Improve the Decision Tree

August 10, 2020

In this document, we discuss the bagging and random forest approaches to improve the prediction performance. Both approaches use trees as the basic module.

1 Bagging

One major problem with trees is their high variance. If we partition the training dataset into two equal-sized small sets, we may obtain two trees which are totally different. Here, we introduce a common approach, named bagging, to reduce the high variance. Usually, this approach is particularly useful for the decision tree.

Recall that given a set of n independent observations $Z_1, Z_2, ..., Z_n$, each with variance σ^2, the variance of the mean $\bar{Z}$ of the observation is given by $\frac{\sigma^2}{n}$. In other words, averaging a set of observations reduces variance. Hence a natural way to reduce the variance and hence increase the prediction accuracy of a machine learning method is to take many training sets from the population, build a separate prediction model using each training set, and average the resulting predictions. In other word, we could calculate $\hat{f}^1(x), \hat{f}^2(x), ..., \hat{f}^B(x)$ using B separate training sets, and average them in order to obtain a single low-variance learning model, given by

$$\hat{f}_{avg}(x) = \frac{1}{B}\sum_{b=1}^{B} \hat{f}^b(x). \tag{1}$$

Of course, this is not practical because we generally do not have access to multiple training sets. Instead, we can bootstrap, by taking repeated samples from the (single) training dataset. In this approach, we generate B different bootstrapped training datasets. We then train our method on the bth bootstrapped training set in order to get $\hat{f}^{*b}(x)$, and finally average all predictions, to obtain

$$\hat{f}_{bag}(x) = \frac{1}{B}\sum_{b=1}^{B} \hat{f}^{*b}(x). \tag{2}$$

This is called bagging.

While bagging can improve predictions for many regression methods, it is particularly useful for decision trees. To apply bagging to regression trees, we simply construct B regression trees using B bootstrapped training sets, and average the resulting predictions. These trees are grown deep, and are not pruned. Hence each individual tree has high variance, but low bias. Averaging these B trees reduces the variance. Bagging has been demonstrated to give impressive improvements in accuracy by combining together hundreds or even thousands of trees into a single procedure.

Thus far, we have described the bagging procedure in the regression context, to predict a quantitative outcome Y. How can bagging be extended to a classification problem where Y is qualitative? In that situation, there are a few possible approaches, but the simplest is as follows. For a given test observation, we can record the class predicted by each of the B trees, and take a *majority vote*: the overall prediction is the most commonly occurring majority class among the B predictions.

1.1 Variable Importance Measures

As we have discussed, bagging typically results in improved accuracy over prediction using a single tree. Unfortunately, however, it can be difficult to interpret the resulting model. Recall that one of the advantages of decision trees is the attractive and easily interpreted diagram that results. However, when we bag a large number of trees, it is no longer possible to represent the resulting learning procedure using a single tree, and it is no longer clear which variables are most important to the procedure. Thus, bagging improves prediction accuracy at the expense of interpretability.

Although the collection of bagged trees is much more difficult to interpret than a single tree, one can obtain an overall summary of the importance of each attribute using the RSS (for bagging regression trees) or the Entropy index (for bagging classification trees). In the case of bagging regression trees, we can record the total amount that the RSS is decreased due to splits over a given attribute, averaged over all B trees. A large value indicates an important attribute. Similarly, in the context of bagging classification trees, we can add up the total amount that the entropy is decreased by splits over a given attribute, averaged over all B trees.

2 Random Forest

Random forests provide an improvement over bagged trees by way of a small tweak that *de-correlates the trees.* As in bagging, we build a number of decision trees on bootstrapped training samples. But when building these decision trees, each time a split in a tree is considered, a random sample of m attributes is chosen as split candidates from the full set of p attributes. The split is allowed to use only one of those m attributes. A fresh sample of m attributes is taken at each split, and typically we choose $m = \lceil \sqrt{p} \rceil$—that is, the number of attributes considered at each split is approximately equal to the square root of the total number of attributes.

In other words, in building a random forest, at each split in the tree, the algorithm is not even allowed to consider a majority of the available attributes. This may sound crazy, but it has a clever rationale. Suppose that there is one very strong attribute in the data set, along with a number of other moderately strong attributes. Then in the collection of bagged trees, most or all of the trees will use this strong attribute in the top split. Consequently, all of the bagged trees will look quite similar to each other. Hence the predictions from the bagged trees will be highly correlated. Unfortunately, averaging many highly correlated quantities does not lead to as large of a reduction in variance as averaging many uncorrelated quantities. In particular, this means that bagging will not lead to a substantial reduction in variance over a single tree in this setting.

Random forests overcome this problem by forcing each split to consider only a subset of the attributes. Therefore, on average $(p - m)/p$ of the splits will not even consider the strong attribute, and so other attributes will have more of a chance. We can think of this process as de-correlating the trees, thereby making the average of the resulting trees less variable and hence more reliable.

The main difference between bagging and random forests is the choice of attribute subset size m. For instance, if a random forest is built using $m = p$, then this amounts simply to bagging. Using a small value of m in building a random forest will typically be helpful when we have a large number of correlated attributes.