Package 'TRES'

November 26, 2018

Type Package

Title Tensor Regression with Envelope Structure and Three Generic Envelope Estimation Approaches

Version 0.1.0

Author Wenjing Wang [aut, cre] <wenjing.wang@stat.fsu.edu>, Xin Zhang [aut, cre] <henry@stat.fsu.edu>

Maintainer Wenjing Wang <wenjing.wang@stat.fsu.edu>

Description Provides three estimators for tensor response regression (TRR) and tensor predictor regression (TPR) models with tensor envelope structure. The three types of estimation approaches are generic and can be applied to any envelope estimation problems. The full Grassmannian (FG) optimization is often associated with likelihood-based estimation but requires heavy computation and good initialization; the one-directional optimization approaches (1D and ECD algorithms) are faster, stable and does not require carefully chosen initial values; the SIMPLS-type is motivated by the partial least squares regression and is computationally the least expensive.

License GPL-3

Encoding UTF-8

Depends R (>= 3.5.0), ManifoldOptim, rTensor, MASS

Imports pracma, mvtnorm, methods

RcppModules ManifoldOptim_module

RoxygenNote 6.1.0

NeedsCompilation no

Repository CRAN

Date/Publication 2018-11-26 20:20:06 UTC

R topics documented:

`ballGBB1D_bic` *Envelope dimension selection based on 1D-BIC*

Description

This function selects envelope subspace dimension use 1D-BIC by Zhang, X., & Mai, Q. (2018). The constrained optimization in the 1D algorithm is based on the method proposed by Wen and Yin (2013).

Usage

```
ballGBB1D_bic(M, U, n, multiD=1, bic_max=10, opts=NULL)
```

Arguments

`M`	M matrix in the envelope objective function. A p-by-p positive semi-definite matrix.
`U`	U matrix in the envelope objective function. A p-by-p positive semi-definite matrix.
`n`	Sample size.
`multiD`	A constant, see Zhang, X., & Mai, Q. (2018), the default value is 1.
`bic_max`	The maximum dimension to consider, `bic_max` is smaller than p, the default value is 10.
`opts`	Option structure for GBB algorithm. See function `OptStiefelGBB`.

Value

bicval	The BIC values for different envelope dimensions.
u	The dimension selected which corresponds to the smallest BIC values.

References

Zhang, X., & Mai, Q. (2018). Model-free envelope dimension selection. Electronic Journal of Statistics, 12(2), 2193-2216.
Wen, Z., & Yin, W. (2013). A feasible method for optimization with orthogonality constraints. Mathematical Programming, 142(1-2), 397-434.

Examples

```
##simulate two matrices M and U with an envelope structure#
data <- MenvU_sim(n=200, p=20, u=5)
Mhat <- data$Mhat
Uhat <- data$Uhat

res <- ballGBB1D_bic(Mhat, Uhat, n=200)

## visualization
plot(1:10, res$bicval, type="o", xlab="Envelope Dimension", ylab="BIC values",
main="Envelope Dimension Selection")
```

ECD *ECD algorithm for estimating the envelope subspace*

Description

Estimate the envelope subspace with specified dimension based on ECD algorithm that described in Cook, R. D., & Zhang, X. (2018).

Usage

```
ECD(M, U, u, maxiter=500, epsilon=1e-08)
```

Arguments

M	M matrix in the envelope objective function. A p-by-p positive semi-definite matrix.
U	U matrix in the envelope objective function. A p-by-p positive semi-definite matrix.
u	Envelope dimension. An integer between 0 and p.
maxiter	Maximum number of iterations.
epsilon	Convergence criterion. $\|F_k - F_{k-1}\| < \epsilon$, where F_k is the objective function.

Details

Estimate M-envelope contains span(U) where M > 0 and is symmetric. The dimension of the envelope is u.

Value

Ghat	The orthogonal basis of the envelope subspace with each column represent the sequential direction. For example, the 1st column is the most informative direction.

References

Cook, R. D., & Zhang, X. (2018). Fast envelope algorithms. Statistica Sinica, 28(3), 1179-1197.

Examples

```
##simulate two matrices M and U with an envelope structure#
data <- MenvU_sim(n=200, p=20, u=5)
Mhat <- data$Mhat
Uhat <- data$Uhat

Ghat_ECD <- ECD(Mhat, Uhat, u=5)
```

EnvMU *Estimate envelope subspace basis*

Description

SIMPLS-type algorithm for estimating the M-envelope of span(U) without manifold optimization. This algorithm is a generalization of De Jong, S. (1993) and Cook, R. D., Helland, I. S., & Su, Z. (2013). It generalizes from predictor envelopes to an arbitrary M-envelope of span(U).

Usage

```
EnvMU(M, U, u)
```

Arguments

M	M matrix in the envelope objective function. An p-by-p positive semi-definite matrix.
U	U matrix in the envelope objective function. An p-by-p positive semi-definite matrix.
u	The envelope dimension.

Value

Gamma	The basis of M-envelope of span(U).

References

Cook, R. D., Helland, I. S., & Su, Z. (2013). Envelopes and partial least squares regression. Journal of the Royal Statistical Society: Series B (Statistical Methodology), 75(5), 851-877.

De Jong, S. (1993). SIMPLS: an alternative approach to partial least squares regression. Chemometrics and intelligent laboratory systems, 18(3), 251-263.

`FGfun` *The Objective function and its gradient*

Description

The generic objective function listed below for estimating `M` envelope of `span(U)` and its gradient value. For the detailed description, see Cook, R. D., & Zhang, X. (2016).

$$F(\Gamma) = \log|\Gamma^T M\Gamma| + \log|\Gamma^T(M+U)^{-1}\Gamma|$$

Usage

```
FGfun(W, M, U)
```

Arguments

`M`	M matrix in the envelope objective function. A p-by-p positive semi-definite matrix.
`U`	U matrix in the envelope objective function. A p-by-p positive semi-definite matrix.
`W`	A vector of p by u.

Details

This is the objective function and its gradient for estimating `M`-envelope contains `span(U)`, where `M > 0` and is symmetric, the dimension of the envelope is u.

Value

`F`	The value of objective function given `W`.
`G`	The value of the gradient function given `W`.

References

Cook, R. D., & Zhang, X. (2016). Algorithms for envelope estimation. Journal of Computational and Graphical Statistics, 25(1), 284-300.

FG_TPR *Envelope estimation of tensor predictor regression with the full Grassmannian optimization*

Description

This function is used for envelope estimation of tensor predictor regression with the full Grassmannian (FG) optimization.

Usage

```
FG_TPR(Yn, Xn, Gamma_init)
```

Arguments

Yn	The predictor matrix of dimension $r \times n$.
Xn	The predictor tensor instance or dimension $p_1 \times p_2 \times \cdots \times p_m \times n$, where n is the sample size.
Gamma_init	The initial estimation of envelope subspace basis, can be derived from TPR.

Value

Bhat	The estimation of regression coefficient tensor.
Gamma_hat	The FG estimation of envelope subspace basis.

Examples

```
rm(list = ls())

p <- c(10, 10, 10)
u <- c(1, 1, 1)
m <- 3; r <- 5; n <- 200
eta <- array(runif(prod(u,r)), c(u,r))
eta <- as.tensor(eta)

Gamma <- Gamma0 <- Omega <- Omega0 <- Sig <- Sigsqrtm <- NULL
for(i in 1:m) {
  tmp <- matrix(runif(p[i]*u[i]), p[i], u[i])
  Gamma[[i]] <- qr.Q(qr(tmp))
  Gamma0[[i]] <- qr.Q(qr(tmp), complete=TRUE)[, (u[i]+1):p[i]]
  Omega[[i]] <- diag(u[i])
  Omega0[[i]] <- 0.01*diag(p[i]-u[i])
  Sig[[i]] <- Gamma[[i]] %*% Omega[[i]] %*% t(Gamma[[i]])+
    Gamma0[[i]] %*% Omega0[[i]] %*% t(Gamma0[[i]])
  Sig[[i]] <- 2*Sig[[i]]/norm(Sig[[i]], type="F")
  Sigsqrtm[[i]] <- pracma::sqrtm(Sig[[i]])$B
}
```

```
B <- ttl(eta,Gamma, ms = c(1:m))
A <- matrix(runif(r^2), r, r)
SigY <- A %*% t(A)
SigY <- SigY/norm(SigY, type="F")


##generate data
Epsilon <- MASS::mvrnorm(n, mu=rep(0, r), Sigma=SigY)
tmp2 <- array(rnorm(prod(p, n)), c(p, n))
Xn <- as.tensor(tmp2)
Xn <- ttl(Xn, Sigsqrtm, ms = c(1:m))
vecXn <- matrix(Xn@data, prod(p), n)
Y_tmp <- matrix(NA, r, n)
tmp <- array(NA, c(p, r))
for (j in 1:n) {
  for (s in 1:r) {
    tmp[, , , s] <-  B@data[, , , s]*Xn@data[, , , j]
  }
  Y_tmp[, j] <- apply(tmp, 4, sum)
}
Yn <-  Y_tmp + t(Epsilon)



res_1D = TPR(Yn, Xn, u, method="1D")
res_FG = FG_TPR(Yn, Xn, Gamma_init=res_1D$Gamma_hat)

rTensor::fnorm(B-res_1D$Bhat)
rTensor::fnorm(B-res_FG$Bhat)
```

FG_TRR *Envelope estimation of tensor response regression with the full Grassmannian optimization*

Description

This function is used for envelope estimation of tensor response regression with the full Grassmannian (FG) optimization.

Usage

```
FG_TRR(Yn, Xn, Gamma_init)
```

Arguments

Yn	The response tensor instance or dimension $r_1 \times r_2 \times \cdots \times r_m \times n$, where n is the sample size.
Xn	The predictor matrix of dimension $p \times n$.
Gamma_init	The initial estimation of envelope subspace basis, can be derived from TRR.

Value

Bhat	The estimation of regression coefficient tensor.
Gamma_hat	The FG estimation of envelope subspace basis.

Examples

```
rm(list=ls())

r <- c(10, 10, 10)
m <- length(r)
u <- c(2, 2, 2)
p <- 5
n <- 100


set.seed(1)
eta <- array(runif(prod(u)*p), c(u, p))
eta <- as.tensor(eta)

Gamma <- Gamma0 <- Omega <- Omega0 <- Sig <- Sigsqrtm <- NULL
for (i in 1:m){
  tmp <- matrix(runif(r[i]*u[i]), r[i], u[i])
  Gamma[[i]] <- qr.Q(qr(tmp))
  Gamma0[[i]] <- qr.Q(qr(Gamma[[i]]),complete=TRUE)[,(u[i]+1):r[i]]

  A <- matrix(runif(u[i]^2), u[i], u[i])
  Omega[[i]] <- A %*% t(A)
  A <- matrix(runif((r[i]-u[i])^2), (r[i]-u[i]), (r[i]-u[i]))
  Omega0[[i]] <- A %*% t(A)
  Sig[[i]] <- Gamma[[i]] %*% Omega[[i]] %*% t(Gamma[[i]])+
    Gamma0[[i]] %*% Omega0[[i]] %*% t(Gamma0[[i]])
  Sig[[i]] <- 10*Sig[[i]]/norm(Sig[[i]], type="F")+0.01*diag(r[i])
  Sigsqrtm[[i]] <- pracma::sqrtm(Sig[[i]])$B
}
B <- ttl(eta, Gamma, ms=1:m)
Xn <- matrix(rnorm(p*n), p, n)
Xn_inv <- MASS::ginv(Xn %*% t(Xn)) %*% Xn
Epsilon <- array(rnorm(prod(r)*n), c(r, n))
Epsilon <- as.tensor(Epsilon)
Epsilon <- ttl(Epsilon, Sigsqrtm, ms=1:m)
Yn <- Epsilon + ttm(B, t(Xn), m+1)


res_1D = TRR(Yn, Xn, u, method="1D")
res_FG = FG_TRR(Yn, Xn, Gamma_init=res_1D$Gamma_hat)

rTensor::fnorm(B-res_1D$Bhat)
rTensor::fnorm(B-res_FG$Bhat)
```

fun1D *The 1D objective function and its gradient*

Description

The objective function and its gradient value that defined in equation (4.1) of Cook, R. D., & Zhang, X. (2016). A speicial case of `FGfun` where `W` is a one-dimensional vector.

Usage

```
fun1D(W, M, U)
```

Arguments

`M`	M matrix in the envelope objective function. A p-by-p positive semi-definite matrix.
`U`	U matrix in the envelope objective function. A p-by-p positive semi-definite matrix.
`W`	A vector of p by 1.

Details

This is the objective function and its gradient for the constrained optimization in the 1D algorithm.

Value

`F`	The value of objective function given `W`.
`G`	The value of the gradient function given `W`.

References

Cook, R. D., & Zhang, X. (2016). Algorithms for envelope estimation. Journal of Computational and Graphical Statistics, 25(1), 284-300.

get_ini1D *Initial value for 1D algorithm*

Description

This function gives the initial value for 1D algorithm, which is searched from the eigenvectors of matrices `M` and `M+U` corresponding to their objective function value.

Usage

```
get_ini1D(M, U)
```

Arguments

M	M matrix in the envelope objective function. A p-by-p positive semi-definite matrix.
U	U matrix in the envelope objective function. A p-by-p positive semi-definite matrix.

Value

W0	Initial value for the 1D algorihtm

---

kroncov *The covariance estimation of a tensor random variable*

Description

This function estimates the covariance of a tensor random variable. We assume the covariance of the tensor random variable has a seperable Kronecker covariance structure, i.e. $\mathbf{\Sigma} = \mathbf{\Sigma}_m \otimes \cdots \otimes \mathbf{\Sigma}_1$. This algorithm is described in Manceur, A. M., & Dutilleul, P. (2013).

Usage

```
kroncov(Tn)
```

Arguments

Tn	A $p_1 \times \cdots p_m \times n$ data array, where n is the sample size.

Value

lambda	The normalizing constant.
S	A matrix lists with each element being the individual estimation of the seperable Kronecker covariance element $\mathbf{\Sigma}_m, \ldots, \mathbf{\Sigma}_1$.

References

Manceur, A. M., & Dutilleul, P. (2013). Maximum likelihood estimation for the tensor normal distribution: Algorithm, minimum sample size, and empirical bias and dispersion. Journal of Computational and Applied Mathematics, 239, 37-49.

`manifold1D` *Estimate the envelope subspace (ManifoldOptim 1D)*

Description

The 1D algorithm to estimate the envelope subspace with specified dimension based on R package "ManifoldOptim".

Usage

```
manifold1D(M, U, u, params=NULL)
```

Arguments

`M`	M matrix in the envelope objective function. A p-by-p positive semi-definite matrix.
`U`	U matrix in the envelope objective function. A p-by-p positive semi-definite matrix.
`u`	Dimension of the envelope. An integer between 0 and p.
`params`	Option structure with fields: `"max_iter"` – max number of iterations. `"tol"` – Tolerance used to assess convergence. See Huang et al (2018) for details on how this is used. `"method"` – Name of optimization method supported by R package **ManifoldOptim**. • `"LRBFGS"`: Limited-memory RBFGS • `"LRTRSR1"`: Limited-memory RTRSR1 • `"RBFGS"`: Riemannian BFGS • `"RBroydenFamily"`: Riemannian Broyden family • `"RCG"`: Riemannian conjugate gradients • `"RNewton"`: Riemannian line-search Newton • `"RSD"`: Riemannian steepest descent • `"RTRNewton"`: Riemannian trust-region Newton • `"RTRSD"`: Riemannian trust-region steepest descent • `"RTRSR1"`: Riemannian trust-region symmetric rank-one update • `"RWRBFGS"`: Riemannian BFGS `"check"` – Should internal manifold object check inputs and print summary message before optimization (TRUE or FALSE). The default values are: `"max_iter"=500; "tol"=1e-08; "method"="RCG"; "check"="False"`.

Details

Estimate M-envelope contains span(U) where M > 0 and is symmetric. The dimension of the envelope is u.

Value

Ghat	The orthogonal basis of the envelope subspace with each column represent the sequential direction. For example, the 1st column is the most informative direction.

References

Huang, W., Absil, P. A., Gallivan, K. A., & Hand, P. (2018). ROPTLIB: an object-oriented C++ library for optimization on Riemannian manifolds. ACM Transactions on Mathematical Software (TOMS), 44(4), 43.

Examples

```
##simulate two matrices M and U with an envelope structure#
data <- MenvU_sim(n=200, p=20, u=5)
Mhat <- data$Mhat
Uhat <- data$Uhat
G <- data$Gamma

Ghat_1D <- manifold1D(Mhat, Uhat, u=5, params=NULL)
subspace(Ghat_1D, G)
```

---

manifoldFG *Full Grassmann manifold optimization algorithm (ManifoldOptim)*

Description

Estimate the envelope subspace based on R package "ManifoldOptim".

Usage

```
manifoldFG(M, U, u, G_ini, params=NULL)
```

Arguments

M	M matrix in the envelope objective function. A p-by-p positive semi-definite matrix.
U	U matrix in the envelope objective function. A p-by-p positive semi-definite matrix.
u	Dimension of the envelope. An integer between 0 and p.
G_ini	The initial value for mannifold optimization.

params — Option structure with fields:
"max_iter" – max number of iterations.
"tol" – Tolerance used to assess convergence. See Huang et al (2018) for details on how this is used.
"method" – Name of optimization method supported by R package **ManifoldOptim**.

- "LRBFGS": Limited-memory RBFGS
- "LRTRSR1": Limited-memory RTRSR1
- "RBFGS": Riemannian BFGS
- "RBroydenFamily": Riemannian Broyden family
- "RCG": Riemannian conjugate gradients
- "RNewton": Riemannian line-search Newton
- "RSD": Riemannian steepest descent
- "RTRNewton": Riemannian trust-region Newton
- "RTRSD": Riemannian trust-region steepest descent
- "RTRSR1": Riemannian trust-region symmetric rank-one update
- "RWRBFGS": Riemannian BFGS

"check" – Should internal manifold object check inputs and print summary message before optimization (TRUE or FALSE).

The default values are: "max_iter"=500; "tol"=1e-08; "method"="RCG"; "check"="False".

Details

Estimating M-envelope contains span(U) where M > 0 and is symmetric and the dimension of the envelope is u.

Value

G_hat — The orthogonal basis of the envelope subspace.

Examples

```
##simulate two matrices M and U with an envelope structure#
data <- MenvU_sim(n=200, p=20, u=5)
Mhat <- data$Mhat
Uhat <- data$Uhat
G <- data$Gamma

Ghat_1D <- manifold1D(Mhat, Uhat, u=5)
subspace(Ghat_1D, G)

Ghat_FG <- manifoldFG(Mhat, Uhat, u=5, Ghat_1D)
subspace(Ghat_FG, G)
```

MenvU_sim *Generate sample estimator of* M *and* U

Description

This function is to generate sample estimator of M and U, which are the matrices in M-envelope of U.

Usage

```
MenvU_sim(n, p, u, Omega=NULL, Omega0=NULL, Phi=NULL)
```

Arguments

n	Sample size.
p	Dimension of M.
u	Envelope Dimension.
Omega	The positive definite matrix Ω in $M = \Gamma^T\Omega\Gamma + \Gamma_0^T\Omega_0\Gamma_0^T$. The defalut is generated by $\Omega = AA^T$ with the elements in A is generated from a standard uniform distribution.
Omega0	The positive definite matrix Ω_0 in $M = \Gamma^T\Omega\Gamma + \Gamma_0^T\Omega_0\Gamma_0^T$. The defalut is generated by $\Omega = AA^T$ with the elements in A is generated from a standard uniform distribution.
Phi	The positive definite matrix Φ in $U = \Gamma^T\Phi\Gamma$. The defalut is generated by $\Omega = AA^T$ with the elements in A is generated from a standard uniform distribution.

Value

Mhat	Sample estimator of M.
Uhat	Sample estimator of U.
Gamma	Matrix Γ in generating M and U.

Examples

```
data <- MenvU_sim(n=200, p=20, u=5, Omega=NULL, Omega0=NULL, Phi=NULL)
Mhat <- data$Mhat
Uhat <- data$Uhat
```

`OptimballGBB1D` *Estimate the envelope subspace (Feasi 1D)*

Description

The 1D algorithm to estimate the envelope subspace with specified dimension based on Wen and Yin (2013).

Usage

```
OptimballGBB1D(M, U, u, opts=NULL)
```

Arguments

`M`	M matrix in the envelope objective function. A p-by-p positive semi-definite matrix.
`U`	U matrix in the envelope objective function. A p-by-p positive semi-definite matrix.
`u`	Dimension of the envelope. An integer between 0 and p.
`opts`	Option structure with fields: `"record = 0"` – no print out. `"mxitr"` – max number of iterations. `"xtol"` – stop control for $\|\|X_k - X_{k-1}\|\|$. `"gtol"` – stop control for the projected gradient. `"ftol"` – stop control for $\frac{\|F_k - F_{k-1}\|}{(1+\|F_{k-1}\|)}$ usually with `max{xtol, gtol} > ftol`. The default values are: `"xtol"=1e-08; "gtol"=1e-08; "ftol"=1e-12; "mxitr"=500.`

Details

Estimate `M`-envelope contains `span(U)` where `M > 0` and is symmetric. The dimension of the envelope is `u`.

Value

`Ghat`	The orthogonal basis of the envelope subspace with each column represent the sequential direction. For example, the 1st column is the most informative direction.

References

Wen, Z., & Yin, W. (2013). A feasible method for optimization with orthogonality constraints. Mathematical Programming, 142(1-2), 397-434.

Examples

```
##simulate two matrices M and U with an envelope structure#
data <- MenvU_sim(n=200, p=20, u=5, Omega=NULL, Omega0=NULL, Phi=NULL)
Mhat <- data$Mhat
Uhat <- data$Uhat
G <- data$Gamma

Ghat_1D <- OptimballGBB1D(Mhat, Uhat, u=5)
subspace(Ghat_1D, G)
```

`OptManiMulitBallGBB` *Line search algorithm for optimization on manifold*

Description

Line search algorithm for optimization on Stiefel manifold based on Wen and Yin (2013). Used for the 1D-algorithm to estimate the envelope subspace.

Usage

```
OptManiMulitBallGBB(X, opts=NULL, fun, ...)
```

Arguments

`X`	n by k matrix such that $X'X = I$
`opts`	Option structure with fields: `"record = 0"` – no print out. `"mxitr"` – max number of iterations. `"xtol"` – stop control for $\|\|X_k - X_{k-1}\|\|$. `"gtol"` – stop control for the projected gradient. `"ftol"` – stop control for $\frac{\|F_k - F_{k-1}\|}{(1+\|F_{k-1}\|)}$ usually with `max{xtol, gtol} > ftol`.
`fun`	Objective function and its gradient: `fun(X, data1, data2)` `data1, data2` are addtional data.
`...`	Additional input for `fun`, Calling syntax: `OptManiMulitBallGBB(X0, fun, opts, data1, data2)`.

Value

`X`	Solution.
`g`	Gradient of X.
`Out`	Output information, include estimation error, function value, iteration times etc.

References

Wen, Z., & Yin, W. (2013). A feasible method for optimization with orthogonality constraints. Mathematical Programming, 142(1-2), 397-434.

`OptStiefelGBB` *Optimization on Stiefel manifold (GBB)*

Description

Curvilinear search algorithm for optimization on Stiefel manifold based on Wen and Yin (2013). Used for estimating the envelope subspace.

Usage

```
OptStiefelGBB(X, opts=NULL, fun, ...)
```

Arguments

`X`	n by k matrix such that $X'X = I$
`opts`	Option structure with fields: `"record = 0"` – no print out. `"mxitr"` – max number of iterations. `"xtol"` – stop control for $\|\|X_k - X_{k-1}\|\|$. `"gtol"` – stop control for the projected gradient. `"ftol"` – stop control for $\frac{\|F_k - F_{k-1}\|}{(1+\|F_{k-1}\|)}$ usually with `max{xtol, gtol} > ftol`. The default values are: `"xtol"=1e-08; "gtol"=1e-08; "ftol"=1e-12; "mxitr"=500.`
`fun`	Objective function and its gradient: `fun(X, data1, data2)` `data1, data2` are addtional data.
`...`	Additional input for `fun`, Calling syntax: `OptStiefelGBB(X0, fun, opts, data1, data2)`.

Value

`X`	Solution
`Out`	Output information, inclusde estimation error, function value, iteration times etc.

References

Wen, Z., & Yin, W. (2013). A feasible method for optimization with orthogonality constraints. Mathematical Programming, 142(1-2), 397-434.

Examples

```
fun <- function(X,  W) {
      G = -2*(W %*% X)
      F = -sum(diag(t(X) %*% W %*% X))
      return(list(F = F, G = G))
```

```
}
n = 1000; k = 6;
W = matrix(rnorm(n^2), n, n); W = t(W) %*% W

opts=c()
opts$record = 0;
opts$mxitr  = 1000;
opts$xtol = 1e-5;
opts$gtol = 1e-5;
opts$ftol = 1e-8;

X0 = matrix(rnorm(n*k), n, k);
X0 = qr.Q(qr(X0));

eva <- OptStiefelGBB(X0, opts, fun, W)
X <- eva$X
out <- eva$out
out$fval = -2*out$fval;
```

PMSE *Predictions and Mean squared error for tensor predictor regression*

Description

Evaluate tensor predictor regression through prediction mean squared error.

Usage

```
PMSE(Xn, Yn, Bhat)
```

Arguments

Xn	A predictor tensor.
Yn	A response vector.
Bhat	An estimation of coefficient tensor.

Value

mse	Mean squared error. Defined as $trace(\sum_{i=1}^{n}(\mathbf{Y}_i - \hat{\mathbf{Y}}_i)(\mathbf{Y}_i - \hat{\mathbf{Y}}_i)'/n)$, where $\hat{\mathbf{Y}}_i$ is the predictions.
Yhat	The predictions of tensor predictor regression.

subspace *The distance between two subspaces* span(A) *and* span(B).

Description

This function calculate the distance between two subspaces span(A) and span(B).

Usage

```
subspace(A, B)
```

Arguments

A	A p-by-u matrix.
B	A p-by-u matrix.

Value

Returns a distance metric that is between 0 and 1

TensEnv_dim *Envelope dimension selection for tensor response regression*

Description

This function use the 1D-BIC criterion by by Zhang, X., & Mai, Q. (2018) to select envelope dimensions in tensor response regression.

Usage

```
TensEnv_dim(Yn, Xn, multiD=1, bic_max=10, opts=NULL)
```

Arguments

Xn	A vector predictor of dimension p.
Yn	The response tensor instance $r_1 \times \cdots \times r_m$.
multiD	The parameter in ballGBB1D_bic.
bic_max	The maximum envelope dimension to be considered, the parameter in ballGBB1D_bic.
opts	The parameter in ballGBB1D_bic.

Value

u	The envelope dimension of $(u_1, u_2, \cdots, u_m)$.

`TensPLS_cv2d3d` *Envelope dimension by cross-validation for tensor predictor regression*

Description

This function obtain the envelope dimension by cross-validation for tensor predictor regression.

Usage

```
TensPLS_cv2d3d(X0, Y0, maxdim, nfolds)
```

Arguments

`X0`	A predictor tensor instance.
`Y0`	The response vector.
`maxdim`	The largest dimension to be considered for selection.
`nfolds`	Number of folds for cross-validation.

Value

`mincv`	The minimum sum of squared error.
`u`	The envelope subspace dimension selected.

References

Zhang, X., & Li, L. (2017). Tensor Envelope Partial Least-Squares Regression. Technometrics, 59(4), 426-436.

`TensPLS_fit` *Tensor envelope partial least squares (PLS) regression*

Description

This function estimates the factor matrix $W_k, k = 1, \cdots, m$ in tensor PLS algorithm for tensor predictor regression, see Zhang, X., & Li, L. (2017).

Usage

```
TensPLS_fit(Xn, Yn, SigX, u)
```

Arguments

`Xn`	A predictor tensor of dimension $p_1 \times \cdots \times p_m$.
`Yn`	The response vector of dimension r.
`SigX`	A matrix lists $\mathbf{\Sigma}_k, k = 1, \cdots, m$, which determins the estimation of covariance matrix $\mathbf{\Sigma} = \mathbf{\Sigma}_m \otimes \cdots \otimes \mathbf{\Sigma}_1$.
`u`	The dimension of envelope subspace, $u = (u_1, \cdots, u_m)$.

Value

`Gamma`	The estimation of factor matrix $W_k, k = 1, \cdots, m$.
`PGamma`	The projection matrix $W_k(W_k'\mathbf{\Sigma}_k W_k)^{-1}W_k'\mathbf{\Sigma}_k, k = 1, \cdots, m$.

References

Zhang, X., & Li, L. (2017). Tensor Envelope Partial Least-Squares Regression. Technometrics, 59(4), 426-436.

`Tenv` *Tensor response envelope estimator*

Description

This function gives the tensor envelope estimator for tensor response regression.

Usage

```
Tenv(Yn, Xn, u, opts=NULL)
```

Arguments

`Yn`	The response tensor instance $r_1 \times r_2 \times \cdots \times r_m \times n$, where n is the sample size.
`Xn`	The predictor matrix of dimension $p \times n$.
`u`	The dimension of envelope subspace. $u = (u_1, \cdots, u_m)$.
`opts`	The option structure for Feasi. See function `OptimballGBB1D`.

Value

`Btil`	The ordinary least square estimator (OLS).
`Bhat`	The envelope based estimator.
`PGamma`	The projection matrix onto envelope subspace.

References

Li, L., & Zhang, X. (2017). Parsimonious tensor response regression. Journal of the American Statistical Association, 112(519), 1131-1146.

Tenv_Pval *The p-values of each elements in the tensor response regression coefficient estimator*

Description

Obtain p-values of each elements in the tensor regression coefficient estimator. Two-sided t-tests are applied on the coefficent estimator, where the estimator use the asymptotic covariance of the OLS estimator.

Usage

```
Tenv_Pval(Yn, Xn, B_est)
```

Arguments

Yn	The response tensor instance $r_1 \times r_2 \times \cdots \times r_m$.
Xn	A vector predictor of dimension p.
B_est	The tensor regression coefficient estimator for tensor response regression.

Value

P_OLS	The p-value of OLS estimator.
P_val	The p-value of B_est.
se_mat	The standard error matrix of $\mathrm{vec}(\hat{B})$.

TPR *Tensor Predictor Regression*

Description

This function is used for estimation of tensor predictor regression. The avaiable method including standard OLS type estimation, PLS type of estimation as well as envelope estimation with 1D and ECD approaches.

Usage

```
TPR(Yn, Xn, u, method)
```

Arguments

Yn	The response matrix of dimension $r \times n$, where n is the sample size.
Xn	The response tensor instance or dimension $p_1 \times p_2 \times \cdots \times p_m \times n$, where n is the sample size.
u	The dimension of envelope subspace. $u = (u_1, \cdots, u_m)$.
method	The method used for estimation of tensor response regression. There are four possible choices.

- `"standard"`: The standard OLS type estimation.
- `"1D"`: Envelope estimation with one dimenaional optimization approaches by 1D algorithm.
- `"ECD"`: Envelope estimation with one dimenaional optimization approaches by ECD algorithm.
- `"PLS"`: The SIMPLS-type estimation without mannifold optimization.

Value

Bhat	The estimation of regression coefficient tensor.
Gamma_hat	The estimation of envelope subspace basis.
Sig	A matrix lists of $\mathbf{\Sigma}_{\mathrm{xk}}$, where $\mathbf{\Sigma}_{\mathrm{x}} = \mathbf{\Sigma}_{\mathrm{xm}} \otimes \cdots \otimes \mathbf{\Sigma}_{\mathrm{x1}}$.

Note

This function supports tensor predictor regression for 1-dimensional, 2-dimensional or 3-dimensional tensor predictor in current version.

Examples

```
rm(list = ls())

p <- c(10, 10, 10)
u <- c(1, 1, 1)
m <- 3; r <- 5; n <- 200
eta <- array(runif(prod(u,r)), c(u,r))
eta <- as.tensor(eta)

Gamma <- Gamma0 <- Omega <- Omega0 <- Sig <- Sigsqrtm <- NULL
for(i in 1:m) {
  tmp <- matrix(runif(p[i]*u[i]), p[i], u[i])
  Gamma[[i]] <- qr.Q(qr(tmp))
  Gamma0[[i]] <- qr.Q(qr(tmp), complete=TRUE)[, (u[i]+1):p[i]]
  Omega[[i]] <- diag(u[i])
  Omega0[[i]] <- 0.01*diag(p[i]-u[i])
  Sig[[i]] <- Gamma[[i]] %*% Omega[[i]] %*% t(Gamma[[i]])+
    Gamma0[[i]] %*% Omega0[[i]] %*% t(Gamma0[[i]])
  Sig[[i]] <- 2*Sig[[i]]/norm(Sig[[i]], type="F")
  Sigsqrtm[[i]] <- pracma::sqrtm(Sig[[i]])$B
}
```

```
B <- ttl(eta,Gamma, ms = c(1:m))
A <- matrix(runif(r^2), r, r)
SigY <- A %*% t(A)
SigY <- SigY/norm(SigY, type="F")


##generate data
Epsilon <- MASS::mvrnorm(n, mu=rep(0, r), Sigma=SigY)
tmp2 <- array(rnorm(prod(p, n)), c(p, n))
Xn <- as.tensor(tmp2)
Xn <- ttl(Xn, Sigsqrtm, ms = c(1:m))
vecXn <- matrix(Xn@data, prod(p), n)
Y_tmp <- matrix(NA, r, 200)
tmp <- array(NA, c(p, r))
for (j in 1:n) {
  for (s in 1:r) {
    tmp[, , , s] <-  B@data[, , , s]*Xn@data[, , , j]
  }
  Y_tmp[, j] <- apply(tmp, 4, sum)
}
Yn <-  Y_tmp + t(Epsilon)


res_ECD = TPR(Yn, Xn, u, method="ECD")
res_1D = TPR(Yn, Xn, u, method="1D")
res_pls = TPR(Yn, Xn, u, method="PLS")
res_std = TPR(Yn, Xn, u, method="standard")

rTensor::fnorm(B-res_ECD$Bhat)
rTensor::fnorm(B-res_1D$Bhat)
rTensor::fnorm(B-res_pls$Bhat)
rTensor::fnorm(B-res_std$Bhat)
```

TRR *Tensor response regression*

Description

This function is used for estimation of tensor response regression. The avaiable method including standard OLS type estimation, PLS type of estimation as well as envelope estimation with 1D and ECD approaches.

Usage

```
TRR(Yn, Xn, u, method)
```

Arguments

Yn	The response tensor instance or dimension $r_1 \times r_2 \times \cdots \times r_m \times n$, where n is the sample size.
Xn	The predictor matrix of dimension $p \times n$.
u	The dimension of envelope subspace. $u = (u_1, \cdots, u_m)$.
method	The method used for estimation of tensor response regression. There are four possible choices. • "standard": The standard OLS type estimation. • "1D": Envelope estimation with one dimenaional optimization approaches by 1D algorithm. • "ECD": Envelope estimation with one dimenaional optimization approaches by ECD algorithm. • "PLS": The SIMPLS-type estimation without mannifold optimization.

Value

Bhat	The estimation of regression coefficient tensor.
Gamma_hat	The estimation of envelope subspace basis.
Sig	A matrix lists of $\mathbf{\Sigma}_k$, where $\mathbf{\Sigma} = \mathbf{\Sigma}_m \otimes \cdots \otimes \mathbf{\Sigma}_1$.

Examples

```
rm(list=ls())

r <- c(10, 10, 10)
m <- length(r)
u <- c(2, 2, 2)
p <- 5
n <- 100


set.seed(1)
eta <- array(runif(prod(u)*p), c(u, p))
eta <- as.tensor(eta)

Gamma <- Gamma0 <- Omega <- Omega0 <- Sig <- Sigsqrtm <- NULL
for (i in 1:m){
  tmp <- matrix(runif(r[i]*u[i]), r[i], u[i])
  Gamma[[i]] <- qr.Q(qr(tmp))
  Gamma0[[i]] <- qr.Q(qr(Gamma[[i]]),complete=TRUE)[,(u[i]+1):r[i]]

  A <- matrix(runif(u[i]^2), u[i], u[i])
  Omega[[i]] <- A %*% t(A)
  A <- matrix(runif((r[i]-u[i])^2), (r[i]-u[i]), (r[i]-u[i]))
  Omega0[[i]] <- A %*% t(A)
  Sig[[i]] <- Gamma[[i]] %*% Omega[[i]] %*% t(Gamma[[i]])+
    Gamma0[[i]] %*% Omega0[[i]] %*% t(Gamma0[[i]])
  Sig[[i]] <- 10*Sig[[i]]/norm(Sig[[i]], type="F")+0.01*diag(r[i])
```

```
  Sigsqrtm[[i]] <- pracma::sqrtm(Sig[[i]])$B
}
B <- ttl(eta, Gamma, ms=1:m)
Xn <- matrix(rnorm(p*n), p, n)
Xn_inv <- MASS::ginv(Xn %*% t(Xn)) %*% Xn
Epsilon <- array(rnorm(prod(r)*n), c(r, n))
Epsilon <- as.tensor(Epsilon)
Epsilon <- ttl(Epsilon, Sigsqrtm, ms=1:m)
Yn <- Epsilon + ttm(B, t(Xn), m+1)

res_ECD = TRR(Yn, Xn, u, method="ECD")
res_1D = TRR(Yn, Xn, u, method="1D")
res_pls = TRR(Yn, Xn, u, method="PLS")
res_std = TRR(Yn, Xn, u, method="standard")

rTensor::fnorm(B-res_ECD$Bhat)
rTensor::fnorm(B-res_1D$Bhat)
rTensor::fnorm(B-res_pls$Bhat)
rTensor::fnorm(B-res_std$Bhat)
```

`ttt` *Inner product of tensor X and Y*

Description

`ttt(X,Y,dims)` computes the inner product of tensors `X` and `Y` in the dimensions specified by the vector `dims`. The sizes of the dimensions specified by `dims` must match, which is `size(X, dims)` must equal `size(Y, dims)`.

Usage

```
ttt(X, Y, dims)
```

Arguments

`X`	A tensor instance.
`Y`	A tensor instance.
`dims`	The indices of the modes to map onto the column space.

Value

`mat`	The inner product of tensors X and Y in the dimensions specified by the vector dims, which is a matrix.

Examples

```
X = as.tensor(array(runif(24), c(3, 4, 2)));
Y = as.tensor(array(runif(24), c(3, 4, 2)));
Z = ttt(X, Y, 1:2)
```

Index