machine learning(732A99) lab1 Block 2

Anubhav Dikshit(anudi287)

26 November 2018

Contents

Assignment 1 **1**
Loading The Libraries 1

1.Your task is to evaluate the performance of Adaboost classification trees and random forests **2**
Loading Input files 2
Splitting into Train and Test with 66% and 33% ratio. 2
Trainning the Model 2
2 Your task is to implement the EM algorithm for mixtures of multivariate Benoulli distributions. Please use the template in the next page to solve the assignment. Then, use your implementation to show what happens when your mixture models has too few and too many components, i.e. set K = 2, 3,4 and compare results. Please provide a short explanation as well. 3

Apendix **10**

Assignment 1

Loading The Libraries

```
## Warning: unable to access index for repository http://www.stats.ox.ac.uk/pub/RWin/bin/windows/contri
##   cannot open URL 'http://www.stats.ox.ac.uk/pub/RWin/bin/windows/contrib/3.5/PACKAGES'

## package 'partykit' successfully unpacked and MD5 sums checked

## Warning: cannot remove prior installation of package 'partykit'

## package 'mboost' successfully unpacked and MD5 sums checked

## Warning: cannot remove prior installation of package 'mboost'

##
## The downloaded binary packages are in
##  C:\Users\anubh\AppData\Local\Temp\RtmpGCBzZP\downloaded_packages

## Warning in p_install(package, character.only = TRUE, ...):

## Warning in library(package, lib.loc = lib.loc, character.only = TRUE,
## logical.return = TRUE, : there is no package called 'mboost'

## Warning in pacman::p_load(xlsx, mboost, randomForest, dplyr, ggplot2): Failed to install/load:
## mboost
```

1.Your task is to evaluate the performance of Adaboost classification trees and random forests

on the spam data. Specifically, provide a plot showing the error rates when the number of trees considered are 10,20,..,100. To estimate the error rates, use 2/3 of the data for training and 1/3 as hold-out test data.

Loading Input files

```
spam_data <- read.csv(file = "spambase.data", header = FALSE)
colnames(spam_data)[58] <- "Spam"
spam_data$Spam <- factor(spam_data$Spam, levels = c(0,1), labels = c("0", "1"))
```

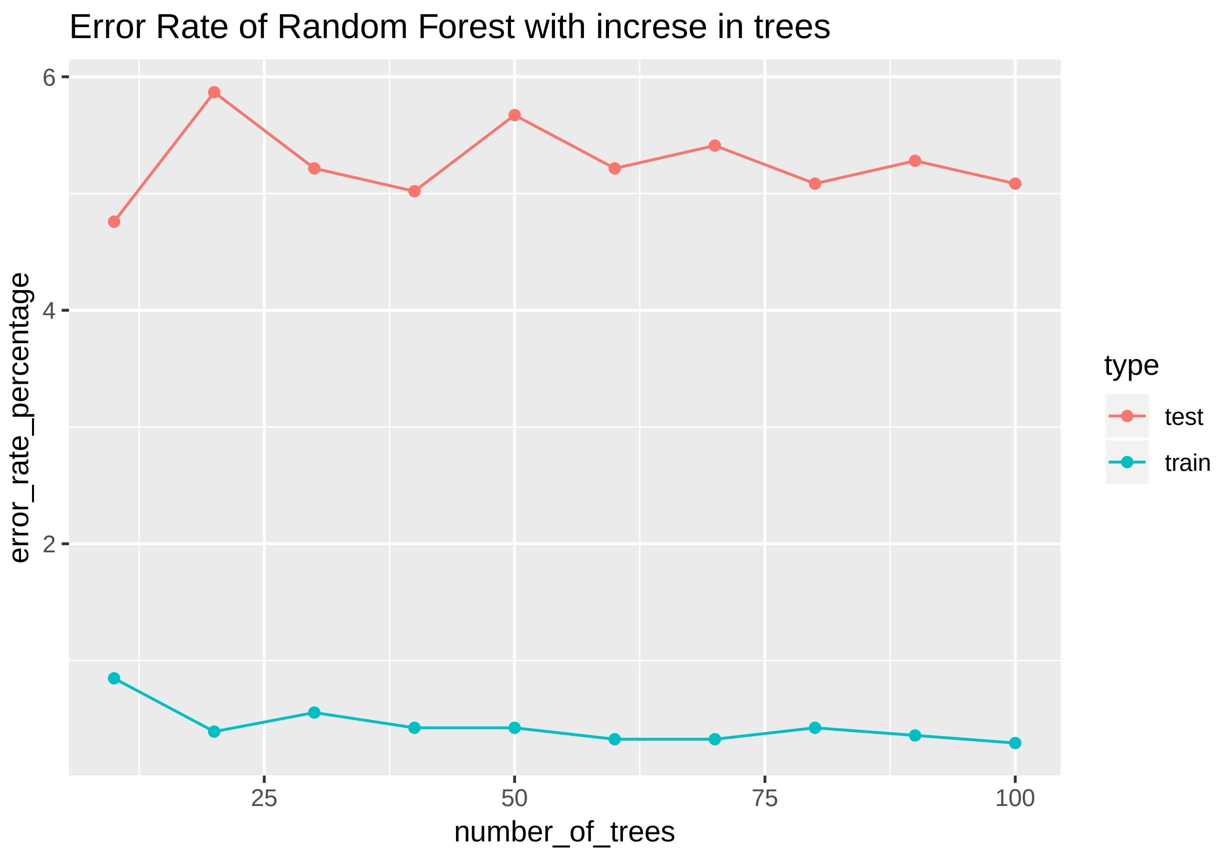
Splitting into Train and Test with 66% and 33% ratio.

```
set.seed(12345)

n =  NROW(spam_data)
id = sample(1:n, floor(n*(2/3)))
train = spam_data[id,]
test = spam_data[-id,]
```

Trainning the Model

Adaboost with varying depth

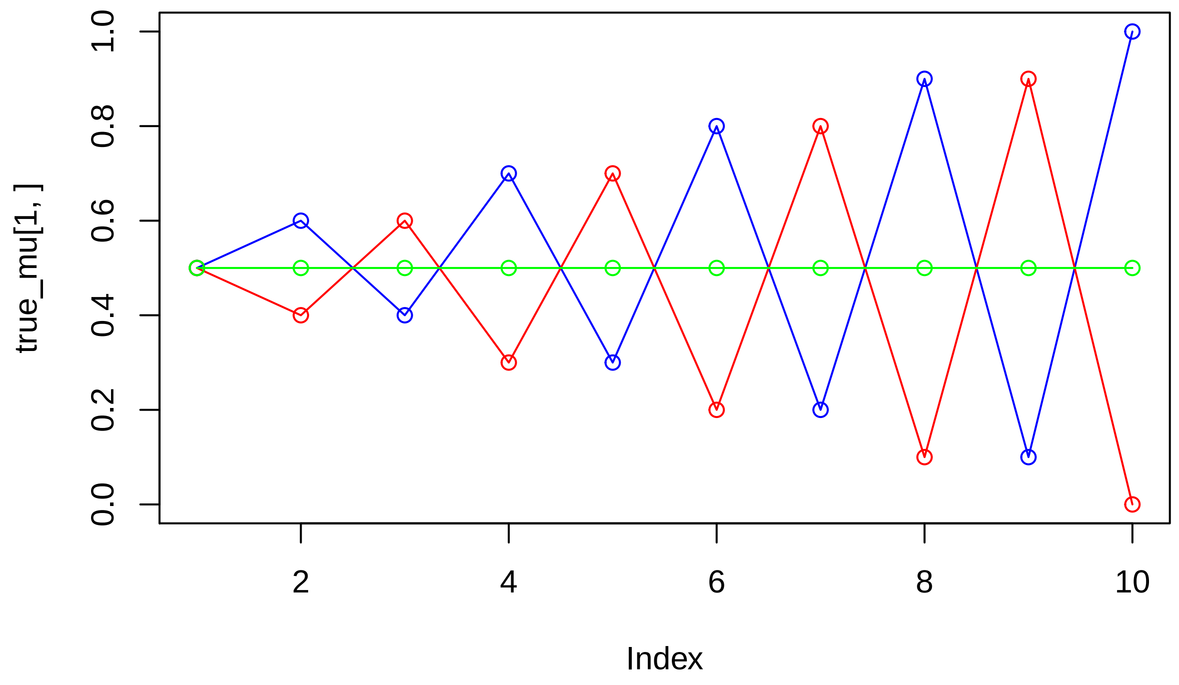
```
final_result <- NULL
for(i in seq(from = 10, to = 100, by = 10)){

# ada_model <- mboost::blackboost(Spam~.,
#                                  data = train,
#                                  family = AdaExp(),
#                                  tree_controls = partykit::ctree_control(maxdepth = i))

forest_model <- randomForest(Spam~., data = train, ntree = i)


#ada_model_predict <- predict(ada_model, newdata = temp, type = c("response"))
train_forest_model_predict <- predict(forest_model, newdata = train)
train_predict_correct <- ifelse(train$Spam == train_forest_model_predict, 1, 0)

test_forest_model_predict <- predict(forest_model, newdata = test)
test_predict_correct <- ifelse(test$Spam == test_forest_model_predict, 1, 0)


train_score <- sum(train_predict_correct)/NROW(train)
test_score <- sum(test_predict_correct)/NROW(test)

iteration_result <- data.frame(number_of_trees = i,
                                accuracy = c(train_score, test_score),
                                type  = c("train", "test"))


final_result <- rbind(iteration_result, final_result)
}

final_result$error_rate_percentage <- 100*(1 - final_result$accuracy)
ggplot(data = final_result, aes(x = number_of_trees, y = error_rate_percentage, group = type, color = ty
```



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

```
set.seed(1234567890)
max_it <- 100 # max number of EM iterations
min_change <- 0.1 # min change in log likelihood between two consecutive EM iterations
N=1000 # number of training points
D=10 # number of dimensions
x <- matrix(nrow=N, ncol=D) # training data
true_pi <- vector(length = 3) # true mixing coefficients
```

```r
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")
```



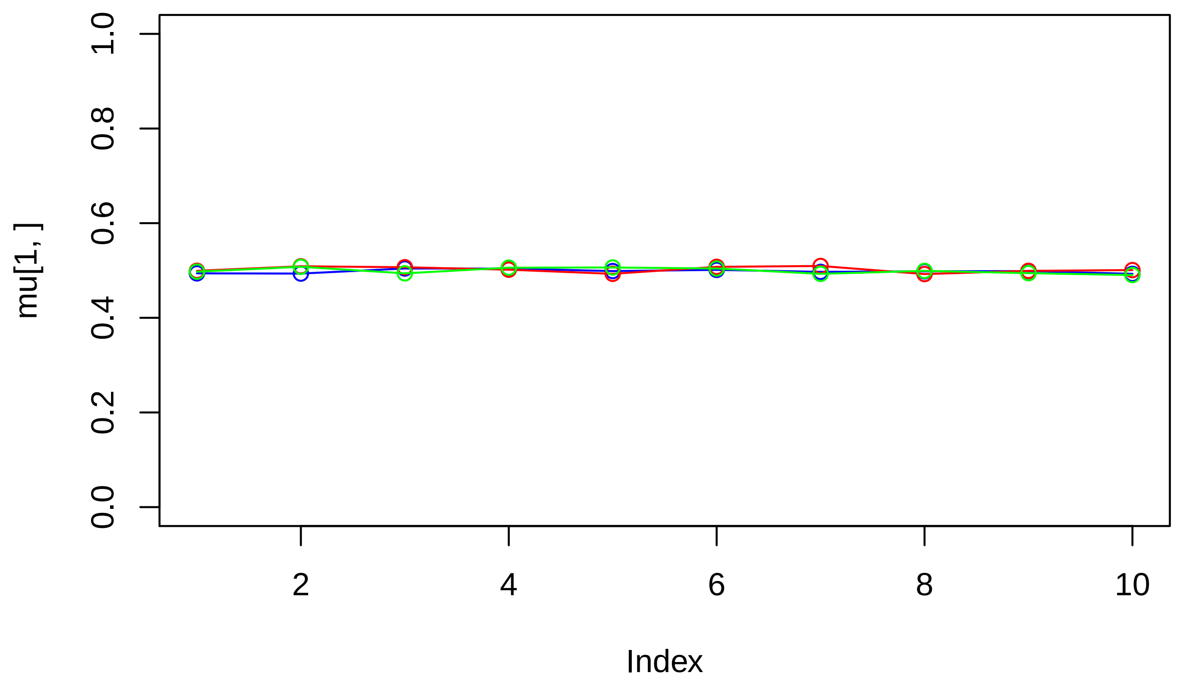
```r
# Producing the training data
for(n in 1:N) {
k <- sample(1:3,1,prob=true_pi)
for(d in 1:D) {
x[n,d] <- rbinom(1,1,true_mu[k,d])
}
}
K=3 # number of guessed components
z <- matrix(nrow=N, ncol=K) # fractional component assignments
pi <- vector(length = K) # mixing coefficients
mu <- matrix(nrow=K, ncol=D) # conditional distributions
llik <- vector(length = max_it) # log likelihood of the EM iterations
# Random initialization of the paramters
pi <- runif(K,0.49,0.51)
pi <- pi / sum(pi)
for(k in 1:K) {
mu[k,] <- runif(D,0.49,0.51)
}
pi
```

```
## [1] 0.3326090 0.3336558 0.3337352
```

```r
mu
```

```
##           [,1]      [,2]      [,3]      [,4]      [,5]      [,6]      [,7]
## [1,] 0.4939877 0.4935375 0.5042511 0.5040286 0.4987810 0.5012754 0.4971036
## [2,] 0.4993719 0.5088453 0.5068730 0.5016720 0.4929275 0.5077146 0.5095075
## [3,] 0.4975302 0.5077926 0.4939841 0.5059821 0.5063490 0.5041462 0.4929400
##           [,8]      [,9]     [,10]
## [1,] 0.4982144 0.4987654 0.4929075
## [2,] 0.4924574 0.4992470 0.5008651
## [3,] 0.4992362 0.4943482 0.4903974
```

```r
for(it in 1:max_it) {
plot(mu[1,], type="o", col="blue", ylim=c(0,1))
points(mu[2,], type="o", col="red")
points(mu[3,], type="o", col="green")
#points(mu[4,], type="o", col="yellow")
Sys.sleep(0.5)
# E-step: Computation of the fractional component assignments
# Your code here
#Log likelihood computation.
# Your code here
cat("iteration: ", it, "log likelihood: ", llik[it], "\n")
flush.console()
# Stop if the lok likelihood has not changed significantly
# Your code here
#M-step: ML parameter estimation from the data and fractional component assignments
# Your code here
}
```



```
## iteration:  1 log likelihood:  FALSE
## iteration:  2 log likelihood:  FALSE
## iteration:  3 log likelihood:  FALSE
## iteration:  4 log likelihood:  FALSE
## iteration:  5 log likelihood:  FALSE
## iteration:  6 log likelihood:  FALSE
## iteration:  7 log likelihood:  FALSE
## iteration:  8 log likelihood:  FALSE
## iteration:  9 log likelihood:  FALSE
## iteration:  10 log likelihood:  FALSE
## iteration:  11 log likelihood:  FALSE
## iteration:  12 log likelihood:  FALSE
## iteration:  13 log likelihood:  FALSE
## iteration:  14 log likelihood:  FALSE
## iteration:  15 log likelihood:  FALSE
## iteration:  16 log likelihood:  FALSE
## iteration:  17 log likelihood:  FALSE
## iteration:  18 log likelihood:  FALSE
## iteration:  19 log likelihood:  FALSE
## iteration:  20 log likelihood:  FALSE
## iteration:  21 log likelihood:  FALSE
## iteration:  22 log likelihood:  FALSE
## iteration:  23 log likelihood:  FALSE
## iteration:  24 log likelihood:  FALSE
## iteration:  25 log likelihood:  FALSE
## iteration:  26 log likelihood:  FALSE
## iteration:  27 log likelihood:  FALSE
## iteration:  28 log likelihood:  FALSE
## iteration:  29 log likelihood:  FALSE
## iteration:  30 log likelihood:  FALSE
## iteration:  31 log likelihood:  FALSE
## iteration:  32 log likelihood:  FALSE
## iteration:  33 log likelihood:  FALSE
## iteration:  34 log likelihood:  FALSE
## iteration:  35 log likelihood:  FALSE
## iteration:  36 log likelihood:  FALSE
## iteration:  37 log likelihood:  FALSE
## iteration:  38 log likelihood:  FALSE
## iteration:  39 log likelihood:  FALSE
## iteration:  40 log likelihood:  FALSE
## iteration:  41 log likelihood:  FALSE
## iteration:  42 log likelihood:  FALSE
## iteration:  43 log likelihood:  FALSE
## iteration:  44 log likelihood:  FALSE
## iteration:  45 log likelihood:  FALSE
## iteration:  46 log likelihood:  FALSE
## iteration:  47 log likelihood:  FALSE
## iteration:  48 log likelihood:  FALSE
## iteration:  49 log likelihood:  FALSE
## iteration:  50 log likelihood:  FALSE
## iteration:  51 log likelihood:  FALSE
## iteration:  52 log likelihood:  FALSE
## iteration:  53 log likelihood:  FALSE
## iteration:  54 log likelihood:  FALSE
## iteration:  55 log likelihood:  FALSE
## iteration:  56 log likelihood:  FALSE
## iteration:  57 log likelihood:  FALSE
## iteration:  58 log likelihood:  FALSE
## iteration:  59 log likelihood:  FALSE
## iteration:  60 log likelihood:  FALSE
## iteration:  61 log likelihood:  FALSE
## iteration:  62 log likelihood:  FALSE
## iteration:  63 log likelihood:  FALSE
## iteration:  64 log likelihood:  FALSE
## iteration:  65 log likelihood:  FALSE
## iteration:  66 log likelihood:  FALSE
## iteration:  67 log likelihood:  FALSE
## iteration:  68 log likelihood:  FALSE
## iteration:  69 log likelihood:  FALSE
## iteration:  70 log likelihood:  FALSE
## iteration:  71 log likelihood:  FALSE
## iteration:  72 log likelihood:  FALSE
## iteration:  73 log likelihood:  FALSE
## iteration:  74 log likelihood:  FALSE
## iteration:  75 log likelihood:  FALSE
## iteration:  76 log likelihood:  FALSE
## iteration:  77 log likelihood:  FALSE
## iteration:  78 log likelihood:  FALSE
## iteration:  79 log likelihood:  FALSE
## iteration:  80 log likelihood:  FALSE
## iteration:  81 log likelihood:  FALSE
## iteration:  82 log likelihood:  FALSE
## iteration:  83 log likelihood:  FALSE
## iteration:  84 log likelihood:  FALSE
## iteration:  85 log likelihood:  FALSE
## iteration:  86 log likelihood:  FALSE
## iteration:  87 log likelihood:  FALSE
## iteration:  88 log likelihood:  FALSE
## iteration:  89 log likelihood:  FALSE
## iteration:  90 log likelihood:  FALSE
## iteration:  91 log likelihood:  FALSE

## iteration:  92 log likelihood:  FALSE

## iteration:  93 log likelihood:  FALSE

## iteration:  94 log likelihood:  FALSE

## iteration:  95 log likelihood:  FALSE

## iteration:  96 log likelihood:  FALSE

## iteration:  97 log likelihood:  FALSE

## iteration:  98 log likelihood:  FALSE

## iteration:  99 log likelihood:  FALSE

## iteration:  100 log likelihood:  FALSE
```

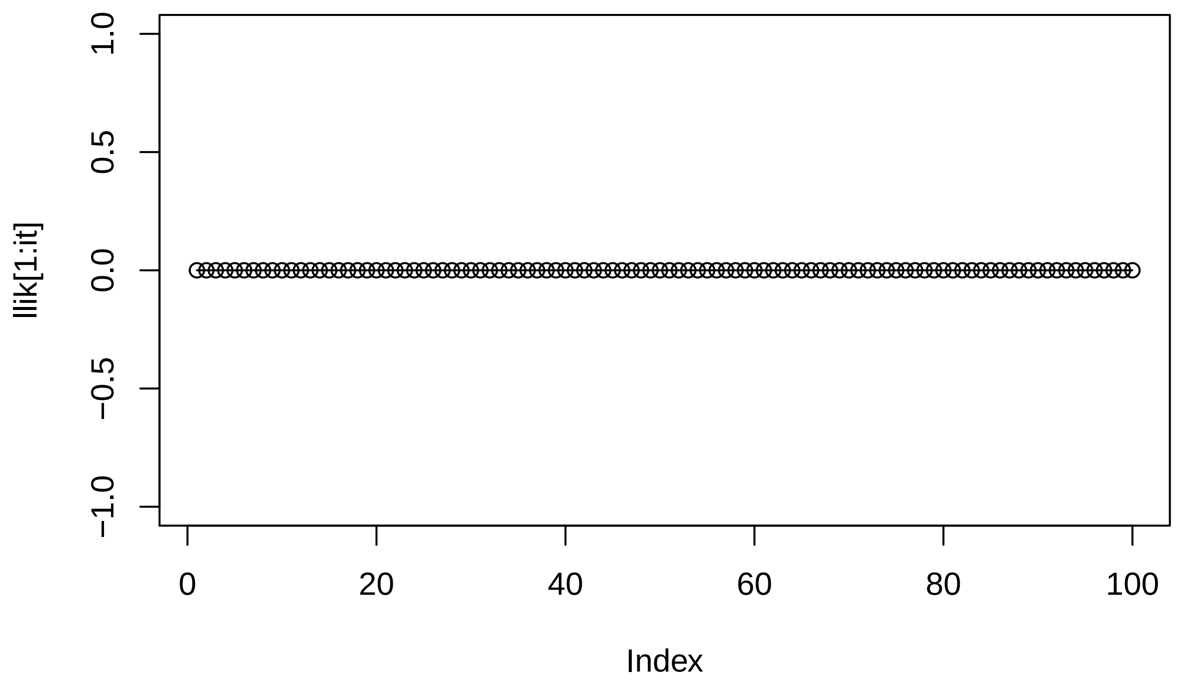
```
pi
```

```
## [1] 0.3326090 0.3336558 0.3337352
```

```
mu
```

```
##           [,1]      [,2]      [,3]      [,4]      [,5]      [,6]      [,7]
## [1,] 0.4939877 0.4935375 0.5042511 0.5040286 0.4987810 0.5012754 0.4971036
## [2,] 0.4993719 0.5088453 0.5068730 0.5016720 0.4929275 0.5077146 0.5095075
## [3,] 0.4975302 0.5077926 0.4939841 0.5059821 0.5063490 0.5041462 0.4929400
##           [,8]      [,9]     [,10]
## [1,] 0.4982144 0.4987654 0.4929075
## [2,] 0.4924574 0.4992470 0.5008651
## [3,] 0.4992362 0.4943482 0.4903974
```

```
plot(llik[1:it], type="o")
```



Apendix

```
knitr::opts_chunk$set(echo = TRUE)
if (!require("pacman")) install.packages("pacman")
pacman::p_load(xlsx, mboost, randomForest, dplyr, ggplot2)

options(scipen = 999)

spam_data <- read.csv(file = "spambase.data", header = FALSE)
colnames(spam_data)[58] <- "Spam"
spam_data$Spam <- factor(spam_data$Spam, levels = c(0,1), labels = c("0", "1"))
set.seed(12345)

n =  NROW(spam_data)
id = sample(1:n, floor(n*(2/3)))
train = spam_data[id,]
test = spam_data[-id,]

final_result <- NULL
for(i in seq(from = 10, to = 100, by = 10)){

# ada_model <- mboost::blackboost(Spam~.,
#                                   data = train,
```

```r
#                                   family = AdaExp(),
#                                   tree_controls = partykit::ctree_control(maxdepth = i))

forest_model <- randomForest(Spam~., data = train, ntree = i)


#ada_model_predict <- predict(ada_model, newdata = temp, type = c("response"))
train_forest_model_predict <- predict(forest_model, newdata = train)
train_predict_correct <- ifelse(train$Spam == train_forest_model_predict, 1, 0)

test_forest_model_predict <- predict(forest_model, newdata = test)
test_predict_correct <- ifelse(test$Spam == test_forest_model_predict, 1, 0)


train_score <- sum(train_predict_correct)/NROW(train)
test_score <- sum(test_predict_correct)/NROW(test)

iteration_result <- data.frame(number_of_trees = i,
                               accuracy = c(train_score, test_score),
                               type  = c("train", "test"))


final_result <- rbind(iteration_result, final_result)
}

final_result$error_rate_percentage <- 100*(1 - final_result$accuracy)
ggplot(data = final_result, aes(x = number_of_trees, y = error_rate_percentage, group = type, color = ty

set.seed(1234567890)
max_it <- 100 # max number of EM iterations
min_change <- 0.1 # min change in log likelihood between two consecutive EM iterations
N=1000 # number of training points
D=10 # number of dimensions
x <- matrix(nrow=N, ncol=D) # training data
true_pi <- vector(length = 3) # true mixing coefficients
true_mu <- matrix(nrow=3, ncol=D) # true conditional distributions
true_pi=c(1/3, 1/3, 1/3)
true_mu[1,]=c(0.5,0.6,0.4,0.7,0.3,0.8,0.2,0.9,0.1,1)
true_mu[2,]=c(0.5,0.4,0.6,0.3,0.7,0.2,0.8,0.1,0.9,0)
true_mu[3,]=c(0.5,0.5,0.5,0.5,0.5,0.5,0.5,0.5,0.5,0.5)
plot(true_mu[1,], type="o", col="blue", ylim=c(0,1))
points(true_mu[2,], type="o", col="red")
points(true_mu[3,], type="o", col="green")
# Producing the training data
for(n in 1:N) {
k <- sample(1:3,1,prob=true_pi)
for(d in 1:D) {
x[n,d] <- rbinom(1,1,true_mu[k,d])
}
}
K=3 # number of guessed components
z <- matrix(nrow=N, ncol=K) # fractional component assignments
pi <- vector(length = K) # mixing coefficients
```

```r
mu <- matrix(nrow=K, ncol=D) # conditional distributions
llik <- vector(length = max_it) # log likelihood of the EM iterations
# Random initialization of the paramters
pi <- runif(K,0.49,0.51)
pi <- pi / sum(pi)
for(k in 1:K) {
mu[k,] <- runif(D,0.49,0.51)
}
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")
points(mu[3,], type="o", col="green")
#points(mu[4,], type="o", col="yellow")
Sys.sleep(0.5)
# E-step: Computation of the fractional component assignments
# Your code here
#Log likelihood computation.
# Your code here
cat("iteration: ", it, "log likelihood: ", llik[it], "\n")
flush.console()
# Stop if the lok likelihood has not changed significantly
# Your code here
#M-step: ML parameter estimation from the data and fractional component assignments
# Your code here
}
pi
mu
plot(llik[1:it], type="o")
```