Machine Learning Computer Lab 1 Block2 (Group A7)

Qinyuan Qi(qinqi464) Satya Sai Naga Jaya Koushik Pilla (satpi345)
Daniele Bozzoli(danbo826)

2023-12-07

Assignment 1: ENSEMBLE METHODS

Answer:

Following is the functions for the assignments.

```
###############  Assignment 1 Ensemble methods #################################

compare_method1 <- function(x1, x2) {
  return(x1 < x2)
}

compare_method2 <- function(x1, x2) {
  return(x1 < 0.5)
}

compare_method3 <- function(x1, x2) {
  return((x1 > 0.5 & x2 > 0.5) | (x1 < 0.5 & x2 < 0.5))
}

ensemble_method <- function(record_number, tree_number, node_size, compare_method) {
  # init data set (copy from pdf file)


  x1 <- runif(record_number)
  x2 <- runif(record_number)
  trdata <- cbind(x1, x2)
  y <- as.numeric(compare_method(x1, x2))
  trlabels <- as.factor(y)
  #plot(x1, x2, col = (y + 1))

  # training the model using random forest
  rf_model <- randomForest(trdata, trlabels, ntree = tree_number,
                           nodesize = node_size, keep.forest = TRUE)

  rf_pred <- predict(rf_model, newdata = trdata, class = "classification")

  # calculate the misclassification rate
  misclassification_rate <- sum(as.numeric(rf_pred != trlabels)) /
  length(trlabels)

  return (misclassification_rate)
}
```

When node_size = 25 and record_numbers = 100 , and check x1 < x2 , we get the following result.

```
## The misclassification rate for 1 tree is:  0.05
```

```
## The misclassification rate for 10 tree is:  0.04
```

```
## The misclassification rate for 100 tree is:  0.03
```

When we set node_size = 25 and record_numbers = 100 and 1000 and check x1 < x2 , and run it 1000 times, we get the following result.

```
## The mean of misclassification rate for 1 tree with record number = 100 is:  0.15479
```

```
## The mean of misclassification rate for 10 tree with record number = 100 is:  0.07448
```

```
## The mean of misclassification rate for 100 tree with record number = 100 is:  0.0473
```

```
## The var of misclassification rate for 1 tree with record number = 100 is:  0.003607563
```

```
## The var of misclassification rate for 10 tree with record number = 100 is:  0.001012943
```

```
## The var of misclassification rate for 100 tree with record number = 100 is:  0.0006695796
```

```
## The mean of misclassification rate for 1 tree with record number = 1000 is:  0.047593
```

```
## The mean of misclassification rate for 10 tree with record number = 1000 is:  0.014158
```

```
## The mean of misclassification rate for 100 tree with record number = 1000 is:  0.00654
```

```
## The var of misclassification rate for 1 tree with record number = 1000 is:  0.0001721915
```

```
## The var of misclassification rate for 10 tree with record number = 1000 is:  2.100204e-05
```

```
## The var of misclassification rate for 100 tree with record number = 1000 is:  8.570971e-06
```

When we set node_size = 25 and record_numbers = 100 and 1000 and check x1 < 0.5 , and run it 1000 times, we get the following result.

```
## The mean of misclassification rate for 1 tree with record number = 100 is:  0.06218
```

```
## The mean of misclassification rate for 10 tree with record number = 100 is:  0.00426
```

```
## The mean of misclassification rate for 100 tree with record number = 100 is:  0.00017
```

```
## The var of misclassification rate for 1 tree with record number = 100 is:  0.01056021
```

```
## The var of misclassification rate for 10 tree with record number = 100 is:  0.0001467992
```

```
## The var of misclassification rate for 100 tree with record number = 100 is:  4.875976e-06
```

```
## The mean of misclassification rate for 1 tree with record number = 1000 is:  0.005881
```

```
## The mean of misclassification rate for 10 tree with record number = 1000 is:  9.5e-05
```

```
## The mean of misclassification rate for 100 tree with record number = 1000 is:  0
```

```
## The var of misclassification rate for 1 tree with record number = 1000 is:  0.0001275724
```

```
## The var of misclassification rate for 10 tree with record number = 1000 is:  1.881632e-07
```

```
## The var of misclassification rate for 100 tree with record number = 1000 is:  0
```

When we set node_size = 25 and record_numbers = 100 and 1000 and check ((x1 > 0.5 & x2 > 0.5) | (x1 < 0.5 & x2 < 0.5)) , and run it 1000 times, we get the following result.

```
## The mean of misclassification rate for 1 tree with record number = 100 is:  0.15206
```

```
## The mean of misclassification rate for 10 tree with record number = 100 is:  0.03319
## The mean of misclassification rate for 100 tree with record number = 100 is:  0.00708
## The var of misclassification rate for 1 tree with record number = 100 is:  0.007047204
## The var of misclassification rate for 10 tree with record number = 100 is:  0.0006103342
## The var of misclassification rate for 100 tree with record number = 100 is:  9.236597e-05
## The mean of misclassification rate for 1 tree with record number = 1000 is:  0.018026
## The mean of misclassification rate for 10 tree with record number = 1000 is:  0.000471
## The mean of misclassification rate for 100 tree with record number = 1000 is:  1.1e-05
## The var of misclassification rate for 1 tree with record number = 1000 is:  0.0001785078
## The var of misclassification rate for 10 tree with record number = 1000 is:  6.958549e-07
## The var of misclassification rate for 100 tree with record number = 1000 is:  1.088989e-08
```

Using the results generated by code, we have the following data. Each row has 3 numbers, which is the value for 1,10,100 trees respectively.

mean of 100 size(Method1)

0.15479/0.07448/0.0473

var of 100 size(Method1)

0.003607563/0.001012943/0.0006695796

mean of 1000 size(Method1)

0.047593/0.014158/0.00654

var of 1000 size(Method1)

0.0001721915/2.100204e-05/8.570971e-06

mean of 100 size(Method2)

0.06218/0.00426/0.00017

var of 100 size(Method2)

0.01056021/0.0001467992/4.875976e-06

mean of 1000 size(Method2)

0.005881/9.5e-05/0

var of 1000 size(Method2)

0.0001275724/1.881632e-07/0

mean of 100 size(Method3)

0.15206/0.03319/0.00708

var of 100 size(Method3)

0.007047204/0.0006103342/9.236597e-05

mean of 1000 size(Method3)

0.018026/0.000471/1.1e-05

var of 1000 size(Method3)

0.0001785078/6.958549e-07/1.088989e-08

According to the result, we can found that with the number of trees in the random forest grow, the mean value of classification will decrease most of the time. This is mainly because of the we get more random trees and we will get a better result. This is also because of more trees will help to increase the robustness of the noisy data and increase stability.

For the 3rd data set, since it's more complicated, but with more trees, the mean of the mis classification rate drop dramatically with number of trees grow. It seems that with more complicated dataset, increase the size of trees will get more performance improvement than the relative simple data set.

Assignment 2: MIXTURE MODELS

Answer:

The code implemented and listed as below.

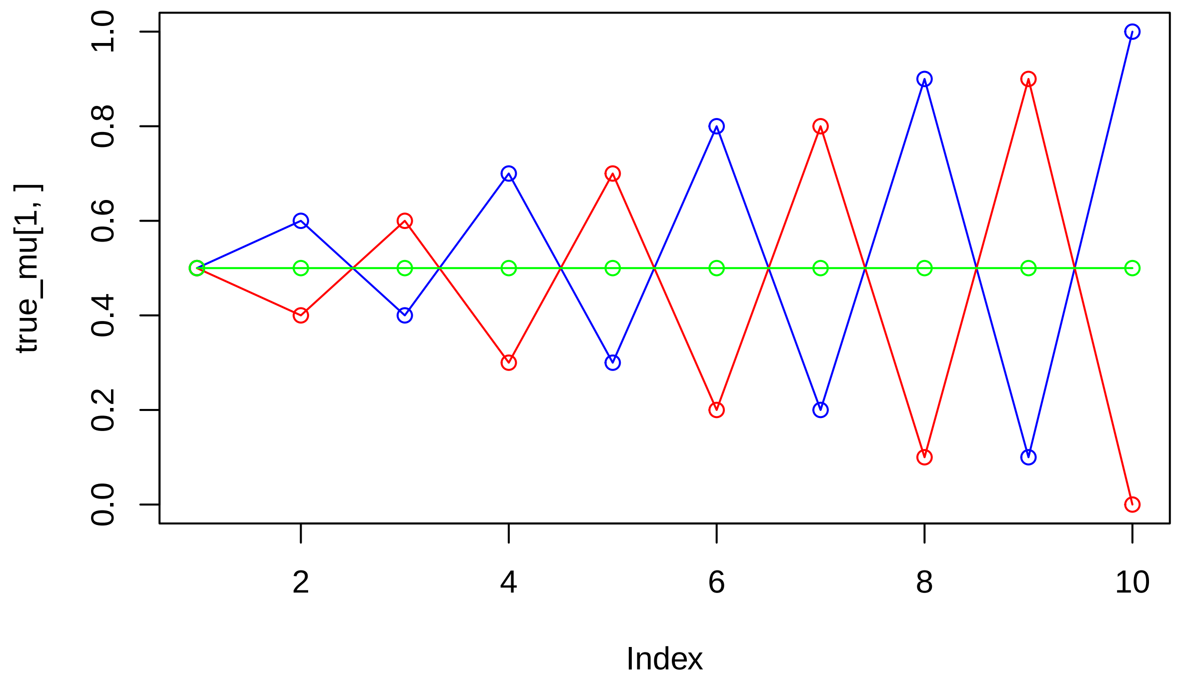
According to the output, when we change M value to 2,3 and 4, the output graph do not have significant change.

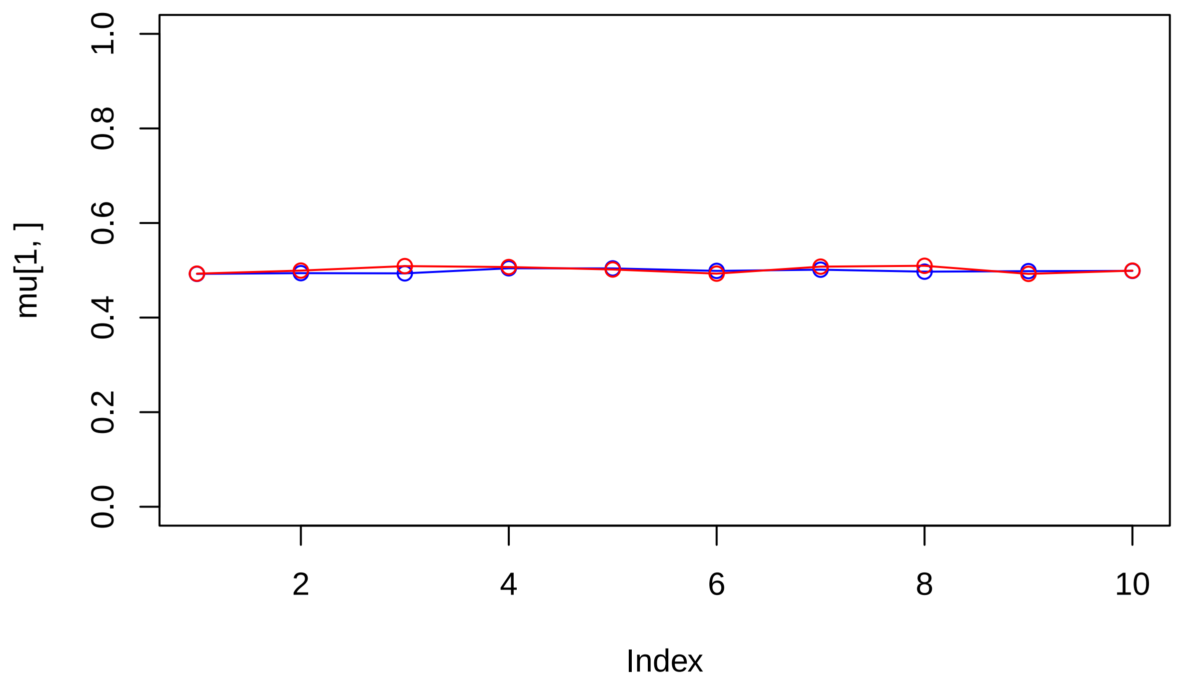
```r
######################## Assignment 2 MIXTURE MODELS ###########################

mixel_model_fun <- function(M = 3){
    cat("Generate model when M=",M)

    # init data set (copy from pdf file)
    set.seed(1234567890)

    # max number of EM iterations
    max_it <- 5

    # min change in log lik between two consecutive iterations
    min_change <- 0.1

    # number of training points
    n <- 1000

    # number of dimensions
    D <- 10

    # training data
    x <- matrix(nrow = n, ncol = D)

    # true mixing coefficients
    true_pi <- vector(length = 3)

    # true conditional distributions
    true_mu <- matrix(nrow = 3, ncol = D)

    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")
```

```r
# Producing the training data
for(i in 1:n) {
m <- sample(1:3, 1, prob = true_pi)
for(d in 1:D) {
    x[i, d] <- rbinom(1, 1, true_mu[m, d])
}
}

# number of clusters, set by function parameter
# M = 3

w <- matrix(nrow = n, ncol = M) # weights
pi <- vector(length = M) # mixing coefficients
mu <- matrix(nrow = M, ncol = D) # conditional distributions
llik <- vector(length = max_it) # log likelihood of the EM iterations

# Random initialization of the parameters
pi <- runif(M, 0.49, 0.51)
pi <- pi / sum(pi)
for (m in 1:M) {
mu[m, ] <- 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")
if (M > 2){
    points(mu[3,], type="o", col="green")
}
if (M > 3){
    points(mu[4,], type="o", col="yellow")
}
#Sys.sleep(2)

# E-step: Computation of the weights
for (i in 1:n) {
    for (m in 1:M) {
    w[i, m] <- pi[m] * prod((mu[m,] ^ x[i,]) * ((1 - mu[m,]) ^ (1 - x[i,])))
    }
    w[i,] <- w[i,] / sum(w[i,])
}

#Log likelihood computation.
llik[it] <- sum(log(apply(x, 1,
    function(xi) sum(pi * apply(mu, 1, function(mui)
                prod(mui^xi * (1 - mui)^(1 - xi)))))))

cat("iteration: ", it, "log likelihood: ", llik[it], "\n")
flush.console()
```

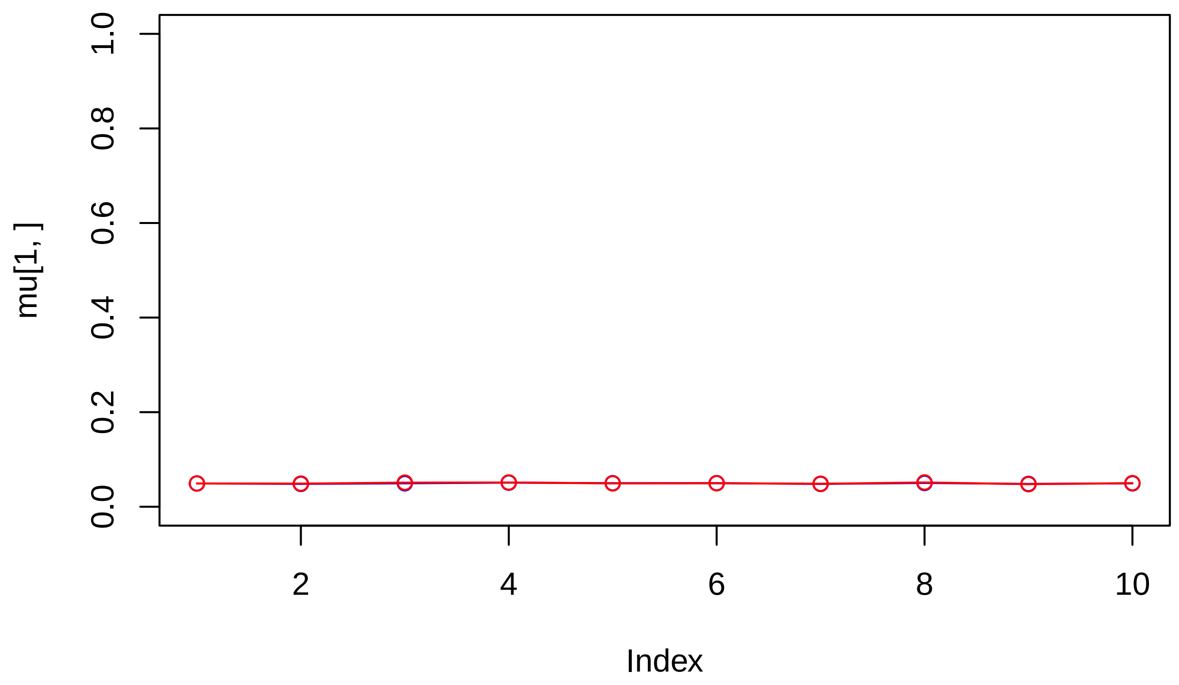
```
    # Stop if the lok likelihood has not changed significantly
    if (it > 1 && abs(llik[it] - llik[it - 1]) < min_change) {
        break
    }

    #M-step: ML parameter estimation from the data and weights
    for (m in 1:M) {
        pi[m] <- sum(w[, m]) / n
        mu[m,] <- colSums(x * w[, m]) / (sum(w[, m]) * D)
    }
    }

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

mixel_model_fun(2)
```

```
## Generate model when M= 2
```

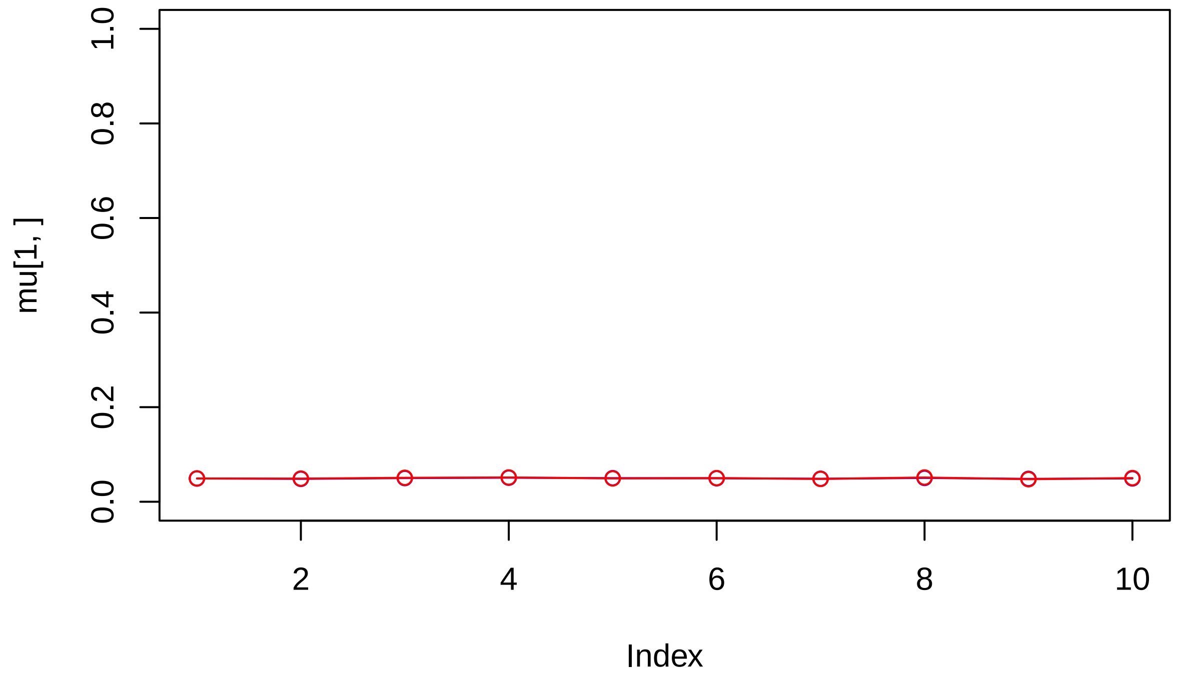




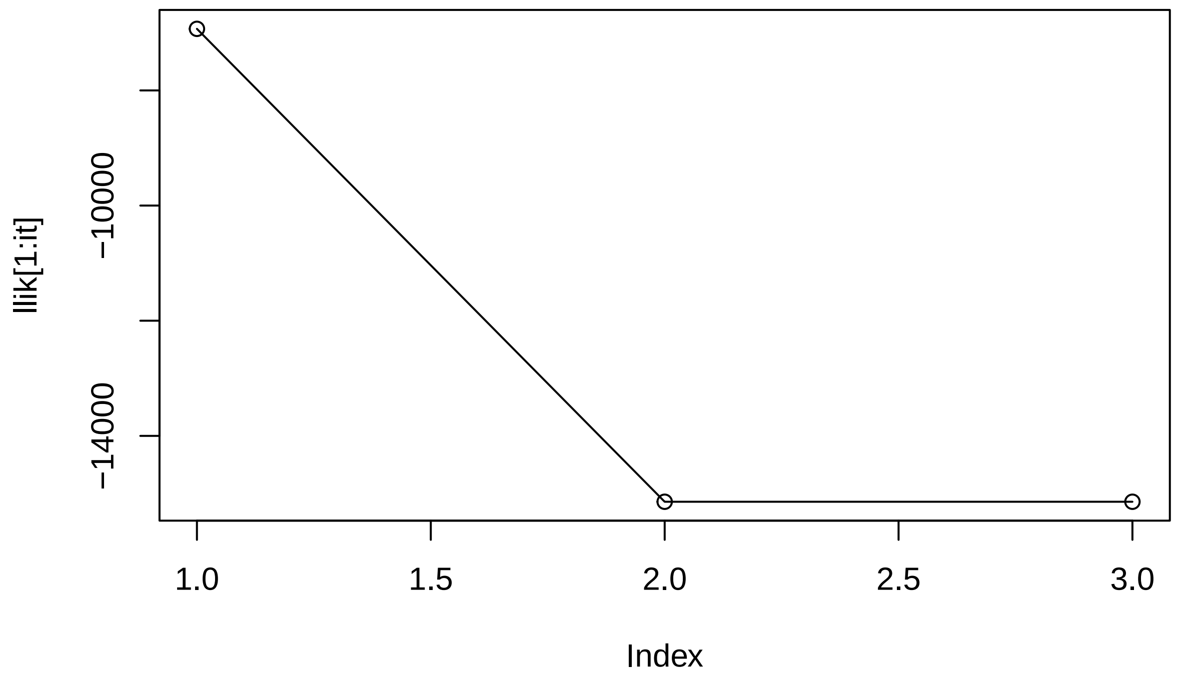
```
## iteration:  1 log likelihood:  -6930.975
```



```
## iteration:  2 log likelihood:  -15143.91
```

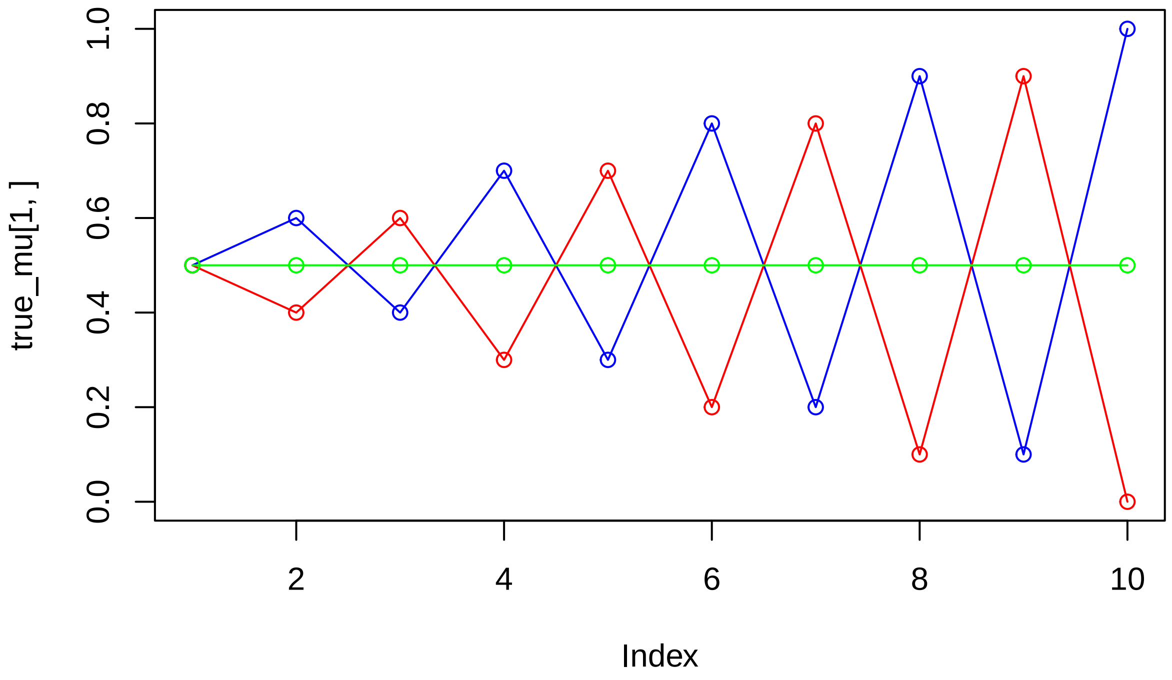


```
## iteration:  3 log likelihood:  -15143.93
```

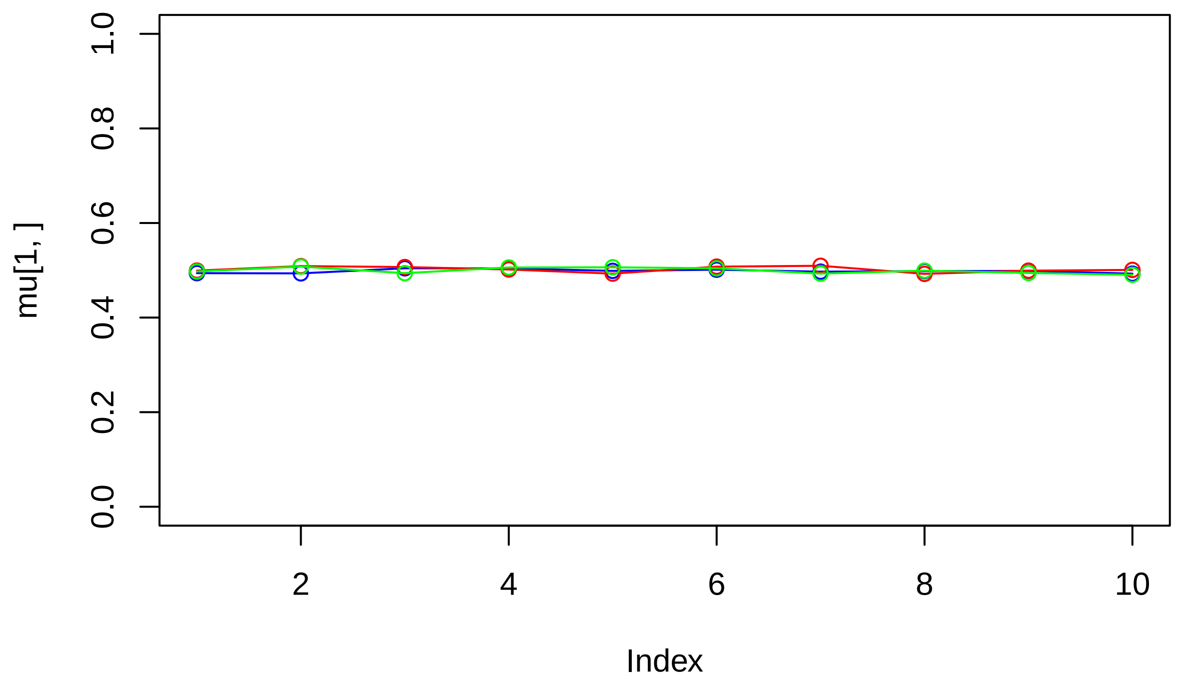


```
mixel_model_fun(3)
```

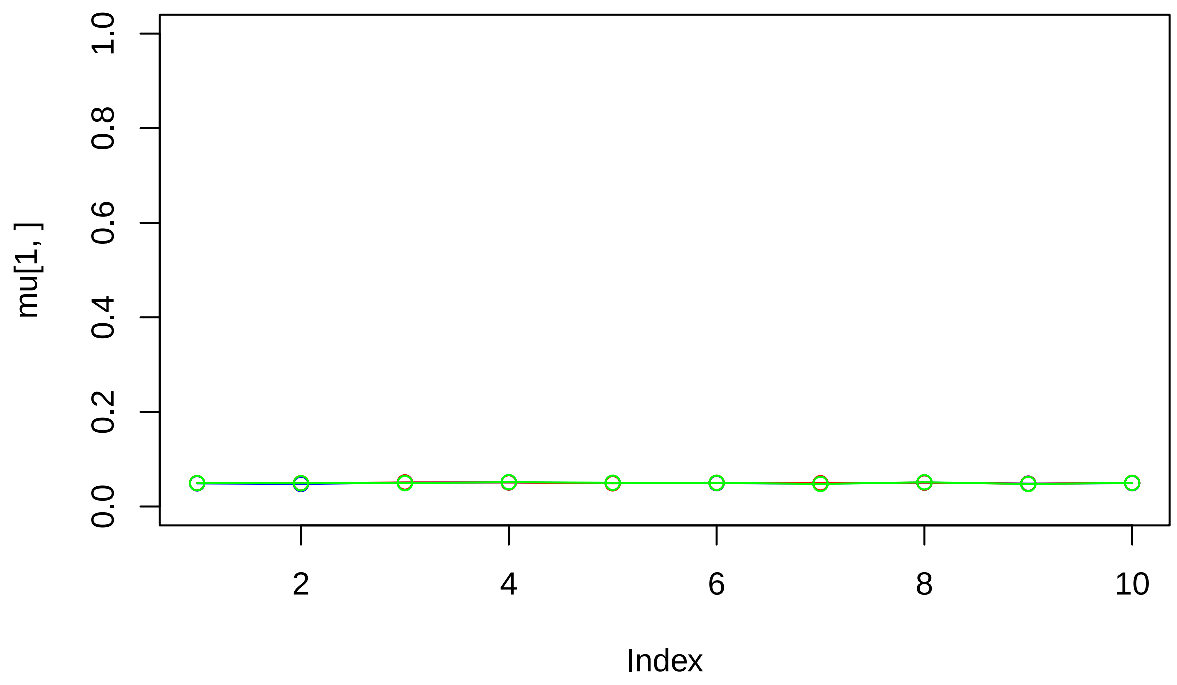
```
## Generate model when M= 3
```



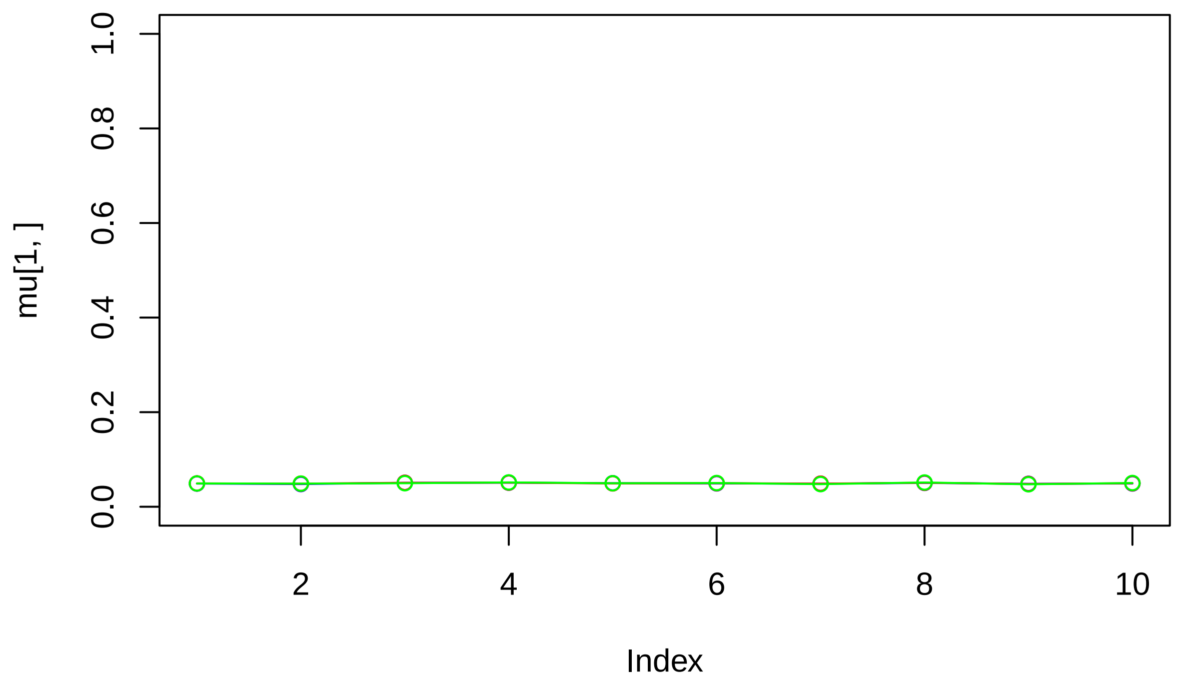
1.0
0.8
0.6
0.4
0.2
0.0
true_mu[1,]
2
4
6
8
10
Index



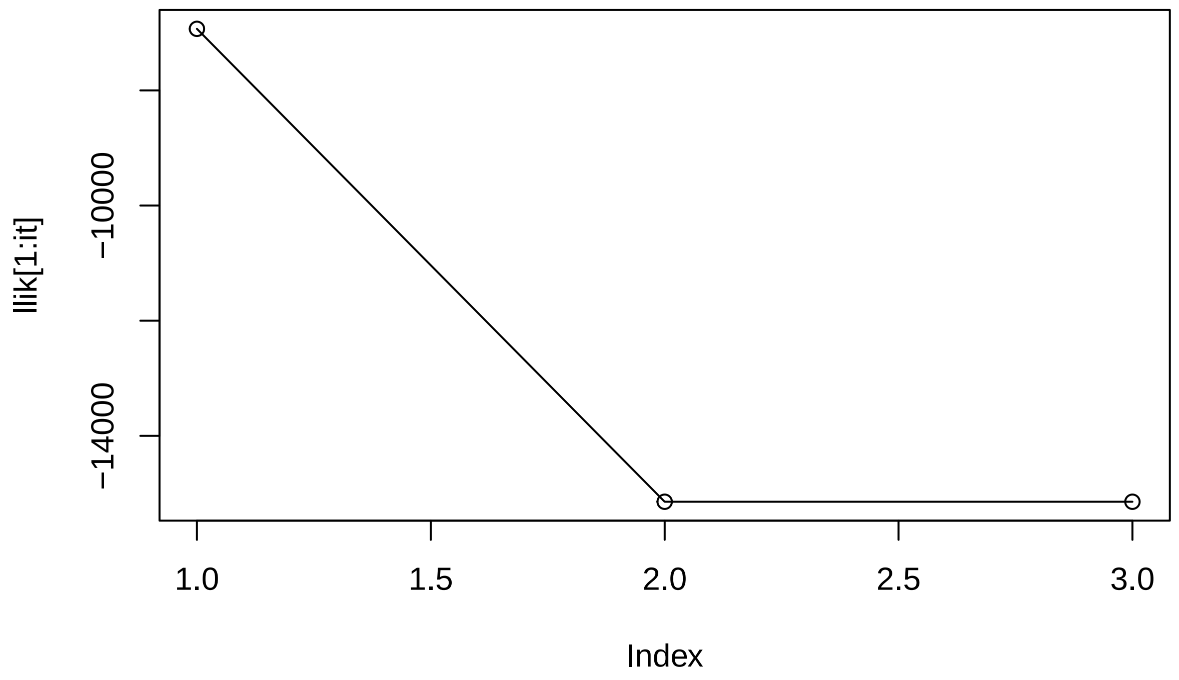
```
## iteration:  1 log likelihood:  -6931.482
```



```
## iteration:  2 log likelihood:  -15143.96
```

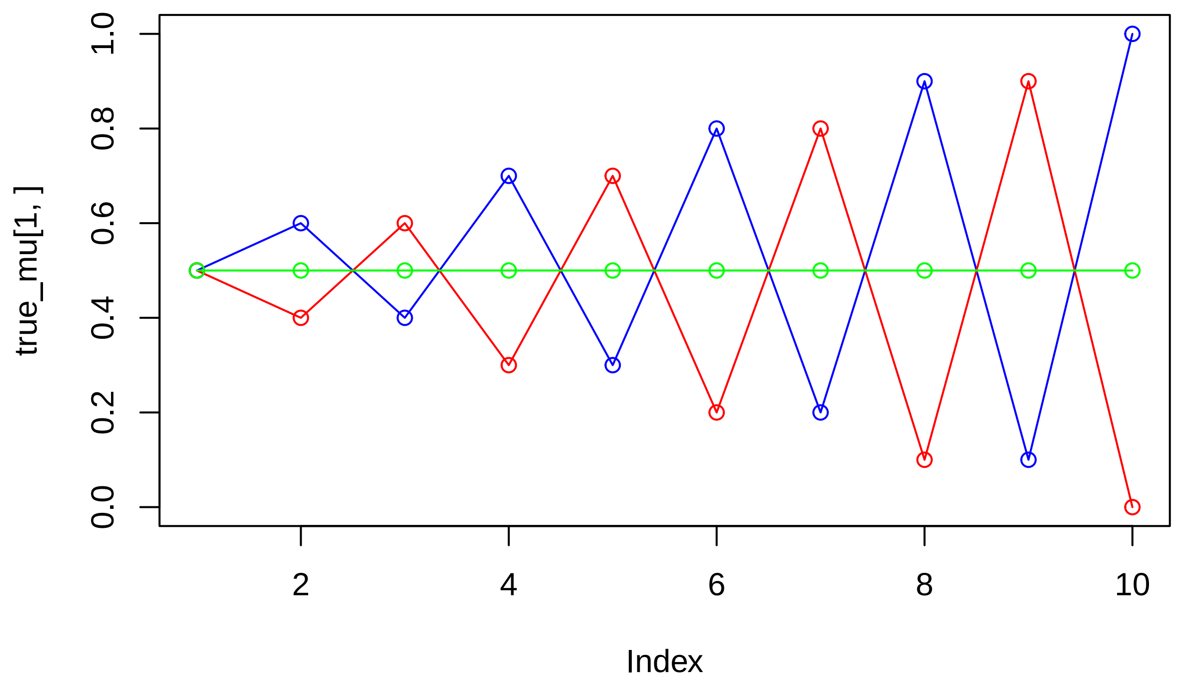


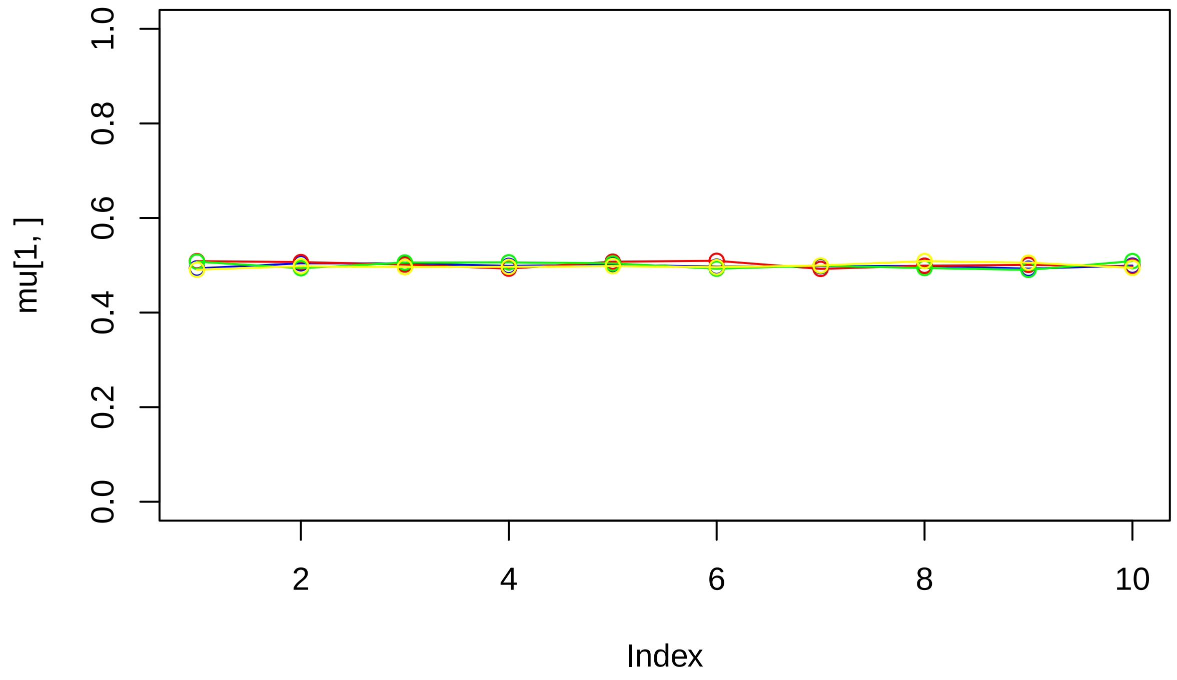
```
## iteration:  3 log likelihood:  -15143.93
```



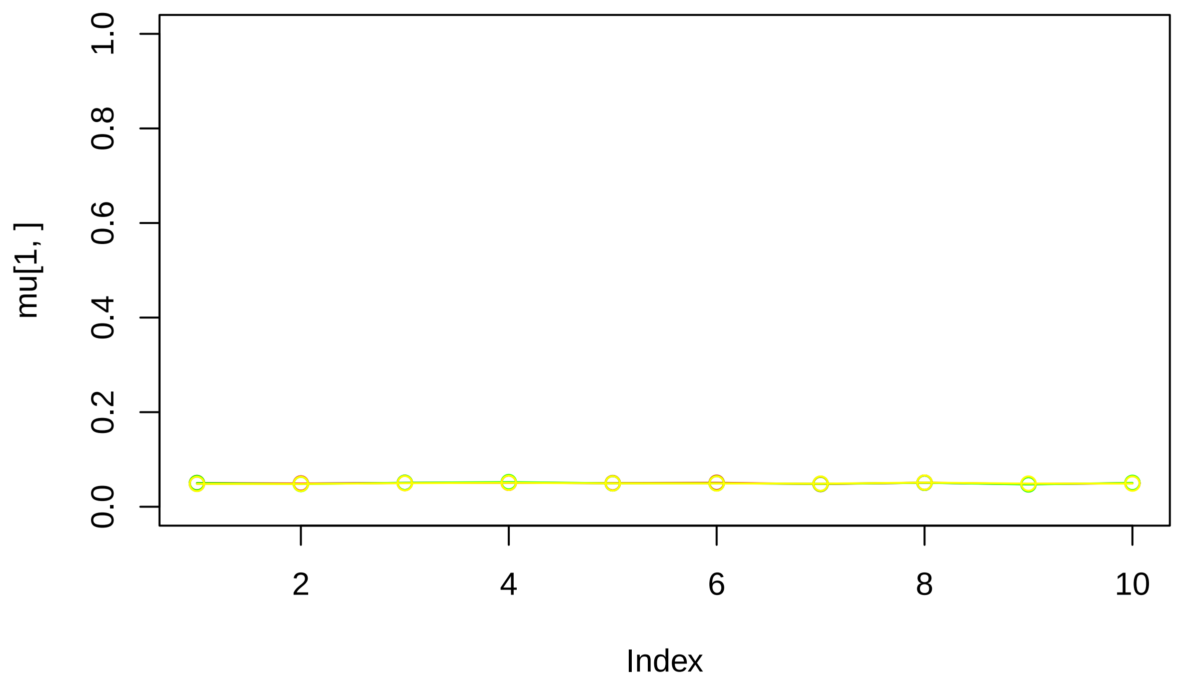
```r
mixel_model_fun(4)
```

```
## Generate model when M= 4
```

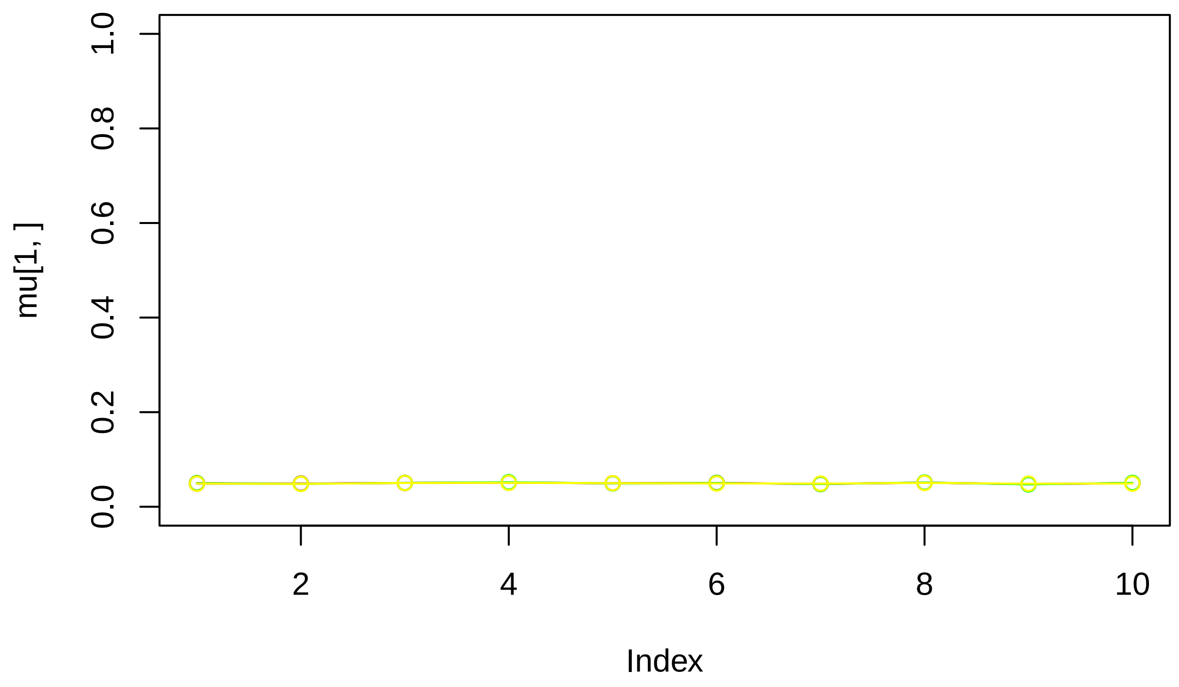

true_mu[1,]
Index
0.0
0.2
0.4
0.6
0.8
1.0
2
4
6
8
10



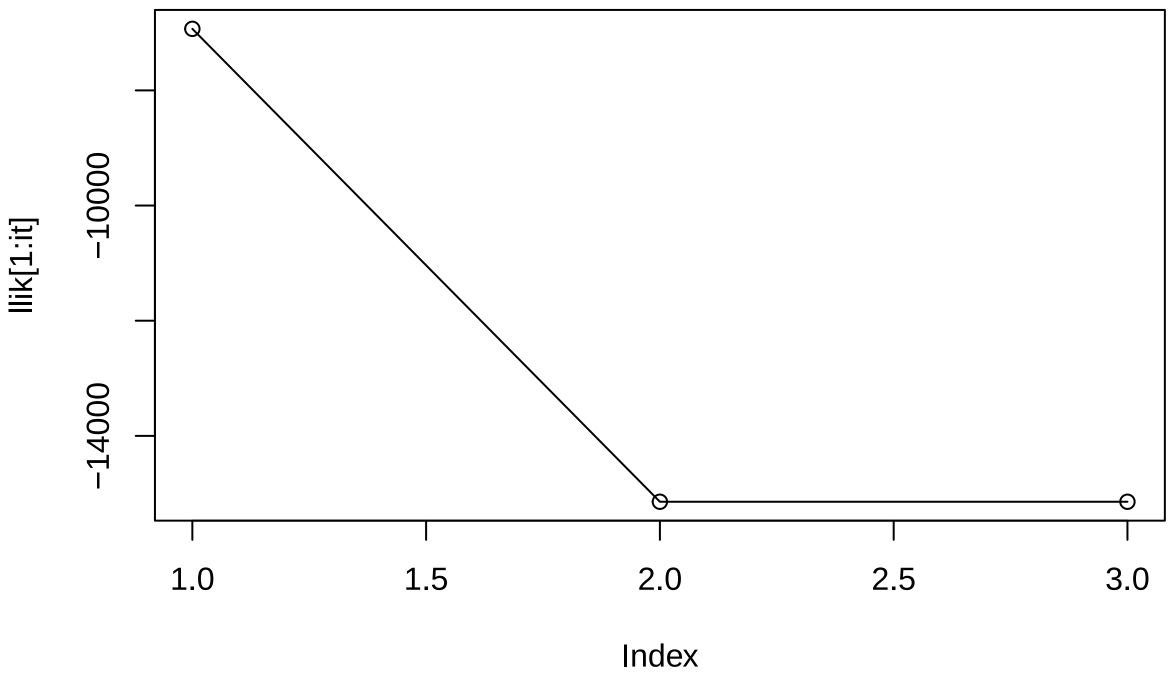
```
## iteration:  1 log likelihood:  -6931.372
```



```
## iteration:  2 log likelihood:  -15143.97
```



```
## iteration:  3 log likelihood:  -15143.91
```


−10000
−14000
llik[1:it]
1.0
1.5
2.0
2.5
3.0
Index

Appendix: All code for this report

```r
###########################  Init code For Assignment 1 ########################
rm(list = ls())
knitr::opts_chunk$set(echo = TRUE)
library(randomForest)
set.seed(1234)
###############  Assignment 1 Ensemble methods #################################

compare_method1 <- function(x1, x2) {
  return(x1 < x2)
}

compare_method2 <- function(x1, x2) {
  return(x1 < 0.5)
}

compare_method3 <- function(x1, x2) {
  return((x1 > 0.5 & x2 > 0.5) | (x1 < 0.5 & x2 < 0.5))
}

ensemble_method <- function(record_number, tree_number, node_size, compare_method) {
  # init data set (copy from pdf file)


  x1 <- runif(record_number)
  x2 <- runif(record_number)
  trdata <- cbind(x1, x2)
  y <- as.numeric(compare_method(x1, x2))
  trlabels <- as.factor(y)
  #plot(x1, x2, col = (y + 1))

  # training the model using random forest
  rf_model <- randomForest(trdata, trlabels, ntree = tree_number,
                           nodesize = node_size, keep.forest = TRUE)

  rf_pred <- predict(rf_model, newdata = trdata, class = "classification")

  # calculate the misclassification rate
  misclassification_rate <- sum(as.numeric(rf_pred != trlabels)) /
  length(trlabels)

  return (misclassification_rate)
}


################################################################################
# call the function
record_numbers <- c(100, 1000)
tree_numbers <- c(1, 10, 100)

node_size <- 25
# data record number = 100
result1 <- ensemble_method(record_numbers[1], tree_numbers[1], node_size,
                          compare_method1)
result2 <- ensemble_method(record_numbers[1], tree_numbers[2], node_size,
                          compare_method1)
result3 <- ensemble_method(record_numbers[1], tree_numbers[3], node_size,
                          compare_method1)

cat("The misclassification rate for 1 tree is: ", result1, "\n")
cat("The misclassification rate for 10 tree is: ", result2, "\n")
cat("The misclassification rate for 100 tree is: ", result3, "\n")

################################################################################
# When we set node_size = 25 and record_numbers = 100 and 1000 and check x1 < x2 ,
# and run it 1000 times

misclassification_rates_1.1 <- rep(0, 1000)
misclassification_rates_10.1 <- rep(0, 1000)
misclassification_rates_100.1 <- rep(0, 1000)

misclassification_rates_1.2 <- rep(0, 1000)
misclassification_rates_10.2 <- rep(0, 1000)
misclassification_rates_100.2 <- rep(0, 1000)

node_size = 25
for (i in 1:1000) {

  # data record number = 100
  misclassification_rates_1.1[i] <- ensemble_method(record_numbers[1],
                                                    tree_numbers[1], node_size,
                                                    compare_method1)
  misclassification_rates_10.1[i] <- ensemble_method(record_numbers[1],
                                                     tree_numbers[2], node_size,
                                                     compare_method1)
  misclassification_rates_100.1[i] <- ensemble_method(record_numbers[1],
                                                      tree_numbers[3], node_size,
                                                   compare_method1)
  # data record number = 1000
  misclassification_rates_1.2[i] <- ensemble_method(record_numbers[2],
                                                    tree_numbers[1], node_size,
                                                    compare_method1)
  misclassification_rates_10.2[i] <- ensemble_method(record_numbers[2],
                                                     tree_numbers[2], node_size,
                                                     compare_method1)
  misclassification_rates_100.2[i] <- ensemble_method(record_numbers[2],
                                                      tree_numbers[3], node_size,
                                                   compare_method1)
}

cat("The mean of misclassification rate for 1 tree with record number = 100 is: ",
    mean(misclassification_rates_1.1), "\n")

cat("The mean of misclassification rate for 10 tree with record number = 100 is: ",
    mean(misclassification_rates_10.1), "\n")
```

```r
cat("The mean of misclassification rate for 100 tree with record number = 100 is: ",
    mean(misclassification_rates_100.1), "\n")

cat("The var of misclassification rate for 1 tree with record number = 100 is: ",
    var(misclassification_rates_1.1), "\n")

cat("The var of misclassification rate for 10 tree with record number = 100 is: ",
    var(misclassification_rates_10.1), "\n")

cat("The var of misclassification rate for 100 tree with record number = 100 is: ",
    var(misclassification_rates_100.1), "\n")


cat("The mean of misclassification rate for 1 tree with record number = 1000 is: ",
    mean(misclassification_rates_1.2), "\n")

cat("The mean of misclassification rate for 10 tree with record number = 1000 is: ",
    mean(misclassification_rates_10.2), "\n")

cat("The mean of misclassification rate for 100 tree with record number = 1000 is: ",
    mean(misclassification_rates_100.2), "\n")

cat("The var of misclassification rate for 1 tree with record number = 1000 is: ",
    var(misclassification_rates_1.2), "\n")

cat("The var of misclassification rate for 10 tree with record number = 1000 is: ",
    var(misclassification_rates_10.2), "\n")

cat("The var of misclassification rate for 100 tree with record number = 1000 is: ",
    var(misclassification_rates_100.2), "\n")
################################################################################

misclassification_rates_1.1 <- rep(0, 1000)
misclassification_rates_10.1 <- rep(0, 1000)
misclassification_rates_100.1 <- rep(0, 1000)

misclassification_rates_1.2 <- rep(0, 1000)
misclassification_rates_10.2 <- rep(0, 1000)
misclassification_rates_100.2 <- rep(0, 1000)


for (i in 1:1000) {

  # data record number = 100
  misclassification_rates_1.1[i] <- ensemble_method(record_numbers[1],
                                                 tree_numbers[1], node_size,
                                                 compare_method2)
  misclassification_rates_10.1[i] <- ensemble_method(record_numbers[1],
                                                  tree_numbers[2], node_size,
                                                  compare_method2)
  misclassification_rates_100.1[i] <- ensemble_method(record_numbers[1],
                                                   tree_numbers[3], node_size,
                                                   compare_method2)
  # data record number = 1000
  misclassification_rates_1.2[i] <- ensemble_method(record_numbers[2],
                                                      tree_numbers[1], node_size,
                                                      compare_method2)
  misclassification_rates_10.2[i] <- ensemble_method(record_numbers[2],
                                                       tree_numbers[2], node_size,
                                                       compare_method2)
  misclassification_rates_100.2[i] <- ensemble_method(record_numbers[2],
                                                        tree_numbers[3], node_size,
                                                        compare_method2)

}

cat("The mean of misclassification rate for 1 tree with record number = 100 is: ",
    mean(misclassification_rates_1.1), "\n")

cat("The mean of misclassification rate for 10 tree with record number = 100 is: ",
    mean(misclassification_rates_10.1), "\n")

cat("The mean of misclassification rate for 100 tree with record number = 100 is: ",
    mean(misclassification_rates_100.1), "\n")

cat("The var of misclassification rate for 1 tree with record number = 100 is: ",
    var(misclassification_rates_1.1), "\n")

cat("The var of misclassification rate for 10 tree with record number = 100 is: ",
    var(misclassification_rates_10.1), "\n")

cat("The var of misclassification rate for 100 tree with record number = 100 is: ",
    var(misclassification_rates_100.1), "\n")



cat("The mean of misclassification rate for 1 tree with record number = 1000 is: ",
    mean(misclassification_rates_1.2), "\n")

cat("The mean of misclassification rate for 10 tree with record number = 1000 is: ",
    mean(misclassification_rates_10.2), "\n")

cat("The mean of misclassification rate for 100 tree with record number = 1000 is: ",
    mean(misclassification_rates_100.2), "\n")

cat("The var of misclassification rate for 1 tree with record number = 1000 is: ",
    var(misclassification_rates_1.2), "\n")

cat("The var of misclassification rate for 10 tree with record number = 1000 is: ",
    var(misclassification_rates_10.2), "\n")

cat("The var of misclassification rate for 100 tree with record number = 1000 is: ",
    var(misclassification_rates_100.2), "\n")

################################################################################
# When we set node_size = 25 and record_numbers = 100 and check
```

```r
#((x1 > 0.5 & x2 > 0.5) | (x1 < 0.5 & x2 < 0.5)) , and run  it 1000 times

misclassification_rates_1.1 <- rep(0, 1000)
misclassification_rates_10.1 <- rep(0, 1000)
misclassification_rates_100.1 <- rep(0, 1000)

misclassification_rates_1.2 <- rep(0, 1000)
misclassification_rates_10.2 <- rep(0, 1000)
misclassification_rates_100.2 <- rep(0, 1000)

node_size <- 12
# data record number = 100
for (i in 1:1000) {

  # data record number = 100
  misclassification_rates_1.1[i] <- ensemble_method(record_numbers[1],
                                                     tree_numbers[1], node_size,
                                                     compare_method3)
  misclassification_rates_10.1[i] <- ensemble_method(record_numbers[1],
                                                      tree_numbers[2], node_size,
                                                      compare_method3)
  misclassification_rates_100.1[i] <- ensemble_method(record_numbers[1],
                                                       tree_numbers[3], node_size,
                                                       compare_method3)

  # data record number = 1000
  misclassification_rates_1.2[i] <- ensemble_method(record_numbers[2],
                                                     tree_numbers[1], node_size,
                                                     compare_method3)
  misclassification_rates_10.2[i] <- ensemble_method(record_numbers[2],
                                                      tree_numbers[2], node_size,
                                                      compare_method3)
  misclassification_rates_100.2[i] <- ensemble_method(record_numbers[2],
                                                       tree_numbers[3], node_size,
                                                       compare_method3)
}

cat("The mean of misclassification rate for 1 tree with record number = 100 is: ",
    mean(misclassification_rates_1.1), "\n")

cat("The mean of misclassification rate for 10 tree with record number = 100 is: ",
    mean(misclassification_rates_10.1), "\n")

cat("The mean of misclassification rate for 100 tree with record number = 100 is: ",
    mean(misclassification_rates_100.1), "\n")

cat("The var of misclassification rate for 1 tree with record number = 100 is: ",
    var(misclassification_rates_1.1), "\n")

cat("The var of misclassification rate for 10 tree with record number = 100 is: ",
    var(misclassification_rates_10.1), "\n")

cat("The var of misclassification rate for 100 tree with record number = 100 is: ",
    var(misclassification_rates_100.1), "\n")



cat("The mean of misclassification rate for 1 tree with record number = 1000 is: ",
    mean(misclassification_rates_1.2), "\n")

cat("The mean of misclassification rate for 10 tree with record number = 1000 is: ",
    mean(misclassification_rates_10.2), "\n")

cat("The mean of misclassification rate for 100 tree with record number = 1000 is: ",
    mean(misclassification_rates_100.2), "\n")

cat("The var of misclassification rate for 1 tree with record number = 1000 is: ",
    var(misclassification_rates_1.2), "\n")

cat("The var of misclassification rate for 10 tree with record number = 1000 is: ",
    var(misclassification_rates_10.2), "\n")

cat("The var of misclassification rate for 100 tree with record number = 1000 is: ",
    var(misclassification_rates_100.2), "\n")

################################################################################

###########################  Init code For Assignment 2 ########################
rm(list = ls())
knitr::opts_chunk$set(echo = TRUE)
######################## Assignment 2 MIXTURE MODELS ###########################

mixel_model_fun <- function(M = 3){
    cat("Generate model when M=",M)

    # init data set (copy from pdf file)
    set.seed(1234567890)

    # max number of EM iterations
    max_it <- 5

    # min change in log lik between two consecutive iterations
    min_change <- 0.1

    # number of training points
    n <- 1000

    # number of dimensions
    D <- 10

    # training data
    x <- matrix(nrow = n, ncol = D)

    # true mixing coefficients
    true_pi <- vector(length = 3)
```

```
# true conditional distributions
true_mu <- matrix(nrow = 3, ncol = D)

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

# number of clusters, set by function parameter
# M = 3

w <- matrix(nrow = n, ncol = M) # weights
pi <- vector(length = M) # mixing coefficients
mu <- matrix(nrow = M, ncol = D) # conditional distributions
llik <- vector(length = max_it) # log likelihood of the EM iterations

# Random initialization of the parameters
pi <- runif(M, 0.49, 0.51)
pi <- pi / sum(pi)
for (m in 1:M) {
mu[m, ] <- 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")
if (M > 2){
    points(mu[3,], type="o", col="green")
}
if (M > 3){
    points(mu[4,], type="o", col="yellow")
}
#Sys.sleep(2)

# E-step: Computation of the weights
for (i in 1:n) {
    for (m in 1:M) {
        w[i, m] <- pi[m] * prod((mu[m,] ^ x[i,]) * ((1 - mu[m,]) ^ (1 - x[i,])))
        }
        w[i,] <- w[i,] / sum(w[i,])
    }

    #Log likelihood computation.
    llik[it] <- sum(log(apply(x, 1,
        function(xi) sum(pi * apply(mu, 1, function(mui)
                    prod(mui^xi * (1 - mui)^(1 - xi)))))))

    cat("iteration: ", it, "log likelihood: ", llik[it], "\n")
    flush.console()

    # Stop if the lok likelihood has not changed significantly
    if (it > 1 && abs(llik[it] - llik[it - 1]) < min_change) {
        break
    }

    #M-step: ML parameter estimation from the data and weights
    for (m in 1:M) {
        pi[m] <- sum(w[, m]) / n
        mu[m,] <- colSums(x * w[, m]) / (sum(w[, m]) * D)
    }
    }

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

mixel_model_fun(2)
mixel_model_fun(3)
mixel_model_fun(4)
```