lab1 block 2 group 5 report

Bjorn Hansen, Erik Anders, Ahmed Alhasan

12/4/2019

1. ENSEMBLE METHODS

To start, the data is loaded into test and train sets with 2/3 of the data being used for training and 1/3 used for testing.

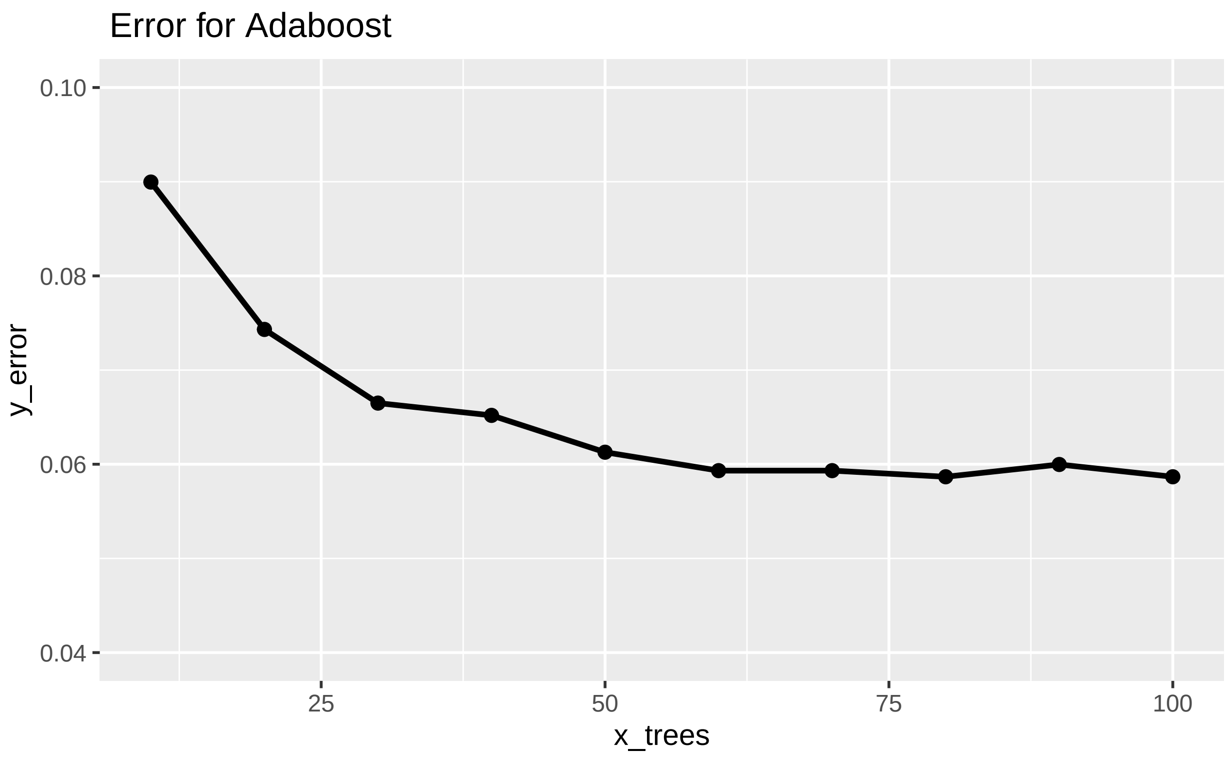
Adaboost

Adaboost classification trees are used to train the model. Inside the boost control argument a value of 0.5 for "nu" was used. Adaboost classify the emails through majority voting by setting type="class". As can be seen from the confusion matrix below (using 100 trees) the diagonal values are quite high meaning that the ada model is predicting with high accuaracy.

```
##    Y hat
## Y     0   1
##   0 882  40
##   1  51 561
```

```
## [1] "Error for ada model: 0.059322033898305"
```

Below is a plot of the recorded error for 10 up to 100 trees.

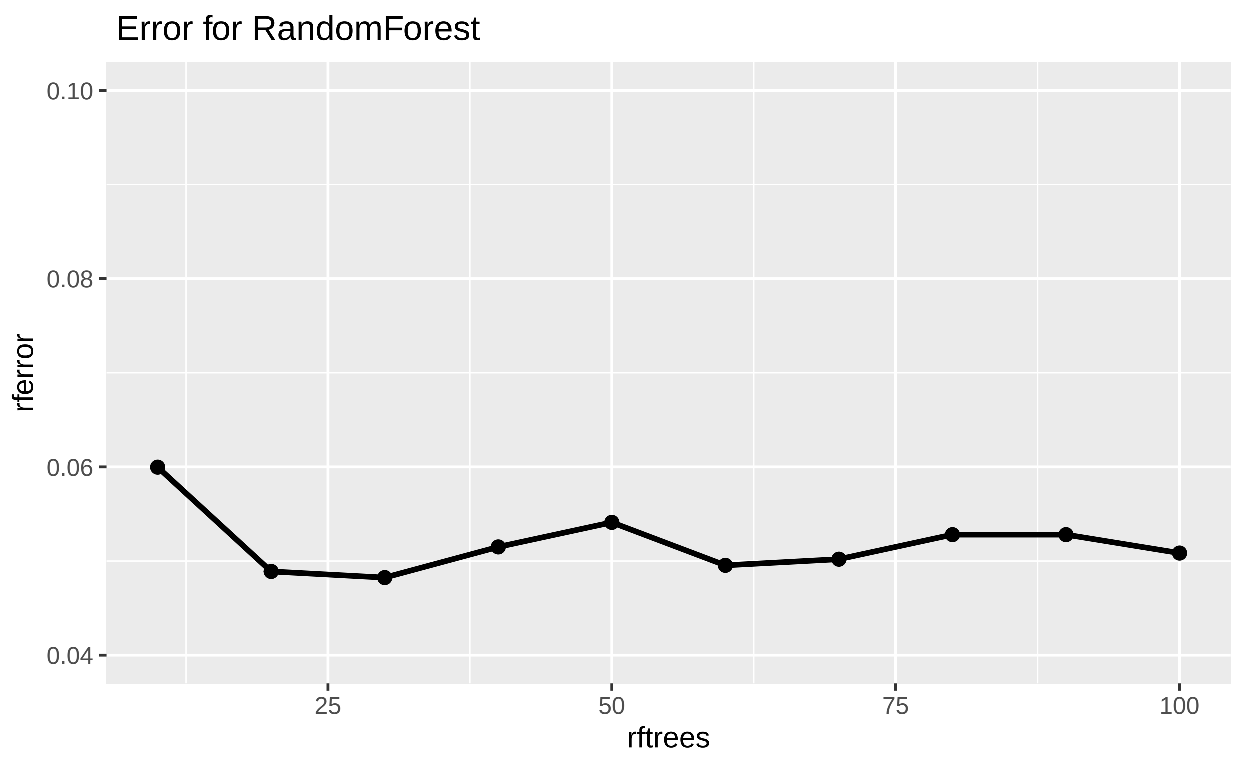


Random Forest

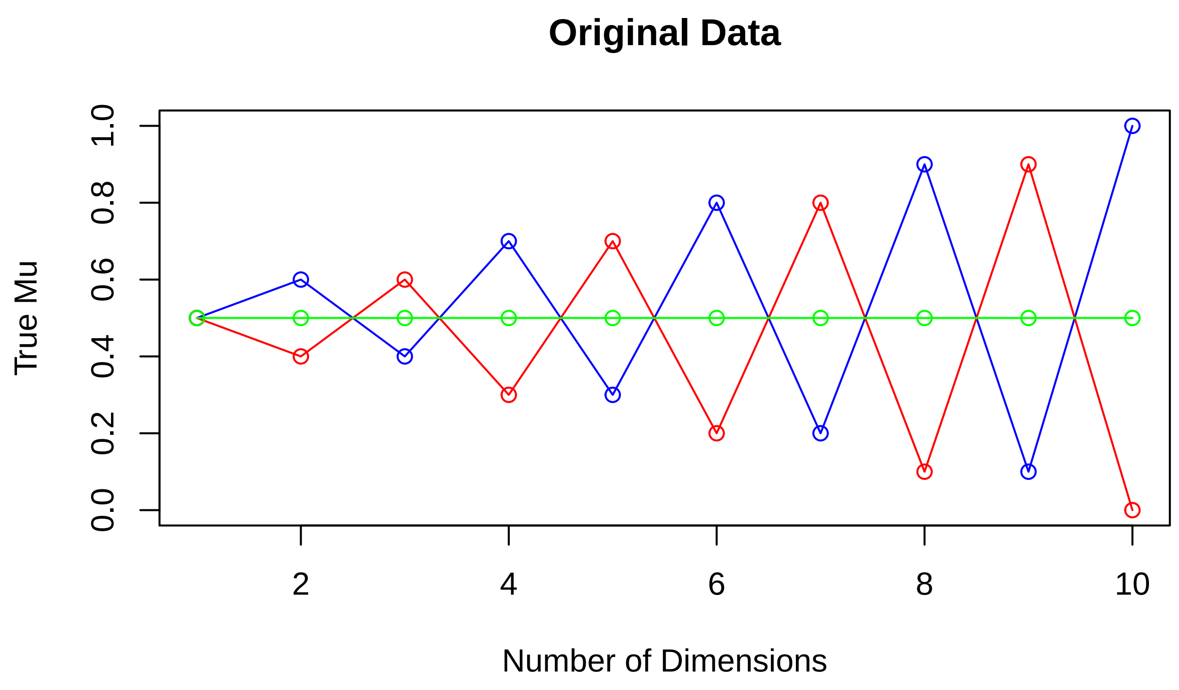
The random forest algorithm is used to classify the mails as spam and non-spam. As can be seen from the confusion matrix below the range of error didn't change widely (the error rate was around 5%) between 20 to 100 trees, the diagonal values are very high meaning that the randomForest model is predicting with high accuaracy.

Below is a plot of the recorded error for 10 up to 100 trees.

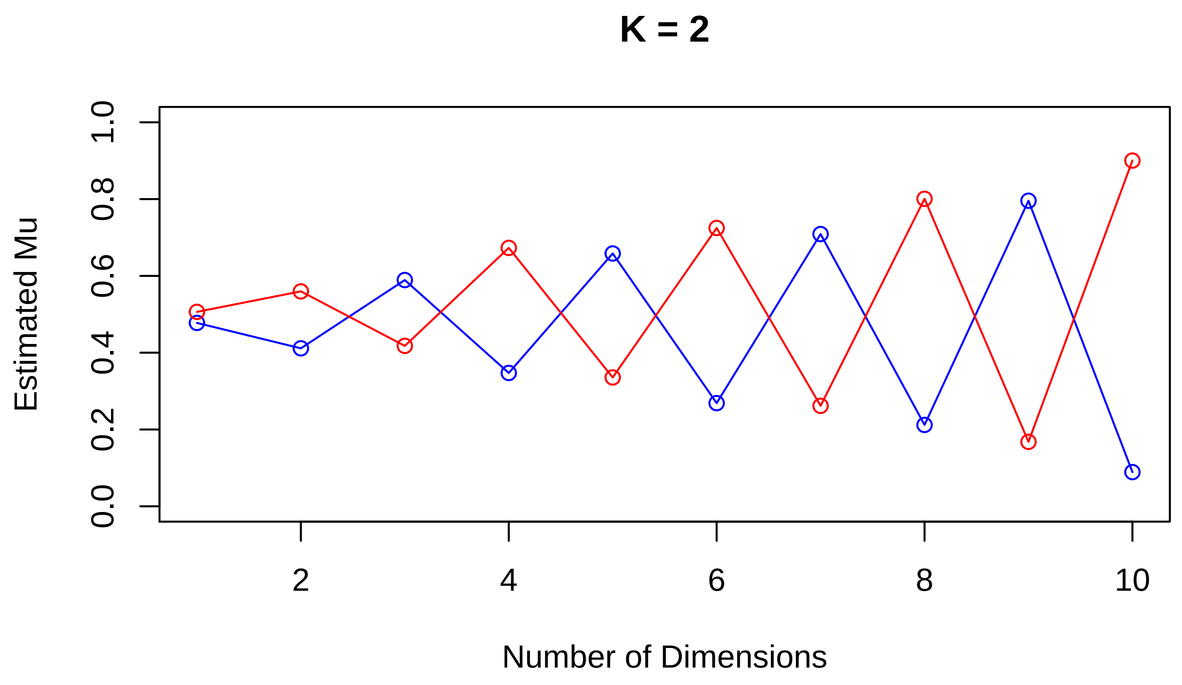
```
## $`Missclassification Rate`
## [1] 0.0482399
##
## $`Confusion Matrix`
##          Actual
## Predicted   0   1
##         0 890  46
##         1  32 566
```

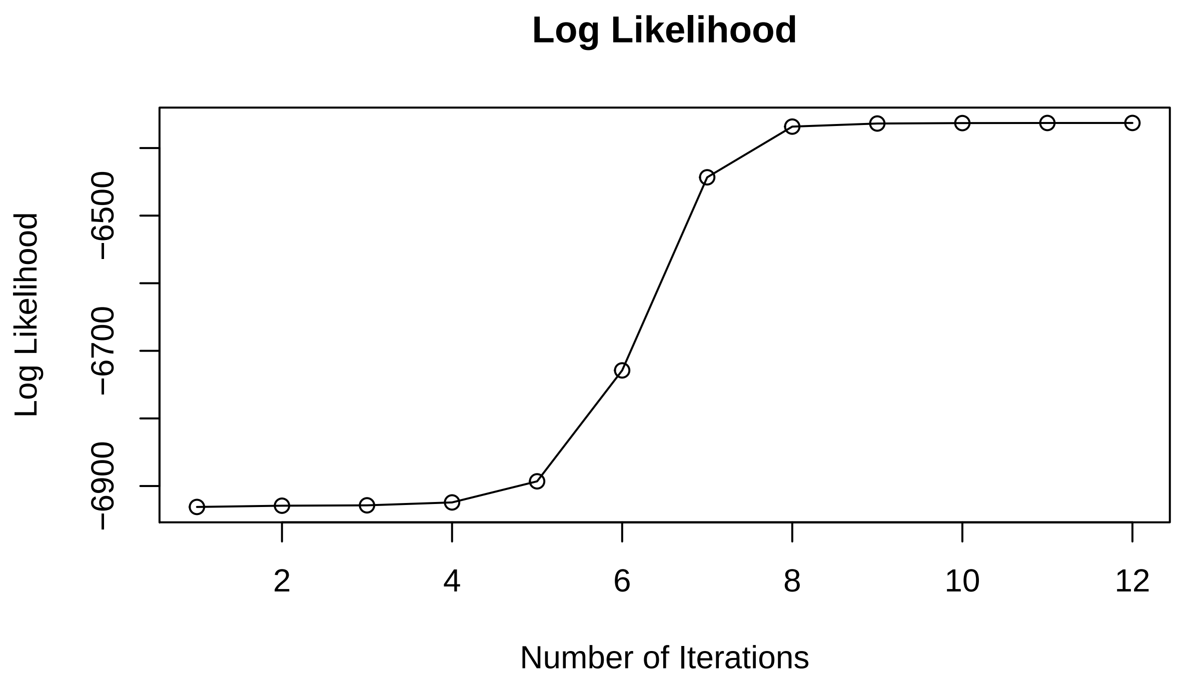


2. Mixture Models

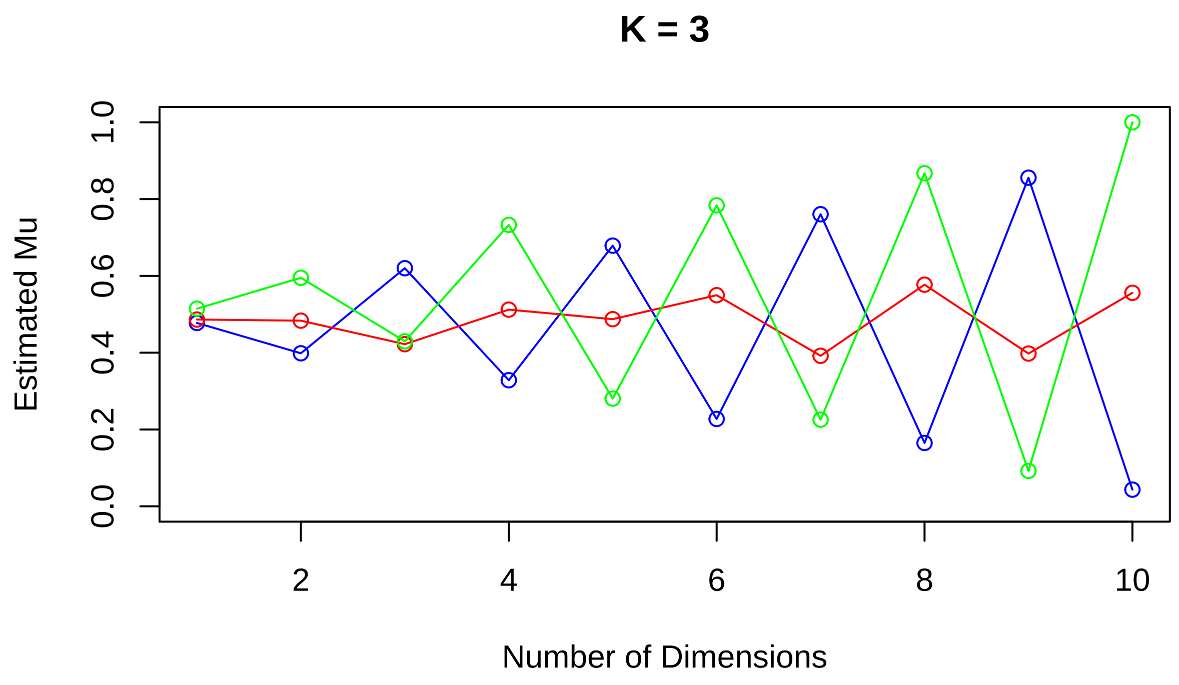


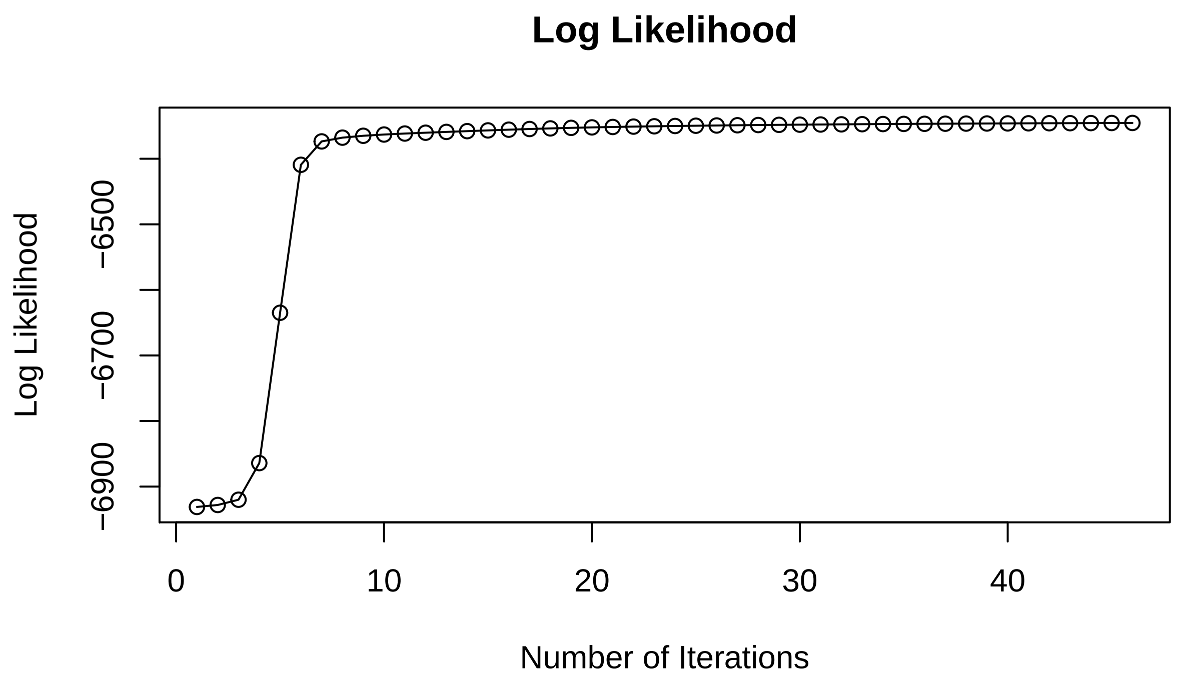
```
## iteration:  1 log likelihood:  -6930.975
## iteration:  2 log likelihood:  -6929.125
## iteration:  3 log likelihood:  -6928.562
## iteration:  4 log likelihood:  -6924.281
## iteration:  5 log likelihood:  -6893.055
## iteration:  6 log likelihood:  -6728.948
## iteration:  7 log likelihood:  -6443.28
## iteration:  8 log likelihood:  -6368.318
## iteration:  9 log likelihood:  -6363.734
## iteration:  10 log likelihood:  -6363.109
## iteration:  11 log likelihood:  -6362.947
## iteration:  12 log likelihood:  -6362.897
```



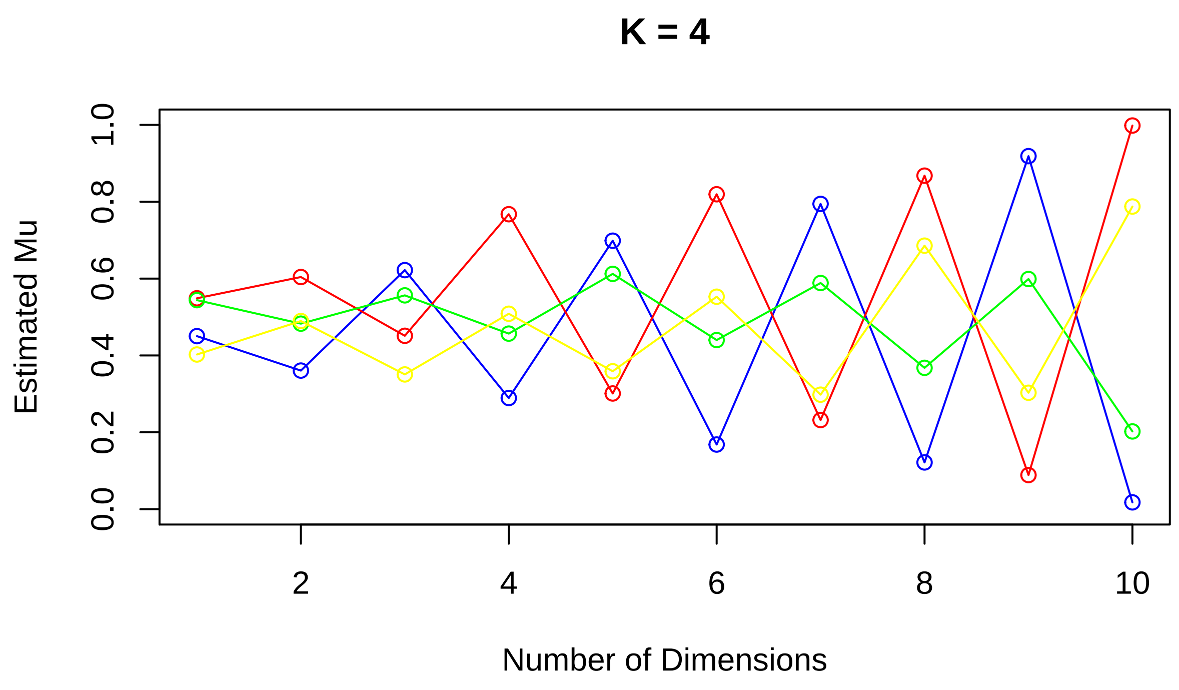


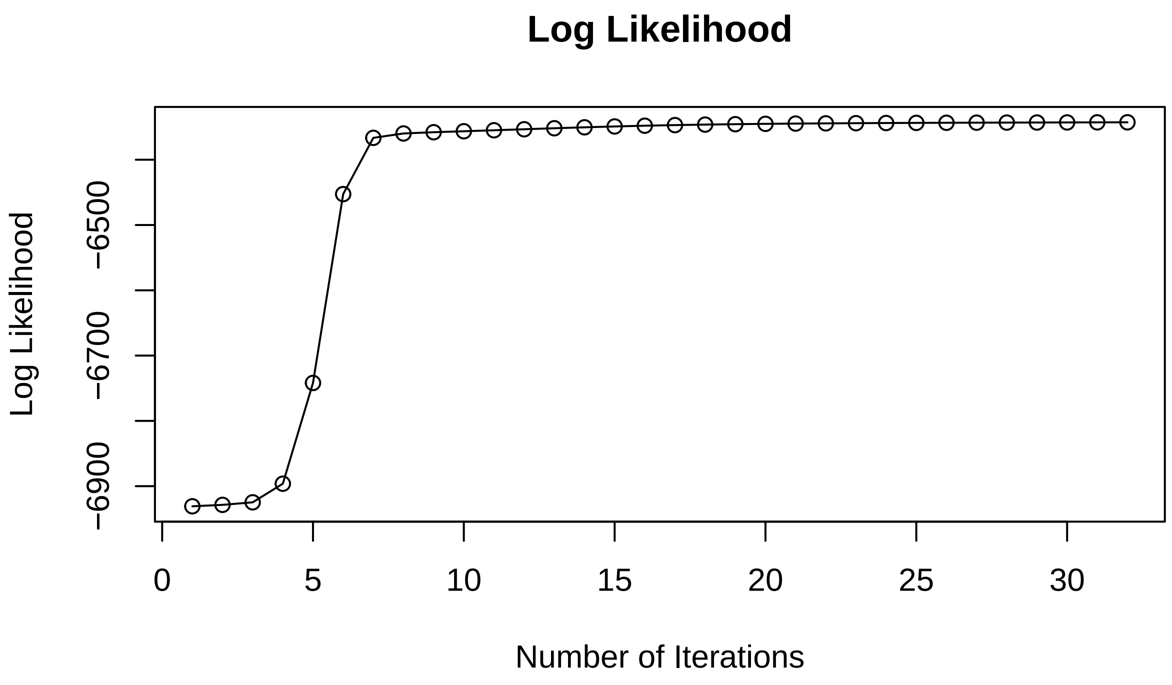
```
## iteration:  1 log likelihood:  -6931.064
## iteration:  2 log likelihood:  -6928.051
## iteration:  3 log likelihood:  -6920.026
## iteration:  4 log likelihood:  -6864.176
## iteration:  5 log likelihood:  -6634.916
## iteration:  6 log likelihood:  -6409.234
## iteration:  7 log likelihood:  -6373.593
## iteration:  8 log likelihood:  -6367.833
## iteration:  9 log likelihood:  -6364.983
## iteration:  10 log likelihood:  -6363.074
## iteration:  11 log likelihood:  -6361.594
## iteration:  12 log likelihood:  -6360.309
## iteration:  13 log likelihood:  -6359.103
## iteration:  14 log likelihood:  -6357.93
## iteration:  15 log likelihood:  -6356.786
## iteration:  16 log likelihood:  -6355.689
## iteration:  17 log likelihood:  -6354.668
## iteration:  18 log likelihood:  -6353.742
## iteration:  19 log likelihood:  -6352.92
## iteration:  20 log likelihood:  -6352.199
## iteration:  21 log likelihood:  -6351.567
## iteration:  22 log likelihood:  -6351.011
## iteration:  23 log likelihood:  -6350.515
## iteration:  24 log likelihood:  -6350.069
## iteration:  25 log likelihood:  -6349.661
## iteration:  26 log likelihood:  -6349.286
## iteration:  27 log likelihood:  -6348.938
## iteration:  28 log likelihood:  -6348.616
## iteration:  29 log likelihood:  -6348.315
## iteration:  30 log likelihood:  -6348.036
## iteration:  31 log likelihood:  -6347.776
## iteration:  32 log likelihood:  -6347.534
## iteration:  33 log likelihood:  -6347.308
## iteration:  34 log likelihood:  -6347.099
## iteration:  35 log likelihood:  -6346.904
## iteration:  36 log likelihood:  -6346.722
## iteration:  37 log likelihood:  -6346.553
## iteration:  38 log likelihood:  -6346.394
## iteration:  39 log likelihood:  -6346.246
## iteration:  40 log likelihood:  -6346.107
## iteration:  41 log likelihood:  -6345.977
## iteration:  42 log likelihood:  -6345.854
## iteration:  43 log likelihood:  -6345.739
## iteration:  44 log likelihood:  -6345.63
## iteration:  45 log likelihood:  -6345.528
## iteration:  46 log likelihood:  -6345.431
```





```
## iteration:  1 log likelihood:  -6930.838
## iteration:  2 log likelihood:  -6928.641
## iteration:  3 log likelihood:  -6924.748
## iteration:  4 log likelihood:  -6896.25
## iteration:  5 log likelihood:  -6741.896
## iteration:  6 log likelihood:  -6452.658
## iteration:  7 log likelihood:  -6366.493
## iteration:  8 log likelihood:  -6359.764
## iteration:  9 log likelihood:  -6357.876
## iteration:  10 log likelihood:  -6356.372
## iteration:  11 log likelihood:  -6354.86
## iteration:  12 log likelihood:  -6353.31
## iteration:  13 log likelihood:  -6351.776
## iteration:  14 log likelihood:  -6350.33
## iteration:  15 log likelihood:  -6349.03
## iteration:  16 log likelihood:  -6347.908
## iteration:  17 log likelihood:  -6346.968
## iteration:  18 log likelihood:  -6346.196
## iteration:  19 log likelihood:  -6345.566
## iteration:  20 log likelihood:  -6345.055
## iteration:  21 log likelihood:  -6344.637
## iteration:  22 log likelihood:  -6344.293
## iteration:  23 log likelihood:  -6344.008
## iteration:  24 log likelihood:  -6343.768
## iteration:  25 log likelihood:  -6343.563
## iteration:  26 log likelihood:  -6343.387
## iteration:  27 log likelihood:  -6343.233
## iteration:  28 log likelihood:  -6343.097
## iteration:  29 log likelihood:  -6342.975
## iteration:  30 log likelihood:  -6342.864
## iteration:  31 log likelihood:  -6342.762
## iteration:  32 log likelihood:  -6342.668
```





Analysis:

- After we created our data points using parameters mu and pi, the EM function takes only the data points and guesses mu and pi(pi here is like a prior probablity based on our best guess that the responsiblities are equal but it can be set differently)
- In E-step we compute the posterior responsiblities for each observation based on Bayes Theorem, if K=2 we make the assumbtion that the points are in two clusters, and if it is 3 then we assume 3 clusters and so on.

$$\gamma(z_{nk}) = \frac{\pi_k * p(x_n|\mu_k)}{\sum_{j=1}^{K} \pi_j * p(x_n|\mu_j)}, where\ i = 1, 2, ..., N$$

- In the M-step we updated our mu and pi based on the new responsiblities.

$$\mu_k^{new} = \frac{1}{N_k} \sum_{n=1}^{N} \gamma(z_{nk}) x_n$$

$$\pi_k^{new} = \frac{N_k}{N}$$

- Log likelikelihood then is calculated (doesn't matter if it is done before M-step or after) to check if the updated values of mu and pi are converging to the true values.

- In the case of K=2 little convergence is gained after the 8th iteration and stopped at 12th iteration when we started to get minimum change. The resulted values were close to the true μ_1 and μ_2, this is because μ_3 value was 0.5 in the middle of the other clusters, so this third cluster split between the first two clusters without complications.
- When K=3 the convergence also starts to plateau when we reached the 8th iteration and stopped at the 46th iteration. This is because the values of first two true mus are overlapping the the third mu, making it complicated for the algorithm to distinguish the third cluster from the other two (which are more distinct from each other).
- The same thing happened when K=4 also the convergence also start to plateau when we reached the 8th iteration and stopped at 32nd iteration this time, this is because the third cluster around true Mu3 got split into two cluster, one that is close to the first Mu and another one close to the second Mu.
- We can conclude from fact that the algorithm starts to gain little convergence at the 8th iteration. This is because the first two original clusters are more distinguishable from the third cluster because their Mus overlapp the third mu, and once the estimated Mus for these two clusters are gained, it's difficult for the algorithm to recognize the third cluster.

##Apendix

```r
setwd('D:/Machine Learning/Workshop/Machine Learning/Block 2/Lab 1 Block 2')
RNGversion('3.5.1')

sp = as.data.frame(read.csv2("Data/spambase.csv"))
sp$Spam = as.factor(sp$Spam)

n=dim(sp)[1]
set.seed(12345)
id=sample(1:n, floor(n*(2/3)))
train=sp[id,]
test=sp[-id,]

library(mboost)

ada_model = blackboost(Spam ~., data = train, control = boost_control(mstop = 100, nu = 0.5),
                       family = AdaExp())
fitted.results_test= predict(ada_model,newdata=test, type= "class")
ada_confmat_test = table("Y"=test$Spam,"Y hat"=fitted.results_test)
print(table("Y"=test$Spam,"Y hat"=fitted.results_test))

print(paste('Error for ada model:',1-sum(diag(ada_confmat_test)) / sum(ada_confmat_test)))


library(ggplot2)

acc=0
n = seq(10, 100, by = 10)
for(i in n){
  ada_model = blackboost(Spam ~., data = train, control = boost_control(mstop = i, nu = 0.5),
                       family = AdaExp())
  fitted.results_test= predict(ada_model,newdata=test)
  fitted.results_test = ifelse(fitted.results_test > 0.1,1,0)
  misClasificError_test = mean(fitted.results_test != test$Spam)
  ada_confmat_test = table("Y"=test$Spam,"Y hat"=fitted.results_test)
  acc[i] = sum(diag(ada_confmat_test)) / sum(ada_confmat_test)
}


y_error <- na.omit(1-acc)
y_error = y_error[-1]
x_trees <- n
ada_m <- na.omit(data.frame(x_trees, y_error))

ggplot()+
  ylim(0.04,0.10)+
  ggtitle(" Error for Adaboost ")+
  geom_point(data=ada_m, aes(x=x_trees, y=y_error), size=2)+
  geom_line(data=ada_m, aes(x=x_trees, y=y_error), size=1)

library(randomForest)


misClasificError <- 0
for (i in seq(10, 100, 10)) {
  rf<-randomForest(Spam ~ ., data= train, ntree=i)
  p <- predict(rf, newdata = test, type = "class")
  mat <- table("Predicted" = p, "Actual" = test$Spam)
  misClasificError[(i / 10)] <- mean(p != test$Spam)
}
list("Missclassification Rate" = min(misClasificError), "Confusion Matrix" = mat)


rf_df<-data.frame("rferror"=misClasificError,"rftrees"=seq(10,100,10))

ggplot()+
  ylim(0.04,0.10)+
  ggtitle(" Error for RandomForest ")+
  geom_point(data=rf_df, aes(x=rftrees, y=rferror), size=2)+
  geom_line(data=rf_df, aes(x=rftrees, y=rferror), size=1)

## Mixed models / EM algorithm ##

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),
     main = "Original Data",
     xlab = "Number of Dimensions",
     ylab = "True Mu")
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])
  }
}
em_algorithm <- function(c){
  K=c # 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
  # Random initialization of the paramters
  pi <- runif(K,0.49,0.51)
  pi <- pi / sum(pi)
  for(k in 1:K) {
    mu[k,] <- runif(D,0.49,0.51)
  }
  pi
  mu
  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 (responsiblities)
    # Your code here
    for (n in 1:N){
      phi = c()
      for (j in 1:K){
        y1 = mu[j,]^x[n,]
        y2 = (1- mu[j,])^(1-x[n,])
        phi = c(phi, prod(y1,y2))
      }

      z[n,] = (pi*phi) / sum(pi*phi)
    }
    #Log likelihood computation.
    # Your code here
    likelihood = matrix(0,1000,K)
    llik[it] = 0
    for(n in 1:N){
      for (k in 1:K){
        likelihood[n,k] = pi[k]*prod( ((mu[k,]^x[n,])*((1-mu[k,])^(1-x[n,]))))
      }
      llik[it] = sum(log(rowSums(likelihood)))
    }
    cat("iteration: ", it, "log likelihood: ", llik[it], "\n")
    flush.console()
    # Stop if the lok likelihood has not changed significantly
    # Your code here
    if (it > 1){
      if (llik[it]-llik[it-1] < min_change){
        if(K == 2){
          plot(mu[1,],
               type="o",
               col="blue",
               ylim=c(0,1),
               main = "K = 2",
               xlab = "Number of Dimensions",
               ylab = "Estimated Mu")
          points(mu[2,], type="o", col="red")
        }
        else if(K == 3){
          plot(mu[1,],
               type="o",
               col="blue",
               ylim=c(0,1),
               main = "K = 3",
               xlab = "Number of Dimensions",
               ylab = "Estimated Mu")
          points(mu[2,], type="o", col="red")
          points(mu[3,], type="o", col="green")
        }
        else if(K == 4){
          plot(mu[1,],
               type="o",
               col="blue",
               ylim=c(0,1),
               main = "K = 4",
               xlab = "Number of Dimensions",
               ylab = "Estimated Mu")
          points(mu[2,], type="o", col="red")
          points(mu[3,], type="o", col="green")
          points(mu[4,], type="o", col="yellow")
        }
        break()
      }
    }
    #M-step: ML parameter estimation from the data and fractional component assignments
    # Your code here
    mu = (t(z) %*% x) /colSums(z)
    # N - Total no. of observations
    pi = colSums(z)/N
  }
  pi
  mu
```

```
  # plot(llik[1:it],
  #      type="o",
  #      main = "Log Likelihood",
  #      xlab = "Number of Iterations",
  #      ylab = "Log Likelihood")
}

em_algorithm(2)
em_algorithm(3)
em_algorithm(4)
```