Machine Learning Computer Lab 1 Block2 (Group A7)

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

2024-01-16

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.

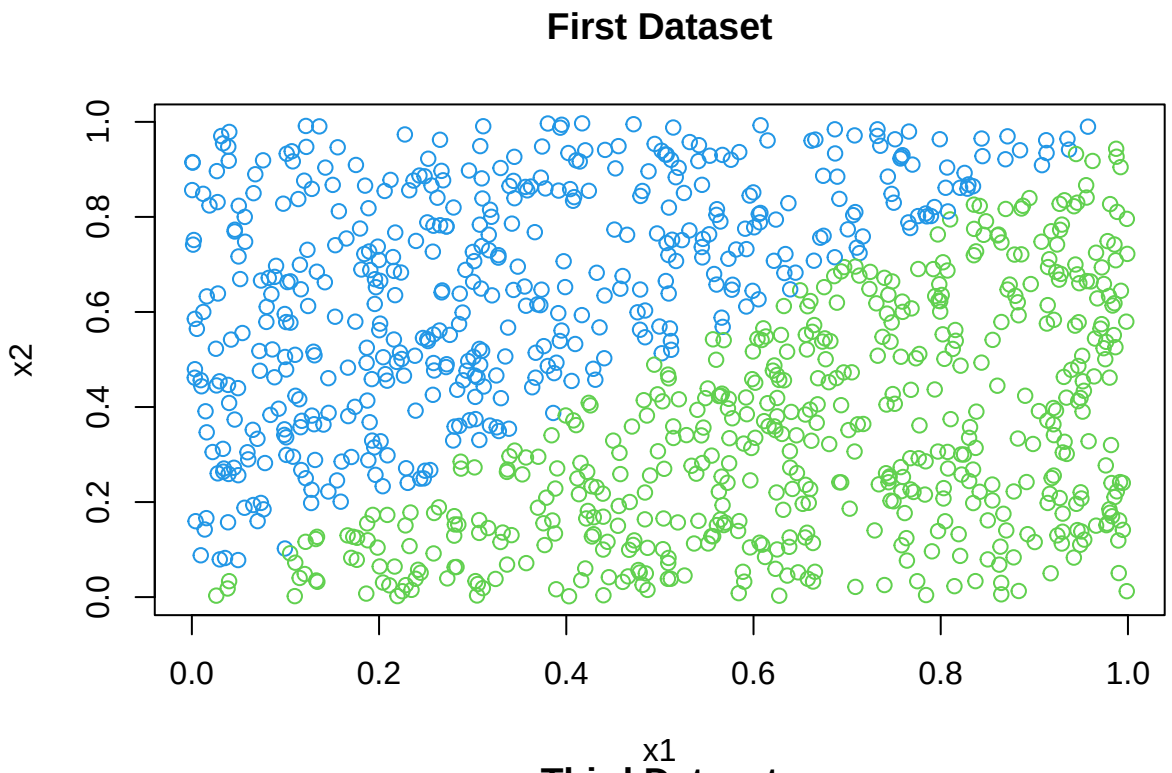
As discussed above, decreasing the node size will increase the tree size which yields better results.

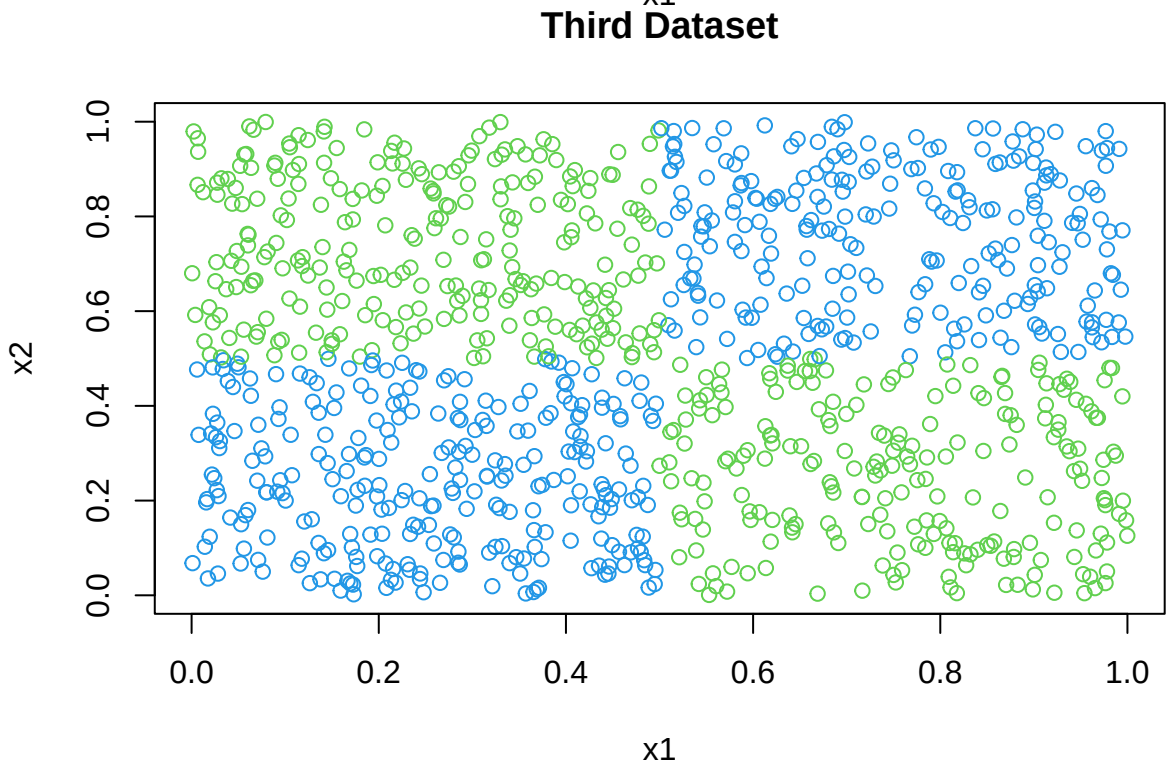
Task 6: The third dataset represents a slightly more complicated classification problem than the first one. Still, you should get better performance for it when using sufficient trees in the random forest. Explain why you get better performance

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.

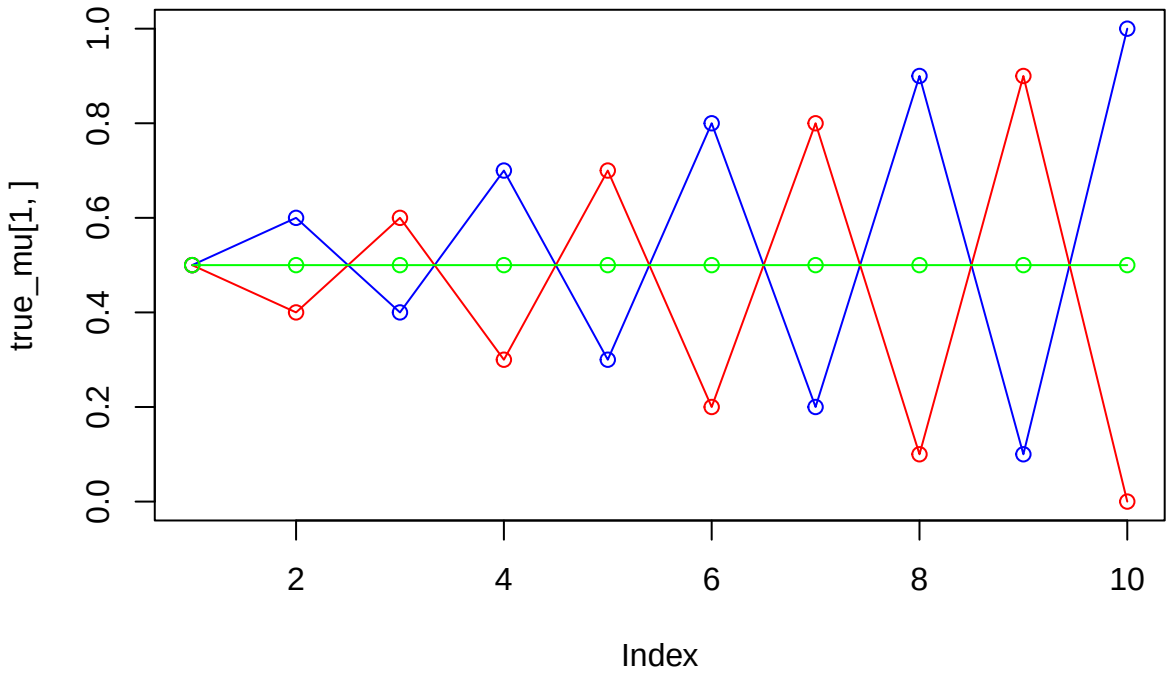




Assignment 2: MIXTURE MODELS

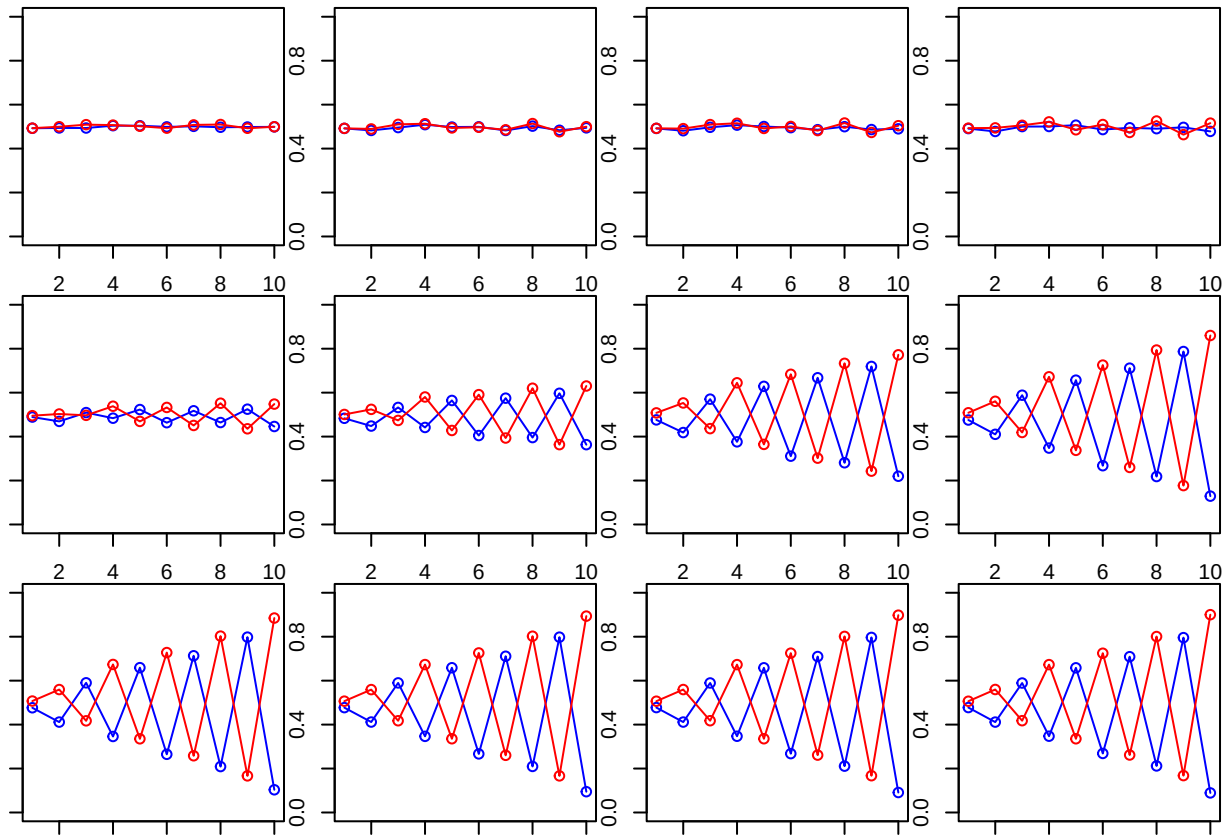
Answer:

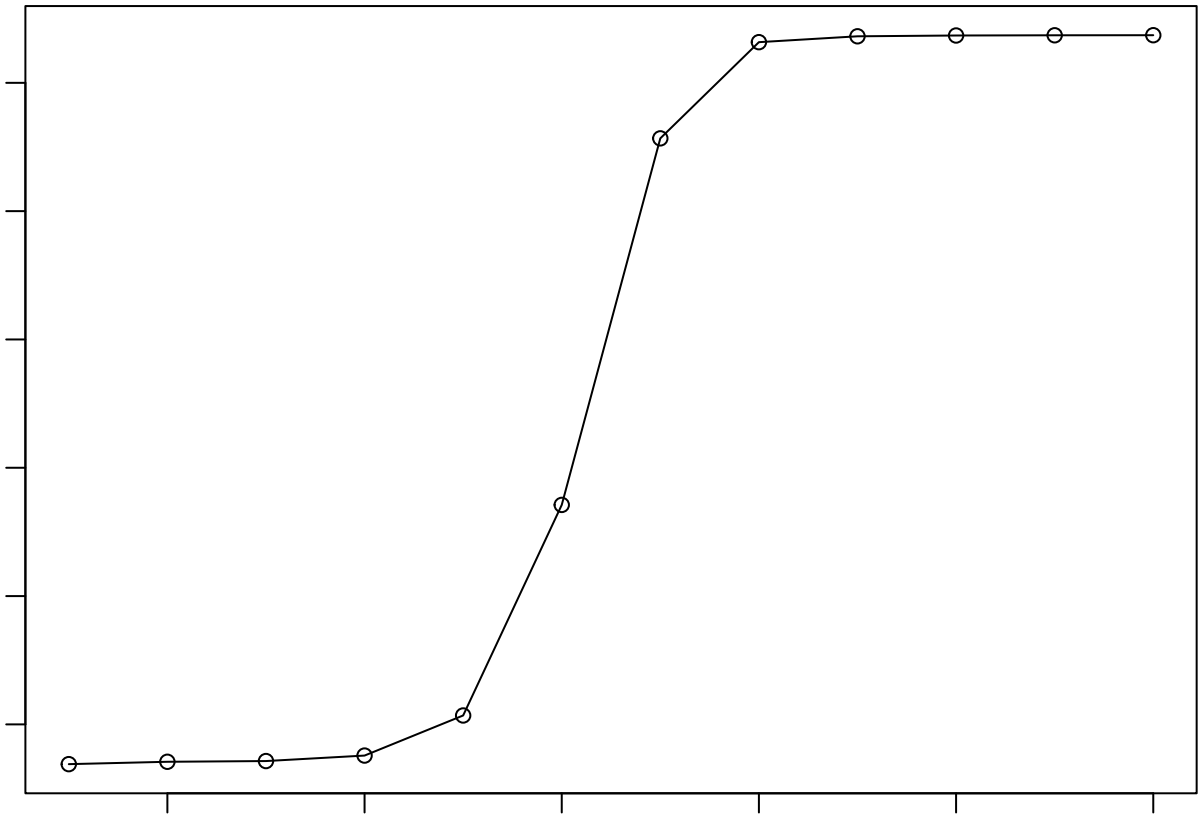
We will generate the π and μ



When M=2, we get the following result. Iteration runs 12 times, and the final log likelihood is -6362.89725846549, π values are[0.497125,0.502875].

The final shape is almost the same as the true μ shape after running 12 times.

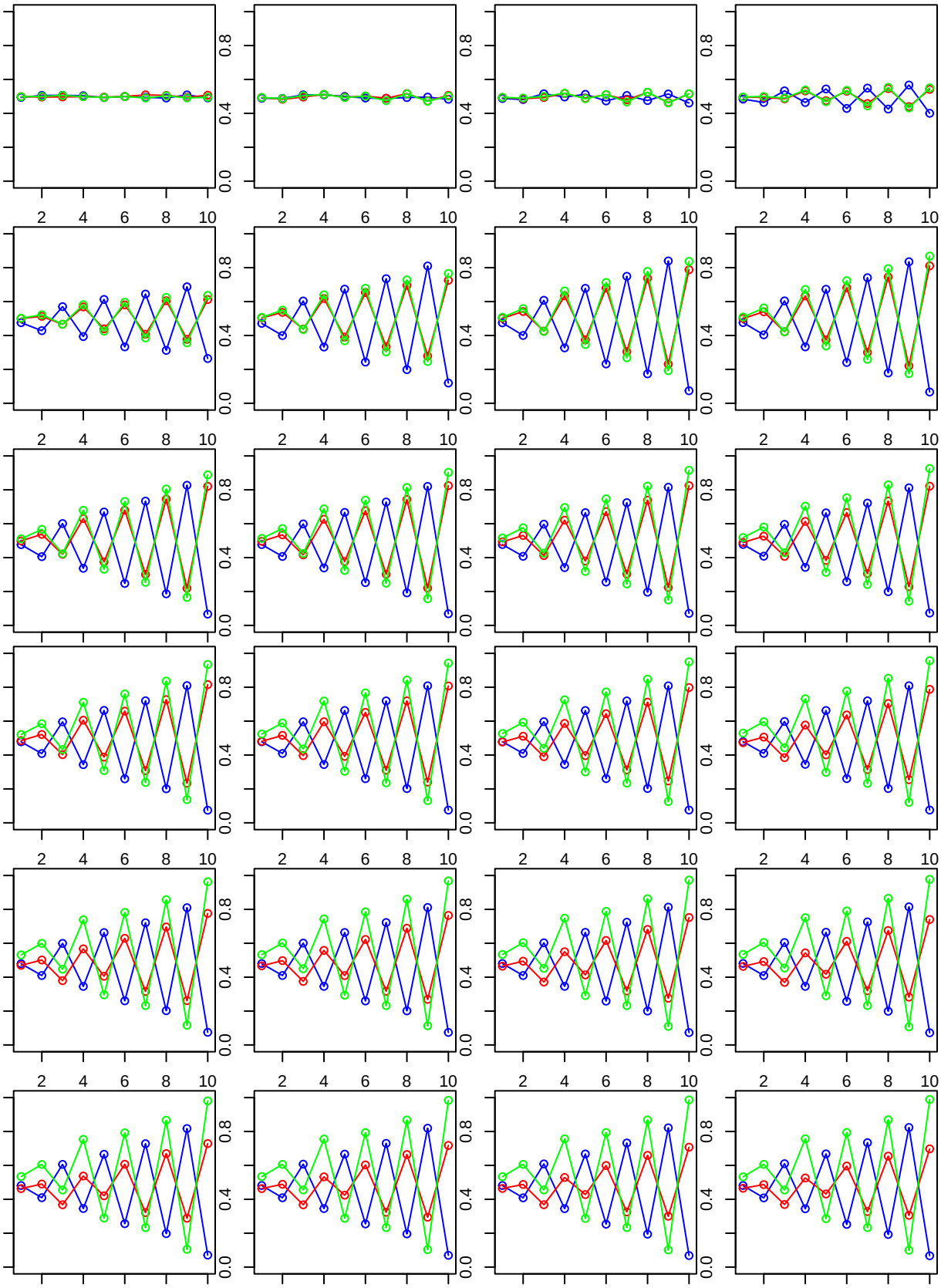


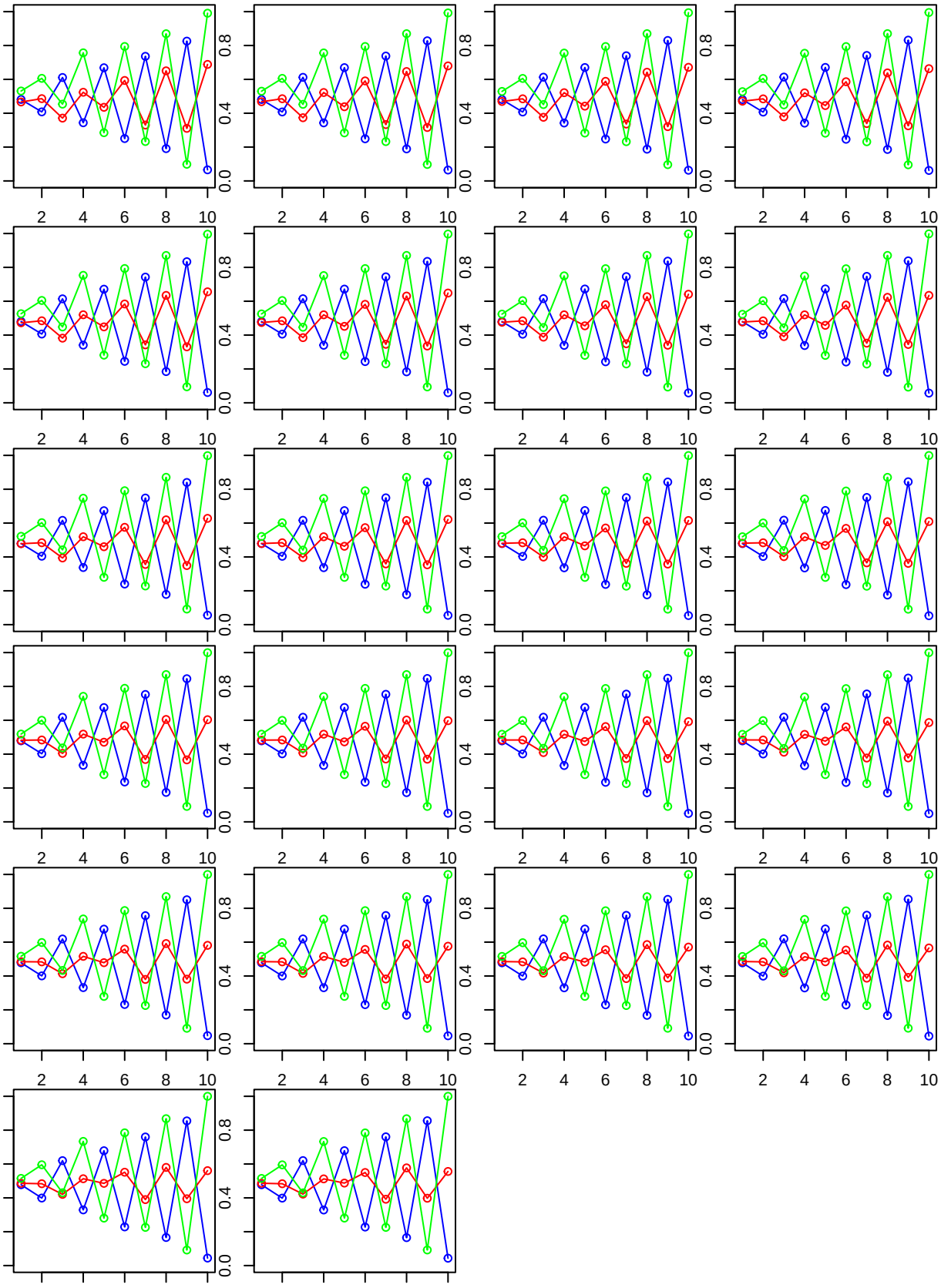


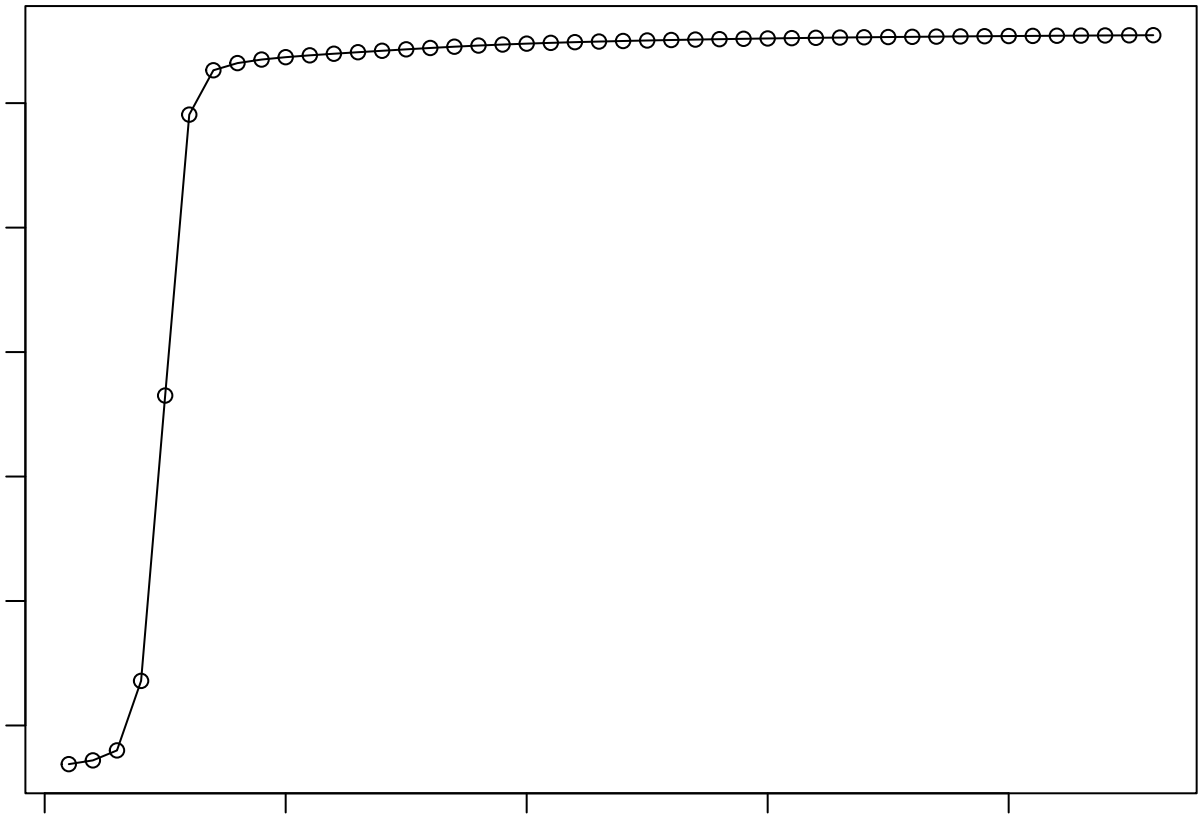
```
##  [1] "iteration is  1 log likelihood  -6930.97504223822"
##  [2] "iteration is  2 log likelihood  -6929.12473628598"
##  [3] "iteration is  3 log likelihood  -6928.56238318311"
##  [4] "iteration is  4 log likelihood  -6924.28062067272"
##  [5] "iteration is  5 log likelihood  -6893.05518568101"
##  [6] "iteration is  6 log likelihood  -6728.94831274647"
##  [7] "iteration is  7 log likelihood  -6443.28038652938"
##  [8] "iteration is  8 log likelihood  -6368.31837509574"
##  [9] "iteration is  9 log likelihood  -6363.73371302523"
## [10] "iteration is  10 log likelihood  -6363.10912783944"
## [11] "iteration is  11 log likelihood  -6362.94687236738"
## [12] "iteration is  12 log likelihood  -6362.89725846549"
## [1] "Pi value are"
## [1] 0.497125 0.502875
## [1] "Mu value are"
##           [,1]      [,2]      [,3]      [,4]      [,5]      [,6]      [,7]
## [1,] 0.4775488 0.4113939 0.5892308 0.3472420 0.6583712 0.2686589 0.7089490
## [2,] 0.5062860 0.5597531 0.4177551 0.6728856 0.3354854 0.7247188 0.2616231
##           [,8]      [,9]      [,10]
## [1,] 0.2118629 0.7957549 0.08905747
## [2,] 0.8007511 0.1678555 0.90027808
```

When M=3, we get the following result. Iteration runs 46 times, and the final log likelihood is -6345.43095278126, π values are[0.3964172,0.2787080,0.3248749].

The green and blue lines of the final shape are almost the same as the true μ shape after running 46 times. but the red one is not.


0.0
0.4
0.8
2
4
6
8
10


0.0
0.4
0.8
2
4
6
8
10

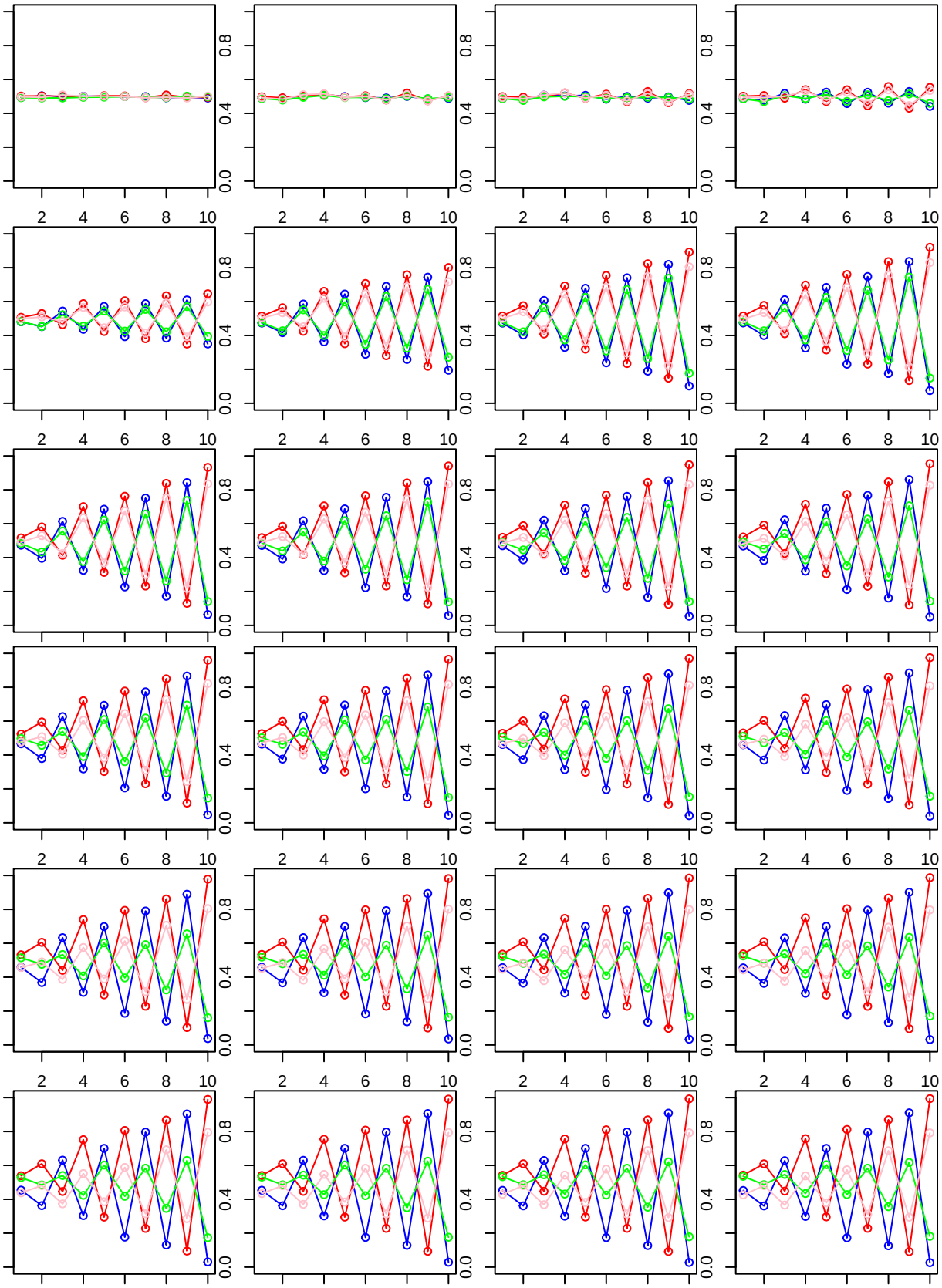


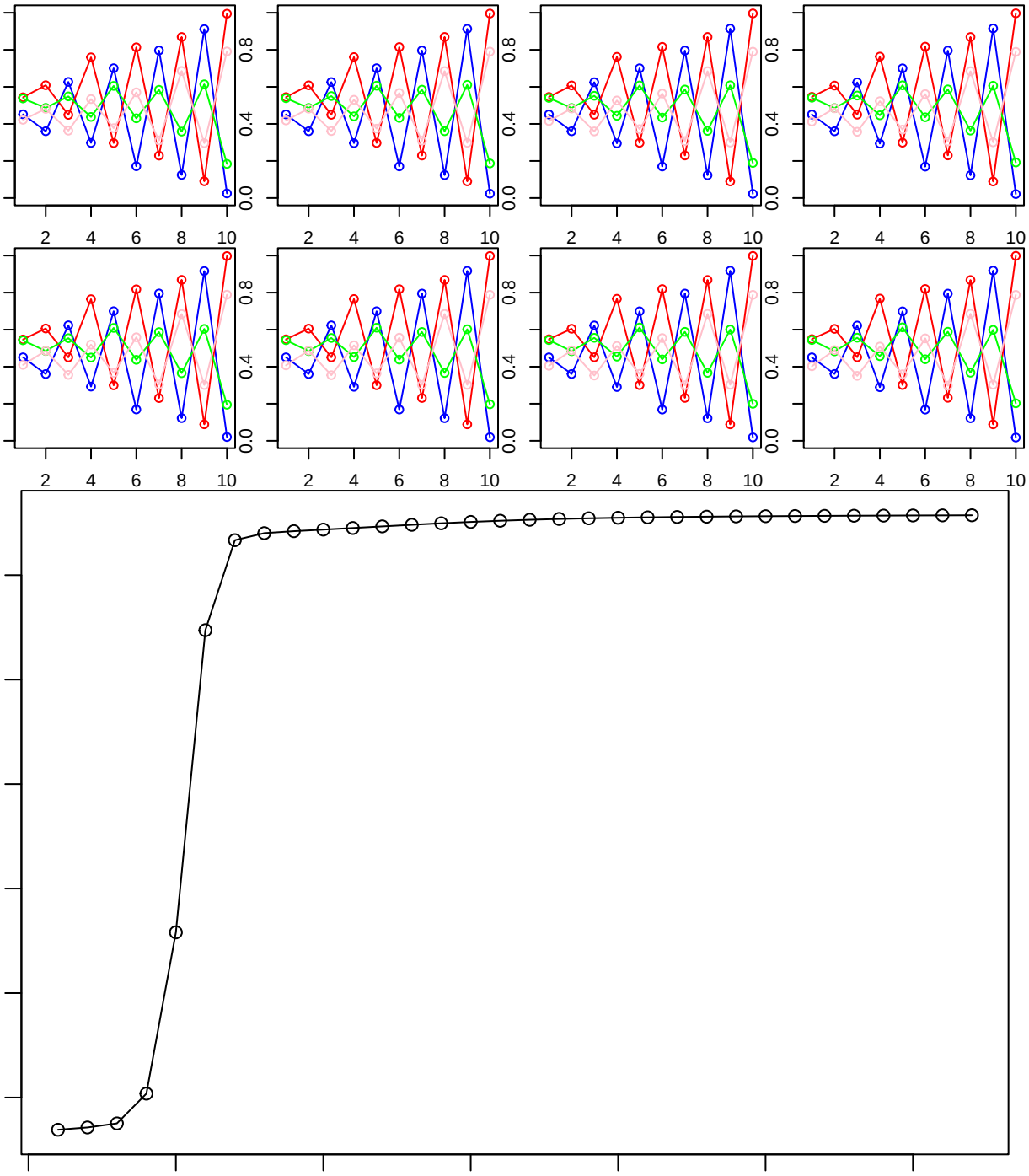
```
##  [1] "iteration is  1 log likelihood  -6931.06413254665"
##  [2] "iteration is  2 log likelihood  -6928.0514551885"
##  [3] "iteration is  3 log likelihood  -6920.02646571997"
##  [4] "iteration is  4 log likelihood  -6864.17585215607"
##  [5] "iteration is  5 log likelihood  -6634.91588360016"
##  [6] "iteration is  6 log likelihood  -6409.23350572715"
##  [7] "iteration is  7 log likelihood  -6373.59338896635"
##  [8] "iteration is  8 log likelihood  -6367.83252066043"
##  [9] "iteration is  9 log likelihood  -6364.98287644728"
## [10] "iteration is  10 log likelihood  -6363.07443097003"
## [11] "iteration is  11 log likelihood  -6361.59420090813"
## [12] "iteration is  12 log likelihood  -6360.30895253865"
## [13] "iteration is  13 log likelihood  -6359.10342638122"
## [14] "iteration is  14 log likelihood  -6357.93038899127"
## [15] "iteration is  15 log likelihood  -6356.78576962725"
## [16] "iteration is  16 log likelihood  -6355.68940153526"
## [17] "iteration is  17 log likelihood  -6354.66790300579"
## [18] "iteration is  18 log likelihood  -6353.7419024208"
## [19] "iteration is  19 log likelihood  -6352.91996103104"
## [20] "iteration is  20 log likelihood  -6352.19882582218"
## [21] "iteration is  21 log likelihood  -6351.5673258098"
## [22] "iteration is  22 log likelihood  -6351.0109633494"
## [23] "iteration is  23 log likelihood  -6350.51542353769"
## [24] "iteration is  24 log likelihood  -6350.06852095831"
## [25] "iteration is  25 log likelihood  -6349.66085317177"
## [26] "iteration is  26 log likelihood  -6349.28565083297"
## [27] "iteration is  27 log likelihood  -6348.93825426503"
## [28] "iteration is  28 log likelihood  -6348.61550332777"
```

```
## [29] "iteration is  29 log likelihood  -6348.31519563042"
## [30] "iteration is  30 log likelihood  -6348.03567497076"
## [31] "iteration is  31 log likelihood  -6347.77555616288"
## [32] "iteration is  32 log likelihood  -6347.5335647161"
## [33] "iteration is  33 log likelihood  -6347.30846036779"
## [34] "iteration is  34 log likelihood  -6347.09901441273"
## [35] "iteration is  35 log likelihood  -6346.90401655246"
## [36] "iteration is  36 log likelihood  -6346.72229408616"
## [37] "iteration is  37 log likelihood  -6346.5527327479"
## [38] "iteration is  38 log likelihood  -6346.39429354425"
## [39] "iteration is  39 log likelihood  -6346.24602342023"
## [40] "iteration is  40 log likelihood  -6346.10705969373"
## [41] "iteration is  41 log likelihood  -6345.97662927898"
## [42] "iteration is  42 log likelihood  -6345.8540441135"
## [43] "iteration is  43 log likelihood  -6345.73869418694"
## [44] "iteration is  44 log likelihood  -6345.6300393498"
## [45] "iteration is  45 log likelihood  -6345.5276007896"
## [46] "iteration is  46 log likelihood  -6345.43095278126"
## [1] "Pi value are"
## [1] 0.3964172 0.2787080 0.3248749
## [1] "Mu value are"
##           [,1]      [,2]      [,3]      [,4]      [,5]      [,6]      [,7]
## [1,] 0.4774399 0.3984407 0.6200854 0.3283412 0.6787726 0.2274644 0.7605397
## [2,] 0.4863053 0.4834283 0.4221438 0.5121400 0.4872729 0.5497432 0.3918968
## [3,] 0.5146518 0.5950474 0.4294967 0.7329049 0.2804651 0.7837214 0.2255769
##           [,8]       [,9]      [,10]
## [1,] 0.1649014 0.85568380 0.04340808
## [2,] 0.5771323 0.39774130 0.55592505
## [3,] 0.8673460 0.09215422 0.99992821
```

When M=4, we get the following result. Iteration runs 32 times, and the final log likelihood is -6342.66757470638, π values are [0.2690190,0.2863050,0.2457246,0.1989515].

The green and blue lines of the final shape are almost the same as the true μ shape after running 46 times. but the red one is not.


0.0
0.4
0.8
2
4
6
8
10




```
##  [1] "iteration is  1 log likelihood  -6930.83750078059"
##  [2] "iteration is  2 log likelihood  -6928.64051824251"
##  [3] "iteration is  3 log likelihood  -6924.74771319715"
##  [4] "iteration is  4 log likelihood  -6896.24964661278"
##  [5] "iteration is  5 log likelihood  -6741.89647187186"
##  [6] "iteration is  6 log likelihood  -6452.65775131851"
##  [7] "iteration is  7 log likelihood  -6366.49305307265"
##  [8] "iteration is  8 log likelihood  -6359.76358587706"
##  [9] "iteration is  9 log likelihood  -6357.87597603285"
## [10] "iteration is  10 log likelihood  -6356.37173523068"
```

```
## [11] "iteration is  11 log likelihood  -6354.86027971753"
## [12] "iteration is  12 log likelihood  -6353.31039058433"
## [13] "iteration is  13 log likelihood  -6351.77631441857"
## [14] "iteration is  14 log likelihood  -6350.33008309037"
## [15] "iteration is  15 log likelihood  -6349.03024577277"
## [16] "iteration is  16 log likelihood  -6347.90819779292"
## [17] "iteration is  17 log likelihood  -6346.96821722835"
## [18] "iteration is  18 log likelihood  -6346.19557790646"
## [19] "iteration is  19 log likelihood  -6345.56630537181"
## [20] "iteration is  20 log likelihood  -6345.05457158999"
## [21] "iteration is  21 log likelihood  -6344.63677606174"
## [22] "iteration is  22 log likelihood  -6344.29308268213"
## [23] "iteration is  23 log likelihood  -6344.00753168042"
## [24] "iteration is  24 log likelihood  -6343.76755528191"
## [25] "iteration is  25 log likelihood  -6343.56334837779"
## [26] "iteration is  26 log likelihood  -6343.3872852444"
## [27] "iteration is  27 log likelihood  -6343.23343829888"
## [28] "iteration is  28 log likelihood  -6343.09719894829"
## [29] "iteration is  29 log likelihood  -6342.97498345941"
## [30] "iteration is  30 log likelihood  -6342.86400514758"
## [31] "iteration is  31 log likelihood  -6342.76209727182"
## [32] "iteration is  32 log likelihood  -6342.66757470638"
## [1] "Pi value are"
## [1] 0.2690190 0.2863050 0.2457246 0.1989515
## [1] "Mu value are"
##           [,1]      [,2]      [,3]      [,4]      [,5]      [,6]      [,7]
## [1,] 0.4502917 0.3606587 0.6220817 0.2892407 0.6986320 0.1681768 0.7943990
## [2,] 0.5487864 0.6040921 0.4511711 0.7675478 0.3010522 0.8195305 0.2318913
## [3,] 0.5439579 0.4827437 0.5563603 0.4568300 0.6123061 0.4400351 0.5885625
## [4,] 0.4025047 0.4895637 0.3506597 0.5085745 0.3588983 0.5528693 0.2979403
##           [,8]       [,9]      [,10]
## [1,] 0.1215732 0.91870837 0.01774129
## [2,] 0.8679302 0.08877946 0.99833984
## [3,] 0.3677877 0.59890299 0.20227253
## [4,] 0.6857315 0.30292227 0.78760041
```

Above all, all likelihood plots show that it increases monotonically. It increases sharply before the 7th iteration. After that, the likelihood increases slowly.

Also as mentioned before, not all lines fit the true μ value well. especially when M=3 and 4.

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)
}
```

```r
##############################################################################################
# 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], "misclas
misclassification_df <- rbind(misclassification_df, data.frame("tree_number" = tree_numbers[2], "misclas
misclassification_df <- rbind(misclassification_df, data.frame("tree_number" = tree_numbers[3], "misclas

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

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
                                                  "misclassification_var" = var(misclassification_rates_
                                                  ))

misclassification_mean_var_df <- rbind(misclassification_mean_var_df,
                                       data.frame("tree_number" = 100,
                                                  "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" = 1,
                                                  "record_number" = 1000,
                                                  "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" = 1000,
                                                  "misclassification_mean" = mean(misclassification_rates
                                                  "misclassification_var" = var(misclassification_rates_
                                                  ))

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

misclassification_mean_var_df
##############################################################################
```

```r
# 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_rate
                                                  "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_rate
                                                  "misclassification_var" = var(misclassification_rates_
                                                  ))
```

```
misclassification_mean_var_df <- rbind(misclassification_mean_var_df,
                                       data.frame("tree_number" = 100,
                                                  "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" = 1,
                                                  "record_number" = 1000,
                                                  "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" = 1000,
                                                  "misclassification_mean" = mean(misclassification_rates
                                                  "misclassification_var" = var(misclassification_rates_
                                                  ))

misclassification_mean_var_df <- rbind(misclassification_mean_var_df,
                                       data.frame("tree_number" = 100,
                                                  "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 = 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_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
                                                  "misclassification_var" = var(misclassification_rates_
                                                  ))

misclassification_mean_var_df <- rbind(misclassification_mean_var_df,
                                       data.frame("tree_number" = 100,
                                                  "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" = 1,
                                                  "record_number" = 1000,
                                                  "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" = 1000,
                                                  "misclassification_mean" = mean(misclassification_rates
                                                  "misclassification_var" = var(misclassification_rates_
                                                  ))

misclassification_mean_var_df <- rbind(misclassification_mean_var_df,
                                       data.frame("tree_number" = 100,
```

```
                                                "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
                                                  "misclassification_var" = var(misclassification_rates_
                                                  ))

misclassification_mean_var_df <- rbind(misclassification_mean_var_df,
                                       data.frame("tree_number" = 100,
                                                  "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" = 1,
                                                  "record_number" = 1000,
                                                  "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" = 1000,
                                                  "misclassification_mean" = mean(misclassification_rates
                                                  "misclassification_var" = var(misclassification_rates_
                                                  ))

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

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)
```

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

# define a function to implement the mixture model
mixel_model_fun <- function(M){
  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)
  }

  par(mfrow=c(3,4),mar = c(1, 1, 1, 1))

  iterLog <- vector(length = max_it)

  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="pink")
    }
    Sys.sleep(2)

    # E-step: Computation of the weights
    for (i in 1:n) {
        pxi <- 0
        for (m in 1:M) {
          bern <- 1
          for(d in 1:D) {
            bern <- bern * (mu[m,d]^x[i,d]) * (1-mu[m,d])^(1-x[i,d])
          }
          pxi <- pxi + pi[m] * bern
        }

        for (m in 1:M) {
          bern <- 1
          for(d in 1:D) {
            bern <- bern * (mu[m,d]^x[i,d]) * (1-mu[m,d])^(1-x[i,d])
          }
          w[i, m] <- pi[m] * bern / pxi
        }
        w[i,] <- w[i,] / sum(w[i,])
    }

    #Log likelihood computation.
```

```
    llik[it] <- 0
    for (i in 1:n) {
      pxi <- 0
      for (m in 1:M) {
        bern <- 1
        for(d in 1:D) {
          bern <- bern * (mu[m,d]^x[i,d]) * (1-mu[m,d])^(1-x[i,d])
        }
        pxi <- pxi + pi[m] * bern
      }
      llik[it] <- llik[it] + log(pxi)
    }

    iterLog[it] <- paste("iteration is ", it,
                         "log likelihood ", llik[it])

    flush.console()

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

    pi <- apply(w,2,mean)
    mu <- t(w) %*% x / colSums(w)
  }

  par(mfrow=c(1,1),mar = c(1, 1, 1, 1))
  plot(llik[1:it], type = "o")
  print(iterLog[1:it])

  print("Pi value are")
  print(pi)
  print("Mu value are")
  print(mu)
}
# init data set (copy from pdf file)
set.seed(1234567890)
# max number of EM iterations
max_it <- 100
# min change in log likelihood 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)
```

```r
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])
  }
}
mixel_model_fun(M=2)
mixel_model_fun(M=3)
mixel_model_fun(M=4)
```