STA6704-19Spring 0001 : Homework for Boosting

Mithun Mohanraj

Ex. 10.4

A, Write a program implementing AdaBoost with trees.

R code for Adaboost with trees

```
#training and testing data
x=data.frame(matrix(rnorm(120000),12000,10))

y=2*as.numeric(apply(x,1,function(x){(sum(x^2)>9.34)}))-1
train.id = sample(c(1:12000),2000)

y_train<-y[train.id]
x_train<-x[train.id,]
y_test<-y[-train.id]
x_test<-x[-train.id,]
traindim=length(y_train)
testdim=length(y_test)


train_error<-rep(0,500)
# Keep track of errors
test_error<-rep(0,500)
f<-rep(0,traindim) # 2000pts in training data
f_test<-rep(0,testdim) # 10000 pts in test data
i<-1

#Adaboost
library(rpart)
while(i <= 500){
w <-exp(-y_train*f)

# This is a shortcut to compute w
w <-w/sum(w)
```

```
fit <-rpart(y_train ~.,data =x_train,weights = w, method="class")
g<--1 + 2*(predict(fit,x_train)[,2]>.5)
# make -1 or 1
g_test <--1+2*(predict(fit,x_test)[,2]>.5)
e <-sum(w*(y_train*g<0))
# If tree perfectly gets data, boosting terminates
if(abs(e) < 1e-8){
f=g;
f_test=g_test;
break
}
alpha <-.5*log ( (1-e) / e )
f <-f + alpha*g
f_test <-f_test + alpha*g_test
train_error[i] <-sum(1*f*y_train<0)/traindim
test_error[i] <-sum(1*f_test*y_test<0)/testdim
i<-i+1
}
```

#confusion matrix and error plots

```
y_hat_train = sign(f);
table(y_hat_train,y_train)
y_hat_test = sign(f_test);
table(y_test, y_hat_test)
#y_hat_test#y_test -1 1#-1 34 0#1 0 16
plot(seq(1,500),test_error,type="l",ylim=c(0,.5),ylab="Error
Rate",xlab="Iterations",lwd=2, main='AdaBoost: class1 class2')
lines(train_error,lwd=2,col="purple")
legend(50,.2,c("Training Error","Test Error"), col=c("purple","black"),lwd=2)
```

(b) Redo the computations for the example of Figure 10.2. Plot the training error as well as test error, and discuss its behavior.

As given in the example of figure 10.2 ,

Number of training cases = 2000
Number of testing cases = 10,000

The above case is implemented in r code as follows :

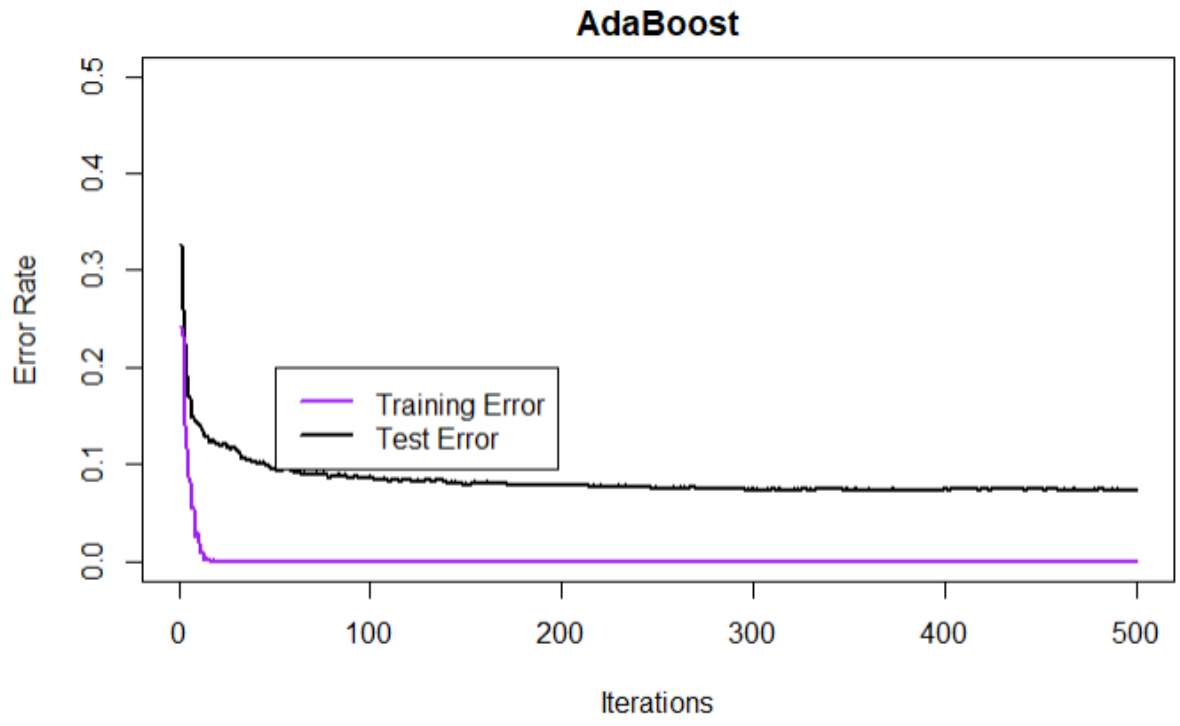
```
x=data.frame(matrix(rnorm(120000),12000,10))

y=2*as.numeric(apply(x,1,function(x){(sum(x^2)>9.34)}))-1
train.id = sample(c(1:12000),2000)

y_train<-y[train.id]
x_train<-x[train.id,]
y_test<-y[-train.id]
x_test<-x[-train.id,]
```

the results obtained are :

Confusion matrix :

```
            y_train
y_hat_train   -1    1
         -1  986    0
          1    0 1014
       y_hat_test
y_test   -1    1
    -1 4626  404
    1   323 4647
```

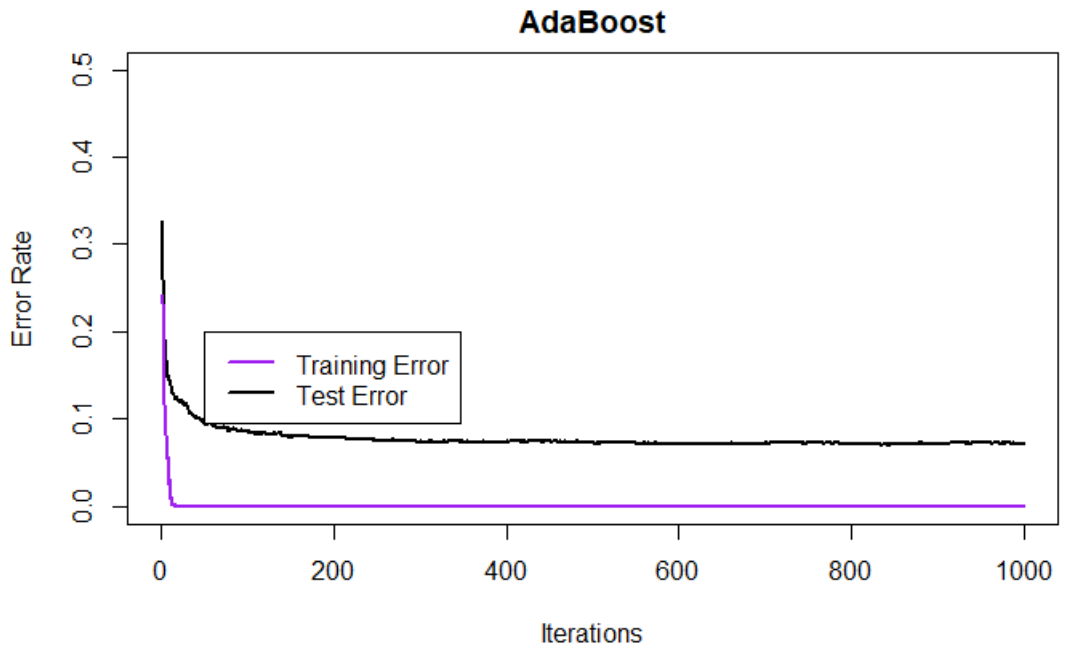
Test and train error of the adaboost after redoing the computations is:



In the above error plot , we can see that training error has become very low as the number of iterations increased. this means that the model fits closely to the data.

The test error gradually decreases as the number of iterations increase and becomes constant after 100 iterations.

(c) Investigate the number of iterations needed to make the test error finally start to rise.



The test error did not rise even after running 1000 iterations tells us that the test error never rises in the above data setting.

(d) Change the setup of this example as follows: define two classes, with the features in Class 1 being X1,X2, . . . ,X10, standard independent Gaussian variates. In Class 2, the features X1,X2, . . . ,X10 are also standard independent Gaussian, P but conditioned on the event
j X2
j > 12. Now the classes have significant overlap in feature space. Repeat the AdaBoost experiments as in Figure 10.2 and discuss the results.

As described in the question and example of figure 10.2 ,the dataset settings are:

Training set : 2000 instances
Class 1(+1) : 1000 instances with 10 features[x1…x10]
Class 2(-1) : 1000 instances with 10 features[x1…x10]

Test set : 10000 instances
Class 1(+1) : 5000 instances with 10 features[x1…x10]
Class 2(-1) : 5000 instances with 10 features[x1…x10]

The above case is implemented in r code as follows :

```
x_train_class2 = matrix(rnorm(10000),1000,10);
x_train_class1 = matrix(rnorm(35000),3500,10);
x_train_class1 =
x_train_class1[apply(x_train_class2,1,function(x){(sum(x^2)>12)}),]
y1_train
=2*as.numeric(apply(x_train_class1,1,function(x){(sum(x^2)>9.34)}))-1

x_train=rbind(x_train_class1,x_train_class2)

y2_train=sample(-1, size=1000,replace = TRUE)
y_train=c(y1_train,y2_train)
traindim=length(y_train)

x_test_class2 = matrix(rnorm(50000),5000,10)
x_test_class1 = matrix(rnorm(17500),17500,10);
x_test_class1=
x_test_class1[apply(x_test_class1,1,function(x){(sum(x^2)>12)}),]
x_test=rbind(x_test_class1,x_test_class2)


y1_test =2*as.numeric(apply(x_test_class1,1,function(x){(sum(x^2)>9.34)}))-
1


y2_test=sample(-1, size=5000,replace = TRUE)
y_test=c(y1_test,y2_test)
```

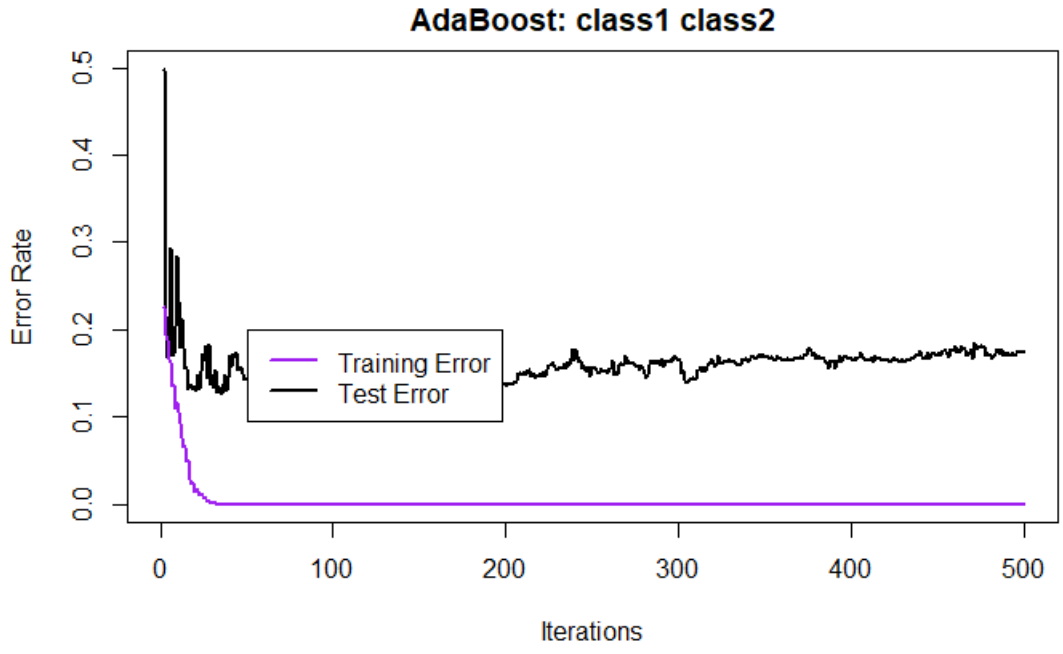
```
testdim=length(y_test)
train.index <- sample(c(1:dim(x_train)[1]))
x_train= x_train[train.index,]
y_train=y_train[train.index]
x_train=as.data.frame(x_train)
x_test=as.data.frame(x_test)
```

The result obtained are:

Confusion matrix

```
            y_train
y_hat_train   -1    1
         -1 1469    0
          1    0  447
       y_hat_test
y_test   -1    1
    -1 4139  861
    1   857 3986
```

Testing and training error:



In the above error plot , we can see that training error has become very low and constant as the number of iterations increased. this means that the model fits closely to the data.

On the other hand , the test error drops significantly after few iterations but increases for few iterations before it reaches a constant range of 0.15 to 0.2 .