Package 'CommonSplines'

May 11, 2018

Title Regression Spline and Smoothing Spline

Version 1.0.0

Imports MASS

Date 2018-05-11

Authors Xingchen LIU <e0225109@u.nus.edu>, Yuchen SHI <yuchenshinus@gmail.com>, Xiaozhou Yang <yang_xiaozhou@icloud.com>

Description This is an R package that covers commonly seem regression spline and smoothing spline. For regression spline, commonly seen basis functions are provided such as truncated power basis, natural spline basis and B-spline basis. For smoothing spline, penalties on second order derivative are provided, i.e., cubic smoothing spline.

Depends R (>= 3.3.2)

License Apache License 2.0

Encoding UTF-8

LazyData true

RoxygenNote 6.0.1

Suggests knitr, rmarkdown

VignetteBuilder knitr

R topics documented:

bs_basis *Generating B-spline basis*

Description

This function generates B-spline basis. The B-splines are defined following the recursive formulas due to de Boor. Only univariate input can be used.

Usage

```
bs_basis(x, y, order, innerknots)
```

Arguments

x	The input vector of training dataset.
y	The output vector of training dataset.
order	The order of B-spline functions. The default is order=4 for cubic B-splines.
innerknots	The internal knots that define the spline. innerknots should not contain knots on the boundary.

Value

A list with the following components:

beta	The coefficients of nonparametric regression.
basis	The B-spline basis matrix of dimension c(length(x), df). df = length(innerknots) + order.
knots	The knots used to construct the B-splines, including innerknots, boundary knots and phantom knots
order	The order of basis functions. order=degree+1

Examples

```
x<-seq(0, 1, 0.001)
y <- x^3 * 3 - x^2 * 2 + x + exp(1)+rnorm(length(x),0,0.1)
plot(x,y)
innerknots <- seq(0.1, 0.9, 0.1)
order<-4

basis<-bs_basis(x,y,order,innerknots)
plot(x,rep(0,length(x)),type="l",ylim=c(0,1))
for (i in 1: (length(innerknots)+order)){
  lines(x,basis$basismatrix[,i])
}
```

bs_predict *Regression using B-spline basis*

Description

This function provides nonparametric regressions using B-splines. The B-splines are generated by the function `bs_basis`. The return value of `bs_basis` is required as an argument of `bs_predict`

Usage

```
bs_predict(x_test, basis)
```

Arguments

`x_test`	The input values at which evaluations are required.
`basis`	The return value of function `bs_basis`.

Value

The evaluated output at x_test.

Examples

```
x<-seq(0, 1, 0.001)
y <- x^3 * 3 - x^2 * 2 + x + exp(1)+rnorm(length(x),0,0.1)
plot(x,y)
innerknots <- seq(0.1, 0.9, 0.01)
order<-4
basis<-bs_basis(x,y,order,innerknots)

x_test<-seq(0, 1, 0.01)
fit<-bs_predict(x_test,basis)
plot(x_test,fit)
lines(x_test,x_test^3 * 3 - x_test^2 * 2 + x_test + exp(1),col="red")
```

cal_loo_cv_error *Calculte leave-one-out CV error*

Description

Calculte leave-one-out CV error

Usage

```
cal_loo_cv_error(y, f_hat, S)
```

Arguments

`y`	response variable values
`f_hat`	fitted response variable values
`S`	smoother matrix

Value

leave-one-out cross-validation error

`css_predict` *Prediction using smoothing spline with squared 2nd derivative penalty*

Description

This function takes the coefficients trained by `CubicSmoothingSpline.Train` and evaluate the output at x_test

Usage

```
css_predict(basis, x_test)
```

Arguments

`basis`	The return value of function `CubicSmoothingSpline.Train`.
`x_test`	The input values at which evaluations are required.

Value

The evaluated output at x_test.

Examples

```
x<-seq(0, 1, 0.0015)
y <- x^3 * 3 - x^2 * 2 + x + exp(1)+rnorm(length(x),0,0.1)
plot(x,y)
lambda<-0.001
basis<-css_train(x,y,lambda)

x_test<-seq(0, 1, 0.1)
fit<-css_predict(basis,x_test)

plot(x_test,fit)
lines(x_test,x_test^3 * 3 - x_test^2 * 2 + x_test + exp(1),col="red")
```

`css_train` *Train a smoothing spline with squared 2nd derivative penalty using natural cubic spline*

Description

This function trains a smoothing spline with squared 2nd derivative penalty. It has an explicit,finite-dimensional, unique minimizer which is a natural cubic spline. This function can be used for small or moderate number of knots. When the number of data N<=50, all knots are included. When N>50, 50 knots are uniformly chosen from the training dataset.

Usage

```
css_train(x, y, lambda)
```

Arguments

x	The input vector of training dataset.
y	The output vector of training dataset.
lambda	A fixed smoothing parameter.

Value

A list with the following components:

beta	The coefficients of natural splines.
S	The smoother matrix.
knots	The knots used to construct the B-splines, including innerknots, boundary knots and phantom knots

Examples

```
x<-seq(0, 1, 0.001)
y <- x^3 * 3 - x^2 * 2 + x + exp(1)+rnorm(length(x),0,0.1)
plot(x,y)
lambda<-0.001

basis<-css_train(x,y,lambda)
cat("the knots chosen are: ",basis$knots)
```

---

ncs *Regression using natural cubic splines*

Description

This function provides regression using natural cubic splines with truncated power basis functions. Only univariate input can be used.

Usage

```
ncs(x_train, y_train, x_test, df = NULL, knots = NULL)
```

Arguments

x_train	The input vector of training dataset.
y_train	The output vector of training dataset.
x_test	The input values at which evaluations are required.
df	Degrees of freedom. One can supply df rather than knots; ncs() then chooses (df + 1) knots at uniform quantiles of x. The default, df = 4, sets 5 knots with 3 inner knots at uniform quantiles of x.
knots	Breakpoints that define the spline. The default is five knots at uniform quantiles c(0, .25, .5, .75, 1). Typical values are the mean or median for one knot, quantiles for more knots.

Value

y_pred	A vector of dimension length(x), the prediction vector evaluated at x_test values.

Examples

```
x_train <- seq(1, 10, 0.1)
y_train <- cos(x_train)^3 * 3 - sin(x_train)^2 * 2 + x_train + exp(1)+rnorm(length(x_train),0,1)
plot(x_train,y_train)
x_test <- seq(1, 10, 0.1)
lines(x_test,cos(x_train)^3 * 3 - sin(x_train)^2 * 2 + x_train + exp(1),col="red")

df <- 2
y_pred <- ncs(x_train, y_train, x_test, df)
lines(x_test,y_pred, col='blue')
df <- 4
y_pred <- ncs(x_train, y_train, x_test, df)
lines(x_test,y_pred, col='green')
df <- 10
y_pred <- ncs(x_train, y_train, x_test, df)
lines(x_test,y_pred, col='black')
legends <- c("Actual", "Prediction: 2 df", "Prediction: 4 df", "Prediction: 10 df")
legend('topleft', legend=legends, col=c('red', 'blue', 'green', 'black'), lty=1, cex=0.8)
title('Smoothing Comparison of Different Degrees of Freedom')
```

---

ncs_eval_basis *Evaluate basis functions as each x and return the evaluated basis matrix N*

Description

Evaluate basis functions as each x and return the evaluated basis matrix N

Usage

```
ncs_eval_basis(x, knots, nknots)
```

Arguments

x	Predictor variable vector.
knots	Knots location in terms of quantiles of x_train, optional, default will be evenly spaced quantiles based on number of knots.
nknots	Number of knots useded in training.

Value

Basis matrix evaluated at each x value.

ncs_predict *Prediction based on trained regression model*

Description

Prediction based on trained regression model

Usage

```
ncs_predict(x_test, betas, knots, nknots)
```

Arguments

x_test	The input values at which evaluations are required.
betas	Least sqaure fit parameters obtained from training.
knots	Knots location in terms of quantiles of x_train, optional, default will be evenly spaced quantiles based on number of knots.
nknots	Number of knots used in training.

Value

y_pred	A vector of dimension length(x), the prediction vector evaluated at x_test values.

ncs_train *Generate an evaluated basis matrix for natural cubic splines*

Description

Generate an evaluated basis matrix for natural cubic splines

Usage

```
ncs_train(x_train, y_train, df = NULL, knots = NULL)
```

Arguments

x_train	The input vector of training dataset.
y_train	The output vector of training dataset.
df	Degrees of freedom. One can supply df rather than knots; ncs() then chooses (df + 1) knots at uniform quantiles of x. The default, df = 4, sets 5 knots with 3 inner knots at uniform quantiles of x.
knots	Breakpoints that define the spline, in terms of quantiles of x. The default is five knots at uniform quantiles c(0, .25, .5, .75, 1). Typical values are the mean or median for one knot, quantiles for more knots.

Value

A list of following components:

nknots	Number of knots.
knots	A vector of knot locations.
N	Basis matrix evaluated at each x value.
betas	Least sqaure fit parameters.

Examples

```
x_train <- seq(1, 10, 0.1)
y_train <- cos(x_train)^3 * 3 - sin(x_train)^2 * 2 + x_train + exp(1)+rnorm(length(x_train),0,1)
plot(x_train,y_train)
x_test <- seq(1, 10, 0.1)
df <- 10
train_result <- ncs_train(x_train, y_train, df)
print(train_result$betas)
print(train_result$N[1:5,1:5])
```

---

place_knots *Find evenly spaced knots by quantile*

Description

Knots found include boundary knots at 0th and 100th quantile.

Usage

```
place_knots(nknots, x)
```

Arguments

nknots	Number of knots to be located.
x	Data vector on which knots are placed.

Value

A named vector with knot quantiles and values.

PowerBasisSpline *Regression using Power Basis spline*

Description

This function is a generalization of CubicPowerBasisSpline with arbitrary order

Usage

```
PowerBasisSpline(x, y, x_test, order, innerknots)
```

Arguments

x	The input vector of training dataset.
y	The output vector of training dataset.
x_test	The input values at which evaluations are required.
order	The order that defines the spline.
innerknots	The internal knots that define the spline.

Details

Only univariate input can be used.

Value

A list with the following components:

beta	The coefficients of nonparametric regression.
basis	The spline basis matrix of dimension c(length(x), NumKnots+order)
f	The evaluated output at x_test.

Examples

```
n <- 100
t <- seq(0,2*pi,length.out = 100)
a <- 3
b <- 2
c.unif <- runif(n)
amp <- 2
set.seed(1)
y1 <- a*sin(b*t)+c.unif*amp # uniform error
innerknots <- 2*pi*c(1/4,2/4,3/4)
order <- 4
solution <- PowerBasisSpline(t,y1,t,order,innerknots)
y.hat <- solution$f
plot(t, y1, t="l")
lines(t, y.hat, col=2)
```

`sel_smoothing_para` *Select smoothing parameter based on leave-one-out CV error*

Description

Select smoothing parameter based on leave-one-out CV error

Usage

```
sel_smoothing_para(x, y, cv_lambda)
```

Arguments

`x`	predictor variable
`y`	response variable
`cv_lambda`	vector of candidate lambda values

Value

lamdba value that minimizes leave-one-out CV error

Index