Decision Tree

Sean Wade

OVERVIEW

Decision trees are a predictive model that creates a graph for decision making. To make a prediction one must only follow the graph to a leaf node with the appropriate class.

1. Implement the Decision Tree

My implementation of the decision tree uses the ID3 splitting criteria. Each split in the tree is calculated by choosing the highest info gain (measured with entropy). The structure is a class that consists of a tree of decisionNodes. This tree is built during train time and predicting is simply moving down the tree by checking the values.

2. Using the Tree

The first tests with my decision tree were on the voting and cars datasets. For the following tables I ran a 10-fold cross validation and stored the train and test accuracies. We note that the test accuracy for both is either perfect or really high. This highlights the sensitivity of this model to becoming overfit. The reason for this is without stopping criteria the model will go until it is only one class or nothing. This makes it so learning is simply memorizing the training data.

TRAIN	TEST
0.895954	1.0
0.919075	1.0
0.907514	1.0
0.947977	1.0
0.895954	1.0
0.953757	1.0
0.942197	1.0
0.936416	1.0
0.953488	1.0
0.924419	1.0

TRAIN	TEST
95.7447	99.7585
97.8261	99.7590
95.6522	1.00000
97.8261	99.7590
91.3043	99.7590
91.3043	99.7590
95.6522	1.00000
95.6522	99.7590
91.3043	99.7590
97.8261	99.7590

3. What has it Learned?

One of the benefits of decision trees is their interpretability. By looking at where and when we split we can see feature importance and a human understandable map of the decision process. When we look at the voting dataset we see the first split was on the physician-fee-freeze. That means that just knowing that we would get the most gain. So much in fact that the second layer nodes were around 80% pure aft er this. The subsequent layers split on the features synfuels-corporation-cutback and adoption-of-the-bu dget-resolution. This would give some incremental increase in purity, but not as much as the first layer.

The cars dataset went 6 layers deep. The first split was on safety. This split alone was completely able t o split one class to a pure leaf node. The other nodes then proceeded to splitting on persons. By the thi rd layer the purity was about 75% for all the leaf nodes.

4. Handling Unknown Variables

To make my model robust to missing data I trained the tree such that missing data was a category that the feature could take. The value to this approach is that there could be a significant reason why the data is missing. This would make it so that reason is codified and accounted for within the model. Also no data is thrown out and we have more to train on.

5. Preventing Overfitting

Decision trees are commonly implemented with various ways to prevent overfitting. One of the best approached is a method called pruning the tree. For this we take the full tree and drop off nodes. If dropping a node makes it perform better on the validation set we "prune" it from the tree. To add this to my code I simply took a validation set out of my training data and added a prune method to the class. This would recursively go through the tree and save the best results.

WITH PRUNING

Train Acc	Test Acc	Depth	Num Nodes
0.975845	0.978723	10.0	35.0
0.987952	0.956522	10.0	53.0
0.985542	0.956522	17.0	75.0
0.987952	0.913043	17.0	71.0
0.990361	0.913043	11.0	61.0
0.992771	0.956522	17.0	75.0
0.990361	0.934783	11.0	47.0
0.987952	0.978261	17.0	75.0
0.983133	0.956522	17.0	63.0
0.983133	0.956522	17.0	55.0

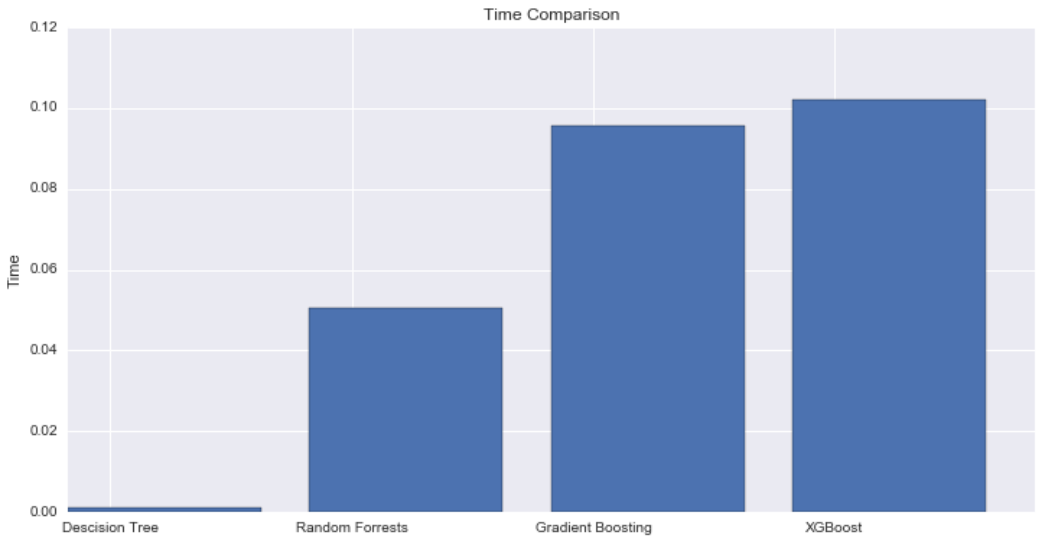
WITHOUT PRUNING

Train Acc	Test Acc	Depth	Num Nodes
0.980676	0.957447	10.0	45.0
0.997590	0.956522	10.0	53.0
0.987952	0.934783	17.0	75.0
0.987952	0.913043	17.0	71.0
0.990361	0.891304	11.0	61.0
0.992771	0.956522	17.0	75.0
0.987952	0.913043	17.0	63.0
0.987952	0.978261	17.0	75.0
0.985542	0.934783	17.0	65.0
0.983133	0.956522	17.0	57.0

From these tables we can see the benefits of pruning. There were times accuracy improved, even when the depth was cut by 5 and the number of nodes was cut by 15. The train accuracy could sometimes be a little lower, but the test is usually significantly better. This indicates we have helped the overfitting problem.

6. Extending Decision Trees

For my extra project I decided to look at the extensions of decision trees. A simple improvement to this algorithm is called random forests. In random forests we create many independent decision trees and then ensemble them together to vote for the class. Another improvement is gradient boosting where we use special criteria when we create these trees by optimizing over a cost function. The final method I want to explore is extreme gradient boosting. This is very similar to regular gradient boosting but has some improvements. These are a better regularization term in the cost function and incorporating the decision function directly into creating the trees.



These show that by extending the decision tree we can get 2% improvement on the voting and cars datasets. As nice as this is we do sacrifice training time. While this is an issue, these algorithms are highly parallelizable and can be scaled very well. But one of my favorite benefits of decision trees is how easy they are to understand. Below is the decision tree for cars. Now given any data point I can trace the tree by hand and classify it.