Package 'MTE'

August 29, 2017

Type Package

Title Maximum Tangent Likelihood and Other Robust Estimation for High-Dimensional Regression

Version 1.0.0

Author Shaobo Li and Yichen Qin

Maintainer Shaobo Li <lis6@mail.uc.edu>

Description

Provides several robust estimators for linear regression and variable selection. They are Maximum tangent likelihood estimator (Qin, et al. (2017) <arXiv:1708.05439>), least absolute deviance estimator, and Huber loss. 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.

Depends R (>= 3.1.0)

License GPL-3

LazyData TRUE

RoxygenNote 6.0.1

Imports stats, quantreg, parcor

NeedsCompilation no

Repository CRAN

Date/Publication 2017-08-29 12:18:35 UTC

R topics documented:

`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(y, X, beta.ini, lambda, alpha = 2, adaptive = T,
  intercept = FALSE)
```

Arguments

`y`	response vector.
`X`	design matrix, standardization is recommended.
`beta.ini`	initial estimates of beta. Using unpenalized Huber or LAD is recommended under high-dimensional setting.
`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.

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.HuberLasso=huber.lasso(y,X, beta.ini=LAD(y, X), lambda=0.2, adaptive=TRUE)
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(y, X, intercept = FALSE)
```

Arguments

y	reponse vector
X	design matrix
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.MTE=LAD(y,X,intercept=TRUE)
cbind(c(-2,beta), beta.ls, beta.MTE)
```

LADlasso *LAD-Lasso for Linear Regression*

Description

LAD-Lasso for Linear Regression

Usage

```
LADlasso(y, X, beta.ini, lambda, adaptive = TRUE, intercept = FALSE)
```

Arguments

y	reponse vector
X	design matrix, standardization is recommended.
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.

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(y,X, beta.ini=LAD(y, X), lambda=0.2, adaptive=TRUE)
beta.est=output.LADLasso$beta
```

lnt *Tangent-likelihood function.*

Description

The function calculates tangent-likelihood given the value of density.

Usage

```
lnt(f, t, p)
```

Arguments

f	values of density function.
t	tangent point. It is positive and close to 0. If 0, the tangent-likelihood is equivalent to log-likelihood.
p	Taylor expansion order, the value can be set up to 3.

Examples

```
set.seed(2017)
x=c(rnorm(80), rnorm(20, 10, 10))
f=dnorm(x)
y=lnt(f, 0.1, 2)
```

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

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(y, X, beta.ini, p, lambda, adaptive = T, t, method = "MTE",
  intercept = FALSE, ...)
```

Arguments

y	response vector.
X	design matrix, standardization is recommended.
beta.ini	initial estimates of beta. Using unpenalized MTE or LAD is recommended under high-dimensional setting.
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 tangent point. You may specify a sequence of values, so that the function automatically select the optimal one.
method	it can be ("MTE", "MLE"). The default is MTE.
intercept	logical input that indicates if intercept needs to be estimated. Default is FALSE.
...	other arguments that are used in function "adalasso()" that is called form parcor package.

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 interatively updating coefficients.

`beta`	sparse regression coefficient
`fitted`	predicted response
`t`	optimal tangent point

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))
beta0=MTE(y, X, rep(0,50), 0.1, 2)$beta
output.MTELasso=MTElasso(y,X, p=2, beta.ini=beta0, t=seq(0, 0.1, 0.01), method="MTE")
beta.est=output.MTELasso$beta
```

Index