Homework #8: Tree Ensembles

Ben Wilson

Due: Wed Nov 2 | 11:45am

Contents

DS 6030 | Fall 2022 | University of Virginia

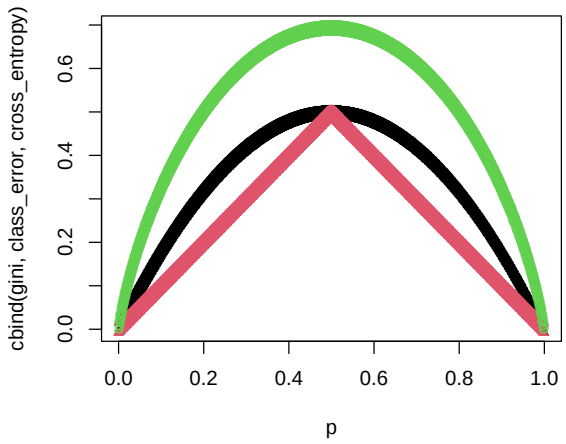
This is an **independent assignment**. Do not discuss or work with classmates.

Required R packages and Directories

```
data.dir = 'https://mdporter.github.io/DS6030/data/' # data directory
library(R6030)      # functions for DS-6030
library(tidyverse) # functions for data manipulation
library(randomForest)
library(caTools)
```

Problem 1: Tree Splitting for classification

```
p <- seq(0, 1, 0.001)
gini <- 2 * p * (1 - p)
class_error <- 1 - pmax(p, 1 - p)
cross_entropy <- - (p * log(p) + (1 - p) * log(1 - p))
matplot(p, cbind(gini, class_error, cross_entropy))
```



Problem 2: Combining bootstrap estimates

a. ISLR 8.2 describes the *majority vote* approach for making a hard classification from a set of bagged classifiers. What is the final classification for this example using majority voting?

```
red_obs = sum(p_red > 0.5)
green_obs = sum(p_red < 0.5)

red_obs
```

```
#> [1] 4
```

```
green_obs
```

```
#> [1] 6
```

Using a threshold of 0.5 for selecting the classes between red and green (where > 0.5 is red and < 0.5 is green), there are 6 observations < 0.5 signifying green and 4 observations > 0.5 signifying red.

Given that green is the most commonly occurring class (4 red observations, 6 green observations), X can be classified as green using the majority vote approach.

b. An alternative is to base the final classification on the average probability. What is the final classification for this example using average probability?

```
mean(p_red)
```

```
#> [1] 0.535
```

Given that the average probability of p_red is 0.535, exceeding our threshold of 0.5, the classification is red.

c. Suppose the cost of mis-classifying a Red observation (as Green) is twice as costly as mis-classifying a Green observation (as Red). How would you modify both approaches to make better final classifications under these unequal costs? Report the final classifications.

```
#turn prob into list
p_red_list = list(p_red)

#create df with list
df_pred <- as.data.frame(p_red_list)
names(df_pred)[1] <- "p_red"

#misclass cost
misclass_red = 2
misclass_green = 1

#enter misclass cost as column
df_pred$misclass = ifelse(df_pred$p_red > 0.5, misclass_green, misclass_red)

#calc new prob with misclass
df_pred$misclass_pred = df_pred$p_red * df_pred$misclass
```

Majority vote with misclassification cost

```
red_obs_mis = sum(df_pred$misclass_pred > 0.49)
green_obs_mis = sum(df_pred$misclass_pred < 0.49)

red_obs_mis
```

```
#> [1] 9
```

```
green_obs_mis
```

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

Given that red is the most commonly occurring class (9 red observations, 1 green observation), X can be classified as red using the majority vote approach with misclassification cost included.

Average probability with misclassification cost

```
mean(df_pred$misclass_pred)
```

```
#> [1] 0.735
```

Given the higher cost if we misclassify, the average probability has naturally increased and is still identifying the classification as red given it is exceeding the threshold of 0.5.

Problem 3: Random Forest Tuning

```
#load mass package with boston data
library(MASS)
```

```
#>
#> Attaching package: 'MASS'

#> The following object is masked from 'package:dplyr':
#>
#>     select
```

```
df_boston = Boston

#view data
head(df_boston)
```

```
#>      crim zn indus chas   nox    rm  age   dis rad tax ptratio black lstat medv
#> 1 0.00632 18  2.31    0 0.538 6.575 65.2 4.090   1 296    15.3 396.9  4.98 24.0
#> 2 0.02731  0  7.07    0 0.469 6.421 78.9 4.967   2 242    17.8 396.9  9.14 21.6
#> 3 0.02729  0  7.07    0 0.469 7.185 61.1 4.967   2 242    17.8 392.8  4.03 34.7
#> 4 0.03237  0  2.18    0 0.458 6.998 45.8 6.062   3 222    18.7 394.6  2.94 33.4
#> 5 0.06905  0  2.18    0 0.458 7.147 54.2 6.062   3 222    18.7 396.9  5.33 36.2
#> 6 0.02985  0  2.18    0 0.458 6.430 58.7 6.062   3 222    18.7 394.1  5.21 28.7
```

a. List all of the random forest tuning parameters in the `randomForest::randomForest()` function. Note any tuning parameters that are specific to classification or regression problems. Indicate the tuning parameters you think will be most important to optimize?

mt ry - # of predictors evaluated for each split or sampled at each node (if use a smaller m, higher bias)

type - indicates whether regression, classification, or unsupervised model

cv.folds - indicates number of cross validation folds to run

ntree - # of trees grown

oob.times - # of times cases are out of bag

Classification parameters exclusively - err.rate, confusion, and votes

Regression parameters exclusively - mse, rsq

b. Use a random forest model to predict `medv`, the median value of owner-occupied homes (in $1000s). Use the default parameters and report the 10-fold cross-validation MSE.

```
#train test split for cross-val
spl <- sample.split(Boston$medv, SplitRatio = 0.7)
train <- subset(Boston, spl == TRUE)
test <- subset(Boston, spl == FALSE)

#run random forest
rf_boston = randomForest(medv ~., data=train, mtry=ncol(Boston)-1, importance=TRUE, cv.folds = 10)
rf_boston
```

```
#>
#> Call:
#>  randomForest(formula = medv ~ ., data = train, mtry = ncol(Boston) -      1, importance = TRUE, cv.
#>                Type of random forest: regression
#>                      Number of trees: 500
#> No. of variables tried at each split: 13
#>
#>           Mean of squared residuals: 14.77
#>                     % Var explained: 83.52
```

c. Now we will vary the tuning parameters of `mtry` and `ntree` to see what effect they have on performance.

```
#set seed for reproducibility
set.seed(100)

#new train test split x values
train <- sample(1:nrow(Boston), nrow(Boston) / 2)
b_train <- Boston[train, -14]
b_test <- Boston[-train, -14]
```

```r
#train test split y values
y_train <- Boston[train, 14]
y_test <- Boston[-train, 14]

#four random forests for mapping
rf_b1 <- randomForest(b_train, y = y_train, xtest = b_test, ytest = y_test,
                      mtry = ncol(Boston) - 1, ntree = 500)
rf_b2 <- randomForest(b_train, y = y_train, xtest = b_test, ytest = y_test,
                      mtry = (ncol(Boston) - 1) / 2, ntree = 500)
rf_b3 <- randomForest(b_train, y = y_train, xtest = b_test, ytest = y_test,
                      mtry = sqrt(ncol(Boston) - 1), ntree = 500)
rf_b4 <- randomForest(b_train, y = y_train, xtest = b_test, ytest = y_test,
                      mtry = (ncol(Boston)^(1/3) - 1), ntree = 500)

#plot mse results
plot(1:500, rf_b1$test$mse, col = "darkorchid", type = "l",
     xlab = "# Trees", ylab = "MSE")
lines(1:500, rf_b2$test$mse, col = "darkblue", type = "l")
lines(1:500, rf_b3$test$mse, col = "darkgreen", type = "l")
lines(1:500, rf_b4$test$mse, col = "firebrick", type = "l")
legend("topright", c("m = p", "m = p/2", "m = p^1/2", "m = p^1/3"),
       col = c("green", "red", "blue", "orange"), cex = 1, lty = 1)
```