Package ‘MTE’

January 20, 2025

Type Package

Title Maximum Tangent Likelihood Estimation for Robust Linear Regression and Variable Selection

Version 1.2

Date 2023-04-08

Maintainer Shaobo Li <shaobo.li@ku.edu>

Description Several robust estimators for linear regression and variable selection are provided. Included are Maximum tangent likelihood estimator by Qin, et al., (2017) <arxiv:1708.05439>, least absolute deviance estimator and Huber regression. The penalized version of each of these estimator incorporates L1 penalty function, i.e., LASSO and Adaptive Lasso. They are able to produce consistent estimates for both fixed and high-dimensional settings.

URL GitHub: `https://github.com/shaobo-li/MTE`

Depends R (>= 3.1.0)

License GPL-3

RoxygenNote 7.1.1

Imports stats, quantreg, glmnet, rqPen

NeedsCompilation no

Author Shaobo Li [aut, cre],
Yichen Qin [aut]

Repository CRAN

Date/Publication 2023-04-11 09:20:05 UTC

Contents

huber.lasso 2
huber.reg 3
huberloss 4
LAD 4
LADlasso 5
MTE 6
MTElasso 7

Index **9**

`huber.lasso` *Huber-Lasso estimator*

Description

This function is L1 penalized Huber estimator for linear regression under both fixed and high-dimensional settings. Currently, the function does not support automatic selection of huber tuning parameter.

Usage

```
huber.lasso(
  X,
  y,
  beta.ini,
  lambda,
  alpha = 2,
  adaptive = TRUE,
  intercept = TRUE,
  penalty.factor = rep(1, ncol(X))
)
```

Arguments

`X`	design matrix, standardization is recommended.
`y`	response vector.
`beta.ini`	initial estimates of beta. If not specified, LADLasso estimates from `rq.lasso.fit()` in `rqPen` is used. Otherwise, robust estimators are strongly recommended.
`lambda`	regularization parameter of Lasso or adaptive Lasso (if adaptive=TRUE).
`alpha`	1/alpha is the huber tuning parameter. Larger alpha results in smaller portion of squared loss.
`adaptive`	logical input that indicates if adaptive Lasso is used. Default is TRUE.
`intercept`	logical input that indicates if intercept needs to be estimated. Default is FALSE.
`penalty.factor`	can be used to force nonzero coefficients. Default is rep(1, ncol(X)) as in glmnet.

Value

`beta`	the regression coefficient estimates.
`fitted`	predicted response.
`iter.steps`	iteration steps.

Examples

```
set.seed(2017)
n=200; d=500
X=matrix(rnorm(n*d), nrow=n, ncol=d)
beta=c(rep(2,6), rep(0, d-6))
y=X%*%beta+c(rnorm(150), rnorm(30,10,10), rnorm(20,0,100))
output.HuberLasso=huber.lasso(X,y)
beta.est=output.HuberLasso$beta
```

`huber.reg` *Huber estimation for linear regression*

Description

This function produces Huber estimates for linear regression. Initial estimates is required. Currently, the function does not support automatic selection of huber tuning parameter.

Usage

```
huber.reg(y, X, beta.ini, alpha, intercept = FALSE)
```

Arguments

`y`	the response vector
`X`	design matrix
`beta.ini`	initial value of estimates, could be from OLS.
`alpha`	1/alpha is the huber tuning parameter delta. Larger alpha results in smaller portion of squared loss.
`intercept`	logical input that indicates if intercept needs to be estimated. Default is FALSE.

Value

`beta`	the regression coefficient estimates
`fitted.value`	predicted response
`iter.steps`	iteration steps.

Examples

```
set.seed(2017)
n=200; d=4
X=matrix(rnorm(n*d), nrow=n, ncol=d)
beta=c(1, -1, 2, -2)
y=-2+X%*%beta+c(rnorm(150), rnorm(30,10,10), rnorm(20,0,100))
beta0=beta.ls=lm(y~X)$coeff
beta.huber=huber.reg(y, X, beta0, 2, intercept=TRUE)$beta
cbind(c(-2,beta), beta.ls, beta.huber)
```

`huberloss` *Huber Loss*

Description

Huber Loss

Usage

```
huberloss(r, alpha)
```

Arguments

`r`	residual, y-Xbeta
`alpha`	1/alpha is the huber tuning parameter delta. Larger alpha results in smaller portion of squared loss.

Value

it returns huber loss that will be called in Huber estimation.

`LAD` *Least Absolute Deviance Estimator for Linear Regression*

Description

Least Absolute Deviance Estimator for Linear Regression

Usage

```
LAD(X, y, intercept = FALSE)
```

Arguments

`X`	design matrix
`y`	reponse vector
`intercept`	logical input that indicates if intercept needs to be estimated. Default is FALSE.

Value

coefficient estimates

Examples

```
set.seed(1989)
n=200; d=4
X=matrix(rnorm(n*d), nrow=n, ncol=d)
beta=c(1, -1, 2, -2)
y=-2+X%*%beta+c(rnorm(150), rnorm(30,10,10), rnorm(20,0,100))
beta.ls=lm(y~X)$coeff
beta.LAD=LAD(X,y,intercept=TRUE)
cbind(c(-2,beta), beta.ls, beta.LAD)
```

`LADlasso` *LAD-Lasso for Linear Regression*

Description

LAD-Lasso for Linear Regression

Usage

```
LADlasso(
  X,
  y,
  beta.ini,
  lambda = NULL,
  adaptive = TRUE,
  intercept = FALSE,
  penalty.factor = rep(1, ncol(X))
)
```

Arguments

`X`	design matrix, standardization is recommended.
`y`	reponse vector
`beta.ini`	initial estimates of beta. Using unpenalized LAD is recommended under high-dimensional setting.
`lambda`	regularization parameter of Lasso or adaptive Lasso (if adaptive=TRUE).
`adaptive`	logical input that indicates if adaptive Lasso is used. Default is TRUE.
`intercept`	logical input that indicates if intercept needs to be estimated. Default is FALSE.
`penalty.factor`	can be used to force nonzero coefficients. Default is rep(1, ncol(X)) as in glmnet.

Value

`beta`	the regression coefficient estimates.
`fitted`	predicted response.
`iter.steps`	iteration steps.

Examples

```
set.seed(2017)
n=200; d=50
X=matrix(rnorm(n*d), nrow=n, ncol=d)
beta=c(rep(2,6), rep(0, 44))
y=X%*%beta+c(rnorm(150), rnorm(30,10,10), rnorm(20,0,100))
output.LADLasso=LADlasso(X, y, beta.ini=LAD(X, y))
beta.est=output.LADLasso$beta
```

MTE — *Maximum Tangent-likelihood Estimation*

Description

It estimates linear regression coefficient using MTE. The function produces robust estimates of linear regression. Outliers and contamination would be downweighted. It is robust to Gaussian assumption of the error term. Initial estimates need to be provided.

Usage

```
MTE(y, X, beta.ini, t, p, intercept = FALSE)
```

Arguments

`y`	the response vector
`X`	design matrix
`beta.ini`	initial value of estimates, could be from OLS.
`t`	the tangent point. You may specify a sequence of values, so that the function automatically select the optimal one.
`p`	Taylor expansion order, up to 3.
`intercept`	logical input that indicates if intercept needs to be estimated. Default is FALSE.

Value

Returns estimates from MTE method.

`beta`	the regression coefficient estimates
`fitted.value`	predicted response
`t`	the optimal tangent point through data-driven method

Examples

```
set.seed(2017)
n=200; d=4
X=matrix(rnorm(n*d), nrow=n, ncol=d)
beta=c(1, -1, 2, -2)
y=-2+X%*%beta+c(rnorm(150), rnorm(30,10,10), rnorm(20,0,100))
beta0=beta.ls=lm(y~X)$coeff
beta.MTE=MTE(y,X,beta0,0.1,2, intercept=TRUE)$beta
cbind(c(-2,beta), beta.ls, beta.MTE)
```

MTElasso *MTE-Lasso estimator*

Description

MTELasso is the penalized MTE for robust estimation and variable selection for linear regression. It can deal with both fixed and high-dimensional settings.

Usage

```
MTElasso(
  X,
  y,
  beta.ini,
  p = 2,
  lambda = NULL,
  adaptive = TRUE,
  t = 0.01,
  intercept = TRUE,
  penalty.factor = rep(1, ncol(X)),
  ...
)
```

Arguments

X	design matrix, standardization is recommended.
y	response vector.
beta.ini	initial estimates of beta. If not specified, LADLasso estimates from `rq.lasso.fit()` in `rqPen` is used. Otherwise, robust estimators are strongly recommended.
p	Taylor expansion order.
lambda	regularization parameter for LASSO, but not necessary if "adaptive=TRUE".
adaptive	logic argument to indicate if Adaptive-Lasso is used. Default is TRUE.
t	the tuning parameter that controls for the tradeoff between robustness and efficiency. Default is t=0.01.

`intercept`	logical input that indicates if intercept needs to be estimated. Default is FALSE.
`penalty.factor`	can be used to force nonzero coefficients. Default is rep(1, ncol(X)) as in glmnet.
`...`	other arguments that are used in `glmnet`.

Value

It returns a sparse vector of estimates of linear regression. It has two types of penalty, LASSO and AdaLasso. Coordinate descent algorithm is used for iteratively updating coefficients.

`beta`	sparse regression coefficient
`fitted`	predicted response

Examples

```
set.seed(2017)
n=200; d=500
X=matrix(rnorm(n*d), nrow=n, ncol=d)
beta=c(rep(2,6), rep(0, d-6))
y=X%*%beta+c(rnorm(150), rnorm(30,10,10), rnorm(20,0,100))
output.MTELasso=MTElasso(X, y, p=2, t=0.01)
beta.est=output.MTELasso$beta
```

Index

huber.lasso, 2
huber.reg, 3
huberloss, 4

LAD, 4
LADlasso, 5

MTE, 6
MTElasso, 7