Package 'tensorApp'

February 23, 2020

Type Package

Title tensorApp

Version 0.1.0

Author Xu Liu [aut,cre].

Maintainer Xu Liu <liu.xu@sufe.edu.cn>

Description
High-order SVD approximation of a tensor by Tucker or CP decomposition and rank selection.

License GPL (>= 2)

Imports Rcpp (>= 0.11.15), RcppEigen (>= 0.3.2.3.0)

LinkingTo Rcpp, RcppEigen

RoxygenNote 6.0.1

NeedsCompilation yes

Repository github

URL https://github.com/xliusufe/tensorApp

Encoding UTF-8

R topics documented:

tensorApp-package *High-order SVD approximation of a tensor* Y *by Tucker or CP decomposition and selection of ranks*

Description

High-order SVD approximation of a tensor Y by Tucker decomposition or CANDECOMP/PARAFAC (CP) decomposition and selection of ranks. Alternating Least Squares algorithm is applied to Tucker decomposition, and both Alternating Least Squares algorithm or Tensor Power Method are applied to CP decomposition. This package provides several generator functions, which generate low-rank tensor or low-rank semi-symmetric tensor.

Details

High-order SVD approximation of a tensor Y by Tucker decomposition or CANDECOMP/PARAFAC (CP) decomposition and selection of ranks.

Author(s)

Xu Liu

Maintainer: Xu Liu <liu.xu@sufe.edu.cn>

References

Allen, G., (2012). Sparse higher-order principal components analysis, in: International Conference on Artificial Intelligence and Statistics, pp. 27-36.

De Lathauwer, L., De Moor, B., Vandewalle, J., (2000). A multilinear singular value decomposition. SIAM Journal on Matrix Analysis and Applications, **21**, 1253-1278.

De Lathauwer, L., De Moor, B., Vandewalle, J., (2000). On the best rank-1 and rank-$(r_1, r_2, \cdots, r_n)$ approximation of higher-order tensors. SIAM Journal on Matrix Analysis and Applications, **21**, 1324-1342.

Kolda, T.G., (2001). Orthogonal tensor decompositions. SIAM Journal on Matrix Analysis and Applications, **23**, 243-255.

Kolda, T., Bader, B., (2009). Tensor decompositions and applications. SIAM Review, **51**, 455-500.

cpals *High-order SVD approximation of a tensor* Y *by CP decomposition*

Description

High-order SVD approximation of a tensor Y by CANDECOMP/PARAFAC (CP) decomposition with rank preseted or to be selected. The Alternating Least Squares (als) algorithm is applied.

Usage

```
cpals(Y=NULL, d0=NULL, dims=NULL, dr=10, isfixr=FALSE,
                               D0=NULL, eps=1e-4, max_step=50, thresh=1e-6)
```

Arguments

Y	An array with dimension dims, or a $n_{d0} \times N/n_{d0}$ numeric matrix of responses that is the mode d0-unfolding of tensor in $\mathcal{R}^{n_1 \times \cdots \times n_d}$, where $N = n_1 \times \cdots \times n_d$.
d0	d0 is the mode. Y is the mode-d0 unfolding of the tensor. d0 can be NULL (the default) if Y is an array with dimension dims.
dims	The size of tensor Y, which is a d-vector $(n_1, \cdots, n_d)$. dims can be NULL (the default) if Y is an array with dimension dims. If the length of dims is 2, it is the ordinary SVD decomposition of a matrix.
dr	The user-specified rank for CP decomposition. Default is 10.
isfixr	A logical value indicating whether the rank is fixed. The rank is selected automatically if it is FALSE. Default is FALSE.
D0	A user-specified list of initial matrices of $U_1, U_2, \cdots, U_d$ and core tensor S, D0=list$(U_1 = U_1, \cdots, U_d = U_d, S = S)$. By default, initial matrices are provided by random.
eps	Convergence threshhold. The algorithm iterates until the relative change in any coefficient is less than eps. Default is 1e-4.
max_step	Maximum number of iterations. Default is 50.
thresh	Convergence threshhold in the outer loop. The algorithm iterates until the relative change in any coefficient is less than eps. Default is 1e-6.

Details

This function gives a $n_{d0} \times N/n_{d0}$ matrix, which is the mode-d0 unfolding, and approximates Y.

Value

Tnew	Approximation of Y.
Tn	A list of estimated matrices of $U_1, U_2, \cdots, U_d$ and core tensor S, Tn=list$(U_1 = U_1, \cdots, U_d = U_d, S = S)$.
ranks	The ranks of estimated tensor Tnew. It is an integer.

See Also

hosvd_dr

Examples

```
dims <- c(8,8,10,10,6)
N <- length(dims)
ranks <- rep(2,N)
S0 <- matrix(runif(prod(ranks),3,7),ranks[N])
T1 <- matrix(rnorm(dims[1]*ranks[1]),nrow = dims[1])
tmp <- qr.Q(qr(T1))
for(k in 2:(N-1)){
  T1 <- matrix(rnorm(dims[k]*ranks[k]),nrow = dims[k])
  tmp <- kronecker(qr.Q(qr(T1)),tmp)
}
T1 <- matrix(rnorm(dims[N]*ranks[N]),nrow = dims[N])
U <- qr.Q(qr(T1))
```

```
Y <- U%*%S0%*%t(tmp)

fit <- cpals(Y,N,dims)
Tnew <- fit$Tnew
ranks1 <- fit$ranks
U1 <- fit$Tn[[1]]
TNew1 <- ttu(Tnew,N,1,dims)
```

---

cpsym2 *High-order SVD approximation of a tensor* Y *by CP decomposition*

Description

High-order SVD approximation of a semi-symmetric tensor Y by CANDECOMP/PARAFAC (CP) decomposition with rank preseted or to be selected. The Tensor Power Method is applied. The semi-symmetric tensor means that both mode-r1 and mode-r2 unfoldings are equal, that is, $Y_{(r1)} = Y_{(r2)}$. For semi-symmetric tensor approximation, the r1th and r2th dimensions must be smaller than others.

Usage

```
cpsym2(Y=NULL, r1=1, r2=2, d0=NULL, dims=NULL, dr=10, D0=NULL, isfixr=FALSE,
                            isOrth=FALSE, eps=1e-4, max_step=50, thresh=1e-6)
```

Arguments

Y	An array with dimension dims, or a $n_{d0} \times N/n_{d0}$ numeric matrix of responses that is the mode d0-unfolding of tensor in $\mathcal{R}^{n_1 \times \cdots \times n_d}$, where $N = n_1 \times \cdots \times n_d$.
r1	Both r1 and r2 are the user-specified modes, which means that both mode-r1 and mode-r2 unfoldings are equal. Default is r1 = 1 and r2 = 2.
r2	Both r1 and r2 are the user-specified modes, which means that both mode-r1 and mode-r2 unfoldings are equal. Default is r1 = 1 and r2 = 2.
d0	d0 is the mode. Y is the mode-d0 unfolding of the tensor. d0 can be NULL (the default) if Y is an array with dimension dims.
dims	The size of tensor Y, which is a d-vector $(n_1, \cdots, n_d)$. dims can be NULL (the default) if Y is an array with dimension dims. If the length of dims is 2, it is the ordinary SVD decomposition of a matrix.
dr	The user-specified rank for CP decomposition. Default is 10.
D0	A user-specified list of initial matrices of $U_1, U_2, \cdots, U_d$ and core tensor S, D0=list$(U_1 = U_1, \cdots, U_d = U_d, S = S)$. By default, initial matrices are provided by random.
isfixr	A logical value indicating whether the rank is fixed. The rank is selected automatically if it is FALSE. Default is FALSE.
isOrth	A logical value indicating whether it outputs orthonognal PCs if CP decomposition is used. Default is FALSE.
eps	Convergence threshhold. The algorithm iterates until the relative change in any coefficient is less than eps. Default is 1e-4.
max_step	Maximum number of iterations. Default is 50.
thresh	Convergence threshhold in the outer loop. The algorithm iterates until the relative change in any coefficient is less than eps. Default is 1e-6.

Details

This function gives a $n_{d0} \times N/n_{d0}$ matrix, which is the mode-d0 unfolding, and approximates Y.

Value

Tnew	Approximation of Y.
Tn	A list of estimated matrices of $U_1, U_2, \cdots, U_d$ and core tensor S, Tn=list($U_1 = U_1, \cdots, U_d = U_d, S = S$).
ranks	The ranks of estimated tensor Tnew. It is an integer.

See Also

hosvd_dr, tdsym2

Examples

```
dims <- c(6,6,8,7,7)
N <- length(dims)
ranks <- rep(2,N)
dm <- prod(ranks)
r1 <- 1
r2 <- 2
S1 <- matrix(runif(dm,3,7),nrow = ranks[r1])
S2 <- ttu(S1,r1,r2,ranks)
S1 <- (S1+S2)/2
S0 <- ttu(S1,r1,N,ranks)

T1 <- matrix(rnorm(dims[1]*ranks[1]),nrow = dims[1])
tmp <- qr.Q(qr(T1))
Uj <- kronecker(tmp,tmp)
for(k in 3:(N-1)){
  T1 <- matrix(rnorm(dims[k]*ranks[k]),nrow = dims[k])
  tmp <- qr.Q(qr(T1))
  Uj <- kronecker(tmp,Uj)
}
T1 <- matrix(rnorm(dims[N]*ranks[N]),nrow = dims[N])
U <- qr.Q(qr(T1))
Y <- U%*%S0%*%t(Uj)

fit <- cpsym2(Y,r1=1,r2=2,d0=N,dims=dims)
Tnew <- fit$Tnew
ranks1 <- fit$ranks
U1 <- fit$Tn[[r1]]
U2 <- fit$Tn[[r2]]
TNew1 <- ttu(Tnew,N,r1,dims)
TNew2 <- ttu(Tnew,N,r2,dims)
```

gtcp *Generate a low-rank tensor characterizing the form of CP decomposition*

Description

This function generates a low-rank tensor characterizing the form of CP decomposition with dimension dims and factors lambda.

Usage

```
gtcp(dims, lambda=NULL, d0=NULL, dr=NULL, seed_id=2)
```

Arguments

dims	The size of the tensor, which is a vector $(n_1, \cdots, n_d)$. dims must be specified.
lambda	The factors of CP decomposition. It is an vector. Factors lambda will be given randomly if it is NULL.
d0	d0 is the mode. The output tensor is the mode-d0 unfolding of the tensor. d0 can be NULL (the default) if the output tensor is an array with dimension dims.
dr	The user-specified rank. Default is 10.
seed_id	A positive integer, the seed for generating the random numbers. Default is 2.

Details

This function generates a low-rank tensor characterizing the form of CP decomposition with dimension dims and factors lambda.

Value

Dn	the output mode-d0-unfolding, $D_{(d_0)}$. Or an array with dimesion dims if d0 is NULL.

See Also

gtcpsem, gtt

Examples

```
dims <- c(8,8,10,10,6)
N <- length(dims)
lambda <- seq(6,1,by=-1)
dr <- 5

T1 <- gtcp(dims=dims,lambda=lambda,d0=1,dr=dr)
T2 <- ttu(T1,1,2,dims)
```

gtcpsem *Generate a low-rank semi-symmetric tensor characterizing the form of CP decomposition*

Description

This function generates a low-rank semi-symmetric tensor characterizing the form of CANDECOMP/PARAFAC (CP) decomposition with the dimension `dims` and factors `lambda`. The semi-symmetric tensor means that both mode-`r1` and mode-`r2` unfoldings are equal, that is, $T_{(r1)} = T_{(r2)}$ for the output tensor T.

Usage

```
gtcpsem(dims, lambda=NULL, r1=1, r2=2, d0=NULL, dr=NULL, seed_id=2)
```

Arguments

`dims`	The size of the tensor, which is a vector $(n_1, \cdots, n_d)$. `dims` must be specified.
`lambda`	The factors of CP decomposition. It is an vector. Factors `lambda` will be given randomly if it is `NULL`.
`r1`	Both `r1` and `r2` are the user-specified modes, which means that both mode-`r1` and mode-`r2` unfoldings are equal. Default is `r1 = 1` and `r2 = 2`.
`r2`	Both `r1` and `r2` are the user-specified modes, which means that both mode-`r1` and mode-`r2` unfoldings are equal. Default is `r1 = 1` and `r2 = 2`.
`d0`	`d0` is the mode. The output tensor is the mode-`d0` unfolding of the tensor. `d0` can be NULL (the default) if the output tensor is an array with dimension `dims`.
`dr`	The user-specified rank. Default is `10`.
`seed_id`	A positive integer, the seed for generating the random numbers. Default is 2.

Details

This function generates a low-rank semi-symmetric tensor characterizing the form of CP decomposition with dimension `dims` and factors `lambda`.

Value

`Dn`	the output mode-`d0`-unfolding, $D_{(d_0)}$. Or an array with dimesion `dims` if `d0` is `NULL`.

See Also

gtcp, gttsem

Examples

```
dims <- c(8,6,10,6,7)
N <- length(dims)
lambda <- seq(6,1,by=-1)
r1 <- 2
r2 <- 4
```

```
dr <- 5

T1 <- gtcpsem(dims=dims,lambda=lambda,r1=r1,r2=r2,d0=r1,dr=dr)
T2 <- ttu(T1,r1,r2,dims)
```

---

gtt *Generate a low-rank tensor characterizing the form of Tucker decomposition*

Description

This function generates a low-rank tensor characterizing the form of Tucker decomposition with dimension dims and core tensor S.

Usage

```
gtt(dims, S=NULL, d0=NULL, ranks=NULL, seed_id=2)
```

Arguments

dims	The size of the tensor, which is a vector $(n_1, \cdots, n_d)$. dims must be specified.
S	The core tensor. An array with dimension dims, or a mode-d1-unfolding of core tensor with size $n_1 \times \cdots \times n_d$. Core tensor S will be given randomly if it is NULL.
d0	d0 is the mode. The output tensor is the mode-d0 unfolding of the tensor. d0 can be NULL (the default) if the output tensor is an array with dimension dims.
ranks	The user-specified ranks. It is a vector with length d. If ranks is NULL (the default), this function outputs a tensor without low-rank.
seed_id	A positive integer, the seed for generating the random numbers. Default is 2.

Details

This function generates a low-rank tensor characterizing the form of Tucker decomposition with dimension dims and core tensor S.

Value

Dn	the output mode-d0-unfolding, $D_{(d_0)}$. Or an array with dimesion dims if d0 is NULL.

See Also

gtcp, gttsem

Examples

```
dims <- c(8,8,10,10,6)
N <- length(dims)
ranks <- rep(2,N)

T1 = gtt(dims=dims,d0=1,ranks=ranks)
T2 <- ttu(T1,1,2,dims)
```

gttsem *Generate a low-rank semi-symmetric tensor characterizing the form of Tucker decomposition*

Description

This function generates a low-rank semi-symmetric tensor characterizing the form of Tucker decomposition with dimension `dims` and core tensor `S`. The semi-symmetric tensor means that both mode-`r1` and mode-`r2` unfoldings are equal, hat is, $T_{(r1)} = T_{(r2)}$ for the output tensor T, where the absolute difference of `r1` and `r2` is restricted to no more than 3.

Usage

```
gttsem(dims, S=NULL, r1=1, r2=2, d0=NULL, ranks=NULL, seed_id=2)
```

Arguments

`dims`	The size of the tensor, which is a vector $(n_1, \cdots, n_d)$. `dims` must be specified.
`S`	The core tensor. An array with dimension `dims`, or a mode-`d1`-unfolding of core tensor with size $n_1 \times \cdots \times n_d$. Core tensor `S` will be given randomly if it is `NULL`.
`r1`	Both `r1` and `r2` are the user-specified modes, which means that both mode-`r1` and mode-`r2` unfoldings are equal. Default is `r1 = 1` and `r2 = 2`.
`r2`	Both `r1` and `r2` are the user-specified modes, which means that both mode-`r1` and mode-`r2` unfoldings are equal. Default is `r1 = 1` and `r2 = 2`.
`d0`	`d0` is the mode. The output tensor is the mode-`d0` unfolding of the tensor. `d0` can be NULL (the default) if the output tensor is an array with dimension `dims`.
`ranks`	The user-specified ranks. It is a vector with length d. If `ranks` is `NULL` (the default), this function outputs a semi-symmetric tensor without low-rank.
`seed_id`	A positive integer, the seed for generating the random numbers. Default is 2.

Details

This function generates a low-rank semi-symmetric tensor characterizing the form of Tucker decomposition with dimension `dims` and core tensor `S`.

Value

`Dn`	the output mode-`d0`-unfolding, $D_{(d_0)}$. Or an array with dimesion `dims` if `d0` is `NULL`.

See Also

gtt, gtcpsem

Examples

```
dims <- c(8,6,8,6,7)
N <- length(dims)
ranks <- rep(2,N)
r1 <- 2
r2 <- 4

T1 <- gttsem(dims=dims,r1=r1,r2=r2,d0=r1,ranks=ranks)
T2 <- ttu(T1,r1,r2,dims)
```

`hosvd` *High-order SVD approximation of a tensor* `Y` *by Tucker or CP decomposition*

Description

High-order SVD approximation of a tensor `Y` by Tucker decomposition or CANDECOMP/PARAFAC (CP) decomposition with preseted rank. Alternating Least Squares algorithm is applied to Tucker decomposition, and Tensor Power Method is applied to CP decomposition.

Usage

```
hosvd(Y=NULL, d0=NULL, dims=NULL, isCP=TRUE, ranks=NULL, dr=20,
                                  D0=NULL,isOrth=FALSE, eps=1e-6, max_step=100)
```

Arguments

`Y`	An array with dimension `dims`, or a $n_{d0} \times N/n_{d0}$ numeric matrix of responses that is the mode `d0`-unfolding of tensor in $\mathcal{R}^{n_1 \times \cdots \times n_d}$, where $N = n_1 \times \cdots \times n_d$.
`d0`	`d0` is the mode. `Y` is the mode-`d0` unfolding of the tensor. `d0` can be NULL (the default) if `Y` is an array with dimension `dims`.
`dims`	The size of tensor `Y`, which is a d-vector $(n_1, \cdots, n_d)$. `dims` can be NULL (the default) if `Y` is an array with dimension `dims`. If the length of `dims` is 2, it is the ordinary SVD decomposition of a matrix.
`isCP`	A logical value indicating whether CP decomposition will be used. Default is `TRUE`.
`ranks`	The user-specified ranks. It is a vector with length d. Default is $(2, \cdots, 2)$.
`dr`	The user-specified rank for CP decomposition. It is useless if Tucker decomposition is used. Default is 20.
`D0`	A user-specified list of initial matrices of $U_1, U_2, \cdots, U_d$ and the mode-$d0$ unfolding $S_{(d0)}$ of the core tensor S, `D0=list`$(U_1 = U_1, \cdots, U_d = U_d, S = S_{(d0)})$. By default, initial matrices are provided randomly.
`isOrth`	A logical value indicating whether it outputs orthonognal PCs if CP decomposition is used. Default is `FALSE`.
`eps`	Convergence threshhold. The algorithm iterates until the relative change in any coefficient is less than `eps`. Default is `1e-6`.
`max_step`	Maximum number of iterations. Default is 100.

Details

This function gives a $n_{d0} \times N/n_{d0}$ matrix, which is the mode-d0 unfolding, and approximates Y.

Value

Tnew	Approximation of Y.
Tn	A list of estimated matrices of $U_1, U_2, \cdots, U_d$ and the mode-$d0$ unfolding $S_{(d0)}$ of the core tensor S, Tn=list$(U_1 = U_1, \cdots, U_d = U_d, S = S_{(d0)})$.
ranks	The ranks of estimated tensor Tnew. It is a vector with the same length as dims if Tucker decomposition is used, or an integer if CP decomposition is used.

See Also

hosvd_dr

Examples

```
# Example 1
dims <- c(8,8,10,10,6)
N <- length(dims)
ranks <- rep(2,N)
S0 <- matrix(runif(prod(ranks),3,7),ranks[N])
T1 <- matrix(rnorm(dims[1]*ranks[1]),nrow = dims[1])
tmp <- qr.Q(qr(T1))
for(k in 2:(N-1)){
  T1 <- matrix(rnorm(dims[k]*ranks[k]),nrow = dims[k])
  tmp <- kronecker(qr.Q(qr(T1)),tmp)
}
T1 <- matrix(rnorm(dims[N]*ranks[N]),nrow = dims[N])
U <- qr.Q(qr(T1))
Y <- U%*%S0%*%t(tmp)

fit <- hosvd(Y,N,dims,isCP=TRUE)
Tnew <- fit$Tnew
ranks1 <- fit$ranks
lambda <- fit$Tn[[N+1]]
U1 <- fit$Tn[[1]]
TNew1 <- ttu(Tnew,N,1,dims)


# Example 2
library(png)
bat = readPNG(system.file("data", "bat.png", package="tensorApp"))
writePNG(bat,target = "bat.png")

Tn = hosvd(bat,dr=20,dims=dim(bat))
writePNG(Tn$Tnew,target = "batCP.png")
Tn = hosvd(bat,dr=50,dims=dim(bat),isCP=F,ranks = c(20,20,3))
writePNG(Tn$Tnew,target = "batTucker.png")

# Example 3
img = readPNG(system.file("data", "Rlogo.png", package="tensorApp"))
writePNG(img,target = "Rlogo.png")
```

```
Tn = hosvd(img,dr=20,dims=dim(img))
writePNG(Tn$Tnew,target = "RlogoCP.png")
Tn = hosvd(img,dr=20,dims=dim(img),isCP=F,ranks = c(20,20,4))
writePNG(Tn$Tnew,target = "RlogoTucker.png")


# Example 4
SarsCov2 = readPNG(system.file("data", "SarsCov2.png", package="tensorApp"))
writePNG(SarsCov2,target = "SarsCov2.png")

Tn = hosvd(SarsCov2,dr=20,dims=dim(SarsCov2))
writePNG(Tn$Tnew,target = "SarsCov2CP.png")
Tn = hosvd(SarsCov2,dr=50,dims=dim(SarsCov2),isCP=F,ranks = c(20,20,3))
writePNG(Tn$Tnew,target = "SarsCov2Tucker.png")
```

`hosvd_dr` *High-order SVD approximation of a tensor* `Y` *by Tucker or CP decomposition and selection of ranks*

Description

High-order SVD approximation of a tensor `Y` by Tucker decomposition or CANDECOMP/PARAFAC (CP) decomposition and selection of ranks. Alternating Least Squares algorithm is applied to Tucker decomposition, and Tensor Power Method is applied to CP decomposition.

Usage

```
hosvd_dr(Y=NULL, d0=NULL, dims=NULL, isCP=TRUE, ranks=NULL, dr=100,
                        D0=NULL, isOrth=FALSE, eps=1e-6, max_step=100, thresh=1e-6)
```

Arguments

`Y`	An array with dimension `dims`, or a $n_{d0} \times N/n_{d0}$ numeric matrix of responses that is the mode `d0`-unfolding of tensor in $\mathcal{R}^{n_1 \times \cdots \times n_d}$, where $N = n_1 \times \cdots \times n_d$.
`d0`	`d0` is the mode. `Y` is the mode-`d0` unfolding of the tensor. `d0` can be NULL (the default) if `Y` is an array with dimension `dims`.
`dims`	The size of tensor `Y`, which is a d-vector $(n_1, \cdots, n_d)$. `dims` can be NULL (the default) if `Y` is an array with dimension `dims`. If the length of `dims` is 2, it is the ordinary SVD decomposition of a matrix.
`isCP`	A logical value indicating whether CP decomposition will be used. Default is `TRUE`.
`ranks`	The user-specified ranks. It is a vector with length `d`. Default is $(2, \cdots, 2)$.
`dr`	The user-specified rank for CP decomposition. The maximum rank `dm=min(dr, max(ranks))` if Tucker decomposition is used. Default is 100.
`D0`	A user-specified list of initial matrices of $U_1, U_2, \cdots, U_d$ and the mode-$d0$ unfolding $S_{(d0)}$ of the core tensor S, `D0=list`$(U_1 = U_1, \cdots, U_d = U_d, S = S_{(d0)})$. By default, initial matrices are provided randomly.
`isOrth`	A logical value indicating whether it outputs orthonognal PCs if CP decomposition is used. Default is `FALSE`.

eps	Convergence threshhold in the inner loop. The algorithm iterates until the relative change in any coefficient is less than eps. Default is 1e-6.
max_step	Maximum number of iterations. Default is 100.
thresh	Convergence threshhold in the outer loop. The algorithm iterates until the relative change in any coefficient is less than thresh. Default is 1e-6.

Details

This function gives a $n_{d0} \times N/n_{d0}$ matrix, which is the mode-$d0$ unfolding, and approximates Y.

Value

Tnew	Approximation of Y.
Tn	A list of estimated matrices of $U_1, U_2, \cdots, U_d$ and the mode-$d0$ unfolding $S_{(d0)}$ of the core tensor S, Tn=list$(U_1 = U_1, \cdots, U_d = U_d, S = S_{(d0)})$.
ranks	The ranks of estimated tensor Tnew. It is a vector with the same length as dims if Tucker decomposition is used, or an integer if CP decomposition is used.

See Also

hosvd

Examples

```
dims <- c(8,8,10,10,6)
N <- length(dims)
ranks <- rep(2,N)
S0 <- matrix(runif(prod(ranks),3,7),ranks[N])
T1 <- matrix(rnorm(dims[1]*ranks[1]),nrow = dims[1])
tmp <- qr.Q(qr(T1))
for(k in 2:(N-1)){
  T1 <- matrix(rnorm(dims[k]*ranks[k]),nrow = dims[k])
  tmp <- kronecker(qr.Q(qr(T1)),tmp)
}
T1 <- matrix(rnorm(dims[N]*ranks[N]),nrow = dims[N])
U <- qr.Q(qr(T1))
Y <- U%*%S0%*%t(tmp)

fit_dr <- hosvd_dr(Y,N,dims,isCP=TRUE)
Tnew <- fit_dr$Tnew
ranks1 <- fit_dr$ranks
lambda <- fit_dr$Tn[[N+1]]
U1 <- fit_dr$Tn[[1]]
TNew1 <- ttu(Tnew,N,1,dims)
```

spcacp *High-order SVD approximation of a tensor* Y *by sparse CP decomposition*

Description

High-order SVD approximation of a tensor Y by sparse CANDECOMP/PARAFAC (CP) decomposition with rank preseted or to be selected. The Alternating Least Squares (als) algorithm is applied.

Usage

```
spcacp(Y=NULL, d0=NULL, dims=NULL, nactive=NULL, dr=10, criteria="BIC",
             penalty="LASSO", D0=NULL, lambda=NULL, nlam=50, lam_min=1e-4,
             gamma=2, alpha=1, eps=1e-4, max_step=20)
```

Arguments

Y	An array with dimension dims, or a $n_{d0} \times N/n_{d0}$ numeric matrix of responses that is the mode d0-unfolding of tensor in $\mathcal{R}^{n_1 \times \cdots \times n_d}$, where $N = n_1 \times \cdots \times n_d$.
d0	d0 is the mode. Y is the mode-d0 unfolding of the tensor. d0 can be NULL (the default) if Y is an array with dimension dims.
dims	The size of tensor Y, which is a d-vector $(n_1, \cdots, n_d)$. dims can be NULL (the default) if Y is an array with dimension dims. If the length of dims is 2, it is the ordinary SVD decomposition of a matrix.
nactive	It is an integer vector, which are dimensional indeces without requirement of penalization. Default is NULL, which means all dimensions will be penalized.
dr	The user-specified rank for CP decomposition. Default is 10.
D0	A user-specified list of initial matrices of $U_1, U_2, \cdots, U_d$ and core tensor S, D0=list$(U_1 = U_1, \cdots, U_d = U_d, S = S)$. By default, initial matrices are provided by random.
criteria	The criteria to be applied to select tuning parameters. Either BIC (the default), AIC, or GCV.
penalty	The penalty to be applied to the model. Either "LASSO" (the default), "SCAD", or "MCP".
lambda	A user-specified sequence of lambda values. By default, a sequence of values of length nlam is computed, equally spaced on the log scale.
nlam	The number of lambda values. Default is 50.
lam_min	The smallest value for lambda, as a fraction of lambda.max. Default is 1e-4.
gamma	The tuning parameter of the MCP/SCAD penalty (see details).
alpha	Tuning parameter for the Mnet estimator which controls the relative contributions from the LASSO, MCP/SCAD penalty and the ridge, or L2 penalty. alpha=1 is equivalent to LASSO, MCP/SCAD penalty, while alpha=0 would be equivalent to ridge regression. However, alpha=0 is not supported; alpha may be arbitrarily small, but not exactly 0.
eps	Convergence threshhold. The algorithm iterates until the relative change in any coefficient is less than eps. Default is 1e-4.
max_step	Maximum number of iterations. Default is 20.

Details

This function gives a $n_{d0} \times N/n_{d0}$ matrix, which is the mode-d0 unfolding, and approximates sparse tensor Y.

Value

Tnew	Approximation of Y.
Tn	A list of estimated matrices of $U_1, U_2, \cdots, U_d$ and core tensor S, Tn=list($U_1 = U_1, \cdots, U_d = U_d, S = S$).
ranks	The ranks of estimated tensor Tnew. It is an integer.
rss	Residual sum of squares (RSS).
lambda	The sequence of regularization parameter values in the path.
selectedID	The index of lambda corresponding to lambda_opt.
lambda_opt	The value of lambda with the minimum BIC value.
df	Degrees of freedom.

See Also

pcals

Examples

```
# Example 1

dims <- c(8,8,10,10,6)
N <- length(dims)
ranks <- rep(2,N)
S0 <- matrix(runif(prod(ranks),3,7),ranks[N])
T1 <- matrix(rnorm(dims[1]*ranks[1]),nrow = dims[1])
tmp <- qr.Q(qr(T1))
for(k in 2:(N-1)){
  T1 <- matrix(rnorm(dims[k]*ranks[k]),nrow = dims[k])
  tmp <- kronecker(qr.Q(qr(T1)),tmp)
}
T1 <- matrix(rnorm(dims[N]*ranks[N]),nrow = dims[N])
U <- qr.Q(qr(T1))
Y <- U%*%S0%*%t(tmp)

fit <- spcacp(Y,N,dims)
Tnew <- fit$Tnew
ranks1 <- fit$ranks
U1 <- fit$Tn[[1]]
TNew1 <- ttu(Tnew,N,1,dims)


# Example 2
img = readPNG(system.file("data", "Rlogo.png", package="tensorApp"))
writePNG(img,target = "Rlogo.png")

Tn = spcacp(img,dr=20,dims=dim(img))
writePNG(Tn$Tnew,target = "RlogoCP.png")
```

tdsym2 *High-order SVD approximation of a tensor* Y *by Tucker decomposition*

Description

High-order SVD approximation of a semi-symmetric tensor Y by Tucker decomposition with rank preseted or to be selected. The Alterating Least Squares (als) is applied. The semi-symmetric tensor means that both mode-r1 and mode-r2 unfoldings are equal, that is, $Y_{(r1)} = Y_{(r2)}$. For semi-symmetric tensor approximation, the r1th and r2th dimensions must be smaller than others.

Usage

```
tdsym2(Y=NULL, r1=1, r2=2, d0=NULL, dims=NULL, ranks=NULL, dr=10, D0=NULL,
                            isfixr=TRUE, eps=1e-4, max_step=50, thresh=1e-6)
```

Arguments

Y	An array with dimension dims, or a $n_{d0} \times N/n_{d0}$ numeric matrix of responses that is the mode d0-unfolding of tensor in $\mathcal{R}^{n_1 \times \cdots \times n_d}$, where $N = n_1 \times \cdots \times n_d$.
r1	Both r1 and r2 are the user-specified modes, which means that both mode-r1 and mode-r2 unfoldings are equal. Default is r1 = 1 and r2 = 2.
r2	Both r1 and r2 are the user-specified modes, which means that both mode-r1 and mode-r2 unfoldings are equal. Default is r1 = 1 and r2 = 2.
d0	d0 is the mode. Y is the mode-d0 unfolding of the tensor. d0 can be NULL (the default) if Y is an array with dimension dims.
dims	The size of tensor Y, which is a d-vector $(n_1, \cdots, n_d)$. dims can be NULL (the default) if Y is an array with dimension dims. If the length of dims is 2, it is the ordinary SVD decomposition of a matrix.
ranks	The user-specified ranks. It is a vector with length d. Default is $(2, \cdots, 2)$.
dr	The user-specified rank for Tucker decomposition. Default is 10.
D0	A user-specified list of initial matrices of $U_1, U_2, \cdots, U_d$ and core tensor S, D0=list$(U_1 = U_1, \cdots, U_d = U_d, S = S)$. By default, initial matrices are provided by random.
isfixr	A logical value indicating whether the rank is fixed. The rank is selected automatically if it is TRUE. Default is TRUE.
eps	Convergence threshhold. The algorithm iterates until the relative change in any coefficient is less than eps. Default is 1e-4.
max_step	Maximum number of iterations. Default is 50.
thresh	Convergence threshhold in the outer loop. The algorithm iterates until the relative change in any coefficient is less than eps. Default is 1e-6.

Details

This function gives a $n_{d0} \times N/n_{d0}$ matrix, which is the mode-d0 unfolding, and approximates Y.

Value

Tnew	Approximation of Y.
Tn	A list of estimated matrices of $U_1, U_2, \cdots, U_d$ and core tensor S, Tn=list($U_1 = U_1, \cdots, U_d = U_d, S = S$).
ranks	The ranks of estimated tensor Tnew. It is an integer.

See Also

hosvd_dr, cpsym2

Examples

```
# Example 1
dims <- c(6,6,8,7,7)
N <- length(dims)
ranks <- rep(2,N)
dm <- prod(ranks)
r1 <- 1
r2 <- 2
S1 <- matrix(runif(dm,3,7),nrow = ranks[r1])
S2 <- ttu(S1,r1,r2,ranks)
S1 <- (S1+S2)/2
S0 <- ttu(S1,r1,N,ranks)

T1 <- matrix(rnorm(dims[1]*ranks[1]),nrow = dims[1])
tmp <- qr.Q(qr(T1))
Uj <- kronecker(tmp,tmp)
for(k in 3:(N-1)){
  T1 <- matrix(rnorm(dims[k]*ranks[k]),nrow = dims[k])
  tmp <- qr.Q(qr(T1))
  Uj <- kronecker(tmp,Uj)
}
T1 <- matrix(rnorm(dims[N]*ranks[N]),nrow = dims[N])
U <- qr.Q(qr(T1))
Y <- U%*%S0%*%t(Uj)

fit <- tdsym2(Y,r1=1,r2=2,d0=N,dims=dims)
Tnew <- fit$Tnew
ranks1 <- fit$ranks
U1 <- fit$Tn[[r1]]
U2 <- fit$Tn[[r2]]
TNew1 <- ttu(Tnew,N,r1,dims)
TNew2 <- ttu(Tnew,N,r2,dims)


# Example 2

Y = gttsem(dims,r1=r1,r2=r2,d0=N,ranks=ranks)
fit <- tdsym2(Y,r1=1,r2=2,d0=N,dims=dims)
Tnew <- fit$Tnew
ranks1 <- fit$ranks
U1 <- fit$Tn[[r1]]
U2 <- fit$Tn[[r2]]
```

```
TNew1 <- ttu(Tnew,N,r1,dims)
TNew2 <- ttu(Tnew,N,r2,dims)
```

---

tmp *Modal Product*

Description

Modal product calculates the product an order d tensor S and a matrix M, that is $T = S \times_{d0} M$ that means $T_{(d0)} = M \cdot S_{(d0)}$, where $S_{(d0)}$ is the mode-$d0$ unfloding of the tensor S.

Usage

```
tmp(S=NULL, M=NULL, d0=NULL, dims=NULL)
```

Arguments

S	An order d tensor with dimension dims, where dims is $(n_1, \cdots, n_d)$.
M	A matrix with K rows and n_{d0} columns. The modal product produces a new order d tensor with dimension dims replaced the d0th dimnesion by K.
dims	The dimension of tensor Y, which is a d-vector $(n_1, \cdots, n_d)$.
d0	d0 is the mode. M multiplies the mode-d0 unfoldings of S.

Details

This function gives the modal product of tensor S and matrix M, which is the product of M and the mode-d0 unfolding of tensor S, that is $S \times_{d0} M = MS_{(d0)}$, where $S_{(d0)}$ is the mode-d0 unfolding of the tensor S. The modal product produce a new order d tensor with dimension dims replaced the d0 dimnesion by K, where K is the number of rows of matrix M.

Value

Tnew	Product of S and M.

See Also

TransUnfoldingsT

Examples

```
dims <- c(8,8,10,10,6)
N <- length(dims)
ranks <- rep(2,N)
S0 <- matrix(runif(prod(ranks),3,7),ranks[N])
T1 <- matrix(rnorm(dims[1]*ranks[1]),nrow = dims[1])
tmp <- qr.Q(qr(T1))
for(k in 2:(N-1)){
  T1 <- matrix(rnorm(dims[k]*ranks[k]),nrow = dims[k])
  tmp <- kronecker(qr.Q(qr(T1)),tmp)
}
```

```
T1 <- matrix(rnorm(dims[N]*ranks[N]),nrow = dims[N])
U <- qr.Q(qr(T1))
Y <- U%*%S0%*%t(tmp)
Y1 <- array(ttu(Y,N,1,dims),dims)
M <- matrix(1:(4*dims[3]),4)

X1 <- tmp(Y1,M,3,dims)
X <- ttu(matrix(X1,dims[1]),1,N,dims)

dim(Y1)
dim(X1)
print(Y[,1])
print(X[,1])
```

---

ttu *Transfer a tensor's modal unfolding to another.*

Description

Transfer a tensor's modal unfolding to another.

Usage

```
ttu(S=NULL, d1=NULL, d2=0, dims=NULL)
```

Arguments

S	An array with dimension dims, or a mode-d1-unfolding of a tensor with size $n_1 \times \cdots \times n_d$.
d1	An integer, the mode of unfolding $S_{(d_1)}$. d1 can be NULL (the default) if S is an array with dimension dims.
d2	An integer, the mode of output unfolding $S_{(d_2)}$. It transfers S to an array with dimension dims if d2=0. The default is 0.
dims	The size of tensor S, which is a vector $(n_1, \cdots, n_d)$. dims can be NULL (the default) if S is an array with dimension dims.

Details

This function transfers an input mode-d1-unfolding $S_{(d_1)}$ to mode-d2-unfolding $S_{(d_2)}$

Value

Td2	the output mode-d2-unfolding, $S_{(d_2)}$.

Examples

```
T1 <- matrix(1:24,nrow = 4) # A tensor unfolding with size 4*6
T2 <- ttu(T1,1,2,c(4,3,2))

T0 <- ttu(T2,2,dims=c(4,3,2))
```

Index