Block 2 Lab 1 Group A3

varsi146, aswma317, akssr921

2022-12-22

Statement of Contribution

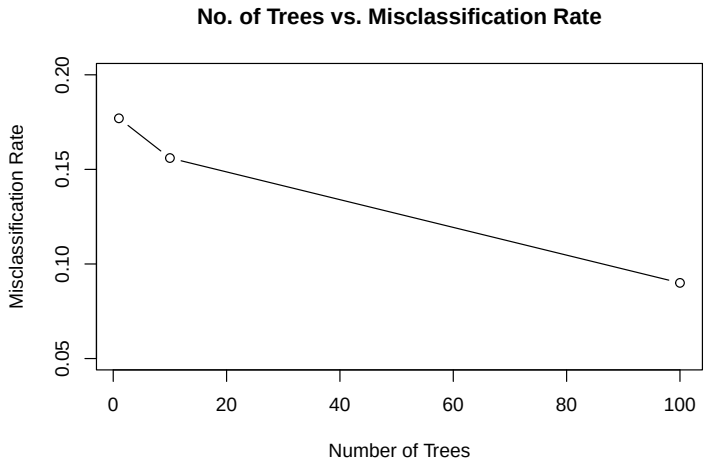
Question 1 was done by Varun and Akshath, and question 2 was done by Aswath.

Question 1: Ensemble Methods

Task 1 : Learning random forests using given test and training data for *ntree* = 1, 10 and 100 trees, *nodesize* = 25 and *keep.forest* = TRUE.

We can see from the table and the graph below, that, as the the number of trees in a random forest grows, the mis-classification rates are decreasing.

```
##   Number_of_Tree Missclass_Rate
## 1              1          0.177
## 2             10          0.156
## 3            100          0.090
```



Task 2: Repeat above procedure for 1000 training datasets of size 100 and compute the mean and variance of misclassification errors. Report results for when the random forest has 1, 10, 100 trees:

Below table shows the results of mean and variance of mis-classification errors of random forest models of trees 1, 10, 100 fitted on each of the 1000 datasets created. The trend reported is that the mean and variance of mis-classification errors of the models reduce as the number of trees increase:

```
##          Mean_MisClass_Trees Variance_MisClass_Trees
## 1_Tree              0.211223            0.0030422615
## 10_Tree             0.135612            0.0009412167
## 100_Tree            0.113609            0.0008993194
```

Task 3: Repeat the exercise above but this time use the condition (x1<0.5) instead of (x1<x2) when producing the training and test datasets.

Below table shows the results of mean and variance of mis-classification errors of random forest models of trees 1, 10, 100 fitted on each of the 1000 datasets created. The trend reported is that the mean and variance of mis-classification errors of the models reduce as the number of trees increase. However, the mean and variance of mis-classification errors are much lower as compared to the previous case:

```
##          Mean_MisClass_Trees Variance_MisClass_Trees
## 1_Tree              0.093448            1.759429e-02
## 10_Tree             0.017582            8.072165e-04
## 100_Tree            0.005781            5.288993e-05
```

Task 4: Repeat the exercise above but this time use the condition ((x1<0.5 & x2<0.5) | (x1>0.5 & x2>0.5)) instead of (x1<x2) when producing the training and test datasets. Unlike above, use nodesize = 12 for this exercise.

Below table shows the results of mean and variance of mis-classification errors of random forest models of trees 1, 10, 100 fitted on each of the 1000 datasets created. The trend reported is that the mean and variance of mis-classification errors of the models reduce as the number of trees increase. However, the mean and variance of mis-classification errors are much lower as compared to the first case but only as the number of trees grow:

```
##          Mean_MisClass_Trees Variance_MisClass_Trees
## 1_Tree              0.257896             0.013731443
## 10_Tree             0.118106             0.002994876
## 100_Tree            0.072766             0.001279689
```

Task 4: Answer the following questions:

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

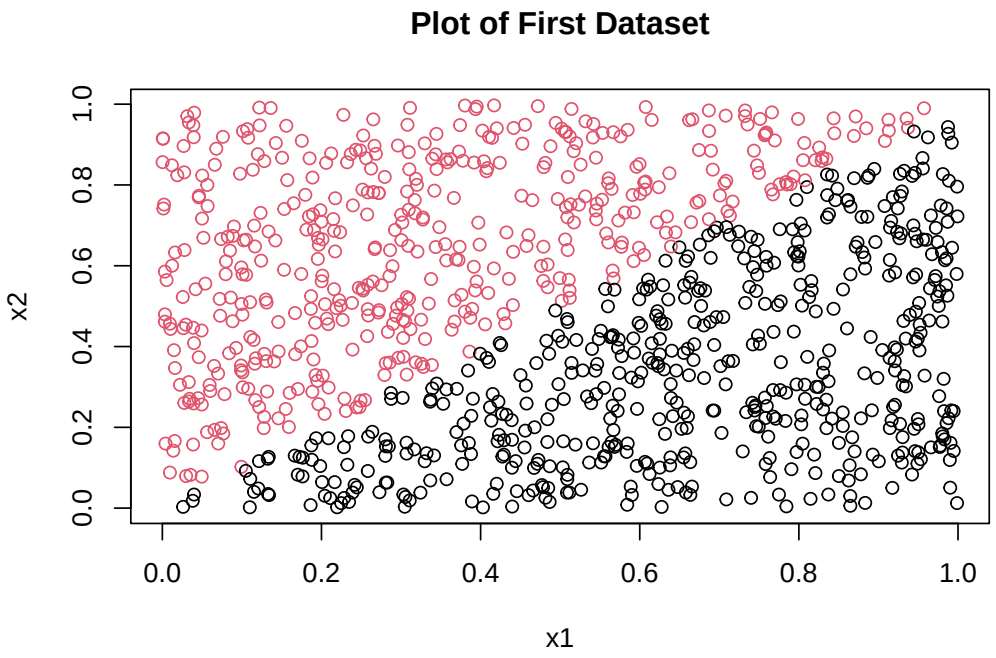
In the below table, the first three columns represent the mean of mis-classification rates for 1, 10, 100 trees for each case where the condition on which test data is generated changes. The last three columns similarly represent variance of misclassification errors:

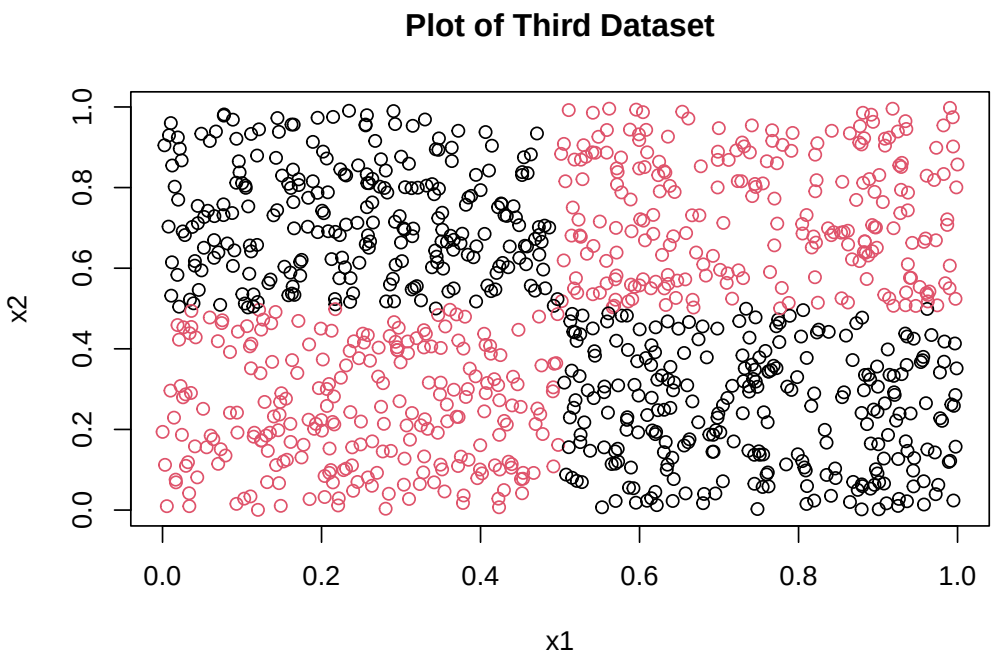
In each of the case demonstrated below, with respect to mean and variance of misclassification error, the values decrease as the number of trees in the random forest grows.

```
##         Mean_Case1 Mean_Case2 Mean_Case3    Var_Case1    Var_Case2   Var_Case3
## 1Tree     0.211223   0.093448   0.257896 0.0030422615 1.759429e-02 0.013731443
## 10Tree    0.135612   0.017582   0.118106 0.0009412167 8.072165e-04 0.002994876
## 100Tree   0.113609   0.005781   0.072766 0.0008993194 5.288993e-05 0.001279689
```

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?

We can see the plots for first and third case below, by observing the first plot we can say that we can also classify the two classes by adding decision boundary using logistic regression, but when we observe the plot of third case we can see it has slightly more complicated classification problem, we cannot use one linear decision boundary to classify the classes, this needs multiple decision boundaries for classification. So, decision trees and random forest works better in third case. We will get better results using sufficient trees in random forest by preventing over fitting problems. Moreover, for third dataset the decision boundaries can be parallel to the axes, using sufficient trees in the random forest will give us better performance and also in the third case, we are reducing the nodesize which will grow larger trees in the forest and yields better results.





Question 2: Mixture Models

Implement EM algorithm for a Bernoulli mixture model. What happens when there are too few and to many clusters?

Derivation of log-likelihood:

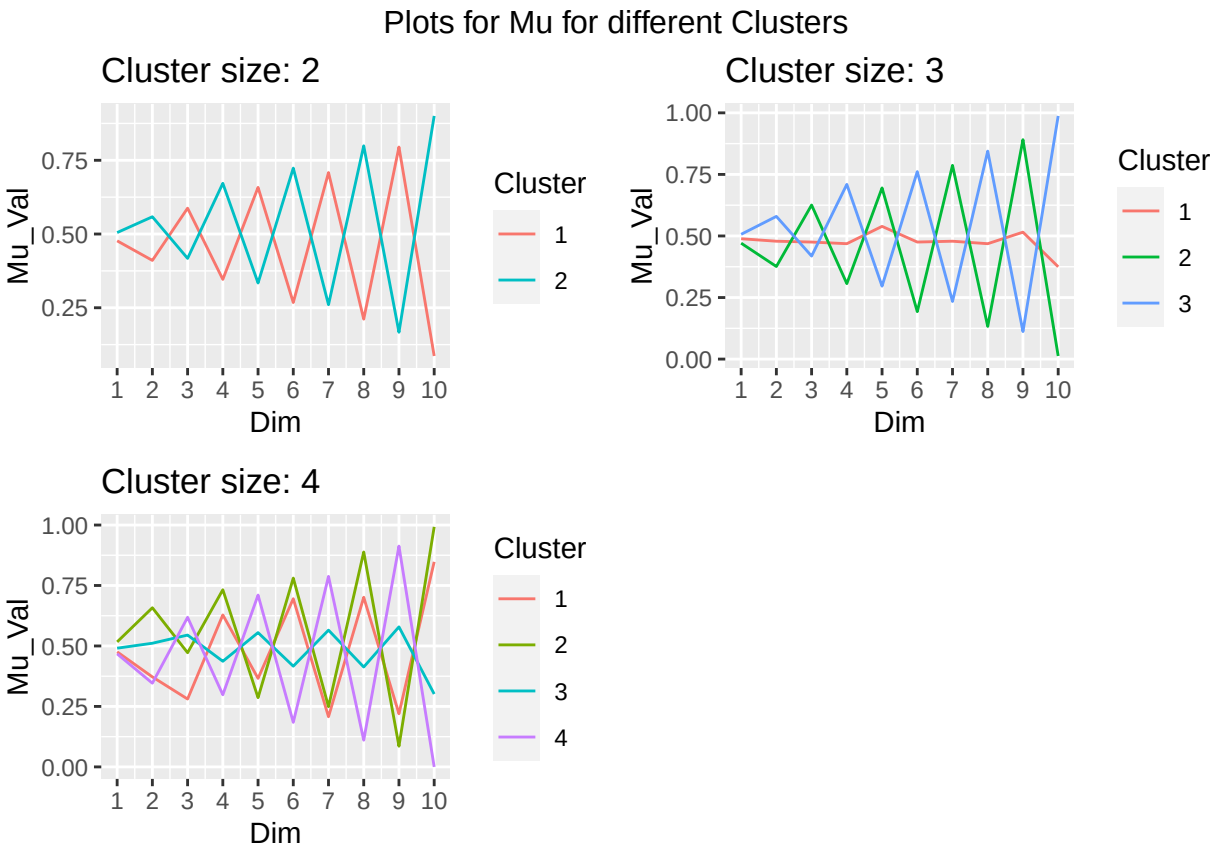
$$\sum_{i=1}^{n}\sum_{m=1}^{M} w_{i,m}\{ln[Bern(x_i|\mu)] + ln\pi_m\}$$

where, it is given that:

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

Plugging this in the above log likelihood formula, we get:

$$\sum_{i=1}^{n}\sum_{m=1}^{M} w_{i,m} * \{\sum_{d=1}^{D}[x_d * ln\mu_{m,d} + (1-x_d) * ln(1-\mu_{m,d}))] + ln\pi_m\}$$



As seen in the above graphs, for cluster size 2, 2 of the true clusters are classified under 1 cluster. For cluster of size 3, 2 cluster distributions almost matches with the true distribution, however, the mu of the 3rd distribution slightly deviates from the true mu. In the 3rd graph with cluster of size 4; the line which is supposed to run through 0.5 gets pulled to the top or bottom based on the probability of the observed data belonging to different clusters. Here 2 clusters try to capture the 1 true cluster.

```
## pi value for 2 clusters: 0.4968944 0.5031056
```

```
## pi value for 3 clusters: 0.2987225 0.3164898 0.3847877
```

```
## pi value for 4 clusters: 0.1639437 0.2690014 0.2953993 0.2716556
```

As seen, the true pi is almost the same cluster of size 2 and the deviation increases as the cluster size increases. The reason for it is same as the rationale given above for mu as the parameters pi and mu are calculated from the weight.

As seen in the loglikelihood graphs, it is clearly visible that as the number of clusters increases, it becomes difficult for the algorithm to find the local minima, hence there is an increase in iterations for convergence as cluster size increases.

Appendix

```
library(randomForest)
library(ggplot2)
library(gridExtra)
knitr::opts_chunk$set(echo = TRUE)
# Training Data Generation

set.seed(54321)
x1 <- runif(100)

set.seed(7890)
x2 <- runif(100)

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

# Random forest model for ntree = 1
```

```
rf_model1 <- randomForest(as.factor(trlabels)~.,
                          data = trdata,
                          ntree = 1,
                          nodesize = 25,
                          keep.forest = TRUE)

# Random forest model for ntree = 10
rf_model2 <- randomForest(as.factor(trlabels)~.,
                          data = trdata,
                          ntree = 10,
                          nodesize = 25,
                          keep.forest = TRUE)

# Random forest model for ntree = 100
rf_model3 <- randomForest(as.factor(trlabels)~.,
                          data = trdata,
                          ntree = 100,
                          nodesize = 25,
                          keep.forest = TRUE)

model_list <- list(A = rf_model1,
                   B = rf_model2,
                   C = rf_model3)
# Test Data Generation

set.seed(1234)

x1 <- runif(1000)
x2 <- runif(1000)
tedata <- cbind(x1, x2)

y <- as.numeric(x1<x2)
telabels <- as.factor(y)
tedata <- cbind(tedata, telabels)
tedata[,3] <- as.factor(tedata[,3])


# plot(x1, x2, col = (y+1))

# Function to compute misclassification
missclass=function(X,X1){
  n=length(X)
  return(1-sum(diag(table(X,X1)))/n)
}

# Data frame to store the misclassification rates of each of the random forest
# models above.

misclass_df <- data.frame(matrix(ncol = 2, nrow = 0))
colnames(misclass_df) <- c('Number_of_Tree', 'Missclass_Rate')

# Computing misclassification rates of the models created above:
```

```r
for (i in 1:length(model_list)) {
  pred_vals <- c()
  misclass_rates <- c()
  pred_vals <- predict(object = model_list[[i]], newdata = tedata, type = 'class')
  misclass_rates <- missclass(tedata[,3], pred_vals)
  misclass_df <- rbind(misclass_df, data.frame('Number_of_Tree' = model_list[[i]]$ntree,
                                               'Missclass_Rate' = misclass_rates))
}
misclass_df
plot(x = misclass_df$Number_of_Tree,
     y = misclass_df$Missclass_Rate,
     type = 'b',
     ylim = c(0.05,0.2),
     xlab = 'Number of Trees', ylab = 'Misclassification Rate',
     main = 'No. of Trees vs. Misclassification Rate')
# Empty list to store 1000 training datasets:
list_df <- list()

# Creation of 1000 training datasets of size 100 and storing them in the above list:
for (i in 1:1000) {
  x1 <- runif(100)
  x2 <- runif(100)
  trdata <- as.data.frame(matrix(ncol = 3, nrow = 100))
  colnames(trdata) <- c('x1', 'x2', 'trlabels')
  trdata[,1] <- x1
  trdata[,2] <- x2
  y <- as.numeric(x1 < x2)
  trlabels <- as.factor(y)
  trdata[,3] <- trlabels
  list_df[[i]] <- (trdata)
}

tree_no <- c(1,10,100)
misclass_rates <- c()
missrates_1 <- c()
missrates_10 <- c()
missrates_100 <- c()

# Fitting a random forest model of 1, 10, 100 trees on each of the 1000 datasets
# created above:

for (i in 1:length(tree_no)) {
  for (j in 1:length(list_df)) {
    rf_model <- randomForest(as.factor(trlabels)~.,
                             data = as.data.frame.list(list_df[[j]]),
                             ntree = tree_no[i],
                             nodesize = 25,
                             keep.forest = TRUE)
    pred_vals <- c()

    pred_vals <- predict(object = rf_model, newdata = tedata, type = 'class')
    misclass_rates <- missclass(tedata[,3], pred_vals)
```

```
    if (tree_no[i] == 1) {
      missrates_1 <- rbind(missrates_1, misclass_rates)
    }else if (tree_no[i] == 10) {
      missrates_10 <- rbind(missrates_10, misclass_rates)
    }
    else{
      missrates_100 <- rbind(missrates_100, misclass_rates)
    }
  }
}

new_misclass_df <- data.frame(Tree1_Misclass = missrates_1,
                              Tree10_Misclass = missrates_10,
                              Tree100_Misclass = missrates_100)

row.names(new_misclass_df) <- NULL

mean_tree1 <- mean(new_misclass_df[,1])
mean_tree2 <- mean(new_misclass_df[,2])
mean_tree3 <- mean(new_misclass_df[,3])

variance_tree1 <- var(new_misclass_df[,1])
variance_tree2 <- var(new_misclass_df[,2])
variance_tree3 <- var(new_misclass_df[,3])

case1_df <- as.data.frame(matrix(ncol = 2, nrow = 0))
colnames(case1_df) <- c('Mean_MisClass_Trees', 'Variance_MisClass_Trees')
case1_df <- rbind(case1_df, data.frame('Mean_MisClass_Trees' = c(mean_tree1,
                                                                 mean_tree2,
                                                                 mean_tree3),
                                       'Variance_MisClass_Trees' = c(variance_tree1,
                                                                     variance_tree2,
                                                                     variance_tree3)))
rownames(case1_df) <- c('1_Tree', '10_Tree', '100_Tree')
case1_df
# Empty list to store 1000 training datasets for second condition:
list_df_2 <- list()

# Creation of 1000 training datasets of size 100 and storing them in the above list:
for (i in 1:1000) {
  x1 <- runif(100)
  x2 <- runif(100)
  trdata_2 <- as.data.frame(matrix(ncol = 3, nrow = 100))
  colnames(trdata_2) <- c('x1', 'x2', 'trlabels')
  trdata_2[,1] <- x1
  trdata_2[,2] <- x2
  y <- as.numeric(x1 < 0.5)
  trlabels <- as.factor(y)
  trdata_2[,3] <- trlabels
  list_df_2[[i]] <- (trdata_2)
}

# Generating new Test Data based on condition x1<0.5 instead of x1<x2
```

```r
set.seed(9000)

x1 <- runif(1000)
x2 <- runif(1000)
tedata_2 <- cbind(x1, x2)

y <- as.numeric(x1<0.5)
telabels <- as.factor(y)
tedata_2 <- cbind(tedata_2, telabels)
tedata_2[,3] <- as.factor(tedata_2[,3])
# plot(x1, x2, col = (y+1))
misclass_rates_2 <- c()
missrates_1_case2 <- c()
missrates_10_case2 <- c()
missrates_100_case2 <- c()

# Fitting a random forest model of 1, 10, 100 trees on each of the 1000 datasets
# created above:
for (i in 1:length(tree_no)) {
  for (j in 1:length(list_df_2)) {
    rf_model_2 <- randomForest(as.factor(trlabels)~.,
                               data = as.data.frame.list(list_df_2[[j]]),
                               ntree = tree_no[i],
                               nodesize = 25,
                               keep.forest = TRUE)
    pred_vals <- c()

    pred_vals <- predict(object = rf_model_2, newdata = tedata_2, type = 'class')
    misclass_rates_2 <- missclass(tedata_2[,3], pred_vals)

    if (tree_no[i] == 1) {
      missrates_1_case2 <- rbind(missrates_1_case2, misclass_rates_2)
    }else if (tree_no[i] == 10) {
      missrates_10_case2 <- rbind(missrates_10_case2, misclass_rates_2)
    }
    else{
      missrates_100_case2 <- rbind(missrates_100_case2, misclass_rates_2)
    }
  }
}

new_misclass_df_case2 <- data.frame(Tree1_Misclass = missrates_1_case2,
                                    Tree10_Misclass = missrates_10_case2,
                                    Tree100_Misclass = missrates_100_case2)

row.names(new_misclass_df_case2) <- NULL

mean_tree1_case2 <- mean(new_misclass_df_case2[,1])
mean_tree2_case2 <- mean(new_misclass_df_case2[,2])
mean_tree3_case2 <- mean(new_misclass_df_case2[,3])

variance_tree1_case2 <- var(new_misclass_df_case2[,1])
variance_tree2_case2 <- var(new_misclass_df_case2[,2])
```

```
variance_tree3_case2 <- var(new_misclass_df_case2[,3])

case2_df <- as.data.frame(matrix(ncol = 2, nrow = 0))
colnames(case2_df) <- c('Mean_MisClass_Trees', 'Variance_MisClass_Trees')
case2_df <- rbind(case2_df, data.frame('Mean_MisClass_Trees' = c(mean_tree1_case2,
                                                                 mean_tree2_case2,
                                                                 mean_tree3_case2),
                                       'Variance_MisClass_Trees' = c(variance_tree1_case2,
                                                                     variance_tree2_case2,
                                                                     variance_tree3_case2)))
rownames(case2_df) <- c('1_Tree', '10_Tree', '100_Tree')
case2_df
# Empty list to store 1000 training datasets for second condition:
list_df_3 <- list()

# Creation of 1000 training datasets of size 100 and storing them in the above list:
for (i in 1:1000) {
  x1 <- runif(100)
  x2 <- runif(100)
  trdata_3 <- as.data.frame(matrix(ncol = 3, nrow = 100))
  colnames(trdata_3) <- c('x1', 'x2', 'trlabels')
  trdata_3[,1] <- x1
  trdata_3[,2] <- x2
  y <- as.numeric((x1<0.5 & x2<0.5)
                  | (x1>0.5 & x2>0.5))
  trlabels <- as.factor(y)
  trdata_3[,3] <- trlabels
  list_df_3[[i]] <- (trdata_3)
}

# Generating new Test Data based on condition (x1<0.5 & x2<0.5) | (x1>0.5 & x2>0.5)
set.seed(4567)

x1 <- runif(1000)
x2 <- runif(1000)
tedata_3 <- cbind(x1, x2)

y <- as.numeric((x1<0.5 & x2<0.5)
                | (x1>0.5 & x2>0.5))
telabels <- as.factor(y)
tedata_3 <- cbind(tedata_3, telabels)
tedata_3[,3] <- as.factor(tedata_3[,3])
# plot(x1, x2, col = (y+1))
misclass_rates_3 <- c()
missrates_1_case3 <- c()
missrates_10_case3 <- c()
missrates_100_case3 <- c()

# Fitting a random forest model of 1, 10, 100 trees on each of the 1000 datasets
# created above:
for (i in 1:length(tree_no)) {
  for (j in 1:length(list_df_3)) {
    rf_model_3 <- randomForest(as.factor(trlabels)~.,
```

```
                              data = as.data.frame.list(list_df_3[[j]]),
                              ntree = tree_no[i],
                              nodesize = 12,
                              keep.forest = TRUE)
    pred_vals <- c()

    pred_vals <- predict(object = rf_model_3, newdata = tedata_3, type = 'class')
    misclass_rates_3 <- missclass(tedata_3[,3], pred_vals)

    if (tree_no[i] == 1) {
      missrates_1_case3 <- rbind(missrates_1_case3, misclass_rates_3)
    }else if (tree_no[i] == 10) {
      missrates_10_case3 <- rbind(missrates_10_case3, misclass_rates_3)
    }
    else{
      missrates_100_case3 <- rbind(missrates_100_case3, misclass_rates_3)
    }
  }
}

new_misclass_df_case3 <- data.frame(Tree1_Misclass = missrates_1_case3,
                                    Tree10_Misclass = missrates_10_case3,
                                    Tree100_Misclass = missrates_100_case3)


row.names(new_misclass_df_case3) <- NULL

mean_tree1_case3 <- mean(new_misclass_df_case3[,1])
mean_tree2_case3 <- mean(new_misclass_df_case3[,2])
mean_tree3_case3 <- mean(new_misclass_df_case3[,3])

variance_tree1_case3 <- var(new_misclass_df_case3[,1])
variance_tree2_case3 <- var(new_misclass_df_case3[,2])
variance_tree3_case3 <- var(new_misclass_df_case3[,3])

case3_df <- as.data.frame(matrix(ncol = 2, nrow = 0))
colnames(case3_df) <- c('Mean_MisClass_Trees', 'Variance_MisClass_Trees')
case3_df <- rbind(case3_df, data.frame('Mean_MisClass_Trees' = c(mean_tree1_case3,
                                                                 mean_tree2_case3,
                                                                 mean_tree3_case3),
                                       'Variance_MisClass_Trees' = c(variance_tree1_case3,
                                                                     variance_tree2_case3,
                                                                     variance_tree3_case3)))
rownames(case3_df) <- c('1_Tree', '10_Tree', '100_Tree')
case3_df
mean_var_df <- data.frame(matrix(ncol = 6, nrow = 3))
colnames(mean_var_df) <- c('Mean_Case1',
                           'Mean_Case2',
                           'Mean_Case3',
                           'Var_Case1',
                           'Var_Case2',
                           'Var_Case3')
rownames(mean_var_df) <- c('1Tree',
```

```
                          '10Tree',
                          '100Tree')
mean_var_df[1,] <- c(mean_tree1,
                     mean_tree1_case2,
                     mean_tree1_case3,
                     variance_tree1,
                     variance_tree1_case2,
                     variance_tree1_case3)
mean_var_df[2,] <- c(mean_tree2,
                     mean_tree2_case2,
                     mean_tree2_case3,
                     variance_tree2,
                     variance_tree2_case2,
                     variance_tree2_case3)
mean_var_df[3,] <- c(mean_tree3,
                       mean_tree3_case2,
                       mean_tree3_case3,
                       variance_tree3,
                       variance_tree3_case2,
                       variance_tree3_case3)
mean_var_df

set.seed(1234)
x1<-runif(1000)
x2<-runif(1000)
tedata<-cbind(x1,x2)
y<-as.numeric(x1<x2)
telabels<-as.factor(y)
plot(x1,x2,col=(y+1), main = 'Plot of First Dataset')

set.seed(4567)

x1 <- runif(1000)
x2 <- runif(1000)
tedata_3 <- cbind(x1, x2)

y <- as.numeric((x1<0.5 & x2<0.5)
                | (x1>0.5 & x2>0.5))
telabels <- as.factor(y)

plot(x1,x2,col=(y+1), main = 'Plot of Third Dataset')

# Expectation-Maximization Algorithm
set.seed(1234567890)
max_it <- 100 # max number of EM iterations
min_change <- 0.1 # min change in log lik between two consecutive iterations
n=1000 # number of training points
D=10 # number of dimensions
x <- matrix(nrow=n, ncol=D) # training data

#For 3 classes
true_pi <- vector(length = 3) # true mixing coefficients
true_mu <- matrix(nrow=3, ncol=D) # true conditional distributions
```

```
true_pi=c(1/3, 1/3, 1/3)
true_mu[1,]=c(0.5,0.6,0.4,0.7,0.3,0.8,0.2,0.9,0.1,1)
true_mu[2,]=c(0.5,0.4,0.6,0.3,0.7,0.2,0.8,0.1,0.9,0)
true_mu[3,]=c(0.5,0.5,0.5,0.5,0.5,0.5,0.5,0.5,0.5,0.5)

# plot(true_mu[1,], type="o", col="blue", ylim=c(0,1))
# 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])
  }
}
# #For m = 1 &  n = 1
# bern <- c(mu[2,]^x[1,] #mu^x
#           , (1 - mu[2,])^(1-x[1,])) #(1-mu)^(1-x)
# numerator <- prod(pi[2], bern) #Numerator for 10.5
get_bern <- function(m,n,mu,x){
  bern <- c(mu[m,]^x[n,] #mu^x
            , (1 - mu[m,])^(1-x[n,])) #(1-mu)^(1-x)
  return(bern)
}
get_pi_bern_prod <- function(m, n, pi, mu, x){
  bern <- get_bern(m,n,mu,x)
  pi_bern_prod <- prod(pi[m], bern)
  return(pi_bern_prod)
}

run_em <- function(M){
  set.seed(1234567890)
  w <- matrix(nrow=n, ncol=M) # weights
  pi <- vector(length = M) # mixing coefficients
  mu <- matrix(nrow=M, ncol=D) # conditional distributions
  llik <- c()# 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)
  }

  for(it in 1:max_it) {

    # plot(mu[1,], type="o", col="blue", ylim=c(0,1))
    # if(M == 2){
    #   points(mu[2,], type="o", col="red")
    # }else if(M == 3){
    #   points(mu[2,], type="o", col="red")
    #   points(mu[3,], type="o", col="green")
    # }else if (M == 4){
```

```
  #   points(mu[2,], type="o", col="red")
  #   points(mu[3,], type="o", col="green")
  #   points(mu[4,], type="o", col="yellow")
  # }

  Sys.sleep(0.5)

  #Expectation-step: Computation of the weights
  #The formula of is available in 10.5, w(m) = P(y = m|x, theta)
  for (n in 1:nrow(x)) {
    w_temp <- rep(1,M)
    for (m in 1:M) {
      w_temp[m] <- get_pi_bern_prod(m=m, n=n, pi = pi, mu = mu, x = x)
    }
    w[n,] <- w_temp/sum(w_temp)
  }
  # pi
  #Log likelihood computation: From formula 10.15
  log_lik <- c()
  for (n in 1:nrow(x)) {
    for (m in 1:M) {
      log_lik <- c(log_lik, w[n,m] * (sum((x[n,] * log(mu[m,]+1e-15)) +
                                           ((1-x[n,]) * log(1-mu[m,]+1e-15))) + log(pi[m]))
      )
    }
  }
  llik <- c(llik, sum(log_lik))
  flush.console()
  # Stop if the log-likelihood has not changed significantly
  if(it > 1){
    llik_diff <- llik[it] - llik[it-1]
    # cat("iteration: ", it,
    #     "log likelihood: ", llik[it],
    #     'difference: ' ,llik_diff, "\n")
    if(abs(llik_diff) <= min_change) break
  }
  # else(
  #   cat("iteration: ", it,
  #       "log likelihood: ", llik[it], "\n")
  # )

  #M-step: ML parameter estimation from the data and weights
  #From 10.16
  pi <- colSums(w)/nrow(w)
  for (m in 1:M) {
    for (d in 1:D) {
      mu[m,d] <- as.integer(w[,m] %*% x[,d])/sum(w[,m])
    }
  }
}
ret_list <- list()
plot_df <- as.data.frame(matrix(nrow = 0, ncol = 3))
colnames(plot_df) <- c('Dim', 'Mu_Val', 'Cluster')
```

```
  for (i in 1:nrow(mu)) {
    plot_df <- rbind(plot_df, data.frame(
      Dim = seq(1:10), Mu_Val = mu[i,],
      Cluster = i
      ))
  }
  plot_df$Cluster <- as.factor(plot_df$Cluster)
  ret_list$plot_df <- plot_df
  llik_df <- as.data.frame(matrix(nrow = 0, ncol = 2))
  colnames(llik_df) <- c('Iter', 'Likelihood')
  for (i in 1:length(llik)) {
    llik_df <- rbind(llik_df, data.frame(
      Iter = i, Likelihood = llik[i]
    ))
  }
  ret_list$llik <- llik_df
  ret_list$pi <- pi
  # pi
  # mu
  return(ret_list)
}

M = 2
ret_list <- run_em(M)
pi_2 <- ret_list$pi
mu_plot_2 <- ggplot(ret_list$plot_df, aes(x = Dim, y = Mu_Val)) +
  geom_line(aes(color = Cluster)) +
  ggtitle(paste('Cluster size:',M)) +
  scale_x_continuous(breaks = seq(1,10,1))
llik_plot_2 <- ggplot(ret_list$llik, aes(x = Iter, y = Likelihood))+
  geom_line() + ggtitle(paste('Cluster size:',M))

M = 3
ret_list <- run_em(M)
pi_3 <- ret_list$pi
mu_plot_3 <- ggplot(ret_list$plot_df, aes(x = Dim, y = Mu_Val)) +
  geom_line(aes(color = Cluster)) +
  ggtitle(paste('Cluster size:',M)) +
  scale_x_continuous(breaks = seq(1,10,1))
llik_plot_3 <- ggplot(ret_list$llik, aes(x = Iter, y = Likelihood))+
  geom_line()+ ggtitle(paste('Cluster size:',M))

M = 4
ret_list <- run_em(M)
pi_4 <- ret_list$pi
mu_plot_4 <- ggplot(ret_list$plot_df, aes(x = Dim, y = Mu_Val)) +
  geom_line(aes(color = Cluster)) +
  ggtitle(paste('Cluster size:',M)) +
  scale_x_continuous(breaks = seq(1,10,1))
llik_plot_4 <- ggplot(ret_list$llik, aes(x = Iter, y = Likelihood))+
  geom_line()+ ggtitle(paste('Cluster size:',M))

grid.arrange(mu_plot_2, mu_plot_3, mu_plot_4, nrow = 2,
```

```
            top = 'Plots for Mu for different Clusters')
cat('pi value for 2 clusters:', pi_2)
cat('pi value for 3 clusters:', pi_3)
cat('pi value for 4 clusters:', pi_4)
grid.arrange(llik_plot_2, llik_plot_3, llik_plot_4, nrow = 2,
            top = 'Plots Log-likelihood for different clusters')
```