Decision Trees in Machine Learning

Prashant Gupta Follow
May 17, 2017

A tree has many analogies in real life, and turns out that it has influenced a wide area of **machine learning**, covering both **classification and regression**. In decision analysis, a decision tree can be used to visually and explicitly represent decisions and decision making. As the name goes, it uses a tree-like model of decisions. Though a commonly used tool in data mining for deriving a strategy to reach a particular goal, its also widely used in machine learning, which will be the main focus of this article.

How can an algorithm be represented as a tree?

For this let's consider a very basic example that uses titanic data set for predicting whether a passenger will survive or not. Below model uses 3 features/attributes/columns from the data set, namely sex, age and sibsp (number of spouses or children along).

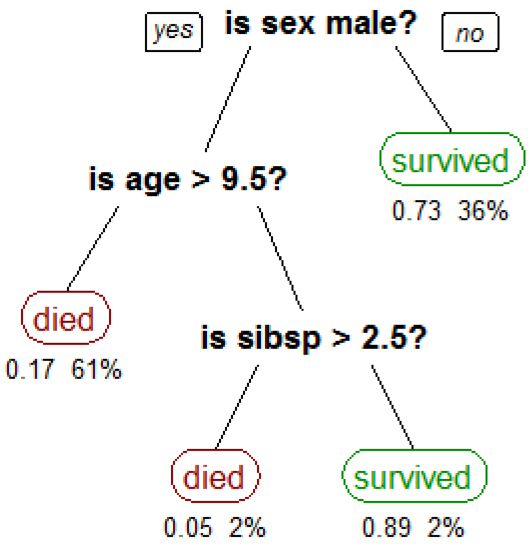

Image taken from wikipedia

A decision tree is drawn upside down with its root at the top. In the image on the left, the bold text in black represents a condition/**internal node**, based on which the tree splits into branches/ **edges**. The end of the branch that doesn't split anymore is the decision/**leaf**, in this case, whether the passenger died or survived, represented as red and green text respectively.

Although, a real dataset will have a lot more features and this will just be a branch in a much bigger tree, but you can't ignore the simplicity of this algorithm. The **feature importance is clear** and relations can be viewed easily. This methodology is more commonly known as **learning decision tree from data** and above tree is called **Classification tree** as the target is to classify passenger as survived or died. **Regression trees**are represented in the same manner, just they predict continuous values like price of a house. In general, Decision Tree algorithms are referred to as CART or Classification and Regression Trees.

So, what is actually going on in the background? Growing a tree involves deciding on **which features to choose** and **what conditions to use** for splitting, along with knowing when to stop. As a tree generally grows arbitrarily, **you will need to trim it down** for it to look beautiful. Lets start with a common technique used for splitting.

Recursive Binary Splitting

In this procedure all the features are considered and different split points are tried and tested using a cost function. The split with the best cost (or lowest cost) is selected.

Consider the earlier example of tree learned from titanic dataset. In the first split or the root, all attributes/features are considered and the training data is divided into groups based on this split. We have 3 features, so will have 3 candidate splits. Now we will ***calculate how much accuracy each split will cost us, using a function***. ***The split that costs least is chosen***, which in our example is sex of the passenger. This ***algorithm is recursive in nature*** as the groups formed can be sub-divided using same strategy. Due to this procedure, this algorithm is also known as the **greedy algorithm**, as we have an excessive desire of lowering the cost. **This makes the root node as best predictor/classifier.**

Cost of a split

Lets take a closer look at **cost functions used for classification and regression**. In both cases the cost functions try to **find most homogeneous branches, or branches having groups with similar responses**. This makes sense we can be more sure that a test data input will follow a certain path.

> *Regression : sum(y—prediction)²*

Lets say, we are predicting the price of houses. Now the decision tree will start splitting by considering each feature in training data. The mean of responses of the training data inputs of particular group is considered as prediction for that group. The above function is applied to all data points and cost is calculated for all candidate splits. *Again the split with lowest cost is chosen*. Another cost function involves reduction of standard deviation, more about it can be found here.

> *Classification : G = sum(pk * (1—pk))*

A Gini score gives an idea of how good a split is by how mixed the response classes are in the groups created by the split. Here, pk is proportion of same class inputs present in a particular group. A perfect class purity occurs when a group contains all inputs from the same class, in which case pk is either 1 or 0 and G = 0, where as a node having a 50–50 split of classes in a group has the worst purity, so for a binary classification it will have pk = 0.5 and G = 0.5.

When to stop splitting?

You might ask ***when to stop growing a tree?*** As a problem usually has a large set of features, it results in large number of split, which in turn gives a huge tree. Such trees are *complex and can lead* *to overfitting*. So, we need to know when to stop? One way of doing this is to **set a minimum number of training inputs to use on each leaf.** For example we can use a minimum of 10 passengers to reach a decision(died or survived), and ignore any leaf that takes less than 10 passengers. Another way is to set **maximum depth** of your model. **Maximum depth refers to the the length of the longest path from a root to a leaf.**

Pruning

The performance of a tree can be further increased by ***pruning***. *It involves* ***removing the branches that make use of features having low importance***. This way, we reduce the complexity of tree, and thus increasing its predictive power by reducing overfitting.

Pruning can start at either root or the leaves. The simplest method of pruning starts at leaves and removes each node with most popular class in that leaf, this change is kept if it doesn't deteriorate accuracy. Its also called **reduced error pruning**. More sophisticated pruning methods can be used such as **cost complexity pruning** where a learning parameter (alpha) is used to weigh whether nodes can be removed based on the size of the sub-tree. This is also known as **weakest link pruning.**

Advantages of CART

Simple to understand, interpret, visualize.

Decision trees *implicitly perform variable screening or feature selection.*

Can *handle both numerical and categorical data*. Can also *handle multi-output problems.*

Decision trees require relatively *little effort from users for data preparation.*

Nonlinear relationships between parameters do not affect tree performance.

Disadvantages of CART

Decision-tree learners *can create over-complex trees* that do not generalize the data well. This is called *overfitting*.

Decision trees can be unstable because *small variations in the data might result in a completely different tree being*

> *generated.* This is called *variance*, which needs to be *lowered by methods like bagging and* ***boosting***.
>
> Greedy algorithms cannot guarantee to return the globally optimal decision tree. This can be mitigated by training multiple trees, where the features and samples are randomly sampled with replacement.
>
> Decision tree learners create *biased trees if some classes dominate.* It is therefore recommended to balance the data set prior to fitting with the decision tree.

This is all the basic, to get you at par with decision tree learning. An improvement over decision tree learning is made using technique of**boosting**. A popular library for implementing these algorithms is **Scikit-Learn**. It has a wonderful api that can get your model up an running with **just a few lines of code in python**.

If you liked this article, be sure to click ♥ below to recommend it and if you have any questions, **leave a comment** and I will do my best to answer.

For being more aware of the world of machine learning, **follow me**. It's the best way to find out when I write more articles like this.

You can also follow me on **Twitter**, **email me directly** or **find me on linkedin**. I'd love to hear from you.

That's all folks, Have a nice day :)