Week 5 exercises

Brad McNeney

Recursive partitioning by forward selection

- In this exercise you will write a version of recursive partitioning by forward selection following Algorithm 1 of Friedman (1991), with a few notational changes that will make the transition to the MARS forward selection algorithm easier.
- The following code gives the framework for the function `recpart_fwd()`. You are asked to write several of the support functions described below the code chunk.
- You need to understand `recpart_fwd()` so that you can modify it to the MARS forward stepwise algorithm.

```
recpart_fwd <- function(y,x,Mmax){
  #-----------------------------------------------------
  # Error checking:
  #-----------------------------------------------------
  # Initialize:
  N <- length(y) # sample size
  n <- ncol(x) # number of predictors
  B <- init_B(N,Mmax) # Exercise: write init_B()
  splits <- data.frame(m=rep(NA,Mmax),v=rep(NA,Mmax),t=rep(NA,Mmax))
  #-----------------------------------------------------
  # Looping for forward selection:
  for(M in 1:Mmax) { # contrast to indexing 2...Mmax in Friedman
    lof_best <- Inf
    for(m in 1:M) { # choose a basis function to split
      for(v in 1:n){ # select a variable to split on
        tt <- split_points(x[,v],B[,m]) # Exercise: write split_points()
        for(t in tt) {
          Bnew <- data.frame(B[,(1:M)[-m]],
                       Btem1=B[,m]*(x[,v]>t),Btem2=B[,m]*(x[,v]<=t))
          gdat <- data.frame(y=y,Bnew)
          lof <- LOF(y~.,gdat) #  Use your LOF() from week 4
          if(lof < lof_best) {
            lof_best <- lof
            splits[M,] <- c(m,v,t)
          } # end if
        } # end loop over splits
      } # end loop over variables
    } # end loop over basis functions to split
    m <- splits[M,1]; v <- splits[M,2]; t <- splits[M,3]
    B[,M+1] <- B[,m]*(x[,v]<=t)
    B[,m] <- B[,m]*(x[,v]>t)
  } # end loop over M
```

```
  return(list(B=B,splits=splits))
}
LOF <- function(form,data) {
  ff <- lm(form,data)
  return(sum(residuals(ff)^2))
}
#-------------------------------------
```

Exercises:

1. Write a code snippet that checks whether the input `Mmax` is greater than or equal to 2. If not, throw a warning and set Mmax to 2.
2. Write the function `init_B()`, which takes the sample size `N` and max number of basis functions `Mmax` as input and returns a data frame with 1's in the first column, NAs in all others, and column names B0,B1,...,BMmax.
3. Write `split_points()`. This one is different from the one you wrote in the week 4 exercises. Here we take predictor variable x_v, and basis function B_m as input. The eligible split points are x_v values such that basis function $B_m(x) > 0$. Return ordered unique values, and remember to exclude the largest possible value.
4. Test `recpart_fwd(y,x,Mmax=2)`, with your functions from the exercises, runs on the following test data. These are the same as in the week 4 exercises. See if you can find these first two splits on the drawing from week 4:

image

```
set.seed(123); n <- 10
x <- data.frame(x1=rnorm(n),x2=rnorm(n))
y <- rnorm(n)
# rp <- recpart_fwd(y,x,Mmax=2)
```

- You will create tests of your implementation in lab 4.