Lab 2 Block 2

Emil K Svensson

8 December 2016

```
library(tree)
library(mboost)
library(randomForest)
library(ggplot2)
```

Part A

Assignment 2

2.1

```
bfr.SE <- 0
set.seed(1234567890)
for (i in 1:100) {
  samptrain<-train[sample(nrow(train),replace = TRUE),]
  bfr.tree        <- tree(Bodyfat_percent ~. ,data = samptrain)
  bfr.predictions <- predict(bfr.tree,test)
  bfr.SE[i]       <- mean((bfr.predictions - test$Bodyfat_percent)^2)
}
mean(bfr.SE)
```

```
## [1] 37.10301
```

The upper bound for the MSE is 37.10301

2.2

```
bfr.SE2 <- c()
set.seed(1234567890)

bfr.tree22 <- tree(Bodyfat_percent ~. ,data = BFR)
bfr.cv <- cv.tree(bfr.tree22, K = 3)
best.size <- bfr.cv$size[which.min(bfr.cv$dev)]

for (i in 1:100){
  BFRre<- BFR[sample(nrow(BFR),replace = TRUE),]
  bfr.tree22 <- tree(Bodyfat_percent ~. ,data = BFRre )
  bfr.tree22 <- prune.tree(bfr.tree22, best = best.size)
  bfr.SE2[i] <- mean( (predict(bfr.tree22, newdata = BFR) - BFR$Bodyfat_percent)^2)
}
```

```
## Warning in prune.tree(bfr.tree22, best = best.size): best is bigger than
## tree size
```

```r
mean(bfr.SE2)
```

```
## [1] 29.1264
```

2.3

For the 2.1 the trees to the user would look like this.

```r
for (i in 1:100) {
  samptrain<-train[sample(nrow(train),replace = TRUE),]
  bfr.tree        <- tree(Bodyfat_percent ~. ,data = samptrain)
  bfr.predictions <- predict(bfr.tree,test)
  bfr.SE[i]       <- mean((bfr.predictions - test$Bodyfat_percent)^2)
}
```

In this case the trees are fitted with all data instead of just the traning data.

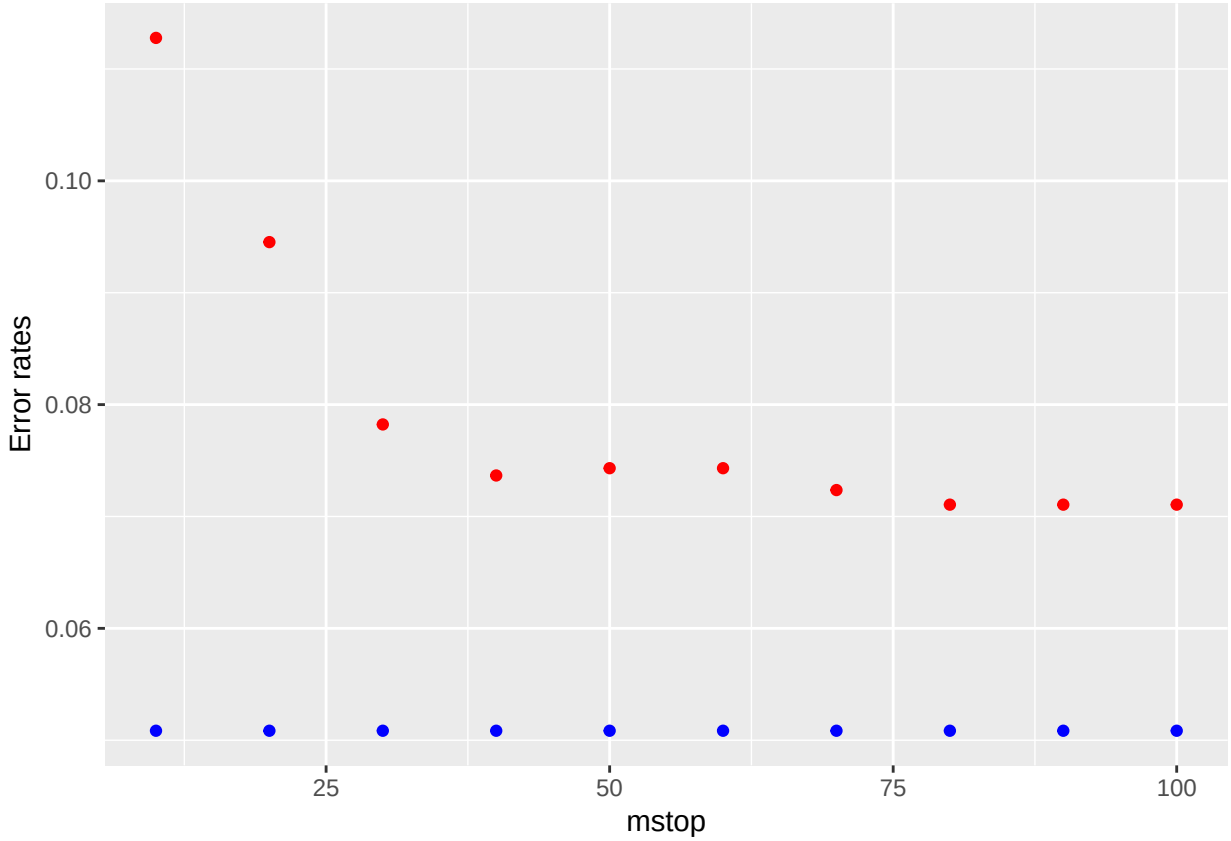
For the 2.3 Cross Validation it's this case

Here there is something else or the same, who knows.

Assignment 4

```r
ada.trees<- sapply(X = seq(10,100,10), FUN = function(y) {
  set.seed(1234567890)
  adaTree <- blackboost(Spam~., data = spamTrain, family = AdaExp(), control = boost_control(mstop = y))
  predvals <- predict(adaTree, newdata = spamTest, type = "class")
  return(table(Predicted = predvals, Observed = spamTest$Spam))

})


ten.trees<- sapply(X = seq(10,100,10), FUN = function(y){
  set.seed(1234567890)
wierdTree <- randomForest(formula = Spam ~., data = spamTrain,  control = boost_control(mstop = y))
RFpredvals <- predict(wierdTree, newdata = spamTest, type = "class")
table(Predicted = RFpredvals, Observed = spamTest$Spam)
})
```

```r
mcrplot <- data.frame(mstop = seq(10,100,10))
mcrplot$ada.trees <- 1 - colSums(ada.trees[c(1,4),])/colSums(ada.trees)


mcrplot$ten.trees <- 1- colSums(ten.trees[c(1,4),])/colSums(ten.trees)

ggplot(data = mcrplot) +
  geom_point( aes(x = mstop, y=ada.trees), col = "red") +
  geom_point( aes(x = mstop, y=ten.trees), col = "blue") + labs(y = "Error rates")
```



The missclassification rate for the randomforest-trees, represented by the blue dots is stable even for low number of trees compared to the Adaboost classification trees (represented by the red dots) that have high error rates for low number of trees.

Part B

```
set.seed(1234567890)
max_it <- 100 # max number of EM iterations
min_change <- 0.1 # min change in log likelihood between two consecutive EM iterations
N=1000 # number of training points
D=10 # number of dimensions
x <- matrix(nrow = N, ncol = D) # training data
true_pi <- vector(length = 3) # true mixing coefficients
true_mu <- matrix(nrow = 3, ncol = D) # true conditional distributions
true_pi <- c(1/3, 1/3, 1/3)
true_mu[1,] <- c(0.5,0.6,0.4,0.7,0.3,0.8,0.2,0.9,0.1,1)
true_mu[2,] <- c(0.5,0.4,0.6,0.3,0.7,0.2,0.8,0.1,0.9,0)
true_mu[3,] <- c(0.5,0.5,0.5,0.5,0.5,0.5,0.5,0.5,0.5,0.5)
#plot(true_mu[1,], type = "o", col = "blue", ylim = c(0, 1))
#points(true_mu[2,], type="o", col="red")
#points(true_mu[3,], type="o", col="green")


# Producing the training data
for(n in 1:N) {
```

```
  k <- sample(1:3,1,prob=true_pi)
  for(d in 1:D) {
    x[n,d] <- rbinom(1,1,true_mu[k,d])
  }
}

K <- 3 # number of guessed components
z <- matrix(nrow=N, ncol=K) # fractional component assignments
pi <- vector(length = K) # mixing coefficients
mu <- matrix(nrow=K, ncol=D) # conditional distributions
llik <- vector(length = max_it) # log likelihood of the EM iterations


# Initialization of the paramters (in a random manner)
pi <- runif(K,0.49,0.51)
pi <- pi / sum(pi)


for(k in 1:K) {
  mu[k,] <- runif(D,0.49,0.51)
}



for(it in 1:max_it) {
  #plot(mu[1,], type="o", col="blue", ylim=c(0, 1))
  #points(mu[2,], type="o", col="red")
  #points(mu[3,], type="o", col="green")
  #points(mu[4,], type="o", col="yellow")
  #Sys.sleep(0.5)

  # E-step: Computation of the fractional component assignments

  for (i in 1:nrow(x)){
    for (j in 1:nrow(mu)){

    z[i,j]<-prod(mu[j,]^(x[i,])*(1-mu[j,])^(1-x[i,])) * pi[j]
    }
  }
  for (l in 1:nrow(z)){
    z[l,]<- z[l,]/sum(z[l,])

  }

    part2<- c()
  tempvar <- c()
  #Log likelihood computation.
  for (rad in 1:nrow(x)){
    for (klass in 1:nrow(mu)){
  part1 <- x[rad, ] * log( mu[klass, ] ) + (1 - x[rad, ])*log(1 - mu[klass, ])
  part2[klass]<-z[rad, klass]*(log(pi[klass]) + sum(part1))
    }
    tempvar[rad] <- sum(part2)
```

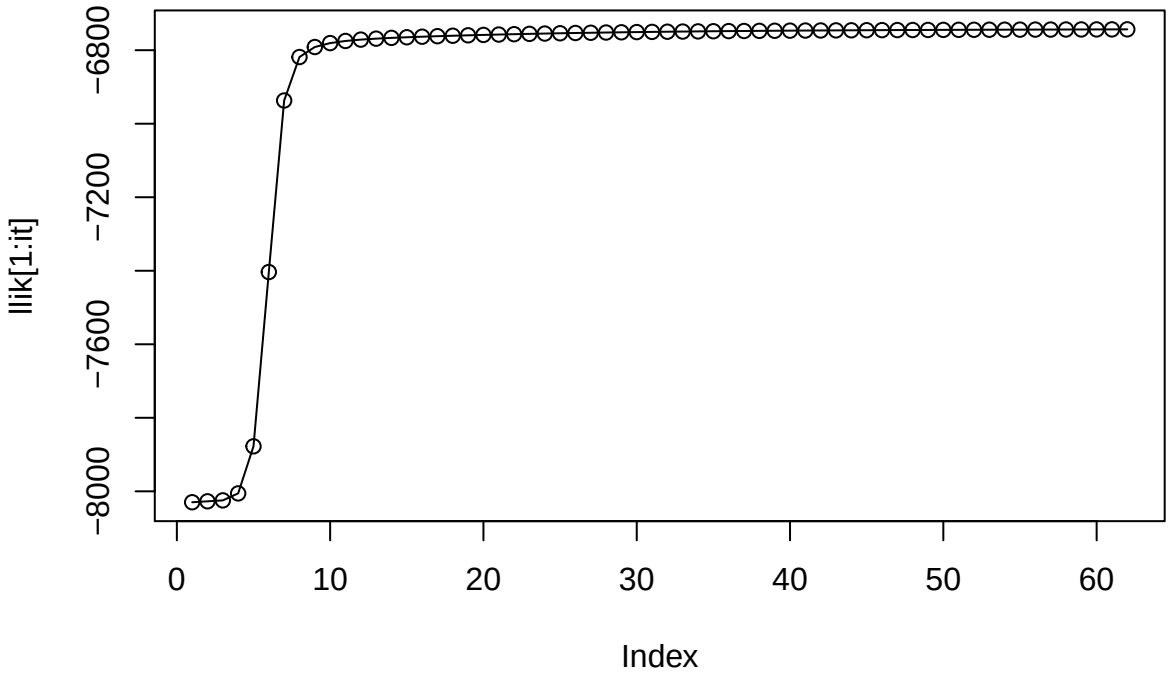
```
  }

  llik[it] <- sum(tempvar)
  #cat("iteration: ", it, "log likelihood: ", llik[it], "\n")
  #flush.console()
  # Stop if the lok likelihood has not changed significantly
  if (it >1){
    if(abs(abs(llik[it]) - abs(llik[it-1])) < min_change){
      returning<- "The log-likelihood as not change significantly, returning from loop"
  break
    }
  }
  #M-step: ML parameter estimation from the data and fractional component assignments
  pi <- colSums(z) / 1000 # pi_k-ML

  for (class in 1:nrow(mu)){
    for (column in 1:ncol(mu)){
      mu[class,column] <- sum( z[,class]*x[,column] )/sum( z[,class] )
    }
  }

}

pi
```

```
## [1] 0.3259592 0.3044579 0.3695828
```

```
mu
```

```
##           [,1]      [,2]      [,3]      [,4]      [,5]      [,6]      [,7]
## [1,] 0.4737193 0.3817120 0.6288021 0.3086143 0.6943731 0.1980896 0.7879447
## [2,] 0.4909874 0.4793213 0.4691560 0.4791793 0.5329895 0.4928830 0.4643990
## [3,] 0.5089571 0.5834802 0.4199272 0.7157107 0.2905703 0.7667258 0.2320784
##           [,8]      [,9]      [,10]
## [1,] 0.1349651 0.8912534 0.01937869
## [2,] 0.4902682 0.4922194 0.39798407
## [3,] 0.8516111 0.1072226 0.99981353
```

```
plot(llik[1:it], type="o")
```


llik[1:it]
−8000
−7600
−7200
−6800
0
10
20
30
40
50
60
Index