Statistical Learning, Tutorato #6

Veronica Vinciotti, Marco Chierici

April 19, 2021

Exercise 1

Consider once again the `Boston` dataset (`MASS` library), which was introduced in Tutorato 5. Our objective is to predict `medv` from the other variables through random forests. To this aim:

a. Split the data into training/test partitions.
b. Apply a random forest model on the training set using `mtry=6` and `ntree=25`.
c. Consider now a more comprehensive range of values for `mtry` and `ntree`: use this range to create a plot displaying the test error resulting from random forests on these data. You can model your plot after Figure 8.10 in the textbook.
d. Describe the results obtained and draw a conclusion on the optimal model to use. Use the `importance()` function to determine which variables are most important.

Hints:

- Random forests are implemented in the library `randomForest` (function `randomForest()`)
- The function can accept the usual formula `fit <- randomForest(outcome ~ ., data=dataf, mtry, ntree)`, as well as the syntax `fit <- randomForest(x, y, xtest, ytest, mtry, ntree)`, where `x` and `xtest` contain only the explanatory variables. When the latter is used, the output will contain an additional slot (`fit$test`) with the test set predictions (i.e., `fit$test$predicted`) as well as test set metrics (i.e., `fit$test$mse`) for each tree.
- Use the option `importance=TRUE` when fitting a random forest to assess the importance of predictors.

Exercise 2

Meet the `Carseats` data set, containing simulated observations of sales of child car seats at 400 different stores. It is part of the `ISLR` library.

```
head(Carseats)
```

```
##   Sales CompPrice Income Advertising Population Price ShelveLoc Age Education
## 1  9.50       138     73          11        276   120       Bad  42        17
## 2 11.22       111     48          16        260    83      Good  65        10
## 3 10.06       113     35          10        269    80    Medium  59        12
## 4  7.40       117    100           4        466    97    Medium  55        14
## 5  4.15       141     64           3        340   128       Bad  38        13
## 6 10.81       124    113          13        501    72       Bad  78        16
##   Urban  US
## 1   Yes Yes
## 2   Yes Yes
## 3   Yes Yes
## 4   Yes Yes
## 5   Yes  No
## 6    No Yes
```

The aim is to predict the quantitative variable `Sales` from the other variables using regression trees and related approaches.

a. Split the data set into a training set and a test set.
b. Fit a regression tree by recursive binary splitting to the training set. Plot the tree, and interpret the results. What test MSE do you obtain?
c. Use cross-validation in order to determine the optimal level of tree complexity for pruning. Produce a plot of the cross-validation deviance as a function of tree size. What is the optimal size? If you prune the tree according to this optimal size, does the test MSE improve?
d. Use the bagging approach in order to analyze this data. What test MSE do you obtain? Use the `importance()` function to determine which variables are most important and comment on what you obtain.
e. Use random forests to analyze this data. What test MSE do you obtain? Describe the effect of m (the number of variables considered at each split) on the error rate obtained. Use the `importance()` function to determine which variables are most important.

Hints:

- Regression trees are implemented in the function `tree()`, library `tree`. This function will fit a full regression tree via recursive binary splitting, with default tuning parameters set in `?tree.control` (`minsize` could be reduced further if we want to grow an even deeper tree);
- Use the usual formula syntax to fit a regression tree; `predict()` can be used to obtain predictions;
- The function `cv.tree()` on a tree object performs 10-fold cross-validation for choosing tree complexity;
- Pruning can be performed by using `prune.tree()`;

Exercise 3

For this exercise, we explore boosting on a simulated dataset. In order to simulate the data, run the following code:

```
set.seed(78)
sim <- mlbench::mlbench.friedman1(400, sd = 1)
sim <- cbind(sim$x, sim$y)
sim <- as.data.frame(sim)
colnames(sim)[ncol(sim)] <- "y"
```

Consider now the data set `sim`, containing simulated data with a response variable `y` and 10 explanatory variables `V1`, `V2`, ..., `V10`. Our aim is to use boosting to predict `y`.

a. Create a training/test partition, splitting the data into two halves.
b. Perform boosting on the training set with 1,000 trees for a range of values of the shrinkage parameter λ. Plot the different shrinkage values on the x-axis and the corresponding training set MSE on the y-axis.
c. Produce a similar plot as the previous one, this time using the test set MSE. Comment on what you observe from comparing these two plots.
d. Which variables appear to be the most important predictors in the boosted model? Now read the documentation on the **mlbench.friedman1** function that was used to simulated the data. Are the selected variables those that you would expect to?

Hints:

- Use `gbm()` (library `gbm`) to perform boosting, with `distribution="gaussian"` for regression tasks.