Homework 4

STAT6306: Due Nov. 2 at 9:30 am

Introduction

For this assignment, let's attempt to make a spam filter. Usually, this would involve a lot of text processing on a huge number of emails. In this case, someone has created a feature matrix for us. The feature matrix has rows given by individual emails and columns given by the number of each word or character that appears in that email, as well as three different numerical measures regarding capital letters (average length of consecutive capitals, longest sequence of consecutive capitals, and total number of capital letters).
The supervisor, Y, is given by the user supplied label marking that email as either spam (Y = 1) or not (Y = 0). Here is a function that may be useful for this assignment:

```
misClass =function(pred.class,true.class,produceOutput=FALSE){
  confusion.mat = table(pred.class,true.class)
  if(produceOutput){
    return(1-sum(diag(confusion.mat))/sum(confusion.mat))
  }
  else{
    print('miss-class')
    print(1-sum(diag(confusion.mat))/sum(confusion.mat))
    print('confusion mat')
    print(confusion.mat)
  }
}
# this can be called using:
#     (assuming you make the appropriately named test predictions)
# misClass(Y.hat,Y_0)
```

Read in the R data set:

```
load("spam.Rdata")
```

Let's make a training and test set.

```
train = spam$train
test  = !train
X     = spam$XdataF[train,]
X_0   = spam$XdataF[test,]
Y     = factor(spam$Y[train])
Y_0   = factor(spam$Y[test])
```

Install necessary packages

```
repos = 'http://cran.us.r-project.org'
if(!require('randomForest')){install.packages('randomForest',repos = repos);require('randomForest')}
```

```
## Loading required package: randomForest
```

```
## randomForest 4.6-12
```

```
## Type rfNews() to see new features/changes/bug fixes.
```

Question 1: Choosing the number of bagging iterations

To save computations, it is common to iteratively compute batches of random trees until the OOB error rate stabilizes. The following function implements does this as well as demonstrate some typical programming strategies:

```
checkNumberItersRF = function(ntrees = 5, tolParm = 1, maxIter = 10, verbose = 0){
  ###
  # tolParm: iterations will continue until the percent decrease
  #          is less than tolParm
  ###
  misClass_out   = list()
  totalTrees_out = list()

  n              = nrow(X)
  votes          = matrix(0,nrow=n,ncol=2)
  totalTrees     = 0
  iterations     = 0
  misClass_old   = 1
  while(iterations < maxIter){
    votes[is.nan(votes)] = 0
    iterations    = iterations + 1
    totalTrees    = totalTrees + ntrees
    if(verbose >= 2){cat('Total trees: ',totalTrees,'\n')}
    out.rf        = randomForest(X, Y,ntree = ntrees)

    oob.times        = out.rf$oob.times
    votes_iterations = out.rf$votes*oob.times
    votes[oob.times>0,] = matrix(votes + votes_iterations,nrow=n)[oob.times>0,]
    if(min(apply(votes,1,sum)) == 0){next}

    Yhat          = apply(votes,1,which.max) - 1
    misClass_new  = misClass(Yhat,Y,produceOutput = TRUE)
    misClass_out[[iterations]]   = misClass_new
    totalTrees_out[[iterations]] = totalTrees
    percentChange = 100*(misClass_new - misClass_old)/misClass_old
    if(verbose >= 1){cat('% change: ',percentChange,'\n')}
    if(percentChange > -tolParm){break}
    misClass_old = misClass_new
  }
  if(iterations == maxIter){
    stop("too many iterations, try a larger ntrees or maxIter value")
  }
  return(list('misClass' = unlist(misClass_out),
              'totalTree' = unlist(totalTrees_out)))
}
```

Comment on the roll of each of these pieces in the above function:

- next: SOLUTION:
- maxIter: SOLUTION:
- verbose: SOLUTION:
- while: SOLUTION:
- tolParm: SOLUTION:
- misClass_old: SOLUTION:

- stop: SOLUTION:

Call this function with a suitable value of maxIter and verbose so that the function produces the minimal amount of output. Report back the number of iterations you find

```
# SOLUTION
```

SOLUTION

Question 2: Random Forest classifications

What is the test misclassification rate, sensitivity, specificity, precision, recall, and confusion matrix for the random forest chosen in the previous question? How does the test misclassification rate compare with the OOB misclassification rate?

```
#SOLUTION
```

Question 3: Variable importance for random forests

Get a variable importance plot for the permuted OOB importance measure. Also, produce a proximity plot to visualize the emails. Describe both of these plots. Identify an extreme observation via the proximity plot. What is something notable about this email (I'm leaving this vague. There can be many answers to this question. Investigate!)

```
#SOLUTION
```

Question 4: Pruned classification tree

Fit an unpruned classification tree to the training data. Prune the tree via weakest-link pruning (i.e. using the cv.tree and prune.misclass pair of functions as shown in lecture). Plot the pruned tree.

```
#SOLUTION
```

Question 5: Random Forest vs. Pruned Tree

Compare the test and training misclassification rates for the unpruned tree, the CV pruned tree, and random forest (do two different random forests, one with mtry set at the default and another set at mtry = p).

```
#SOLUTION
```