Package ‘pbssim’

April 3, 2018

Type Package

Title Locally penalized single index model using B-splines and spherical coordinates

Version 0.1

Author Jae-Hwan Jhong, Jae-Young Kim, Jae-Deok Lee, Ja-yong Koo

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

Description This package contains functions for pbssim.

License GPL (>= 2)

Encoding UTF-8

LazyData true

Imports Rcpp (>= 0.12.13), msir

LinkingTo Rcpp

NeedsCompilation yes

R topics documented:

`pbssim` *Locally penalized single index model using B-splines and spherical coordinates*

Description

`bpssim` fits `bpssim` given data.

Usage

```
pbssim(responses, predictors, initial_alpha, degree, number_interior_knots,
       number_lambdas_alpha, number_lambdas_beta, lambda_alpha_max = 100,
       lambda_beta_max = 10, epsilon_lambda = 1e-6, maxiter = 200,
       epsilon_iterations = 1e-4)
```

Arguments

- `responses` Numeric response vector of length n.
- `predictors` Numeric predictor matrix of $n \times p$.
- `initial_alpha` Initial spherical coordinates. Numeric vector of length p.
- `degree` Degree of B-splines. 1 fits linear, 2 fits quadratic, and 3 fits cubic splines.
- `number_interior_knots` Positive interger that decides maximum number of interior knots.
- `number_lambda_alpha` Number of search for index vector
- `number_lambda_beta` Numer of search for function smoothing
- `lambda_alpha_max` Positive real value that decides the maximum lambda value for index vector
- `lambda_beta_max` Positive real value that decides the maximum lambda value for function smoothing
- `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

Details

This function develops an estimation of penalized single index regression model. Unknown link function is estimated by B-splines, index vector is estimated by approximation and least square method with spherical coordinates reparameterization. Two tuning parameters controls index sparsity and function smoothing each. Total variation penalty is adopted for function smoothing, and local penalty is adopted for variable selection. For more details, see Jhong et al(2018).

Value

A list contains the whole fits of grid search. First index of list is the order of lambda alpha, second index means the order of lambda beta. For example, `result[[i]][[j]]` indicates the fit of i th lambda alpha and j th lambda beta. Last index of the list contains the meta data such as BIC, AIC, lambda sequence

Author(s)

Jae-Hwan Jhong, Jae-Young Kim, Jae-deok Lee, Ja-Yong Koo

Source

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

References

Scrucca, Luca. "Model-based SIR for dimension reduction." Computational Statistics & Data Analysis 55.11 (2011): 3010-3026.

Friedman, Jerome, Trevor Hastie, and Rob Tibshirani. "Regularization paths for generalized linear models via coordinate descent." Journal of statistical software 33.1 (2010): 1.

Examples

```
#Toy example for pbssim

library(pbssim)
library(msir)

N = 200
p = 5
xi = c(1/sqrt(3), 1/sqrt(3), 1/sqrt(3), 0, 0)

x = matrix(runif(N * p, -1.5, 1.5), ncol = p)
single_index = colSums(t(x) * xi)

f = 5 * single_index
f[single_index < 0] = 0
y = f + rnorm(N) * 0.3

#PLot for true index and resposne
plot(sort(single_index), y[order(single_index)], col = 'gray')
lines(sort(single_index), f[order(single_index)], col = 'blue', lwd = 2)

##Model fit
#-----------------------------------------------
#If dimension is low, model-based SIR is
#good approach for initial index vector.
#In high dimensional case, one can use
#LASSO regression coefficients with glmnet()
#for initial index vector
#-----------------------------------------------

#msir
initial_xi = msir(x, y)$basis[ ,1]
initial_alpha = xi2alpha(initial_xi)
results = pbssim(y, x, initial_alpha, degree = 3, number_interior_knots = sqrt(N),
                 number_lambdas_alpha = 10, number_lambdas_beta = 10)

#Optimal fit with BIC criteria
bic_matrix = results[[11]]$bic_matrix
bic_optimal = which(bic_matrix == min(bic_matrix), arr.ind = TRUE) #(8, 5)th optimal!

#Fit results
optimal = results[[bic_optimal[1]]][[bic_optimal[2]]]

optimal$xi
plot(sort(single_index), y[order(single_index)], col = 'gray')
lines(sort(single_index), f[order(single_index)], col = 'blue', lwd = 2)
hat_single_index = colSums(t(x) * optimal$xi)
lines(sort(hat_single_index), optimal$fitted_value[order(hat_single_index)],
      col = 'red', lwd = 2)
legend('topleft', legend = c('True', 'pbssim'), lwd = 2, col = c('blue', 'red'))
```

Index