Package 'TransGraph'

October 19, 2023

Type Package

Title Transfer Graph Learning

Version 1.0.1

Maintainer Mingyang Ren <renmingyang17@mails.ucas.ac.cn>

Description Transfer learning, aiming to use auxiliary domains to help improve learning of the target domain of interest when multiple heterogeneous datasets are available, has always been a hot topic in statistical machine learning. The recent transfer learning methods with statistical guarantees mainly focus on the overall parameter transfer for supervised models in the ideal case with the informative auxiliary domains with overall similarity. In contrast, transfer learning for unsupervised graph learning is in its infancy and largely follows the idea of overall parameter transfer as for supervised learning.
In this package, the transfer learning for several complex graphical models is implemented, including Tensor Gaussian graphical models, non-Gaussian directed acyclic graph (DAG), and Gaussian graphical mixture models. Notably, this package promotes local transfer at node-level and subgroup-level in DAG structural learning and Gaussian graphical mixture models, respectively, which are more flexible and robust than the existing overall parameter transfer. As by-products, transfer learning for undirected graphical model (precision matrix) via D-trace loss, transfer learning for mean vector estimation, and single non-Gaussian learning via topological layer method are also included in this package.
Moreover, the aggregation of auxiliary information is an important issue in transfer learning, and this package provides multiple user-friendly aggregation methods, including sample weighting, similarity weighting, and most informative selection.
Reference:
Ren, M., Zhen Y., and Wang J. (2022) <arXiv:2211.09391> ``Transfer learning for tensor graphical models".
Ren, M., He X., and Wang J. (2023) <arXiv:2310.10239> ``Structural transfer learning of non-Gaussian DAG".
Zhao, R., He X., and Wang J. (2022) <https://jmlr.org/papers/v23/21-1173.html> ``Learning linear non-Gaussian directed acyclic graph with diverging number of nodes".

License GPL-2

Encoding UTF-8

Imports MASS, rTensor, Tlasso, glasso, clime, doParallel, expm, HeteroGGM, dcov, huge, EvaluationMeasures

RoxygenNote 7.2.3

Depends R (>= 3.5.0)

Suggests knitr, rmarkdown

VignetteBuilder knitr, rmarkdown

NeedsCompilation no

Author Mingyang Ren [aut, cre] (<https://orcid.org/0000-0002-8061-9940>),
Ruixuan Zhao [aut],
Xin He [aut],
Junhui Wang [aut]

Repository CRAN

Date/Publication 2023-10-19 10:40:05 UTC

R topics documented:

`Evaluation.DAG` *Evaluation function for the estimated DAG.*

Description

Evaluation function for the estimated DAG.

Usage

```
Evaluation.DAG(estimated.adjace, true.adjace, type.adj=2)
```

Arguments

`estimated.adjace`	The target data, a n * p matrix, where n is the sample size and p is data dimension.
`true.adjace`	The auxiliary data in K auxiliary domains, a list with K elements, each of which is a nk * p matrix, where nk is the sample size of the k-th auxiliary domain.
`type.adj`	The type of adjacency matrix. 1: the entries of matrix contains just two value, 0 and 1, which indicate the existence of edges; 2 (default): the matrix also measures connection strength, and 0 means no edge.

Value

A result list including Recall, FDR, F1score, MCC, Hamming Distance,and estimated error of adjacency matrix on F-norm.

Author(s)

Ruixaun Zhao ruixuanzhao2-c@my.cityu.edu.hk.

References

Zhao, R., He X., and Wang J. (2022). Learning linear non-Gaussian directed acyclic graph with diverging number of nodes. Journal of Machine Learning Research.

`Evaluation.GGM` *Evaluation function for the estimated GGM.*

Description

Evaluation function for the estimated GGM.

Usage

```
Evaluation.GGM(est.precision, true.precision)
```

Arguments

`est.precision`	The estimated precision matrix.
`true.precision`	The true precision matrix.

Value

A result list including Recall, FDR, F1score, MCC, Hamming Distance,and estimated error of adjacency matrix on F-norm.

Author(s)

Mingyang Ren renmingyang17@mails.ucas.ac.cn.

`layer_adj` *The function of converting the adjacency matrix into the topological layer.*

Description

The function of converting the adjacency matrix into the topological layer.

Usage

```
layer_adj(true_adjace)
```

Arguments

`true_adjace`	a p * p adjacency matrix

Value

Layer_true: a p * 2 matrix to store the information of layer. The first column is the node label, and the second column is the corresponding layer labels.

Author(s)

Mingyang Ren renmingyang17@mails.ucas.ac.cn.

References

Zhao, R., He X., and Wang J. (2022). Learning linear non-Gaussian directed acyclic graph with diverging number of nodes. Journal of Machine Learning Research.

`tensor.GGM.trans` *Transfer learning for tensor graphical models.*

Description

The main function for Transfer learning for tensor graphical models.

Usage

```
tensor.GGM.trans(t.data, A.data, A.lambda, A.orac = NULL, c=0.6,
                         t.lambda.int.trans=NULL, t.lambda.int.aggr=NULL,
                         theta.algm="cd", cov.select="inverse",
                         cov.select.agg.size = "inverse",
                         cov.select.agg.diff = "tensor.prod",
                         symmetric = TRUE, init.method="Tlasso",
                         init.method.aux="Tlasso", mode.set = NULL,
```

```
                    init.iter.Tlasso=2, cn.lam2=seq(0.1,2,length.out =10),
                    c.lam.Tlasso=20, c.lam.sepa=20, adjust.BIC=FALSE,
                    normalize = TRUE, inti.the=TRUE, sel.ind="fit")
```

Arguments

t.data	The tensor data in the target domain, a p1 * p2 * ... * pM * n array, where n is the sample size and pm is dimension of the m-th tensor mode. M should be larger than 2.
A.data	The tensor data in auxiliary domains, a list with K elements, each of which is a p1 * p2 * ... * pM * nk array, where nk is the sample size of the k-th auxiliary domain.
A.lambda	The tuning parameters used for initialization in auxiliary domains, a list with K elements, each of which is a M-dimensional vector corresponding to M modes.
A.orac	The set of informative auxiliary domains, and the default setting is NULL, which means that no set is specified.
c	The c of subjects in the target domain are used for initialization of the transfer learning, and the remaining 1-c of subjects are used for the model selection step. The default setting is 0.8.
t.lambda.int.trans	The tuning parameters used for initialization in the target domain (based on c subjects used for transfer learning), that is, the tuning lambda for Tlasso (PAMI, 2020) & Separable method (JCGS, 2022)
t.lambda.int.aggr	The tuning parameters used for initialization in the target domain (based on 1-c subjects used for the model selection step).
theta.algm	The optimization algorithm used to solve $\widehat{\Omega}$ in step 2(b), which can be selected as "admm" (ADMM algorithm) or "cd" (coordinate descent).
cov.select	Methods used to calculate covariance matrices for initialization in both target and auxiliary domains, which can be selected as "tensor.prod" (tensor product based on tensor subject and the initial estimate of the precision matrix, TPAMI, 2020) and "inverse" (direct inversion of the initial estimate of the precision matrix)
cov.select.agg.size	Methods used to calculate covariance matrices for model selection step in the target domain.
cov.select.agg.diff	Methods used to calculate covariance matrices for model selection step in the target domain.
symmetric	Whether to symmetrize the final estimated precision matrices, and the default is True.
init.method	The initialization method for tensor precision matrices in the target domain, which can be selected as "Tlasso" (PAMI, 2020) & "sepa" (Separable method, JCGS, 2022). Note that the "sepa" method has not been included in the current version of this R package to circumvent code ownership issues.

`init.method.aux`
: The initialization method for tensor precision matrices in auxiliary domains.

`mode.set`
: Whether to estimate only the specified mode, and the default setting is NULL, which means estimating all mode.

`init.iter.Tlasso`
: The number of maximal iteration when using Tlasso for initialization, default is 2.

`cn.lam2`
: The coefficient set in tuning parameters used to solve $\widehat{\Omega}$ in step 2(b), default is seq(0.1,1,length.out =10).

`c.lam.Tlasso`
: The coefficient in tuning parameters for initialization (when using Tlasso): $c.lam.Tlasso*\sqrt{(pm*\log(pm)/(n*p1*...*pM))}$, default is 20 suggested in (PAMI, 2020).

`c.lam.sepa`
: The coefficient in tuning parameters for initialization (when using sepa): $c.lam.sepa*\sqrt{(pm*\log(pm)/(n*p1*...*pM))}$.

`adjust.BIC`
: Whether to use the adjusted BIC to select lambda2, the default setting is F.

`normalize`
: The normalization method of precision matrix. When using Tlasso, $\Omega_{11}=1$ if normalize = F and $\|\Omega_{11}\|_F=1$ if normalize = T. Default value is T.

`inti.the`
: T: the initial values in Step 2(b) is Omega0.

`sel.ind`
: The approach to model selection, which can be selected from c("fit", "predict").

Value

A result list including:

Omega.list The final estimation result of the target precision matrices after the model selection of transfer learning-based estimation and initial estimation (in which the initial covariance matrices of auxiliary domains is weighted by sample sizes).

Omega.sym.list The symmetrized final estimation result in Omega.list.

Omega.list.diff The final estimation result of the target precision matrices after the model selection of transfer learning-based estimation and initial estimation (in which the initial covariance matrices of auxiliary domains is weighted by the differences with the target domain).

Omega.sym.list.diff The symmetrized final estimation result in Omega.list.diff.

res.trans.list Transfer learning-based estimation results.

Author(s)

Mingyang Ren renmingyang17@mails.ucas.ac.cn, Yaoming Zhen, and Junhui Wang

References

Ren, M., Zhen Y., and Wang J. (2022). Transfer learning for tensor graphical models.

Examples

```
library(TransGraph)
library(Tlasso)
# load example data from github repository
# Please refer to https://github.com/Ren-Mingyang/example_data_TransGraph
# for detailed data information
githublink = "https://github.com/Ren-Mingyang/example_data_TransGraph/"
load(url(paste0(githublink,"raw/main/example.data.tensorGGM.RData")))
t.data = example.data$t.data
A.data = example.data$A.data
t.Omega.true.list = example.data$t.Omega.true.list
normalize = TRUE

K = length(A.data)
p.vec = dim(t.data)
M = length(p.vec) - 1
n = p.vec[M+1]
p.vec = p.vec[1:M]
tla.lambda = 20*sqrt( p.vec*log(p.vec) / ( n * prod(p.vec) ))
A.lambda = list()
for (k in 1:K) {
  A.lambda[[k]] = 20*sqrt( log(p.vec) / ( dim(A.data[[k]])[M+1] * prod(p.vec) ))
}

res.final = tensor.GGM.trans(t.data, A.data, A.lambda, normalize = normalize)
Tlasso.Omega.list = Tlasso.fit(t.data, lambda.vec = tla.lambda,
                    norm.type = 1+as.numeric(normalize))

i.Omega = as.data.frame(t(unlist(est.analysis(res.final$Omega.list, t.Omega.true.list))))
i.Omega.diff = t(unlist(est.analysis(res.final$Omega.list.diff, t.Omega.true.list)))
i.Omega.diff = as.data.frame(i.Omega.diff)
i.Tlasso = as.data.frame(t(unlist(est.analysis(Tlasso.Omega.list, t.Omega.true.list))))
i.Omega.diff     # proposed.v
i.Omega          # proposed
i.Tlasso         # Tlasso
```

`Theta.est` *Sparse precision matrix estimation.*

Description

The fast sparse precision matrix estimation in step 2(b).

Usage

```
Theta.est(S.hat.A, delta.hat, lam2=0.1, Omega.hat0=NULL,
                n=100, max_iter=10, eps=1e-3, method = "cd")
```

Arguments

S.hat.A	The sample covariance matrix.
delta.hat	The divergence matrix estimated in step 2(a). If the precision matrix is estimated in the common case (Liu and Luo, 2015, JMVA), it can be set to zero matrix.
lam2	A float value, a tuning parameter.
Omega.hat0	The initial values of the precision matrix, which can be unspecified.
n	The sample size.
max_iter	Int, maximum number of cycles of the algorithm.
eps	A float value, algorithm termination threshold.
method	The optimization algorithm, which can be selected as "admm" (ADMM algorithm) or "cd" (coordinate descent).

Value

A result list including:

Theta.hat.m The optimal precision matrix.

BIC.summary The summary of BICs.

Theta.hat.list.m The precision matrices corresponding to a sequence of tuning parameters.

Author(s)

Mingyang Ren renmingyang17@mails.ucas.ac.cn.

References

Ren, M., Zhen Y., and Wang J. (2022). Transfer learning for tensor graphical models. Liu, W. and Luo X. (2015). Fast and adaptive sparse precision matrix estimation in high dimensions, Journal of Multivariate Analysis.

Examples

```
p = 20
n = 200
omega = diag(rep(1,p))
for (i in 1:p) {
  for (j in 1:p) {
    omega[i,j] = 0.3^(abs(i-j))*(abs(i-j) < 2)
  }
}
Sigma = solve(omega)
X = MASS::mvrnorm(n, rep(0,p), Sigma)
```

```
S.hat.A = cov(X)
delta.hat = diag(rep(1,p)) - diag(rep(1,p))
omega.hat = Theta.est(S.hat.A, delta.hat, lam2=0.2)
```

`Theta.tuning` *Sparse precision matrix estimation with tuning parameters.*

Description

The fast sparse precision matrix estimation in step 2(b).

Usage

```
Theta.tuning(lambda2, S.hat.A, delta.hat, Omega.hat0, n.A,
                    theta.algm="cd", adjust.BIC=FALSE)
```

Arguments

`lambda2`	A vector, a sequence of tuning parameters.
`S.hat.A`	The sample covariance matrix.
`delta.hat`	The divergence matrix estimated in step 2(a). If the precision matrix is estimated in the common case (Liu and Luo, 2015, JMVA), it can be set to zero matrix.
`Omega.hat0`	The initial values of the precision matrix.
`n.A`	The sample size.
`theta.algm`	The optimization algorithm used to solve $\widehat{\Omega}$ in step 2(b), which can be selected as "admm" (ADMM algorithm) or "cd" (coordinate descent).
`adjust.BIC`	Whether to use the adjusted BIC to select lambda2, the default setting is F.

Value

A result list including:

Theta.hat.m The optimal precision matrix.

BIC.summary The summary of BICs.

Theta.hat.list.m The precision matrices corresponding to a sequence of tuning parameters.

Author(s)

Mingyang Ren renmingyang17@mails.ucas.ac.cn.

References

Ren, M., Zhen Y., and Wang J. (2022). Transfer learning for tensor graphical models. Liu, W. and Luo X. (2015). Fast and adaptive sparse precision matrix estimation in high dimensions, Journal of Multivariate Analysis.

Examples

```
p = 20
n = 200
omega = diag(rep(1,p))
for (i in 1:p) {
  for (j in 1:p) {
    omega[i,j] = 0.3^(abs(i-j))*(abs(i-j) < 2)
  }
}
Sigma = solve(omega)
X = MASS::mvrnorm(n, rep(0,p), Sigma)
S.hat.A = cov(X)
delta.hat = diag(rep(1,p)) - diag(rep(1,p))
lambda2 = seq(0.1,0.5,length.out =10)
res = Theta.tuning(lambda2, S.hat.A, delta.hat, n.A=n)
omega.hat = res$Theta.hat.m
```

TLLiNGAM *Learning linear non-Gaussian DAG via topological layers.*

Description

Learning linear non-Gaussian DAG via topological layers.

Usage

```
TLLiNGAM (X, hardth=0.3, criti.val=0.01, precision.refit = TRUE,
                precision.method="glasso", B.refit=TRUE)
```

Arguments

X	The n * p sample matrix, where n is the sample size and p is data dimension.
hardth	The hard threshold of regression.
criti.val	The critical value of independence test based on distance covariance.
precision.refit	Whether to perform regression for re-fitting the coefficients in the precision matrix to improve estimation accuracy, after determining the non-zero elements of the precision matrix. The default is True.
precision.method	Methods for Estimating Precision Matrix, which can be selected from "glasso" and "CLIME".
B.refit	Whether to perform regression for re-fitting the coefficients in structural equation models to improve estimation accuracy, after determining the parent sets of all nodes. The default is True.

Value

A result list including:

A The information of layer.

B The coefficients in structural equation models.

Author(s)

Ruixuan Zhao ruixuanzhao2-c@my.cityu.edu.hk, Xin He, and Junhui Wang

References

Zhao, R., He X., and Wang J. (2022). Learning linear non-Gaussian directed acyclic graph with diverging number of nodes. Journal of Machine Learning Research.

Examples

```
library(TransGraph)
# load example data from github repository
# Please refer to https://github.com/Ren-Mingyang/example_data_TransGraph
# for detailed data information
githublink = "https://github.com/Ren-Mingyang/example_data_TransGraph/"
load(url(paste0(githublink,"raw/main/example.data.singleDAG.RData")))
true_adjace = example.data.singleDAG$true_adjace
t.data = example.data.singleDAG$X
res.single = TLLiNGAM(t.data)
Evaluation.DAG(res.single$B, true_adjace)$Eval_result
```

`trans.local.DAG` *Structural transfer learning of non-Gaussian DAG.*

Description

Structural transfer learning of non-Gaussian DAG.

Usage

```
trans.local.DAG(t.data, A.data, hardth=0.5, hardth.A=hardth, criti.val=0.01,
                      precision.method="glasso", precision.method.A = "CLIME",
                      cov.method="opt", cn.lam2=seq(1,2.5,length.out=10),
                      precision.refit=TRUE, ini.prec=TRUE, cut.off=TRUE,
                      preselect.aux=0, sel.type="L2")
```

Arguments

t.data	The target data, a n * p matrix, where n is the sample size and p is data dimension.
A.data	The auxiliary data in K auxiliary domains, a list with K elements, each of which is a nk * p matrix, where nk is the sample size of the k-th auxiliary domain.
hardth	The hard threshold of regression in the target domain.
hardth.A	The hard threshold of regression in the auxiliary domains.
criti.val	The critical value of independence test based on distance covariance, and the default setting is 0.01.
precision.method	The initial method of estimating the target precision matrix, which can be selected as "CLIME" or "glasso".
precision.method.A	The initial method of estimating the auxiliary precision matrices, which can be selected as "CLIME" or "glasso".
cov.method	The method of aggregating K auxiliary covariance matrices, which can be selected as "size" (the sum weighted by the sample sizes), "weight" (the sum weighted by the differences), or "opt" (select the optimal one).
cn.lam2	A vector or a float value: the coefficients set in tuning parameters used to solve the target precision matrix, default is cn.lam2*sqrt(log(p) / n).
precision.refit	Whether to perform regression for re-fitting the coefficients in the precision matrix to improve estimation accuracy, after determining the non-zero elements of the precision matrix. The default is True.
ini.prec	Whether to store the initial estimation of the precision matrix, and the default is True.
cut.off	Whether to truncate the finally estimated coefficients in the structural equation models at threshold "hardth", and the default is True.
preselect.aux	Whether to pre-select informative auxiliary domains based on the distance between initially estimated auxiliary and target parameters. The default is 0, which means that pre-selection will not be performed. If "preselect.aux" is specified as a real number greater than zero, then the threshold value is forpreselect.aux*s*sqrt(log(p) / n).
sel.type	If pre-selection should be performed, "sel.type" is the type of distance. The default is L2 norm, and can be specified as "L1" to use L1 norm.

Value

A result list including:

A The information of layer.

B The coefficients in structural equation models.

prec.res0 The results about estimating the prscision matrix via transfer learning.

prec.res0$Theta.hat The estimated prscision matrix via transfer learning.

prec.res0$Theta.hat0 The estimated prscision matrix based on the target domain only.

Author(s)

Mingyang Ren renmingyang17@mails.ucas.ac.cn, Xin He, and Junhui Wang

References

Ren, M., He X., and Wang J. (2023). Structural transfer learning of non-Gaussian DAG.

Examples

```
library(TransGraph)
# load example data from github repository
# Please refer to https://github.com/Ren-Mingyang/example_data_TransGraph
# for detailed data information
githublink = "https://github.com/Ren-Mingyang/example_data_TransGraph/"
load(url(paste0(githublink,"raw/main/example.data.DAG.RData")))
t.data = example.data.DAG$target.DAG.data$X
true_adjace = example.data.DAG$target.DAG.data$true_adjace
A.data = example.data.DAG$auxiliary.DAG.data$X.list.A

# transfer method
res.trans = trans.local.DAG(t.data, A.data)
# Topological Layer method-based single-task learning (JLMR, 2022)
res.single = TLLiNGAM(t.data)

Evaluation.DAG(res.trans$B, true_adjace)$Eval_result
Evaluation.DAG(res.single$B, true_adjace)$Eval_result
```

---

trans_GGMM *Transfer learning of high-dimensional Gaussian graphical mixture models.*

Description

Transfer learning of high-dimensional Gaussian graphical mixture models.

Usage

```
trans_GGMM(t.data, lambda.t, M, A.data, lambda.A.list, M.A.vec,
                 pseudo.cov="soft", cov.method="opt", cn.lam2=0.5, clambda.m=1,
                 theta.algm="cd", initial.selection="K-means", preselect.aux=0,
                  sel.type="L2", trace=FALSE )
```

Arguments

t.data	The target data, a n * p matrix, where n is the sample size and p is data dimension.
lambda.t	A list, the sequences of the tuning parameters (lambda1, lambda2, and lambda3) used in the initialization of the target domain.
M	Int, a selected upper bound of the true numbers of subgroups in the target domain.
A.data	The auxiliary data in K auxiliary domains, a list with K elements, each of which is a nk * p matrix, where nk is the sample size of the k-th auxiliary domain.
lambda.A.list	A list consisting of K lists, the k-th list is the sequences of the tuning parameters (lambda1, lambda2, and lambda3) used in the initialization of the k-th auxiliary domain.
M.A.vec	A vector composed of K integers, the k-th element is a selected upper bound of the true numbers of subgroups in the k-th auxiliary domain.
pseudo.cov	The method for calculating pseudo covariance matricex in auxiliary domains, which can be selected from "soft"(default, subgroups based on samples of soft clustering via posterior probability) and "hard" (subgroups based on samples of hard clustering).
cov.method	The method of aggregating K auxiliary covariance matrices, which can be selected as "size" (the sum weighted by the sample sizes), "weight" (the sum weighted by the differences) or "opt" (select the optimal one).
cn.lam2	A vector or a float value: the coefficients set in tuning parameters used to solve the target precision matrix, default is cn.lam2*sqrt(log(p) / n).
clambda.m	The coefficients set in tuning parameters used in transfer learning for mean eatimation, and the default setting is clambda.m * sqrt(log(p) / n).
theta.algm	The optimization algorithm used to solve the precision, which can be selected as "admm" (ADMM algorithm) or "cd" (coordinate descent).
initial.selection	The different initial values from two clustering methods, which can be selected from c("K-means","dbscan").
preselect.aux	Whether to pre-select informative auxiliary domains based on the distance between initially estimated auxiliary and target parameters. The default is 0, which means that pre-selection will not be performed. If "preselect.aux" is specified as a real number greater than zero, then the threshold value is forpreselect.aux*s*sqrt(log(p) / n).
sel.type	If pre-selection should be performed, "sel.type" is the type of distance. The default is L2 norm, and can be specified as "L1" to use L1 norm.
trace	The logical variable, whether or not to output the number of identified subgroups during the search for parameters in the initialization.

Value

A result list including:

res.target A list including transfer learning results of the target domain.

res.target$opt_Mu_hat The final estimation of means in all detected subgroups via transfer learning.

res.target$opt_Theta_hat The final estimation of precision matrices in all detected subgroups via transfer learning.

res.target0 A list including initial results of the target domain.

res.target0$opt_Mu_hat The initial estimation of means in all detected subgroups.

res.target0$opt_Theta_hat The initial estimation of precision matrices in all detected subgroups.

t.res A list including results of the transfer precision matrix for each subgroup.

Author(s)

Mingyang Ren renmingyang17@mails.ucas.ac.cn.

References

Ren, M. and Wang J. (2023). Local transfer learning of Gaussian graphical mixture models.

Examples

```
"Will be supplemented in the next version."
```

`trans_mean` *Transfer learning for mean estimation.*

Description

Transfer learning for mean estimation.

Usage

```
trans_mean(t.mean.m, A.mean, n, clambda=1)
```

Arguments

`t.mean.m`	The estimated target p-dimensional mean vector, where p is mean dimension.
`A.mean`	A K*p matrix with the k-th row being the estimated p-dimensional mean vector of the k-th auxiliary domain.
`n`	The target sample size.
`clambda`	The coefficients set in tuning parameters used in transfer learning for mean eatimation, and the default setting is clambda.m * sqrt(log(p) / n).

Value

t.mean.m.hat: The transfer learning estimation of the target p-dimensional mean vector.

Author(s)

Mingyang Ren renmingyang17@mails.ucas.ac.cn.

References

Ren, M. and Wang J. (2023). Local transfer learning of Gaussian graphical mixture models.

`trans_precision` *Transfer learning for vector-valued precision matrix (graphical model).*

Description

The transfer learning for vector-valued precision matrix via D-trace loss method.

Usage

```
trans_precision(t.data=NULL, A.data=NULL, precision.method="CLIME",
                        cov.method="opt", cn.lam2=seq(1,2.5,length.out=10),
                        theta.algm="cd", adjust.BIC=FALSE, symmetry=TRUE,
                        preselect.aux=0, sel.type="L2", input.A.cov=FALSE,
                        A.cov=NULL, nA.vec=NULL, t.Theta.hat0=NULL,
                        t.n=NULL, correlation=FALSE)
```

Arguments

`t.data`	The target data, a n * p matrix, where n is the sample size and p is data dimension.
`A.data`	The auxiliary data in K auxiliary domains, a list with K elements, each of which is a nk * p matrix, where nk is the sample size of the k-th auxiliary domain.
`precision.method`	The initial method of estimating the target precision matrix, which can be selected as "CLIME" or "glasso".
`cov.method`	The method of aggregating K auxiliary covariance matrices, which can be selected as "size" (the sum weighted by the sample sizes), "weight" (the sum weighted by the differences) or "opt" (select the optimal one).
`cn.lam2`	A vector or a float value: the coefficients set in tuning parameters used to solve the target precision matrix, default is cn.lam2*sqrt(log(p) / n).
`theta.algm`	The optimization algorithm used to solve the precision, which can be selected as "admm" (ADMM algorithm) or "cd" (coordinate descent).
`adjust.BIC`	Whether to use the adjusted BIC to select lambda2, the default setting is FALSE.
`symmetry`	Whether to symmetrize the final estimated precision matrices, and the default is True.

`preselect.aux` Whether to pre-select informative auxiliary domains based on the distance between initially estimated auxiliary and target parameters. The default is 0, which means that pre-selection will not be performed. If "preselect.aux" is specified as a real number greater than zero, then the threshold value is forpreselect.aux*s*sqrt(log(p) / n).

`sel.type` If pre-selection should be performed, "sel.type" is the type of distance. The default is L2 norm, and can be specified as "L1" to use L1 norm.

`input.A.cov` Whether to input the covariance matrices of the auxiliary domains. The default setting is FALSE, which means that the raw data of the auxiliary domain is input, and the covariance will be calculated within this function. If input.A.cov=T, then the calculated covariance matrices must be input through parameter "A.cov", and parameter "A.data" can be defaulted at this time. This setting is suitable for situations where raw data cannot be obtained but the covariance matrix can be obtained.

`A.cov` If input.A.cov=T, the "A.cov" must be auxiliary covariance matrices in K auxiliary domains, a list with K elements, each of which is a p * p matrix.

`nA.vec` If input.A.cov=T, the "nA.vec" must be a vector consisting of sample sizes of K auxiliary domains.

`t.Theta.hat0` Whether to input the estimated target precision matrix based on the target domain only, and the default setting is NULL. If "t.Theta.hat0" is specified as an estimated precision matrix, it will not be recalculated in the initialization phase. This parameter mainly plays a role in transfer learning of GGMMs.

`t.n` Whether to input the target sample size, and the default setting is NULL. This parameter mainly plays a role in transfer learning of GGMMs.

`correlation` Whether to use correlation matrix for initial parameters in both target and auxiliary domains. The default setting is F.

Value

A result list including:

Theta.hat The target precision matrix via transfer learning.

Theta.hat0 The initial target precision matrix.

k.check The number of the optimal auxiliary domain.

N The minimum sample size for auxiliary domain.

Author(s)

Mingyang Ren renmingyang17@mails.ucas.ac.cn.

References

Ren, M., Zhen Y., and Wang J. (2022). Transfer learning for tensor graphical models. Ren, M., He X., and Wang J. (2023). Structural transfer learning of non-Gaussian DAG.

Examples

```
library(TransGraph)
# load example data from github repository
# Please refer to https://github.com/Ren-Mingyang/example_data_TransGraph
# for detailed data information
githublink = "https://github.com/Ren-Mingyang/example_data_TransGraph/"
load(url(paste0(githublink,"raw/main/example.data.GGM.RData")))
t.data = example.data.GGM$target.list$t.data
t.precision = example.data.GGM$target.list$t.precision
A.data = example.data.GGM$A.data
A.data.infor = example.data.GGM$A.data.infor

# using all auxiliary domains
res.trans.weight = trans_precision(t.data, A.data, cov.method="weight")
res.trans.opt = trans_precision(t.data, A.data, cov.method="opt")
res.trans.size = trans_precision(t.data, A.data, cov.method="size")
Theta.trans.weight = res.trans.weight$Theta.hat
Theta.trans.opt = res.trans.opt$Theta.hat
Theta.trans.size = res.trans.size$Theta.hat
Theta.single = res.trans.weight$Theta.hat0  # initial rough estimation via the target domain
Theta.single[abs(Theta.single)<0.0001] = 0

Evaluation.GGM(Theta.single, t.precision)
Evaluation.GGM(Theta.trans.weight, t.precision)
Evaluation.GGM(Theta.trans.opt, t.precision)
Evaluation.GGM(Theta.trans.size, t.precision)

# using informative auxiliary domains
res.trans.size.oracle = trans_precision(t.data, A.data.infor, cov.method="size")
Evaluation.GGM(res.trans.size.oracle$Theta.hat, t.precision)
```

Index