January 26th, 2005

1 Introduction to Classification Trees:

Classification Trees serve to limit the class of classifiers. They are not the most accurate in this regard, but have two important properties:

1. They can be built efficiently
2. They are easy to interpret (also known as transparency)
    - Easy compared to implementing neural networks, which are not as intuitive.

Classification trees are easy to interpret because the representation provides a lot of intuition into what is going on. This gives confidence to the user that the classifications indeed produce the correct result.

2 What is a Classification Tree?

A classification tree is a model with a tree-like structure. It contains nodes and edges. There are two types of nodes:

- Intermediate nodes - An intermediate node is labeled by a single attribute, and the edges extending from the intermediate node are predicates on that attribute
- Leaf nodes - A leaf node is labeled by the class label which contains the values for the prediction.

The attributes that appear near the top of the tree typically indicate they are more important in making the classifications.

A classification tree example

Suppose we have information about drivers in a database. (which we can consider to be the training data for making the classification tree)

Table 1.1: Example Training Database

Car Type	Driver Age	Children	Lives in Suburb?
sedan	23	0	yes
sports	31	1	no
sedan	36	1	no
truck	25	2	no
sports	30	0	no
sedan	36	0	no
sedan	25	0	yes
truck	36	1	no
sedan	30	2	yes
sedan	31	1	yes
sports	25	0	no
sedan	45	1	yes
sports	23	2	no
truck	45	0	yes

Figure 1. Example Training Database

Here, Car Type, Driver Age, and Children are discrete attributes and Lives in Suburb is the prediction attribute. We are trying to find a mapping between the discrete attributes to the prediction attribute.

A possible classification tree is shown below:

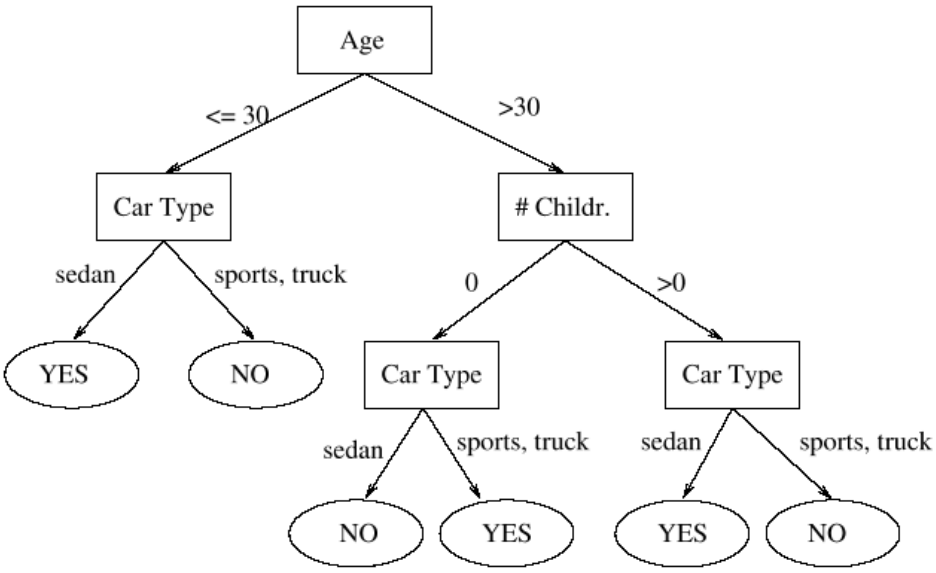

Figure 2. Example of a Classification Tree for the training data in Figure 1.

The resulting predictions can be made using this classification tree:

CarType	Age	of Children		Lives in Suburb?
Sedan	23	0		YES
Sports Car	16	0	Predicts →	NO
Sedan	35	1	Predicts →	YES

Classification trees are piece-wise predictors. Suppose we map each of the three criteria attributes in a 3-dimensional space. If you begin to partition the space using the classification tree shown in Figure 2, you will see that the resulting partitions are disjoint and occupy the entire defined space. Each partition can then be marked as either Yes or No to answer the question of Lives in Suburb. This can be seen by the following diagram:

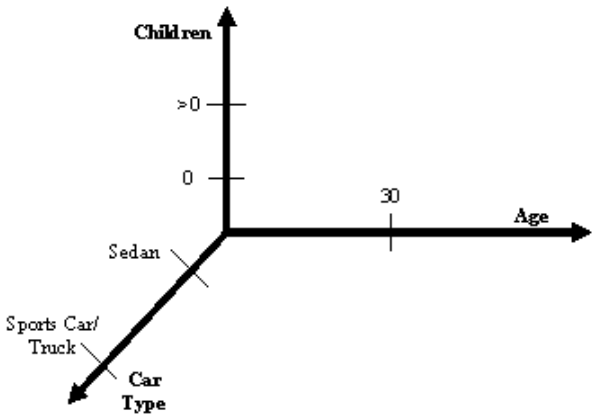

Figure 3. Graph for partitioning all possible combinations of attribute values.

Another representation of the classification tree can be made by a list of rules:

If Age is $<= 30$ and CarType = Sedan
Then return yes
If Age is $<= 30$ and CarType = Truck/Sports car
Then return no
If Age is > 30 and Children $= 0$ and CarType = Sedan
Then return no
If Age is > 30 and Children $= 0$ and CarType = Truck/Sports car
Then return yes
If Age is > 30 and Children > 0 and CarType = Sedan
Then return yes
If Age is > 30 and Children > 0 and CarType = Truck/Sports car
Then return no

This representation is not as compact as the tree representation.

Types of Classification Trees:

Simple Subtask

We are trying to build a classification tree with 1 split and 2 leaves. What is the best attribute and predicate to use?

To determine this, we must determine which type of classification tree we are using. There are two types of classification trees:

1. Quinlan (1986), which is modeled from a machine learning perspective

2. Breiman (1984), which is modeled from a statistics perspective

For continuous attributes, both Quinlan and Breiman split the predicates with the left side containing $<=$ and the right side containing $>$. For discrete attributes, Quinlan splits on all values of the attributes, while Breiman splits values into two sets, essentially making it a binary tree.

Quinlan approach:

Advantage

- You only need to decide what the split attribute is, after which it is immediately obvious what the split predicates should be.

Disdvantage

- A lot of work is repeated. The data can be fragmented too quickly. This approach can be easily influenced by noise in the data.

Breiman approach:

Advantage

- Avoids repeating work.

Disdvantage

- Because every node is split into two groups, the number of splits will be $2^{(\text{attributes})}$, which can be tricky to deal with as the number of attributes grows.

How to determine whether a split is good?

The probability of splitting something to the left can be represented as P(x) and the probability of splitting something to the right can be represented as P(x). Goodness of the split can then be measure by a formula

$$\text{Goodness}(x, P(x))$$

Using the training data D as the attribute label, we can represent a predictor (yes/no) with N_Y and N_N, which represent the number of yes and the number of no predictions. After a split the left side D_L will have L_Y and L_N, which represent the number of yes and the number of no predictions in the left side. The same can be said for the right side

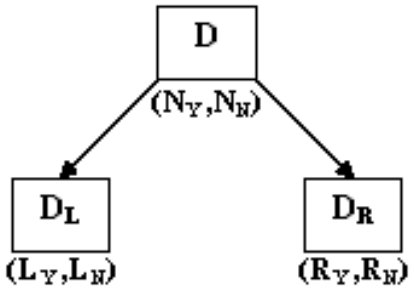

Figure 4. Splitting the data into two disjoint datasets.

The ideal case is that only YES occurs on one side and that only NO occurs on the other. The worst case is if the ratio of N_y and N_n in the parent node is preserved after the split. The ratio can be expressed by the following formula:

$$P[Y|D] = N_Y/(N_N + N_Y)$$
$$P[N|D] = N_N/(N_N + N_Y)$$

There are ways to measure the quality of the split. This is called an impurity measure. Here a good split reduces the impurity. A very simple impurity measure can be represented by the following formula.

$$i(P_Y, P_N) = \begin{cases} 0, & \text{if } P_Y = 1 \text{ or } P_N = 1 \\ \max, & \text{if } P_Y = P_N \end{cases}$$

Figure 5. A simple impurity formula.

Types of Impurity Measures:

Gini index (of diversity)
Here, if the split is pure, then there is no diversity. The formula for a Gini index can be represented by the following:

$$\text{gini}(P_y, P_n) = 1P_y^2P_n^2$$

The graph of this function is as follows:

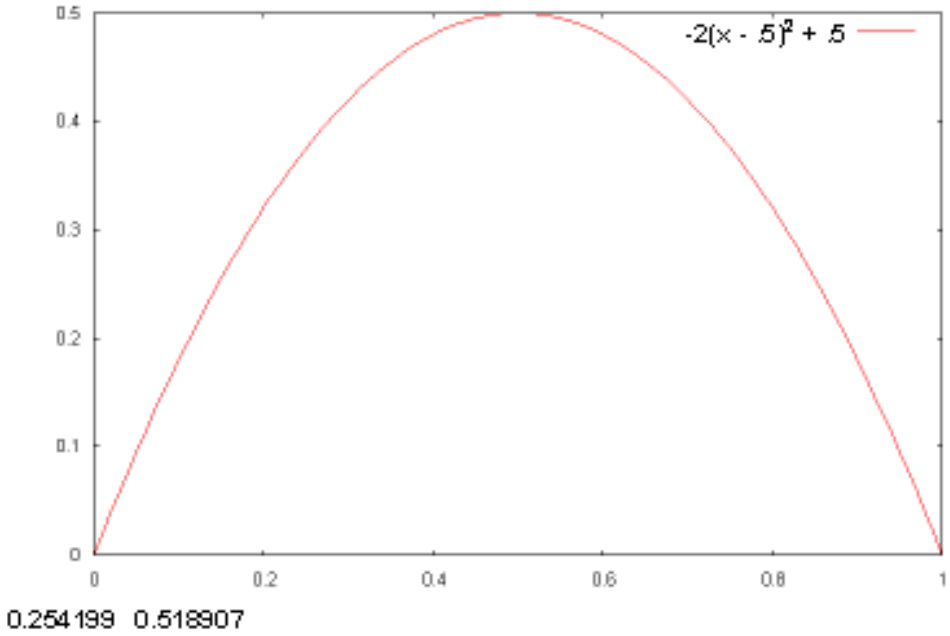

Figure 6. A graph of a standard gigiindex.

Entropy
This can be represented by the following formula

$$\text{entropy}(P_i, \ldots, P_k) = -\sum_{i}^{k}(P_i lg(P_i))$$

Scribed by: Vahe Koestline and Tim Clark