Package ‘flars’

June 3, 2016

Type Package
Title Functional LARS
Version 1.0
Date 2016-05-28
Author Yafeng Cheng, Jian Qing Shi
Maintainer Yafeng Cheng <yafeng.cheng@mrc-bsu.cam.ac.uk>
License GPL (>= 2)
Depends R (>= 3.2.0), MASS
Imports fda, Matrix, parallel, Rcpp
LinkingTo Rcpp(>= 0.12.0), RcppEigen
NeedsCompilation yes
Repository CRAN
Date/Publication 2016-06-03 20:02:02

R topics documented:

flars-package .............................................................. 2
data_generation ......................................................... 2
fccaGen ................................................................. 4
fccaXX ................................................................. 6
fccaXXcv ............................................................... 7
flars ................................................................. 8
flars_TrainTest .......................................................... 11
predict.flars ............................................................ 12
RealDa ................................................................. 13

Index 15
flars-package

*Functional least angle regression for functional linear regression with scalar response and mixed scalar and functional covariates.*

**Description**

This is a package for the variable selection problem in the functional linear regression model. The model we target on has a scalar response, in other words, a continuous random variable following Normal distribution. The candidate covariates could be either functional or scalar or a mixture of the two. The algorithm is able to do selection when number of candidate variables is larger than the sample size. The efficiency is from the idea of the Least Angle Regression and the stopping rule that we designed for this algorithm.

**Details**

<table>
<thead>
<tr>
<th>Package</th>
<th>flars</th>
</tr>
</thead>
<tbody>
<tr>
<td>Type</td>
<td>Package</td>
</tr>
<tr>
<td>Version</td>
<td>1.0</td>
</tr>
<tr>
<td>Date</td>
<td>2016-05-28</td>
</tr>
<tr>
<td>License</td>
<td>GPL (&gt;= 2)</td>
</tr>
</tbody>
</table>

**Author(s)**

Yafeng Cheng, Jian Qing Shi

Maintainer: Yafeng Cheng <yafeng.cheng@mrc-bsu.cam.ac.uk>

**References**


data_generation

*Data generation function for examples.*

**Description**

This function generates a few types of data with different correlation structures. The generated data can be used in the examples provided in other functions such as the calculation of the functional canonical correlation analysis and the functional least angle regression.
Usage

data_generation(seed, nsamples=80, hyper=NULL, var_type=c('f', 'm'),
                   cor_type=1:6, uncorr=TRUE, nVar=8)

Arguments

seed  Set the seed for random numbers.
nsamples  Sample size of the data to generate.
hyper  Hyper parameters used in the Gaussian process (GP). GP is used for building
       the covariance structure of the functional variables.
var_type  Two choices of the variable types. See details for more information.
cor_type  Correlation structures. See details for more information.
uncorr  Whether the variables are built based on linearly uncorrelated variables. See
        details for more information.
nVar  Number of base variables to generate. Note that this is not the exact number of
       variables generated at the end.

Details

type could be either 'f' or 'm'. If var_type='f', only functional variables will be generated.
If var_type='m', both functional variables and scalar variables will be generated.

When uncorr is TRUE, a few linearly uncorrelated variables will be generated. This is to better
control the correlation structure of the variables using cor_type. If you want to generate a large
number of variables, uncorr should be FALSE.

cor_type are numbers from 1 to 6 or from 1 to 4 depending on the choices of var_type. This
is ONLY useful when we use the default number of variables, i.e., nVar=8 and the initial variables
are linearly uncorrelated, i.e., uncorr=TRUE. Bigger value of cor_type means more complicated
 correlation structures.
If no correlation restriction is required for the variables, we can use cor_type=1.

nVar is the number of the base variables generated. It is recommanded that users can modify the
function to get their own data set. The other way is to use this function repeatedly to get enough both
functional and scalar variables. The response variable can be re-generated by the user. Increasing
the value of this argument may give NaN for the response variables.

Value

x  List of covariates.
y  Response variable.
BetaT  True shape of the functional coefficients and true values of the scalar variables.
bConst  Normalizing constants of the functional coefficients. True functional coefficients are the shape times the corresponding normalizing constant.
noise  Random noise.
mu  True intercept.
Examples

library(flars)

data(t=data_generation(seed = 1, uncorr = TRUE, nVar = 8, nsamples = 120, 
  var_type = ‘f’, cor_type = 1))

fccagen

Description

This function carries out the canonical correlation analysis between a scalar variable and a list
of mixed scalar and functional variables. There are four choices of the returned values and three
representation methods of the functional variables.

Usage

fccagen(xL, yVec, type = c(’dir’, ’cor’, ’a’, ’all’), method = c(’basis’, ’gq’, ’raw’), GCV = TRUE, control = list())

Arguments

- **xL**: The mixed scalar and functional variables. If there is only one functional variable, xL can be a matrix. If there is only scalar variables, xL can be a vector or a matrix. If there are more than one functional variables, or there are mixed functional and scalar variables, xL should be a list. If xL is a list, each item of the list should correspond to one variable.
- **yVec**: The scalar variable. It should be a matrix.
- **type**: The choice of outcomes. See details for more information.
- **method**: The representative methods for the functional coefficients. The method could be one of the ’basis’, ’gq’ and ’raw’ for basis function expression, Gaussian quadrature and representative data points, respectively.
- **GCV**: Use generalized cross validation (GCV) or not to choose the tuning parameter. Logic argument. Currently the only choice is to use GCV.
- **control**: List of elements that controls the details of the functional coefficients. See details for more information.

Details

There are four choices of type in the function. ’dir’ means that the function only returns the direction coefficients like the one in the traditional Canonical correlation analysis. ’cor’ means that the function only returns the correlation coefficients. ’a’ means that the function only returns the normalized direction coefficients. With this normalization, the direction coefficients are equivalent to the coefficients from a linear regression with response variable yVec and covariates xL. ’all’ means that the function returns all three outcomes mentioned above.

The argument control is a list. It changes when different representative methods are used for the functional coefficients. If (type==’basis’), the list contains the following items:
• nbasis: Number of B-spline basis functions. Default value is 18.
• norder: Order of the basis functions. Default value is 6.
• pen1: The candidate values of the smoothing parameter. Default values are $10^\ast(seq((-20),5,\text{len}=41))$
• pen2: The candidate values of the ridge tuning parameter. Default value is 0.01
• t: IMPORTANT! The time points correspond to the discrete data points of the functional variables. Default to be $\text{seq}(0,1,\text{len}=\max(sapply(xL,ncol),\text{na.rm}=T))$. Do NOT change the starting and ending point of the sequence.

If (type=='gq'), the list contains the following items:

• nP: Number of Gaussian quadrature points. Default value is 18.
• pen1: The candidate values of the smoothing parameter. Default values are $10^\ast(seq((-20),5,\text{len}=21))$
• pen2: The candidate values of the ridge tuning parameter. Default value is 0.01
• t: IMPORTANT! The time points correspond to the discrete data points of the functional variables. Default to be $\text{seq}(-1,1,\text{len}=\max(sapply(xL,ncol),\text{na.rm}=T))$. Do NOT change the starting and ending point of the sequence.

If (type=='raw'), the list contains the following items:

• pen1: The candidate values of the smoothing parameter. Default values are $10^\ast(seq((-20),5,\text{len}=21))$
• pen2: The candidate values of the ridge tuning parameter. Default value is 0.01
• t: IMPORTANT! The time points correspond to the discrete data points of the functional variables. Default to be $\text{seq}(0,1,\text{len}=\max(sapply(xL,ncol),\text{na.rm}=T))$. Do NOT change the starting and ending point of the sequence.

The function is designed to be able to handle the situation when different functional variables have different number of discrete data points and the discrete data points could be non-evenly spaced. This would require a list of t to input in the argument. However, this is not fully tested at the moment. For convenient, especially when we have a large number of functional variables, a universal setting of t is recommended.

**Value**

corr: Correlation coefficient. It is returned when type='corr' or type='all'.
a: Normalized direction coefficients. It is returned when type='a' or type='all'.
dir: Direction coefficients. It is returned when type='dir'.
K: Penalized covariance matrix. It is returned when type='all'.
gq: Gaussian quadrature weights. It is returned when type='all'.
phiL: Known part of the functional coefficients. E.g, basis functions. It is returned when type='all'.
S: Hat matrix. It is returned when type='all'.
lam1: The selected smoothing parameter. It is returned when type='all'.
lam2: The selected ridge parameter. It is returned when type='all'.
GCV_mat: The GCV value. It is returned when type='all'.
TraceHat: Trace of the hat matrix. It is returned when type='all'.
Examples

library(flars)
# Generate some data.
data1=data_generation(seed = 1, uncorr = TRUE, nVar = 8, nsamples = 120,
    var_type = 'm', cor_type = 1)

# If there is only one functional variable
# out1=fccagen(dataL$x[1], dataL$y, type='all', method='basis')

# If there are only a few scalar variables
# x=matrix(rnorm(3*length(dataL$y)),ncol=3)
# out2=fccagen(x, dataL$y, type='all', method='basis')

# If there are mixed scalar and functional variables
# out3=fccagen(dataL$x, dataL$y, type='all', method='basis')

fccaxx  Canonical correlation analysis between two groups of mixed functional and scalar variables

Description

This function carries out the canonical correlation analysis between two groups of mixed functional and scalar variables. Three different representing methods can be used for the functional coefficients. The tuning parameters should be specified in the arguments control1 and control2 for the two groups xL1 and xL2, respectively.

Usage

fccaxx(xL1,xL2,centre=TRUE,method=c('basis','gq','raw'),control1=list(),
    control2=list(),tol=1e-7)

Arguments

xL1  The mixed scalar and functional variables. For any number and any type of variables, xL1 should be a list. Each item of the list should correspond to one variable.

xL2  Same as xL1.

centre  Logic argument. Default is TRUE, which means the variables do need to be centred.

method  The representative methods for the functional coefficients. The method could be one of the 'basis', 'gq' and 'raw' for basis function expression, Gaussian quadrature and representative data points, respectively.

control1  List of elements that controls the details of the functional coefficients for xL1. See details for more information. See the argument control in function fccagen for details.

control2  Similar to control1.

tol  The threshold to decide whether the correlation is to small to be non-zero.
Details

This function uses Moore-Penrose generalized inverse in the calculation to avoid singular problem.

Value

corr
All the non-zero canonical correlation.

coef1
The corresponding coefficients (weights) for the x1.

coef2
The corresponding coefficients (weights) for the x2.

Examples

```r
# library(flasr)
# library(fda)
## Generate some data sets.
# dataL1=data_generation(seed = 1, uncorr = FALSE, nVar = 8, nsamples = 120,
#     var_type = 'm', cor_type = 1)
# dataL1=dataL1$x

# dataL2=data_generation(seed = 2, uncorr = FALSE, nVar = 8, nsamples = 120,
#     var_type = 'm', cor_type = 1)
# dataL2=dataL2$x

## cross validation
# outCV=fccaXXcv(xL1 = dataL1[1:2], xL2 = dataL2[1:2], method = 'basis'
#     , alpha = 10^{seq(-6,0, len=5)})

# cvCor=outCV$cor
# calculate the correlation
# out=fccaXX(dataL1, dataL2, method = 'basis', control1 = list(pen1=
#     outCV$alpha[which.max(cvCor)]), control2 = list(pen1=
#     outCV$alpha[which.max(cvCor)]))
```

```
This function finds the best smoothing parameter for the canonical correlation analysis for both groups of variables by using leave-one-out (sample) cross validation. The criterion here is to maximise the first canonical correlation.

Description

This function carries out the canonical correlation analysis between a scalar variable and a list of mixed scalar and functional variables. There are four choices of the returned values and three representation methods of the functional variables.

Usage

```r
fccaXXcv(xL1,xL2,method=c('basis','gg','raw'),centre = TRUE,tol=1e-7,
Control1=list(),Control2=list(),alpha=10^{seq(-6,1, len=10)})
```

Arguments

xL1 The mixed scalar and functional variables. For any number and any type of variables, xL1 should be a list. Each item of the list should correspond to one variable.

xL2 Same as xL1.

method The representative methods for the functional coefficients. The method could be one of the 'basis', 'gq' and 'raw' for basis function expression, Gaussian quadrature and representative data points, respectively.

centre Logic argument. Default is TRUE, which means the variables do need to be centred.

tol The threshold to decide whether the correlation is too small to be non-zero.

Control1 List of elements that controls the details of the functional coefficients for xL1. See details for more information. See the argument control in function fccaGen for details.

Control2 Similar to Control1.

alpha Candidate tuning parameters for the smoothness of the functional coefficients.

Details

Note that the smoothing parameters for both groups of variables are assumed to be the same. This is due to high computational cost of cross validation. See the example in fccaXX.

Value

cor A vector of the first canonical correlation. Each element of the vector is corresponding to one of the candidate tuning parameters.

alpha The corresponding tuning parameters.

flars Functional least angle regression.

Description

This is the main function for the functional least angle regression algorithm. Under certain conditions, the function only needs the input of two arguments: x and y. This function can do both variable selection and parameter estimation.

Usage

flars(x,y,method=c('basis','gq','raw'),max_selection,cv=c('gcv'),
       normalize=c('trace','rank','norm','raw'),lasso=TRUE,check=1,
       select=TRUE,VarThreshold=0.1,SignThreshold=0.8,
       control=list())
Arguments

\( x \)  
The mixed scalar and functional variables. Note that each of the functional variables is expected to be stored in a matrix. Each row of the matrix should represent a sample or a curve. If there is only one functional variable, \( x \) can be a matrix. If there is only scalar variables, \( x \) can be a vector or a matrix. If there are more than one functional variables, or there are mixed functional and scalar variables, \( x \) should be a list. If \( x \) is a list, each item of the list should correspond to one variable.

\( y \)  
The scalar variable. It can be a matrix or a vector.

\( \text{method} \)  
The representative methods for the functional coefficients. The method could be one of the 'basis', 'gq' and 'raw' for basis function expression, Gaussian quadrature and representative data points, respectively.

\( \text{max_selection} \)  
The number of maximum selections when stopping the algorithm. Set a reasonable number for this argument to increase the calculation speed.

\( \text{cv} \)  
The choice of cross validation. At the moment, the only choice is the generalized cross validation, i.e., cv='gcv'.

\( \text{lasso} \)  
Use lasso modification or not. In other words, can variables selected in the former iterations be removed in the later iterations.

\( \text{check} \)  
The type of check methods for lasso modification. 1 means variance check, 2 means sign check. check=1 is much better than the other one.

\( \text{select} \)  
If TRUE, the aim is to do selection rather than parameter estimation, and the stopping rule can be used when lasso=TRUE. If FALSE, the stopping rule may not work when lasso=TRUE.

\( \text{VarThreshold} \)  
Threshold for removing variables based on variation explained. More specifically, one condition to remove a variable is that the variation explained by a variable is less than \( \text{VarThreshold} \times \text{Var}(y) \). To remove this variable, there is another condition: the variation explained by this variable is less than largest variation it explained in the previous iterations.

\( \text{SignThreshold} \)  
This is a similar argument to \( \text{VarThreshold} \). If a functional coefficient has less than \( \text{SignThreshold} \) same as that from the previous iteration, the variable is removed.

\( \text{normalize} \)  
The choice of normalization methods. This is to remove any effects caused by the different dimensions of functional variables and scalar variables. Currently we have trace, rank, norm, raw. norm and raw are recommended.

\( \text{control} \)  
List of control elements for the functional coefficients. See \textit{fccagen} for details.

Value

\( \text{Mu} \)  
Estimated intercept from each of the iterations

\( \text{Beta} \)  
Estimated functional coefficients from each of the iterations

\( \text{alpha} \)  
Distance along the directions from each of the iterations

\( \text{p2_norm} \)  
Normalization constant applied to each of the iterations

\( \text{AllIndex} \)  
All the index. If one variable is removed, it will become a negative index.
index: All the index at the end of the selection.
CD: Stopping rule designed for this algorithm. The algorithm should stop when this value is very small. Normally we can observe an obvious and severe drop of the value.
resid: Residual from each of the iteration.
RowMeans: Point-wise mean of the functional variables and mean of the scalar variables.
RowSds: Point-wise sd of the functional variables and sd of the scalar variables.
yMean: Mean of the response variable.
ySD: SD of the response variable.
p0: The projections obtained from each iteration without normalization.
cor1: The maximum correlation obtained from the first iteration.
lasso: Weather have lasso step or not.
df: The degrees of freedom calculated at the end of each iteration.
Sigma2Bar: Estimated $\sigma^2$.
StopStat: Conventional stopping criteria.
varSplit: The variation explained by each of the candidate variables at each iteration.
SignCheckF: The proportion of sign changing for each of the candidate variables at each iteration.

Examples

```r
library(flars)
library(fda)
### Ex1 ###
## Generate some data.
dataL = data_generation(seed = 1, uncorr = TRUE, nVar = 8, nsamples = 120,
                             var_type = 'm', cor_type = 3)

## Do the variable selection
outL = flars(dataL$x, dataL$y, method='basis', max_selection=9,
             normalize='norm', lasso=FALSE)

## Check the stopping point with CD
plot(2:length(out$alpha), out$CD) # plot the CD with the iteration number

## In simple problems we can try
(iterL = which.max(diff(out$CD)) + 2)

### Ex2 ###
## Generate some data.
# dataL = data_generation(seed = 1, uncorr = FALSE, nVar = 8, nsamples = 120,
#                         var_type = 'm', cor_type = 3)
## add more variables to the candidate
## for(i in 2:4){
##   dataL0 = data_generation(seed = i, uncorr = FALSE, nVar = 8, nsamples = 120,
```
flars_TrainTest

Internal function for doing simulation using functional lars.

Description
This is a function built for doing data generation and variable selection using functional lars with different settings and data with different correlation structures.

Usage
flars_TrainTest(seed=1,nsamples=120,nTrain=80,var_type=c(‘f’,’m’),
VarThreshold0=0.1,SignThreshold0=0.8,cor_type=1:5,
lasso=TRUE, check = 1,uncorr=T,nVar=8,Discrete_Norm_ID=1:12,
NoRaw_max=12,raw_max=9,hyper=NULL,RealX=NULL,RealY=NULL,
data==NULL,nCor=0,control=list())

Arguments
seed Set the seed for random numbers.
samples Sample size of the data to generate.
nTrain Sample size of the training data.
var_type Two choices of the variable types. See details for more information.
cor_type Correlation structures. See details for more information.
VarThreshold0 Threshold for removing variables based on variation explained. See flars for more details.
predict.flars

SignThreshold: Same as VarThreshold
lasso: Use lasso modification or not. In other words, can variables selected in the former iterations be removed in the later iterations.
check: Type of lasso check. 1 means variance check, 2 means sign check. check=1 is much better than the other one.
uncorr: If the variables are uncorrelated or not. See details for more information.
nVar: Number of variables to generate.
Discrete_Norm_ID: Which discrete method and which norm to use. 1 to 12.
NoRaw_max: Number of variables to select when not using RDP discretising method.
raw_max: Number of variables to select when using RDP discretising method.
hyper: Hyper parameters used in the Gaussian process. GP is used for building the covariance structure of the functional variables.
RealX: Real data input X.
RealY: Real data input Y.
dataL: Real input data list rather than generate in the function. It should has the same structure as that generated.
nCor: Number of cores to use.
control: List of control items. See fccaGen for more details.

Value
A list of results using different normalization methods and different representation methods for the functional coefficients.


Description
This is the function that carries out the prediction of the new observations.

Usage
## S3 method for class 'flars'
predict(object, newdata, ...)

Arguments
object: This must be a flars object from the function flars.
newdata: A list of new observations. The format of this set of data must be the same as the training data, including the order of the variables.
...: Other arguments to input.
Value

A matrix of predictions. Since the input flars object may have more than one estimated coefficients, the number of predictions may be more than one set. Each column of the outcome is corresponding to one set of coefficients.

Examples

```r
library(flars)
library(fda)
## Generate some data.
dataL=data_generation(seed = 1,uncorr = TRUE,nVar = 8,nsamples = 120,
   var_type = 'm',cor_type = 3)

## Split the training data and the testing data
nTrain=80
nsamples=120
TrainIdx=seq(nTrain)
TestIdx=seq(nsamples)[-TrainIdx]
fsmTrain=lapply(dataL$X,function(fsmI) fsmI[TrainIdx,,drop=FALSE])
fsmTest=lapply(dataL$X,function(fsmI) fsmI[TestIdx,,drop=FALSE])
yTrain=dataL$y[TrainIdx]
yTest=dataL$y[TestIdx]

## Do the variable selection
out=flars(fsmTrain,yTrain,method='basis',max_selection=9,
   normalize='norm',lasso=FALSE)

## Do the prediction
pred=predict(out,newdata = fsmTest)

# apply(pred,2,flars:::rmse,yTest)
```

RealDa  

A subset of real data from Limbs Alive project.

Description

A subset of scalar response variable, functional variables and scalar variables.

Usage

RealDa

Format

A list with one response and one list of covariates. The covariates contains 15 functional variables and 11 scalar variables. Each element of the covariates is a matrix.
Details

The data set is from Limbs Alive project. The functional variables are trajectories from patients movements. The scalar variables are summary statistics of some more complex movements and time from stroke to the recording time. Patients' indices are not included in the date set.
Index

data_generation, 2

fccGen, 4, 6, 8, 9, 12
fccXX, 6, 8
fccXXcv, 7
flars, 8, 11, 12
flars-package, 2
flars_TrainTest, 11

predict.flars, 12

RealDa, 13