# Homework 4

*Yi Chen(yc3356)*

*April 2, 2018*

# Homework 4

## train function

```
# 1.Decision stumps
train<-function(x,w,y){
  n<-nrow(x)
  p<-ncol(x)

  met<-matrix(nrow=p)
  theta<-matrix(nrow=p)
  loss<-matrix(nrow=p)

  for (j in 1:p){
    index<-order(x[,j])
    x_j<-x[index,j]
    w_cum<-cumsum(w[index]*y[index]) # compute cumulative sum
    w_cum[duplicated(x_j)==1]<-NA # multiple occurrences of same x_j value
    # optimal threshold
    m<-max(abs(w_cum), na.rm=TRUE)
    maxIndex<-min(which(abs(w_cum)==m))
    met[j]<-(w_cum[maxIndex]<0)*2 - 1
    theta[j] <- x_j[maxIndex]
    c <- ((x_j > theta[j])*2 - 1) * met[j]
    loss[j]<-w %*% (c!=y)
  }

  m<-min(loss)
  j_opt<-min(which(loss==m))
  pars<-list(j=j_opt,theta=theta[j_opt],mode=met[j_opt])
  return(pars)
}
```

## classify function

```
classify<-function(x,pars){
  j <- pars$j
  t <- pars$theta
  m <- pars$mode
  l <- x[,j]
  pred <- m * (l-t)
  pred[pred < 0] <- -1
  pred[pred >= 0] <- 1
  return(pred)
}
```

# adaboost function

```
# 1.AdaBoost algorithm
adaboost<-function(x,y,B){
  alpha<-rep(0,B)
  allPars<-rep(list(list()),B)
  n<-nrow(x)
  w<-rep(1/n, times=n) # for the first round we that all the weight as 1/w

  for (b in 1:B){
    allPars[[b]]<-train(x,w,y) # train base classifier
    missclass<-as.numeric(y!=classify(x,allPars[[b]])) # error
    e<-(w%*%missclass/sum(w))[1]
    alpha[b]<-log((1-e)/e) # voting weight
    w<-w*exp(alpha[b]*missclass) # recompute weight
  }

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

# agg_class function

```
# evaluate aggregated classifier on x
agg_class<-function(x,alpha,allPars){
  n<-nrow(x)
  B<-length(alpha)
  labels<-matrix(0,nrow=n,ncol=B)
  for(b in 1:B){
    labels[,b]<-classify(x,allPars[[b]])
  }
  labels<-labels %*% alpha
  c_hat<-sign(labels)
  return(c_hat)
}
```

# read the data

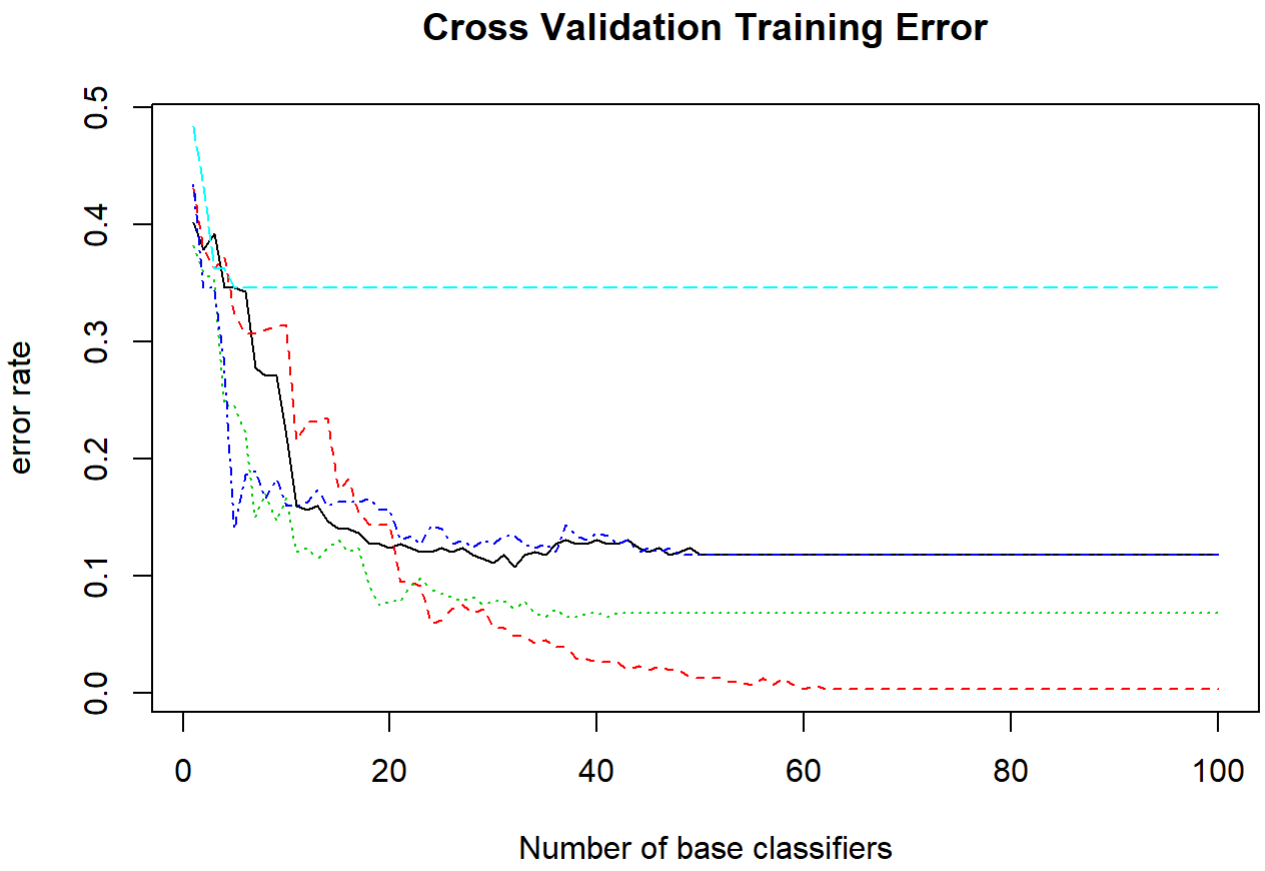
```
# 3.Run algorithm on USPS data, evaluate results using cross validation
train.3<-read.table("train_3.txt",header = FALSE, sep=",")
train.8<-read.table("train_8.txt",header = FALSE, sep=",")
xtrain<-rbind(as.matrix(train.3),as.matrix(train.8))
ytrain<-as.matrix(rep(c(-1,1),c(nrow(train.3),nrow(train.8))))
test<-as.matrix(read.table("zip_test.txt"))
ytest<-test[,1]
xtest<-test[ytest==3|ytest==8,-1]
ytest<-as.matrix(ytest[ytest==3|ytest==8])
ytest[ytest==3]<--1
ytest[ytest==8]<-1
# combine train and test for future cv
X<-rbind(xtrain,xtest)
Y<-rbind(ytrain,ytest)
```

# Cross Validation

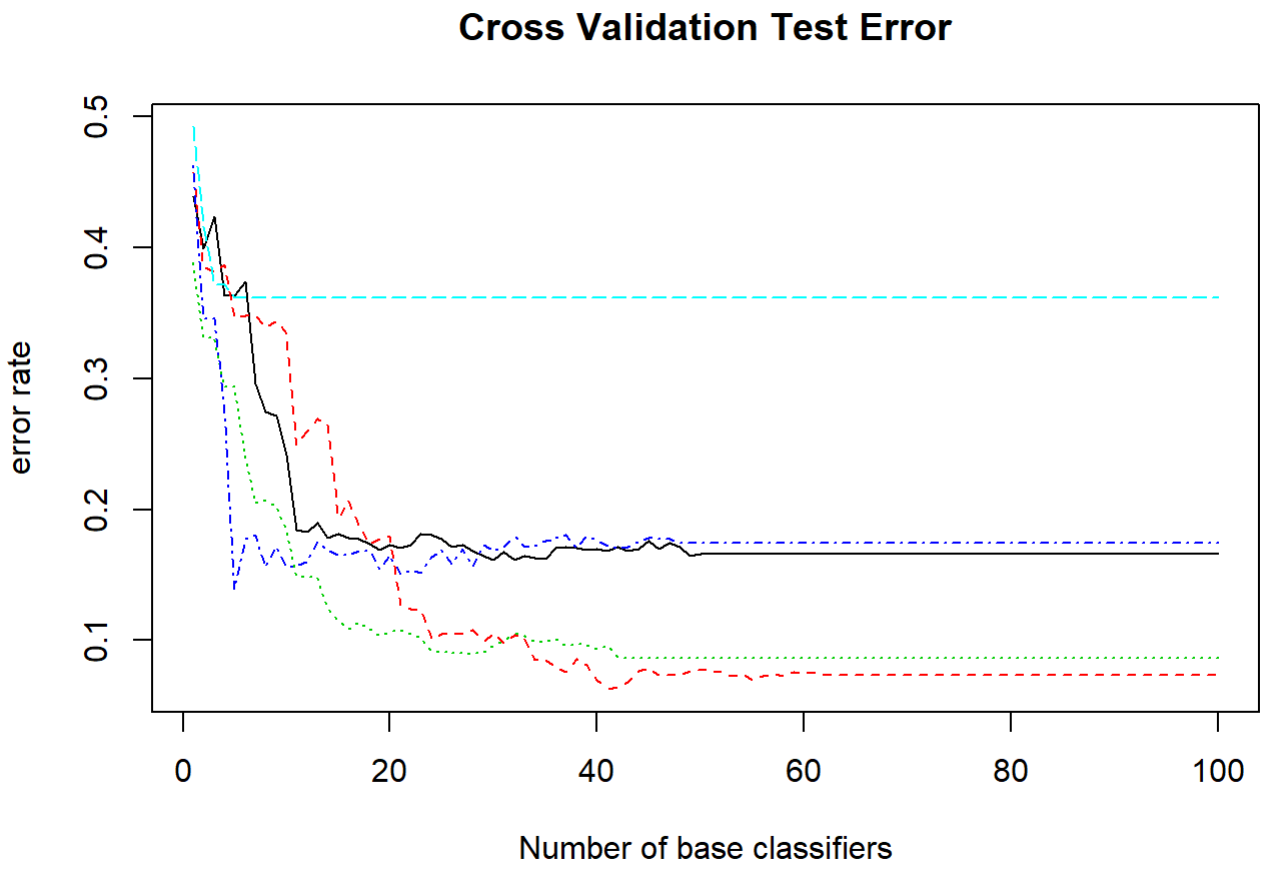
```
n<-nrow(X)
B_max<-100
nCV<-5

set.seed(1)

testErrorRate<-matrix(0,nrow=B_max,ncol=nCV)
trainErrorRate<-matrix(0,nrow=B_max,ncol=nCV)

p <- sample.int(n)

for (i in 1:nCV){
  trainIndex<-p[((i-1)*round(n/5)+1):(i*round(n/5))]
  testIndex<-p[-(((i-1)*round(n/5)+1):(i*round(n/5)))]
  ada<-adaboost(X[trainIndex,],Y[trainIndex],B_max)
  allPars<-ada$allPars
  alpha<-ada$alpha
  # error rate
  for(B in 1:B_max){
    c_hat_test<-agg_class(X[testIndex,],alpha[1:B],allPars[1:B])
    testErrorRate[B,i]<-mean(Y[testIndex] != c_hat_test)
    c_hat_train<-agg_class(X[trainIndex,],alpha[1:B],allPars[1:B])
    trainErrorRate[B,i]<-mean(Y[trainIndex] != c_hat_train)
  }
}
```

# Draw the plot

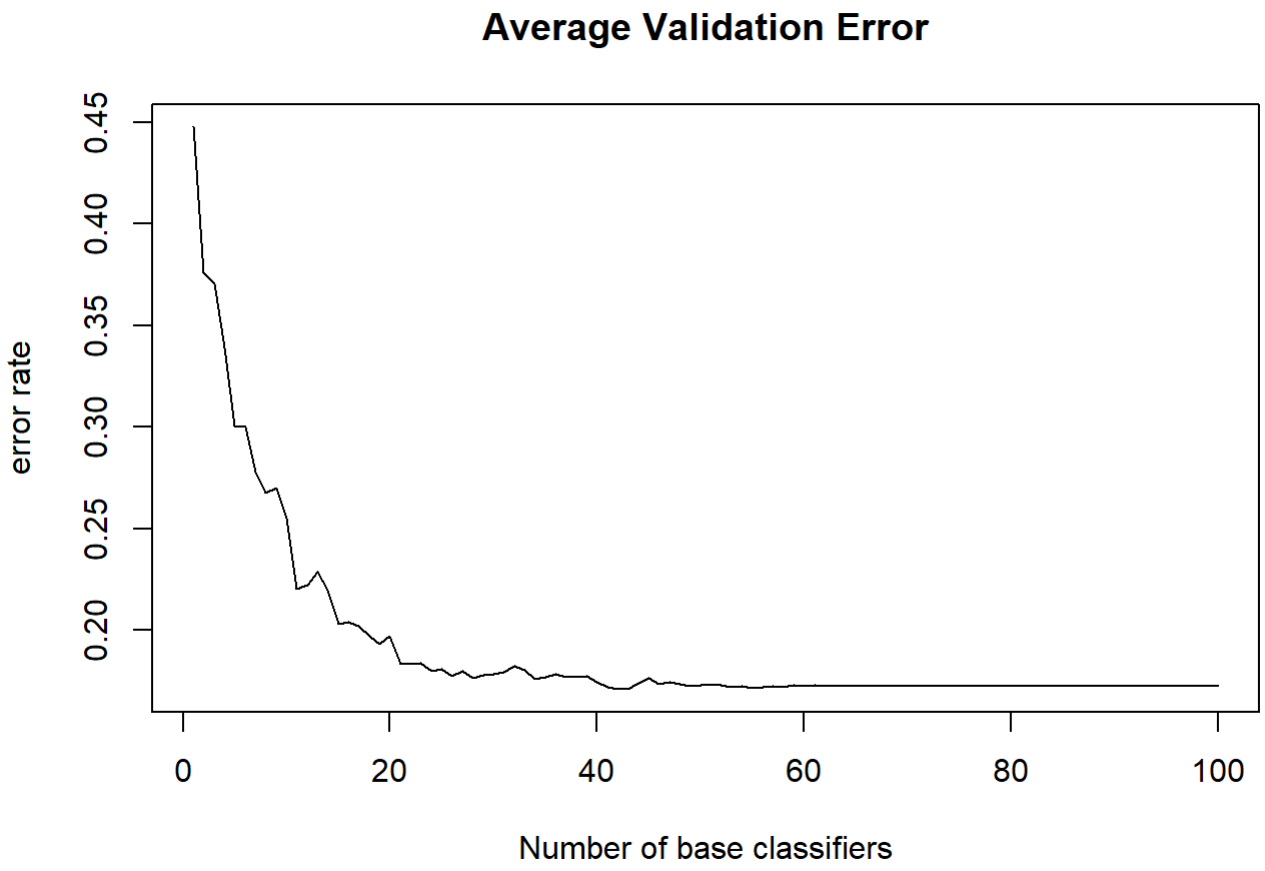
```
# 4. Plot train error and test error
matplot(trainErrorRate,type="l",lty=1:nCV,main="Cross Validation Training Error",xlab="Number of base clas
sifiers",ylab="error rate")
```




```
matplot(testErrorRate,type="l",lty=1:nCV,main="Cross Validation Test Error",xlab="Number of base classifiers",ylab="error rate")
```




```
# sum up the validation error rate for different B
Average_testErrorRate <- apply(testErrorRate,1,FUN = mean)
plot(Average_testErrorRate,type="l",lty=1:nCV,main="Average Validation Error",xlab="Number of base classifiers",ylab="error rate")
```




Thus, we can pick B = 50. Because after B=50 the error rate keeps same.