Lab 5, Week 7

Sidi Wu

More on `recpart_fwd()`

We are going to make one more round of modifications to our `recpart_fwd()` before generalizing to the MARS `fwd_stepwise()` function.

1. Let $H(\eta)$ be the step function defined on page 11 of the Friedman paper. Write an R function to implement this. In `recpart_fwd()`, when you split parent basis function B_m into two children use your $H()$ instead of `(x[,v]>t)` and `(x[,v]<=t)`; i.e., make the children with `B[,m]*H(+(x[,v]-t))` and `B[,m]*H(-(x[,v]-t))`.

2. As shown in equation (20) of the MARS paper, the basis functions we build in the forward part of the recursive partitioning algorithm are products of step functions (the H's). The terms in the product are obtained when the algorithm chooses a split in the three inner for-loops. We will record each basis vector as a data frame with rows for step functions and columns corresponding to the (s, v, t) triplets that make up each basis function. Here s is the sign (± 1) in front of `(x[,v]-t)` in the basis function. We will store the basis function data frames in a list called `Bfuncs`. That is, if the mth basis function is $B_m(x) = \prod_{k=1}^{K_m} H[s_{km}(x_{v(k,m}} - t_{km})]$, then `Bfuncs[[m]]` is a data frame with elements

$$\begin{bmatrix} s_{1m} & v(1,m) & t_{1m} \\ s_{2m} & v(2,m) & t_{2m} \\ \cdots & \cdots & \cdots \\ s_{K_m m} & v(K_m,m) & t_{K_m m} \end{bmatrix}$$

   a) Initialize `Bfuncs` to be an empty list of length $M_{max}+1$ inside `recpart_fwd()`.
   b) When the mth basis function `B[,m]` is chosen to split into children `B[,m]*H(+(x[,v]-t))` and `B[,m]*H(-(x[,v]-t))`, make the following modifications to `Bfuncs`: (i) copy the data frame `Bfuncs[[m]]` into `Bfuncs[[M+1]]` and add a row (s,v,t) to `Bfuncs[[M+1]]` with `s=-1` and v and t from the best split, and (ii) add a row (s,v,t) to `Bfuncs[[m]]` where `s=+1` and v and t are from the best split.
   c) Add `Bfuncs` to the return list of `recpart_fwd()`.

3. Test your code as follows.

```
# Test
set.seed(123); n <- 10
x <- data.frame(x1=rnorm(n),x2=rnorm(n))
y <- rnorm(n)
rp_fwd <- recpart_fwd(y,x,Mmax=9)
rp_fwd$Bfuncs
```

Start an R package

In the second part of this lab you will turn your skeleton implementation of MARS into an R package using the tools in the `devtools` package, as outlined in lecture 5. In particular, you will need to

1. call `create_package()` to initialize an R package directory and a new project
    - Remember to specify a directory that is not part of an existing project and not under version control when you call `create_pacakge()`
    - You do not need to call `use_git()` to create a git repository, because you already have one from lab 1. We will copy the files back to the directory you have under version control in step 10 below.
2. copy your R scripts to the R directory of your new package and call `load_all()` to load them into your R session
3. call `devtools::check()` to check that the package builds
4. edit the DESCRIPTION file
5. add a licence
6. start your documentation by inserting an Roxygen skeleton for your main `mars()` function and then fill in your title, function arguments, return value, an example and any imports you require
7. call `document()` to generate an .Rd file and update NAMESPACE
8. call `use_package()` to add any package dependencies/imports to DESCRIPTION
9. call `devtools::check()` again to make sure the package will still build
10. copy your R package files (copying the entire `mars` directory is probably easiest) to the local directory for your Stat 360 project, commit the new files and then push them to the Rpackges directory in the SFUStat360Projects repository (https://github.com/sidiwu/SFUStat360Projects).