Data Mining II Boosting Assignment

Allen Rahrooh

01 April 2020

Contents

Problem 10.2

Part A

Write a program implementing AdaBoost with trees.

```
library(gbm)
```

```
## Loaded gbm 2.1.5
```

```
set.seed(0)
gen_eq_10_2_data = function(N=100,p=10){
  X = matrix( rnorm( N*p ), nrow=N, ncol=p )
  Y = matrix( -1, nrow=N, ncol=1 )
  threshold = qchisq(0.5,df=p)
  indx = rowSums( X^2 ) > threshold
  Y[indx] = +1
  data.frame(X,Y)
}

p = 10 # the dimension of the feature vector
N_train = 2000 # number samples training
N_test = 10000 # number of samples testing

# Extract the data we will to classify:
D_train = gen_eq_10_2_data(N=N_train,p=p)
D_train[ D_train$Y==-1, p+1 ] = 0 # Map the response "-1" to the value of "0"

# formula used to fit our model with:

terms = paste( colnames(D_train)[1:p], collapse="+" )
formula = formula( paste( colnames(D_train)[p+1], " ~ ", terms ) )

# Do training with the maximum number of trees:

n_trees = 400
if( T ){
  print( "Running Adaboost ..." )
  m = gbm( formula, data=D_train, distribution='adaboost', n.trees=n_trees,
           shrinkage=1, verbose=TRUE )
}else{
  print( "Running Bernoulli Boosting ..." )
  m = gbm( formula, data=D_train, distribution='bernoulli',
           n.trees=n_trees, verbose=TRUE )
}
```

```
## [1] "Running Adaboost ..."
## Iter   TrainDeviance   ValidDeviance   StepSize   Improve
##      1        0.9821             nan     1.0000    0.0060
##      2        0.9650             nan     1.0000    0.0079
##      3        0.9477             nan     1.0000    0.0044
##      4        0.9283             nan     1.0000    0.0115
```

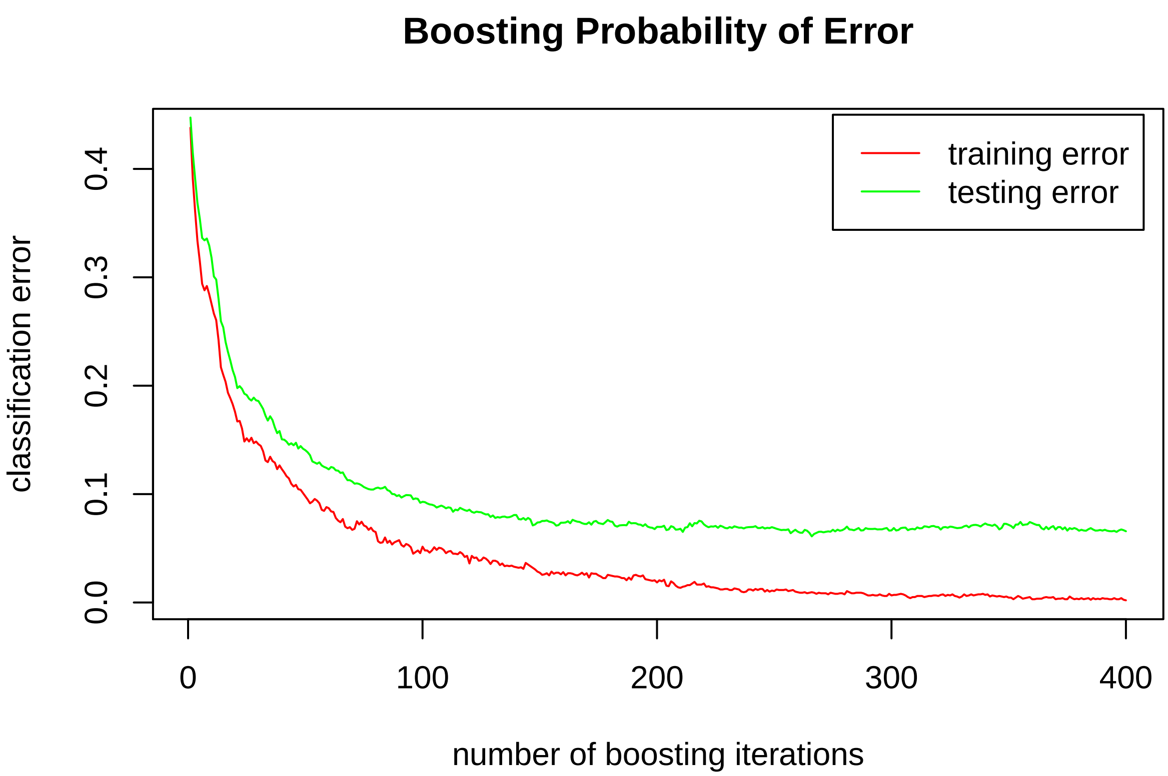
```
##      5        0.9071             nan     1.0000    0.0141
##      6        0.8846             nan     1.0000    0.0161
##      7        0.8671             nan     1.0000    0.0105
##      8        0.8476             nan     1.0000    0.0142
##      9        0.8302             nan     1.0000    0.0002
##     10        0.8135             nan     1.0000    0.0112
##     20        0.6680             nan     1.0000    0.0039
##     40        0.4943             nan     1.0000   -0.0007
##     60        0.3925             nan     1.0000    0.0015
##     80        0.3141             nan     1.0000    0.0006
##    100        0.2656             nan     1.0000   -0.0041
##    120        0.2313             nan     1.0000    0.0015
##    140        0.2026             nan     1.0000   -0.0016
##    160        0.1765             nan     1.0000   -0.0009
##    180        0.1528             nan     1.0000    0.0012
##    200        0.1364             nan     1.0000   -0.0037
##    220        0.1237             nan     1.0000    0.0009
##    240        0.1140             nan     1.0000    0.0018
##    260        0.1030             nan     1.0000   -0.0020
##    280        0.0987             nan     1.0000   -0.0023
##    300        0.0928             nan     1.0000   -0.0036
##    320        0.0832             nan     1.0000   -0.0011
##    340        0.0796             nan     1.0000   -0.0022
##    360        0.0745             nan     1.0000   -0.0008
##    380        0.0701             nan     1.0000   -0.0019
##    400        0.0666             nan     1.0000    0.0007
```

```r
# training error as a function of the number of trees:

training_error = matrix( 0, nrow=n_trees, ncol=1 )
for( nti in seq(1,n_trees) ){
  Fhat = predict( m, D_train[,1:p], n.trees=nti )
  pcc = mean( ( ( Fhat <= 0 ) & ( D_train[,p+1] == 0 ) ) |
                ( ( Fhat > 0 ) &
                         ( D_train[,p+1] == 1 ) ) )
  training_error[nti] = 1 - pcc
}

# Lets plot the testing error as a function of the number of trees:

D_test = gen_eq_10_2_data(N=N_test,p=p)
D_test[ D_test$Y==-1, p+1 ] = 0 # Map the response "-1"
#to the value of "0" (required format for the call to gbm):

test_error = matrix( 0, nrow=n_trees, ncol=1 )
for( nti in seq(1,n_trees) ){
  Fhat = predict( m, D_test[,1:p], n.trees=nti )
  pcc = mean( ( ( Fhat <= 0 ) & ( D_test[,p+1] == 0 ) ) |
                ( ( Fhat > 0 ) &
                           ( D_test[,p+1] == 1 ) ) )
  test_error[nti] = 1 - pcc
}
```

Part B

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

```
plot( seq(1,n_trees), training_error, type="l", main="Boosting Probability of Error",
      col="red", xlab="number of boosting iterations", ylab="classification error" )
lines( seq(1,n_trees), test_error, type="l", col="green" )

legend( 275, 0.45, c("training error", "testing error"),
        col=c("red", "green"), lty=c(1,1) )
```



Part C

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

Comment on this

Part D

Change the setup of this example as follows: define two classes, with the features in Class 1 being $X_1, X2, \ldots, X10$, standard independent Gaussian variates. In Class 2, the features $X1, X2, \ldots, X10$ are also standard independent Gaussian, but conditioned on the event $\sum_j X_j^2 > 12$. Now the classes have significant overlap in feature space. Repeat the AdaBoost experiments as in Figure 10.2 and discuss the results.