XIAOYU WANG

1): Implement the AdaBoost algorithm in R

```r
adaBoost <- function(X,y,B) {

  w <- rep(1/nrow(X),nrow(X))
  alpha <- rep(0,B);alpha
  allPars <- list(rep(list(), B))

  for(b in 1:B) {

      err_cv <- c()
      classifier <- list(rep(list(), 5))

      #cv k-folds
      for(i in 1:5) {

      valid_X <- X[i:(i + nrow(X)/5),]
      train_X <- X[-(i:(i + nrow(X)/5)),]
      valid_y <- y[i:(i + nrow(X)/5)]
      train_y <- y[-(i:(i + nrow(X)/5))]
      valid_w <- w[i:(i + nrow(X)/5)]
      train_w <- w[-(i:(i + nrow(X)/5))]

      #find classifier
      classifier[[i]] <- train(train_X, train_w, train_y)

      #check error rate on the valid set
      pred_cv <- classify(valid_X, classifier[[i]])
      err_cv[i] <- sum(pred_cv != valid_y) / length(valid_y)

      }

     #find cv classifiers
     allPars[[b]] <- classifier[[which.min(err_cv)]]
     label <- classify(X,allPars[[b]])
     #check missclassified
     miss <- sign(label != y)
     #compute error & alpha
     err <- (sum(w * miss)/sum(w))
     alpha[b] <- log((1 - err) / err);alpha[b]

     #compute new w
     w <- w * exp(alpha[b] * miss)
  }

return(list(allPars = allPars , alpha = alpha))

  }
```

aggregated classifier

```
agg_class = function(X, alpha, allPars) {

  fx <- rep(0, nrow(X))
  n <- length(alpha)
  n
  for(i in 1 : n) {
    fx <- fx + alpha[i] * classify(X, allPars[[i]])
  }

  label <- sign(fx)
  return(label)
}
```

Part 2): Implement the functions train and classify for decision stumps.

```
#define train funciton
train = function(X, w, y) {

  d <- ncol(X);d
  theta <- rep(0,d)
  m <- rep(0,d)
  rate <- rep(0,d)

  for(i in 1 : d) {

    theta[i] <- runif(1, min(X[,i]), max(X[,i]))
    m[i] <- 1
    pred_label <- sign(m[i] * (X[,i] - theta[i]))
    rate[i] <- sum(pred_label != y) / nrow(X)

  #choose weak learner
    if(rate[i] > 0.5) {
      m[i] <- -1
      pred_label <- sign(m[i] * (X[,i] - theta[i] ))
      rate[i] <- sum(pred_label != y) / nrow(X)
      }

  }

  j <- which.min(rate)
  theta <- theta[j]
  m <- m[j]

  return(list(j = j, theta = theta, m = m))
}
```

```
# weak learner classification routine
classify = function(X, pars) {
  classifier <- sign(pars$m * (X[,pars$j] - pars$theta))
  return(classifier)

  }
```

part 3): run your algorithm on the USPS data and evaluate your results using cross validation.

```
library(ggplot2)
```

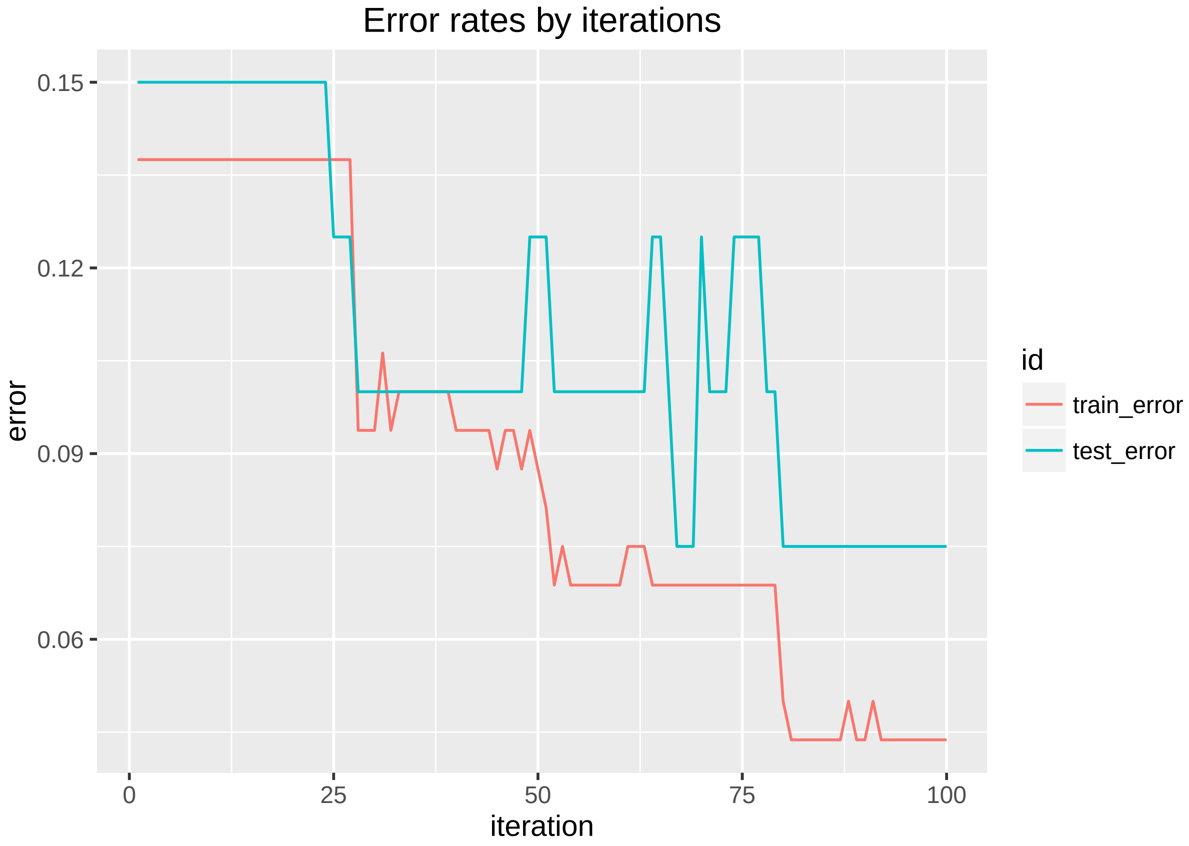
```
## Warning: package 'ggplot2' was built under R version 3.2.4
```

```
set.seed(211)
label <- read.table("~/Desktop/uspscl.txt")
data <-read.table("~/Desktop/uspsdata.txt")
#split dataset
i <- sample(1:200,160)
train_data <- data[i,]
test_data <- data[-i,]
train_label<- label[i,]
test_label<- label[-i,]

X <- train_data
y <- train_label
B <- 100

adaBoost <- adaBoost(X,y,B)
allPars <- adaBoost$allPars
alpha <- adaBoost$alpha

train_error <- c()
test_error <- c()
#compute train error
for(b in 1:B) {

train <- agg_class(X, alpha[1:b], allPars[1:b])
train_error[b] <- sum(train != y ) / nrow(X)

test <- agg_class(test_data, alpha[1:b], allPars[1:b])
test_error[b] <- sum(test != test_label) / nrow(test_data)

}
```

part 4): Plot the training error and the test error as a function of b

```
train <- data.frame(iteration = 1:B, error = train_error, id = rep("train_error",B))
test <- data.frame(iteration = 1:B, error = test_error, id = rep("test_error",B))
```

```r
result <- rbind(train, test)
ggplot(result, aes(iteration, error)) + geom_line(aes(color = id)) + labs(title = "Error rates by itera
```


Error rates by iterations
error
iteration
id
train_error
test_error