Block2 lab1

LiuId- priku577

Priya Kurian pullolickal

November 28, 2017

1.Ensemble Method

Ensemble method uses multiple learning algorithms instead of using a single learning algorithm to obtain better predictions.The ensemble methods in this problem are Adaboost classification and random.We evaluate the perfomance of both of this on a csv containing data classified as spam or regular mail.

We first divide the 2/3 of data as training and 1/3 as test

```
data<-read.csv("/home/george/Documents/732A95/lab1_block2/spambase.csv", sep=";", dec=",")
n=dim(data)[1]
set.seed(12345)
id=sample(1:n, floor(n*0.75))
train=data[id,]
test=data[-id,]
```

Adaboost classification

```
library(mboost)
```

```
## Loading required package: parallel
```

```
## Loading required package: stabs
```

```
## This is mboost 2.8-1. See 'package?mboost' and 'news(package  = "mboost")'
## for a complete list of changes.
```

```
set.seed(1234)
#no of trees 10,20...,100
trees=seq(10,100,by=10)
adaBoost<-c()
misclassification_ada<-c()
#loop over the lenght of tree
for(i in 1:length(trees))
{
adaBoost[[i]] <- blackboost(as.factor(Spam) ~ .,
                        data = train,
                        family = AdaExp(),
                        control = boost_control(trees[i]))

ada_test <- predict(adaBoost[[i]], newdata=test, type='class')
#confusion matrix
confusion_mat_ada<-table(ada_test,as.factor(test$Spam))
#finding the misclassification
misclassification_ada[i]<- 1-(sum(diag(confusion_mat_ada)))/sum(confusion_mat_ada)
}
misclassification_ada
```

```
##  [1] 0.13205908 0.10686360 0.09122502 0.08253692 0.08079930 0.08079930
##  [7] 0.07906169 0.07819288 0.07558645 0.07471764
```

We use the *blackboost* function of *mboost* package to do the Adaboost classification.Since we need to plot the error rate for different number of trees,we loop the adaBoost over the different trees using a for loop.All those values are stored in *adaboost*.we then do the prediction on test data.Confusion matrix and misclassification is then calculated.*misclassification_ada* gives the error.*Spam* is considered against all other variables in *adaBoost* in the *training* data.*family=AdaExp()* specifies the loss function to be optimised by the boosting algorithm. we can give each of the *tree* using *control = boost_control(trees[i])*.

From the result it is clear that the MSE decreases with increase in the number of trees.

Boosting is the ensemble method that creates a strong classifier from a number of weak classifiers.At first a model is created from the training data and then the next model is created which will try to correct hte errors of the first model.The models are added as long as the training set is predicted perfectly or models reach the maximum limit.Each time the classifier gets better and better.

Random Forest

```r
library(randomForest)
```

```
## randomForest 4.6-12
```

```
## Type rfNews() to see new features/changes/bug fixes.
```

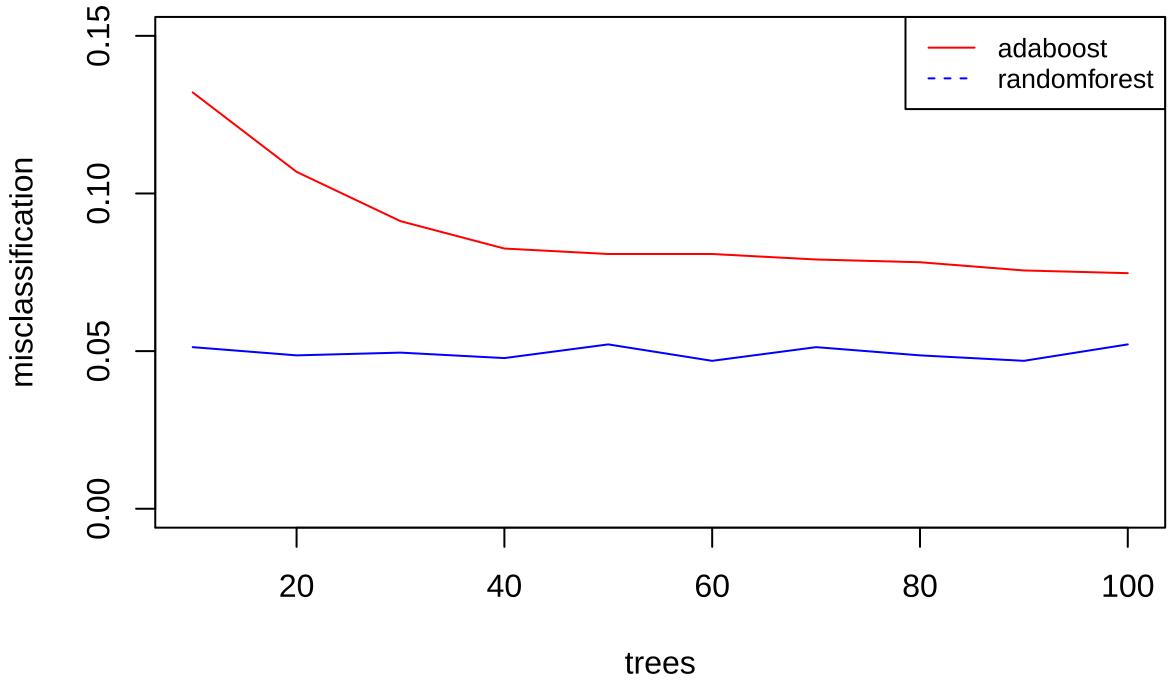
```r
##random forest train values for different number of trees
rf_train<-c()
misclassification_rf<-c()

for(i in 1:length(trees))
{
rf_train[[i]] <- randomForest(as.factor(Spam) ~ .,
                   data = train,ntree=trees[i])
rf_test <- predict(rf_train[[i]], newdata=test, type='class')
#confusion matrix
confusion_mat_rf<-table(rf_test,as.factor(test$Spam))
#misclassification rate
misclassification_rf[i]<- 1-(sum(diag(confusion_mat_rf)))/sum(confusion_mat_rf)
}
misclassification_rf
```

```
##  [1] 0.05125977 0.04865334 0.04952215 0.04778454 0.05212858 0.04691573
##  [7] 0.05125977 0.04865334 0.04691573 0.05212858
```

The *randomForest* function is used to perform the random forest classification.*misclassification_rf* gives the error.The data used is same for the adaboost and random forest.

From the result it is clear that the MSE error is less for Random forest when compared to the adaBoost.

```r
plot(y=misclassification_ada,x=trees,type="l",col="red",ylim=c(0,0.15),ylab="misclassification")
lines(y=misclassification_rf,x=trees,type="l",col="blue",ylim=c(0,0.15))
legend("topright", legend=c("adaboost", "randomforest"),
       col=c("red", "blue"), lty=1:2, cex=0.8)
```



From the plot it is clear that the random forest has a better perfomance.We earlier found that random forest has lower MSE value when compared to adaBoost.

2.MIXTURE MODELS

Mixture model is the mixture distribution of the probability distributions of observations of sampled data.EM algorithm for mixtures of multivariate Benouilli distributions is implemeted below. Mixture of multivariate Bernouilli distributions:

$$Bern(x|\mu_k) = \prod_{i}^{D} \mu_{ki}^{xi}(1-\mu_{ki})^{(1-x_i)}$$

E-step :

Compute p(Z|X) for all k and n

$$p(z_{nk}|x_n, \mu, \pi) = \frac{\pi_k P(x_n|\mu_k)}{\sum \pi_k P(x_n|\mu_k)}$$

So the Z is the value of p(Z|X) which is coded as below.

```
# E-step: Computation of the fractional component assignments
 # Your code here

 px <- integer(N)

 for(n in 1:1000){
   probSum <- 0
   for(k in 1:K){

     probSum <- probSum + pi[k]*prod((mu[k,]^x[n,])*(1-mu[k,])^(1-x[n,]))
     }
```

```
      px[n] <- probSum
    }

  for(n in 1:1000){
    for(k in 1:K){
       z[n,k]<- (pi[k]*prod((mu[k,]^x[n,])*(1-mu[k,])^(1-x[n,])))/px[n]
    }
  }
```

Log likelihood computation

The log likelihood function is

$$\log p(x_n, z_n|\mu, \pi) = \sum_n \log \prod_k [\pi_k \prod_i \mu_{ki}^{X_{ni}} (1-\mu_{ki})^{(1-X_{ni})}]^{z_{nk}}$$

The code below is the calculation of log likelihood.

```
logL <- matrix(0, nrow = 1000, ncol = 3)
  for(n in 1:1000){
    for(k in 1:K){
      logL[n,k] <- (pi[k]*(prod((mu[k,]^x[n,])*(1-mu[k,])^(1-x[n,]))))

    }
  llik[it] <- sum(log(rowSums(logL)))
  }
```

M-step:

Finally in M-step ,ML parameter estimation from the data and fractional component assignments is

$$\pi_k^{ML} = \frac{\sum_n p(z_{nk}|x_n, \mu, \pi)}{N}$$

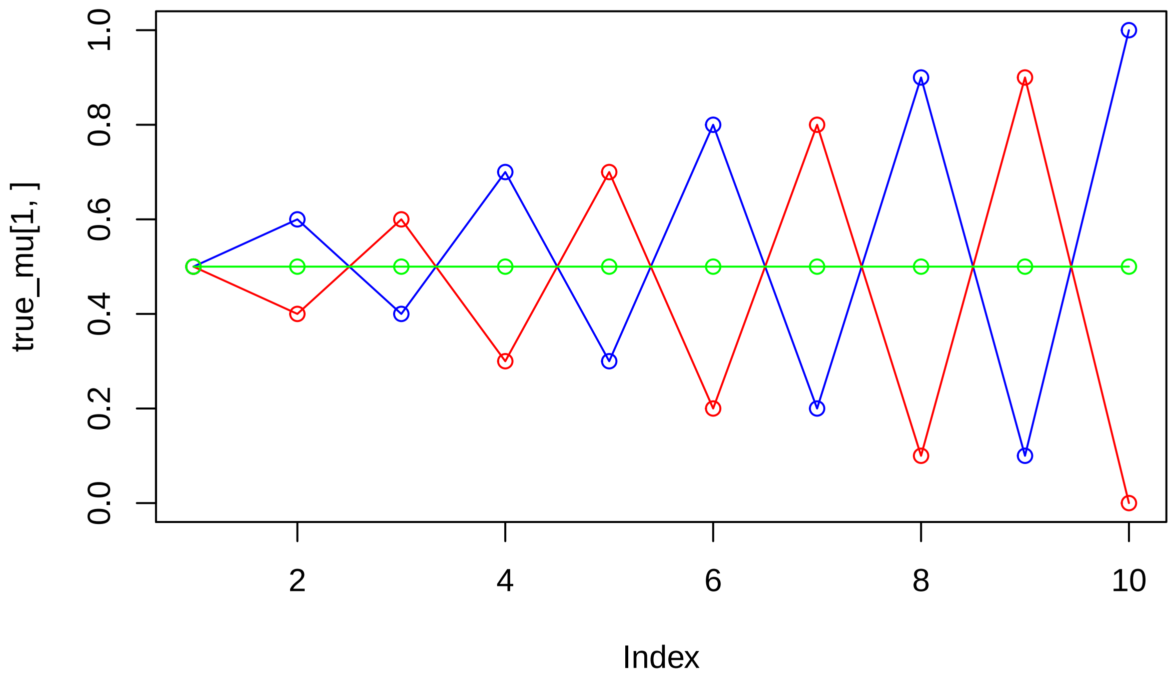
And,

$$\mu_{ki}^{ML} = \frac{\sum_n x_{ni} p(z_{nk}|x_n, \mu, \pi)}{\sum_n p(z_{nk}|x_n, \mu, \pi)}$$

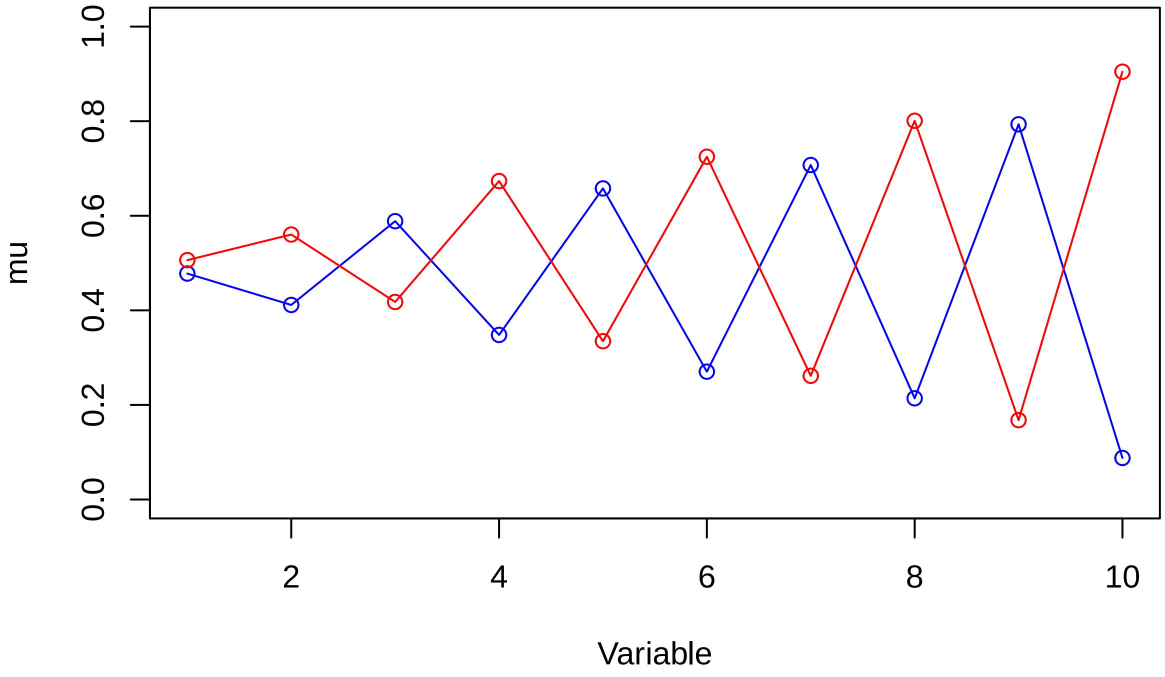
The code below is the interpretation in R

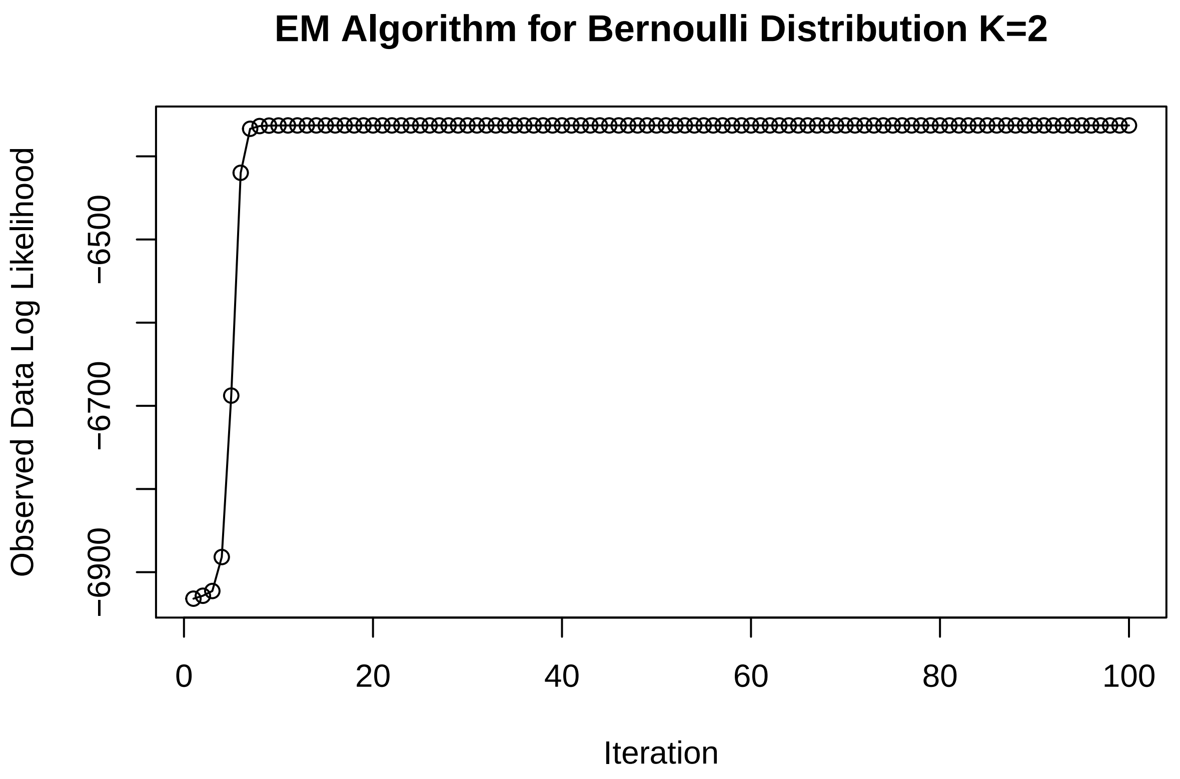
```
pi<-colSums(z)/N
  mu<-t(z)%*%x/colSums(z)
```

The threee bernoulli distribution is generated as data to be combined in a mixure model.Below plot shows the true distribution of three distribution.

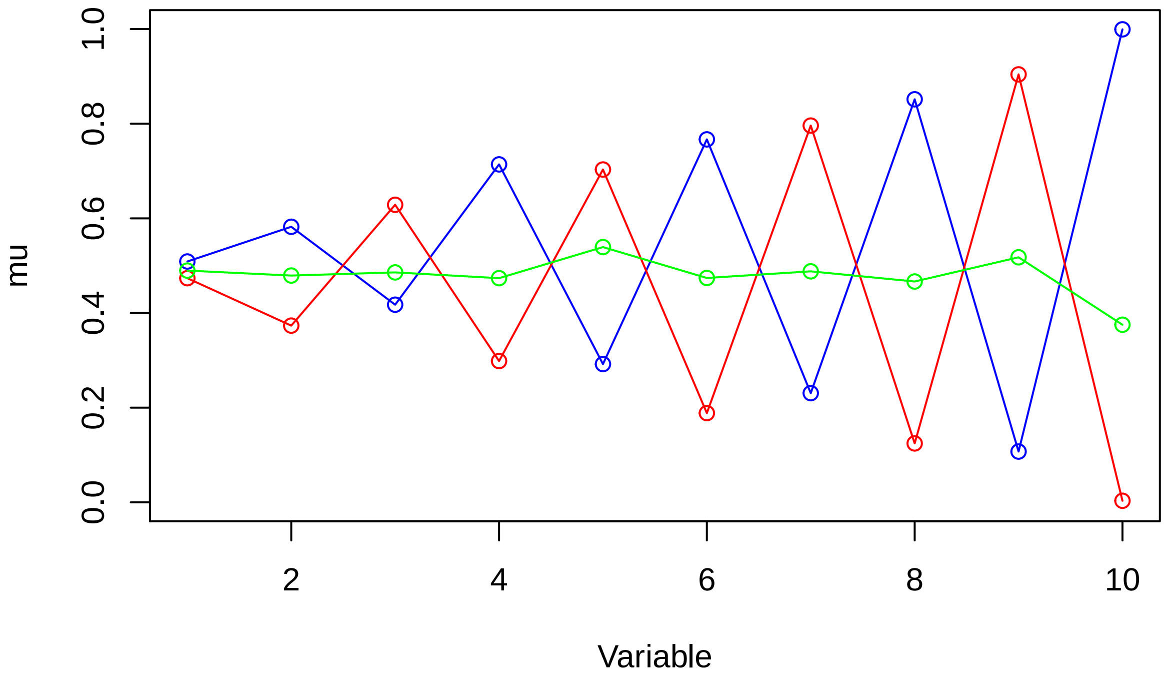


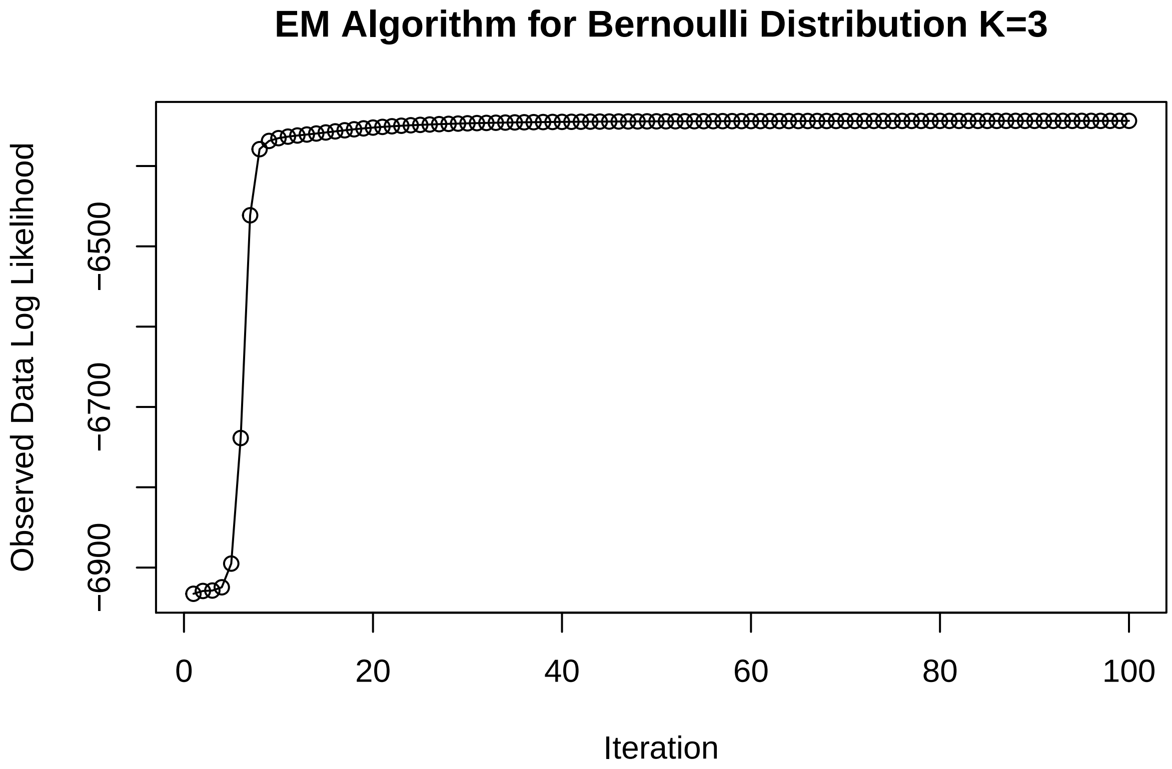
Below plot is generated using K = 2,so the number of components are assumed to be 2. As the mixing coeffients are almost equal (.49 and .51). The parameters estimated by the EM algorithm are almost the same.The third distribution merged into one of them. The plotting of observed data log-likelihood as a function of the iteration number is produced by the EM algorithm is given next for K=2. [Updated]



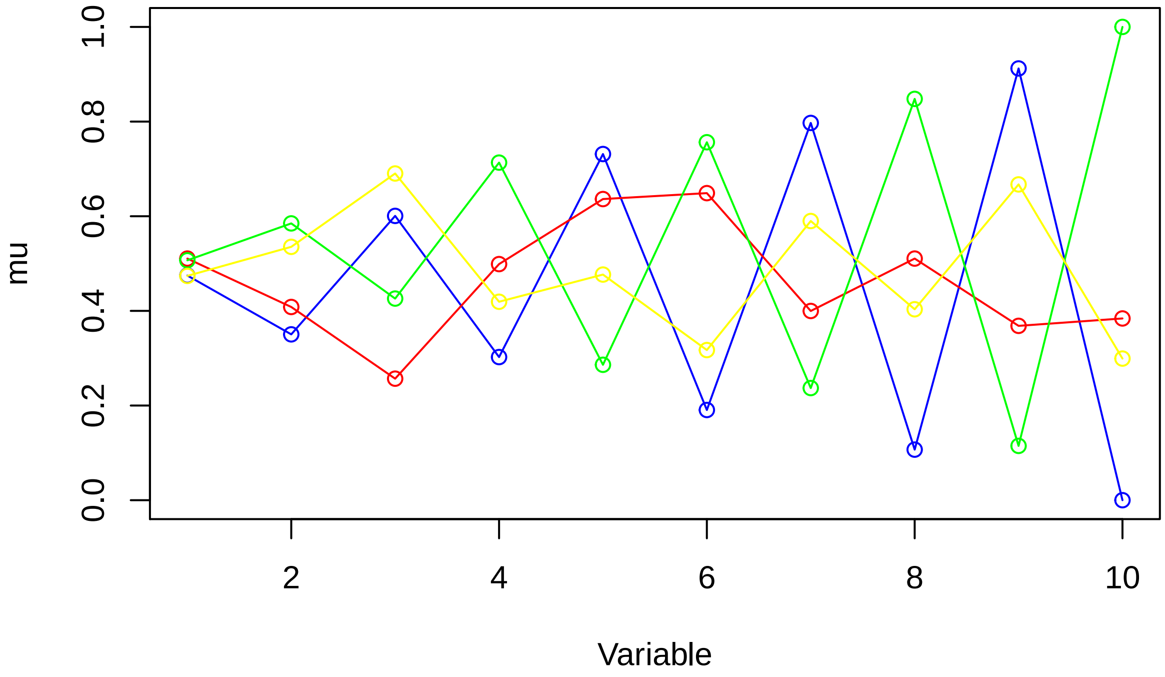


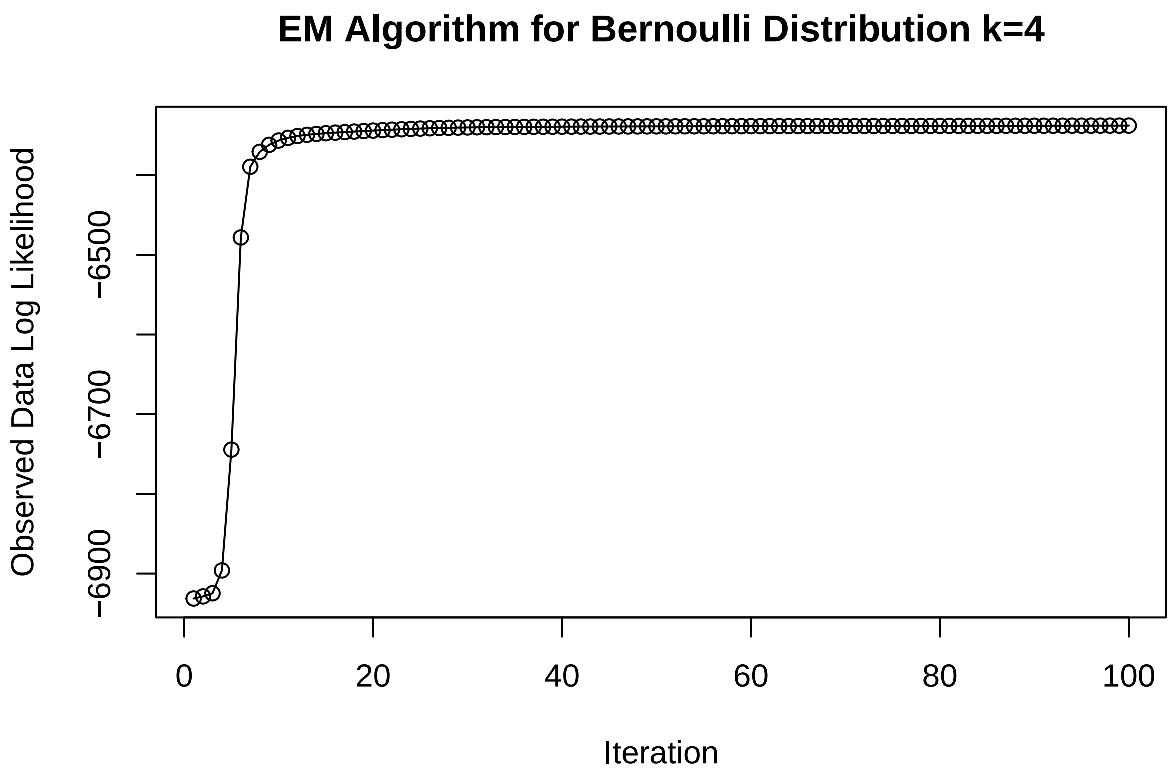
Again when K=3 three bernoulli distributions mixure model produced below result which is similar to the true distribution. The plotting of observed data log-likelihood as a function of the iteration number is produced by the EM algorithm is given next for K=3. [Updated]





Again when K=4 three bernoulli distributions mixure model produced below result two of them are similar to true distribution and third and fourth becomes random. The plotting of observed data log-likelihood as a function of the iteration number is produced by the EM algorithm is given next for K=4.[Updated]





So to sum up,the log likelihood plot describes the maximum log likelihood of each iteration. With the increase of iteration from 15 log likelihood becomes more consistent. With the changing of parameters the log likelihood is also changed but after 15 iteration it becomes consistent. For mixture component K=2 the model underfits the data where as when mixture component is 4 it overfits the data. But using K =3 the output model is almost similar to true data.[Updated]

```r
##Adaboost

data<-read.csv("/home/george/Documents/732A95/lab1_block2/spambase.csv", sep=";", dec=",")
n=dim(data)[1]
set.seed(12345)
id=sample(1:n, floor(n*0.75))
train=data[id,]
test=data[-id,]
library(mboost)
set.seed(1234)
#no of trees 10,20...,100
trees=seq(10,100,by=10)
adaBoost<-c()
misclassification_ada<-c()
#loop over the lenght of tree
for(i in 1:length(trees))
{
adaBoost[[i]] <- blackboost(as.factor(Spam) ~ .,
                       data = train,
                       family = AdaExp(),
                       control = boost_control(trees[i]))

ada_test <- predict(adaBoost[[i]], newdata=test, type='class')
#confusion matrix
confusion_mat_ada<-table(ada_test,as.factor(test$Spam))
#finding the misclassification
misclassification_ada[i]<- 1-(sum(diag(confusion_mat_ada)))/sum(confusion_mat_ada)
}
misclassification_ada

##Random forest
library(randomForest)
##random forest train values for different number of trees
rf_train<-c()
misclassification_rf<-c()

for(i in 1:length(trees))
{
rf_train[[i]] <- randomForest(as.factor(Spam) ~ .,
                   data = train,ntree=trees[i])
rf_test <- predict(rf_train[[i]], newdata=test, type='class')
#confusion matrix
confusion_mat_rf<-table(rf_test,as.factor(test$Spam))
#misclassification rate
misclassification_rf[i]<- 1-(sum(diag(confusion_mat_rf)))/sum(confusion_mat_rf)
}
misclassification_rf

#Ploting data

plot(y=misclassification_ada,x=trees,type="l",col="red",ylim=c(0,0.15),ylab="misclassification")
lines(y=misclassification_rf,x=trees,type="l",col="blue",ylim=c(0,0.15))
legend("topright", legend=c("adaboost", "randomforest"),
       col=c("red", "blue"), lty=1:2, cex=0.8)


####Assignment 2

#Mixture models
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   #matrix 100x10
true_pi <- vector(length = 3) # true mixing coefficients
true_mu <- matrix(nrow=3, ncol=D) # true conditional distributions  #matrix 3x10
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   #prob = 1 and 0
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=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) {
   px <- integer(N)

for(n in 1:1000){
   probSum <- 0
   for(k in 1:K){
   probSum <- probSum + pi[k]*prod((mu[k,]^x[n,])*(1-mu[k,])^(1-x[n,]))
     }
      px[n] <- probSum
    }

  for(n in 1:1000){
    for(k in 1:K){
       z[n,k]<- (pi[k]*prod((mu[k,]^x[n,])*(1-mu[k,])^(1-x[n,])))/px[n]
    }
  }

  #Log likelihood computation. #llik
  # Your code here
  logL <- matrix(0, nrow = 1000, ncol = K)
  for(n in 1:1000){
    for(k in 1:K){
      logL[n,k] <- (pi[k]*(prod((mu[k,]^x[n,])*(1-mu[k,])^(1-x[n,]))))

    }
  llik[it] <- sum(log(rowSums(logL)))
  }

  cat("iteration: ", it, "log likelihood: ", llik[it], "\n")
  pi<-colSums(z)/N

  mu<-(t(z)%*%x)/colSums(z)
  }
```

```
plot(mu[1,], type="o", col="blue", ylim=c(0,1),xlab ="Variable",ylab = "mu")
points(mu[2,], type="o", col="red")
plot(llik[1:it], type="o",main="EM Algorithm for Bernoulli Distribution K=2",
xlab = "Iteration",ylab = "Observed Data Log Likelihood")

#------------------------------------------------------------
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) {
   px <- integer(N)

  for(n in 1:1000){
    probSum <- 0
    for(k in 1:K){
      probSum <- probSum + pi[k]*prod((mu[k,]^x[n,])*(1-mu[k,])^(1-x[n,]))
    }
    px[n] <- probSum
  }

  for(n in 1:1000){
    for(k in 1:K){
      z[n,k]<- (pi[k]*prod((mu[k,]^x[n,])*(1-mu[k,])^(1-x[n,])))/px[n]
    }
  }

  #Log likelihood computation. #llik
  # Your code here
  logL <- matrix(0, nrow = 1000, ncol = K)
  for(n in 1:1000){
    for(k in 1:K){
      logL[n,k] <- (pi[k]*(prod((mu[k,]^x[n,])*(1-mu[k,])^(1-x[n,]))))

    }
    llik[it] <- sum(log(rowSums(logL)))
  }

  cat("iteration: ", it, "log likelihood: ", llik[it], "\n")
    pi<-colSums(z)/N

  mu<-(t(z)%*%x)/colSums(z)
}
```

```
plot(mu[1,], type="o", col="blue", ylim=c(0,1),xlab ="Variable",ylab = "mu")
points(mu[2,], type="o", col="red")
points(mu[3,], type="o", col="green")
plot(llik[1:it], type="o",main="EM Algorithm for Bernoulli Distribution K=3",
     xlab = "Iteration",ylab = "Observed Data Log Likelihood")

#------------------------------------------------


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) {
  px <- integer(N)

  for(n in 1:1000){
    probSum <- 0
    for(k in 1:K){
      probSum <- probSum + pi[k]*prod((mu[k,]^x[n,])*(1-mu[k,])^(1-x[n,]))
    }
    px[n] <- probSum
  }

  for(n in 1:1000){
    for(k in 1:K){
      z[n,k]<- (pi[k]*prod((mu[k,]^x[n,])*(1-mu[k,])^(1-x[n,])))/px[n]
    }
  }

  #Log likelihood computation. #llik
  # Your code here
  logL <- matrix(0, nrow = 1000, ncol = K)
  for(n in 1:1000){
    for(k in 1:K){
      logL[n,k] <- (pi[k]*(prod((mu[k,]^x[n,])*(1-mu[k,])^(1-x[n,]))))

    }
    llik[it] <- sum(log(rowSums(logL)))
  }

  cat("iteration: ", it, "log likelihood: ", llik[it], "\n")
 pi<-colSums(z)/N
```

```r
  mu<-(t(z)%*%x)/colSums(z)
}



plot(mu[1,], type="o", col="blue", ylim=c(0,1),xlab ="Variable",ylab = "mu")
points(mu[2,], type="o", col="red")
points(mu[3,], type="o", col="green")
points(mu[4,], type="o", col="yellow")
plot(llik[1:it], type="o",main="EM Algorithm for Bernoulli Distribution K=4",
     xlab = "Iteration",ylab = "Observed Data Log Likelihood")
```