Package: HDMAADMM (via r-universe)

September 2, 2024

Type Package

Title ADMM for High-Dimensional Mediation Models

Version 0.1.0

Date 2024-08-30

Maintainer Pei-Shan Yen <peishan0824@gmail.com>

Description We use the Alternating Direction Method of Multipliers (ADMM) for parameter estimation in high-dimensional, single-modality mediation models. To improve the sensitivity and specificity of estimated mediation effects, we offer the sure independence screening (SIS) function for dimension reduction. The available penalty options include Lasso, Elastic Net, Pathway Lasso, and Network-constrained Penalty. The methods employed in the package are based on Boyd, S., Parikh, N., Chu, E., Peleato, B., & Eckstein, J. (2011). <doi:10.1561/2200000016>, Fan, J., & Lv, J. (2008) <doi:10.1111/j.1467-9868.2008.00674.x>, Li, C., & Li, H. (2008) <doi:10.1093/bioinformatics/btn081>, Tibshirani, R. (1996) <doi:10.1111/j.2517-6161.1996.tb02080.x>, Zhao, Y., & Luo, X. (2022) <doi:10.4310/21-sii673>, and Zou, H., & Hastie, T. (2005) <doi:10.1111/j.1467-9868.2005.00503.x>.

License MIT + file LICENSE

Depends R (>= 4.0.0)

Imports Rcpp (>= 1.0.0), dqrng, RcppEigen

LinkingTo Rcpp, RcppEigen

Suggests testthat, roxygen2

Encoding UTF-8

URL https://github.com/psyen0824/HDMAADMM

BugReports https://github.com/psyen0824/HDMAADMM/issues

Roxygen list(markdown = TRUE)

RoxygenNote 7.3.2

Repository https://psyen0824.r-universe.dev

RemoteUrl https://github.com/psyen0824/hdmaadmm

RemoteRef HEAD

RemoteSha 5ab01e892464785b9b570e09801d6a348f6f53dd

Contents

`HDMAADMM-package` `HDMAADMM` *Package*

Description

This package enables the estimation of single-modality high-dimensional mediation models. We employ penalized maximum likelihood and solve the estimation using the Alternating Direction Method of Multipliers (ADMM) to provide high-dimensional mediator estimates. To improve the sensitivity and specificity of non-zero mediators, we offer the sure independence screening (SIS) function for dimension reduction. The available penalty options include Lasso, Elastic Net, Pathway Lasso, and Network-constrained Penalty.

Author(s)

Maintainer: Pei-Shan Yen <`peishan0824@gmail.com`> (ORCID)

Authors:

- Ching-Chuan Chen <`zw12356@gmail.com`> (ORCID)

References

1. Tibshirani, R. (1996). Regression Shrinkage and Selection via the Lasso. Journal of the Royal Statistical Society. Series B (Methodological), 58(1), 267–288.
2. Zou, H., & Hastie, T. (2005). Regularization and Variable Selection via the Elastic Net. Journal of the Royal Statistical Society. Series B (Statistical Methodology), 67(2), 301–320.
3. Li, C., Li, H. (2008). Network-constrained regularization and variable selection for analysis of genomic data, Bioinformatics, 24(9), 1175–1182,
4. Zhao, Y., & Luo, X. (2022). Pathway Lasso: pathway estimation and selection with high-dimensional mediators. Statistics and its interface, 15(1), 39.

See Also

Useful links:

- https://github.com/psyen0824/HDMAADMM
- Report bugs at https://github.com/psyen0824/HDMAADMM/issues

cvSingleModalityAdmm *Cross Validation for High-dimensional Single Mediation Models*

Description

Cross Validation for High-dimensional Single Mediation Models

Usage

```
cvSingleModalityAdmm(
  X,
  Y,
  M1,
  numFolds = 10,
  typeMeasure = "rmse",
  lambda1a,
  lambda1b,
  lambda1g,
  lambda2a,
  lambda2b,
  rho = 1,
  penalty = "ElasticNet",
  penaltyParameterList = list(),
  SIS = FALSE,
  SISThreshold = 2,
  maxIter = 3000,
  tol = 1e-04,
  verbose = FALSE,
  debug = FALSE
)
```

Arguments

`X` The matrix of independent variables (exposure/treatment/group).

`Y` The vector of dependent variable (outcome response).

`M1` The single-modality mediator.

`numFolds` The number of folds. The default is 10. Although nfolds can be as large as the sample size (leave-one-out CV), it is not recommended for large datasets. Smallest value allowable is nfolds=3.

`typeMeasure` Default is "rmse".

`rho, lambda1g, lambda1a, lambda1b, lambda2a, lambda2b, penaltyParameterList`
: Allow to put sequences for each parameter. Please refer to the function, `singleModalityAdmm` for the details.

`penalty, SIS, SISThreshold, maxIter, tol, verbose, debug`
: Please refer to the function, `singleModalityAdmm`.

Value

An `cvSingleModalityAdmm` object which is a matrix containing all the combinations of parameter sequences with an additional column called `measure`.

Examples

```
## Generate Empirical Data
simuData <- modalityMediationDataGen(seed = 20231201)

## Cross-Validation for ElasticNet penalty
cvElasticNetResults <- cvSingleModalityAdmm(
  X = simuData$MediData$X, Y = simuData$MediData$Y, M1 = simuData$MediData$M1,
  numFolds = 5, typeMeasure = "rmse",
  rho = c(0.9, 1, 1.1), lambda1a = c(0.1, 0.5, 1), lambda1b = c(0.1, 0.3),
  lambda1g = c(1, 2), lambda2a = c(0.5, 1), lambda2b = c(0.5, 1),
  penalty = "ElasticNet"
)

## Cross-Validation for Pathway Lasso penalty (lambda2a, lambda2b are not tuned.)
cvPathwayLassoResults <- cvSingleModalityAdmm(
  X = simuData$MediData$X, Y = simuData$MediData$Y, M1 = simuData$MediData$M1,
  numFolds = 5, typeMeasure = "rmse",
  rho = c(0.9, 1, 1.1), lambda1a = c(0.1, 0.5, 1), lambda1b = c(0.1, 0.3),
  lambda1g = c(1, 2), lambda2a = 1, lambda2b = 1,
  penalty = "PathwayLasso", penaltyParameterList = list(kappa = c(0.5, 1), nu = c(1, 2))
)
```

`fitted.SingleModalityAdmm` *Fitted Response of SingleModalityAdmm Fits*

Description

Fitted Response of SingleModalityAdmm Fits

Usage

```
## S3 method for class 'SingleModalityAdmm'
fitted(object, ...)
```

Arguments

`object` A fitted obejct of class inheriting from `SingleModalityAdmm`.

`...` further arguments passed to or from other methods.

Value

fitted.SingleModalityAdmm returns a vector which is fitted values.

`generateLaplacianMatrix` *Function Generate Laplacian Matrix*

Description

Function Generate Laplacian Matrix

Usage

```
generateLaplacianMatrix(X, Y, M1, type = "beta", interaction = FALSE)
```

Arguments

`X, Y, M1` The input data for the single modality mediation model. Details see `singleModalityAdmm`.

`type` A string to specify the generated Laplacian matrix is for α or β.

`interaction` A logical value to specify whether to refer interation effect.

`modalityMediationDataGen` *Data Generation for High-Dimensional Mediation Model*

Description

Data Generation for High-Dimensional Mediation Model

Usage

```
modalityMediationDataGen(
  n = 100,
  p = 50,
  sigmaY = 1,
  sizeNonZero = c(3, 3, 4),
  alphaMean = c(6, 4, 2),
  alphaSd = 0.1,
  betaMean = c(6, 4, 2),
  betaSd = 0.1,
  sigmaM1 = NULL,
  gamma = 3,
  laplacianA = TRUE,
  laplacianB = TRUE,
  laplacianInteractionA = FALSE,
  laplacianInteractionB = FALSE,
  seed = 20231201
)
```

Arguments

`n` The number of subjects for the high-dimensional mediation model)

`p` The number of high-dimensional mediators.

`sigmaY` The argument "sigmaY" represents the standard deviation (SD) of the error distribution for the dependent variable.

`sizeNonZero` The number of nonzero mediators. Here, we provide simulated scenarios that could produce large, medium, and small mediated effects, generating from a normal distribution.

`alphaMean`, `alphaSd`
The mean and SD vector of the effect between the mediator and independent variable.

`betaMean`, `betaSd`
The mean and SD vector of the effect between the mediator and dependent variable.

`sigmaM1` The covariance matrix of the error distribution among mediators. Default is `diag(p)`.

`gamma` The true value of direct effect.

`laplacianA`, `laplacianB`
Default is TRUE. These two logical values indicate whether to generate the laplacian matrix for network penalty. Details see `generateLaplacianMatrix`.

`laplacianInteractionA`, `laplacianInteractionB`
A logical value to specify to use interaction term. Details see `generateLaplacianMatrix`.

`seed` The random seed. Default is NULL to use the current seed.

Value

A object with three elements.

- MediData: The simulated data for high-dimensional mediation model.
- MediPara: The true value for mediated effect and direct effect.
- Info : The output includes random seed, parameter setting, and Laplacian matrix for generating mediation model.

Examples

```
simuData <- modalityMediationDataGen(seed = 20231201)
```

`predict.SingleModalityAdmm` *Predict Method for SingleModalityAdmm Fits*

Description

Predict Method for SingleModalityAdmm Fits

Usage

```
## S3 method for class 'SingleModalityAdmm'
predict(object, newdata, ...)
```

Arguments

`object`	A fitted obejct of class inheriting from `SingleModalityAdmm`.
`newdata`	Default is `NULL`. A matrix with variables to predict.
`...`	further arguments passed to or from other methods.

Value

predict.SingleModalityAdmm returns a vector which is the predicted values based on `newdata`.

`singleModalityAdmm` *High-dimensional Single Modality Mediation Models*

Description

The single modality mediation model is expressed as below:

- Eq. 1: $M = X\alpha + \epsilon_M$
- Eq. 2: $Y = X\gamma + M\beta + \epsilon_Y$

The penalty options are listed as below:

- Elastic Net: $\lambda_{1g}|\gamma|$ + $\sum_{i=1}^{p}(\lambda_{1a}|\alpha_i| + \lambda_{1b}|\beta_i|)$ + $\lambda_{2a}\alpha^T\alpha + \lambda_{2b}\beta^T\beta$
- Pathway Lasso: $\lambda_{1g}|\gamma|$ + $\sum_{i=1}^{p}(\lambda_{1a}|\alpha_i| + \lambda_{1b}|\beta_i|)$ + $\kappa(\sum_{i=1}^{p}(|\alpha_i\beta_i| + \lambda_{2a}\alpha^T\alpha + \lambda_{2b}\beta^T\beta))$
- Network: $\lambda_{1g}|\gamma|$ + $\sum_{i=1}^{p}(\lambda_{1a}|\alpha_i| + \lambda_{1b}|\beta_i|)$ + $\kappa(\sum_{i=1}^{p}(|\alpha_i\beta_i| + \lambda_{2a}\alpha^T L_\alpha\alpha + \lambda_{2b}\beta^T L_\beta\beta))$
- Pathway Network: $\lambda_{1g}|\gamma|$ + $\sum_{i=1}^{p}(\lambda_{1a}|\alpha_i| + \lambda_{1b}|\beta_i|)$ + $\kappa(\sum_{i=1}^{p}(|\alpha_i\beta_i| + \lambda_{2a}\alpha^T\Sigma_\alpha\alpha + \lambda_{2b}\beta^T\Sigma_\beta\beta))$ where $\Sigma_\alpha = L_\alpha + \lambda^*_{2a}I_p$ and $\Sigma_\beta = L_\beta + \lambda^*_{2b}I_p$

which L_α and L_β are the laplacian matrices which we use `laplacianMatrixA` and `laplacianMatrixB` to represent in the code. Note that in the original work of the Pathway Lasso, certain restrictions are defined for the tuning parameters, including $\lambda_{1a} = \lambda_{1b}$, and $\lambda_{2a} = \lambda_{2b}$. In addition, λ_{2a} and λ_{2b} must be at least 0.5 to ensure that the penalty remains convex for optimization purpose.

Usage

```
singleModalityAdmm(
  X,
  Y,
  M1,
  rho = 1,
  lambda1a,
  lambda1b,
  lambda1g,
  lambda2a,
  lambda2b,
  penalty = "ElasticNet",
  penaltyParameterList = list(),
  SIS = FALSE,
  SISThreshold = 2,
  maxIter = 3000L,
  tol = 0.001,
  verbose = FALSE,
 verboseOptions = list(numIter = 10L, numAlpha = 1L, numBeta = 1L, numGamma = 1L)
)
```

Arguments

`X` The matrix of independent variables (exposure/treatment/group).

`Y` The vector of dependent variable (outcome response).

`M1` The single modality mediator.

`rho` The augmented Lagrangian parameter for ADMM.

`lambda1a` The L1-norm penalty for the effect between mediator and independent variables.

`lambda1b` The L1-norm penalty for the effect between mediator and dependent variable.

`lambda1g` The L1-norm penalty for the direct effect. Default is **10** to address overestimate issue.

`lambda2a` The L2-norm penalty for the effect between mediator and independent variables.

`lambda2b` The L2-norm penalty for the effect between mediator and dependent variable.

`penalty` A string to specify the penalty. Default is `ElasticNet`. Possible methods are Elastic Net (`ElasticNet`), Pathway Lasso (`PathywayLasso`), Network-constrained Penalty (`Network`), and Pathway Network (`PathwayNetwork`).

`penaltyParameterList`

- Penalty=`ElasticNet` don't need other parameters.
- Penalty=`Network` needs two parameters.
  - laplacianMatrixA and laplacianMatrixB The L2-norm penalty for Network.
- Penalty=`PathwayLasso` needs two parameters.
  - kappa The L1-norm penalty for Pathway Lasso.
- Penalty=`PathwayNetwork` needs five parameters.

  - kappa The L1-norm penalty for Pathway Network.
  - lambda2aStar and lambda2bStar The L2-norm penalty for Pathway Network.
  - laplacianMatrixA and laplacianMatrixB The L2-norm penalty for Pathway Network.

`SIS` A logical value to specify whether to perform sure independence screening (SIS).

`SISThreshold` The threshold value for the target reduced dimension for mediators. The default is "2," which reduces the dimension to 2*n/log(n).

`maxIter` The maximum iterations. Default is `3000`.

`tol` The tolerance of convergence threshold. Default is `1e-3`.

`verbose` A logical value to specify whether to print the iteration process.

`verboseOptions` A list of values:

- `numIter`: The number of iterations to print.
- `numAlpha`: The number of `alpha` to print.
- `numBeta`: The number of `beta` to print.
- `numGamma`: The number of `gamma` to print.

Value

A object, SingleModalityAdmm, with three elements.

- gamma: estimated direct effect.
- alpha: estimate effect between mediator and independent variables.
- beta : estimate effect between mediator and dependent variable.

Examples

```
## Generate Empirical Data
simuData <- modalityMediationDataGen(seed = 20231201)

## Parameter Estimation for ElasticNet penalty
modelElasticNet <- singleModalityAdmm(
  X = simuData$MediData$X, Y = simuData$MediData$Y, M1 = simuData$MediData$M1,
  rho = 1, lambda1a = 1, lambda1b = 0.1, lambda1g = 2, lambda2a = 1, lambda2b = 1,
  penalty = "ElasticNet"
)

# fitted & predict
fitted(modelElasticNet)
predict(modelElasticNet, matrix(c(0, 1), ncol=1))

## Parameter Estimation for Pathway Lasso penalty
modelPathwayLasso <- singleModalityAdmm(
  X = simuData$MediData$X, Y = simuData$MediData$Y, M1 = simuData$MediData$M1,
  rho = 1, lambda1a = 1, lambda1b = 0.1, lambda1g = 2, lambda2a = 1, lambda2b = 1,
  penalty = "PathwayLasso", penaltyParameterList = list(kappa = 1)
```

```
)

## Parameter Estimation for Network penalty
modelNetwork <- singleModalityAdmm(
  X = simuData$MediData$X, Y = simuData$MediData$Y, M1 = simuData$MediData$M1,
  rho = 1, lambda1a = 1, lambda1b = 0.1, lambda1g = 2, lambda2a = 1, lambda2b = 1,
  penalty = "Network", penaltyParameterList = list(
    laplacianMatrixA = simuData$Info$laplacianMatrixA,
    laplacianMatrixB = simuData$Info$laplacianMatrixB
  )
)

## Parameter Estimation for Pathway Network penalty
modelPathwayNetwork <- singleModalityAdmm(
  X = simuData$MediData$X, Y = simuData$MediData$Y, M1 = simuData$MediData$M1,
  rho = 1, lambda1a = 1, lambda1b = 0.1, lambda1g = 2, lambda2a = 1, lambda2b = 1,
  penalty = "Network", penaltyParameterList = list(
    kappa = 1, lambda2aStar = 1, lambda2bStar = 1,
    laplacianMatrixA = simuData$Info$laplacianMatrixA,
    laplacianMatrixB = simuData$Info$laplacianMatrixB
  )
)

## Parameter Estimation for Network penalty with a customized Laplacian matrix
set.seed(20231201)
p <- ncol(simuData$MediData$M1)
Wa <- matrix(0, nrow = p, ncol = p)
Wa[lower.tri(Wa)] <- runif(p*(p-1)/2, 0, 1)
Wa[upper.tri(Wa)] <- t(Wa)[upper.tri(Wa)]
diag(Wa) <- 1
La <- weightToLaplacian(Wa)
Wb <- matrix(0, nrow = p, ncol = p)
Wb[lower.tri(Wb)] <- runif(p*(p-1)/2, 0, 1)
Wb[upper.tri(Wb)] <- t(Wb)[upper.tri(Wb)]
diag(Wb) <- 1
Lb <- weightToLaplacian(Wb)
modelNetwork <- singleModalityAdmm(
  X = simuData$MediData$X, Y = simuData$MediData$Y, M1 = simuData$MediData$M1,
  rho = 1, lambda1a = 1, lambda1b = 0.1, lambda1g = 2, lambda2a = 1, lambda2b = 1,
  penalty = "Network", penaltyParameterList = list(
    laplacianMatrixA = La, laplacianMatrixB = Lb
  )
)

## With sure independence screening
simuData <- modalityMediationDataGen(
  n = 50, p = 1000, seed = 20231201, laplacianA = FALSE, laplacianB = FALSE
)

## Parameter Estimation for ElasticNet penalty
modelElasticNetSIS <- singleModalityAdmm(
  X = simuData$MediData$X, Y = simuData$MediData$Y, M1 = simuData$MediData$M1,
  rho = 1, lambda1a = 1, lambda1b = 0.1, lambda1g = 2, lambda2a = 1, lambda2b = 1,
  penalty = "ElasticNet", SIS = TRUE
)
fitted(modelElasticNetSIS)
predict(modelElasticNetSIS, matrix(c(0, 1), ncol=1))
```

`weightToLaplacian` *Helper function to convert Weight Matrix to Laplacian Matrix*

Description

Helper function to convert Weight Matrix to Laplacian Matrix

Usage

```
weightToLaplacian(W)
```

Arguments

`W`	The weight matrix for n nodes which should be `nxn` matrix.

Value

L `nxn` Laplacian matrix.

Examples

```
set.seed(20231201)
p <- 5
W <- matrix(0, nrow = p, ncol = p)
W[lower.tri(W)] <- runif(p*(p-1)/2, 0, 1)
W[upper.tri(W)] <- t(W)[upper.tri(W)]
diag(W) <- 1
(L <- weightToLaplacian(W))
```

Index