Lab 13: Tree-Based Methods

Stat 154, Spring 2018

Introduction

In this lab, we will explore various tree-based methods, namely decision trees, random forests, and boosted trees. This lab follows ISL 8.3: Lab: Decision Trees closely. The dataset we are using in this lab is `Carseats` from the `ISLR` package.

```
library(ISLR)
attach(Carseats)
High <- ifelse(Sales <= 8, "No", "Yes")
carseats <- data.frame(Carseats, High)
```

Decision Trees

We will use the library `tree` to fit a decision tree. The syntax for `tree()` is analogous to `lm()`: `response ~ predictor` for the `formula` argument.

```
library(tree)
tree_carseats <- tree(High ~ .-Sales , data=carseats)
```

Your turn

- Run `summary(tree_carseats)` and describe the output.
- Run `plot(tree_carseats)` and `text(tree_carseats, pretty=0)` and describe the output.
- Display `tree_carseats` and describe the output.

Random Forests

Random forests are considered one of the best "off-the-shelf" classifiers with minimal tuning. The idea is to build many weakly correlated trees (and hence a *forest*) via bagging and random variable selections (and hence *random*). Then the prediction is done via a majority vote. We will use the library `randomForest` to fit a random forest.

Remark: Random forests are *embarrassingly parallel*, meaning that the fitting can be easily separated into a number of parallel tasks. Packages such as `ranger` and `ParallelForest` provide an easy-to-use implementation of paralleled random forests, allowing efficient computations.

Your turn

- Randomly select 80% of the observations as the training set and the other 20% as the test set.
- Using the training set, train a random forest with `High` as the response and all other variables except `Sales` as predictors. Make sure you set `importance=TRUE`.
- Compute the test error rate. How does it compare to the out-of-bag (OOB) error rate?
- Use `importance()` to view the importance of each variable. Create a visualization via `varImpPlot()`.
- Which two predictors are the most important variables?

Boosted Trees

To improve the performance of decision trees, boosting can be used. The idea of boosting is to iteratively fit a small tree to the residuals from the current model as an attempt to improve the model performance on areas where the current model does not do well. There are three tuning parameters for boosted trees: the number of trees B, the shrinkage parameter λ, and the interaction depth d. In this lab we will explore the impact of adjusting B and d on the classification performance. We will use the package `gbm` to fit boosted trees.

Your turn

- Using the same train-test split as before, compute the test error rate for boosted trees. Train the boosted trees with $B = 5000$ trees. Use 0.5 as the cutoff for the predicted probabilities.
- Run `summary()` for the trained boosted trees.
- Based on the output from `summary()`, which are the two most importance variables?
- Note that when using `predict()`, we can specify the number of trees used via `n.trees`. Compute the test error rate with $B \in \{10, 20, 30, ..., 4990, 5000\}$. Plot the test error rate against the number of trees B.
- By default, the *interaction depth* is set to be 1. Redo the last part for $d = 2, 3$, and 4. Do you observe any qualitiative differences among the test error curves?