Package ‘plvs’

March 15, 2019

Type Package

Title Ultra-high dimensional variable selection piecewise linear loss function.

Version 1.0

Date 2017-01-03

Author Xu Liu, Hongmei Jiang and Xingjie Shi

Maintainer Xu Liu <liu.xu@mail.shufe.edu.cn>

Description Select important variables for piecewise linear loss under ultra-high dimensional data and simultaneously estimate the selected parameters.

License GPL (>= 2)

LazyLoad yes

NeedsCompilation yes

R topics documented:

`plvs-package` *Ultra-high dimensional variable selection for piecewise linear loss function*

Description

This package selects the important variables for piecewise linear loss function under ultra-high dimensional data, and simultaneously estimate the corresponding parameters of the selected variables, in which the Coordinate Descend and Minorization and Maximization (CDMM) algorithm is used.

Details

Package:	plvs
Type:	Package
Version:	1.0
Date:	2012-07-03
License:	GPL (>= 2)
LazyLoad:	Yes

Unpenalized Composite Quantile Regression. cqr(x, y, q)

Penalized Composite Quantile Regression. pcqr(x, y, q)

Penalized Single Quantile Regression. pqr(x, y, q)

Author(s)

Xu Liu, Hongmei Jiang and Xingjie Shi

Maintainer: Xu Liu <liu.xu@mail.shufe.edu.cn>

References

Ultra-high dimensional variable selection piecewise linear loss function

See Also

cqr, pqr, pcqr, plot.cv, setuplambda

---

cqr — *Fit usuall single quantile regression which is unpenalized.*

Description

This function coefficients for unpenalized composite quantile regression model, in which the Coordinate Descend and Minorization and Maximization (CDMM) algorithm is used.

Usage

```
cqr <- function(x, y, q = 0.5, maxstep = 1e2,
                                    eps0 = 1e-8, eps = 1e-4)
```

Arguments

`x`	A numeric design matrix for the model
`y`	A numeric vector of responses
`q`	The q^{th} quantile, a scalar or vector with the value in `(0, 1)`. Default is `q = 0.5`.
`maxstep`	Maximum number of iterations. Default is 100.
`eps0`	The perturbation when MM algorithm is used. Default is `eps0=1e-8`.
`eps`	Convergence threshhold. The algorithm iterates until the relative change in any coefficient is less than `eps`. Default is `.0001`.

Details

This function the estimator of β for unpenalized composite quantile regression model.

Value

`hatbeta`	Estimator of β
`beta0`	Intercept term which is numeric or vector dependent on input quatile q.
`...`	other options for Composite Quantile

Author(s)

Xu Liu, Hongmei Jiang and Xingjie Shi

References

Ultra-high dimensional variable selection piecewise linear loss function

Examples

```
# normal
n = 200;p=10
beta <- c(1, 2, 3, rep(0, p-3))
q <- 0.5
x <- matrix(rnorm(n*p), nrow = n)
y <- x%*%beta + sqrt(3)*rnorm(n)
fit <- cqr(x,y,q)
fit$hatbeta
fit$beta0

# T-distribution
n = 200;p=10
beta <- c(1, 2, 3, rep(0, p-3))
q <- c(1:19)/20
x <- matrix(rnorm(n*p), nrow = n)
y <- x%*%beta + rt(n, 3)
fit <- cqr(x,y,q)
fit$hatbeta
fit$beta0

#logistic-distribution
n = 200;p=10
beta <- c(1, 2, 3, rep(0, p-3))
q <- c(1:19)/20
x <- matrix(rnorm(n*p), nrow = n)
y <- x%*%beta + log(tan(runif(n)))
fit <- cqr(x,y,q)
fit$hatbeta
fit$beta0

#T-normal-mixed
n = 200;p=10
beta <- c(1, 2, 3, rep(0, p-3))
q <- c(1:19)/20
x <- matrix(rnorm(n*p), nrow = n)
y <- x%*%beta + sqrt(2)*rnorm(n)/2 + rt(n, 4)/2
```

```
fit <- cqr(x,y,q)
fit$hatbeta
fit$beta0
```

---

pcqr *Fit the entire solution path for composite quantile regression based on three penalties* LASSO, MCP *and* SCAD

Description

This function selects the important variables for composite quantile regression model under ultra-high dimensional data, and simultaneously estimate the corresponding parameters of the selected variables, in which the Coordinate Descend and Minorization and Maximization (CDMM) algorithm is used.

Usage

```
pcqr(x, y, q , penalty = "MCP", lambda, nlambda, eps = 0.001,
                 maxstep = 1000, gamma = 2, alpha = 1, dfmax = NULL,
                 user_lam = 1, eps0 = 1e-8, isbic = 1, nfold = 10)
```

Arguments

x	A numeric design matrix for the model
y	A numeric vector of responses
q	The q^{th} quantile, a scalar or vector with the value in (0, 1). Default is q = c(1:19)/20.
penalty	LASSO, SCAD and MCP. Default is MCP
lambda	A user-specified sequence of lambda values. By default, a sequence of values of length nlambda is computed, equally spaced on the log scale.
nlambda	The length of lambda. Default is 100.
eps	Convergence threshhold. The algorithm iterates until the relative change in any coefficient is less than eps. Default is .001.
maxstep	Maximum number of iterations. Default is 1000.
gamma	The tuning parameter of the MCP/SCAD penalty (see details).
alpha	Tuning parameter for the Mnet estimator which controls the relative contributions from the LASSO, MCP/SCAD penalty and the ridge, or L2 penalty. alpha=1 is equivalent to LASSO, MCP/SCAD penalty, while alpha=0 would be equivalent to ridge regression. However, alpha=0 is not supported; alpha may be arbitrarily small, but not exactly 0.
dfmax	Upper bound for the number of nonzero coefficients. Default is no upper bound. However, for large data sets, computational burden may be heavy for models with a large number of nonzero coefficients.
user_lam	If given lambda? Defaul is FALSE.
eps0	The perturbation when MM algorithm is used. Default is eps0=1e-8.
isbic	Is BIC criteria used to select the tuning parameter λ? BIC isbic=1; CV isbic=2; AIC isbic=3.
nfold	How many fold to be used when cross-validation method is used to select the tuning parameter. Default is 10.

Details

This function gives a series of solution path for composite quantile regression regression model, the corresponding degrees of freedom (df), and the log-likelihood value. Those values can be analysed in `plot.cv` and others. A tuning parameter is also selected by BIC (or CV and AIC) to gives the estimator of β.

Value

`hatbeta`	Estimator of β
`beta0`	Intercept term which is numeric or vector dependent on input quatile q.
`betapath`	Solution path of β
`df`	Degrees of freedom
`bic`	`bic`(λ) which is used to select the tuning parameter λ dependent on selecting criteria (BIC, CV or AIC)
`loglikelih`	Log-likelihood for each λ
`ind0`	Selected index of tuning parameter λ
`...`	Other options for pcqr

Author(s)

Xu Liu, Hongmei Jiang and Xingjie Shi

References

Ultra-high dimensional variable selection piecewise linear loss function

See Also

pqr, plot.cv

Examples

```
# normal
n = 200;p=20
beta <- c(1, 2, 3, rep(0, p-3))
q <- c(1:19)/20
x <- matrix(rnorm(n*p), nrow = n)
y <- x%*%beta + sqrt(3)*rnorm(n)
fit <- pcqr(x,y,q)
fit$ind0
fit$df[fit$ind0]
fit$hatbeta[abs(fit$hatbeta)>0]

# T-distribution
n = 200;p=20
beta <- c(1, 2, 3, rep(0, p-3))
q <- c(1:19)/20
x <- matrix(rnorm(n*p), nrow = n)
y <- x%*%beta + rt(n, 3)
fit <- pcqr(x,y,q)
fit$ind0
fit$df[fit$ind0]
fit$hatbeta[abs(fit$hatbeta)>0]
```

```
#logistic-distribution
n = 200;p=20
beta <- c(1, 2, 3, rep(0, p-3))
q <- c(1:19)/20
x <- matrix(rnorm(n*p), nrow = n)
y <- x%*%beta + log(tan(runif(n)))
fit <- pcqr(x,y,q)
fit$ind0
fit$df[fit$ind0]
fit$hatbeta[abs(fit$hatbeta)>0]

#T-normal-mixed
n = 200;p=20
beta <- c(1, 2, 3, rep(0, p-3))
q <- c(1:19)/20
x <- matrix(rnorm(n*p), nrow = n)
y <- x%*%beta + sqrt(2)*rnorm(n)/2 + rt(n, 4)/2
fit <- pcqr(x,y,q)
fit$ind0
fit$df[fit$ind0]
fit$hatbeta[abs(fit$hatbeta)>0]
```

`plot.cv` *Fit the entire solution path for single quantile regression based on three penalties* `LASSO, MCP` *and* `SCAD`

Description

This function selects the important variables for single quantile regression model under ultra-high dimensional data, and simultaneously estimate the corresponding parameters of the selected variables, in which the Coordinate Descend and Minorization and Maximization (CDMM) algorithm is used.

Usage

```
plot.cv <- function(x, y, q = 0.5, fit = fit, isbic = 1)
```

Arguments

`x`	A numeric design matrix for the model
`y`	A numeric vector of responses
`q`	The q^{th} quantile, a scalar or vector with the value in `(0, 1)`. Default is `q = 0.5`.
`fit`	The fit results by `CompositeQuantile` or `SingleQuantile` including `lambda, ind0` and `bic`.
`isbic`	Is BIC criteria used to select the tuning parameter λ. BIC `isbic=1`; CV `isbic=2`; AIC `isbic=3`.

Details

This function plots the `bic, aic or cv` corresponding to β to check whether the selected tuning parameter is correct or not.

Value

`...` Plot the figure of `bic`, see the details of `bic` refering to `Composite Quantile` or `Single Quantile`

Author(s)

Xu Liu, Hongmei Jiang and Xingjie Shi

References

Ultra-high dimensional variable selection piecewise linear loss function

See Also

CompositeQuantile, SingleQuantile

Examples

```
n=200;p=20
beta <- c(1, 2, 3, rep(0, p-3))
q <- 0.5
x <- matrix(rnorm(n*p), nrow = n)
y <- x%*%beta + rnorm(rnorm(n))
plot.cv(x,y, q)
```

`pqr` *Fit the entire solution path for single quantile regression based on three penalties* `LASSO, MCP` *and* `SCAD`

Description

This function selects the important variables for single quantile regression model under ultra-high dimensional data, and simultaneously estimate the corresponding parameters of the selected variables, in which the Coordinate Descend and Minorization and Maximization (CDMM) algorithm is used.

Usage

```
pqr(x, y, q , penalty = "MCP", lambda, eps = 0.001,
              maxstep = 1000, gamma = 2, alpha = 1, dfmax = NULL,
              user_lam = 1, eps0 = 1e-8, isbic = 1, nfold = 10)
```

Arguments

`x`	A numeric design matrix for the model
`y`	A numeric vector of responses
`q`	The q^{th} quantile, a scalar with the value in `(0, 1)`. Default is `q = 0.5`.
`penalty`	LASSO, SCAD and MCP. Default is MCP
`lambda`	A user-specified sequence of lambda values. By default, a sequence of values of length `nlambda` is computed, equally spaced on the log scale.

eps	Convergence threshhold. The algorithm iterates until the relative change in any coefficient is less than eps. Default is .001.
maxstep	Maximum number of iterations. Default is 1000.
gamma	The tuning parameter of the MCP/SCAD penalty (see details).
alpha	Tuning parameter for the Mnet estimator which controls the relative contributions from the LASSO, MCP/SCAD penalty and the ridge, or L2 penalty. alpha=1 is equivalent to LASSO, , MCP/SCAD penalty, while alpha=0 would be equivalent to ridge regression. However, alpha=0 is not supported; alpha may be arbitrarily small, but not exactly 0.
dfmax	Upper bound for the number of nonzero coefficients. Default is no upper bound. However, for large data sets, computational burden may be heavy for models with a large number of nonzero coefficients.
user_lam	If given lambda? Defaul is FALSE.
eps0	The perturbation when MM algorithm is used. Default is eps0=1e-8.
isbic	Is BIC criteria used to select the tuning parameter λ. BIC isbic=1; CV isbic=2; AIC isbic=3.
nfold	How many fold to be used when cross-validation method is used to select the tuning parameter. Default is 10.

Details

This function gives a series of solution path for single quantile regression model, the corresponding degrees of freedom (df), and the log-likelihood value. Those values can be analysed in plot.cv and others. A tuning parameter also selected by BIC (or CV and AIC) to gives the estimator of β.

Value

hatbeta	Estimator of β
beta0	Intercept term which is numeric or vector dependent on input quatile q.
betapath	Solution path of β
df	Degrees of freedom
bic	bic(λ) which is used to select the tuning parameter λ dependent on selecting criteria (BIC, CV or AIC)
loglikelih	Log-likelihood for each λ
ind0	Selected index of tuning parameter λ
...	other options for Composite Quantile

Author(s)

Xu Liu, Hongmei Jiang and Xingjie Shi

References

Ultra-high dimensional variable selection piecewise linear loss function

See Also

pcqr, plot.cv

Examples

```
# normal
n = 200;p=20
beta <- c(1, 2, 3, rep(0, p-3))
q <- 0.5
x <- matrix(rnorm(n*p), nrow = n)
y <- x%*%beta + sqrt(3)*rnorm(n)
fit <- pqr(x,y,q)
fit$ind0
fit$df[fit$ind0]
fit$hatbeta[abs(fit$hatbeta)>0]

# T-distribution
n = 200;p=50
beta <- c(1, 2, 3, rep(0, p-3))
q <- 0.5
x <- matrix(rnorm(n*p), nrow = n)
y <- x%*%beta + rt(n, 3)
fit <- pqr(x,y,q)
fit$ind0
fit$df[fit$ind0]
fit$hatbeta[abs(fit$hatbeta)>0]

#logistic-distribution
n = 200;p=20
beta <- c(1, 2, 3, rep(0, p-3))
q <- 0.5
x <- matrix(rnorm(n*p), nrow = n)
y <- x%*%beta + log(tan(runif(n)))
fit <- pqr(x,y,q)
fit$ind0
fit$df[fit$ind0]
fit$hatbeta[abs(fit$hatbeta)>0]

#T-normal-mixed
n = 200;p=20
beta <- c(1, 2, 3, rep(0, p-3))
q <- 0.5
x <- matrix(rnorm(n*p), nrow = n)
y <- x%*%beta + sqrt(2)*rnorm(n)/2 + rt(n, 4)/2
fit <- pqr(x,y,q)
fit$ind0
fit$df[fit$ind0]
fit$hatbeta[abs(fit$hatbeta)>0]
```

---

setuplambda *Setup of the tuning parameter λ*

Description

This function sets the tuning parameter λ, which is used in `CompositeQuantile, SingleQuantile`. One can also set up λ by self.

Usage

```
setuplambda <- function(x, y, q, nlam, lam_max = 1, lam_min = 1e-2,
                          alpha = 1, eps0 = 1e-8)
```

Arguments

`x`	A numeric design matrix for the model
`y`	A numeric vector of responses
`q`	The q^{th} quantile, a scalar or vector with the value in `(0, 1)`.
`nlam`	The number of tuning parameter λ to be setuped.
`lam_max`	A multiplier that times maximum λ which is selected by correlation.
`alpha`	Tuning parameter for the Mnet estimator which controls the relative contributions from the LASSO, MCP/SCAD penalty and the ridge, or L2 penalty. `alpha=1` is equivalent to LASSO,, MCP/SCAD penalty, while `alpha=0` would be equivalent to ridge regression. However, `alpha=0` is not supported; `alpha` may be arbitrarily small, but not exactly 0.
`eps0`	the perturbation when MM algorithm is used. Default is `eps0=1e-8`.

Details

This function gives a sery of the tuning parameter λ, which is used in penalties.

Value

`lambda`	Setup of the tuning parameter λ.

Author(s)

Xu Liu, Hongmei Jiang and Xingjie Shi

References

Ultra-high dimensional variable selection piecewise linear loss function

See Also

cqr, pqr, pcqr

Examples

```
# normal
n = 200;p=20
beta <- c(1, 2, 3, rep(0, p-3))
q <- 0.5
x <- matrix(rnorm(n*p), nrow = n)
y <- x%*%beta + sqrt(3)*rnorm(n)
lambda <- setuplambda(x,y,q)

# T-distribution
n = 200;p=20
beta <- c(1, 2, 3, rep(0, p-3))
q <- 0.5
x <- matrix(rnorm(n*p), nrow = n)
```

```
y <- x%*%beta + rt(n, 3)
lambda <- setuplambda(x,y,q)

#logistic-distribution
n = 200;p=20
beta <- c(1, 2, 3, rep(0, p-3))
q <- 0.5
x <- matrix(rnorm(n*p), nrow = n)
y <- x%*%beta + log(tan(runif(n)))
lambda <- setuplambda(x,y,q)

#T-normal-mixed
n = 200;p=20
beta <- c(1, 2, 3, rep(0, p-3))
q <- 0.5
x <- matrix(rnorm(n*p), nrow = n)
y <- x%*%beta + sqrt(2)*rnorm(n)/2 + rt(n, 4)/2
lambda <- setuplambda(x,y,q)
```

Index