CSE 446: Week 2: Decision Trees (Apr 1)

Instructor: Sergey Levine

I. The patient dataset

Here is the dataset from the previous lecture again, for reference:

	fever	cough	strange dreams	has disease?
Patient 1		x	x	
Patient 2	x	x	x	x
Patient 3	x		x	x
Patient 4	x	x		
Patient 5	x			x
Patient 6	x			

II. Define the splitting criterion

In practice, when we work with decision trees, we use a different objective function instead of misclassification that is a bit more sophisticated, and tends to work slightly better in practice. This objective is called information gain, and it requires a little bit of probability theory to understand.

First, let’s go back to the example of the ill-behaved patient in part III. Clearly there is something going on with patient 6: in the deterministic world modeled by decision trees, we can’t have a situation where two identical patients have a different label. Either patient 5 is lying about not having strange dreams (but patient 6 isn’t), or the disease afflicts patients fever but no strange dreams randomly. If we assume the latter case, which is quite reasonable (after all, there is plenty of random chance with diseases), our model of the world is that patients who have a fever but no strange dreams have a *chance* of having the disease.

Exercise VI. 1. What is the chance that patients who have a fever and no strange dreams have the disease? Answer: patients 4, 5, and 6 meet these criteria, and only one of them has the disease, so from this data, we can conclude the probability is 0.33.

While these kinds of stochastic (non-deterministic) leaves are a fact of life, we might intuitively prefer leaves that are more deterministic or “pure.” In fact, the closer the distribution at the leaf is to uniform, the less useful that leaf is for: if we have a leaf where the probability of the disease is 50%, we haven’t really gained anything from building our decision tree, because the best we can do when we get to this node is guess randomly. So can we quantify how good a leaf based on how “far” it is from the uniform distribution?

Information theory provides us with a tool for doing just that, called entropy. Entropy is defined as the number of bits that we need to encode a randomly drawn value of the variable y from the distribution p(y) -- which in our case is the distribution at the leaf. The uniform distribution is the most random, so we need an entire bit (= 1) to encode the information content of a sample. A sample from a deterministic distribution carries no information at all, because we already know it’s deterministic: if we sample from a distribution where p(true = 1.0), we have gained no information, and we have zero bits. In general, for a random variable Y that takes on k values, the entropy (the number of bits in a random sample) is defined as:

$$H(Y) = -\sum_{i=1}^{k} P(Y = y_i) \log_2 P(Y = y_i)$$

For a binary random variable, this becomes:

$$-P(Y = \text{true}) \log_2 P(Y = \text{true}) - (1 - P(Y = \text{true})) \log_2 (1 - P(Y = \text{true}))$$

Now, if we want to evaluate how valuable a given split on some input variable X is, we can consider the conditional entropy: this is the *additional* information obtained from a random draw of Y if we *already know* X. First, note that the conditional distribution of Y given some value of X is simply the fraction of “True” values in the leaf that results from splitting on X. The conditional entropy is then the sum of the entropies of these leaves, weighted by the probability of reaching each leaf, as determined by the dataset:

$$H(Y|X) = -\sum_{j=1}^{v} P(x_j) \sum_{i=1}^{k} P(y_i|x_j) \log_2 P(y_i|x_j)$$

Consider the 6-patient dataset we have. If we on fever, we have P(x = true) = ⅚, P(x = false) = ⅙, the entropy in the false leaf is zero, and the entropy in the true leaf is ⅗ * log ⅗, so the conditional entropy of this split is ⅙ * 0 - ⅚ * (⅗ * log ⅗ + ⅖ * log ⅖).

The objective that is most commonly used to build decision trees is called information gain, and is defined as the difference between the current entropy of a leaf and the conditional entropy of choosing the best split:

$$IG(X) = H(Y) - H(Y|X)$$

At each recurrence, the leaf where the best split gives the largest information gain is split on the best attribute.

III. Entropy vs error rate and other costs

Entropy is most often used as the cost function in decision trees for a variety of reasons, the simplest being that it seems to work a little bit better in practice. The theoretical justification for it is a bit involved, and I should say that the difference is not huge. In fact, there is a third objective called the Gini index, that is discussed in the Murphy book in your assigned reading.

There is a theoretical interpretation of what entropy means in this context. Let's say that instead of simply taking the majority class at each leaf of the tree, we randomly choose one of the datapoints at the leaf and output its label. This approach might hedge our bets a little bit better, because if we have a leaf that is precisely half of one class and half of another, we'll succeed at least some of the time. Sometimes that might be desirable, though in expectation it makes no difference. In this case, we would say that the label for each leaf is uncertain: it has some probability of being true, and some probability of being false. Entropy is then the expectation of the log probability: the expected (negative) log likelihood. Maximizing log-likelihood is a commonly used objective in machine learning, and is referred to as maximum likelihood estimation, so from a theoretical standpoint, this type of cost makes a lot of sense. But this is something that will become clearer after we cover point estimation next week.

From a more intuitive standpoint, we can look at a plot of entropy vs error rate as a function of the fraction of a node that belongs to a given class. This plot is shown in the Murphy text in Figure 16.3. Note that error rate drops off linearly as a function of class composition, while entropy has much steeper drop on either side. Both are zero when the node is "pure" -- that is, all datapoints that belong to it have the same label. But they drop at different rates. In particular, this means that the entropy measure places much greater importance on a class being perfectly pure as opposed to only mostly pure.

Murphy illustrates to some degree why this might be helpful: imagine that we have a dataset with 400 datapoints in each of two classes, and we can choose two possible splits: one split gives us two leaves, one with (300,100) and one with (100,300). The other split gives us a leaf with (200,400) and (200,0). In both cases, we will have a misclassification rate of 0.25, so the error rate metric considers these equally good. But entropy will prefer the latter case with the completely pure class on the right. If there is an attribute that creates a completely pure class like this, it is likely a very useful attribute for classification, so this lends some intuition for why entropy might be a good criterion, particularly when we grow our tree greedily.

At this point, we go back to the slides for an example of some realistic dataset tree growing (see lecture slides).