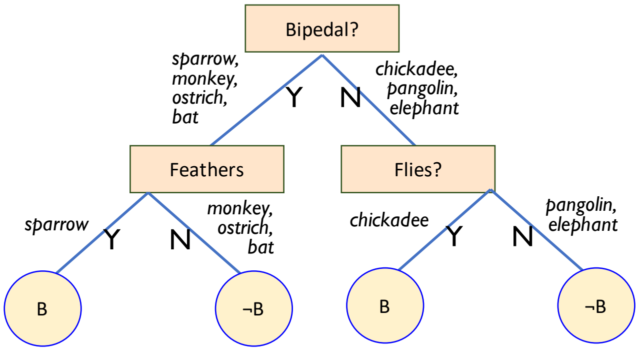
Information Gain for Splitting a Decision Tree

For some dataset, our goal is to get to the point where all the leaves are the same—that is, we can say that things at that leaf node belong to a certain class. Here's an example we did in class, which uses the attributes Bipedal, Feathers, and Flies to classify whether something is a Bird. Each time we split on an attribute, we end up with a *subset* of the data on one side of the split:

Examples (training data)	Attributes			Outcome
	Bipedal	Flies	Feathers	
Sparrow	Y	Y	Y	B
Monkey	Y	N	N	M
Ostrich	Y	N	Y	B
Pangolin	N	N	N	M
Bat	Y	Y	N	M
Elephant	N	N	N	M
Chickadee	N	Y	Y	B



But this is a silly tree, because we could have just split on Feathers. More specifically, our goal is to find the attribute that, when we split on it, leads to subsets of the data that are *most homogeneous* or most *pure*. Splitting on Feathers would give us two subsets of the data, each of which is perfectly homogeneous—birds on one side, not-birds on the other.

So given a set of attributes, how do we work out the "best" split (the one that leads to the purest subsets of the data)? For that, we need some formal measure of homogeneity.

Entropy

Entropy can be used to characterize the (im)purity of an arbitrary collection of examples. The lower the entropy, the higher the homogeneity is. (For how this relates to the concept of information, see the lecture; for now, we're going to focus on how to calculate it.) What we want is a split of the data that is as pure as possible, so we're going to calculate, for any given split, how much it *changes* the entropy of the overall system. Our goal is to progressively reduce the total entropy by subdividing the data into classes.

The entropy of a dataset with *n* classes has to do with the *proportion* of the data in each subclass—that is, the probability of randomly picking a member of that class, p_n. That entropy[1] is defined as:

$$I = -(p_1 * \log_2(p_1) + p_2 * \log_2(p_2) + .. + p_n * \log_2(p_n))$$

This will give us a value of entropy that ranges from 0 (a perfectly homogenous dataset) to 1 (a completely impure dataset). Let's see that from some examples.

[1] We use I for entropy here. Sometimes people use S, sometimes E.

Consider the entropy of a dataset that's split into halves, exactly half one class and half another. This is as impure as a class can get, so we expect the highest possible entropy:

P = (0.5, 0.5); $I(P) = -(0.5 * \log_2(0.5) + (0.5 * \log_2(0.5)) = \mathbf{1}$

Yep, that's bad. However, consider a dataset that's 99% one class and 1% another, that is to say, an almost pure dataset:

P = (0.99, 0.01); $I(P) = -(0.99 * \log_2(0.99) + (0.01 * \log_2(0.01)) = \mathbf{0.08}$

That second one is obviously what we want to split our dataset on—the lower entropy reflects the fact that we're almost to the point where we've gotten to the "pure" classes we need at the leaf nodes of our decision tree.

We can do the same thing but make it a little more concrete. Let's say we have a dataset that has 9 samples from one class, and 5 samples from another. What's the entropy of that dataset?

$$\begin{aligned}\textbf{Entropy([9+, 5-])} &= -(9/14 * \log_2(9/14) + (5/14 * \log_2(5/14)) \\ &= -(0.64 * \log_2(0.64) + (0.357 * \log_2(0.357)) \mathbf{= 0.940}\end{aligned}$$

(You can see the slides for more examples.)

So now we can measure the purity of a set of data points that belong to classes. How do we turn that into a decision on what attribute to split a tree on?

Information Gain

Remember that we're trying to progressively reduce the entropy of the whole system, so for any *possible* split, what we care about is the reduction in entropy before and after we split the data. To do this, we calculate the *information gain*. This is a four-step process:

1. Calculate the entropy before the split.
2. Calculate the entropy of each branch.
3. Combine branch entropies, weighting each by how many data points that branch covers.
4. Subtract the combined entropy of the branches from the original.

So if we split the data intro three branches, we'll calculate entropy four times: once for the original dataset S, and once each for its children S_1, S_2, and S_3. We'll then combine the entropies of S_1, S_2, and S_3, taking into account how much data is in each, and subtract that combined value from S.

Formally, some attribute A divides the training set S into subsets $S_1, \ldots, S_v$ according to their values for A, where A has *v* distinct values. (So for the restaurants dataset, if we split on Type, we'll split into four subsets: French, Italian, Thai, and Burger.)

We can then define the information gain of splitting S on some attribute A as:

$$IG(S,A) = I(S) - \sum_{v \in Values(A)} \frac{|S_v|}{|S|} \times I(S_v)$$

Where $|S_v|$ and $|S|$ are the cardinality of those sets, so each subset is weighted by the proportion of the data it contains.

This is much easier to follow with an example.

Information Gain Example[2]

Consider a dataset S with ten items, split evenly into two classes: S = 🔵🔵🔵🔵🔵🔴🔴🔴🔴🔴. We'll say we're trying to classify things into blue and not-blue. Let's say we're considering splitting on some some attribute A that gives us the following split: S_1 = 🔵🔵🔵🔵, S_2 = 🔵🔴🔴🔴🔴🔴. Let's follow the steps given above:

First we calculate the entropy before the split, I(S):

I(🔵🔵🔵🔵🔵🔴🔴🔴🔴🔴) = 1

Then we calculate the entropy of each branch:

I_1(🔵🔵🔵🔵) = 0 (pure)
I_2(🔵🔴🔴🔴🔴🔴) = (⅙ log2(⅙) + ⅚ log2(⅚)) = 0.65

Then we calculate the entropy of the total split by weighting each branch's entropy by how many data points that branch covers:

I_{split} = (0.4*0) + (0.6*0.65) = 0.39

Then finally we subtract I_{split} from I:

1 - 0.39 = 0.61

And that's it. To build a decision tree, at every step we calculate the possible information gain of every attribute, split on the attribute that gives the highest information gain (that is, the greatest decrease in entropy), and—with luck!—we end up with a compact, expressive tree.

[2] Example from victorzhou.com/blog/information-gain/.