Untitled

Aman Kumar Nayak

12/4/2019

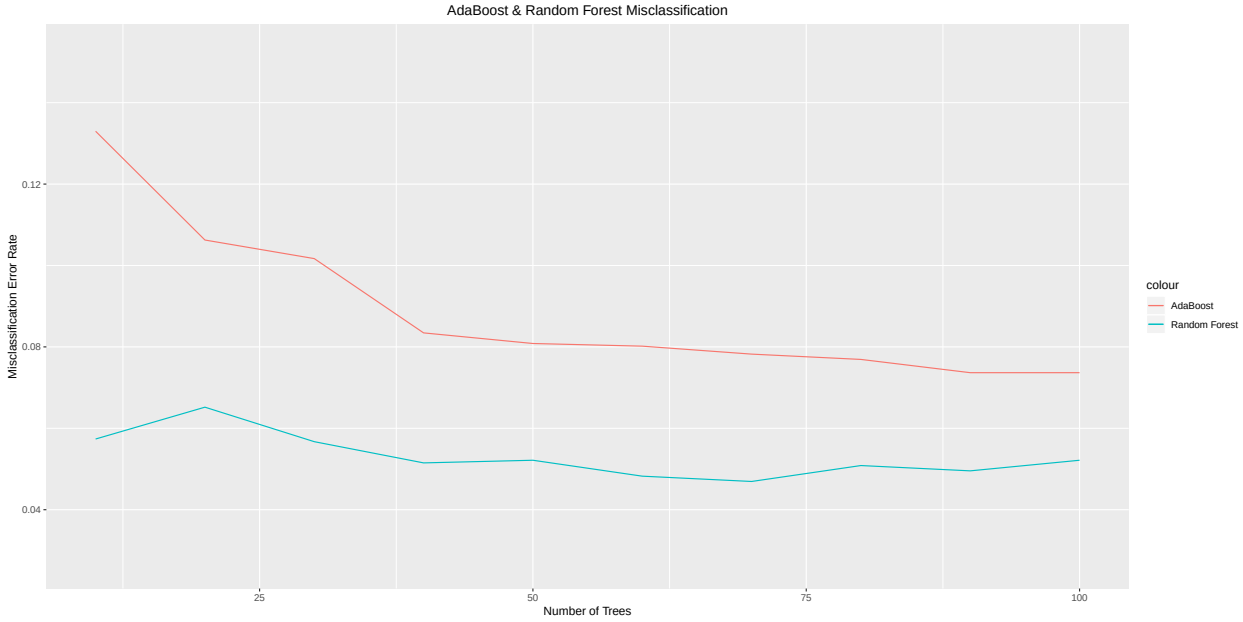
1. Ensemble Methods

```
## [1] "Miscalculation Vector AdaBoost Test Data"
```

```
##  [1] 0.13298566 0.10625815 0.10169492 0.08344198 0.08083442 0.08018253
##  [7] 0.07822686 0.07692308 0.07366362 0.07366362
```

```
## [1] "Miscalculation Vector Random Forest Test Data"
```

```
##  [1] 0.05736636 0.06518905 0.05671447 0.05149935 0.05215124 0.04823990
##  [7] 0.04693611 0.05084746 0.04954368 0.05215124
```



2. Mixture Model

For K = 2

```
## [1] "Number of iteration for K = 2 is :"
```

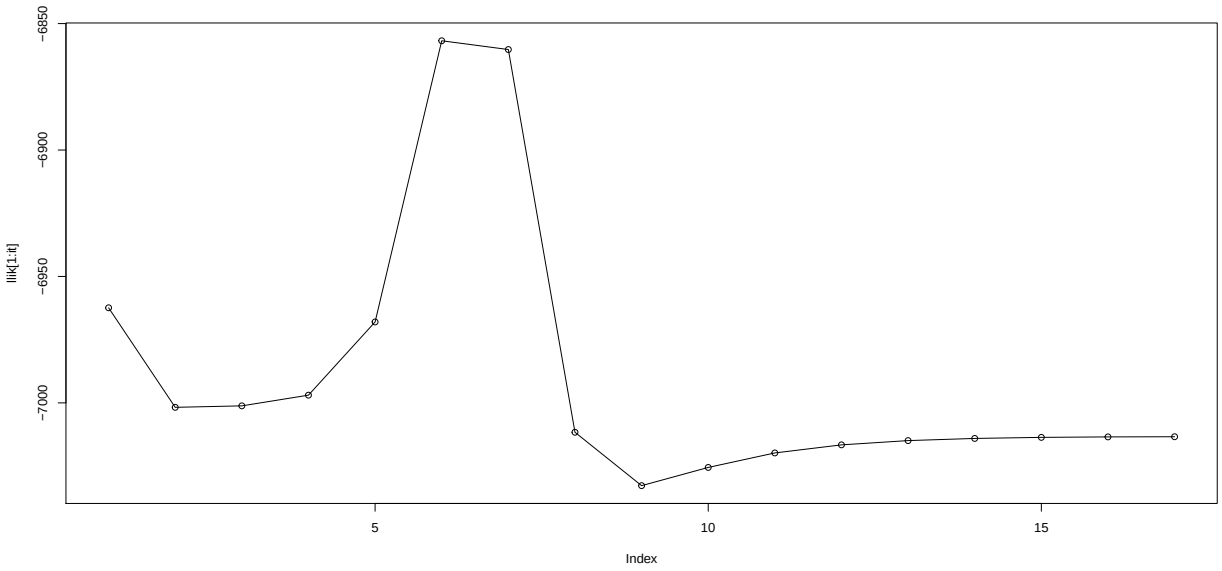
```
## [1] 17
```

```
## [1] "The value for pi is :"
```

```
## [1] 0.4983586 0.5016414
```

```
## [1] "The value of mu is:"
```

```
##           [,1]      [,2]      [,3]      [,4]      [,5]      [,6]      [,7]
## [1,] 0.4777257 0.4113178 0.5887821 0.3477590 0.6580535 0.2699434 0.7077294
## [2,] 0.5061809 0.5601935 0.4177793 0.6731727 0.3350070 0.7245642 0.2617347
##           [,8]      [,9]      [,10]
## [1,] 0.2135532 0.7939559 0.08751855
## [2,] 0.8005199 0.1680987 0.90380175
```



For K = 3

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

```
##           [,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
```

```
## iteration:  1 log likelihood:  -7364.464
## iteration:  2 log likelihood:  -7407.245
## iteration:  3 log likelihood:  -7406.241
## iteration:  4 log likelihood:  -7399.085
## iteration:  5 log likelihood:  -7356.037
## iteration:  6 log likelihood:  -7256.86
## iteration:  7 log likelihood:  -7343.566
## iteration:  8 log likelihood:  -7433.141
## iteration:  9 log likelihood:  -7429.89
## iteration:  10 log likelihood:  -7415.863
## iteration:  11 log likelihood:  -7406.448
## iteration:  12 log likelihood:  -7401.213
## iteration:  13 log likelihood:  -7398.638
```

```
## iteration:  14 log likelihood:  -7397.671
## iteration:  15 log likelihood:  -7397.663
```

```
## [1] "Number of iteration for K = 3 is :"
```

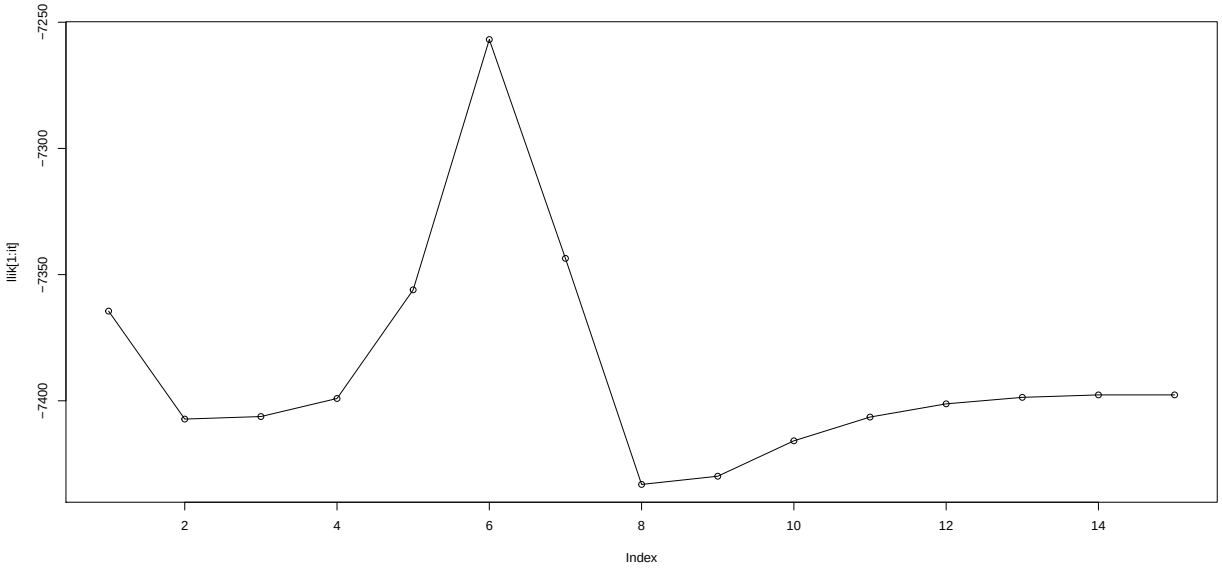
```
## [1] 15
```

```
## [1] "The value for pi is :"
```

```
## [1] 0.3562429 0.2318521 0.4119050
```

```
## [1] "The value of mu is:"
```

```
##           [,1]      [,2]      [,3]      [,4]      [,5]      [,6]      [,7]
## [1,] 0.4737989 0.3882869 0.6262914 0.3223413 0.6855682 0.2122545 0.7733311
## [2,] 0.4917803 0.4800716 0.4580175 0.4579248 0.5413198 0.4870554 0.4659596
## [3,] 0.5078651 0.5738458 0.4216890 0.7040394 0.3065393 0.7512923 0.2439217
##           [,8]      [,9]      [,10]
## [1,] 0.1533106 0.8703909 0.03934392
## [2,] 0.4707099 0.5057109 0.36305191
## [3,] 0.8357489 0.1278917 0.96820791
```

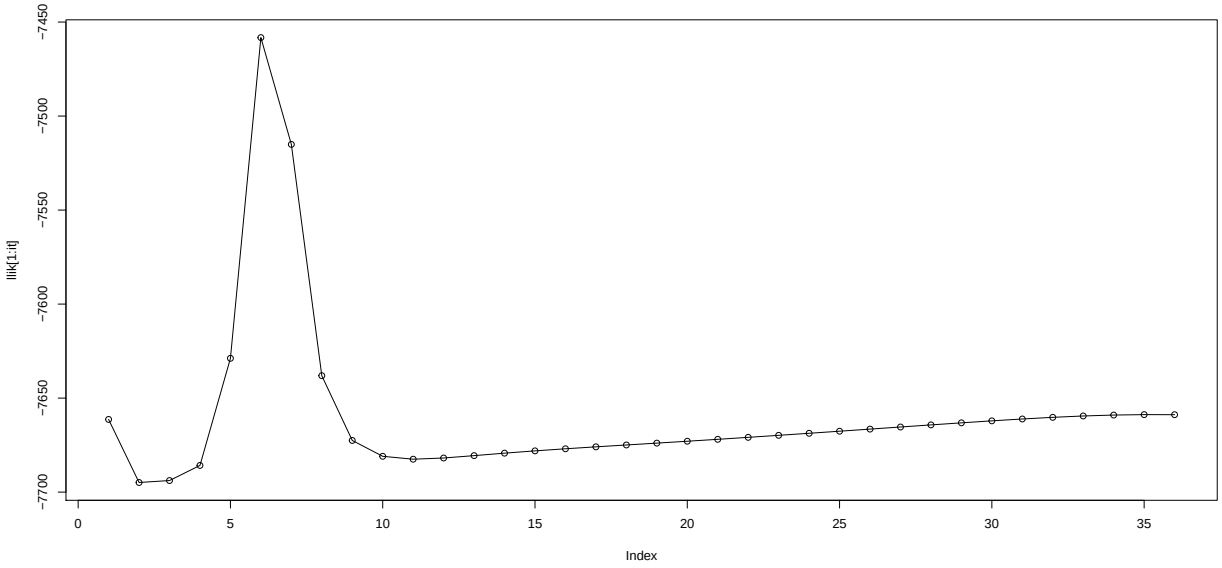


For K = 4

```
## [1] 0.2491838 0.2499680 0.2500275 0.2508207
```

```
##           [,1]      [,2]      [,3]      [,4]      [,5]      [,6]      [,7]
## [1,] 0.4935375 0.5042511 0.5040286 0.4987810 0.5012754 0.4971036 0.4982144
## [2,] 0.5088453 0.5068730 0.5016720 0.4929275 0.5077146 0.5095075 0.4924574
## [3,] 0.5077926 0.4939841 0.5059821 0.5063490 0.5041462 0.4929400 0.4992362
## [4,] 0.4909320 0.4982342 0.4961948 0.4967226 0.4984220 0.4960055 0.4997165
```

```
##           [,8]      [,9]     [,10]
## [1,] 0.4987654 0.4929075 0.4993719
## [2,] 0.4992470 0.5008651 0.4975302
## [3,] 0.4943482 0.4903974 0.5089045
## [4,] 0.5090205 0.5057927 0.4947660

## [1] "Number of iteration for K = 4 is :"

## [1] 36

## [1] "The value for pi is :"

## [1] 0.1532838 0.1437647 0.3438494 0.3591021

## [1] "The value of mu is:"

##           [,1]      [,2]      [,3]      [,4]      [,5]      [,6]      [,7]
## [1,] 0.4614417 0.5826670 0.5474713 0.4619753 0.4543008 0.3879386 0.4583298
## [2,] 0.5166847 0.3852127 0.3499025 0.5326553 0.5693745 0.6508786 0.4025037
## [3,] 0.5109943 0.5905663 0.4238014 0.7245346 0.2821673 0.7770704 0.2267970
## [4,] 0.4769740 0.3849621 0.6211438 0.3187919 0.6891746 0.2165588 0.7738625
##           [,8]       [,9]      [,10]
## [1,] 0.5113622 0.53197738 0.50951500
## [2,] 0.5513537 0.36261877 0.46393477
## [3,] 0.8629739 0.09592275 0.99989474
## [4,] 0.1493118 0.87257009 0.02336084
```



Analysis With K = 2, I could see that maximum value for likelihood is achieved and with increased value of K from 2 to 4 number of iterations also increased.

Higher value of likelihood will give better classification of data.

Appendix

```r
knitr::opts_chunk$set(echo = TRUE,fig.height = 8,fig.width =16 )
#library
#install.packages("mboost")
#install.packages("randomForest")
library(mboost)
library(randomForest)
#for pipe operator
library(dplyr)
library(magrittr)
library(ggplot2)


#import Data
spamData = read.csv2(file = "G:/MS Machine Learning/Term/Term2/ML/ML Assignment/1b/spambase.csv")
spamData$Spam = as.factor(spamData$Spam)
#Part 1
suppressWarnings(RNGkind(sample.kind = "Rounding"))
set.seed(12345)
n = dim(spamData)[1]
id = sample(1:n , floor(n*(2/3)))

train = spamData[id ,]
test  = spamData[-id,]

#blackboost()
#i = 10
misCalAddaBoost = numeric(10)
for(i in 1:length(misCalAddaBoost))
{
  adaboost = blackboost(Spam~. , data = train , family = AdaExp() , control = boost_control(mstop = i*1
  #probabilites = adaboost %>% predict(object = adaboost , newdata = test , type = "response")
  probabilites = predict(object = adaboost , newdata = test)
  #print("Confusion Matrix for Test data with Y= 1 if p(Y=|X) > 0 else 0")
  condition = ifelse(probabilites > 0, 1, 0)
  con = table(test$Spam , condition)
  misCalAddaBoost[i] = 1 - (sum(diag(con)) / sum(con))
}

print("Miscalculation Vector AdaBoost Test Data")

misCalAddaBoost

#randomForest()
#i = 10
misCalRandomForset = numeric(10)
for(i in 1:length(misCalRandomForset))
{
  randomforest = randomForest(Spam~. , data = train , ntree = i*10)
  #probabilites = adaboost %>% predict(object = adaboost , newdata = test , type = "response")
  probabilitesRF = predict(randomforest , newdata = test)
  con = table(test$Spam , probabilitesRF)
  misCalRandomForset[i] = 1 - (sum(diag(con)) / sum(con))
}
```

```r
print("Miscalculation Vector Random Forest Test Data")

misCalRandomForset

plotMissCalRate = ggplot() +
  geom_line(aes(x = seq(from = 10, to = 100, by = 10), y = misCalRandomForset, color = "Random Forest")) +
  geom_line(aes(x = seq(from = 10, to = 100, by = 10), y = misCalAddaBoost, color = "AdaBoost")) +
  ylab("Misclassification Error Rate") + xlab("Number of Trees") +
  ggtitle("AdaBoost & Random Forest Misclassification") +
  #to make graph look interpreteable for min and max values lim is introduced
  ylim(min(misCalRandomForset)-0.02, max(misCalAddaBoost)+ 0.02)+
  theme(plot.title = element_text(hjust = 0.5))

plotMissCalRate


suppressWarnings(RNGkind(sample.kind = "Rounding"))
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 = 2) # true mixing coefficients
true_pi <- vector(length = 3) # true mixing coefficients
#true_pi <- vector(length = 4) # true mixing coefficients

#true_mu <- matrix(nrow=2, ncol=D) # true conditional distributions
true_mu <- matrix(nrow=3, ncol=D) # true conditional distributions
# true_mu <- matrix(nrow=4, ncol=D) # true conditional distributions


#true_pi=c(1/2 , 1/2)
true_pi = c(1/3, 1/3, 1/3)
# 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(0.5,0.5,0.5,0.5,0.5,0.5,0.5,0.5,0.5,0.5)


# 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")
# points(true_mu[4,], type="o", col="yellow")

# Producing the training data
for(n in 1:N) {
  #k <- sample(1:2,1,prob=true_pi)
  k <- sample(1:3,1,prob=true_pi)
```

```
  #k <- sample(1:4,1,prob=true_pi)
  for(d in 1:D) {
    x[n,d] <- rbinom(1,1,true_mu[k,d])
  }
}
K=2 # number of guessed components
#K = gussedComponent
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
# mu
change = 0
for(it in 1:max_it)
{
  Sys.sleep(0.5)

  # E-step: Computation of the fractional component assignments
  for(a in 1:N)
  {
    Q = c()
    for(b in 1:K){
      bernouliProb = prod((mu[b,])^x[a,] , (1 - mu[b,])^(1-x[a,]) )
      Q = c(Q , pi[b]*bernouliProb)
    }#for(b in 1:K){
    z[a,] = Q / sum(Q)
  }#for(a in 1:N){


  #Log likelihood computation.
  logLikelihood = 0
  for(a in 1:N)
  {
    for(b in 1:K){
      logLikelihood =  logLikelihood + z[a,b]*(log(pi[b]) + sum(x[n,]*log(mu[b,]) + (1-x[a,])*log(1-mu[
    }#for(b in 1:K){
  }#for(a in 1:N){


  llik[it] = logLikelihood
  # cat("iteration: ", it, "log likelihood: ", llik[it], "\n")
  # flush.console()

  # Stop if the log likelihood has not changed significantly
  if(abs(change - llik[it]) < min_change){
    break()
```

```
  }else{
    change = llik[it]
  }

  #M-step: ML parameter estimation from the data and fractional component assignments
  pi = colMeans(z)
  for(b in 1:k)
  {
    for(c in 1:D){
      mu[b,c] = sum(z[,b]*x[,c])/sum(z[,b])
    }#for(b in 1:K){
  }#for(a in 1:N){

}#for(it in 1:max_it)

print("Number of iteration for K = 2 is :")
length(llik[1:it])

print("The value for pi is :")
pi

print("The value of mu is:")
mu

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


suppressWarnings(RNGkind(sample.kind = "Rounding"))
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 = 2) # true mixing coefficients
true_pi <- vector(length = 3) # true mixing coefficients
#true_pi <- vector(length = 4) # true mixing coefficients

#true_mu <- matrix(nrow=2, ncol=D) # true conditional distributions
true_mu <- matrix(nrow=3, ncol=D) # true conditional distributions
# true_mu <- matrix(nrow=4, ncol=D) # true conditional distributions


#true_pi=c(1/2 , 1/2)
true_pi = c(1/3, 1/3, 1/3)
# 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(0.5,0.5,0.5,0.5,0.5,0.5,0.5,0.5,0.5,0.5)
```

```
# 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")
# points(true_mu[4,], type="o", col="yellow")

# Producing the training data
for(n in 1:N) {
  #k <- sample(1:2,1,prob=true_pi)
  k <- sample(1:3,1,prob=true_pi)
  #k <- sample(1:4,1,prob=true_pi)
  for(d in 1:D) {
    x[n,d] <- rbinom(1,1,true_mu[k,d])
  }
}
K=3 # number of guessed components
#K = gussedComponent
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
mu
change = 0
for(it in 1:max_it)
{
  Sys.sleep(0.5)

  # E-step: Computation of the fractional component assignments
  for(a in 1:N)
  {
    Q = c()
    for(b in 1:K){
      bernouliProb = prod((mu[b,])^x[a,] , (1 - mu[b,])^(1-x[a,]) )
      Q = c(Q , pi[b]*bernouliProb)
    }#for(b in 1:K){
    z[a,] = Q / sum(Q)
  }#for(a in 1:N){


  #Log likelihood computation.
  logLikelihood = 0
  for(a in 1:N)
  {
    for(b in 1:K){
      logLikelihood =  logLikelihood + z[a,b]*(log(pi[b]) + sum(x[n,]*log(mu[b,]) + (1-x[a,])*log(1-mu[
    }#for(b in 1:K){
```

```
  }#for(a in 1:N){


  llik[it] = logLikelihood
  cat("iteration: ", it, "log likelihood: ", llik[it], "\n")
  flush.console()

  # Stop if the log likelihood has not changed significantly
  if(abs(change - llik[it]) < min_change){
    break()
  }else{
    change = llik[it]
  }

  #M-step: ML parameter estimation from the data and fractional component assignments
  pi = colMeans(z)
  for(b in 1:k)
  {
    for(c in 1:D){
      mu[b,c] = sum(z[,b]*x[,c])/sum(z[,b])
    }#for(b in 1:K){
  }#for(a in 1:N){

}#for(it in 1:max_it)

print("Number of iteration for K = 3 is :")
length(llik[1:it])

print("The value for pi is :")
pi

print("The value of mu is:")
mu

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


suppressWarnings(RNGkind(sample.kind = "Rounding"))
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 = 2) # true mixing coefficients
true_pi <- vector(length = 3) # true mixing coefficients
#true_pi <- vector(length = 4) # true mixing coefficients

#true_mu <- matrix(nrow=2, ncol=D) # true conditional distributions
true_mu <- matrix(nrow=3, ncol=D) # true conditional distributions
# true_mu <- matrix(nrow=4, ncol=D) # true conditional distributions
```

```
#true_pi=c(1/2 , 1/2)
true_pi = c(1/3, 1/3, 1/3)
# 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(0.5,0.5,0.5,0.5,0.5,0.5,0.5,0.5,0.5,0.5)


# 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")
# points(true_mu[4,], type="o", col="yellow")

# Producing the training data
for(n in 1:N) {
  #k <- sample(1:2,1,prob=true_pi)
  k <- sample(1:3,1,prob=true_pi)
  #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
#K = gussedComponent
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
mu
change = 0
for(it in 1:max_it)
{
  Sys.sleep(0.5)

  # E-step: Computation of the fractional component assignments
  for(a in 1:N)
  {
    Q = c()
    for(b in 1:K){
      bernouliProb = prod((mu[b,])^x[a,] , (1 - mu[b,])^(1-x[a,]) )
      Q = c(Q , pi[b]*bernouliProb)
    }#for(b in 1:K){
    z[a,] = Q / sum(Q)
```

```
  }#for(a in 1:N){


  #Log likelihood computation.
  logLikelihood = 0
  for(a in 1:N)
  {
    for(b in 1:K){
      logLikelihood =  logLikelihood + z[a,b]*(log(pi[b]) + sum(x[n,]*log(mu[b,]) + (1-x[a,])*log(1-mu[
    }#for(b in 1:K){
  }#for(a in 1:N){


  llik[it] = logLikelihood
  # cat("iteration: ", it, "log likelihood: ", llik[it], "\n")
  # flush.console()
  #
  # Stop if the log likelihood has not changed significantly
  if(abs(change - llik[it]) < min_change){
    break()
  }else{
    change = llik[it]
  }

  #M-step: ML parameter estimation from the data and fractional component assignments
  pi = colMeans(z)
  for(b in 1:k)
  {
    for(c in 1:D){
      mu[b,c] = sum(z[,b]*x[,c])/sum(z[,b])
    }#for(b in 1:K){
  }#for(a in 1:N){

}#for(it in 1:max_it)

print("Number of iteration for K = 4 is :")
length(llik[1:it])

print("The value for pi is :")
pi

print("The value of mu is:")
mu

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