Professional Certificate in Machine Learning and Artificial Intelligence

Module 10: Decision Trees, Part 2

Learning outcomes:

- Select the most appropriate tree depth for making a prediction.
- Identify important elements of tree ensembles.
- Implement important elements of tree ensembles.

Avoiding overfitting of trees

We use pre-pruning and post-pruning to avoid the overfitting of decision trees. However, there is an alternative, a more powerful approach to avoid overfitting. Instead of constructing a single tree, you will construct a whole forest of trees called a **tree ensemble**.

Each individual tree in such a tree ensemble differs from the other trees by perturbing the training data.

- Use bootstrapping to make your trees more robust against noise
- For new data points, classify or predict the output variable of that point through either majority vote in the case of classification trees or by averaging in the case of a regression tree

One of the disadvantages of using tree ensembles is the loss of interpretability.

Bagging

If you split your training data randomly into two halves and make decision trees classifying each half, the trees will look very different from one another. This indicates a high presence of variance in the estimator. You can reduce this variance by using the bagging algorithm, which is a combination of bootstrapping and averaging.

Tree construction

For each decision tree $b = 1, \ldots, B$ (e.g.: $B = 100$)

1. Sample n training data points with replacement
2. Grow a complete tree from these samples

Prediction

1. Predict outcome of new data point in each tree $b = 1, \ldots, B$.
2. Take a majority vote (classification) or average (regression).

Disadvantage of the bagging algorithm

The multiple trees in bagging are highly correlated as they are essentially based on the same dataset. Averaging over these trees can only help up to a certain degree.

Decorrelating trees

Classification and regression trees derived from bagging are highly correlated. This is undesirable as it does not allow you to reduce the variance as much as you would want to. A workaround here is to decorrelate the trees. You can do this using two popular classes of algorithms:

1. Random forests
2. Boosting

Random forests

In the random forest algorithm, you decorrelate the decision trees by forcing them to look at different subsets of predictors at each level.

1. Consider a collection of decision trees (e.g., 100)
2. For each of the 100 trees, sample n of your training data points with replacement
3. Grow a complete tree, but at each splitting step, consider a random sample of the predictors

4. Decorrelate these decision trees and take a majority vote or an averaged outcome

Boosting

In boosting, the key idea here is to attach weights to each training sample. The higher the weight, the more important that training sample.

1. At first, set the weights of all training data points to be the same.
2. Then, grow your first decision tree, typically a shallow one, only to depth one or two. Use decision tree pre-pruning here.
3. Next, look at the data points and decide which ones have been misclassified. Assign a higher weight to misclassified ones and a lower weight to those that have been classified correctly
4. Now, grow the second decision tree. This, too, will be a shallow one, but with a new set of weights that will put more emphasis on correctly classifying those samples that were previously misclassified.

Often, the contribution of each tree and the boosting algorithm is outweighed by the misclassification rate of that tree. Hence, it is necessary to give more importance to those trees.