Lecture 12: Decision Trees

Iain Styles

22 November 2019

Decision Trees

Decision trees are designed to reflect, in a weak sense, the way in which humans make decisions. After all, it is rather unnatural for us to think in terms of mapping every problem into a vector notation. In a decision tree, we learn an explicit set of binary decisions on the features in the data in order to arrive at a final decision. This is perhaps best illustrated with a (trivial) example. Consider the process that you may go through when considering whether to buy a new laptop. You may ask the following questions:

- Do I need a new laptop?
- Can I afford it?
- Does it meet my specification?

These characteristics (need/afford/specification) are essentially the *features* based on which we will make a classification decision (buy or not). These questions can be visualised in a tree-like diagram and the path by which one might make a decision is shown in Figure 1

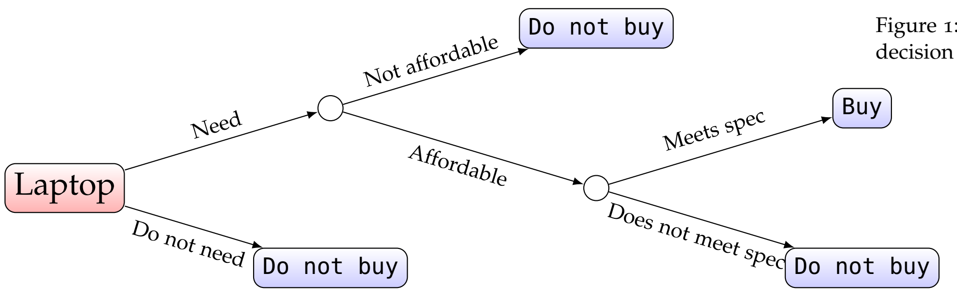

Figure 1: A simple example of a decision tree.

This is, of course, not a unique tree for this example. Figure 2 shows another way in which this tree could be constructed.

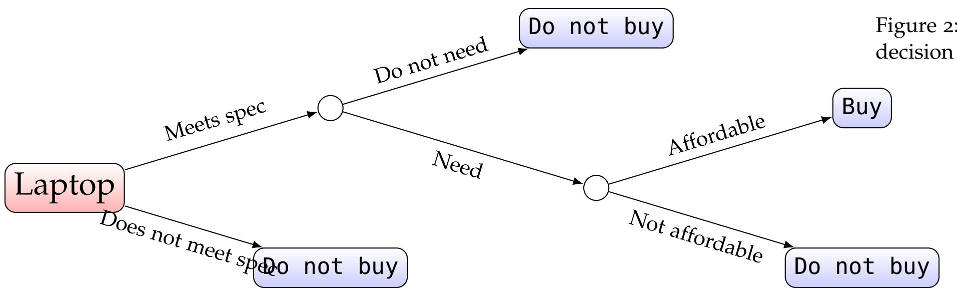

Figure 2: A second example of a decision tree with a different ordering.

The ordering in which we, as individuals, construct this tree will depend on our own personal priorities. In machine learning, we hope to be a little more objective about the way we make our decisions and in constructing a decision tree, we will need to consider how we choose the following aspects of our tree:

- In what order should we consider the features?
- What should our decision-making criteria be for each feature?

To illustrate this, a further refinement of our decision tree is shown in Figure 3. In this formulation, we have taken whether the meets the specification or not to be the most informative feature: if it does not, we absolutely will not consider buying it and the other features are not considered. If it does, then we consider the cost, introducing a specific price above which we will not buy the laptop, regardless of whether we need it. The key idea when learning a decision tree is that we need to learn, from the data:

1. Which feature should we next consider?
2. What the decision threshold value of each feature should be.

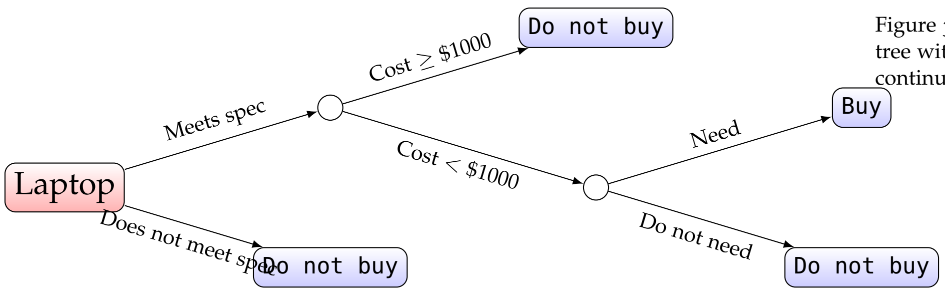

Figure 3: A third example of a decision tree with a different ordering and a continuous variable.

The C4.5 Algorithm

Perhaps the most common method for learning a decision tree is the *C4.5* algorithm developed by Ross Quinlan. This method performs a recursive partioning of the dataset in the following way (a simplified version):

1. Start with a dataset $\mathcal{D} = \{\mathbf{x}_i, t_i\}_{i=1}^{N}$ with binary target variables $t_i = \{-1, 1\}$. Each data point $\mathbf{x}_i$ is a vector of P features.
2. Determine which feature is best able to split the dataset according to its target values and determine the value on which to split.
3. Split the dataset into two according to the decision learned in the previous step.
4. Recurse on the two partitioned subsets.
5. Stop recursing if a subset contains samples from only one of the target classes

The key issue therefore becomes how to split the dataset into two, and how to determine which feature is the best one to split on next. A natural way to make this decision is to choose these factor according to how much they improve our ability to make a decision. A clear decision can be made when each of the two splits contains only data from one of the two target variable classes, so we see that the split has to be chosen to make the two parts as homogeneous as possible – we ideally want to put all samples with one target value in one group, and all those with the other target value in the other group. The choice of feature on which to split is therefore the one that acheives the greatest homogeneity in each half of the split.

The criterion typically used for splitting is the *information gain* following a split, which is related to the *entropy* of the dataset. Entropy is a concept that originated in statistical thermodynamics as a way to reason about the properties of large collections of atoms/molecules. It is a rough measure of the amount of disorder in a system and its definition, $S = k \ln w$ (where k is Boltzmann's constant) reflects this. It states that the entropy S is the log of the number of possible microscopic configurations w that could give rise to the same macroscopic properties. A classical example of this is to consider a box full of some sort of gas. There are many possible arrrangements of the molecules in which they are uniformly distributed in the box, and these will be indistinguishable to us from the outside. This state has a high entropy.

It is also possible (althought very unlikely) that the atoms could end up all packed very close together in one corner of the box. There are very few ways in which this can be done (which is why it is unlikely) and this is therefore a low entropy state. The parallel with our current situation is reasonably obvious: we need to

- Split the data on each feature to minimise the entropy in the two branches (make them as homogeneous as possible)
- Choose the feature that minimises the total entropy

In information theory, entropy is defined in a slightly different way to the thermodynamic entropy. The analogue of w is the probability density p_i for discrete outcomes i, and in terms of which the *information entropy* is defined as

$$S = -\sum_i p(i) \ln p(i) \tag{1}$$

This definition tells us how much information an event contains: unlikely events (small p) give a large value of S and carry more information than highly likely events. To understand this, consider the example of a single binary variable for which $p(0) = p$ and $p(1) = 1 - p$. The entropy of this variable as a function of p is $S = -p \ln p - (1-p) \ln(1-p)$ which we plot in Figure 4. We observe that in the case where the output is entirely predictable ($p = 0$ or $p = 1$) then the entropy is zero: there is no information because the variable is entirely predictable. The entropy is at its highest when $p = 0.5$ and the variable is at its most unpredictable.

In the context of splitting a dataset into two homogeneous parts, we observe that a dataset that is homogeneous has a low entropy (there are no surprising events). It therefore follows that we should choose to split the data in a way that provides the biggest *reduction in entropy*. This is equivalently referred to as the biggest *information gain*. We therefore construct decision trees in a *greedy* manner, by ordering the decisions by the amount of information that is gained at each split.

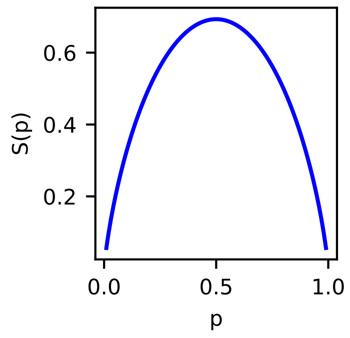

Figure 4: The entropy of a binary variable for which $p(0) = p$ and $p(1) = 1 - p$.

Since lowering entropy implies gaining information, the information gain in splitting the data can be is defined as the difference between the entropy of the parent P and the entropy of the children $C = \{c_i\}$. Given an n-way split of the sample, we write this as the entropy of the parent minus the weighted (by relative probabilities) sum of the children's entropy:

$$G(P,C) = S(P) - S(C) \tag{2}$$

$$- \sum_{i \in P} p(i) \ln p(i) - \sum_{c \in C} p(c) \sum_{i \in c} -p(i|c) \ln p(i|c) \tag{3}$$

N	Need	Afford	Spec	Buy
1	T	F	T	F
2	F	T	F	F
3	T	F	T	T
4	T	F	T	T
5	F	T	F	F
6	T	T	T	T
7	F	F	F	F
8	T	T	T	T
9	F	T	T	T
10	T	F	F	F

Table 1: Table for outcomes for the shoe-buying example.

Let us do a concrete example of this, using our laptop-buying example. A sample dataset is given in Table 1. Let us begin to construct the decision tree.

Our first choice is to determine on which variable we should split first. We need to first compute the entropy of the whole dataset. There are ten samples from which the outcome was true on four occasions and false on six. The entropy of the parent is therefore

$$S(P) = -0.4 \ln 0.4 - 0.6 \ln 0.6 = 0.673 \tag{4}$$

Let us split the dataset on each of the three independent variables. First, we split on "Need", dividing the dataset into two groups: six samples for which "Need" is true, and four for which it is false. Of the six "true's", four lead to "buy" and two to "not buy". Of the four false's, none lead to "buy" and four lead to "not buy". The children's entropy is therefore

$$S(C) = \sum_{c \in C} p(c) \sum_{i \in c} -p(i|c) \ln p(i|c) \tag{5}$$

$$= \left[p(\text{Need}) \times \sum_{i \in \text{Need}} -p_i \ln p_i \right] + \left[p(\neg\text{Need}) \times \sum_{i \in \neg\text{Need}} -p_i \ln p_i \right] \tag{6}$$

$$= 0.6 \times \left(-\frac{4}{6} \ln \frac{4}{6} - \frac{2}{6} \ln \frac{2}{6} \right) + 0.4 \times (-1 \ln 1 - 0 \ln 0) \tag{7}$$

$$= 0.382 \tag{8}$$

The information gained is therefore $0.673 - 0.382 = 0.291$. Let us do this for the other variables.

For Affordability, the split is 5T, 5F. The 5T lead to 2T, 3F outcomes; the 5F lead to 2T, 3F outcomes. The children's entropy is therefore

$$S(C) = 0.5 \times \left(-\frac{2}{5}\ln\frac{2}{5} - \frac{3}{5}\ln\frac{3}{5}\right) + 0.5 \times \left(-\frac{2}{5}\ln\frac{2}{5} - \frac{3}{5}\ln\frac{3}{5}\right) \tag{9}$$
$$= 0.5 \times 0.673 + 0.5 \times 0.673 = 0.673 \tag{10}$$

No information has been gained. We should not split on this variable.

For specification, the split is 6T, 4F. The 6T give 5T, 1F outcomes; the 4F give 0T, 4F outcomes.

$$S(C) = 0.6 \times \left(-\frac{5}{6}\ln\frac{5}{6} - \frac{1}{6}\ln\frac{1}{6}\right) + 0.4 \times (-1\ln 1 - 0\ln 0) \quad = 0.270 + 0 = 0.270 \tag{11}$$

The information gain here is therefore $0.673 - 0.270 = 0.403$.

The best inital predictor for the outcome is therefore whether the laptop has the right specification. This should therefore be the first split in the tree. Subsequent partionings of the data follow the same principle and the tree can be constructed recursively. These arguments can be generalised to deal with continuous variables although we will not consider that here. The aim has been to understand the general principles by which these tree are constructed.

Decision trees are easy to understand and their predictions are very interpretable. For a given sample, each node of the tree can be interrogated to understand why that branch was taken. This makes decision trees an appealing option for situations where each decision needs to be accompanied by a rationale for that decision. However, for a majority of applications, they suffer from a serious problem: they have a very strong tendency to overfit and to change substantially with different data: they are low bias,high variance. A tree that contains enough features and therefore has a high depth can essentially exactly fit its training date, but will often break down quite spectacularly on unseen data. It is therefore very uncommon for single decision trees to be used in isolation. The exception to this would be in situations where the depth of the trees is very limited because insufficient features are available.

One very common way in which the low-bias, high-variance tendency of decision trees can be overcome is to use many of them together, in what is known as an *ensemble*. Next lecture, you will be introduced to two different ensemble-based methods: boosting, and random forests.