Machine Learning - Block02 Assignment 01

Agustín Valencia - aguva779

1. Ensemble Methods

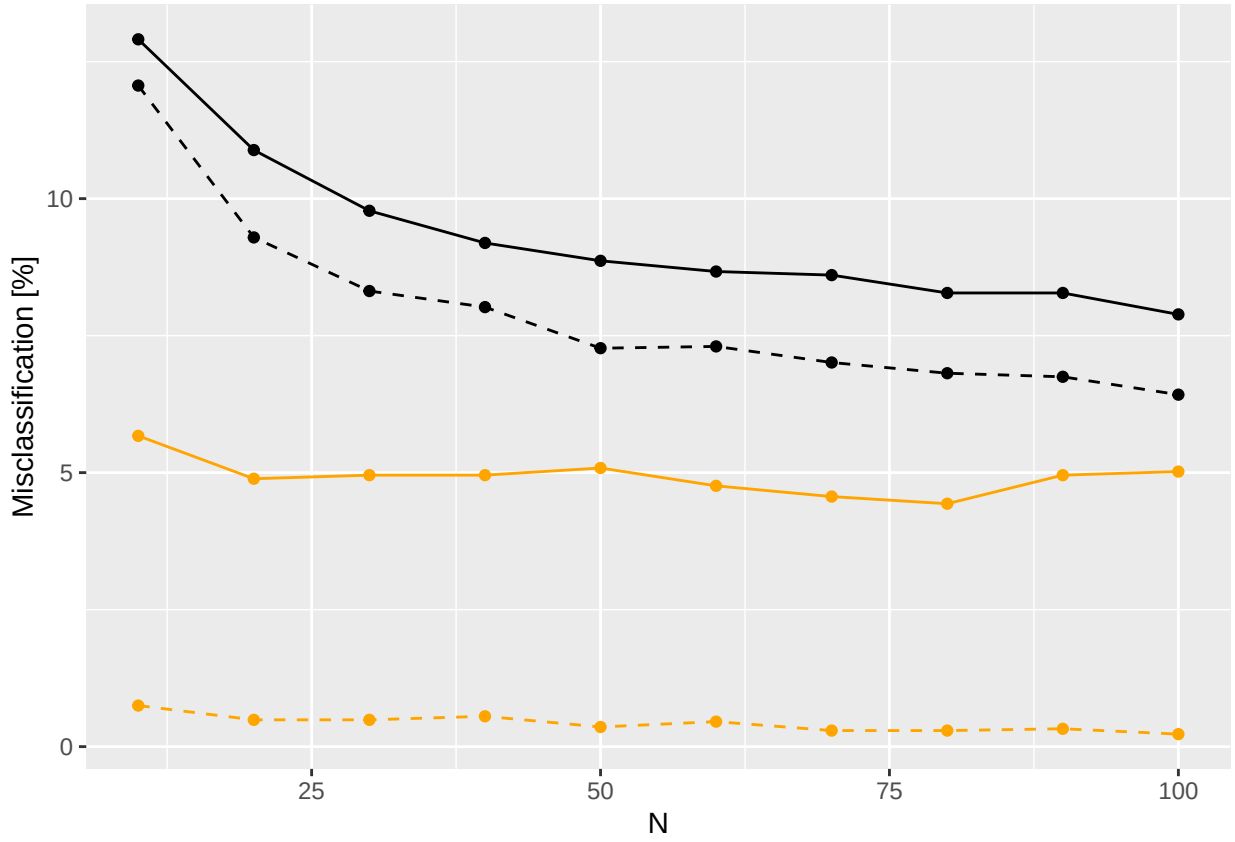
The file spambase.csv contains information about the frequency of various words, characters, etc. for a total of 4601 e-mails. Furthermore, these e-mails have been classified as spams (spam = 1) or regular e-mails (spam = 0). You can find more information about these data at https://archive.ics.uci.edu/ml/datasets/Spambase.

Your task is to evaluate the performance of Adaboost classification trees and random forests on the spam data. Specifically, provide a plot showing the error rates when the number of trees considered are 10, 20, . . . , 100. To estimate the error rates, use 2/3 of the data for training and 1/3 as hold-out test data.

To learn Adaboost classification trees, use the function blackboost() of the R package mboost. Specify the loss function corresponding to Adaboost with the parameter family. To learn random forests, use the function randomForest of the R package randomForest. To load the data, you may want to use the following code:

Solution

For trees trained using adaboost we have:



In the above graph the black lines corresponds to trees trained using adaboost with depth = N and the orange lines corresponds to random forests with N trees. In the same sense, dotted lines refers to training results and continuous to test results.

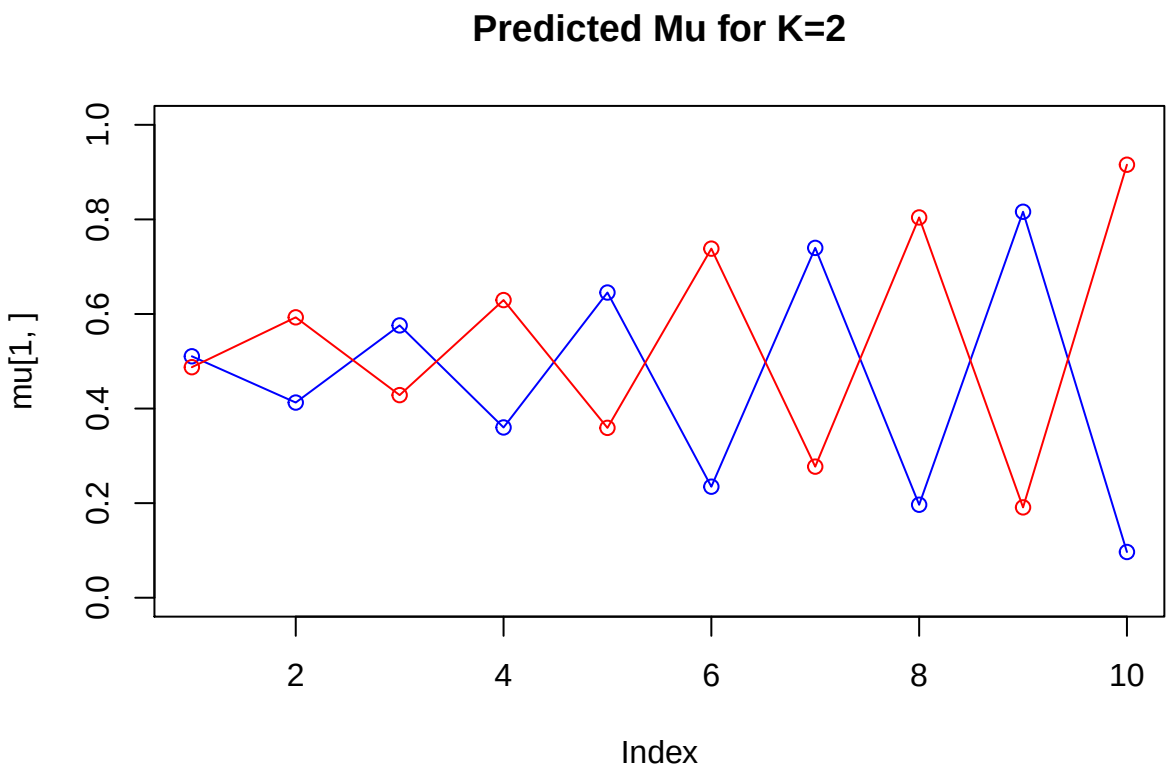
It can be seen that random forests need less effort (depth-wise) than boosted trees to get better results.

2. Mixture Models

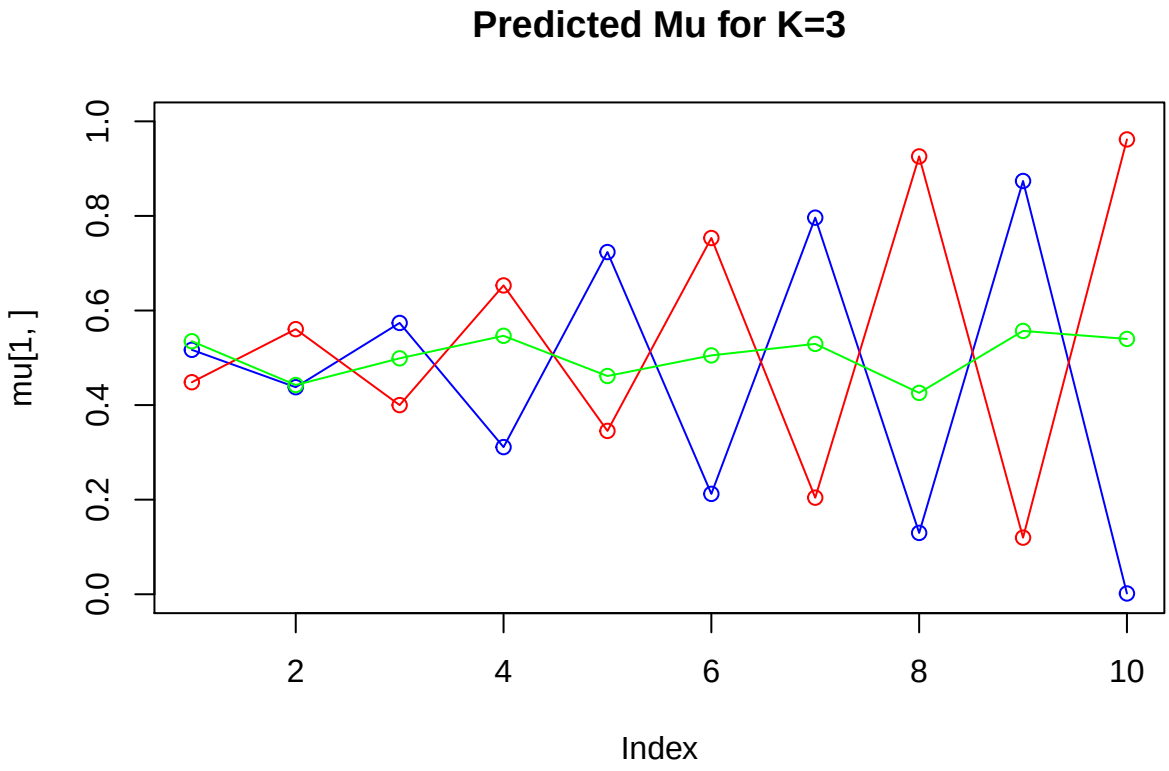
Your task is to implement the EM algorithm for mixtures of multivariate Benoulli distributions. Please use the template in the next page to solve the assignment. Then, use your implementation to show what happens when your mixture models has too few and too many components, i.e. set $K = 2, 3, 4$ and compare results. Please provide a short explanation as well.

Solution

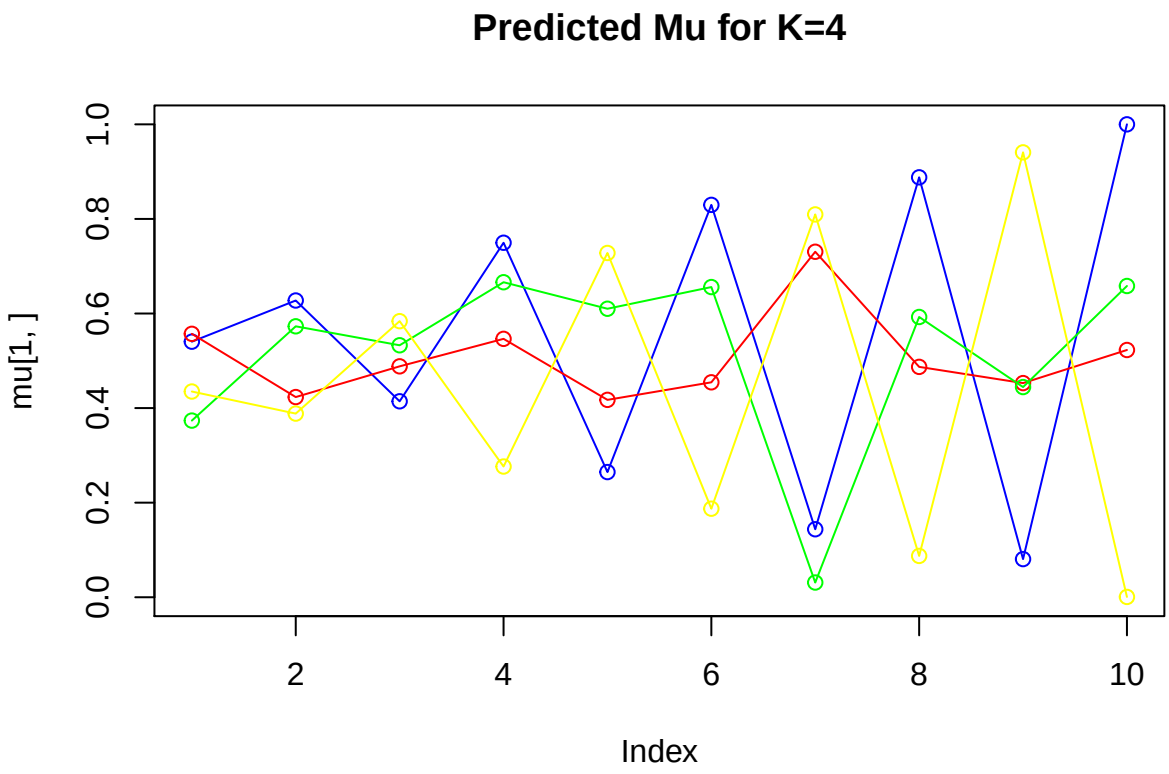
After running EM algorithm for $K = \{2, 3, 4\}$ the following maximum likelihoods are obtained:




```
## The max likelihood for K = 2 :  -6475.789
```

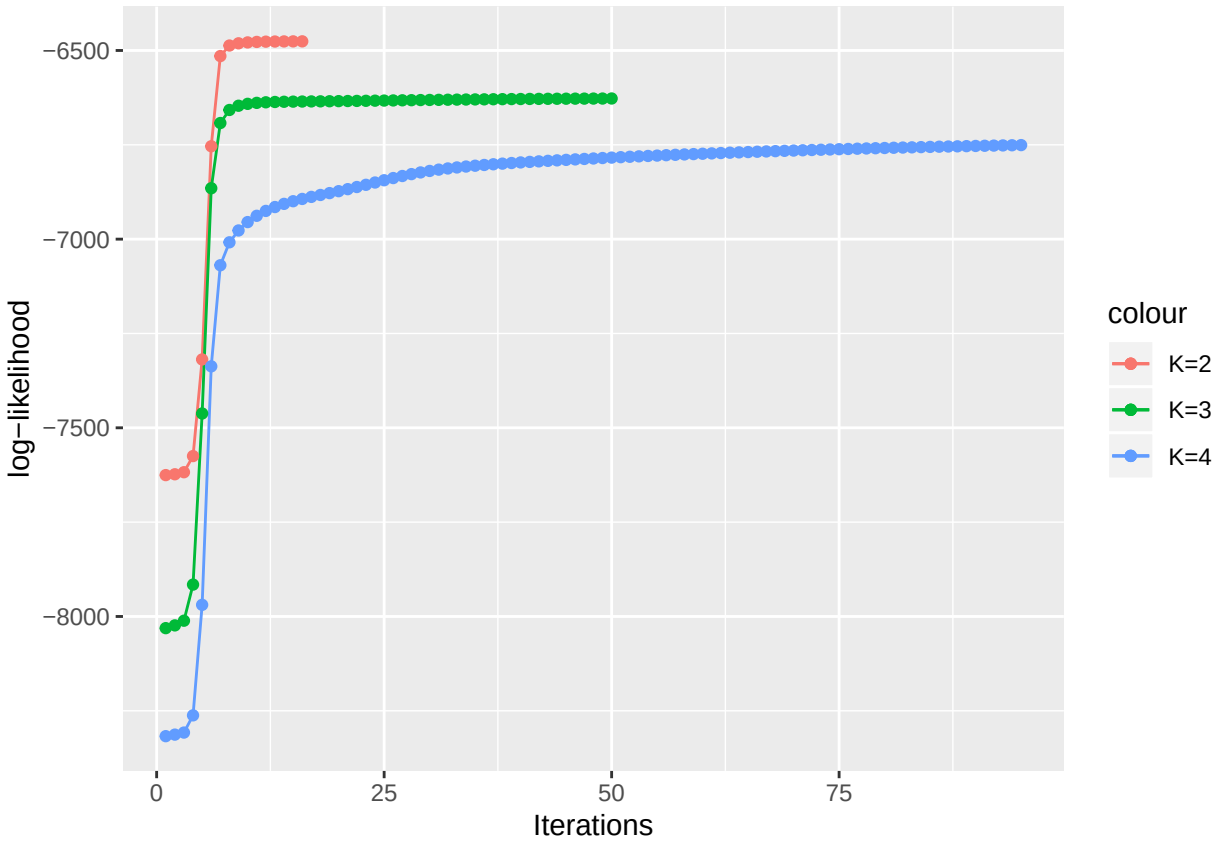



```
## The max likelihood for K = 3 :  -6627.258
```




```
## The max likelihood for K = 4 :  -6750.734
```

The following graph shows the consistent encreasing behavior of the likelihood towards 1, thus the log-likelihood will tend try reach zero.


−6500
−7000
−7500
−8000
log−likelihood
0
25
50
75
Iterations
colour
K=2
K=3
K=4

Apendix A - Code

```
knitr::opts_chunk$set(echo = TRUE)
knitr::opts_knit$set(root.dir = "C:/Users/valen/OneDrive/Documents/Sem01/P02/Machine_Learning/Assignmen
library(mboost)
library(randomForest)
library(partykit)
library(ggplot2)
library(GLDEX)
set.seed(1234567890)
RNGversion("3.5.1")
sp <- read.csv2("spambase.csv")
sp$Spam <- as.factor(sp$Spam)


##### ------------------------------------------------------------------------
#####                                 QUESTION 1
##### ------------------------------------------------------------------------

## Splitting data
n <- dim(sp)[1]
idxs <- sample(1:n, floor(2*n/3))
train <- sp[idxs,]
test <- sp[-idxs,]

get_missed <- function (true, predicted) {
    confussion <- table(true, predicted)
    fn <- confussion[1,2]
    fp <- confussion[2,1]
    total <- sum(confussion)
    miss <- (fp + fn) / total * 100
    return(miss)
}

# Training
nums <- seq(10,100,10)
formula <- Spam ~ .
error_rates_train_bb <- c()
error_rates_test_bb <- c()
error_rates_train_rf <- c()
error_rates_test_rf <- c()
depths <- c()
for (i in nums) {
    bb <- blackboost (
            Spam ~ .,
            data = train,
            family = AdaExp(),
            control = boost_control(mstop = i)
     )

    rf <- randomForest(
            Spam ~ .,
            data = train,
```

```
            ntree = i
    )

    predicted <- predict(bb, train, type = "class")
    miss <- get_missed(train$Spam, predicted)
    error_rates_train_bb <- append(error_rates_train_bb, miss)

    predicted <- predict(bb, test, type = "class")
    miss <- get_missed(test$Spam, predicted)
    error_rates_test_bb <- append(error_rates_test_bb, miss)

    predicted <- predict(rf, train, type = "class")
    miss <- get_missed(train$Spam, predicted)
    error_rates_train_rf <- append(error_rates_train_rf, miss)

    predicted <- predict(rf, test, type = "class")
    miss <- get_missed(test$Spam, predicted)
    error_rates_test_rf <- append(error_rates_test_rf, miss)

    depths <- append(depths, i)
}
df <- data.frame(
    nums = nums,
    error_rates_train_bb = error_rates_train_bb,
    error_rates_test_bb = error_rates_test_bb,
    error_rates_train_rf = error_rates_train_rf,
    error_rates_test_rf = error_rates_test_rf
)
p <- ggplot() +
    geom_line(aes(x = depths, y = error_rates_train_bb), color = "black", linetype = "dashed") +
    geom_line(aes(x = depths, y = error_rates_test_bb), color = "black") +
    geom_line(aes(x = depths, y = error_rates_train_rf), color = "orange", linetype = "dashed") +
    geom_line(aes(x = depths, y = error_rates_test_rf), color = "orange") +
    geom_point(aes(x = depths, y = error_rates_train_bb), color = "black") +
    geom_point(aes(x = depths, y = error_rates_test_bb), color = "black") +
    geom_point(aes(x = depths, y = error_rates_train_rf), color = "orange") +
    geom_point(aes(x = depths, y = error_rates_test_rf), color = "orange") +
    xlab("N") + ylab("Misclassification [%]")
p

##### ------------------------------------------------------------------------
#####                                QUESTION 2
##### ------------------------------------------------------------------------

EM <- function(K) {
    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)
#true_mu[4,]=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])
    }
}
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)
}

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")

    # E-step:
    for (n in 1:N) {
        for (k in 1:K) {
            z[n,k] <- pi[k] * prod((mu[k,]^x[n,]) * ((1-mu[k,])^(1-x[n,])))
        }
        z[n,] <- z[n,] / rowSums(z)[n]
    }

    # Log likelihood computation.
    for (n in 1:N) {
        for (k in 1:K) {
            log_aux <- sum(x[n,] * log(mu[k,]) + (1-x[n,]) * log(1-mu[k,]))
            llik[it] <- llik[it] + z[n,k] * (log(pi[k]) + log_aux)
        }
    }

    forceStop <- FALSE
    if (is.na(llik[it])) {
        forceStop <- TRUE
    }
    # cat("iteration: ", it, "log likelihood: ", llik[it], "\n")
    # Stop if the log likelihood has not changed significantly
    if (forceStop | it > 1) {
```

```
            if (forceStop | (abs(llik[it] - llik[it - 1]) < min_change) | (it == max_it)) {
                titStr <- paste("Predicted Mu for K=", K, sep = "")
                p <- plot(mu[1,], type = "o", col = "blue", ylim = c(0, 1), main = titStr)
                if (K >= 2) {
                    p <- p + points(mu[2,], type = "o", col = "red")
                }
                if (K >= 3) {
                    p <- p + points(mu[3,], type = "o", col = "green")
                }
                if (K >= 4) {
                    p <- p + points(mu[4,], type = "o", col = "yellow")
                }

                return(
                list(
                    pi = pi,
                    mu = mu,
                    llik = llik,
                    p = p
                )
            )
            }
        }

        # M-step:
        pi_ML <- vector(length = K)
        for (k in 1:K) {
            pi_ML[k] <- mean(z[,k])
        }

        mu_ML <- matrix(nrow=K, ncol=D)
        for (k in 1:K){
            mu_ML[k,] <- 0
            for (d in 1:D){
              mu_ML[k,d] <- sum(x[,d] * z[,k])/sum(z[,k])
            }
        }
        pi <- pi_ML
        mu <- mu_ML
    }
}

cleanLoglik <- function(lik) {
    if (length(lik[is.na(lik)]) > 0) {
        lik <- lik[-which.na(lik)]
    }
    lik <- fun.zero.omit(lik)
    return(lik)
}

em2 <- EM(2)
llik2 <- cleanLoglik(em2$llik)
cat("The max likelihood for K = 2 : ", max(llik2), "\n")
```

```
em3 <- EM(3)
llik3 <- cleanLoglik(em3$llik)
cat("The max likelihood for K = 3 : ", max(llik3), "\n")

em4 <- EM(4)
llik4 <- cleanLoglik(em4$llik)
cat("The max likelihood for K = 4 : ", max(llik4), "\n")
ggplot() +
    geom_line(aes(x=1:length(llik2), y=llik2, color="K=2")) +
    geom_line(aes(x=1:length(llik3), y=llik3, color="K=3")) +
    geom_line(aes(x=1:length(llik4), y=llik4, color="K=4")) +
    geom_point(aes(x=1:length(llik2), y=llik2, color="K=2")) +
    geom_point(aes(x=1:length(llik3), y=llik3, color="K=3")) +
    geom_point(aes(x=1:length(llik4), y=llik4, color="K=4")) +
    xlab("Iterations") + ylab("log-likelihood")
```