Package 'spqrs'

July 4, 2018

Type Package

Title Simultaneous Estimation Of Quantile Regression Functions Using B-splines And Total Variation Penalty

Version 0.1.0

Author Jae-Hwan Jhong, Ja-yong Koo

Maintainer Jae-Hwan Jhong <jjh0925@korea.ac.kr>

Description This package contains functions for spqrs.

License GPL (>= 2)

Encoding UTF-8

LazyData true

Imports Rcpp (>= 0.12.13), ElemStatLearn

LinkingTo Rcpp

NeedsCompilation yes

R topics documented:

spqrs 1

Index **4**

`spqrs` *Simultaneous estimation of quantile regression functions using B-splines and total variation penalty*

Description

`spqrs` fits `spqrs` given data.

Usage

```
spqrs(response, predictor, tau, degree, dimension = 10, tau_penalty = 0.5,
      number_lambdas = 3, lambda_max = 1e+2, epsilon_lambda = 1e-10, maxiter = 500,
      epsilon_iterations = 1e-6, non_crossing = FALSE)
```

Arguments

`response`	Numeric response vector of length n.
`predictor`	Numeric predictor vector of length$n$.
`tau`	multiple quantile tau's interested. It should be sorted numeric vector within $[0, 1]$.
`degree`	Degree of B-splines. 0 fits constant, 1 fits linear, 2 fits quadratic, and 3 fits cubic splines.
`dimension`	Positive interger that decides maximum number of interior knots.
`tau_penalty`	Default is 0.5 that makes a general total variation penalty. It can be real value in $[0, 1]$.
`number_lambdas`	Numer of search for function smoothing
`lambda_max`	Positive real value that decides the maximum lambda value.
`epsilon_lambda`	Positive real value that decides the interval of lambda sequence. The smaller the value, the minimum lambda get smaller.
`maxiter`	Positive integer value that decides the maximum number of iterations.
`epsilon_iterations`	Positive real value that controls the iteration stopping criteria. In general, the smaller the value, convergence needs more iterations
`non_crossing`	Logical value that considers non-crossing constraint (TRUE) or not (FALSE). The non-crossing constraint only works when the `degree` is 0 and 1. Default is FALSE.

Details

This function develops an estimation of a finite number of quantile functions simultaneously using B-splines and the total variation penalty. For implementation of simultaneous quantile function estimators, a new coordinate descent algorithm taking care of a special structure of the total variation penalty determined by the B-spline coefficients is developed. For more details, see Jhong and Koo (2018).

Value

A list contains the entire fits of tuning parameter values. For example, `result[[i]]` indicates the fit of i th lambda. Last index of the list contains the meta data such as BIC, AIC, lambda sequence.

Author(s)

Jae-Hwan Jhong, Ja-Yong Koo

Source

This package is built on R version 3.4.3. with `Rcpp`.

References

Jae-Hwan Jhong and Ja-Yong Koo. "Simultaneous estimation of quantile regression functions using B-splines and total variation penalty." Submitted to Computational Statistics and Data Analysis, in revision.

Examples

```
#Toy example for spqrs

# clear
rm(list = ls())
library(ElemStatLearn)
library(spqrs)
# data
data(bone)
BMD = bone[, c(2, 4)]
BMD = BMD[!duplicated(BMD[, 1]), ]
order_BMD = order(BMD[, 1])
BMD_order = BMD[order_BMD, ]
x = as.vector(BMD_order[, 1])
y = as.vector(BMD_order[, 2])
N = length(x)
degree = 3
dimension = round(N / 2)
tau = c(0.1, 0.3, 0.5, 0.7, 0.9)
tau_penalty = 0.5
number_lambdas = 50
epsilon_lambda = 1e-10
# fit
prs = spqrs(response = y, predictor = x, tau = tau, degree = degree,
            dimension = dimension, tau_penalty = tau_penalty,
            number_lambdas = number_lambdas,
            lambda_max = 1e+2,
            epsilon_lambda = epsilon_lambda,
            non_crossing = FALSE)

aic_index = which.min(prs[[number_lambdas + 1]]$aic_vector)

par(mfrow = c(1, 1), pty = "s")
# Best fit by AIC
plot(x, y, col = "gray", cex = 1.5,
     xlab = "Age", ylab = "Relative Change in Spinal BMD", cex.lab = 1.5, cex.axis = 1.5)
leg = rep(0, length(tau))
leg[1] = expression(paste(tau, " = ", "0.1"))
leg[2] = expression(paste(tau, " = ", "0.3"))
leg[3] = expression(paste(tau, " = ", "0.5"))
leg[4] = expression(paste(tau, " = ", "0.7"))
leg[5] = expression(paste(tau, " = ", "0.9"))
legend("topright", legend = leg, lty = c(2:6), lwd = 1.5, cex = 1.5)
for(i in 1:length(tau))
{
   tt = knots2t(prs[[aic_index]]$knots_list[[i]] , degree = degree)
   basis = bsplines(x, tt, degree = degree)
   fit = basis %*% prs[[aic_index]]$coefficients_list[[i]]
   lines(x, fit, lwd = 2, lty = i + 1)
}
```

Index

spqrs, 1