Random Forest

Ky Phong Mai

Overview

Random Forest is a common method used in Machine Learning because of its versatility and high predictive power. At its core, it is built on the concept of decision tree (will be explained in later section) but with some clever scheme to obtain much more robust predictions.

Data type

Random Forest can work for both regression and classification problem. Or in other words, it can be used to predict numerical values or predict which class an observation belongs to. This means that for predictor and response variables, they can take both numerical data type (e.g. height of students or weight of diamonds) and categorical data type (e.g. cut of diamonds: fair, premium, ideal..). This makes random forest a very versatile machine learning model.

How it works

Though the full technical and statistical aspects of random forest is hard to grasp, understanding the idea behind is not that difficult. As the name suggests, random forest comes from two main ideas: **forest** and **randomness**.

Firstly, it is considered as a **forest because it consists of multiple trees** where each can make prediction on its own. These are called decision trees. Decision tree is another great statistical learning method that can be used to predict either a numerical value or a class that an observation falls into. Basically, it performs a series of splitting rules depending on whether it is a regression or classification problem. **Figure 1** shows an example of a decision tree (regression) for predicting the *log salary* with 2 predictors *years* and *hits*. When number of years played is less than 4.5, the predicted *log salary* is 5.11. When number of years is greater than 4.5, the prediction now depends on number of hits: 6 when number of hits is less than 117.5 and 6.74 otherwise. How these splitting thresholds are chosen is rather technical and is not a requirement for domain experts to know before they want to use it for prediction. There is one question: decision tree seems like a good method but why would we want to combine them into forest?

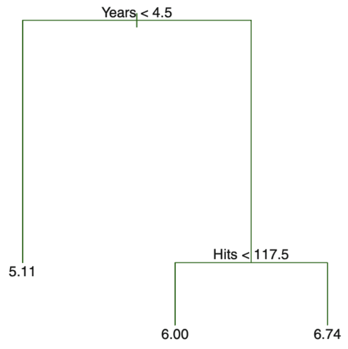

Figure 1: Example of a decision tree for predicting log salary from 2 predictors

Even though decision tree can be interpreted and explained easily for domain experts, it does have one critical drawback which is its non-robustness. To explain simply, during training where the model attempts to learn the structure behind the data, decision tree will learn well, if not too well, on these training data. This will negatively affect its ability to generalize on new data. A good analogy for this is a student who learns by heart all the material in the text book and can only answer questions in the exam if they are covered exactly in the text book but fail to score on questions that are different to a certain degree from the learned material. A solution for this is combining many trees together so that hopefully the average of those will give us a better result. As we are limited by the amount of training data, we can cleverly create different sets of training data based on the original data but the same observations can be repeated in one set (this concept is called **bootstrapping**). Allowing each decision tree to learn from each newly created set of training data and combining to get the final result is called **bagging**. This is analogous to giving a whole class of students to learn by heart from the same textbook with each student learning different versions of the book (some parts repeated more often than others). The concerted effort from all students will likely produce a better result than any student alone. This is the reason why we need forest instead of single decision tree.

We already achieve good results by putting decision trees together, yet we have not really seen randomness appearing in our method (besides the bootstrapping part). As all the decision trees learn from the same data, if there are very strong predictors, the split of trees will be dominated by these predictors. This will result in similar tree structure and hence similar prediction. Instead of allowing the tree to use all predictors, the model randomly selects a small subset of predictors at each split. This forces the tree to learn different aspects of data rather than focusing on the dominating variables. This is the additional **random component of the random forest**. Continuing with the analogy previously, but with random forest method, a student might learn from only chapter 1 and 2 from the textbook even though the dominating theme of the textbook is in chapter 4 and 5. If that student does an exam independently, he/she will probably fail it. However, if each student focus on random parts of the book, the whole class of student will be able to discuss and give much more comprehensive answers as compared to when all students answers the questions based on mostly the main chapters.

After having each decision trees, for regression problem, the final result will be the average results of all the trees. Similarly, for classification, the final predicted class will be the class agreed by most of the decision trees.

Tuning

There are 2 hyperparameters we can tune when specifying the random forest model.

```
# Example of how to specify tuning parameters
rand_forest(mtry = tune(),
            min_n = tune(),
            trees = 500)
```

The first hyperparameter is ***mtry***. This hyperparameter specifies the number of variables randomly taken into consideration at each split. As mentioned earlier, each at split, random forest will only select a few predictors to minimize the dominance of some strong predictors. By default, ***mtry*** is chosen to be $\lfloor p/3 \rfloor$ for regression and $\lfloor \sqrt{p} \rfloor$ for classification.

The second hyperparameter is ***min_n***, which is the minimum number of observations in each region required to be split further. For example, if ***min_n*** is 3 and the number of observations in the current region is 2, it will not be split anymore. By default, ***min_n*** is set as 1 for both classification and regression.

However, note that in random forest, we can also specify the number of decision trees but this will not be used as a tuning parameter. The reason is that the more trees we have, the better the model will perform. Therefore, it is redundant to tune.

Advantages and Disadvantages

Random Forest is great technique used for prediction because of the following advantages:

1. It can be used to predict both numerical and categorical variables
2. Its predictive accuracy is usually quite high
3. It is able to generalize on unseen data as compared to decision tree
4. It works well when data is non-linear
5. It is able to handle categorical predictors without creating dummy variable

However, random forest has some drawbacks that domain experts need to be aware of before they decide to use it:

1. Because of its random component, its output might vary to a certain extent on the same training data. One way to make it replicable is setting a seed right before training the model.
2. Tuning and training random forest requires more time and resources as compared with other machine learning methods.
3. It loses its interpretability of a decision tree because the final output of a random forest combines results from all decision trees