Homework 5

STAT6306: Due Nov. 15 at 12:30 pm

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('misClass')
    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$XdataF[test,]
Y     = spam$Y[train]
Y_0   = spam$Y[test]
```

Install necessary packages

```
repos = 'http://cran.us.r-project.org'
packages = c('randomForest','gbm','e1071')
for(package in packages){
  if(!require(package,character.only=TRUE)){
    install.packages(package,repos = repos)
    require(package,character.only=TRUE)
```

```
  }
}
```

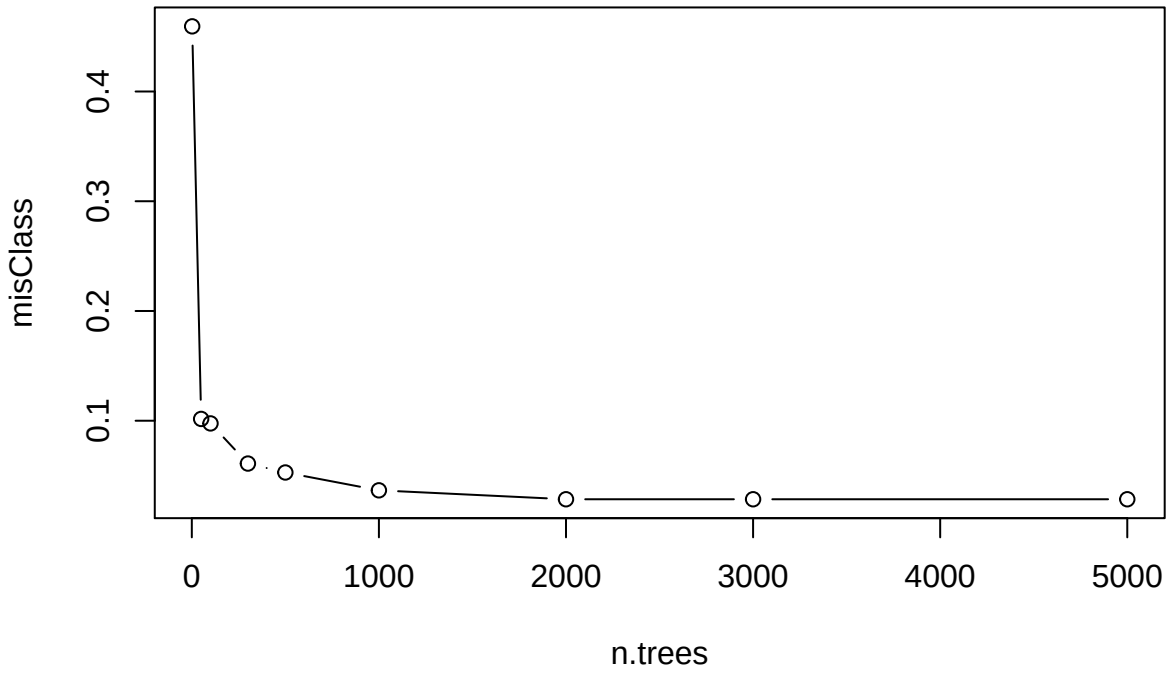
```
## Loading required package: randomForest
## randomForest 4.6-12
## Type rfNews() to see new features/changes/bug fixes.
## Loading required package: gbm
## Loading required package: survival
## Loading required package: lattice
## Loading required package: splines
## Loading required package: parallel
## Loaded gbm 2.1.3
## Loading required package: e1071
```

Question 1: Boosting with gbm

There are 3 main choices involved in boosting trees: the number of boosting iterations, the learning rate, and the tree complexity [see list at end of 8.2.3 of ISL for a brief discussion]. Using the gbm package, explore a variety of values for these parameters and note the test confusion matrices. In your write-up of this problem, I want you to just record your observations about the effects of these parameters. Report the best values. Do your conclusions change depending on if you use Adaboost versus Bernoulli loss?

```
lambdaGrid            = c(.01,.1,.25)
interaction.depthGrid = c(4,5,6)
n.treesGrid           = c(500,1000,2000)
distribution          = 'bernoulli'
set.seed(1)
verbose = 0
for(lambda in lambdaGrid){
  for(interaction.depth in interaction.depthGrid){
    for(n.trees in n.treesGrid){
      boost.out = gbm(Y~.,data=X,
                      n.trees=n.trees, interaction.depth=interaction.depth,
                      shrinkage=lambda, distribution = distribution)
## SOLUTION
      prob.hat  = predict(boost.out,X_0,n.trees=n.trees,type='response')
      Y.hat = rep(0,nrow(X_0))
      Y.hat[prob.hat > 0.5] = 1
      Y.hat = as.factor(Y.hat)
      if(verbose > 0){
      cat('lambda = ',lambda,' interaction.depth = ',interaction.depth, ' lambda = ',lambda,' n.trees =
      }
      if(verbose > 1){
        misClass(Y.hat,Y_0)
      }
    }
  }
}
```

```r
#Here, we saw in the above output that good values would be:
interaction.depth = 5
shrinkage         = 0.01
#Let's look at a grid of B's at these values:
n.treesGrid = c(2,50,100,300,500,1000,2000,3000,5000)
set.seed(1)
boost.out = gbm(Y~.,data = X, n.trees = max(n.treesGrid),
                  interaction.depth = interaction.depth,
                  shrinkage = shrinkage,
                  distribution = distribution)

getClassesF = function(n.trees,features,supervisor){
  prob.hat  = predict(boost.out,features,n.trees=n.trees,type='response')
  Y.hat = rep(0,nrow(features))
  Y.hat[prob.hat > 0.5] = 1
  return(misClass(Y.hat,supervisor,produceOutput=TRUE))
}
results = sapply(n.treesGrid,getClassesF,features=X_0,supervisor=Y_0)

plot(n.treesGrid,results,xlab='n.trees',ylab='misClass',type='b')
```

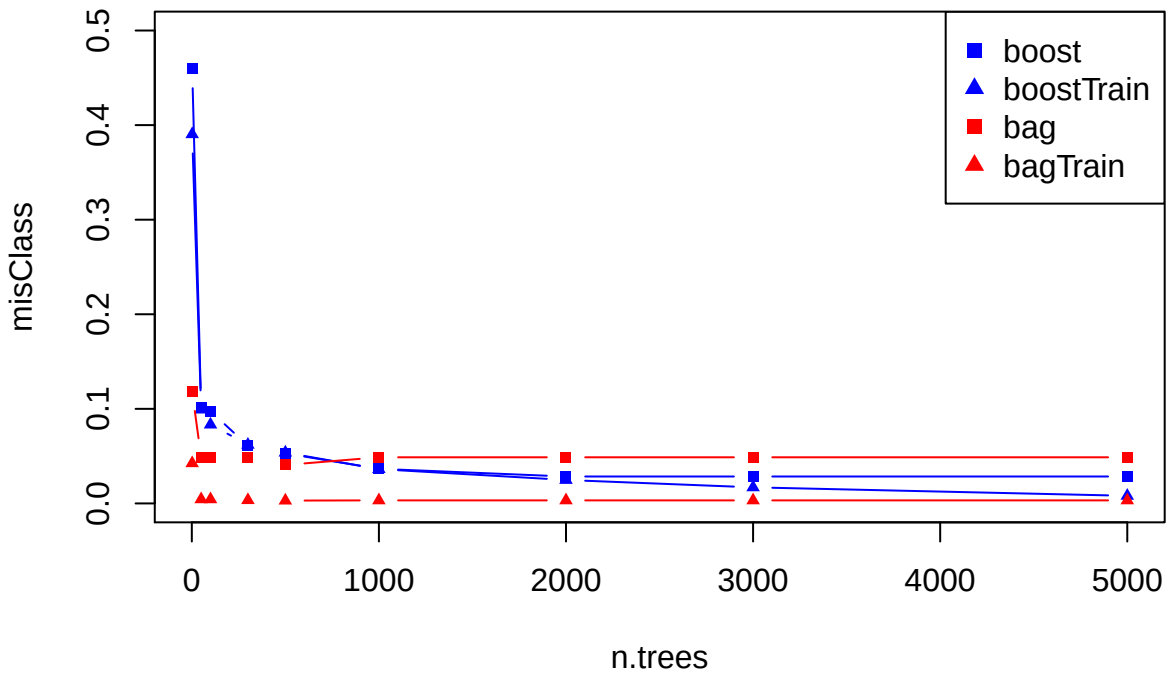


Question 2: Boosting vs. Random Forest

Make a plot of the training misclassifications and test misclassifications for a grid of B values from 3 up to 5000 (it doesn't have to be a dense grid, just choose a representative set of values) for both bagging and boosting (so, one plot with 4 curves). Which method gets the lowest training error? The lowest test error?

```r
#boosting training misclassification rates:
resultsTrain = sapply(n.treesGrid,getClassesF,features=X,supervisor=Y)
```

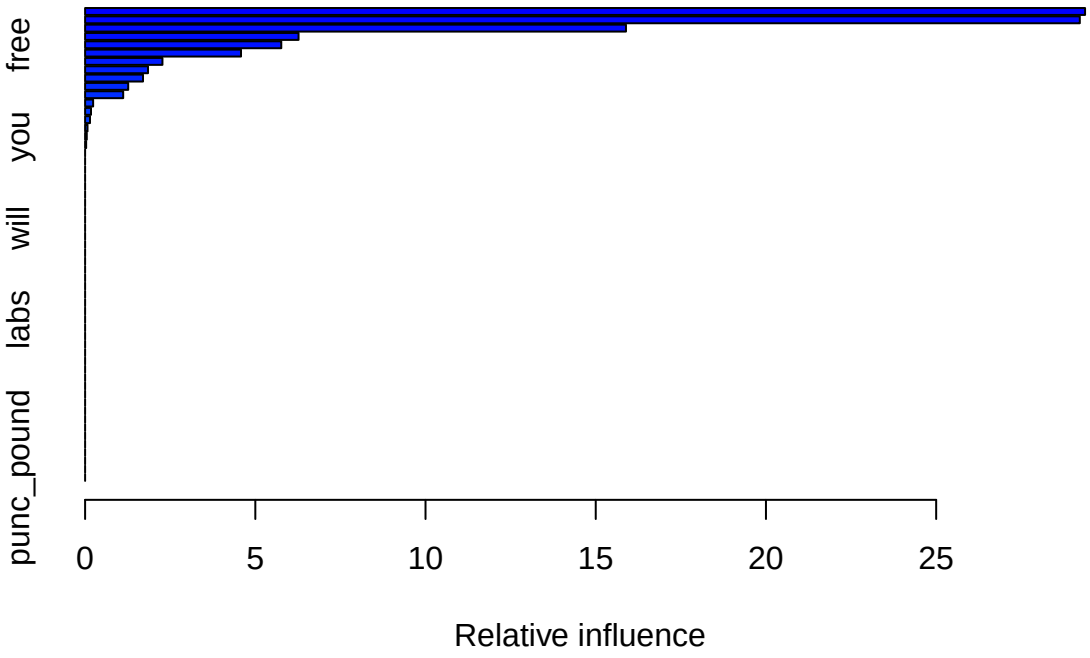
```
require(randomForest)
set.seed(1)
out.rf        = randomForest(X,as.factor(Y),ntree=max(n.treesGrid),
                             xtest=X_0,ytest=as.factor(Y_0))
resultsRF     = out.rf$test$err.rate
out.rf        = randomForest(X,as.factor(Y),ntree=max(n.treesGrid),
                             xtest=X,ytest=as.factor(Y))
resultsRFtrain = out.rf$test$err.rate

#training/testing results for boosting/rf
plot(n.treesGrid,results,xlab='n.trees',ylab='misClass',type='b',pch=15,col='blue',ylim=c(0,.5),cex=.7)
lines(n.treesGrid,resultsTrain,type='b',pch=17,col='blue',cex=.7)
lines(n.treesGrid,resultsRF[n.treesGrid,1],type='b',pch=15,col='red',cex=.7)
lines(n.treesGrid,resultsRFtrain[n.treesGrid,1],type='b',pch=17,col='red',cex=.7)
legend(x='topright',legend=c('boost','boostTrain','bag','bagTrain'),col=c('blue','blue','red','red'),pch
```



Question 3: Boosting importance

Just like in bagging, we can get an idea about variable importance. For each tree in the boosting ensemble, and for each feature, we record the amount the training misclassification rate is reduced by making a split on the j^{th} feature. We take the average of this over all B trees to get the importance. We can have R give us this information from running summary on the output of gbm. What are the 3 most important variables?

```
##SOLUTION
head(summary(boost.out,n.trees=100))
```




```
##                     var   rel.inf
## punc_dollar punc_dollar 29.373775
## punc_exclam punc_exclam 29.219598
## remove           remove 15.887930
## hp                   hp  6.269898
## free               free  5.762105
## capAvg           capAvg  4.579293
```

Question 4: SVMs

Let's look at SVMs for fitting the spam data. Find the (test) confusion matrix for an SVM with nonlinear kernel. Choose the cost parameter and the kernel parameters with cross-validation.

```
## 1
Y = as.factor(Y)
df  = data.frame(X=X, Y=Y)
out = tune(svm,Y~.,data=df,kernel="linear",type='C',
           ranges=list(cost=10^seq(-3,0)))
linear.cv = out$best.performance
out.linear = out$best.model
out = tune(svm,Y~.,data=df,kernel="poly",type='C',
           ranges=list(cost=10^seq(-1,3),degree=c(3,5,10)))
poly.cv = out$best.performance
out.poly = out$best.model
out = tune(svm,Y~.,data=df,kernel="radial",type='C',
           ranges=list(cost=10^seq(-1,3),gamma=c(.5,1,1.5)))
rbf.cv = out$best.performance
out.rbf = out$best.model

testDf = data.frame(X=X_0,Y=Y_0)
Yhat.linear = predict(out.linear,newdata=testDf)
Yhat.poly   = predict(out.poly,testDf)
```

```
Yhat.rbf    = predict(out.rbf,testDf)
print('Linear')
```

```
## [1] "Linear"
```

```
cMat = misClass(Yhat.linear,Y_0)
```

```
## [1] "misClass"
## [1] 0.04471545
## [1] "confusion mat"
##           true.class
## pred.class   0   1
##          0 129   7
##          1   4 106
```

```
cat('Cv est: ',linear.cv)
```

```
## Cv est:  0.07186492
```

```
cMat[1,1]/sum(cMat[,1])
```

```
## [1] 0.9699248
```

```
cMat[2,2]/sum(cMat[,2])
```

```
## [1] 0.9380531
```

```
print('Poly')
```

```
## [1] "Poly"
```

```
cMat = misClass(Yhat.poly,Y_0)
```

```
## [1] "misClass"
## [1] 0.06910569
## [1] "confusion mat"
##           true.class
## pred.class   0   1
##          0 129  13
##          1   4 100
```

```
cat('Cv est: ',poly.cv)
```

```
## Cv est:  0.07645998
```

```
cMat[1,1]/sum(cMat[,1])
```

```
## [1] 0.9699248
```

```
cMat[2,2]/sum(cMat[,2])
```

```
## [1] 0.8849558
```

```
print('RBF')
```

```
## [1] "RBF"
```

```
cMat = misClass(Yhat.rbf,Y_0)
```

```
## [1] "misClass"
## [1] 0.199187
## [1] "confusion mat"
```

```
##            true.class
## pred.class   0   1
##          0 133  49
##          1   0  64
```

```
cat('Cv est: ',rbf.cv)
```

```
## Cv est:  0.1692212
```

```
cMat[1,1]/sum(cMat[,1])
```

```
## [1] 1
```

```
cMat[2,2]/sum(cMat[,2])
```

```
## [1] 0.5663717
```