Package ‘switchnpreg’
February 20, 2015

Type Package
Title Switching nonparametric regression models for a single curve and functional data
Version 0.8-0
Date 2013-07-10
Author Camila de Souza <camila@stat.ubc.ca> and Davor Cubranic <cubranic@stat.ubc.ca>
Maintainer Davor Cubranic <cubranic@stat.ubc.ca>
Description Functions for estimating the parameters from the latent state process and the functions corresponding to the J states as proposed by De Souza and Heckman (2013).
License GPL-3
Depends MASS, splines, fda
Imports expm, HiddenMarkov
NeedsCompilation no
Repository CRAN
Date/Publication 2013-07-11 07:32:58

R topics documented:

switchnpreg .......................................................... 1

Index

switchnpreg Fit a switching nonparametric regression model

Description

Estimate the parameters of a switching nonparametric regression model using the EM algorithm as proposed by De Souza and Heckman (2013). The package allows two different estimation approaches (Bayesian and penalized log-likelihood) and two different types of hidden states (iid and Markov). The smoothing parameters are chosen by cross-validation. Standard errors for the estimates of the parameters governing the distribution of the state process are also provided.
Usage

switchnpreg(x, y, f, alpha, sigma2, lambda, ..., 
   method = c("pl", "bayes"), var.equal = TRUE, 
   z.indep = TRUE, eps.cv, eps.em, maxit.cv, maxit.em)

Arguments

x  The sequence of covariates $x_1, \ldots, x_n$.
y  The sequence of response variables $y_1, \ldots, y_n$.
f  The $n \times J$ matrix of initial values for the functions, where column $j$ corresponds to the function $f_j$.
alpha  The initial values for the parameters of the latent state process. If the latent states are iid alpha is a vector containing the initial mixing proportions $p_j$ for $j = 1, \ldots, J$. If the latent states follow a Markov structure then alpha is a list of two components: A and PI, where A is the initial $J \times J$ matrix of transition probabilities A and PI is the initial $J$-vector of initial probabilities.
sigma2  The initial $J$-vector of regression error variances.
lambda  The initial $J$-vector of smoothing parameters.
...  Optional arguments to parameter update functions.
method  Character string 'pl' or 'bayes' to choose whether the model is fitted using the penalized log-likelihood approach or the Bayesian approach, respectively.
var.equal  Logical indicating whether $\sigma_j^2$ are equal for all $j$.
z.indep  Logical indicating whether the hidden states $z_i, \ldots, z_n$ are considered iid or Markovian.
eps.cv  Convergence value for the cross-validation procedure.
eps.em  Convergence value for the EM algorithm.
maxit.cv  Maximum number of iterations of the EM+CV procedure.
maxit.em  Maximum number of iterations of each EM loop.

Value

A list with following elements:
current  The final estimate of $\theta$, represented as a list with the elements named after the respective model parameter:
f  The final function estimates.
sigma2  The final variance estimates.
alpha  The final estimates for the parameters of the latent state process.
pij  The matrix of size $n \times J$ with $ij$-th element giving the final estimate of $p(z_i = j | y, \theta)$.
lambda  Chosen smoothing parameters.
iter.cv  Number of iterations of the EM+CV procedure.
stderr  Standard errors for the parameter estimates of the latent state process.
switchnpreg

Author(s)
Camila de Souza <camila@stat.ubc.ca> and Davor Cubranic <cubranic@stat.ubc.ca>.

References

de Souza and Heckman (2013), “Switching nonparametric regression models and the motorcycle

See Also
demo(simulated_data_indep_example), demo(simulated_data_Markov_example)

Examples

```r
## The motorcycle data set revisited ##
x <- MASS::mcycle$times
set.seed(30)
x[duplicated(x)] <- round(jitter(x[duplicated(x)]), 3)
y <- MASS::mcycle$accel
n <- length(y)
spline_fit <- smooth.spline(x, y)

## set up the initial functions
f.initial <- t(apply(as.matrix(spline_fit$y), 1, `*`, c(30, 0, -30)))
J <- ncol(f.initial)
sig2 <- rep((sum((y - predict(spline_fit, x)$y)^2) / (n - spline_fit$df))/J, J)

## B and R parameters for penalized log-likelihood method
basis <- create.bspline.basis(range(x), nbasis = 40)
B <- getbasismatrix(x, basis)
R <- getbasispenalty(basis)

estimates <- switchnpreg(x = x, y = y,
                         f = f.initial,
                         alpha = rep(1, J) / J,
                         sigma2 = sig2,
                         lambda = rep(.5, J),
                         B = B, R = R,
                         var.equal = FALSE,
                         interval = log(c(1E-4, 1E3)),
                         eps.cv = rep(1E-1, J),
                         eps.em = rep(c(1E-1, 1E-2, 1E-3), each = J),
                         maxit.cv = 10,
                         maxit.em = 100)
```
plot(x, y, ylim = c(-150, 90),
     ylab = 'Head acceleration',
     xlab = 'Time')
matlines(x, estimates$current$f, type='l', lty = 1, col = 1:J)
matlines(sort(x), f.initial, lty = 2, col = 'gray')
Index

*Topic latent
switchnpreg, 1
*Topic nonparametric
switchnpreg, 1
*Topic regression
switchnpreg, 1
*Topic switching
switchnpreg, 1

switchnpreg, 1