Assignment 1 block 2

Karthikeyan Devarajan - Karde799

1.Ensemble Methods

The classification data is done such that the training data contains 2/3rd of the total data and test data contains 1/3rd of the data.

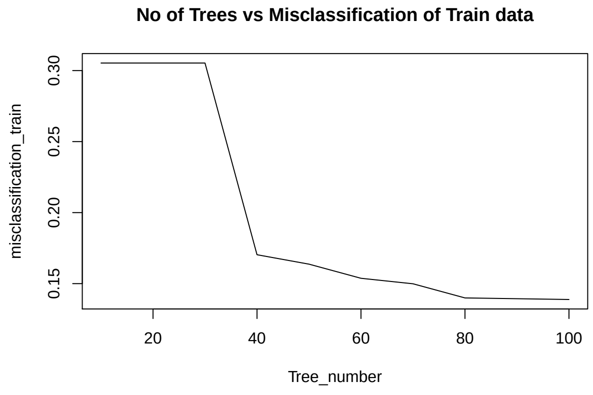
1.1.Adaboost Classification Tree

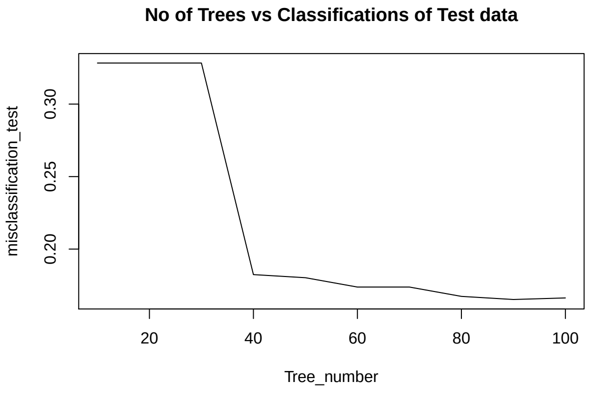
```
## Missclassification Rate of Train data
##  0.1388274
```

```
##  No_of_Trees
##  100
```

```
## Missclassification Rate of Test data
##  0.1652361
```

```
## Number_of_Trees
##  90
```



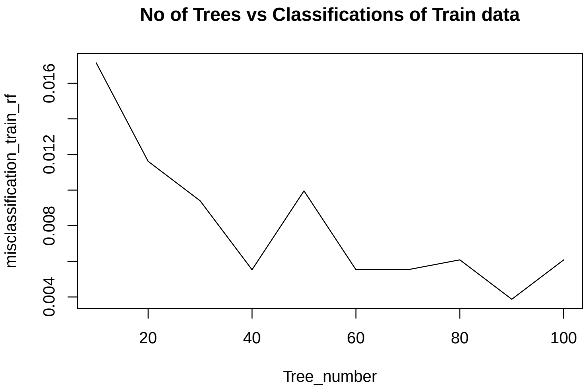


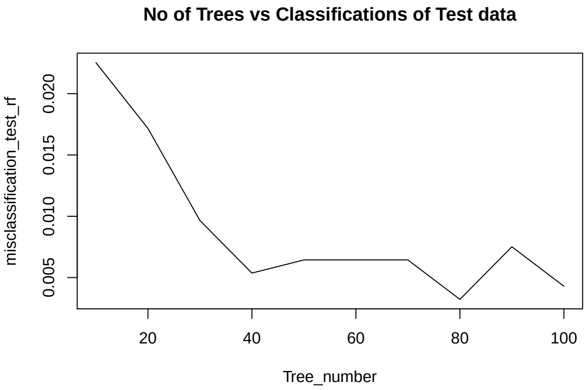
1.2.Random Forest

```
## Missclassification Rate of Train data
##  0.003871681
```

```
##  Number_of_Trees
##  90
## Miss classification Rate of Test data
##  0.003218884
```

```
## Number_of_Trees
##  80
```


No of Trees vs Classifications of Train data

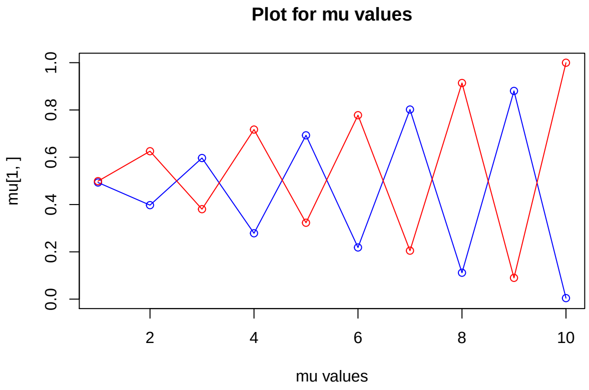

No of Trees vs Classifications of Test data

2.Mixture Model

Maximum likelihood focuses on determining the parameters to maximizes the probability of the given data. In Bernoulli equation pi and mu are the estimators.

i) When K = 2

```
## iteration:  1 log likelihood:  -7623.897
## iteration:  2 log likelihood:  -7610.745
## iteration:  3 log likelihood:  -7463.445
## iteration:  4 log likelihood:  -6575.121
## iteration:  5 log likelihood:  -5731.559
## iteration:  6 log likelihood:  -5656.174
## iteration:  7 log likelihood:  -5648.904
## iteration:  8 log likelihood:  -5646.139
## iteration:  9 log likelihood:  -5644.608
## iteration:  10 log likelihood:  -5643.615
## iteration:  11 log likelihood:  -5642.913
## iteration:  12 log likelihood:  -5642.386
## iteration:  13 log likelihood:  -5641.977
## iteration:  14 log likelihood:  -5641.649
## iteration:  15 log likelihood:  -5641.382
## iteration:  16 log likelihood:  -5641.161
## iteration:  17 log likelihood:  -5640.975
## iteration:  18 log likelihood:  -5640.819
## iteration:  19 log likelihood:  -5640.685
## iteration:  20 log likelihood:  -5640.571
## iteration:  21 log likelihood:  -5640.473
```

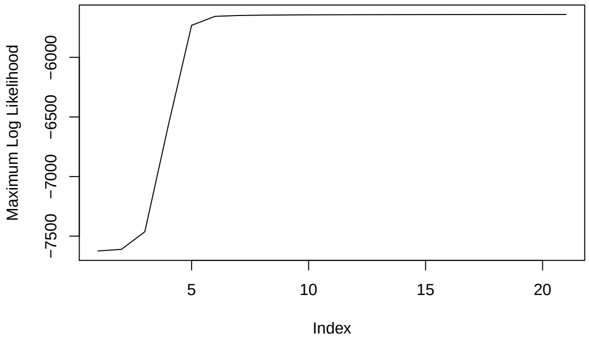


```
## The pi value for two components are
##  0.5110531 0.4889469
```

```
## The mu value for two components are
```

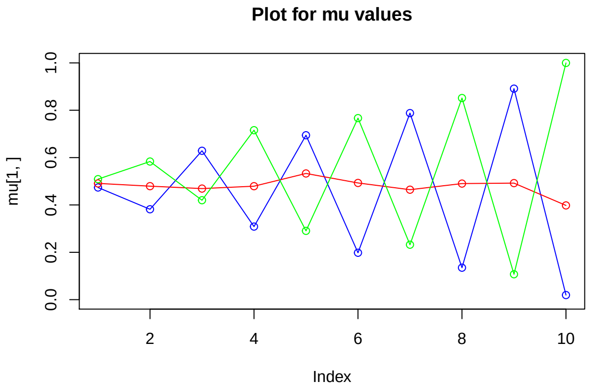
```
##           [,1]      [,2]      [,3]      [,4]      [,5]      [,6]      [,7]
## [1,] 0.4931735 0.3974606 0.5967811 0.2785480 0.6927917 0.2184957 0.8018491
## [2,] 0.4989543 0.6255823 0.3804363 0.7171478 0.3230343 0.7778699 0.2049559
##           [,8]       [,9]       [,10]
## [1,] 0.1116477 0.88054439 0.004290353
## [2,] 0.9140913 0.08997919 0.999714736
```

```
## The number of iteration is 21 and and the maximum likelihood value is -5640.473
```



ii) K = 3

```
## iteration:  1 log likelihood:  -8029.723
## iteration:  2 log likelihood:  -8027.183
## iteration:  3 log likelihood:  -8024.696
## iteration:  4 log likelihood:  -8005.631
## iteration:  5 log likelihood:  -7877.606
## iteration:  6 log likelihood:  -7403.513
## iteration:  7 log likelihood:  -6936.919
## iteration:  8 log likelihood:  -6818.582
## iteration:  9 log likelihood:  -6791.377
## iteration:  10 log likelihood:  -6780.713
## iteration:  11 log likelihood:  -6774.958
## iteration:  12 log likelihood:  -6771.261
## iteration:  13 log likelihood:  -6768.606
## iteration:  14 log likelihood:  -6766.535
## iteration:  15 log likelihood:  -6764.815
## iteration:  16 log likelihood:  -6763.316
## iteration:  17 log likelihood:  -6761.967
## iteration:  18 log likelihood:  -6760.727
## iteration:  19 log likelihood:  -6759.572
## iteration:  20 log likelihood:  -6758.491
## iteration:  21 log likelihood:  -6757.475
## iteration:  22 log likelihood:  -6756.521
## iteration:  23 log likelihood:  -6755.625
## iteration:  24 log likelihood:  -6754.784
## iteration:  25 log likelihood:  -6753.996
## iteration:  26 log likelihood:  -6753.26
## iteration:  27 log likelihood:  -6752.571
## iteration:  28 log likelihood:  -6751.928
## iteration:  29 log likelihood:  -6751.328
## iteration:  30 log likelihood:  -6750.768
## iteration:  31 log likelihood:  -6750.246
## iteration:  32 log likelihood:  -6749.758
## iteration:  33 log likelihood:  -6749.304
## iteration:  34 log likelihood:  -6748.88
## iteration:  35 log likelihood:  -6748.484
## iteration:  36 log likelihood:  -6748.114
## iteration:  37 log likelihood:  -6747.767
## iteration:  38 log likelihood:  -6747.444
## iteration:  39 log likelihood:  -6747.14
## iteration:  40 log likelihood:  -6746.856
## iteration:  41 log likelihood:  -6746.589
## iteration:  42 log likelihood:  -6746.338
## iteration:  43 log likelihood:  -6746.102
## iteration:  44 log likelihood:  -6745.88
## iteration:  45 log likelihood:  -6745.67
## iteration:  46 log likelihood:  -6745.472
## iteration:  47 log likelihood:  -6745.285
## iteration:  48 log likelihood:  -6745.108
## iteration:  49 log likelihood:  -6744.939
## iteration:  50 log likelihood:  -6744.78
## iteration:  51 log likelihood:  -6744.627
## iteration:  52 log likelihood:  -6744.483
## iteration:  53 log likelihood:  -6744.344
## iteration:  54 log likelihood:  -6744.212
## iteration:  55 log likelihood:  -6744.086
## iteration:  56 log likelihood:  -6743.964
## iteration:  57 log likelihood:  -6743.848
## iteration:  58 log likelihood:  -6743.736
## iteration:  59 log likelihood:  -6743.628
## iteration:  60 log likelihood:  -6743.524
## iteration:  61 log likelihood:  -6743.423
## iteration:  62 log likelihood:  -6743.326
```

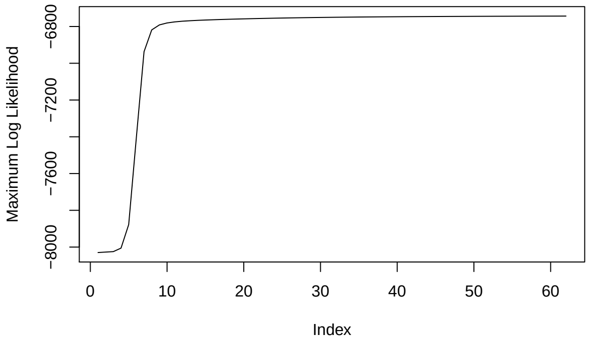


```
## The pi value for three components are
##  0.3259592 0.3044579 0.3695828
```

```
## The mu value for three components are
```

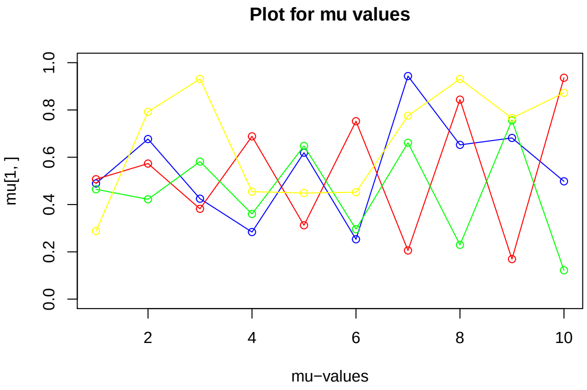
```
##           [,1]      [,2]      [,3]      [,4]      [,5]      [,6]      [,7]
## [1,] 0.4737193 0.3817120 0.6288021 0.3086143 0.6943731 0.1980896 0.7879447
## [2,] 0.4909874 0.4793213 0.4691560 0.4791793 0.5329895 0.4928830 0.4643990
## [3,] 0.5089571 0.5834802 0.4199272 0.7157107 0.2905703 0.7667258 0.2320784
##           [,8]      [,9]      [,10]
## [1,] 0.1349651 0.8912534 0.01937869
## [2,] 0.4902682 0.4922194 0.39798407
## [3,] 0.8516111 0.1072226 0.99981353
```

```
## The number of iteration is 62 and and the maximum likelihood value is -6743.326
```



iii)K = 4

```
## iteration:  1 log likelihood:  -8318.685
## iteration:  2 log likelihood:  -8279.945
## iteration:  3 log likelihood:  -8275.424
## iteration:  4 log likelihood:  -8251.402
## iteration:  5 log likelihood:  -8142.88
## iteration:  6 log likelihood:  -7853.758
## iteration:  7 log likelihood:  -7559.551
## iteration:  8 log likelihood:  -7414.735
## iteration:  9 log likelihood:  -7342.446
## iteration:  10 log likelihood:  -7301.347
## iteration:  11 log likelihood:  -7274.404
## iteration:  12 log likelihood:  -7253.235
## iteration:  13 log likelihood:  -7235.137
## iteration:  14 log likelihood:  -7219.764
## iteration:  15 log likelihood:  -7207.115
## iteration:  16 log likelihood:  -7196.939
## iteration:  17 log likelihood:  -7188.786
## iteration:  18 log likelihood:  -7182.174
## iteration:  19 log likelihood:  -7176.69
## iteration:  20 log likelihood:  -7172.018
## iteration:  21 log likelihood:  -7167.934
## iteration:  22 log likelihood:  -7164.28
## iteration:  23 log likelihood:  -7160.949
## iteration:  24 log likelihood:  -7157.864
## iteration:  25 log likelihood:  -7154.968
## iteration:  26 log likelihood:  -7152.219
## iteration:  27 log likelihood:  -7149.582
## iteration:  28 log likelihood:  -7147.024
## iteration:  29 log likelihood:  -7144.52
## iteration:  30 log likelihood:  -7142.043
## iteration:  31 log likelihood:  -7139.567
## iteration:  32 log likelihood:  -7137.072
## iteration:  33 log likelihood:  -7134.536
## iteration:  34 log likelihood:  -7131.941
## iteration:  35 log likelihood:  -7129.275
## iteration:  36 log likelihood:  -7126.53
## iteration:  37 log likelihood:  -7123.704
## iteration:  38 log likelihood:  -7120.803
## iteration:  39 log likelihood:  -7117.839
## iteration:  40 log likelihood:  -7114.831
## iteration:  41 log likelihood:  -7111.803
## iteration:  42 log likelihood:  -7108.782
## iteration:  43 log likelihood:  -7105.795
## iteration:  44 log likelihood:  -7102.87
## iteration:  45 log likelihood:  -7100.029
## iteration:  46 log likelihood:  -7097.291
## iteration:  47 log likelihood:  -7094.668
## iteration:  48 log likelihood:  -7092.169
## iteration:  49 log likelihood:  -7089.796
## iteration:  50 log likelihood:  -7087.549
## iteration:  51 log likelihood:  -7085.421
## iteration:  52 log likelihood:  -7083.406
## iteration:  53 log likelihood:  -7081.494
## iteration:  54 log likelihood:  -7079.673
## iteration:  55 log likelihood:  -7077.934
## iteration:  56 log likelihood:  -7076.264
## iteration:  57 log likelihood:  -7074.653
## iteration:  58 log likelihood:  -7073.089
## iteration:  59 log likelihood:  -7071.564
## iteration:  60 log likelihood:  -7070.069
## iteration:  61 log likelihood:  -7068.595
## iteration:  62 log likelihood:  -7067.135
## iteration:  63 log likelihood:  -7065.685
## iteration:  64 log likelihood:  -7064.24
## iteration:  65 log likelihood:  -7062.795
## iteration:  66 log likelihood:  -7061.348
## iteration:  67 log likelihood:  -7059.897
## iteration:  68 log likelihood:  -7058.441
## iteration:  69 log likelihood:  -7056.981
## iteration:  70 log likelihood:  -7055.515
## iteration:  71 log likelihood:  -7054.047
## iteration:  72 log likelihood:  -7052.576
## iteration:  73 log likelihood:  -7051.105
## iteration:  74 log likelihood:  -7049.635
## iteration:  75 log likelihood:  -7048.17
## iteration:  76 log likelihood:  -7046.71
## iteration:  77 log likelihood:  -7045.259
## iteration:  78 log likelihood:  -7043.819
## iteration:  79 log likelihood:  -7042.391
## iteration:  80 log likelihood:  -7040.978
## iteration:  81 log likelihood:  -7039.582
## iteration:  82 log likelihood:  -7038.204
## iteration:  83 log likelihood:  -7036.846
## iteration:  84 log likelihood:  -7035.509
## iteration:  85 log likelihood:  -7034.194
## iteration:  86 log likelihood:  -7032.902
## iteration:  87 log likelihood:  -7031.633
## iteration:  88 log likelihood:  -7030.389
## iteration:  89 log likelihood:  -7029.17
## iteration:  90 log likelihood:  -7027.976
## iteration:  91 log likelihood:  -7026.807
## iteration:  92 log likelihood:  -7025.663
## iteration:  93 log likelihood:  -7024.544
## iteration:  94 log likelihood:  -7023.45
## iteration:  95 log likelihood:  -7022.382
## iteration:  96 log likelihood:  -7021.338
## iteration:  97 log likelihood:  -7020.318
## iteration:  98 log likelihood:  -7019.322
## iteration:  99 log likelihood:  -7018.351
## iteration:  100 log likelihood:  -7017.402
```

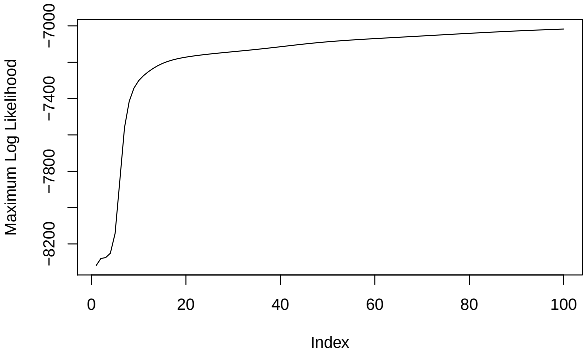


```
## The pi value for four components are
##  0.1666541 0.3451822 0.4011958 0.08696787
```

```
## The mu value for four components are
```

```
##           [,1]      [,2]      [,3]      [,4]      [,5]      [,6]      [,7]
## [1,] 0.4892226 0.6769854 0.4247993 0.2837305 0.6192035 0.2532400 0.9434193
## [2,] 0.5070865 0.5733461 0.3819943 0.6885273 0.3123494 0.7525112 0.2058604
## [3,] 0.4645390 0.4221863 0.5816809 0.3601909 0.6479657 0.2959319 0.6612254
## [4,] 0.2881603 0.7921244 0.9311470 0.4546589 0.4487704 0.4520794 0.7754019
##           [,8]      [,9]     [,10]
## [1,] 0.6523896 0.6816772 0.4982281
## [2,] 0.8436435 0.1692561 0.9360949
## [3,] 0.2296106 0.7561997 0.1220751
## [4,] 0.9307696 0.7656546 0.8723785
```

```
## The number of iteration is 100 and and the maximum likelihood value is -7017.402
```



The log of likelihood is increasing with increase in K value. So, for log of likelihood value will maximize when the negative value is converging to zero. But since the condition of difference between two consecutive iteration is 0.1, it is not possible to find the iteration number which leads to zero.

There was a unusual accuracy in randomforest method but it has been corrected. There is mistake in the formula while calculating the number of parameters but it is corrected.
# Appendix

```r
knitr::opts_chunk$set(echo = TRUE)
library(mboost)
library(randomForest)
library(caret)
sf1 <- read.csv(file.choose())
sf1$Spam <- as.factor(sf1$Spam)
n=dim(sf1)[1]
suppressWarnings(RNGversion("3.5.9"))
id=sample(1:n, floor(n*0.66))
train=sf1[id,]
test=sf1[-id,]
misclassification_train <- vector()
misclassification_test <- vector()
J <- 10
for (i in 1:10){

  model_boost <- blackboost(Spam ~., data = train,family = AdaExp(), control = boost_control(mstop = J)
  Y_pred <- predict.mboost(model_boost, newdata = train, type = "class")
  Y_pred_test <- predict.mboost(model_boost, newdata = test, type = "class" )
  misclassification_train[i] <- mean(Y_pred != train$Spam)
  misclassification_test[i] <- mean(Y_pred_test != test$Spam)
  J <- J+10
}

Tree_number <- seq(10,100,10)
par(mfrow=c(1,1))
cat("Missclassification Rate of Train data\n",min(misclassification_train),"\n")
cat(" No_of_Trees\n",which(misclassification_train == min(misclassification_train)) * 10,"\n")
cat("Missclassification Rate of Test data\n",min(misclassification_test),"\n")
cat("Number_of_Trees\n",which(misclassification_test == min(misclassification_test)) * 10,"\n")
plot(x=Tree_number,y = misclassification_train,type="l")
title(main = "No of Trees vs Misclassification of Train data")
plot(Tree_number,misclassification_test,type="l")
title(main = "No of Trees vs Classifications of Test data")
misclassification_train_rf <- vector()
misclassification_test_rf <- vector()
K <- 10
control <- trainControl(method="repeatedcv", number=15, repeats=3, search="random")
for(i in 1:10)
{
  model_RF <- randomForest(Spam ~., data = sf1, ntree=K,trControl = control)
  Y_pred1 <- predict(model_RF, newdata = train, type="response")
  Y_pred_test1 <- predict(model_RF, newdata = test, type="response")
  misclassification_train_rf[i] <- mean(Y_pred1 != train$Spam)
  misclassification_test_rf[i] <- mean(Y_pred_test1 != test$Spam)
  K <- K+10
}
cat("Missclassification Rate of Train data\n",min(misclassification_train_rf),"\n")
cat(" Number_of_Trees\n",which(misclassification_train_rf == min(misclassification_train_rf)) * 10,"\n"
cat("Miss classification Rate of Test data\n",min(misclassification_test_rf),"\n")
cat("Number_of_Trees\n",which(misclassification_test_rf == min(misclassification_test_rf)) * 10,"\n")
plot(Tree_number,misclassification_train_rf,type="l")
title(main = "No of Trees vs Classifications of Train data")
```

```r
plot(Tree_number,misclassification_test_rf,type="l")
title(main = "No of Trees vs Classifications of Test data")
set.seed(1234567890)
max_it <- 100
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 = 2) # true mixing coefficients
true_mu <- matrix(nrow = 2, ncol = D) # true conditional distributions
true_pi = c(1/2, 1/2)
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)
# true_mu[4,] = c(0.5, 0.7, 0.6, 0.3, 0.6, 0.4, 0.8, 0.8, 0.7, 0.6)
for (n in 1:N) {
  k <- sample(1:2, 1, prob = true_pi)
  for (d in 1:D) {
    x[n, d] <- rbinom(1, 1, true_mu[k, d])
  }
}
K = 2 # 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)
}
for (it in 1:max_it) {
  Sys.sleep(0.5)
  # E-Step
  for (p in 1:N) {
    Component = matrix(1, nrow = 1, ncol = K)
    total_prob = 0
    for (i in 1:K) {
      for (j in 1:D) {
        Component[1,i] = Component[1,i] * (mu[i,j] ^ x[p,j]) * (1 - mu[i, j]) ^ (1 - x[p,j])
      }
      Component[1,i] = Component[1,i] * pi[i]
      total_prob = total_prob + Component[1,i]
    }

    for (i in 1:K) {
      z[p,i] = Component[1,i] / total_prob
    }
  }

  for (i in 1:K) {
    summation = matrix(0, nrow = N, ncol = 1)
    for (j in 1:D)
    {
      summation = summation + x[,j] * log(mu[i,j]) + (1 - x[,j]) * log(1 -mu[i,j])
    }
    llik[it] = llik[it] + sum(z[,i] * (log(pi[i]) + summation))
  }
  cat("iteration: ", it, "log likelihood: ", llik[it], "\n")
  flush.console()
  if (abs(llik[it] - llik[it-1]) < 0.1 && it > 1)
  {
    break
  }

  for(i in 1:K){
    pi[i] = sum(z[,i])/N
  }

  for (i in 1:K) {
    mu[i, ] = colSums(x * z[, i]) / sum(z[,i])
  }
}

plot(mu[1,],type = "o",col = "blue",ylim = c(0, 1),main = "Plot for mu values",xlab = "mu values")
points(mu[2,], type = "o", col = "red")
cat("The pi value for two components are\n",pi,"\n")
cat("The mu value for two components are\n")
mu
cat("The number of iteration is",it,"and and the maximum likelihood value is",llik[it],"\n")
plot(llik[1:it],type="l",xlab = "Index",ylab = "Maximum Log Likelihood")
set.seed(1234567890)
max_it <- 100
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 = 4, 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)
#true_mu[4,] = c(0.5, 0.7, 0.6, 0.3, 0.6, 0.4, 0.8, 0.8, 0.7, 0.6)
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)
}
for (it in 1:max_it) {
  #points(mu[4,], type="o", col="yellow")
  #Sys.sleep(0.5)
  # E-Step
  #Bernoulli function from slide 9
  for (p in 1:N) {
    Component = matrix(1, nrow = 1, ncol = K)
    total_prob = 0
    for (i in 1:K) {
      for (j in 1:D) {
        Component[1,i] = Component[1,i] * (mu[i,j] ^ x[p,j]) * (1 - mu[i, j]) ^ (1 - x[p,j])
      }
      Component[1,i] = Component[1,i] * pi[i]
      total_prob = total_prob + Component[1,i]
    }

    for (i in 1:K) {
      z[p,i] = Component[1,i] / total_prob
    }
  }

  for (i in 1:K) {
    summation = matrix(0, nrow = N, ncol = 1)
    for (j in 1:D)
    {
      summation = summation + x[,j] * log(mu[i,j]) + (1 - x[,j]) * log(1 -mu[i,j])
    }
    llik[it] = llik[it] + sum(z[,i] * (log(pi[i]) + summation))
  }
  cat("iteration: ", it, "log likelihood: ", llik[it], "\n")
  flush.console()
  if (abs(llik[it] - llik[it-1]) < 0.1 && it > 1)
  {
    break
  }

  for(i in 1:K){
    pi[i] = sum(z[,i])/N
  }

  for (i in 1:K) {
    mu[i, ] = colSums(x * z[, i]) / sum(z[,i])
  }
}
plot(mu[1,],type = "o",col = "blue",ylim = c(0, 1),main = "Plot for mu values")
points(mu[2,], type = "o", col = "red")
points(mu[3,], type = "o", col = "green")
```

```r
cat("The pi value for three components are\n",pi,"\n")
cat("The mu value for three components are \n")
mu
cat("The number of iteration is",it,"and and the maximum likelihood value is",llik[it],"\n")
plot(llik[1:it],type="l",xlab = "Index",ylab = "Maximum Log Likelihood")
set.seed(1234567890)
max_it <- 100
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 = 4) # true mixing coefficients
true_mu <- matrix(nrow = 4, ncol = D) # true conditional distributions
true_pi = c(1/4, 1/4, 1/4, 1/4)
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)
true_mu[4,] = c(0.5, 0.7, 0.6, 0.3, 0.6, 0.4, 0.8, 0.8, 0.7, 0.6)
for (n in 1:N) {
  k <- sample(1:4, 1, prob = true_pi)
  for (d in 1:D) {
    x[n, d] <- rbinom(1, 1, true_mu[k, d])
  }
}
K = 4 # 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)
}
for (it in 1:max_it) {
  Sys.sleep(0.5)
  # E-Step
  for (p in 1:N) {
    Component = matrix(1, nrow = 1, ncol = K)
    total_prob = 0
    for (i in 1:K) {
      for (j in 1:D) {
        Component[1,i] = Component[1,i] * (mu[i,j] ^ x[p,j]) * (1 - mu[i, j]) ^ (1 - x[p,j])
      }
      Component[1,i] = Component[1,i] * pi[i]
      total_prob = total_prob + Component[1,i]
    }

    for (i in 1:K) {
      z[p,i] = Component[1,i] / total_prob
    }
  }
  for (i in 1:K) {
    summation = matrix(0, nrow = N, ncol = 1)
    for (j in 1:D)
    {
      summation = summation + x[,j] * log(mu[i,j]) + (1 - x[,j]) * log(1 -mu[i,j])
    }
    llik[it] = llik[it] + sum(z[,i] * (log(pi[i]) + summation))
  }
  cat("iteration: ", it, "log likelihood: ", llik[it], "\n")
  flush.console()
  if (abs(llik[it] - llik[it-1]) < min_change && it > 1)
  {
    break
  }

  for(i in 1:K){
    pi[i] = sum(z[,i])/N
  }

  for (i in 1:K) {
    mu[i, ] = colSums(x * z[, i]) / sum(z[,i])
  }
}
plot(mu[1,],type = "o",col = "blue",ylim = c(0, 1),main = "Plot for mu values",xlab = "mu-values")
points(mu[2,], type = "o", col = "red")
points(mu[3,], type = "o", col = "green")
points(mu[4,],type = "o",col = "yellow")
cat("The pi value for four components are\n",pi,"\n")
cat("The mu value for four components are\n")
mu
cat("The number of iteration is",it,"and and the maximum likelihood value is",llik[it],"\n")
plot(llik[1:it],type="l",ylab = "Maximum Log Likelihood")
```