Package: fastkqr (via r-universe)

October 12, 2024

Type Package

Title A Fast Algorithm for Kernel Quantile Regression

Version 1.0.0

Date 2024-05-07

Maintainer Qian Tang <qian-tang@uiowa.edu>

Description An efficient algorithm to fit and tune kernel quantile regression models based on the majorization-minimization (MM) method. It can also fit multiple quantile curves simultaneously without crossing.

Depends R (>= 3.5.0), methods

Imports graphics, grDevices, stats, utils, dotCall64, rlang, MASS, Matrix

License GPL-2

NeedsCompilation yes

RoxygenNote 7.2.3

Encoding UTF-8

Suggests knitr, rmarkdown

VignetteBuilder knitr

Author Qian Tang [aut, cre], Yuwen Gu [aut], Boxiang Wang [aut]

Date/Publication 2024-05-13 11:13:15 UTC

Repository https://qiantang0326.r-universe.dev

RemoteUrl https://github.com/cran/fastkqr

RemoteRef HEAD

RemoteSha 80a6e89af851212806aee17505b22201e67c0249

Contents

`coef.kqr` *Extract model coefficients from a 'kqr' object.*

Description

Computes the coefficients at the requested value(s) for 'lambda' from a [kqr()] object.

Usage

```
## S3 method for class 'kqr'
coef(object, s = NULL, ...)
```

Arguments

`object`	Fitted [kqr()] object.
`s`	Value(s) of the penalty parameter 'lambda' at which coefficients are required. Default is the entire sequence.
`...`	Not used.

Details

's' is the new vector of 'lambda' values at which predictions are requested. If 's' is not in the lambda sequence used for fitting the model, the 'coef' function will use linear interpolation to make predictions. The new values are interpolated using a fraction of coefficients from both left and right 'lambda' indices.

Value

The coefficients at the requested values for 'lambda'.

See Also

[kqr()] and [predict.kqr()].

Examples

```
library(MASS)
data(GAGurine)
x <- as.matrix(GAGurine$Age)
y <- GAGurine$GAG
lambda <- 10^(seq(1, -4, length.out=10))
fit <- kqr(x, y, lambda=lambda, tau=0.1)
coef(fit)
```

`coef.nckqr` *Extract model coefficients from a 'nckqr' object.*

Description

Computes the coefficients at the requested value(s) for 'lambda1' for a given 'lambda2' from a [nckqr()] object.

Usage

```
## S3 method for class 'nckqr'
coef(object, s1 = NULL, s2, ...)
```

Arguments

`object`	A fitted `nckqr` object.
`s1`	Value(s) of the penalty parameter 'lambda1' at which coefficients are required. Default is the entire sequence used to create the model.
`s2`	Value of the penalty parameter 'lambda2' at which coefficients are required.
`...`	Not used.

Details

's1' is the new vector of 'lambda1' values at which predictions are requested. If 's1' is not in the lambda sequence used for fitting the model, the 'coef' function will use linear interpolation to make predictions. The new values are interpolated using a fraction of coefficients from both left and right 'lambda' indices.

Value

The coefficients for the non-crossing kernel quantile regression model.

See Also

[nckqr()] and [predict.nckqr()].

Examples

```
library(MASS)
data(GAGurine)
x <- as.matrix(GAGurine$Age)
y <- GAGurine$GAG
l2 <- 1e-4
ttau <- c(0.1, 0.3, 0.5, 0.7, 0.9)
l1_list <- 10^seq(-8, 2, length.out=10)
fit <- nckqr(x,y, lambda1=l1_list, lambda2=l2, tau=ttau)
coef(fit, s1=l1_list[1:3], s2=1e-4)
```

`cv.kqr` *cross-validation for selecting the tuning parameter of kernel quantile regression*

Description

Performs k-fold cross-validation for [kqr()]. This function is largely similar [glmnet::cv.glmnet()].

Usage

```
cv.kqr(x, y, tau, lambda = NULL, sigma = NULL, nfolds = 5L, foldid, ...)
```

Arguments

`x`	A numerical input matrix. The dimension is n rows and p columns.
`y`	Response variable.
`tau`	A user-supplied `tau` value for a quantile level.
`lambda`	A user-supplied `lambda` sequence.
`sigma`	Kernel bandwidth.
`nfolds`	The number of folds in cross-validation. Default is 5.
`foldid`	An optional vector which indexed the observations into each cross-validation fold. If supplied, `nfolds` is overridden.
`...`	Additional arguments passed into `kqr`

Details

The function computes the average cross-validation error and reports the standard error.

Value

An object of class [cv.kqr()] is returned, which is a list with the components describing the cross-validation error.

`lambda`	The `lambda` candidate values.
`cvm`	Mean cross-validation error.
`cvsd`	Estimates of standard error of cross-validation error.
`cvup`	The upper curve: `cvm` + `cvsd`.
`cvlo`	The lower curve: `cvm` - `cvsd`.
`lambda.min`	The `lambda` incurring the minimum cross-validation error.
`lambda.1se`	The largest `lambda` whose error is within one standard error of the minimum.
`cv.min`	The cross-validation error at `lambda.min`.
`cv.1se`	The cross-validation error at `lambda.1se`.

Examples

```
library(MASS)
data(GAGurine)
x <- as.matrix(GAGurine$Age)
y <- GAGurine$GAG
lambda <- 10^(seq(1, -4, length.out=10))
cv.fit <- cv.kqr(x, y, lambda=lambda, tau=0.1)
```

---

cv.nckqr *cross-validation for selecting the tuning parameter 'lambda2' of non-crossing kernel quantile regression*

Description

Performs k-fold cross-validation for [nckqr()]. This function is largely similar [glmnet::cv.glmnet()].

Usage

```
cv.nckqr(
  x,
  y,
  tau,
  lambda1 = NULL,
  lambda2 = NULL,
  sigma = NULL,
  nfolds = 5L,
  foldid,
  ...
)
```

Arguments

x	A numerical input matrix. The dimension is n rows and p columns.
y	Response variable.
tau	A user-supplied tau sequence.
lambda1	A user-supplied lambda1 value.
lambda2	A user-supplied lambda2 sequence.
sigma	Kernel bandwidth.
nfolds	The number of folds in cross-validation. Default is 5.
foldid	An optional vector which indexed the observations into each cross-validation fold. If supplied, nfolds is overridden.
...	Additional arguments passed into nckqr

Details

The function computes the average cross-validation error and reports the standard error.

Value

An object of class [cv.nckqr()] is returned, which is a list with the components describing the cross-validation error.

lambda2	The lambda2 candidate values.
cvm	Mean cross-validation error.
cvsd	Estimates of standard error of cross-validation error.
cvup	The upper curve: cvm + cvsd.
cvlo	The lower curve: cvm - cvsd.
lambda.min	The lambda2 incurring the minimum cross-validation error.
lambda.1se	The largest lambda whose error is within one standard error of the minimum.
cv.min	The cross-validation error at lambda.min.
cv.1se	The cross-validation error at lambda.1se.

Examples

```
library(MASS)
data(GAGurine)
x <- as.matrix(GAGurine$Age)
y <- GAGurine$GAG
ttau <- c(0.1, 0.3, 0.5)
l2_list <- 10^(seq(1, -4, length.out=10))
cvres <- cv.nckqr(x, y, ttau, lambda1 = 10, lambda2 = l2_list)
```

kqr *Solve the kernel quantile regression. The solution path is computed at a grid of values of tuning parameter* lambda.

Description

Solve the kernel quantile regression. The solution path is computed at a grid of values of tuning parameter lambda.

Usage

```
kqr(
  x,
  y,
  lambda,
  tau,
  delta = 0.125,
  eps = 1e-05,
  maxit = 1e+06,
  gam = 1e-07,
  sigma = NULL,
  is_exact = FALSE
)
```

Arguments

x	A numerical input matrix. The dimension is n rows and p columns.
y	Response variable. The length is n.
lambda	A user-supplied lambda sequence.
tau	A user-supplied tau value for a quantile level.
delta	The smoothing index for method='huber'. Default is 0.125.
eps	Stopping criterion.
maxit	Maximum number of iterates.
gam	A small number for numerical stability.
sigma	Kernel bandwidth.
is_exact	Exact or approximated solutions. Default is FALSE.

Details

The function implements an accelerated proximal gradient descent to solve kernel quantile regression.

Value

An object with S3 class kqr

alpha	An $n + 1$ by L matrix of coefficients, where n is the number of observations and L is the number of tuning parameters. The first row of alpha contains the intercepts.
lambda	The lambda sequence that was actually used.
delta	The smoothing index.
npass	The total number of iterates used to train the classifier.
jerr	Warnings and errors; 0 if none.
info	A list includes some settings used to fit this object: eps, maxit

.

Examples

```
library(MASS)
data(GAGurine)
x <- as.matrix(GAGurine$Age)
y <- GAGurine$GAG
lambda <- 10^(seq(1, -4, length.out=30))
fit <- kqr(x, y, lambda=lambda, tau=0.1, is_exact=TRUE)
```

---

nckqr *Solve the non-crossing kernel quantile regression*

Description

Trains the kernel quantile regression

Usage

```
nckqr(
  x,
  y,
  lambda1,
  lambda2,
  tau,
  delta = 0.125,
  eps = 1e-08,
  maxit = 5e+06,
  gam = 1e-07,
  sigma = NULL,
  kernel = "rbfdot",
  is_exact = FALSE
)
```

Arguments

x	A numerical input matrix. The dimension is $n + 1$ by $ntau$ by $L1$ by $L2$.
y	Response variable. The length is n.
lambda1	A user-supplied lambda1 sequence. The length is $L1$.
lambda2	A user-supplied lambda2 sequence. The length is $L2$.
tau	A user-supplied tau sequence for quantile levels. The length is $ntau$.
delta	The smoothing index for method='huber'. Default is 0.125.
eps	Stopping criterion.
maxit	Maximum number of iterates.
gam	A small number for numerical stability.
sigma	Kernel bandwidth.
kernel	Name of kernel function. Default is "Gaussian".
is_exact	Exact or approximated solutions.

Details

The function implements the majorization-minimization method to solve non-crossing kernel quantile regression.

Value

An object with S3 class nckqr

alpha	An $n+1$ by L matrix of coefficients, where n represents the number of observations, $ntau$ represents the number of quantile levels, and L denotes the number of tuning parameters.
tau	The tau sequence that was actually used.
lambda1	The lambda1 sequence that was actually used.
lambda2	The lambda2 sequence that was actually used.
delta	The smoothing index.
npass	The total number of iterates used to train the classifier.
jerr	Warnings and errors; 0 if none.
info	A list includes some settings used to fit this object: eps, maxit

.

Examples

```
library(MASS)
lambda2 <- 1e-4
tau <- c(0.1, 0.3, 0.5, 0.7, 0.9)
lambda1 <- 10^seq(-8, 2, length.out=10)
data(GAGurine)
x <- as.matrix(GAGurine$Age)
y <- GAGurine$GAG
fit <- nckqr(x ,y, lambda1 = lambda1 , lambda2 = lambda2, tau = tau)
```

predict.kqr *Predict the fitted values for a* kqr *object.*

Description

Predict the fitted values for a kqr object.

Usage

```
## S3 method for class 'kqr'
predict(object, x, newx = NULL, s = NULL, ...)
```

Arguments

object	A fitted kqr object.
x	The predictor matrix, i.e., the x matrix used when fitting the kqr object.
newx	A matrix of new values for x at which predictions are to be made. Note that newx must be of a matrix form, predict function does not accept a vector or other formats of newx.
s	Value(s) of the penalty parameter 'lambda' at which predictions are required. Default is the entire sequence used to create the model.
...	Not used.

Details

The result is $\beta_0 + K_i'\alpha$ where β_0 and α are from the kqr object and K_i is the ith row of the kernel matrix.

Value

Returns the fitted values.

Examples

```
library(MASS)
data(GAGurine)
x <- as.matrix(GAGurine$Age)
y <- GAGurine$GAG
lambda <- 10^(seq(1, -4, length.out=30))
fit <- kqr(x, y, lambda=lambda, tau=0.1, is_exact=TRUE)
predict(fit, x, tail(x))
```

predict.nckqr *Predict the fitted values for a* nckqr *object.*

Description

Predict the fitted values for a nckqr object.

Usage

```
## S3 method for class 'nckqr'
predict(object, x, newx = NULL, s2, s1 = NULL, ...)
```

Arguments

object	A fitted nckqr object.
x	The predictor matrix, i.e., the x matrix used when fitting the nckqr object.
newx	A matrix of new values for x at which predictions are to be made. Note that newx must be of a matrix form, predict function does not accept a vector or other formats of newx.
s2	Value of the penalty parameter 'lambda2' at which predictions are required.
s1	Value(s) of the penalty parameter 'lambda1' at which predictions are required. Default is the entire sequence used to create the model.
...	Not used.

Value

Returns the fitted values for the non-crossing kernel quantile regression model.

Examples

```
library(MASS)
data(GAGurine)
x <- as.matrix(GAGurine$Age)
y <- GAGurine$GAG
l2 <- 1e-4
ttau <- c(0.1, 0.3, 0.5, 0.7, 0.9)
l1_list <- 10^seq(-8, 2, length.out=10)
fit <- nckqr(x,y, lambda1=l1_list, lambda2=l2, tau=ttau)
predict(fit, x, tail(x), s1=l1_list[1:3], s2=1e-4)
```

Index