Department of Statistics
Lund University

Data Mining
STAN 45

Project 3: Fitting decision trees, boosting, random forest

Perform all requested task. If your work will not be cleared by the instructor during the lab time, please, follow these instructions. On a separate paper give answers to the questions and comment on the performed analysis, attach a printout of the code you have used and all graphs that you deem important for this lab session. You will pass the lab based on this material. Some useful R-code that can help in completing Project 3 can be found **here**.

Part One – Regression by a binary tree

We consider Boston dataset that comes with `'MASS'` package (which has to be installed first). Here is short information about the dataset.

The data set deals with housing values in suburbs of Boston. It has the format of a data frame that has 506 rows and 14 columns. This data frame contains the following columns:

`'crim'` per capita crime rate by town.

`'zn'` proportion of residential land zoned for lots over 25,000 sq.ft.

`'indus'` proportion of non-retail business acres per town.

`'chas'` Charles River dummy variable (= 1 if tract bounds river; 0 otherwise).

`'nox'` nitrogen oxides concentration (parts per 10 million).

`'rm'` average number of rooms per dwelling.

`'age'` proportion of owner-occupied units built prior to 1940.

`'dis'` weighted mean of distances to five Boston employment centres.

`'rad'` index of accessibility to radial highways.

`'tax'` full-value property-tax rate per $10,000.

`'ptratio'` pupil-teacher ratio by town.

`'black'` $1000(Bk - 0.63)^2$ where Bk is the proportion of blacks by town.

`'lstat'` lower status of the population (percent).

`'medv'` median value of owner-occupied homes in $1000s.

1. Activate `Boston` data file. Divide the data on training and testing part and fit the regression tree to the training data set. How many partition sets the tree provided? Are all variables used in the construction of the tree? Plot the tree.

2. Perform the cross-validation study. Use `help(cv.tree)` to see more details about the function. What is the size of the tree that is recommended by the cross-validation? Is it different from the size originally proposed by the tree algorithm?

3. How accurate in the terms of standard deviation is the prediction from the actual values in the test sample? Would you consider this an accurate predictor?

4. Try to build the tree based on a partition of the size 5. Is it performing worse than the one based on the optimal size? Comment on your findings.

Part Two – Boosting

Here we use the `gbm` package, and within it the `gbm()` function, to fit boosted regression trees to the Boston data set. Perform the following tasks by running suitable parts of the code.

1. Run `gbm()` with the option `distribution="gaussian"` since this is a regression problem (if it were a binary classification problem, we would use `distribution="bernoulli"`). The argument `n.trees=5000` indicates that we want 5000 trees, and the option `interaction.depth=4` limits the depth of each tree.

2. Run `summary()` function. Can you identify results produced by its output?

3. Which variables are the most important variables?

4. Produce *partial dependence plots* for these variables. These plots illustrate the marginal effect of the selected variables on the response after integrating out the other variables.

5. Comment on the observed dependence of the median house prices on the most important variables.

6. Use the boosted model to predict the median house prices for on the test set. Evaluate the prediction mean square error. How does it compare with the accuracy of the prediction obtained for the regression tree built in the previous part of the lab.

7. Try to run the boost algorithm for another value of the shrinkage parameter λ (the default value is $\lambda = 0.001$). Do you observe any significant change in the prediction power of the resulting boosted model.

Part Three – Bagging and Random Forests

Here we apply bagging and random forests to the Boston data (which was also previously used in Lab 3), using the `randomForest` R-package. Recall that bagging is simply a special case of a random forest with $m = p$, i.e. at each split all points originally selected to a bootstrap sample are available. Therefore, `randomForest()` function can be used to perform both random forests and bagging. We need also to download `randomForest` package from an R repository, see the code for details.

1. Execute command `help(randomForest)`. By reading the printout find out which parameter is controlling the number of variables to be used at each node for the optimal selection (in the lecture denoted by m). Whato value does one have to set to m to get bagged version of the random forest?

2. What kind of importances types for the input variables one can obtain in the algorithm? How one can build importance values for variables based on the out of bag (oob) samples?

3. First we create a training sample on which we will build a regression tree for the median value of the occupied house.

4. Apply the bagging to the data. How many trees have been generated and what was the mean square error of the residuals to the median price?

5. What is the percentage of variability explained by the model? What does it mathematically mean? (Compute the mean square of the median values and compare with the mean square of residuals.)

6. Investigate the importance of variables using the implemented functions: `importance` and `varImpPlot`. Which variables seem to be the most important?

7. Evaluate the obtained procedure using the testing data. Plot the graph illustrating how well the predictor works vs. the actual data. Comment how this graph looks comparing with an analogous one obtained for a regular regression tree in the previous lab.

8. Evaluate the test set mean square error (MSE) associated with the bagged regression tree. Compare it with the previous MSE for the testing sample obtained using using an optimally-pruned single tree, boosted model. Which of the methods seems to be most effective for the data at hand?

9. Compare also the test MSE with the one obtained in the training phase. Which one is smaller and is it expected?

10. Perform the same analysis but with a much reduced number of bagged trees. Are the results much worse?

11. Growing a random forest proceeds in exactly the same way, except that one uses a smaller value of the m argument. By default, `randomForest()` uses $p/3$ variables when building a random forest of regression trees, and $\sqrt{p}$ variables when building a random forest of classification trees. One can also force m of ones choice. Fit random forests to the data. Compare with the previous results. Do you observe a significant improvement?

12. Remove three the least important variables and redo the analysis using random tree (consider changing m). Report your finding and compare with the previous ones. For this part the code is not provided.

13. Leave three most important variables and redo analysis. What are conclusions now?