes = seq_along(dim(X)),
      algorithm = c("Frobenius", "KL", "IS", "Beta"),
      init = c("dNMF", "Random"),
      Beta = 2, thr = 1e-10, num.iter = 100,
      viz = FALSE,
      figdir = NULL, verbose = FALSE)
```

Arguments

<code>X</code>	K-order input tensor which has $I_1, I_2, \dots$, and I_K dimensions.
<code>M</code>	K-order mask tensor which has $I_1, I_2, \dots$, and I_K dimensions. If the mask tensor has missing values, specify the element as 0 (otherwise 1).
<code>pseudocount</code>	The pseudo count to avoid zero division, when the element is zero (Default: Machine Epsilon).
<code>initS</code>	The initial values of core tensor which has $I_1, I_2, \dots$, and I_K dimensions (Default: NULL).
<code>initA</code>	A list containing the initial values of K factor matrices (A_k , $\langle I_k \times J_k \rangle$, $k=1..K$, Default: NULL).
<code>fixS</code>	Whether the core tensor S is updated in each iteration step (Default: FALSE).
<code>fixA</code>	Whether the factor matrices A_k are updated in each iteration step (Default: FALSE).
<code>Bin_A</code>	A K -length vector containing the parameters for binary (0,1) regularization (Default: <code>rep(1e-10, length=length(dim(X)))</code>).
<code>Ter_A</code>	A K -length vector containing the parameters for ternary (0,1,2) regularization (Default: <code>rep(1e-10, length=length(dim(X)))</code>).

L1_A	A K-length vector containing the paramters for L1 regularitation (Default: rep(1e-10, length=length(dim(X)))). This also works as small positive constant to prevent division by zero, so should be set as 0.
L2_A	A K-length vector containing the paramters for L2 regularitation (Default: rep(1e-10, length=length(dim(X)))).
rank	The number of low-dimension in each mode (Default: 3 for each mode).
modes	The vector of the modes on which to perform the decomposition (Default: 1:K <all modes>).
algorithm	dNTD algorithms. "Frobenius", "KL", "IS", and "Beta" are available (Default: "Frobenius").
init	The initialization algorithms. "NMF", "ALS", and "Random" are available (Default: "NMF").
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 iteration 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 : K-order tensor object, which is defined as S4 class of rTensor package. A : A list containing K 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). Nonnegative 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 Huy Phan et. al. (2011). Extended HALS algorithm for nonnegative Tucker decomposition and its applications for multiway analysis and classification. *Neurocomputing*

See Also

[nnTensor::plotTensor3D](#)

Examples

```

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

```

dNTF

Discretized Non-negative CP Decomposition Algorithms (dNTF)

Description

This function is the discretized version of `nnTensor::NTF`. The input data X is assumed to be a non-negative tensor and decomposed to a product of a diagonal core tensor (S) and low-dimensional factor matrices (A_k , $k=1..K$). Unlike regular NTF, in dNTF, each A_k is estimated by adding binary regularization so that the values are 0 or 1 as much as possible. Likewise, each A_k are estimated by adding ternary regularization so that the values are 0, 1, or 2 as much as possible.

Usage

```

dNTF(X, M=NULL, pseudocount=.Machine$double.eps,
  initA=NULL, fixA=FALSE,
  Bin_A=rep(1e-10, length=length(dim(X))),
  Ter_A=rep(1e-10, length=length(dim(X))),
  L1_A=rep(1e-10, length=length(dim(X))),
  L2_A=rep(1e-10, length=length(dim(X))),
  rank = 3,
  algorithm = c("Frobenius", "KL", "IS", "Beta"),
  init = c("dNMF", "Random"),
  Beta = 2, thr = 1e-10, num.iter = 100, viz = FALSE, figdir = NULL,
  verbose = FALSE)

```

Arguments

<code>X</code>	K-order input tensor which has $I_1, I_2, \dots$, and I_K dimensions.
<code>M</code>	K-order mask tensor which has $I_1, I_2, \dots$, and I_K dimensions. If the mask tensor has missing values, specify the element as 0 (otherwise 1).
<code>pseudocount</code>	The pseudo count to avoid zero division, when the element is zero (Default: Machine Epsilon).
<code>initA</code>	A list containing the initial values of K factor matrices (A_k , $\langle I_k \times J_k \rangle$, $k=1..K$, Default: NULL).
<code>fixA</code>	Whether the factor matrices A_k are updated in each iteration step (Default: FALSE).
<code>Bin_A</code>	A K -length vector containing the paramters for binary (0,1) regularitaton (Default: <code>rep(1e-10, length=length(dim(X)))</code>).
<code>Ter_A</code>	A K -length vector containing the paramters for terary (0,1,2) regularitaton (Default: <code>rep(1e-10, length=length(dim(X)))</code>).

L1_A	A K-length vector containing the paramters for L1 regularitation (Default: rep(1e-10, length=length(dim(X)))). This also works as small positive constant to prevent division by zero, so should be set as 0.
L2_A	A K-length vector containing the paramters for L2 regularitation (Default: rep(1e-10, length=length(dim(X)))).
rank	The number of low-dimension in each mode (Default: 3).
algorithm	dNTF algorithms. "Frobenius", "KL", "IS", and "Beta" are available (Default: "Frobenius").
init	The initialization algorithms. "dNMF", and "Random" are available (Default: "dNMF").
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 iteration 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 : K-order tensor object, which is defined as S4 class of rTensor package. A : A list containing K 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

[nnTensor::plotTensor3D](#)

Examples

```

tensordata <- toyModel(model = "dNTF")
out <- dNTF(tensordata, rank=3, num.iter=2)

```

dPLS

Discretized Partial Least Squares (dPLS)

Description

This function is the discretized version of PLS. The input data objects are assumed to be a list containing multiple matrices ($X_1, X_2, \dots, X_K$), and decomposed to multiple matrix products ($U_1 V_1', U_2 V_2', \dots, U_K V_K'$), where each U_k and V_k ($k=1..K$) is specific in each X_k . Unlike regular PLS, in dPLS, U_k and V_k are estimated by adding ternary regularization so that the values are -1, 0, or 1 as much as possible.

Usage

```

dPLS(X, M=NULL, pseudocount=.Machine$double.eps,
      initV=NULL, fixV=FALSE, Ter_V=1e-10,
      L1_V=1e-10, L2_V=1e-10, eta=1e+10, J = 3,
      thr = 1e-10, 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).
initV	The initial values of factor matrix V, which has M-rows and J-columns (Default: NULL).
fixV	Whether the factor matrix V is updated in each iteration step (Default: FALSE).
Ter_V	Paramter for ternary (-1,0,1) regularization (Default: 1e-10).
L1_V	Paramter for L1 regularization (Default: 1e-10). This also works as small positive constant to prevent division by zero, so should be set as 0.
L2_V	Paramter for L2 regularization (Default: 1e-10).
eta	Stepsize of gradient descent algorithm (Default: 1e+10).
J	The number of low-dimension ($J < \{N, M\}$, Default: 3)
thr	When error change rate is lower than thr, the iteration is terminated (Default: 1E-10).
num.iter	The number of iteration 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 ($J < \{N, M\}$). 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.

Author(s)

Koki Tsuyuzaki

Examples

```
# Test data
matdata <- toyModel(model = "dPLS_Easy")

# Simple usage
out <- dPLS(matdata, J=2, num.iter=2)
```

dsiNMF

Discretized Simultaneous Non-negative Matrix Factorization Algorithms (dsiNMF)

Description

This function is the discretized version of nnTensor::siNMF. The input data objects are assumed to be a list containing multiple non-negative matrices ($X_1, X_2, \dots, X_K$), and decomposed to multiple matrix products ($W H_1', W H_2', \dots, W H_K'$), where W is common across all the data matrices but each H_k ($k=1..K$) is specific in each X_k . Unlike regular siNMF, in dsiNMF, W and H_k are estimated by adding binary regularization so that the values are 0 or 1 as much as possible. Likewise, W and H_k are estimated by adding ternary regularization so that the values are 0, 1, or 2 as much as possible.

Usage

```
dsiNMF(X, M=NULL, pseudocount=.Machine$double.eps,
       initW=NULL, initH=NULL,
       fixW=FALSE, fixH=FALSE,
       Bin_W=1e-10, Bin_H=rep(1e-10, length=length(X)),
       Ter_W=1e-10, Ter_H=rep(1e-10, length=length(X)),
       L1_W=1e-10, L1_H=rep(1e-10, length=length(X)),
       L2_W=1e-10, L2_H=rep(1e-10, length=length(X)),
       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

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

Value

W : A matrix which has N-rows and J-columns ($J < N, M_k$). H : A list which has multiple elements containing M_k -rows and J-columns matrix ($J < N, M_k$). 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 = "dsiNMF_Easy")
out <- dsiNMF(matdata, J=2, num.iter=2)
```

dSVD

Discretized Singular Value Decomposition (dSVD)

Description

This function is the discretized version of SVD. The input data X is decomposed to a matrix product $U V^T$. Unlike regular SVD, in dSVD, U and V are estimated by adding binary regularization so that the values are 0 or 1 as much as possible. Likewise, U and V are estimated by adding ternary regularization so that the values are -1, 0, or 1 as much as possible.

Usage

```
dSVD(X, M=NULL, pseudocount=.Machine$double.eps,
      initU=NULL, initV=NULL, fixU=FALSE, fixV=FALSE,
      Ter_U=1e-10, L1_U=1e-10, L2_U=1e-10, eta=1e+10, J = 3,
      thr = 1e-10, num.iter = 100,
      viz = FALSE, figdir = NULL, verbose = FALSE)
```

Arguments

<code>X</code>	The input matrix which has N-rows and M-columns.
<code>M</code>	The mask matrix which has N-rows and M-columns. If the input matrix has missing values, specify the element as 0 (otherwise 1).
<code>pseudocount</code>	The pseudo count to avoid zero division, when the element is zero (Default: Machine Epsilon).
<code>initU</code>	The initial values of factor matrix U, which has N-rows and J-columns (Default: NULL).
<code>initV</code>	The initial values of factor matrix V, which has M-rows and J-columns (Default: NULL).
<code>fixU</code>	Whether the factor matrix U is updated in each iteration step (Default: FALSE).
<code>fixV</code>	Whether the factor matrix V is updated in each iteration step (Default: FALSE).
<code>Ter_U</code>	Paramter for terary (-1,0,1) regularitation (Default: 1e-10).
<code>L1_U</code>	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.
<code>L2_U</code>	Paramter for L2 regularitation (Default: 1e-10).
<code>eta</code>	Stepsize of gradient descent algorithm (Default: 1e+10).
<code>J</code>	The number of low-dimension ($J < \{N, M\}$, Default: 3)
<code>thr</code>	When error change rate is lower than thr, the iteration is terminated (Default: 1E-10).
<code>num.iter</code>	The number of iteration step (Default: 100).
<code>viz</code>	If viz == TRUE, internal reconstructed matrix can be visualized.
<code>figdir</code>	The directory for saving the figure, when viz == TRUE.
<code>verbose</code>	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 ($J < \{N, M\}$). 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.

Author(s)

Koki Tsuyuzaki

Examples

```
# Test data
matdata <- toyModel(model = "dSVD")

# Simple usage
out <- dSVD(matdata, J=2, num.iter=2)
```

toyModel	<i>Toy model data for using dNMF, dSVD, dsiNMF, djNMF, dPLS, dNTF, and dNTD</i>
----------	---

Description

The data is used to confirm that the algorithm are properly working.

Usage

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

Arguments

model	Single character string is specified. "dNMF", "dSVD", "dsiNMF_Easy", "dsiNMF_Hard", "dPLS_Easy", "dPLS_Hard", "dNTF", and "dNTD" are available (Default: "dNMF").
seeds	Random number for setting set.seeds in the function (Default: 123).

Value

If model is specified as "dNMF" or "dSVD" a matrix is generated. If model is specified as "dsiNMF_Easy", "dsiNMF_Hard", "dPLS_Easy", or "dPLS_Hard" three matrices are generated. Otherwise, a tensor is generated.

Author(s)

Koki Tsuyuzaki

See Also

[dNMF](#), [dSVD](#)

Examples

```
matdata <- toyModel(model = "dNMF", seeds=123)
```

Index

* **methods**

- djNMF, [3](#)
- dNMF, [6](#)
- dNMTF, [7](#)
- dNTD, [10](#)
- dNTF, [12](#)
- dPLS, [14](#)
- dsiNMF, [15](#)
- dSVD, [17](#)
- toyModel, [19](#)

* **package**

- dcTensor-package, [2](#)

dcTensor (dcTensor-package), [2](#)

dcTensor-package, [2](#)

djNMF, [3](#)

dNMF, [3](#), [6](#), [19](#)

dNMTF, [7](#)

dNTD, [10](#)

dNTF, [12](#)

dPLS, [14](#)

dsiNMF, [15](#)

dSVD, [3](#), [17](#), [19](#)

nnTensor::plotTensor3D, [11](#), [13](#)

toyModel, [3](#), [19](#)