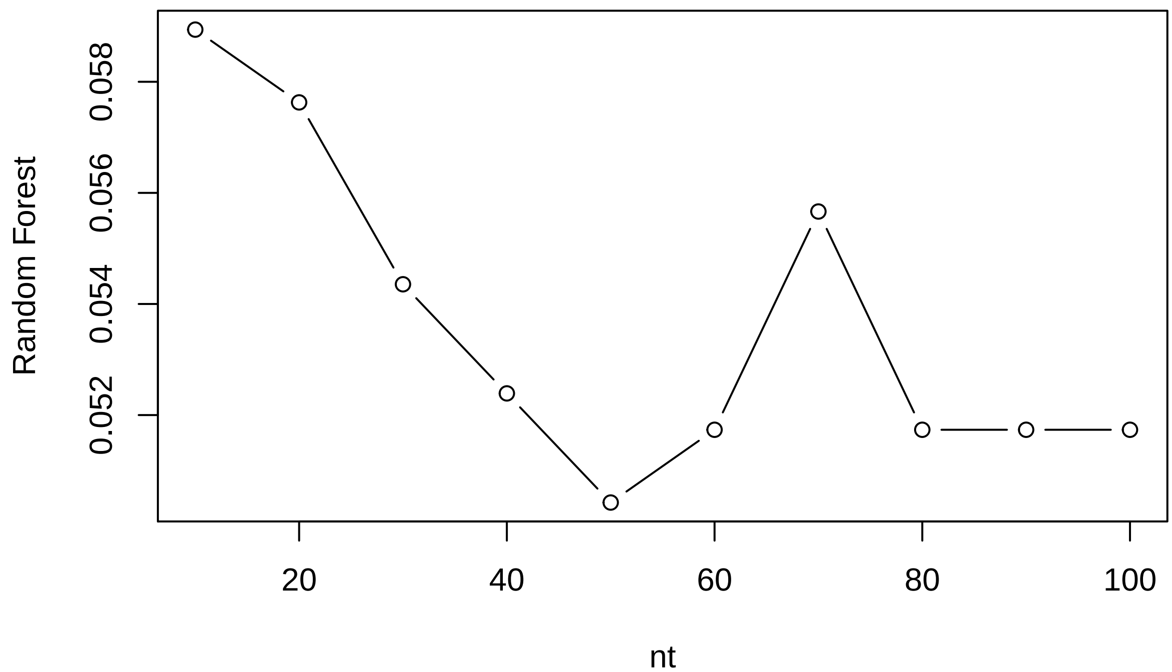
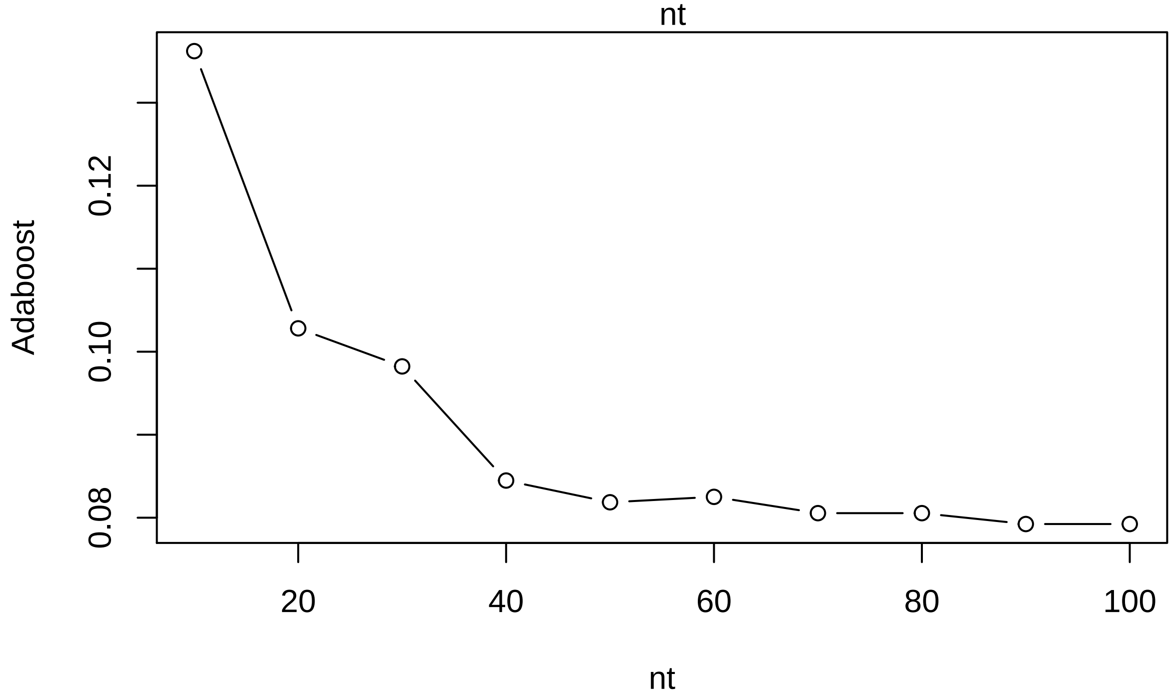
Zhixuan_Duan

2019-12-3

Assignment 1. ENSEMBLE METHODS

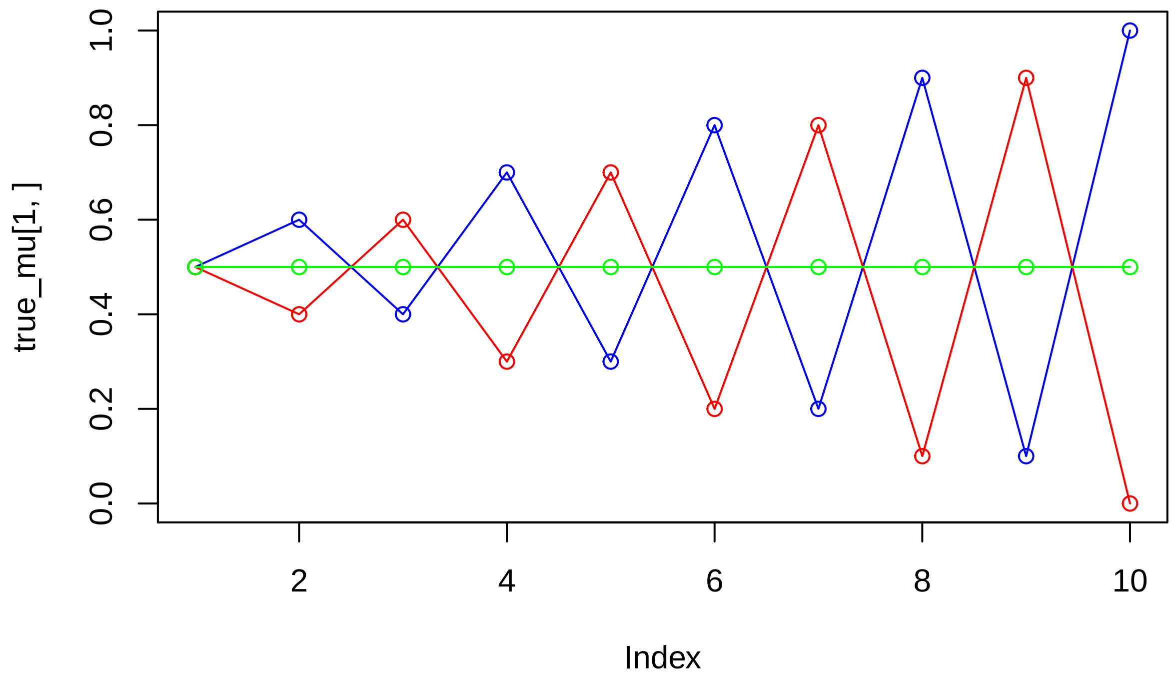




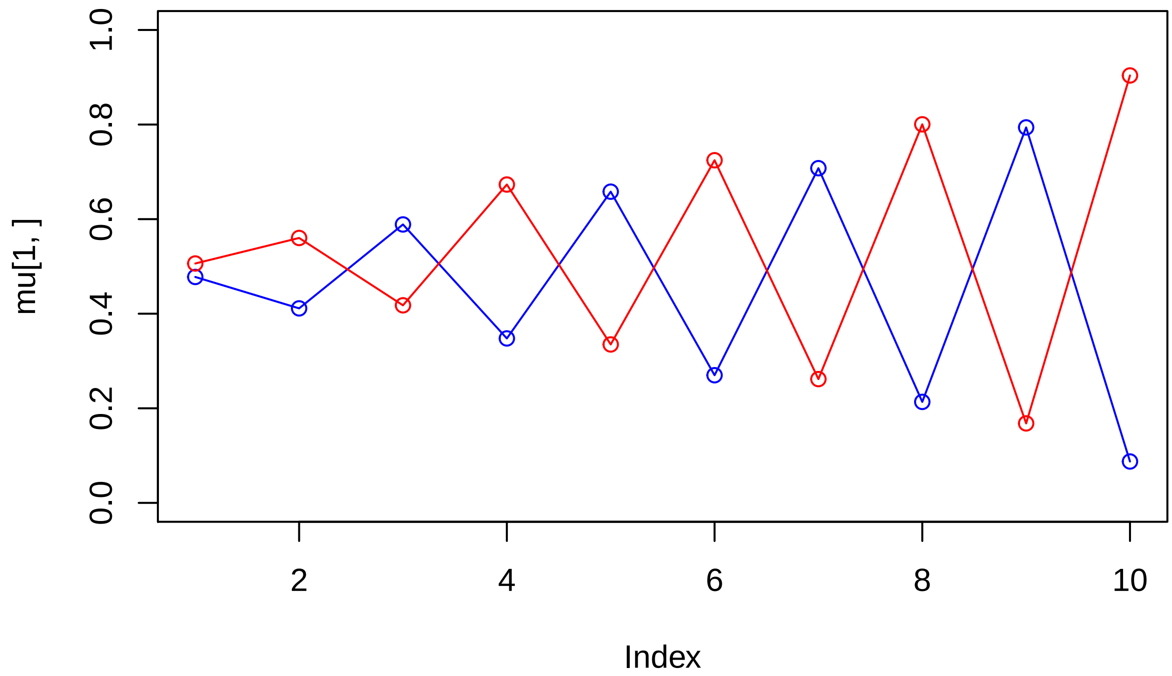
From the two plots, we can see that as the number of tree grows, the error rate of both args decrease, and the adaboost seems more stable than random forest.

Assignment 2. MIXTURE MODELS

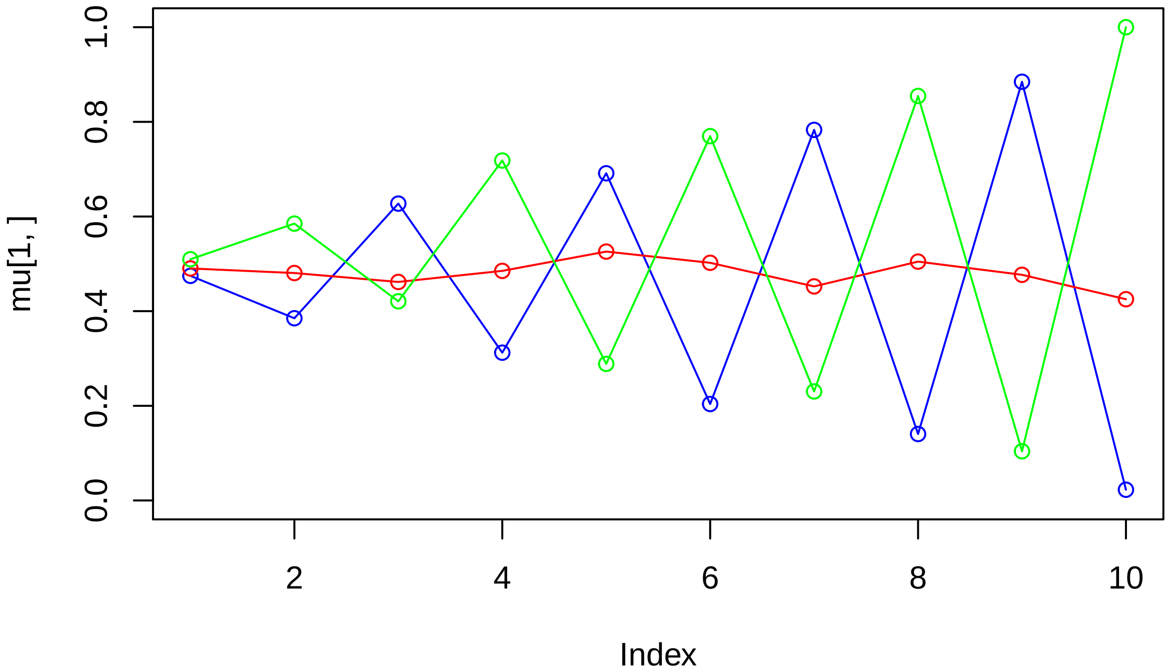
This plot shows the true distribution with three components and same weights to mixture model.



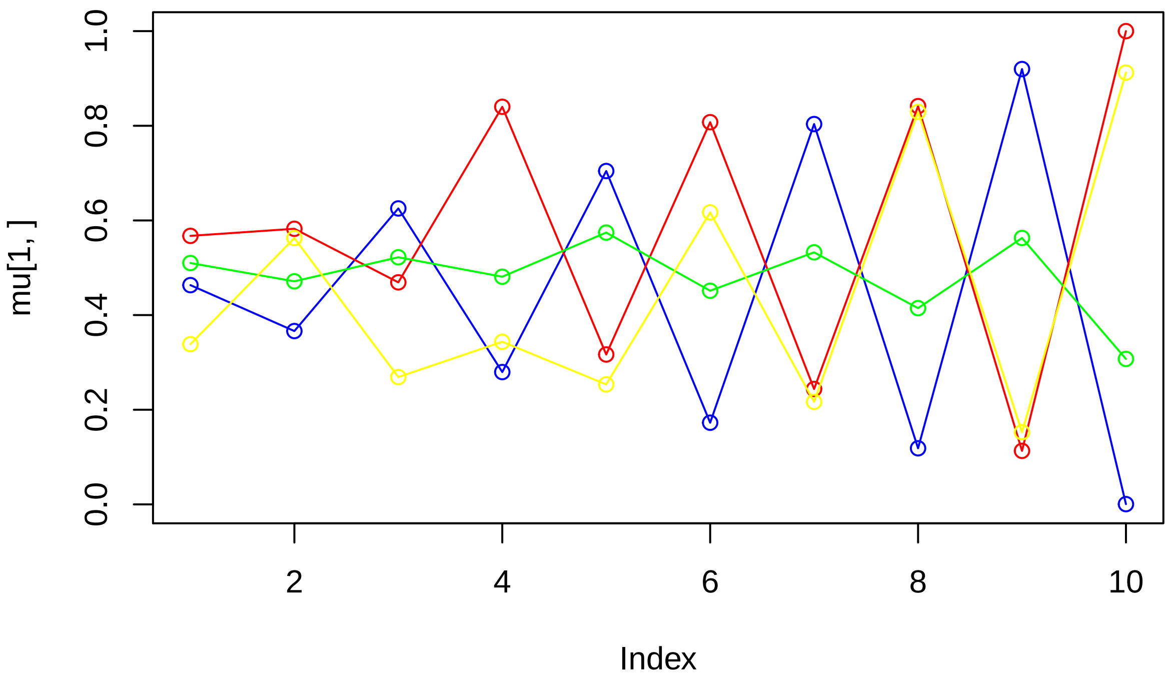
When the K equals 2, here is the final result.



When the K equals 3, here is the final result.



When the K equals 4, here is the final result.



Compare the final plots when K equals 2,3 and 4, when mixture model has too few components, it takes more iterations to approximate the true model, and it may lack of accuracy, when there are too many components, it could reduce the cycle number, however, it may overfit the model.

Appendix

```r
## Assignment 1. ENSEMBLE METHODS
# read the dataset
sp <- read.csv2('/Users/darin/Desktop/ML/spambase.csv')
sp$Spam <- as.factor(sp$Spam)

#install.packages('mboost')
#install.packages('randomForest')
library(mboost)
library('randomForest')

# create train and test data
set.seed(123)
sub <- sample(2, nrow(sp), replace = TRUE, prob = c(2/3, 1/3))
train <- sp[sub == 1,]
test <- sp[sub == 2,]

alist <- c(rep(0,10))
blist <- c(rep(0,10))
nt <- seq(10,100,10)
for (i in nt) {
####RandomForest####
rf <- randomForest(Spam ~ ., data = train, ntree = i, importance = TRUE)
pred1 <- predict(rf, newdata = test)
a <- table(observed=test$Spam,predicted=pred1)
# error rate of rf
acca1 <- ifelse(test$Spam == pred1, 1, 0)
acca2 <- sum(acca1)/nrow(test)
alist[i/10] <- alist[i/10] + (1-acca2)

####Adaboost####
ada <- blackboost(Spam ~ ., data = train, family = AdaExp(),
                  control = boost_control(mstop = i))
pred2 <- predict(ada, newdata = test, type = c("class"))
b <- table(ovserved = test$Spam, predicted = pred2)
# error rate of adaboost
accb1 <- ifelse(test$Spam == pred2, 1, 0)
accb2 <- sum(accb1)/nrow(test)
blist[i/10] <- blist[i/10] + (1-accb2)
}
alist
rfdata <- cbind(nt, alist)
plot(rfdata, type = "b", ylab = "Random Forest", main = "Error rate of RF")
blist
adadata <- cbind(nt,blist)
plot(adadata, type ="b", ylab = "Adaboost", main = "Error rate of Ada")

## 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
```

```r
D=10 # number of dimensions

x <- matrix(nrow=N, ncol=D) # training data
true_pi <- vector(length = 3) # true weight
true_mu <- matrix(nrow=3, ncol=D) # true distribution
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(i in 1:D) {
    x[n,i] <- rbinom(1,1,true_mu[k,i])
  }
}

K=3 # guess the number of components
z <- matrix(nrow=N, ncol=K) # the p of ith obser from kth distri
pi <- vector(length = K) # guess weight
mu <- matrix(nrow=K, ncol=D) # guess distri
llik <- vector(length = max_it) # log likelihood

# inital parameter
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: compute z
  for (n in 1:N) {
    # create a p matrix
    p_x = matrix(c(rep(1,K),0), nrow = 1, ncol = K+1)

    for (k in 1:K) {
      # compute the posterior p
      for (i in 1:D) {
        p_x[1,k] <- p_x[1,k] * (mu[k,i]^x[n,i]) * (1-mu[k,i])^(1-x[n,i])
```

```r
      }
      p_x[1,k] <- p_x[1,k] * pi[k]
      # sum them up
      p_x[1,K+1] <- p_x[1,K+1] + p_x[1,k]
    }
    # get the z matrix
    for (k in 1:K) {
      z[n,k] <- p_x[1,k] / p_x[1,K+1]
    }
  }

  # compute llk
  for (n in 1:N) {
    for (k in 1:K) {
      lg <- 0
      for (i in 1:D) {
        lg <- lg + x[n,i] * log(mu[k,i]) + (1-x[n,i]) * log(1-mu[k,i])
      }
      llik[it] <- llik[it] + z[n,k] * (log(pi[k]) + lg)
    }
  }
  cat("iteration:", it, "log_likelihood:", llik[it], "\n")
  flush.console()

  # M-step
  # update weight
  for (k in 1:K) {
    pi[k] <- sum(z[,k])/N
  }
  # update distri
  for (k in 1:K) {
    mu[k,] = 0
    for (n in 1:N) {
      mu[k,] = mu[k,] + x[n,] * z[n,k]
    }
    mu[k,] = mu[k,] / sum(z[,k])
  }
  # set threshold
  if (it != 1) {
    if (abs(llik[it] - llik[it-1]) < min_change) {
      break
    }
  }
}
```