

Machine Learning Computer Lab 1 Block2 (Group A7)

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

2024-01-15

Statement of Contribution

For Lab 1 Block 2, we decided to split the two assignments equally, Qinyuan and Bozzoli completed Assignment 1, Satya and Bozzoli completed Assignment 2, after which, for verification's sake, we completed each other's assignments as well and validated our findings. The report was also compiled by three of us, with each handling their respective assignments.

Assignment 1: ENSEMBLE METHODS

Answer:

Task 1: Learning Random forest with ntree= 1,10 and 100 trees, nodesize = 25, condition (x1 < x2) and keep.forest = TRUE

We can find from the result that as the number of trees grows, the misclassification rate will decrease. This is because the more trees we have, the more robust the model will be.

```
##   tree_number misclassification_rate
## 1           1                   0.15
## 2          10                   0.25
## 3         100                   0.09
```

Task 2: Learning Random forest with ntree= 1,10 and 100 trees, nodesize = 25, condition (x1 < x2) and keep.forest = TRUE, run 1000 times

We can find from the result that as the number of trees grows, the mean and variance of the misclassification rate will decrease. When the record number grows, the mean and variance of the misclassification rate will also decrease.

```
##   tree_number record_number misclassification_mean misclassification_var
## 1           1           100           0.203970           0.0042127519
## 2           10           100           0.135020           0.0020354350
## 3          100           100           0.109540           0.0016578462
## 4            1          1000           0.204998           0.0030226987
## 5            10          1000           0.136597           0.0009431718
## 6            100          1000           0.111475           0.0008529964
```

Task 3: Learning Random forest with ntree= 1,10 and 100 trees, nodesize = 25, condition (x1 < 0.5) and keep.forest = TRUE, run 1000 times

Same as Task 2. but we also find that all the data, including mean and variance, are smaller than what we get when using condition (x1 < x2).

```
##   tree_number record_number misclassification_mean misclassification_var
```

## 1	1	100	0.098150	1.919527e-02
## 2	10	100	0.014580	8.160396e-04
## 3	100	100	0.005840	1.035980e-04
## 4	1	1000	0.095801	1.996155e-02
## 5	10	1000	0.015724	6.675394e-04
## 6	100	1000	0.005676	4.112014e-05

Task 4: Learning Random forest with ntree= 1,10 and 100 trees, nodesize = 12, condition ((x1 > 0.5 & x2 > 0.5) | (x1 < 0.5 & x2 < 0.5)) and keep.forest = TRUE, run 1000 times

Same as Task 2. We find that this condition is almost the same as condition (x1 < x2).

But we also find that changing the node size to 12 will get a better result than node size = 25.

node_size = 25

##	tree_number	record_number	misclassification_mean	misclassification_var
## 1	1	100	0.377370	0.014077260
## 2	10	100	0.289730	0.008444872
## 3	100	100	0.242210	0.006342558
## 4	1	1000	0.380758	0.011794632
## 5	10	1000	0.288158	0.006376237
## 6	100	1000	0.246760	0.004827270

node_size = 12

##	tree_number	record_number	misclassification_mean	misclassification_var
## 1	1	100	0.242580	0.013738682
## 2	10	100	0.116010	0.003878058
## 3	100	100	0.073270	0.001932339
## 4	1	1000	0.245587	0.013374867
## 5	10	1000	0.115465	0.002809290
## 6	100	1000	0.073213	0.001391795

Task 5: What happens with the mean error rate when the number of trees in the random forest grows? Why?

According to the result, we can say that as the number of trees in the random forest gets larger, the mean value of classification will decrease.

This is because more trees will help to increase the robustness of the noisy data and increase stability.

As discussed above, decreasing the node size will increase the tree size which yields better results.

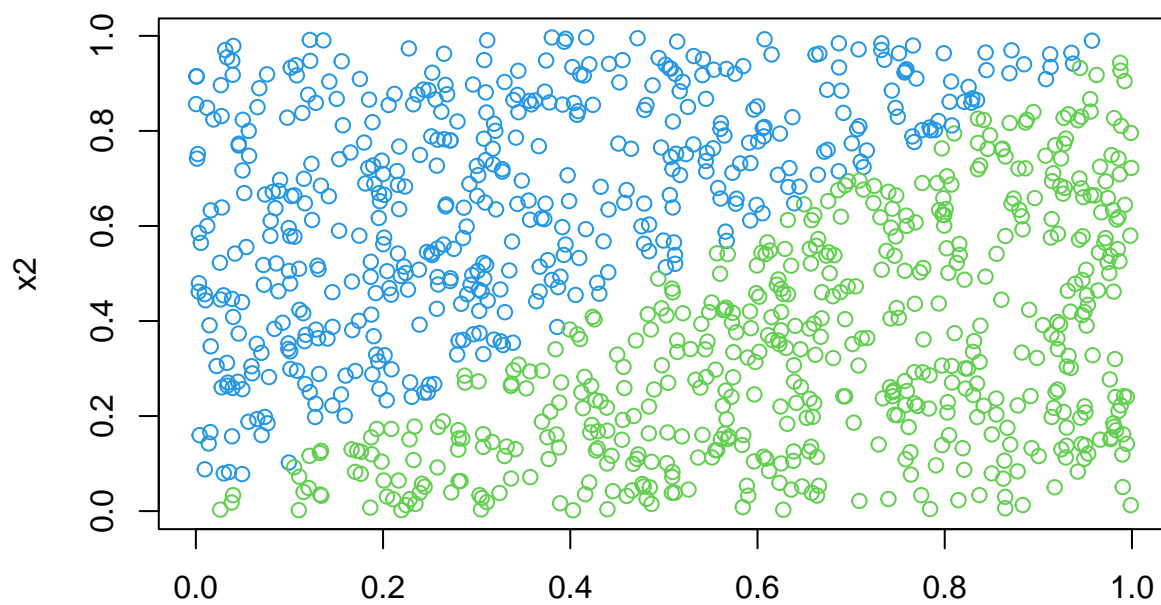
Task 6: The third dataset represents a slightly more complicated classification problem than the first one. Still, you should get better performance for it when using sufficient trees in the random forest. Explain why you get better performance

By observing the graph, we can find that the third dataset is more complicated than the first one.

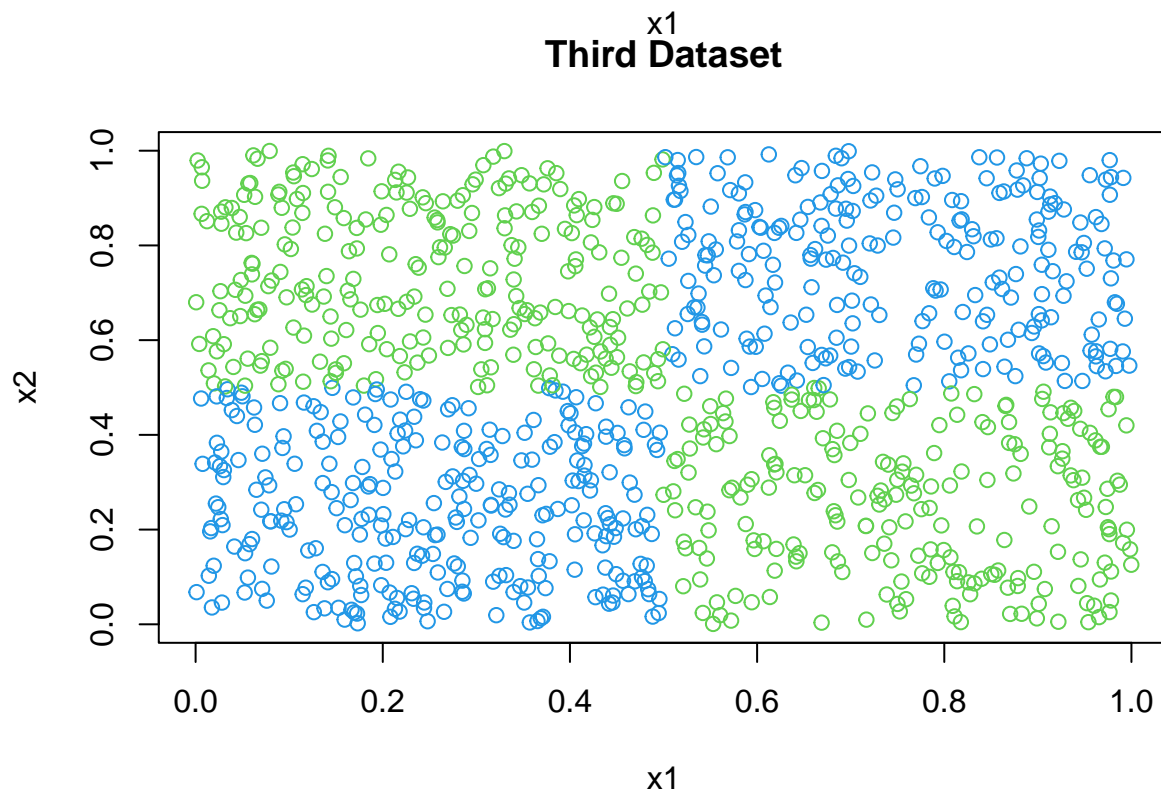
For real problems, we can not use a linear decision boundary to classify the data directly. So it's better to use the third dataset to train the model.

Meanwhile, we can find that the misclassification rate of the third dataset is greater than the first one. Since the model gets complicated, we need to use a sufficient amount of trees to get better performance and also prevent overfitting.

First Dataset



Third Dataset



Assignment 2: MIXTURE MODELS

Answer:

According to the question, bernoulli mixture model is:

$$p(x) = \sum_{m=1}^M \pi_m \text{Bern}(x|\mu_m)$$

where

$$\text{Bern}(x|\mu_m) = \prod_{d=1}^D \mu_{m,d}^{x_d} (1 - \mu_{m,d})^{1-x_d}$$

And log-likelihood is:

$$\sum_{i=1}^n \log(p(x_i))$$

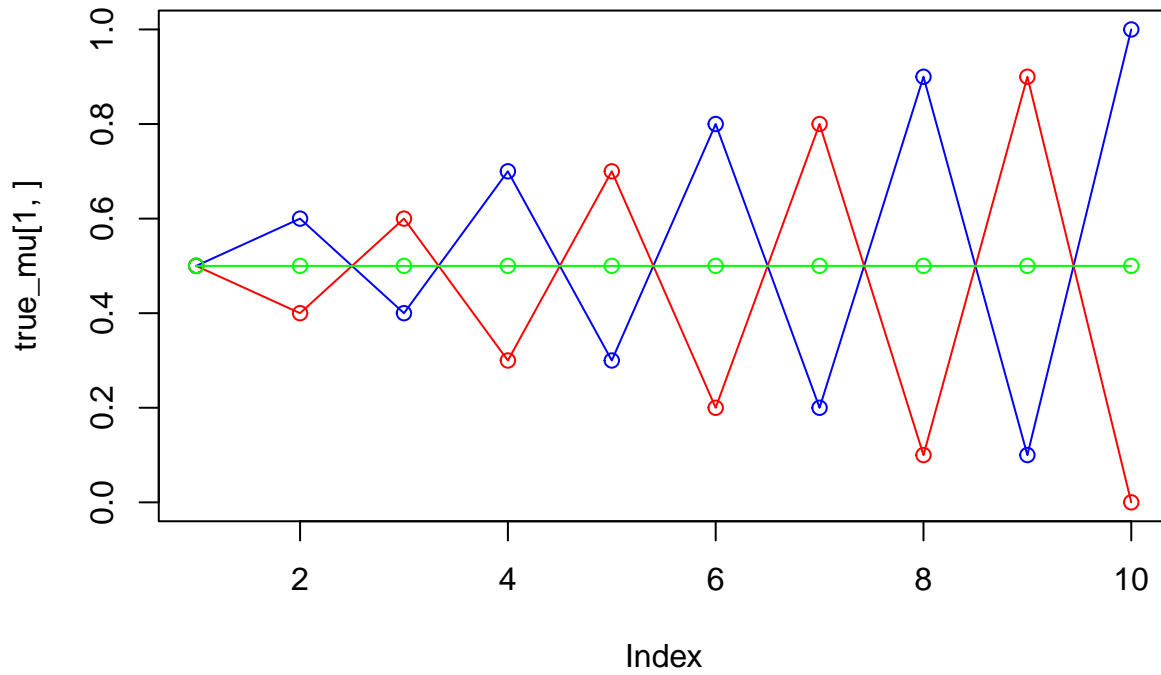
Derivative of log-likelihood is:

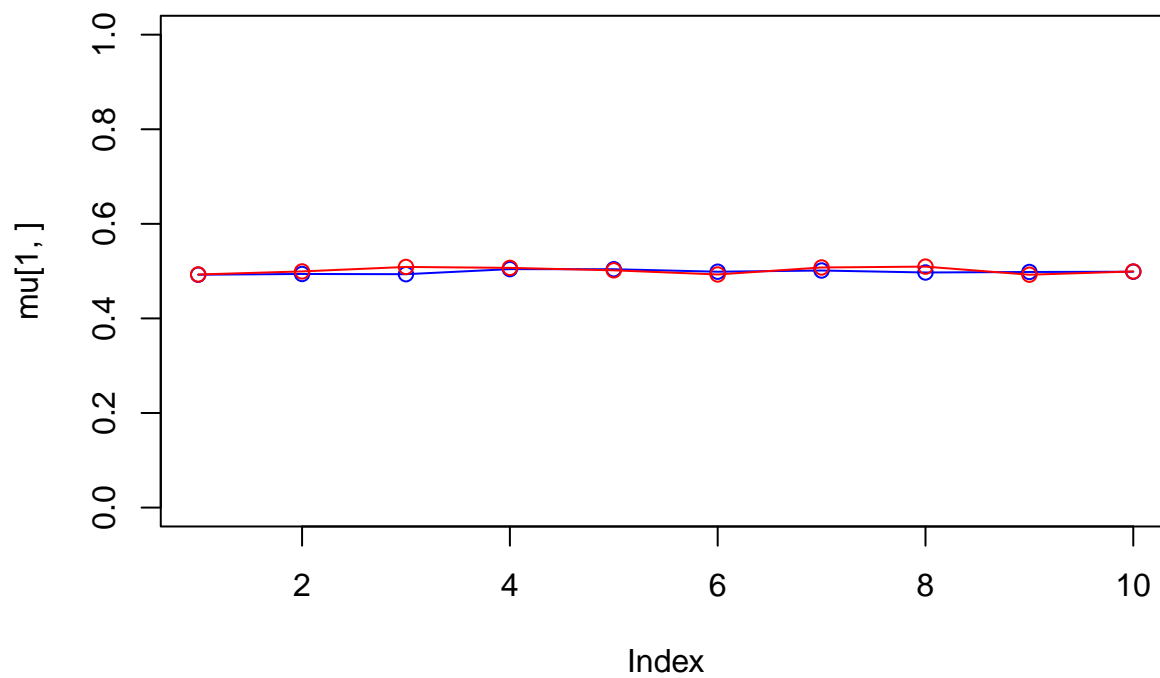
\$\$

\$\$

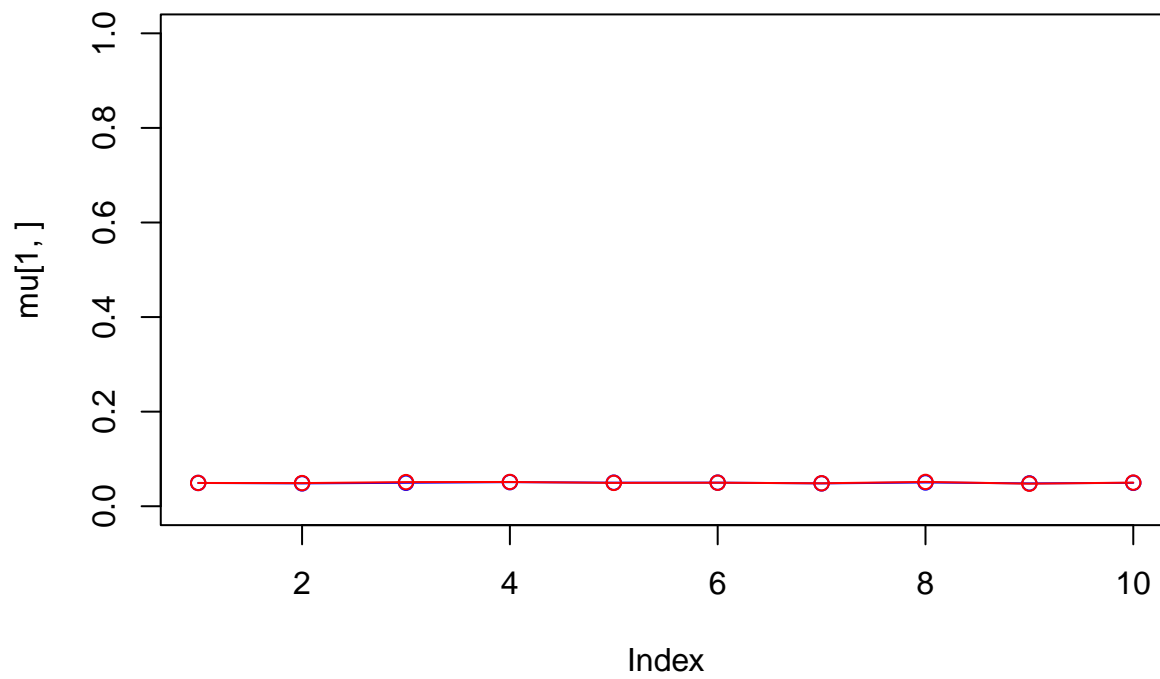
According to the output, when we change M value to 2,3 and 4, the output graph do not have significant change.

Generate model when M= 2

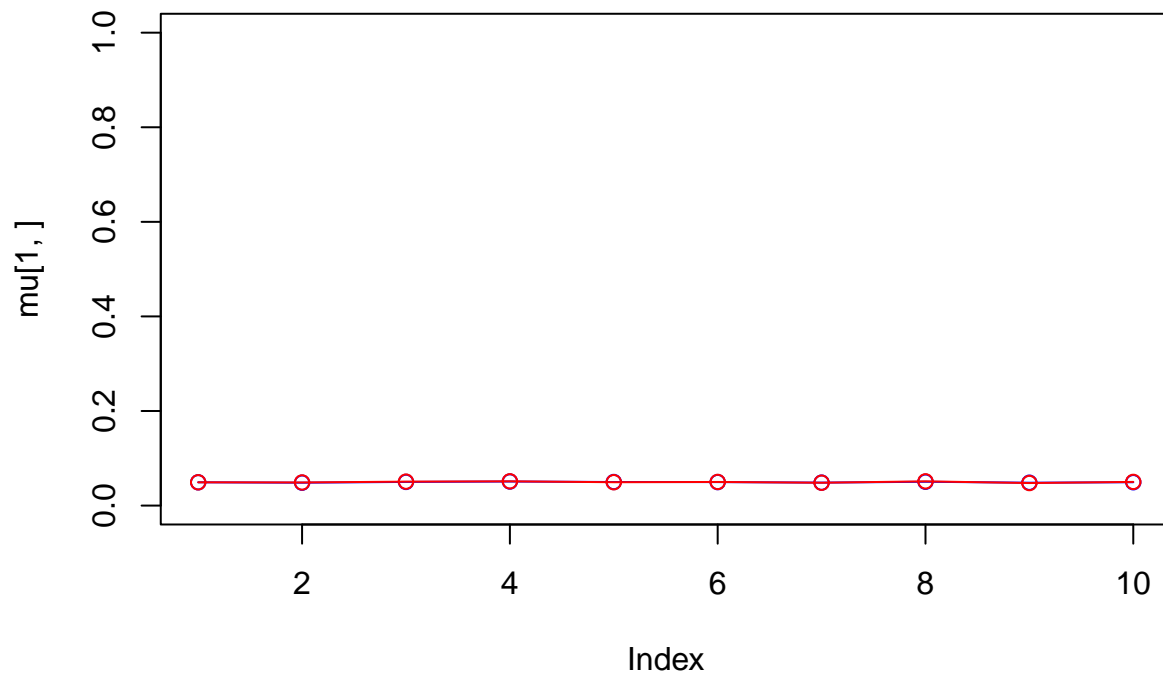




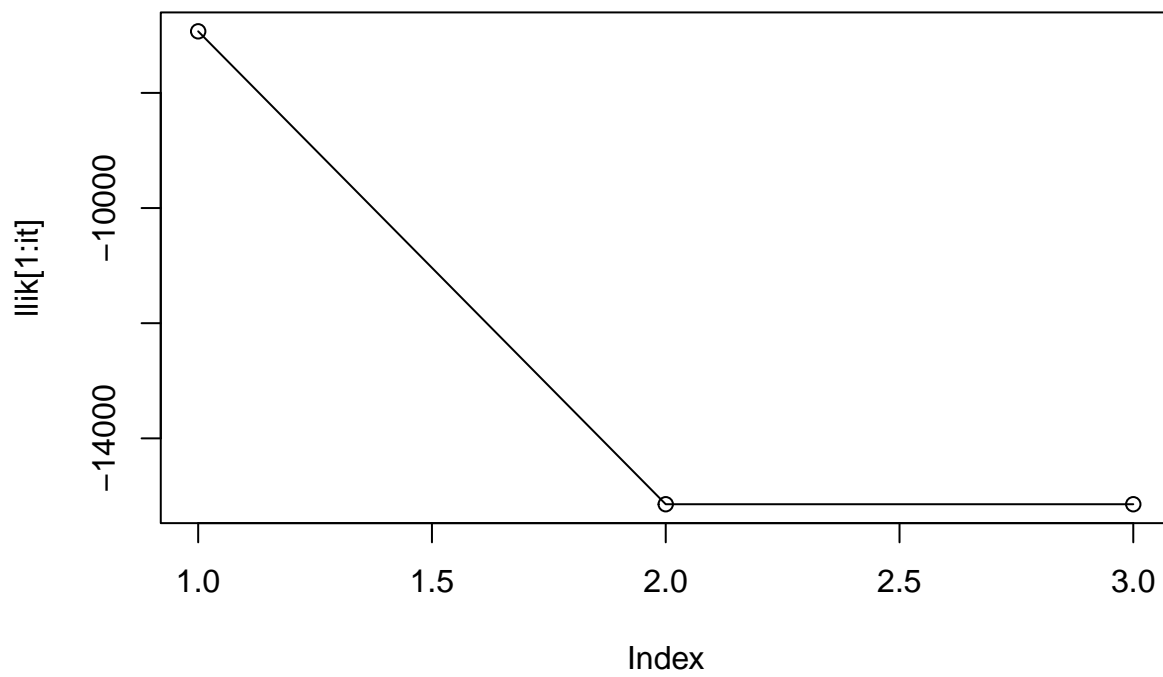
iteration: 1 log likelihood: -6930.975



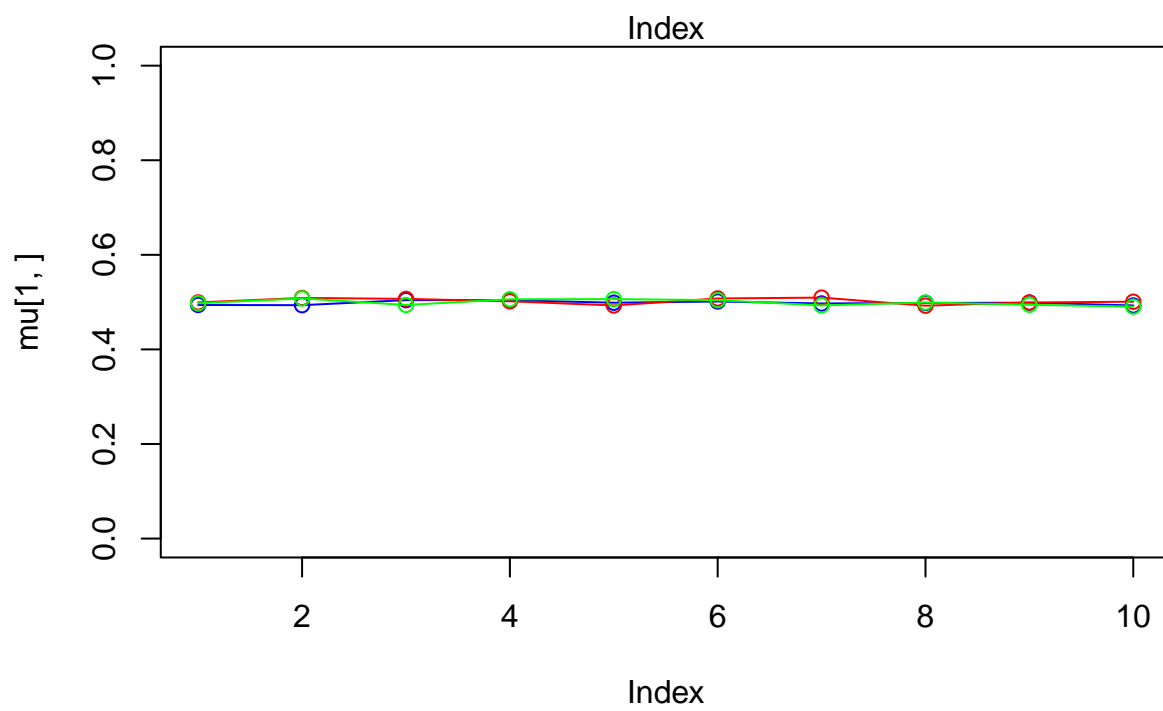
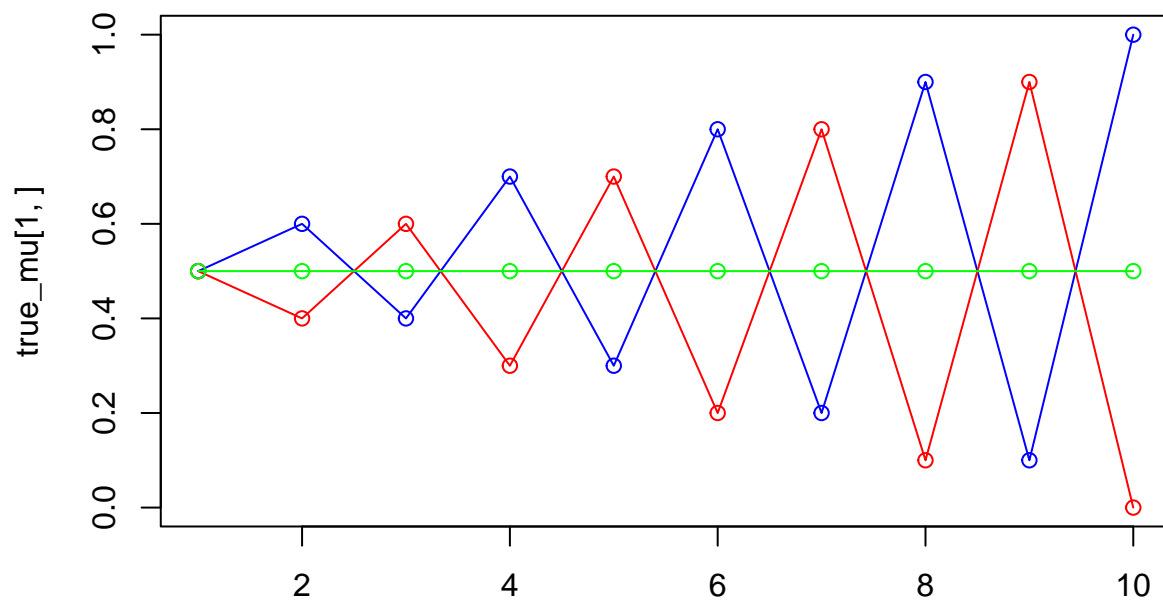
iteration: 2 log likelihood: -15143.91



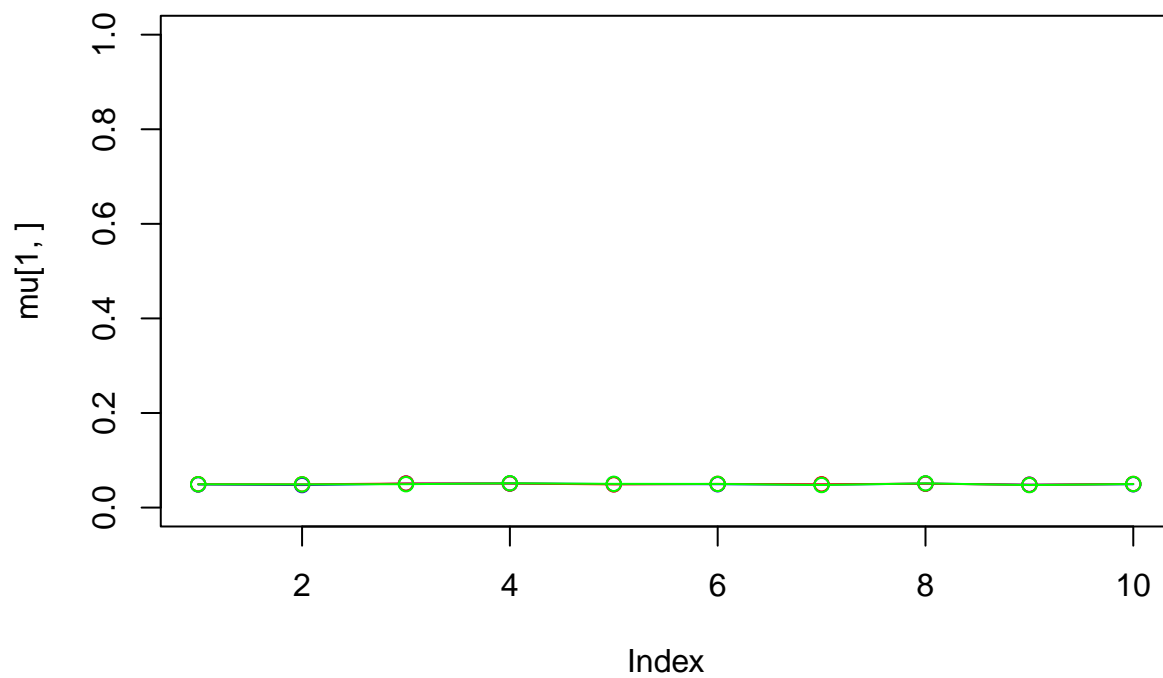
```
## iteration: 3 log likelihood: -15143.93
```



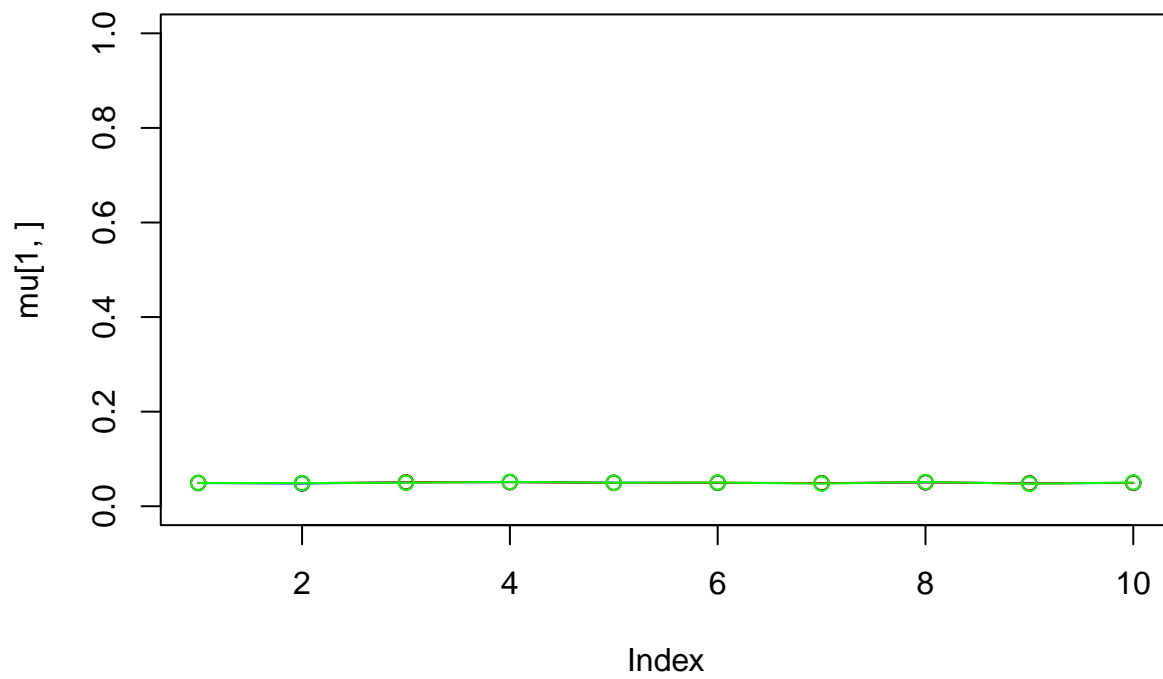
```
## Generate model when M= 3
```



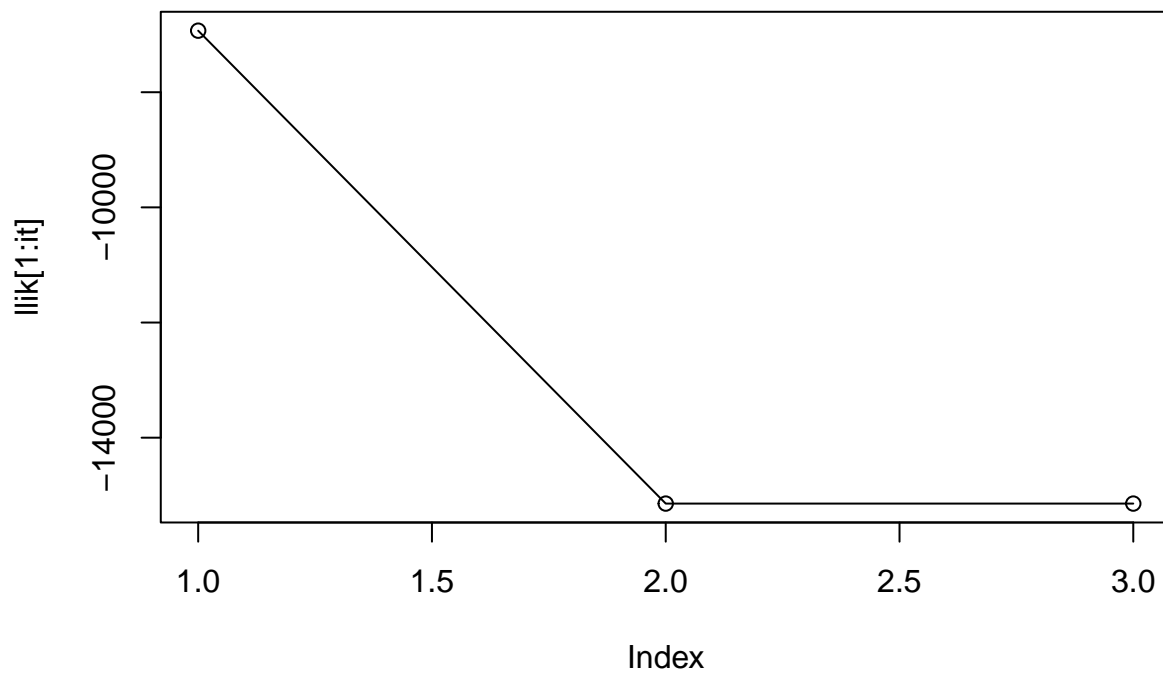
iteration: 1 log likelihood: -6931.482



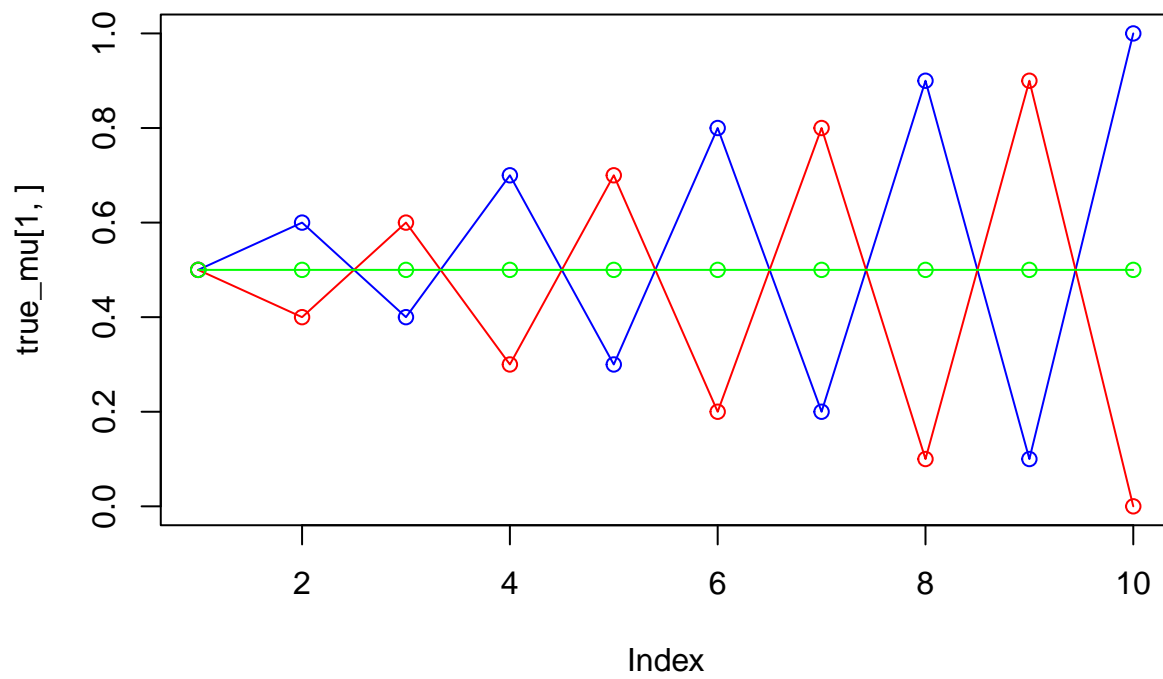
iteration: 2 log likelihood: -15143.96

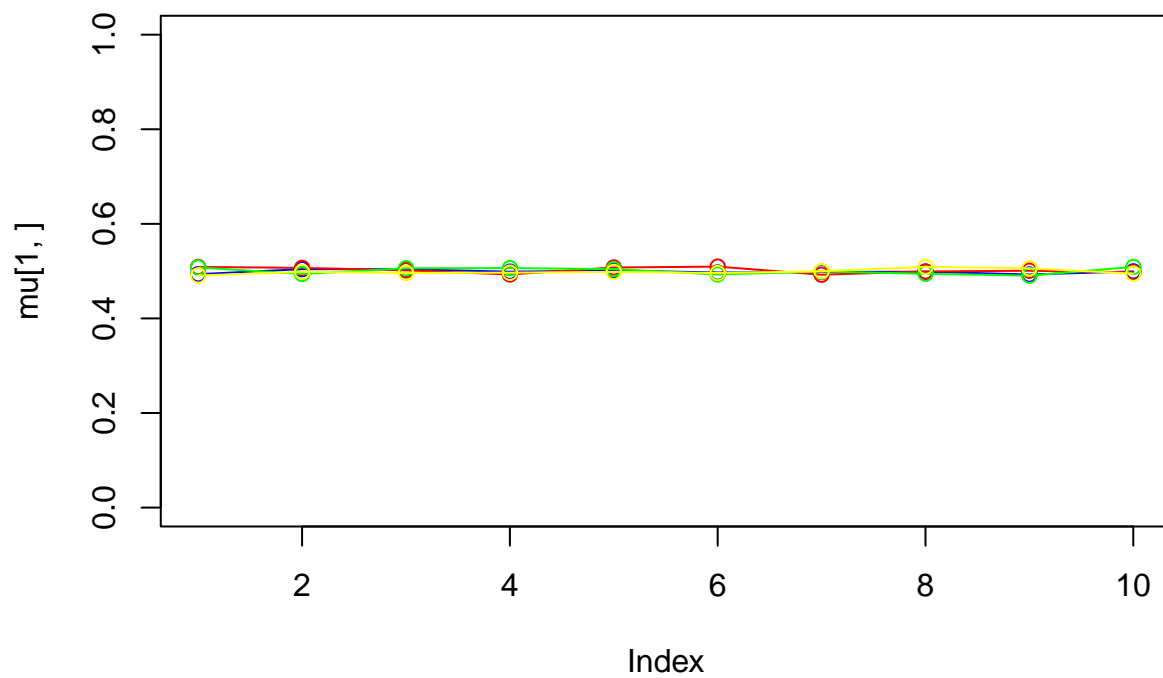


iteration: 3 log likelihood: -15143.93

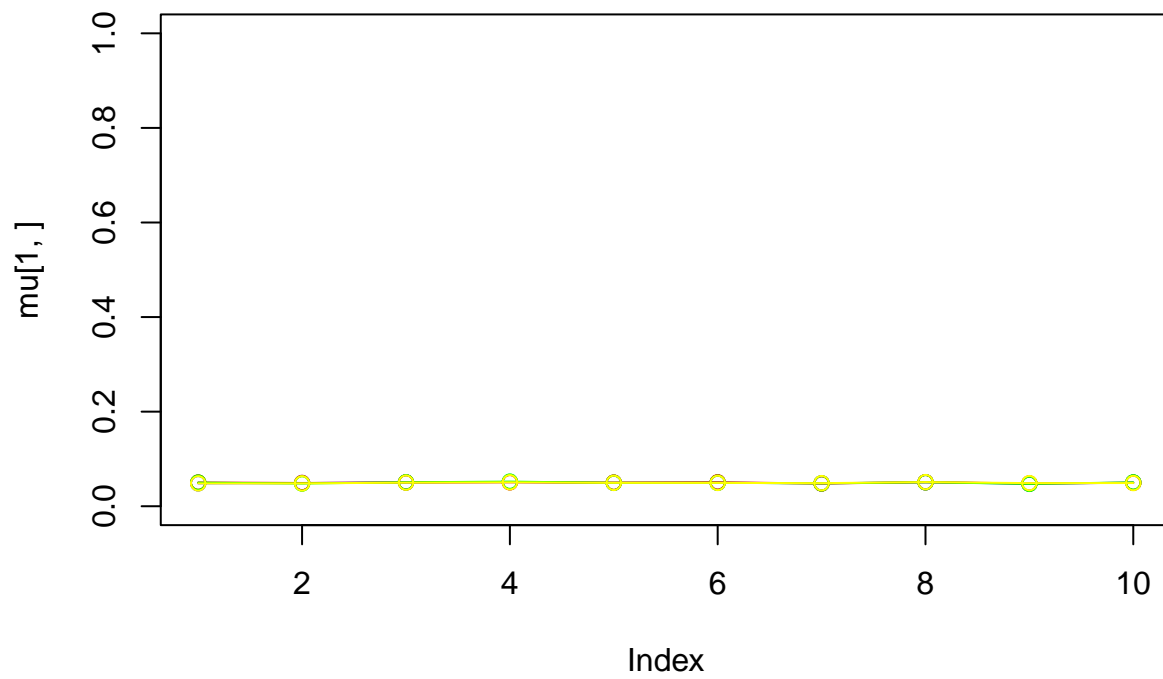


Generate model when M= 4

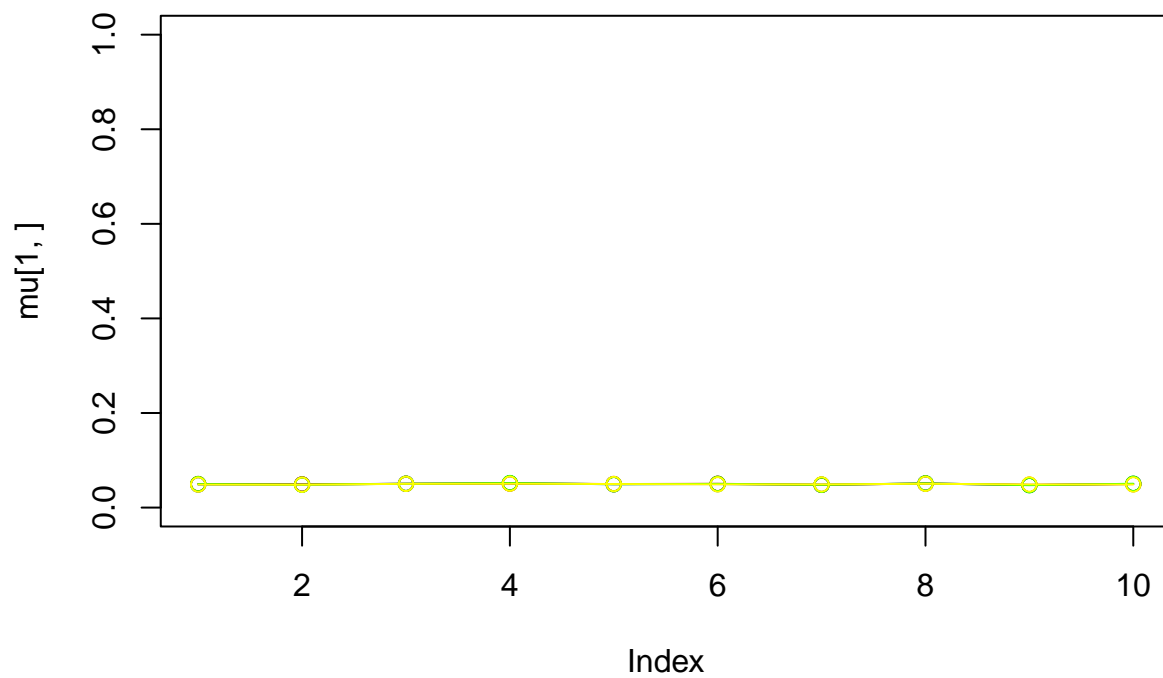




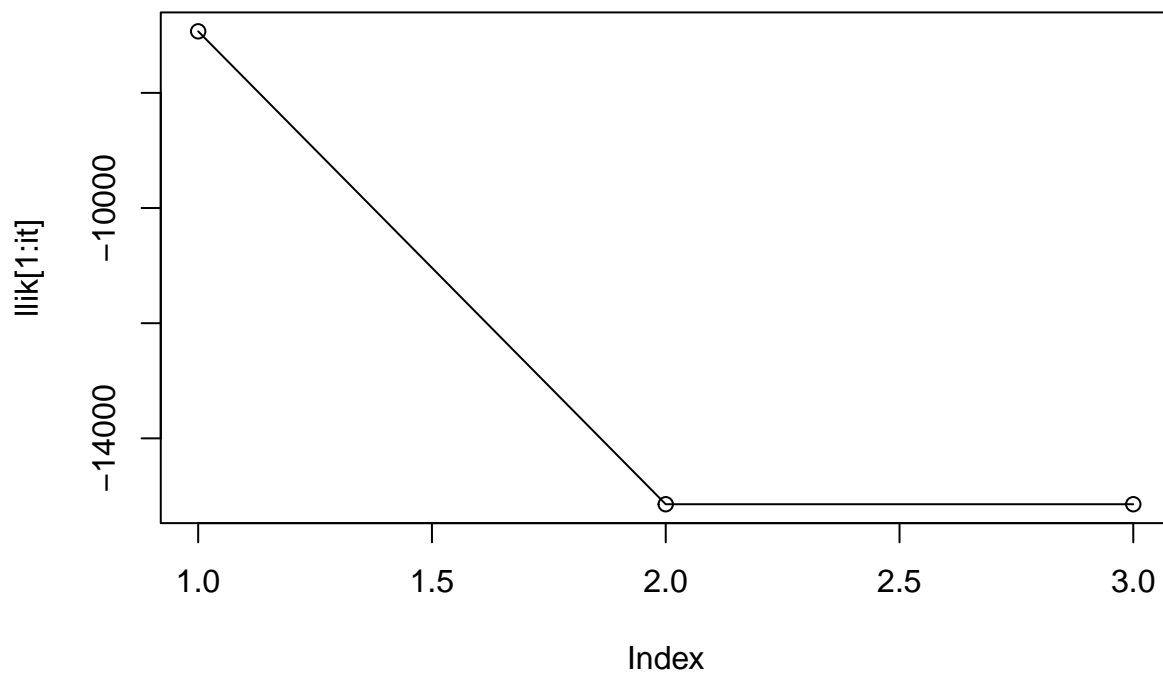
iteration: 1 log likelihood: -6931.372



iteration: 2 log likelihood: -15143.97



iteration: 3 log likelihood: -15143.91



Appendix: All code for this report

```
##### 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(100)
  x2 <- runif(100)

  trdata <- cbind(x1, x2)
  y <- as.numeric(compare_method(x1, x2))
  trlabels <- as.factor(y)
  trdata <- cbind(trdata, trlabels)

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

  x1 <- runif(record_number)
  x2 <- runif(record_number)
  tedata <- cbind(x1, x2)
  y <- as.numeric(compare_method(x1, x2))
  telabels <- as.factor(y)

  rf_pred <- predict(rf_model, newdata = tedata, type = "class")

  # calculate the misclassification rate
  misclassification_rate <- 1- sum(diag(table(rf_pred ,telabels)))/length(rf_pred)

  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)

misclassification_df <- data.frame(matrix(ncol = 2, nrow=0))
colnames(misclassification_df) <- c("tree_number", "misclassification_rate")

misclassification_df <- rbind(misclassification_df, data.frame("tree_number" = tree_numbers[1], "misclassification_rate" = result1))
misclassification_df <- rbind(misclassification_df, data.frame("tree_number" = tree_numbers[2], "misclassification_rate" = result2))
misclassification_df <- rbind(misclassification_df, data.frame("tree_number" = tree_numbers[3], "misclassification_rate" = result3))

misclassification_df
#####
# When we set node_size = 25 and record_numbers = 100 and 1000 and check x1 < x2 ,and tree_num = 1,10,100
# 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)
}
```

```

                                compare_method1)
misclassification_rates_100.2[i] <- ensemble_method(record_numbers[2],
                                                tree_numbers[3], node_size,
                                                compare_method1)
}

misclassification_mean_var_df <- data.frame(matrix(ncol = 4, nrow=0))
colnames(misclassification_mean_var_df) <- c("tree_number", "record_number", "misclassification_mean", "misclassification_var")

misclassification_mean_var_df <- rbind(misclassification_mean_var_df,
                                       data.frame("tree_number" = 1,
                                                  "record_number" = 100,
                                                  "misclassification_mean" = mean(misclassification_rates_100.2[i]),
                                                  "misclassification_var" = var(misclassification_rates_100.2[i])
                                       ))

misclassification_mean_var_df <- rbind(misclassification_mean_var_df,
                                       data.frame("tree_number" = 10,
                                                  "record_number" = 100,
                                                  "misclassification_mean" = mean(misclassification_rates_100.2[i]),
                                                  "misclassification_var" = var(misclassification_rates_100.2[i])
                                       ))

misclassification_mean_var_df <- rbind(misclassification_mean_var_df,
                                       data.frame("tree_number" = 100,
                                                  "record_number" = 100,
                                                  "misclassification_mean" = mean(misclassification_rates_100.2[i]),
                                                  "misclassification_var" = var(misclassification_rates_100.2[i])
                                       ))

misclassification_mean_var_df <- rbind(misclassification_mean_var_df,
                                       data.frame("tree_number" = 1,
                                                  "record_number" = 1000,
                                                  "misclassification_mean" = mean(misclassification_rates_100.2[i]),
                                                  "misclassification_var" = var(misclassification_rates_100.2[i])
                                       ))

misclassification_mean_var_df <- rbind(misclassification_mean_var_df,
                                       data.frame("tree_number" = 10,
                                                  "record_number" = 1000,
                                                  "misclassification_mean" = mean(misclassification_rates_100.2[i]),
                                                  "misclassification_var" = var(misclassification_rates_100.2[i])
                                       ))

misclassification_mean_var_df <- rbind(misclassification_mean_var_df,
                                       data.frame("tree_number" = 100,
                                                  "record_number" = 1000,
                                                  "misclassification_mean" = mean(misclassification_rates_100.2[i]),
                                                  "misclassification_var" = var(misclassification_rates_100.2[i])
                                       ))

misclassification_mean_var_df
#####

```

```

# When we set node_size = 25 and record_numbers = 100 and 1000 and check  $x1 < 0.5$  ,and tree_num = 1,10,
# 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_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)
}

misclassification_mean_var_df <- data.frame(matrix(ncol = 4, nrow=0))
colnames(misclassification_mean_var_df) <- c("tree_number", "record_number", "misclassification_mean",
misclassification_mean_var_df <- rbind(misclassification_mean_var_df,
data.frame("tree_number" = 1,
"record_number" = 100,
"misclassification_mean" = mean(misclassification_rates_1.1),
"misclassification_var" = var(misclassification_rates_1.1)
))

misclassification_mean_var_df <- rbind(misclassification_mean_var_df,
data.frame("tree_number" = 10,
"record_number" = 100,
"misclassification_mean" = mean(misclassification_rates_10.1),
"misclassification_var" = var(misclassification_rates_10.1)
))

```

```

misclassification_mean_var_df <- rbind(misclassification_mean_var_df,
  data.frame("tree_number" = 100,
    "record_number" = 100,
    "misclassification_mean" = mean(misclassification_rates_100.1),
    "misclassification_var" = var(misclassification_rates_100.1)
  ))

misclassification_mean_var_df <- rbind(misclassification_mean_var_df,
  data.frame("tree_number" = 1,
    "record_number" = 1000,
    "misclassification_mean" = mean(misclassification_rates_1.2),
    "misclassification_var" = var(misclassification_rates_1.2)
  ))

misclassification_mean_var_df <- rbind(misclassification_mean_var_df,
  data.frame("tree_number" = 10,
    "record_number" = 1000,
    "misclassification_mean" = mean(misclassification_rates_10.2),
    "misclassification_var" = var(misclassification_rates_10.2)
  ))

misclassification_mean_var_df <- rbind(misclassification_mean_var_df,
  data.frame("tree_number" = 100,
    "record_number" = 1000,
    "misclassification_mean" = mean(misclassification_rates_100.2),
    "misclassification_var" = var(misclassification_rates_100.2)
  ))

misclassification_mean_var_df
#####
# When we set node_size = 12 and record_numbers = 100 and check
# ((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 = 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_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)
}

misclassification_mean_var_df <- data.frame(matrix(ncol = 4, nrow=0))
colnames(misclassification_mean_var_df) <- c("tree_number", "record_number", "misclassification_mean", "misclassification_var")

misclassification_mean_var_df <- rbind(misclassification_mean_var_df,
                                       data.frame("tree_number" = 1,
                                                  "record_number" = 100,
                                                  "misclassification_mean" = mean(misclassification_rates_1.2),
                                                  "misclassification_var" = var(misclassification_rates_1.2)
                                       ))

misclassification_mean_var_df <- rbind(misclassification_mean_var_df,
                                       data.frame("tree_number" = 10,
                                                  "record_number" = 100,
                                                  "misclassification_mean" = mean(misclassification_rates_10.2),
                                                  "misclassification_var" = var(misclassification_rates_10.2)
                                       ))

misclassification_mean_var_df <- rbind(misclassification_mean_var_df,
                                       data.frame("tree_number" = 100,
                                                  "record_number" = 100,
                                                  "misclassification_mean" = mean(misclassification_rates_100.2),
                                                  "misclassification_var" = var(misclassification_rates_100.2)
                                       ))

misclassification_mean_var_df <- rbind(misclassification_mean_var_df,
                                       data.frame("tree_number" = 1,
                                                  "record_number" = 1000,
                                                  "misclassification_mean" = mean(misclassification_rates_1.2),
                                                  "misclassification_var" = var(misclassification_rates_1.2)
                                       ))

misclassification_mean_var_df <- rbind(misclassification_mean_var_df,
                                       data.frame("tree_number" = 10,
                                                  "record_number" = 1000,
                                                  "misclassification_mean" = mean(misclassification_rates_10.2),
                                                  "misclassification_var" = var(misclassification_rates_10.2)
                                       ))

misclassification_mean_var_df <- rbind(misclassification_mean_var_df,
                                       data.frame("tree_number" = 100,
                                                  "record_number" = 1000,
                                                  "misclassification_mean" = mean(misclassification_rates_100.2),
                                                  "misclassification_var" = var(misclassification_rates_100.2)
                                       ))

```

```

"record_number" = 1000,
"misclassification_mean" = mean(misclassification_rates_
"misclassification_var" = var(misclassification_rates_
))

misclassification_mean_var_df

#####
# When we set node_size = 12 and record_numbers = 100 and check
# ((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
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)
}

misclassification_mean_var_df <- data.frame(matrix(ncol = 4, nrow=0))
colnames(misclassification_mean_var_df) <- c("tree_number", "record_number", "misclassification_mean",
misclassification_mean_var_df <- rbind(misclassification_mean_var_df,
data.frame("tree_number" = 1,
"record_number" = 100,
"misclassification_mean" = mean(misclassification_rates_
"misclassification_var" = var(misclassification_rates_
))

```

```

misclassification_mean_var_df <- rbind(misclassification_mean_var_df,
                                     data.frame("tree_number" = 10,
                                                "record_number" = 100,
                                                "misclassification_mean" = mean(misclassification_rates_10_100),
                                                "misclassification_var" = var(misclassification_rates_10_100)
                                     ))

misclassification_mean_var_df <- rbind(misclassification_mean_var_df,
                                     data.frame("tree_number" = 100,
                                                "record_number" = 100,
                                                "misclassification_mean" = mean(misclassification_rates_100_100),
                                                "misclassification_var" = var(misclassification_rates_100_100)
                                     ))

misclassification_mean_var_df <- rbind(misclassification_mean_var_df,
                                     data.frame("tree_number" = 1,
                                                "record_number" = 1000,
                                                "misclassification_mean" = mean(misclassification_rates_1_1000),
                                                "misclassification_var" = var(misclassification_rates_1_1000)
                                     ))

misclassification_mean_var_df <- rbind(misclassification_mean_var_df,
                                     data.frame("tree_number" = 10,
                                                "record_number" = 1000,
                                                "misclassification_mean" = mean(misclassification_rates_10_1000),
                                                "misclassification_var" = var(misclassification_rates_10_1000)
                                     ))

misclassification_mean_var_df <- rbind(misclassification_mean_var_df,
                                     data.frame("tree_number" = 100,
                                                "record_number" = 1000,
                                                "misclassification_mean" = mean(misclassification_rates_100_1000),
                                                "misclassification_var" = var(misclassification_rates_100_1000)
                                     ))

misclassification_mean_var_df

set.seed(1234)
x1 <- runif(1000)
x2 <- runif(1000)

tedata <- cbind(x1, x2)
y <- as.numeric(compare_method1(x1, x2))
telabels <- as.factor(y)
plot(x1, x2, col = y+3, main = "First Dataset")

set.seed(5678)
x1 <- runif(1000)
x2 <- runif(1000)

tedata <- cbind(x1, x2)
y <- as.numeric(compare_method3(x1, x2))
telabels <- as.factor(y)

```

```

plot(x1, x2, col = y+3, main = "Third Dataset")

##### 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 log 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)
```