Example	Attributes										Target
	Alt	*Bar*	*Fri*	*Hun*	*Pat*	*Price*	*Rain*	*Res*	*Type*	*Est*	*Wait*
X_1	T	F	F	T	Some	$$$	F	T	French	0–10	T
X_2	T	F	F	T	Full	$	F	F	Thai	30–60	F
X_3	F	T	F	F	Some	$	F	F	Burger	0–10	T
X_4	T	F	T	T	Full	$	F	F	Thai	10–30	T
X_5	T	F	T	F	Full	$$$	F	T	French	>60	F
X_6	F	T	F	T	Some	$$	T	T	Italian	0–10	T
X_7	F	T	F	F	None	$	T	F	Burger	0–10	F
X_8	F	F	F	T	Some	$$	T	T	Thai	0–10	T
X_9	F	T	T	F	Full	$	T	F	Burger	>60	F
X_{10}	T	T	T	T	Full	$$$	F	T	Italian	10–30	F
X_{11}	F	F	F	F	None	$	F	F	Thai	0–10	F
X_{12}	T	T	T	T	Full	$	F	F	Burger	30–60	T

Figure 1.2: The WillWait Dataset

(4) How many nodes will a tree constructed like the previous problem contain in the worst case? Let n be the number of observations and m be the number of features.

1.3 Measuring the value of knowledge

Measuring Information

Alex and Betty want to exchange messages using an alphabet of 8 characters. They will encode their messages in binary. How many binary bits are required to encode each letter? Clearly 3 will suffice because $2^3 = 8$ and the 8 different 3-digit binary strings can each stand for a different letter. A=000, B=001, etc.

Is it possible to use fewer than 3 bits per letter? At first glance the answer may obviously seem to be "no." But imagine an extreme case where all of the words in the language only used two letters: A and H. Then they could use one bit: A = 0, H = 1 and communicate perfectly well. Even though there are 8 characters, they are not all equally prevalent in the language. If we use frequency analysis of

words in the language and determine some letters are more common than others, we can possibly get by with fewer than 3 bits.

Assume that we determine that A is the most frequent letter, occurring 50% of the time, while B,C,D,E,F,G,H each occur $\frac{1}{14}$ of the time, making up the other half. Let's devise a scheme that uses *on average* fewer than 3 bits per character. By "on average" we mean that if we encode a message of n letters, it will use fewer than $3n$ bits, assuming the letters in the message occur with the frequencies defined above. (More precisely, we're computing the *expected number of bits* in a letter – to be discussed below).

Since A is the most common, we could try encoding it with only 1 bit, say "0". The remaining 7 letters can be distinguished using only 3 bits each: B = 001, C = 010, etc. This would reduce the average bits-per-letter. But there's a problem: messages are ambiguous. How would you decode 00100010? is it ACBA, or BABA? Instead we'll use four bits for B-H, with high bit 1: B = 1000, C = 1001, etc. Now when decoding, a "0" signifies an A while a "1" signifies the start of a 4-bit code. 0010001001 = 0 0 1000 1001 = AABC.

The average bits-per-character, according to the probabilities given, is

$$B_{avg} = \frac{1}{2}(1) + 7 \times \frac{1}{14}(4) = \frac{5}{2} = 2.5.$$

We can interpret this figure, 2.5 bits, as a consequence of the skewed probability distribution of the 8 letters. If letters were equally probable then fully 3 bits are required to communicate each letter. This is a type of worst-case scenario. There is no reason to prefer any letter over any other. There is no advantage to knowing the probability distribution. Every single one of the 3 bits is informative. On the other extreme, a case where only letter "A" is ever used is a best-case scenario. Here 0 bits would be required. There is no information conveyed by knowing the next letter is an "A", because by knowing the distribution, that fact is obvious.

The 2.5 bit case is a middle ground. There is a certain advantage to knowing the distribution beforehand: you can guess the letter will

be A and half the time you'll be right. The leading bit "0" confirms your guess. If the letter is not "A", the leading bit "1" tells you that you need to see 3 more bits to determine the letter. On average there's 2.5 bits of "information" conveyed by knowing the next letter.

This idea of information is formalized in the next section on *entropy*.

Information and Entropy

Colloquially, the "entropy" of a system can be described as the amount of disorder or disarray in the system. Entropy and information are closely related. Consider for example a library. If every book is ordered on the shelves by author and title, then relatively little information is required to describe the arrangement of books. A list of titles and authors may suffice, perhaps along with the position of the shelves. But if somebody comes in and dislodges each book, throwing them all over the various floors of the library, then the system is suddenly very hard to describe. Each individual book will be in a specific position, and orientation, possibly open to a certain page. Some pages might be bent or ripped. Books may be intertwined with each other, opened partially, or bent fully back. The high-entropy arrangement of books is full of information.

These concepts can be applied to probability distributions, as motivated in the previous section. A uniform distribution, where each of n events is equally likely, is a high-entropy, high-information situation because any event could occur with the same probability. Conveying which event occurs takes $\log_2(n)$ bits. On the other hand, a distribution over n elements where only one element has all the probability is low-entropy, low-information. The outcome is already known. 0 bits are required to convey the event, because it's always the same. (And it just so happens that $\log_2(1) = 0$, so a bit of a pattern emerges).

We'll define the *information* of an event with probability p as simply $I(p) = -\log_2(p)$. For example an event that has probability $\frac{1}{8}$ will have an information value of $I(\frac{1}{8}) = -\log_2(\frac{1}{8}) = 3$, in agree-

ment with the previous example. An event with probability 1 has information value 0. An event with probability 2^{-20} has information content 20. Rare events have more information, because they are less likely, and more "news-worthy" when they occur.

The the *Shannon entropy* (or just *entropy*, attributed to Claude Shannon) is defined as the *expected value of the information* of a probability distribution.

$$h(p_1, p_2, \ldots, p_n) = -\sum_{i=1}^{n} p_i \log_2(p_i)$$

(Recall that expected value of a random variable is the sum of the values taken by that random variable, each weighted with its respective probability. For example, if a fair die is rolled, the expected value of the square of the value displayed on the die is $\frac{1}{6}(1+4+9+16+25+36) = \frac{91}{6}$)

Returning to the 8-character alphabet example, the Shannon entropy of the uniform distribution is

$$h(\frac{1}{8}, \frac{1}{8}, \ldots, \frac{1}{8}) = -8(\frac{1}{8}\log_2(\frac{1}{8})) = 3$$

while the skewed distribution is

$$h(\frac{1}{2}, \frac{1}{14}, \ldots, \frac{1}{14}) = -(\frac{1}{2}\log_2(\frac{1}{2}) + \frac{7}{14}\log_2(\frac{1}{14}) = 2.403$$

The first value agrees with the 3 bits required. The second is slightly less than 2.5, suggesting that perhaps a more efficient coding scheme can be found, if these concepts of entropy and information are truly related.

In fact they are very related. If we accept the axiom that an event of probability p takes $-\log_2(p)$ bits to communicate to someone when it occurs, then the definition of entropy given above is exactly equivalent to the definition of *expected number of bits* in a letter given above. Of course the number of bits actually transmitted

depends on the chosen encoding scheme, so the equivalence seems to only be valid in the optimal (or perhaps limiting) case. The Shannon entropy provides a lower bound on the number of bits required in any such encoding scheme and, sometimes, that bound is achievable.[2]

Exercises

(5) What is the entropy of the values in the Wait? column of the WillWait dataset? We will refer to the entropy of the classifications as simply the entropy of the dataset.

(6) If you know that it is Friday and define a new dataset based on only the relative rows, what is the entropy of that dataset? What is the entropy if it is *not* Friday?

(7) Compute an "average entropy" for the value *Friday* by averaging the two entropys corresponding to *Friday=T* and *Friday=F* that you found in th previous exercise. Be sure to weight each probability with its frequency (7/12 of the observations happen when Friday=F). How much, on average, does the entropy decrease from the original dataset if you know the value of "Friday"?

(8) Find a value for a feature in which the resulting entropy is 0. Find a value for a feature in which the resulting entropy is 1. Find a feature where every value has an entropy of 1.

1.4 Decision Tree Learning

All this talk about information and entropy will be put to use now in our decision tree learning algorithm. When any decision tree is

[2] This is the essential content of Shannon's source coding theorem. Two schemes, Huffman codes and arithmetic codes, can be shown to be optimally close to the Shannon bound under certain assumptions.