STAA 577 HW4

Enqun Wang

April 13, 2016

Due: April 18 at noon

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, X, for us. X 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 response, 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:

```
miss.class = 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)
    return(1 - sum(diag(confusion.mat))/sum(confusion.mat))
  }
}
# this can be called using: (assuming you make the
# appropriately named test predictions)
# miss.class(Y.hat, Y_0)
```

1. Read in the R data set `spam.Rdata` and read the documentation file `spambase.Documentation`. What object is loaded into memory? What objects are inside that object? How many emails do we have total? What features are in this data set?

```
load("/Users/ewenwang/Dropbox/CSU-MAS/STAA 577/Homework/spam.rdata")
attributes(spam)
```

```
## $names
## [1] "train"            "Xmat"             "XdataF"
## [4] "Y"                "covariate_labels" "column_labels"
```

- A R data file is loaded into memory; it includes 6 lists, which are shown above. The dataset contains 4601 eamils with 58 features.

2. 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])
```

How many observations are in the training set (that is, what is n)? How many observations are in the test set?

```
length(train)
```

```
## [1] 4601
```

```
length(test)
```

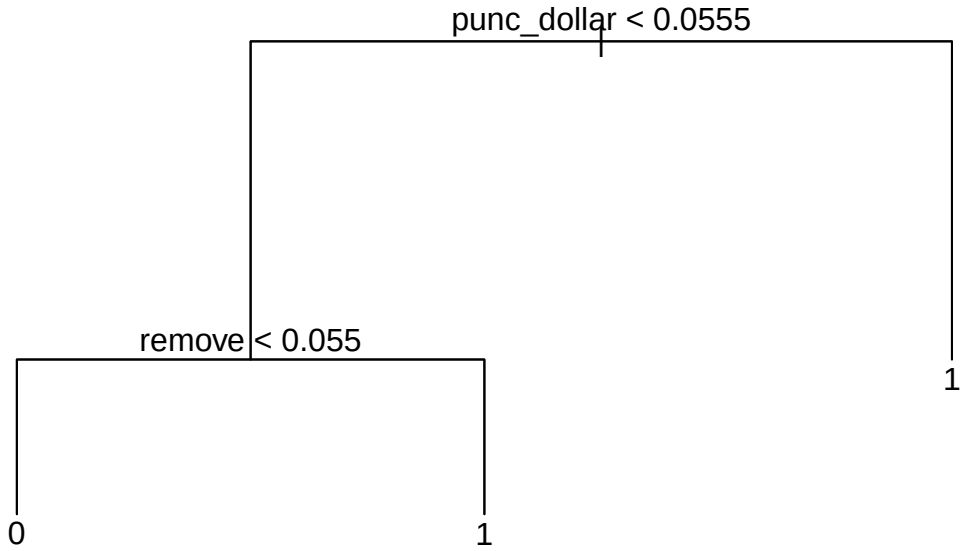
```
## [1] 4601
```

- There are 4601 observations are in the training set; and 4601 observations in the test set.

3. Run the following code

```
require(tree)

set.seed(2016)
out.tree = tree(Y~.,data=X)
tmp.tree = prune.tree(out.tree, best=3)
plot(tmp.tree)
text(tmp.tree)
```



What feature is split on first? Interpet that split point. Make the corresponding partition view for this dendrogram (you don't need to use R for this, just draw the right rectangles and be neat about it).

- The tree split on `punc_dollar` firts. If percentage of words in the e-mail that match `dollar` is greater 5.55%, then it is a spam. When `punc_dollar` is less than 0.0555, if `remove` is greater than 0.0555, it is a spam; otherwise, it is a non-spam.

4. Fit an unpruned classification tree to the training data (hint: you've already done that on this h/w). Get the associated test misclassification rate and test confusion matrix.

```
class.tree = predict(out.tree, X_0, type='class')
misRate = miss.class(class.tree, Y_0)
```

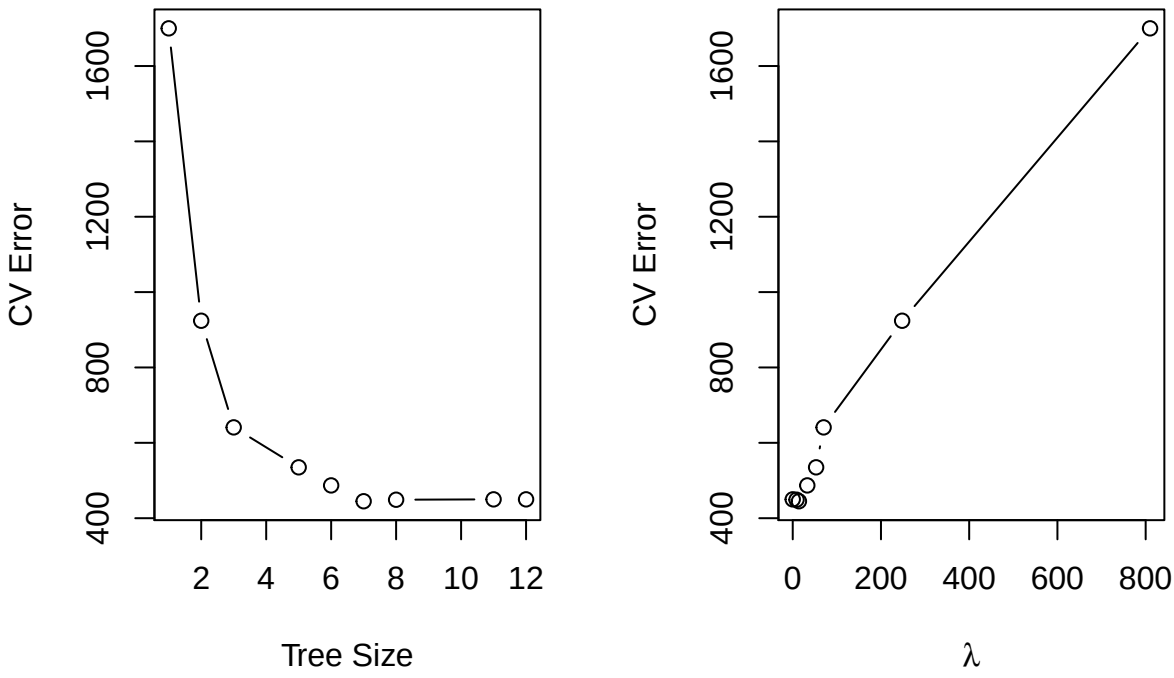
```
## [1] "miss-class"
## [1] 0.07317073
## [1] "confusion mat"
##           true.class
## pred.class   0   1
##          0 127  12
##          1   6 101
```

- The test misclassification rate is 0.0731707.

5. Prune the tree via weakest-link pruning (i.e. using the cv.tree and prune.misclass pair of functions as shown in lecture). What are this tree's test misclassification rate and test confusion matrix?

```
out.tree.cv = cv.tree(out.tree, FUN = prune.misclass)

par(mfrow = c(1, 2))
plot(out.tree.cv$size, out.tree.cv$dev, type="b",
     xlab = "Tree Size", ylab = "CV Error")
plot(out.tree.cv$k, out.tree.cv$dev, type="b",
     xlab = expression(lambda), ylab = "CV Error")
```




```
par(mfrow = c(1, 1))

best.size = out.tree.cv$size[which.min(out.tree.cv$dev)]

out.tree = prune.misclass(out.tree, best=best.size)
class.tree = predict(out.tree, X_0, type='class')

misRate <- miss.class(class.tree, Y_0)
```

```
## [1] "miss-class"
## [1] 0.09349593
## [1] "confusion mat"
##            true.class
## pred.class   0   1
##          0 127  17
##          1   6  96
```

- The test misclassification rate is 0.0934959 and test confusion matrices is shown above.

6. Form a classifier with `randomForest` using the default `mtry`. What are the test misclassification rate and test confusion matrices? How does the test misclassification rate compare with the OOB misclassification rate?

```r
require(randomForest)

set.seed(2016)
out.rf = randomForest(X, Y, importance=T)
class.rf = predict(out.rf, X_0)

misRate <- miss.class(class.rf, Y_0)
```

```
## [1] "miss-class"
## [1] 0.04878049
## [1] "confusion mat"
##           true.class
## pred.class   0   1
##          0 132  11
##          1   1 102
```

```r
out.rf
```

```
##
## Call:
##  randomForest(x = X, y = Y, importance = T)
##                Type of random forest: classification
##                      Number of trees: 500
## No. of variables tried at each split: 7
##
##         OOB estimate of  error rate: 4.55%
## Confusion matrix:
##      0    1 class.error
## 0 2577   78  0.02937853
## 1  120 1580  0.07058824
```

- The test misclassification rate is 0.0487805 and test confusion matrices is shown above. The OOB misclassification rate is 4.55%, which is less than the test misclassification rate.