# Lab 2 Block 2

*Emil K Svensson*

*9 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 estimated to 37.10301

### 2.2

```
set.seed(1234567890)
BFR$index<-c(rep(1,36),rep(2,37),rep(3,37))
#BFR$index<-sample(c(rep(1,36),rep(2,37),rep(3,37)))
bfr.SE2<-matrix(nrow=100,ncol=3)
iter <- c()
#subset(BFR,index != 1)[,c(-4)]

for (set in 1:3){
  BFRa<-subset(BFR,BFR$index != set)[,c(-4)]
  BFRpred<-subset(BFR,BFR$index == set)[,c(-4)]

  print(c(nrow(BFRa),nrow(BFRpred),nrow(BFRa)+nrow(BFRpred)))
  for (i in 1:100){


    BFRre<- BFRa[sample(1:nrow(BFRa),replace = TRUE),]
```

```
    bfr.tree22 <- tree(Bodyfat_percent ~. ,data = BFRre, split = "deviance" )
    bfr.SE2[i,set] <- mean( (predict(bfr.tree22, newdata = BFRpred) - BFRpred$Bodyfat_percent)^2)


  }
}
```

```
## [1]  74  36 110
## [1]  73  37 110
## [1]  73  37 110
```

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

```
## [1] 41.1672
```

The MSE estimated with Cross Validation-bagging was estimated to around 41, higher than the previous bagging estimation.

### 2.3

For both cases the trees returned to the user would look like this.

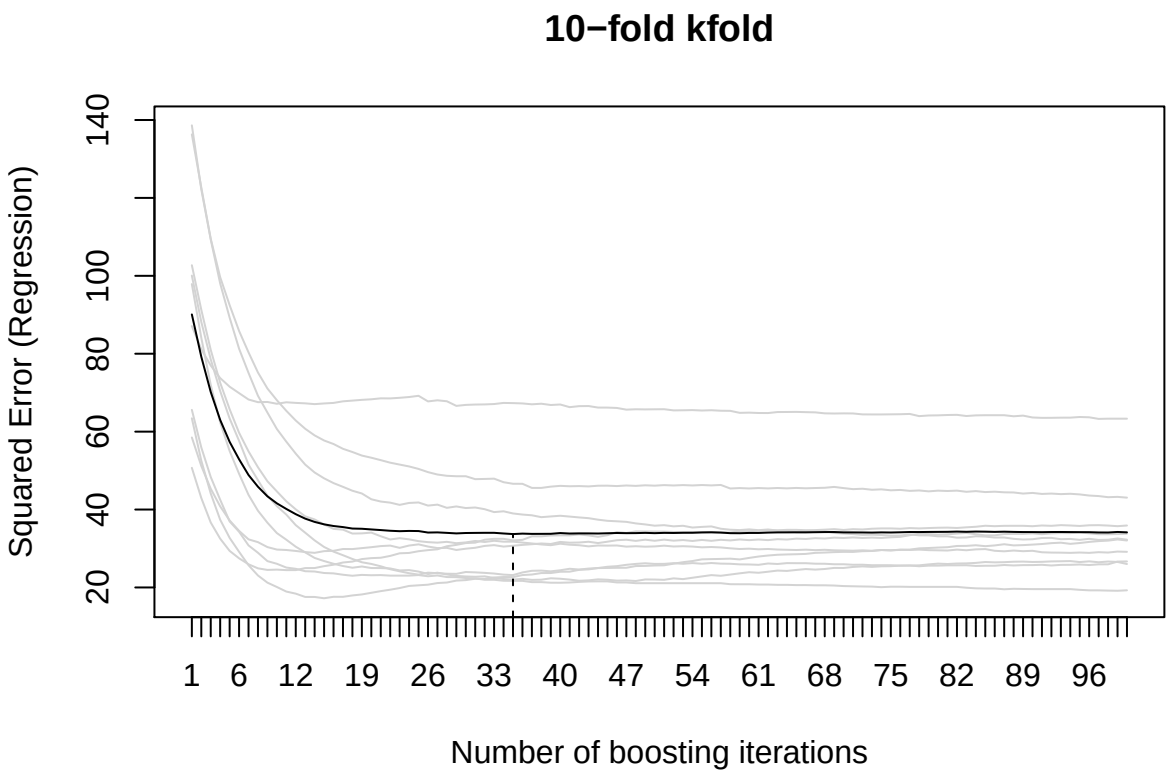
```
trees <- list()

for (i in 1:100) {
  samptrain       <- BFR[sample(nrow(BFR),replace = TRUE),]
  bfr.tree        <- tree(Bodyfat_percent ~. ,data = samptrain)
  trees[[i]]      <- bfr.tree
}
```

Here the bagging-estimation of the regression tree is trained on the whole dataset istead of just two thirds of the data.

## Assignment 3

### 3.1

```
BFR <- read.csv2("bodyfatregression.csv")
set.seed(1234567890)
m <- blackboost(Bodyfat_percent ~ Waist_cm + Weight_kg, data = BFR)
cvf <- cv(model.weights(m), type = "kfold")
cvm <- cvrisk(m, folds = cvf, grid = 1:100)
plot(cvm)
```




The optimal number of boosting itterations are around 35, its hard to tell exactly with such messy ticks at the x-axis. The gray lines represent the different trees squared error that are weighted together as the black line in the boosting method.

### 3.2

```
## 3.2
set.seed(1234567890)
m2 <- blackboost(Bodyfat_percent ~ Waist_cm + Weight_kg, data = train,
                 control=boost_control(mstop=mstop(cvm)))

cvf2 <- cv(model.weights(m2), type = "kfold")
cvm2 <- cvrisk(m2, folds = cvf2, grid = 1:100)

m2.train <- sum( (predict(m2,train) - train$Bodyfat_percent)^2)
m2.test <- sum( (predict(m2,test) - test$Bodyfat_percent)^2)
cat("SSE for traning:",m2.train,"\n SSE for test:",m2.test)
```

```
## SSE for traning: 1530.302
##  SSE for test: 954.3946
```

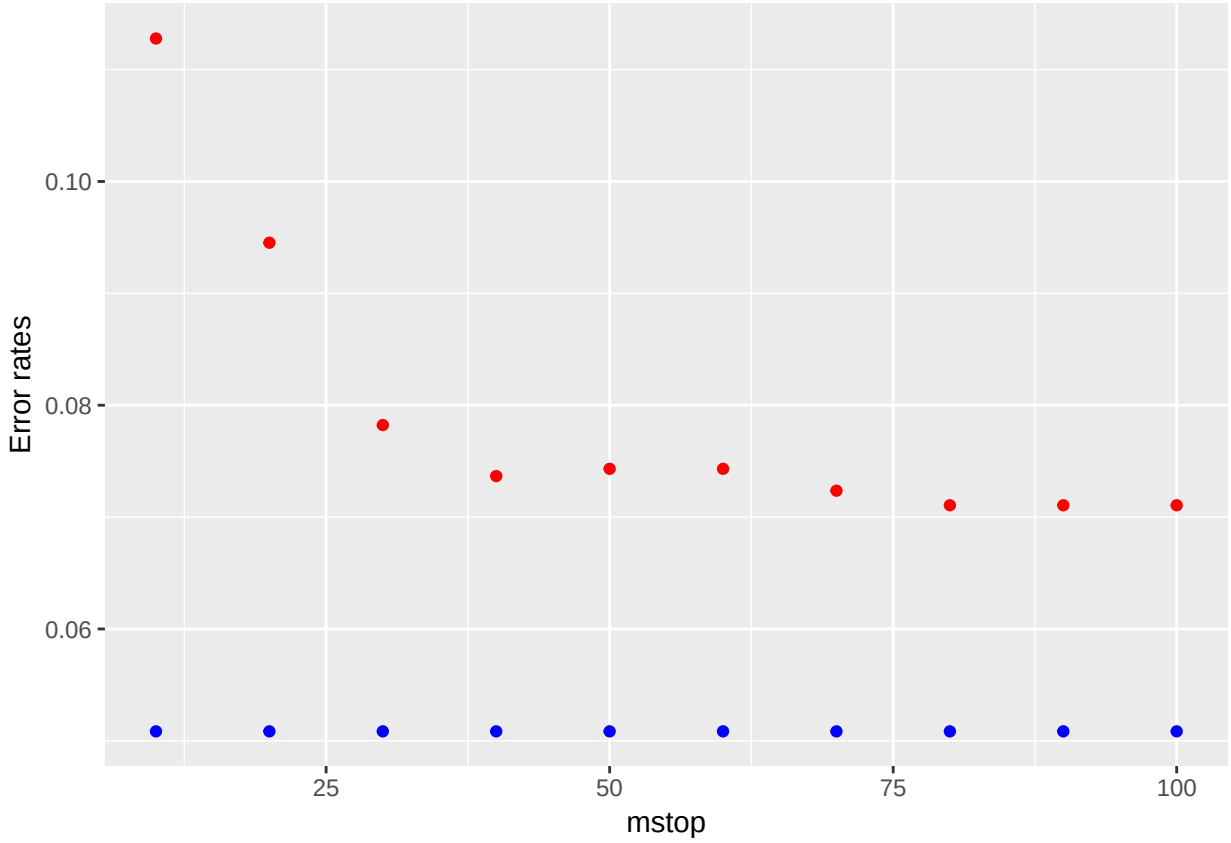
## Assignment 4

```
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)
})
```

Above is the code for the two different estimations, the Ada-tree has the family specified as AdaExp.

```
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 test missclassification rate for the randomforest-trees are 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

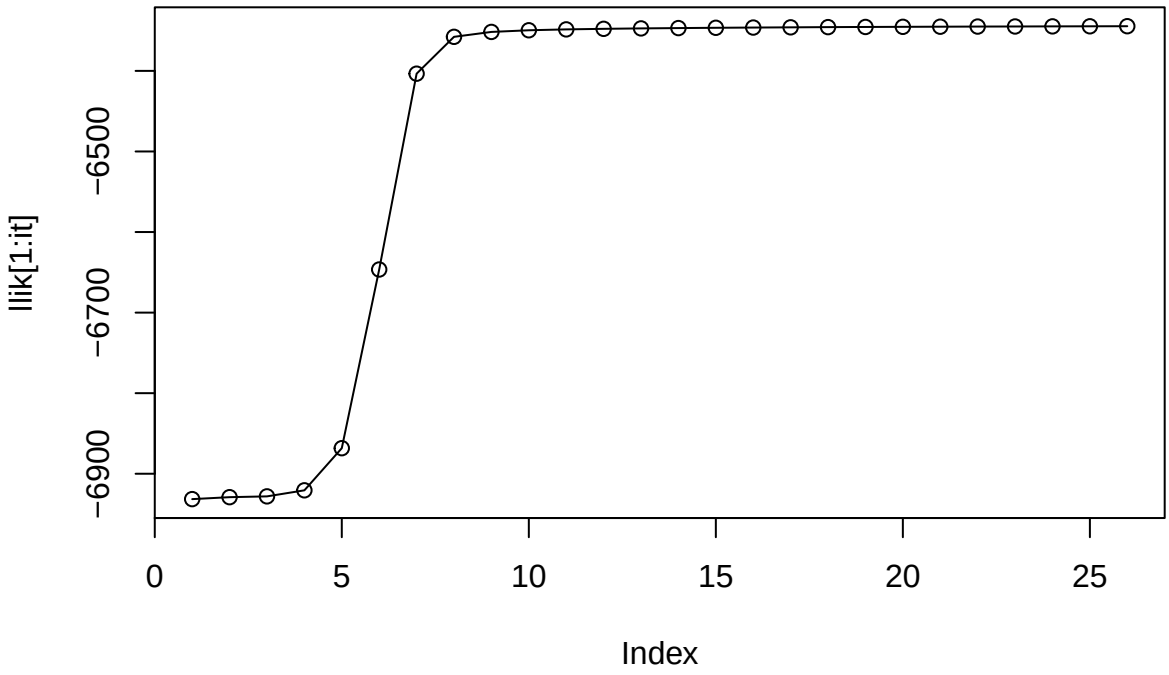
```
set.seed(1234567890)
max_it <- 100 # max number of EM iterations
min_change <- 0.1 # min change in log likelihood between two consecutive EM iterations
N=1000 # number of training points
D=10 # number of dimensions
x <- matrix(nrow = N, ncol = D) # training data
true_pi <- vector(length = 3) # true 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")
  # E-step: Computation of the fractional component assignments
```

```
bern<-matrix(nrow =N, ncol = K)

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

    bern[i,j]<-prod(mu[j,]^(x[i,])*(1-mu[j,])^(1-x[i,]))

  }
}



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

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

}

part<- c()
tempvar <- c()

#Log likelihood computation.
 for (rad in 1:nrow(x)){
   for (klass in 1:nrow(mu)){
            part[klass]<- (pi[klass]*bern[rad,klass])
        }
        tempvar[rad] <- log(sum(part))
      }

llik[it] <- sum(tempvar)
#cat("iteration: ", it, "log likelihood: ", llik[it], "\n")

# Stop if the lok likelihood has not changed significantly
if (it >1){
  if(abs(abs(llik[it]) - abs(llik[it-1])) < min_change){
return("The log-likelihood as not change significantly, returning from loop")
  }
}

#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] )
  }
}
```

```
}
```

```
## [1] "The log-likelihood as not change significantly, returning from loop"
```

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




A gif for the mu parameters through the 26 itterations can be found here.

http://github.com/Emil5KS/732A95/blob/master/B2lab2/BallaBalla.gif?raw=true

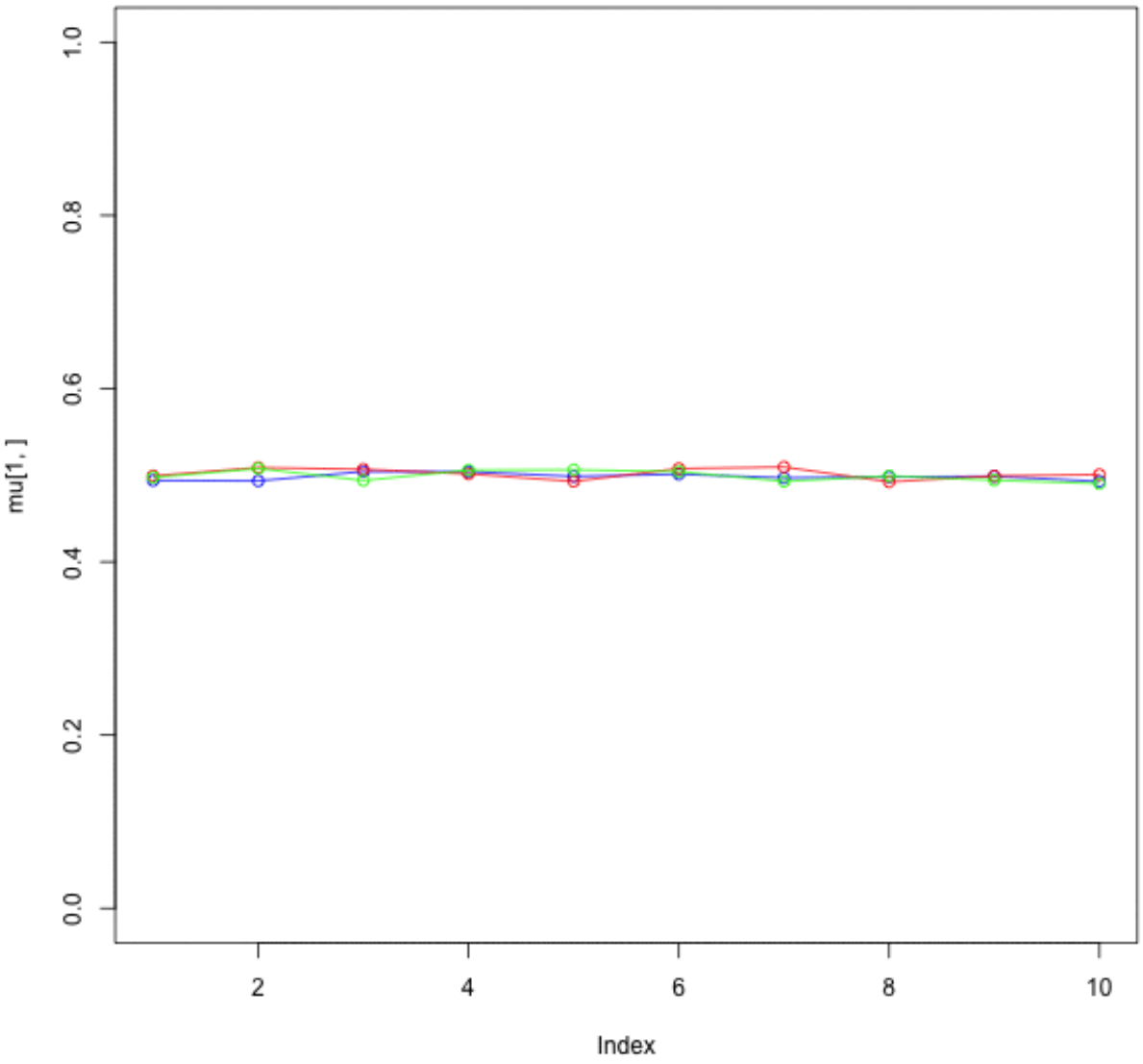



Figure 1:

## Code

```
knitr::opts_chunk$set(echo = TRUE)
library(tree)
library(mboost)
library(randomForest)
library(ggplot2)

BFR <- read.csv2("bodyfatregression.csv")
set.seed(1234567890)
BFR <- BFR[sample(nrow(BFR), replace = FALSE),]

train <- BFR[1:floor((nrow(BFR)*(2/3))),]
test <- BFR[74:nrow(BFR),]

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)

set.seed(1234567890)
BFR$index<-c(rep(1,36),rep(2,37),rep(3,37))
#BFR$index<-sample(c(rep(1,36),rep(2,37),rep(3,37)))
bfr.SE2<-matrix(nrow=100,ncol=3)
iter <- c()
#subset(BFR,index != 1)[,c(-4)]

for (set in 1:3){
  BFRa<-subset(BFR,BFR$index != set)[,c(-4)]
  BFRpred<-subset(BFR,BFR$index == set)[,c(-4)]

  print(c(nrow(BFRa),nrow(BFRpred),nrow(BFRa)+nrow(BFRpred)))
  for (i in 1:100){


    BFRre<- BFRa[sample(1:nrow(BFRa),replace = TRUE),]
    bfr.tree22 <- tree(Bodyfat_percent ~. ,data = BFRre, split = "deviance" )
    bfr.SE2[i,set] <- mean( (predict(bfr.tree22, newdata = BFRpred) - BFRpred$Bodyfat_percent)^2)


  }
}

mean(bfr.SE2)

trees <- list()

for (i in 1:100) {
```

```
  samptrain       <- BFR[sample(nrow(BFR),replace = TRUE),]
  bfr.tree        <- tree(Bodyfat_percent ~. ,data = samptrain)
  trees[[i]]      <- bfr.tree
}

BFR <- read.csv2("bodyfatregression.csv")
set.seed(1234567890)
m <- blackboost(Bodyfat_percent ~ Waist_cm + Weight_kg, data = BFR)
cvf <- cv(model.weights(m), type = "kfold")
cvm <- cvrisk(m, folds = cvf, grid = 1:100)
plot(cvm)

## 3.2
set.seed(1234567890)
m2 <- blackboost(Bodyfat_percent ~ Waist_cm + Weight_kg, data = train,
                 control=boost_control(mstop=mstop(cvm)))

cvf2 <- cv(model.weights(m2), type = "kfold")
cvm2 <- cvrisk(m2, folds = cvf2, grid = 1:100)

m2.train <- sum( (predict(m2,train) - train$Bodyfat_percent)^2)
m2.test <- sum( (predict(m2,test) - test$Bodyfat_percent)^2)
cat("SSE for traning:",m2.train,"\n SSE for test:",m2.test)

spam <- read.csv2("spambase.csv")
spam$Spam <- factor(spam$Spam)
set.seed(1234567890)
spam <- spam[sample(nrow(spam)),]
spamTrain <- spam[1:(nrow(spam)*2/3),]
spamTest <- spam[3068:4601,]


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)
})


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

```r
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")
```

```r
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")
  # E-step: Computation of the fractional component assignments


  bern<-matrix(nrow =N, ncol = K)

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

      bern[i,j]<-prod(mu[j,]^(x[i,])*(1-mu[j,])^(1-x[i,]))

    }
  }



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

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

  }

  part<- c()
  tempvar <- c()

  #Log likelihood computation.
   for (rad in 1:nrow(x)){
     for (klass in 1:nrow(mu)){
               part[klass]<- (pi[klass]*bern[rad,klass])
          }
          tempvar[rad] <- log(sum(part))
        }

  llik[it] <- sum(tempvar)
  #cat("iteration: ", it, "log likelihood: ", llik[it], "\n")

  # Stop if the lok likelihood has not changed significantly
  if (it >1){
    if(abs(abs(llik[it]) - abs(llik[it-1])) < min_change){
  return("The log-likelihood as not change significantly, returning from loop")
    }
  }

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