Package ‘tensorApp’

February 12, 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 by Tucker and CP decomposition and selection of ranks.

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

Archs i386, x64

R topics documented:

---

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

Description

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

Details

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

Author(s)

Xu Liu

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

HOsvd — *High-order SVD approximation by Tucker and CP decomposition*

Description

High-order SVD approximation by Tucker decomposition or CANDECOMP/PARAFAC (CP) decomposition with preset rank.

Usage

```
HOsvd(Y, d0=NULL, dims=NULL, isCP=TRUE, ranks=NULL, dr=20,
                                          D0=NULL, 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 which unfoldings `Y` is. `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 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-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 core tensor S, Tn=list($U_1 = U_1, \cdots, U_d = U_d, S = S$).
ranks	The ranks of estimated tensor Tnew.

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 <- HOsvd(Y,N,dims,isCP=TRUE)
Tnew <- fit$Tnew
ranks1 <- fit$ranks
TNew1 <- TransUnfoldingsT(Tnew,N,1,dims)
```

HOsvd_dr *High-order SVD approximation by Tucker and CP decomposition and selection of ranks*

Description

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

Usage

```
HOsvd_dr(Y, d0=NULL, dims=NULL, isCP=TRUE, ranks=NULL, dr=100,
                                    D0=NULL, 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 which unfoldings `Y` is. `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 100.

`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 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 `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`.

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 <- HOsvd_dr(Y,N,dims,isCP=TRUE)
Tnew <- fit$Tnew
ranks1 <- fit$ranks
TNew1 <- TransUnfoldingsT(Tnew,N,1,dims)
```

`TransUnfoldingsT` *Transfer a tensor's modal unfoldings to another.*

Description

Transfer a tensor's modal unfoldings to another.

Usage

```
TransUnfoldingsT(S, 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 <- TransUnfoldingsT(T1,1,2,c(4,3,2))

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

Index