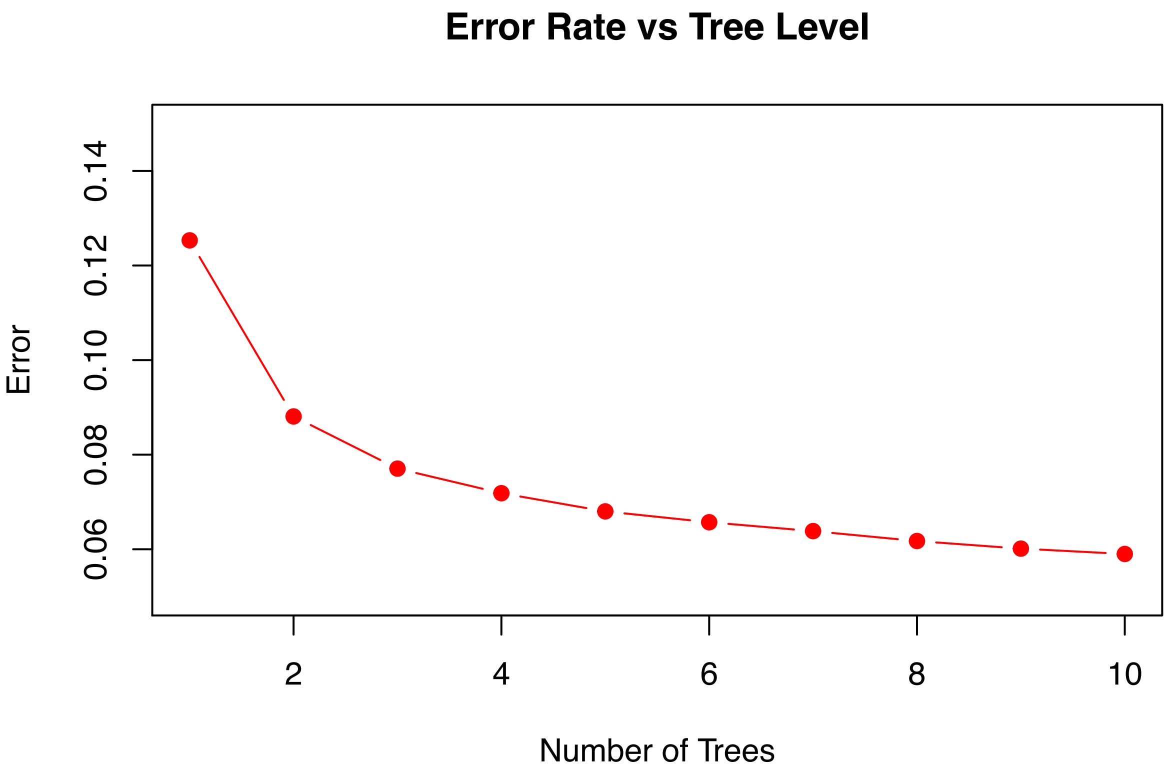
Machine learning Group8 - Lab1 block 2

Group-8

November 28, 2017

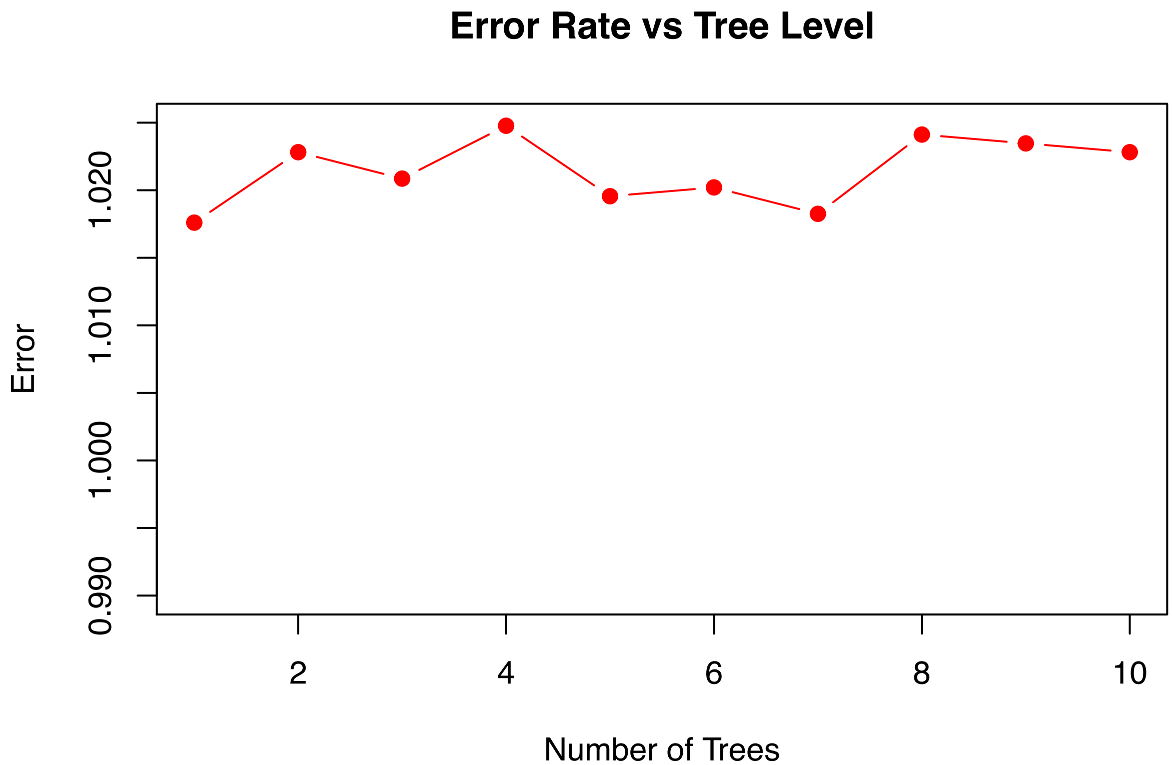
Assignment 1

Adaboost



Loss Function for the selected family is $LossFunction = (y - f)^2$

Random Forest

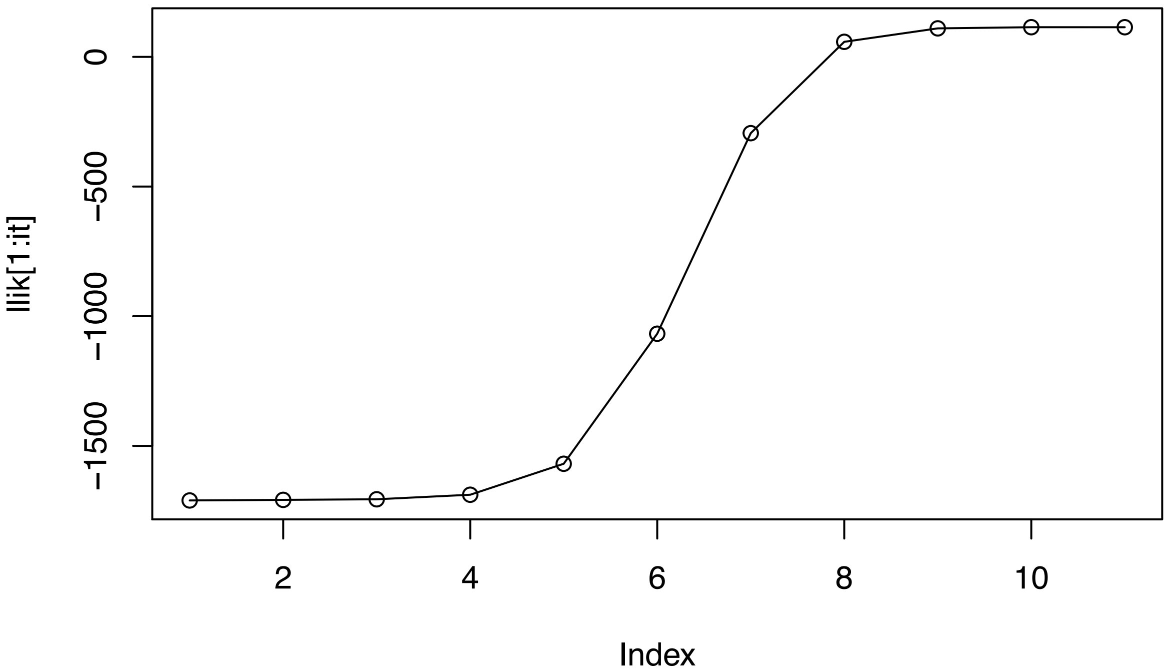


Assignment 2

Mixture Model

For K=2

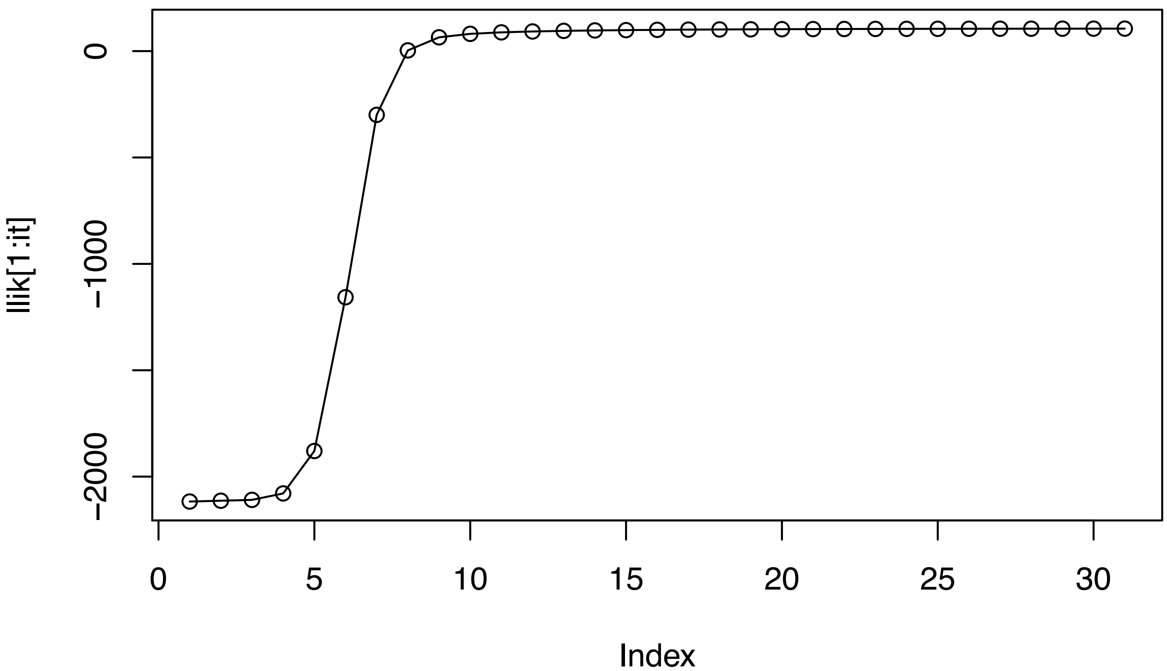
```
## iteration:  1 log likelihood:  -1710.416
## iteration:  2 log likelihood:  -1708.212
## iteration:  3 log likelihood:  -1705.908
## iteration:  4 log likelihood:  -1688.444
## iteration:  5 log likelihood:  -1568.968
## iteration:  6 log likelihood:  -1067.461
## iteration:  7 log likelihood:  -294.244
## iteration:  8 log likelihood:  58.06127
## iteration:  9 log likelihood:  109.9385
## iteration:  10 log likelihood:  114.3999
## iteration:  11 log likelihood:  114.3286
```



For K=3

```
## iteration:  1 log likelihood:  -2116.967
## iteration:  2 log likelihood:  -2113.26
## iteration:  3 log likelihood:  -2109.279
## iteration:  4 log likelihood:  -2078.903
## iteration:  5 log likelihood:  -1879.558
## iteration:  6 log likelihood:  -1156.973
## iteration:  7 log likelihood:  -299.6505
## iteration:  8 log likelihood:  3.388555
## iteration:  9 log likelihood:  64.81808
## iteration:  10 log likelihood:  81.13098
## iteration:  11 log likelihood:  88.18021
## iteration:  12 log likelihood:  92.315
## iteration:  13 log likelihood:  95.11267
## iteration:  14 log likelihood:  97.15403
## iteration:  15 log likelihood:  98.71629
## iteration:  16 log likelihood:  99.95156
## iteration:  17 log likelihood:  100.9507
## iteration:  18 log likelihood:  101.7718
## iteration:  19 log likelihood:  102.4538
## iteration:  20 log likelihood:  103.0246
## iteration:  21 log likelihood:  103.5045
## iteration:  22 log likelihood:  103.9095
## iteration:  23 log likelihood:  104.2518
## iteration:  24 log likelihood:  104.5414
## iteration:  25 log likelihood:  104.7866
## iteration:  26 log likelihood:  104.994
## iteration:  27 log likelihood:  105.1692
## iteration:  28 log likelihood:  105.3169
## iteration:  29 log likelihood:  105.4411
```

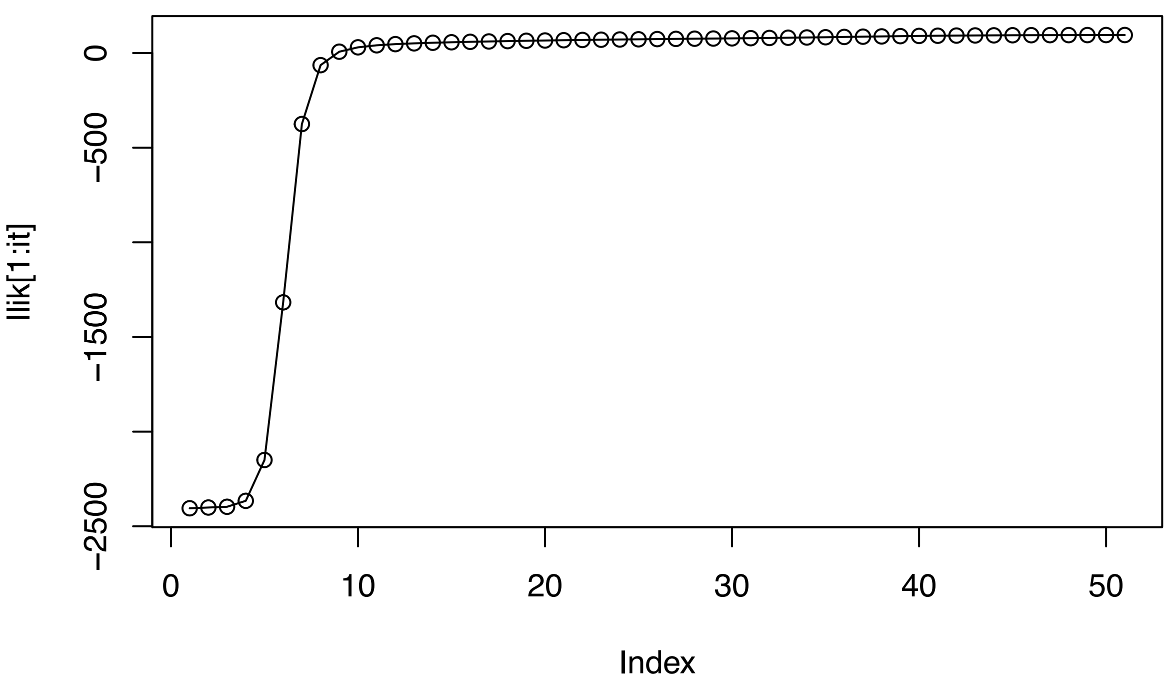
```
## iteration:  30 log likelihood:  105.5451
## iteration:  31 log likelihood:  105.6317
```



For K=4

```
## iteration:  1 log likelihood:  -2404.83
## iteration:  2 log likelihood:  -2400.939
## iteration:  3 log likelihood:  -2396.925
## iteration:  4 log likelihood:  -2365.575
## iteration:  5 log likelihood:  -2149.829
## iteration:  6 log likelihood:  -1317.208
## iteration:  7 log likelihood:  -375.1612
## iteration:  8 log likelihood:  -64.09552
## iteration:  9 log likelihood:  6.910748
## iteration:  10 log likelihood:  30.05542
## iteration:  11 log likelihood:  40.76704
## iteration:  12 log likelihood:  46.91076
## iteration:  13 log likelihood:  51.04236
## iteration:  14 log likelihood:  54.18383
## iteration:  15 log likelihood:  56.78241
## iteration:  16 log likelihood:  59.0457
## iteration:  17 log likelihood:  61.07583
## iteration:  18 log likelihood:  62.92663
## iteration:  19 log likelihood:  64.62901
## iteration:  20 log likelihood:  66.20267
## iteration:  21 log likelihood:  67.66166
## iteration:  22 log likelihood:  69.01716
## iteration:  23 log likelihood:  70.27908
## iteration:  24 log likelihood:  71.45698
## iteration:  25 log likelihood:  72.56097
## iteration:  26 log likelihood:  73.60253
## iteration:  27 log likelihood:  74.59534
```

```
## iteration:  28 log likelihood:  75.55626
## iteration:  29 log likelihood:  76.50611
## iteration:  30 log likelihood:  77.46999
## iteration:  31 log likelihood:  78.47646
## iteration:  32 log likelihood:  79.5548
## iteration:  33 log likelihood:  80.72966
## iteration:  34 log likelihood:  82.01349
## iteration:  35 log likelihood:  83.39881
## iteration:  36 log likelihood:  84.85428
## iteration:  37 log likelihood:  86.32793
## iteration:  38 log likelihood:  87.75797
## iteration:  39 log likelihood:  89.08701
## iteration:  40 log likelihood:  90.27337
## iteration:  41 log likelihood:  91.29589
## iteration:  42 log likelihood:  92.15209
## iteration:  43 log likelihood:  92.85274
## iteration:  44 log likelihood:  93.4158
## iteration:  45 log likelihood:  93.86161
## iteration:  46 log likelihood:  94.2098
## iteration:  47 log likelihood:  94.47779
## iteration:  48 log likelihood:  94.68027
## iteration:  49 log likelihood:  94.82922
## iteration:  50 log likelihood:  94.93427
## iteration:  51 log likelihood:  95.00306
```



Ideally, mixture model should run for parameters that are not too small nor too big to avoid the underfitting and ovverfitting of the model. For too few parameters that is for K=2, the logliklihood function runs for 11 iterations giving μ near to the true values of μ while for too many parameters (K=4) the convergence steps increases resulting in overfitting. For K=3, the logliklihood value increases significantly giving μ_2 that is μ for K=2 and μ_3 that is μ for K=3 near the true values.

APPENDIX

```
## Question 1

spambase <- read.csv2("spambase.csv", header = TRUE, sep = ";", quote = "\"",
                      dec = ",", fill = TRUE)
spambase <- as.data.frame(spambase)

### Adaboost

n=dim(spambase)[1]
set.seed(12345)
id=sample(1:n, floor(n*2/3))
train=spambase[id,]
test=spambase[-id,]


number_of_trees <- seq(from = 10,to = 100, by = 10)

adaboost <- function(ntrees)
{
fit <- blackboost(Spam ~., data = train,
           control = boost_control(mstop = ntrees, nu=0.1),
           family = Gaussian())

ypredict <- predict(fit, test)

error <- mean((test$Spam - ypredict)^2)
}

error_rates_a <- sapply(number_of_trees, adaboost)

plot(error_rates_a,type = "b",main="Error Rate vs Tree Level", xlab= "Number of Trees",
     ylab= "Error",ylim=c(0.05,0.15), col="red", pch=19, cex=1)


### Random Forest


training = sample(1:n,floor(n*2/3))

random_forest <- function(ntrees)
{
  fit <- randomForest(as.factor(Spam) ~ ., data=spambase, subset = training, importance=TRUE,
                      ntree = ntrees)

  ypredict <- as.numeric(predict(fit, test))

  error <- mean((test$Spam - ypredict)^2)
}

error_rates_f <- sapply(number_of_trees, random_forest)
```

```r
plot(error_rates_f,type = "b",main="Error Rate vs Tree Level", xlab= "Number of Trees", ylab= "Error",
     ylim=c(1.01,1.03), col="red", pch=19, cex=1)
## Question 2

mixture_model <- function(my_k)
{
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=my_k # 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(j in 1:my_k) {
   mu[j,] <- runif(D,0.49,0.51)
}
pi
mu

for(it in 1:max_it)
  {
  if(K == 2)
  {
  plot(mu[1,], type="o", col="blue", ylim=c(0,1))
  points(mu[2,], type="o", col="red")
  }
```

```
else if(K==3)
{
  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")
}
else
{
  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 assignment

# Bernoulli distribution
for (n in 1:N)
{
  prob_x=0

  for (k in 1:K)
  {
    prob_x=prob_x+prod( ((mu[k,]^x[n,])*((1-mu[k,])^(1-x[n,]))) )*pi[k] #
  }

  for (k in 1:K)
  {

    z[n,k]=pi[k]*prod( ((mu[k,]^x[n,])*((1-mu[k,])^(1-x[n,]))) ) / prob_x

  }

}

#Log likelihood computation.

likelihood <-0
llik[it] <-0
for(n in 1:N)
{

  for (k in 1:K)
  {
    likelihood <-likelihood + z[n,k]*(pi[k]*prod( ((mu[k,]^x[n,])*((1-mu[k,])^(1-x[n,])))))
  }
  llik[it]<-llik[it]+log(likelihood)
}


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

```r
  # Stop if the lok likelihood has not changed significantly
  if (it > 1)
  {
    if (llik[it]-llik[it-1] < min_change)
    {
      break
    }
  }

  #M-step: ML parameter estimation from the data and fractional component assignments

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

  # N - Total no. of observations
  pi <- colSums(z)/N
}

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

mixture_model(2)
mixture_model(3)
mixture_model(4)
}
```