Lab 7, Week 10

Sidi Wu

In this lab we'll work through some of the changes needed to turn `recpart_fwd()` into `fwd_stepwise()` and test the `fwd_stepwise()` function.

1. **Setup:** Make sure you have recently pulled from the class repository. Look for the folder "Testfiles" within "Lab7". You should see the files `mctest.RData`, `xtest.Rdata` and `fwdtest.RData` in there. Put these files in your working directory and use `load()` to load their contents, `mctest`, `xtest` and `fwdtest`, into your R session. Next, execute the following lines

```
set.seed(123); n <- 10
data <- data.frame(x1=rnorm(n),x2=rnorm(n),y=rnorm(n))
formula <- formula(y ~.)
```

2. **`mars.control` objects:** Follow the instructions in the `Project/mars5.pdf` lecture to create constructor, validator and helper functions for the `mars.control` class. You can refer to the example given in Lecture07 (p.24). Use your helper with no arguments, so that it uses all defaults, to create a `mars.control` object named `mc` and use `all.equal(mc,mctest)` to check that it is the same as `mctest`.

```
# mc <- mars.control()
# all.equal(mc,mctest)
```

3. **model matrix:** The `Project/mars4.pdf` lecture (p.7) gave a code snippet for the main `mars()` function that uses `model.matrix()` to extract the matrix of predictors from the input formula and data frame. Step through the code snippet to create the matrix `x` of predictors. Use `head()` to print the first few rows. The first column is a column of 1's for the intercept and is not a predictor. Remove this column from your `x` and use `all.equal(x,xtest)` to test that the modified `x` is the same as `xtest`.

4. **Updating `B` and `Bfuncs`:** As indicated in the `Project/mars5.pdf` lecture, the list `Bfuncs` should be of length `Mmax+1`, because now `Mmax` is the maximum number of **non-intercept** basis functions. Recall that element `m` of `Bfuncs` is a data frame with columns `s`, `v` and `t`, and as many rows as there are hinge functions that make up B_m. The first element of `Bfuncs` is for the constant (intercept) basis function that is not made of **any** hinge functions, so we will leave it empty.

    i. Initialize `B` with your `init_B()` function and `Bfuncs` to be an empty list of length `mc$Mmax+1`.
    ii. Change the way `B` and `Bfuncs` are updated within the loops over m, v and t, so that we add **pairs** of basis functions.

5. **Changes to loops:** In `recpart_fwd()` there are four nested loops, over (i) the number of basis functions to construct (M loop), (ii) the basis function to split (m loop), (iii) the variable to split on (v loop) and (iv) the split point (t loop).

    i. Replace the M loop with a loop over pairs i, and set the value of M from the value of i.

ii. In the v loop, `recpart_fwd()` was allowed to split on any of the `n` explanatory variables, but `fwd_stepwise()` is restricted to split on variables that are not already in basis function m. For a given m, write the `for()` statement that loops over variables v not already in `Bfuncs[[m]]`. Hints: What is in `Bfuncs[[m]]["v"]`? How might the `setdiff()` function be useful?

6. Write a first draft of `fwd_stepwise()` that returns the list `list(y=y,B=B,Bfuncs=Bfuncs)`. Before you return, set the colnames of `B` with `colnames(B) <- paste0("B",(0:(ncol(B)-1)))`. Use the `y`, `x`, and `mc` from questions 2 and 3 as input to your `fwd_stepwise()` function and save the output as `fwd`. Check that your `fwd` is the same as `fwdtest` with `all.equal(fwd,fwdtest)`.

```
# fwd <- fwd_stepwise(y,x,mc)
# all.equal(fwd,fwdtest)
```