Chapter 6: Decision Trees

Warren Alphonso

March 2019

1 Making Predictions

Decision trees are a versatile ML algorithm that can be used for classification and regression. They can be represented using trees with each branch of a node making a decision about the characteristics of an item. Each node contains certain information:

- A node's `samples` attribute counts how many training instances it applies to.
- A node's `value` attribute counts how many training instances of each class this node applies to.
- A node's `gini` attribute measures its *impurity*: a node is "pure" (`gini=0`) if all training instances it applies to belong to the same class.

$$G_i = 1 - \sum_{k=1}^{n} p_{i,k}^2$$

Figure 1: Gini impurity function where $p_{i,k}$ is the ratio of class k instances among the training instances in the i^{th} node.

2 Estimating Class Probabilities

A Decision Tree can also estimate the probability that an instance belongs to a particular class k. To do so, it traverses the tree to find the leaf node for this instance, then it returns the ratio of training instances of class k in this node.

2.1 The CART Training Algorithm

Scikit-Learn uses the *Classification And Regression Tree* (CART) algorithm to train Decision Trees. The algorithm first splits the training set in two subsets using a single feature k and a threshhold t_k. It chooses by searching for the pair, (k, t_k) that produces the purest subsets (weighted by size). Once it has

successfully split the training set in two, it splits the subsets using the same logic, and continues recursively.
The CART algorithm is a *greedy algorithm* because it searches for an optimum split at the top level, then repeats the process at each level. It does not check whether or not the split will lead to the lowest possible impurity several levels down, so it is not guaranteed to be the optimal solution. Unfortunately, finding the optimal tree is known to be an *NP-Complete* problem.

2.2 Computational Complexity

Decision Trees are generally balanced, so traversing the Decision Tree requires going through roughly $O(log_2(m))$ nodes. Predictions are very fast. However, the training algorithm compares all features which results in complexity of $O(n \text{ x } m \ log(m))$.

2.3 Regularization Hyperparameters

Decision Trees make very few assumptions about the training data (as opposed to linear models which assume the data is linear). This means the tree structure will adapt itself to the training data, most likely overfitting it. This model is called *nonparametric* because the number of parameters is not determined prior to training. To avoid overfitting, we need to restrict the Decision Tree's freedom during training (this is called regularization). Generally, we at least restrict the maximum depth of the Decision Tree.
The `DecisionTreeClassifier` class has a few other parameters that can restrict the shape of the Decision Tree:

- `min_samples_split`: the minimum number of samples a node must have before it can be split
- `min_samples_leaf`: the minimum number of samples a leaf node must have
- `max_leaf_nodes`: the maximum number of leaf nodes
- `max_features`: the maximum number of features that are evaluated for splitting at each node

3 Regression

Decision Trees are also capable of performing regression tasks. The main difference is that instead of predicting a class in each node, it predicts a value. We traverse the tree starting at the root and eventually reach the leaf node that predicts the value. This prediction is simply the average target value of the training instances associated to this leaf node. The CART algorithm works mostly the same way as earlier, except instead of trying to split the training set

in a way that minimizes impurity, it now tries to split the training set in a way that minimizes the MSE.

4 Instability

Decision Trees have a few limitations. First, Decision Trees love orthogonal decision boundaries which makes them sensitive to training set rotation. One way to limit this is to use PCA. Second, and more importantly, Decision Trees are very sensitive to small variations in the training data. They even vary after running the same algorithm on the same dataset because the training algorithm used by Scikit-Learn is stochastic. As we will see, Random Forests can limit this instability by averaging predictions over many trees.