Documentation for the R-code to implement the FLRTI methodology in "Functional Linear Regression That's Interpretable"

GARETH M. JAMES* and JI ZHU†

Description

This is a set of functions that fit the FLRTI approach. The main fitting function is "flrti". In addition "flrti.boot" produces bootstrap confidence intervals for $\beta(t)$, "flrti.cv" performs cross-validation to choose the tuning parameters, "flrti.perm" tests statistical significance between Y and $X(t)$ and "predict.flrti" produces predictions for a new set of $X(t)$ predictors. We provide documentation for all these functions below.

Installation

To install the software download the file flrti, open R and type

```
> source(''flrti.txt'')
```

(or what ever name you gave the file) don't forget to give the directory structure in addition if needed. Now if you type ls() you should see

```
> ls()
[1] "flrti"          "flrti.boot"     "flrti.cv"       "flrti.perm"
[5] "linearoptim"    "makeA"          "predict.flrti"  "testdata"
```

"testdata" is a test data set consisting of the 50 curves and associated responses used to generate Figure 1 in the FLRTI paper. The other objects are various functions. Before flrti() can be run we must attach the lpSolve library (you may need to download this from the web by clicking on Packages and Install Packages on the R gui). Try typing the following commands.

```
> library(lpSolve)
> attach(testdata)
> testfit <- flrti(Y,X,sigma=.0002,weight=.1,plot=T)
```

You should see a plot with a similar shape to that of Figure 1b) in the paper. Type

```
> lines(seq(0,1,length=100),beta,col=2)
```

and you will see that the FLRTI estimate (black line) is almost a perfect estimate for the true beta (red line).

Details of all the functions are provided below.

*Marshall School of Business, University of Southern California
†Department of Statistics, University of Michigan

flrti Function

This is the main function that fits the FLRTI algorithm.

Arguments

flrti allows for up to eight inputs. They are:

Y : A vector of n responses. One for each $X(t)$ curve.

X : This is an n by p matrix. The ith row corresponds to the recorded values for $X_i(t)$. It is assumed that all the predictors have been observed over an evenly spaced grid of p time points between 0 and 1.

sigma : This corresponds to the tuning parameter λ in the FLRTI algorithm. In particular $\lambda = \sigma\sqrt{2\log p}$. In general the larger that σ is the larger the feasible region for the linear program. If σ is chosen large enough $\beta(t)$ will simply be estimated as a horizontal line (this is the sparsest solution). As σ is decreased we force a closer fit to the data. If σ is chosen too small an error message such as

```
[1] "Error: Code =  5"
```

will be returned indicating that there is no feasible solution. flrti.cv() can be used to select σ.

deriv : By default flrti searches for a sparse solution in the zeroth and one other derivative. deriv specifies the value for the second derivative. deriv can be any value between 1 and 4. deriv=1 will produce a stepwise constant estimate, deriv=2 gives a piecewise linear solution, deriv=3 gives a piecewise quadratic and deriv=4 a piecewise cubic. If one does not want to place a constraint on the zeroth derivative (i.e. $\beta(t)$ itself) then weight (see below) can be set to zero). By default deriv=2.

weight : The weight to place on the zeroth derivative relative to the higher order derivative chosen by deriv. By default weight=1. A weight of 0 indicates that the zeroth derivative is ignored. I have generally found FLRTI to produce good results for values of weight around 0.1. weight can also be chosen using flrti.cv().

int : Should the model include an intercept term i.e. β_0. By default int=T.

wrap : This enforces a constraint such that $\beta(0) = \beta(1)$ for situations where we know that $\beta(t)$ should wrap around itself. By default wrap=F.

plot : Should the FLRTI estimate for $\beta(t)$ be plotted? By default plot=F.

Value

flrti() returns a list containing thirteen components. Some of these just correspond to the inputs (such as sigma, weight, deriv, int and wrap). The remaining components in the list are:

beta : A vector corresponding to the FLRTI estimate for $\beta(t)$ at each time point that $X(t)$ was observed at.

beta.zero : The estimate for β_0.

gamma : The sparse values corresponding to the various derivatives from equations (17) and (18) in the paper.

lin.fit : The output produced from the linear programming procedure.

Rsq : The value of R^2 for the FLRTI fit.

fitted.values : The estimated responses ($\hat{Y}$) for from the FLRTI fit.

residuals : The residuals i.e. $Y - \hat{Y}$ for each observation.

flrti.boot Function

This function takes as input an object from flrti and uses a bootstrap procedure to produce pointwise confidence intervals on $\beta(t)$. The intervals are plotted along with the estimate for $\beta(t)$.

Arguments

flrti.boot allows up to six inputs. They are:

Y : A vector of *n* responses. One for each $X(t)$ curve.

X : This is an *n* by *p* matrix. The *i*th row corresponds to the recorded values for $X_i(t)$. It is assumed that all the predictors have been observed over an evenly spaced grid of *p* time points between 0 and 1.

obj : A fitted object from running flrti.

B : The number of bootstrap iterations to use. By default B=100.

CI : The confidence range to use. By default CI=95%.

bootfit : An object from a previous run of flrti.boot. If this is supplied then no new calculations are done and the plot is simply produced. One must either supply Y,X and obj or simply bootfit.

Value

In addition to the plot of the confidence intervals, flrti.boot returns a list containing four elements

obj : The flrti object that was supplied to the function.

bootbeta : A matrix with each row corresponding to the bootstrap estimate for $\beta(t)$.

uci : Vector corresponding to the upper confidence interval.

lci : Vector corresponding to the lower confidence interval.

flrti.cv Function

This function uses cross validation to estimate the optimal values for σ and weight. It takes as input a vector of σ values and a vector of weight values and estimates the cross validated error rates for each combination of the two inputs.

Arguments

flrti.cv allows for up to nine inputs. They are:

Y : A vector of n responses. One for each $X(t)$ curve.

X : This is an n by p matrix. The ith row corresponds to the recorded values for $X_i(t)$. It is assumed that all the predictors have been observed over an evenly spaced grid of p time points between 0 and 1.

sigma : This should be a vector of all the possible values for σ that we wish to consider. Note that if σ is chosen too small an error message such as

```
[1] "Error: Code =  5"
```

will be returned indicating that there is no feasible solution. However, this error message does not necessarily mean there is no feasible solution on the original data.

cv : The number of subsets to divide the data into when performing cross validation. By default cv=10.

deriv : Same as for flrti.

weight : This should be a vector of all the possible values for weight.

s : A permuation of the numbers 1 through n indicating how to divide the data up for cross validation. By default s=NULL in which case the data is divided up randomly.

int : Same as for flrti.

wrap : Same as for flrti.

Value

flrti.cv() will print out the optimal CV values for σ and weight. In addition it returns a list with five components. sigma and weight are just the values that were supplied. In addition

error : A matrix with each row corresponding to cross validated error rates for the different values of weight. And each column the error rates for different values of sigma.

allerrors : An three dimensional array with the errors for each division of the data into CV parts.

s : The permutation of the data that was used.

flrti.perm Function

This function uses a permuation test to produce a p-value for the statistical significance of the relationship between $X(t)$ and Y. One supplies a fit from flrti and a p-value, corresponding to the number of permuted R^2's that are greater than the observed R^2, is produced.

Arguments

fclust.perm takes up to six inputs. They are:

Y : A vector of n responses. One for each $X(t)$ curve.

X : This is an n by p matrix. The ith row corresponds to the recorded values for $X_i(t)$. It is assumed that all the predictors have been observed over an evenly spaced grid of p time points between 0 and 1.

obj : A fitted object from running flrti.

B : The number of permutaions of the response Y to use. By default B=100.

permfit : An object from a previous run of flrti.perm. If this is supplied then no new calculations are done and the plot is simply produced. One must either supply Y,X and obj or simpy permfit.

plot : By default the R^2 values for each of the permuted fits is plotted along with the observed R^2. This can be disabled using plot=F.

Value

In addition to printing the p-value and producing a plot, flrti.perm() provides a list with six components.

obj : The original flrti fit.

SSE : The observed sum of squared errors for the true fit.

permSSE : A vector of length B with the SSE's for each of the permutations of Y.

p : The estimated p-value.

Rsq : The observed R^2 for the true fit.

permRsq : A vector of length B with the R^2 for each of the permutations of Y.

predict.flrti Function

This function takes as input an object from flrti and a new set of $X(t)$ predictors and resturns estimates for the response.

Arguments

predict.flrti takes up to two inputs. They are:

obj : A fitted object from running flrti. This is the fit to the training data.

newdata : This is a matrix with one row for each $X_i(t)$ that we wish to make predictions for.

Value

A vector of $\hat{Y}$ for each of the new predictors.