Information Entropy

Jim Mahoney
April 2005

Preface

Consider a stream of symbols such as ($a\ c\ b\ a\ a\ d\ a\ a\ d\ c$ ...). How much information is contained in such a stream? One way to approach this question is by imagining that we're receiving these symbols one at a time, and trying to guess which symbol will come next based on those alreaady sent. If, for example, we notice that every time a "q" is sent that the next character is always a "u", then sending that "u" doesn't really tell us anything that we don't already expect.

Claude Shannon defined a notion of "information entropy" that gives a rigorous version of this approach in his classic paper " A mathematical theory of communication". The purpose here is to summarize and explore some of his ideas.

Many discussions of this topic assume that a Markov process generates the sequence of symbols; that is, a set of conditional probabilities is given from which the stream is created. Here we'll also do the reverse, extracting the conditional probabilities and appproximations to the entropy from a given finite sequence of symbols.

Definitions

First, here are notations for the alphabet of symbols and sequences of these symbols.

M		number of distinct symbols in our alphabet
$\{x_i\}$	$\{a, b, c, ...\}$	a finite set of symbols, $1 \le i \le M$
$x,\ y,\ z$		variables representing symbols from $\{x_i\}$
N		number of symbols in a sequence S_N
S_N	$a\ b\ a\ ...\ c\ a$	a sequence of N symbols from alphabet $\{x_i\}$
S	$a\ a\ b\ c\ a\ ...$	limit $N \to \infty$ of S_N ; Shannon's "stream"

Second, notations for counting symbols, pairs, triples, and n-tuples, and the associated probabilities.

If the stream is defined by a Markov process, then the probabilities that define that process are the fundamental building blocks, not the symbol counts. In that case, we think of instances of S_N as generated by the probabilities. On the other hand, if we are given S_N, we can estimate probabilities and conditional probabilities by counting the various k-tuples.

To avoid edge effects in the counting, we'll assume a "periodic boundry condition", which takes ..., S_N, S_N, ... as the best approximation to S given S_N. In other words, for any finite sequence S_N, we find N different k-tuples by wrapping around from the end of the string to the beginning. For example, if $S_N = (a\ b\ c\ a)$ then the 4 pairs are $(a\ b)$, $(b\ c)$, $(c\ a)$, $(a\ a)$, and the 4 triples are $(a\ b\ c)$, $(b\ c\ a)$, $(c\ a\ a)$, $(a\ a\ b)$.

$n(x)$		number of occurances of symbol x in S_N
$p(x)$	$= n(x) / N$	probability of symbol x in S_N

$n(x\,y)$		number of occurances of adjacent pairs or 2-tuples ($x\,y$)
$p(x\,y)$	$= n(x\,y)/N$	probability of "$x\,y$" adjacent pairs (Shannon's $p(x,\,y)$)
$n(x\,y\,z)$		number of occurances of adjacent triples or 3-tuples ($x\,y\,z$)
$p(x\,y\,z)$	$= n(x\,y\,z)/N$	probability of "$x\,y\,z$" adjacent triples

Now for conditional probability notation and concepts for a sequence ($x\,y\,z$).

$p(y \mid x)$	$= p(x\,y)/p(x)$	the probability of y given the previous symbol x (Shannon's $p_x(y)$)
$\sum_y p(y \mid x)$	$= 1$	for a given x, the probability of something coming after it is 1
$\sum_x p(x)\,p(y \mid x)$	$= \sum_x p(x\,y) = p(y)$	the probability of y is the same as p(something y)
$p(z \mid x\,y)$	$= p(x\,y\,z)/p(x\,y)$	the probability of z given the preceding pair ($x\,y$)

Given all that, we can now define a decreasing sequence of approximations to H.

H_0	$= -\sum_{i=1}^{M} (1/M)\,\log_2(1/M) = \log_2 M$	zero'th order
H_1	$= -\sum_z p(z)\,\log_2 p(z)$	first order
H_2	$= -\sum_y \left\{ p(y) \sum_z \{ p(z \mid y)\,\log_2 p(z \mid y) \} \right\}$ $= -\sum_{y\,z} p(y\,z)\log_2[p(y\,z)/p(y)]$	second order
H_3	$= -\sum_x \left\{ p(x) \sum_y \left\{ p(y \mid x) \sum_z \{ p(z \mid x\,y)\,\log_2 p(z \mid x\,y) \} \right\} \right\}$ $= -\sum_{x\,y} \left\{ p(x\,y) \sum_z \{ p(z \mid x\,y)\,\log_2 p(z \mid x\,y) \} \right\}$ $= -\sum_{x\,y\,z} p(x\,y\,z)\log_2[\,p(x\,y\,z)/p(x\,y)]$	third order
H	$= H_\infty = \lim_{k\to\infty} H_k$ = bits of information per symbol	information entropy of S
H/H_0	(bits of info per symbol) / (max info per symbol)	
$1 - H/H_0$	redundancy fraction	

Finally, if the stream is generated by a Markov process in which the probability of a symbol is determined by the preceding k symbols, then $H = H_{k+1}$ exactly. Thus a zero-order Markov stream has $H = H_1$, a first-order Markov stream with each symbol's probability determined by its predecessor has $H = H_2$, and so on.

Examples

■ 1. {0, 1} with probabilities {1 / 2, 1 / 2}

If the alphabet is just {0, 1}, with independent probabilities

$p(0) = p(1) = \frac{1}{2}$.

Then

$H = H_1 = -\frac{1}{2}\log_2(\frac{1}{2}) - \frac{1}{2}\log_2(\frac{1}{2}) = 1$,

since this is a 0-order Markov process. (In other words, the probability of any symbol doesn't depend on any previous symbol.) This implies that a random string of 1's and 0's has 1 bit of information per symbol. The maximum entropy per symbol is $H_0 = \log_2 2 = 1$; therefore, the redundancy is 0%. A stream of equally probably random 1's and 0's therefore cannot be compressed without losing information.

■ 2. {*a*, *b*, *c*, *d*} with probabilities {1 / 2, 1 / 4, 1 / 8, 1 / 8}

This example (and its encoding in section **2ii** below) is from Shannon's paper.

Again consider a 0-order Markov process, in which each symbol's probability is indepdent of what's come before, this time with

$p(a) = 1/2$,
$p(b) = 1/4$,
$p(c) = 1/8$,
$p(d) = 1/8$.

Then

$H = H_1 = -\frac{1}{2}\log_2(\frac{1}{2}) - \frac{1}{4}\log_2(\frac{1}{4}) - \frac{1}{8}\log_2(\frac{1}{8}) - \frac{1}{8}\log_2(\frac{1}{8}) = \frac{1}{2} + \frac{2}{4} + \frac{3}{8} + \frac{3}{8} = \frac{7}{4} = 1.75$
$H_0 = \log_2(4) = 2.0$
redundancy $= 1 - H/H_0 = 1 - \frac{7/4}{2} = \frac{1}{8} = 0.125$

Thus a stream of such symbols carries 1.75 bits of information per symbol, as opposed to 2 bits per symbol if the probabilities were all equal. The redundancy is 1/8; therefore, such a stream could be compressed to 7 / 8 its orginal size while still using the same alphabet.

To see this more explicitly, let's look at several binary encodings of these four symbols.

■ 2 i. { 00 = *a*, 11 = *b*, 01 = *c*, 10 = *d* }

First let's consider a 2 bit encoding {*a*, *b*, *c*, *d*} → {00, 11, 01, 10}. With this encoding the stream of four letters turns into a stream of 1's and 0's which consider to be made up of "words" of two (0, 1) symbols. For example, (*d c b a*) would become (10 01 11 00) which consist of four words.

We might guess that since there are twice as many symbols there should be half as much information per symbol, which would imply that the entropy of these 1's and 0's should turn out to be $H = 1.75 / 2 = \frac{7}{8} = 0.875$. So let's see

We've already found the 0th order entropy for a 2 symbol encoding.

$H_0 = \log_2 2 = 1$.

To find the higher order entropies will need the probabilities of various *k*-tuples of these 0's and 1's.

The probabilities for (a, b, c, d) and therefore the words are

symbol	probability	1st (0,1)	2nd (0,1)
a	$\frac{1}{2}$	0_1	0_2
b	$\frac{1}{4}$	1_1	1_2
c	$\frac{1}{8}$	0_1	1_2
d	$\frac{1}{8}$	1_1	0_2

so, averaging over the 1st and 2nd positions within the words, the probabilities for the individual 0's and 1's are

$$p(0) = \frac{1}{2}\,p(0_1) + \frac{1}{2}\,p(0_2) = \frac{1}{2}\left(\frac{1}{2} + 0 + \frac{1}{8} + 0\right) + \frac{1}{2}\left(\frac{1}{2} + 0 + 0 + \frac{1}{8}\right) = \frac{5}{8}\,,$$
$$p(1) = \frac{1}{2}\,p(1_1) + \frac{1}{2}\,p(1_2) = \frac{1}{2}\left(0 + \frac{1}{4} + 0 + \frac{1}{8}\right) + \frac{1}{2}\left(0 + \frac{1}{4} + \frac{1}{8} + 0\right) = \frac{3}{8}$$

where subscripts like 0_1 mean a 0 as the 1st symbol in a word. (Note that with this encoding, the probabilties for the individual 0's and 1's are the same in the beginning and end of words. In other words, $p(0_1) = p(0_2) = p(0)$ and $p(1_1) = p(1_2) = p(1)$).

With these numbers in hand we can calculate the 1st order entropy,

$$H_1 = -\frac{5}{8}\log_2\frac{5}{8} - \frac{3}{8}\log_2\frac{3}{8} \simeq 0.9544\,.$$

One way to look at the probabilities of of pairs of 0's and 1's is to treat pairs within words differently than those across word boundries - after all, the statistics are different for alternating 0's and 1's in this encoding.

The first symbol in each word doesn't depend on the last symbol of the previous word, so for those first symbols the second order entropy is the same as the first order entropy.

$$H_2(\text{first symbol}) = H_1 \simeq 0.9544$$

For the second symbol in each word, the conditional probabilities of the 2nd given the 1st are calculated as follows.

$$p(0_2 \mid 0_1) = p(0_1\,0_2) \,/\, p(0_1) = \left(\tfrac{1}{2}\right)/\left(\tfrac{5}{8}\right) = \tfrac{4}{5}$$
$$p(1_2 \mid 0_1) = p(0_1\,1_2) \,/\, p(0_1) = \left(\tfrac{1}{8}\right)/\left(\tfrac{5}{8}\right) = \tfrac{1}{5}$$
$$p(1_2 \mid 1_1) = p(1_1\,1_2) \,/\, p(1_1) = \left(\tfrac{1}{4}\right)/\left(\tfrac{3}{8}\right) = \tfrac{2}{3}$$
$$p(0_2 \mid 1_1) = p(1_1\,0_2) \,/\, p(1_1) = \left(\tfrac{1}{8}\right)/\left(\tfrac{3}{8}\right) = \tfrac{1}{3}$$

Therefore

$$H_2(\text{second symbol}) =$$
$$-\,p(0_1)\,\left(p(0_2 \mid 0_1)\log_2 p(0_2 \mid 0_1) + p(1_2 \mid 0_1)\log_2 p(1_2 \mid 0_1)\right)$$
$$-\,p(1_1)\,\left(p(0_2 \mid 1_1)\log_2 p(0_2 \mid 1_1) + p(1_1)\,p(1_2 \mid 1_1)\log_2 p(1_2 \mid 1_1)\right)$$
$$= -\frac{5}{8}\left(\frac{4}{5}\log_2\frac{4}{5} + \frac{1}{5}\log_2\frac{1}{5}\right) - \frac{3}{8}\left(\frac{1}{3}\log_2\frac{1}{3} + \frac{2}{3}\log_2\frac{2}{3}\right)$$
$$\simeq 0.7956$$

The average of these two is then

$$H_{2\_\text{average}} = \left(H_2(\text{first symbol in word}) + H_2(\text{second symbol in word})\right)/2$$
$$= \frac{1}{2}\left(-\frac{5}{8}\log_2\frac{5}{8} - \frac{3}{8}\log_2\frac{3}{8}\right) + \frac{1}{2}\left(-\frac{5}{8}\left(\frac{4}{5}\log_2\frac{4}{5} + \frac{1}{5}\log_2\frac{1}{5}\right) - \frac{3}{8}\left(\frac{1}{3}\log_2\frac{1}{3} + \frac{2}{3}\log_2\frac{2}{3}\right)\right)$$
$$= \frac{7}{8} \text{ bits of information per (0, 1) symbol}$$

It isn't quite fair to call this the second order entropy, for we're treating every other bit differently. Still, it's nice to see that the information per symbol is indeed what we expected when we use all the information available.

To find the rigorous second order entropy following Shannon's definition it would be better to treat each symbol the same, rather than doing a different calculation for the first and second symbols in each word of two symbols. Then since the joint probabilities of symbols at the end of one word and the beginning of the next are just the products of the independent single symbol probabilities, we have the following joint probabilities across word

boundaries.

$$p(0_2\,0_1) = p(0_2)\,p(0_1) = \tfrac{5}{8}\,\tfrac{5}{8} = \tfrac{25}{64}$$
$$p(1_2\,1_1) = p(1_2)\,p(1_1) = \tfrac{3}{8}\,\tfrac{3}{8} = \tfrac{9}{64}$$
$$p(0_2\,1_1) = p(0_2)\,p(1_1) = \tfrac{5}{8}\,\tfrac{3}{8} = \tfrac{15}{64}$$
$$p(1_2\,0_1) = p(1_2)\,p(0_1) = \tfrac{3}{8}\,\tfrac{5}{8} = \tfrac{15}{64}$$

Then the pairwise probabilities ignoring word boundaries are the averages of those within and across words, since in choosing a random pair of bits that pair is equally likely to be within one word or straddling a word boundry.

$$p(0\,0) = \tfrac{1}{2}\,(\,p(0_2\,0_1) + p(0_1\,0_2)\,) = \tfrac{1}{2}\,(\,\tfrac{25}{64} + \tfrac{1}{2}\,) = \tfrac{57}{128}$$
$$p(1\,1) = \tfrac{1}{2}\,(\,p(1_2\,1_1) + p(1_1\,1_2)\,) = \tfrac{1}{2}\,(\,\tfrac{9}{64} + \tfrac{1}{4}\,) = \tfrac{25}{128}$$
$$p(0\,1) = \tfrac{1}{2}\,(\,p(0_2\,1_1) + p(0_1\,1_2)\,) = \tfrac{1}{2}\,(\,\tfrac{15}{64} + \tfrac{1}{8}\,) = \tfrac{23}{128}$$
$$p(1\,0) = \tfrac{1}{2}\,(\,p(1_2\,0_1) + p(1_1\,0_2)\,) = \tfrac{1}{2}\,(\,\tfrac{15}{64} + \tfrac{1}{8}\,) = \tfrac{23}{128}$$

This implies that

$$p(0\,|\,0) = p(0\;0) \,/\, p(0) = (\tfrac{57}{128})/(\tfrac{5}{8}) = \tfrac{57}{80}$$
$$p(1\,|\,0) = p(0\;1) \,/\, p(0) = (\tfrac{23}{128})/(\tfrac{5}{8}) = \tfrac{23}{80}$$
$$p(1\,|\,1) = p(1\;1) \,/\, p(1) = (\tfrac{25}{128})/(\tfrac{3}{8}) = \tfrac{25}{48}$$
$$p(0\,|\,1) = p(1\;0) \,/\, p(1) = (\tfrac{23}{128})/(\tfrac{3}{8}) = \tfrac{23}{48}$$

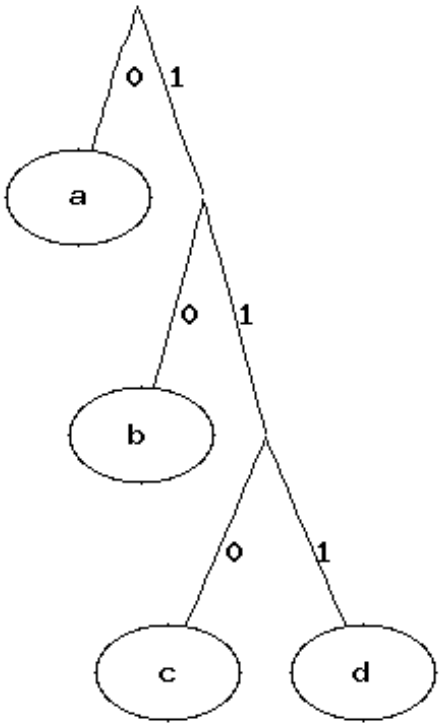
which gives

$$H_2 =$$
$$-\,p(0)\,(p(0\,|\,0)\log_2 p(0\,|\,0) + p(1\,|\,0)\log_2 p(1\,|\,0)\,) - p(1)\,(p(0\,|\,1)\log_2 p(0\,|\,1) + p(1)\,p(1\,|\,1)\log_2 p(1\,|\,1))$$
$$= -\tfrac{5}{8}\,(\tfrac{57}{80}\log_2\tfrac{57}{80} + \tfrac{23}{80}\log_2\tfrac{23}{80}) - \tfrac{3}{8}\,(\tfrac{23}{48}\log_2\tfrac{23}{48} + \tfrac{25}{48}\log_2\tfrac{25}{48})$$
$$\simeq 0.9154$$

Thus for this stream of 0's and 1's we have a decreasing series of approximate values of the entropy $\{H_0,\ H_1,\ H_2,\ ...\} \simeq \{1,\ 0.9544,\ 0.9154,\ ...\}$, which will asymptotically approach the true entropy $H = 0.875$.

■ 2 ii. { 0 = *a*, 10 = *b*, 110 = *c*, 111 = *d* }

Now consider a different binary encoding, $\{a,\ b,\ c,\ d\} \to \{0,\ 10,\ 110,\ 111\}$. This variable length scheme is an example of Huffman coding, and can be visualized by the following tree.



Since the words have different lengths, how can we tell which word a 0 or 1 belongs to when we decode? The answer is that because the begining of each word is unique, there's only one way to decode the words correctly. For example, (*a a b a c b a*) encodes to (0 0 1 0 0 1 1 0 1 0 0). To decode this, we start at the left and look at the first character, which is a 0. "a" is the only symbol which begins with 0, and so we have the first word, (0). The next character is another 0, and so again this must be an "a". Next is a 1. Since there isn't any word made up of just 1, we append the next symbol to get (10), which we see is "b", and so on.

The number of 0's and 1's it takes to encode a string with $N\{a, b, c, d\}$ symbols is just the number of 0's and 1's requirerd to code each letter times its expected frequency.

$$N_{\{0,1\}} = [1(\tfrac{1}{2}) + 2(\tfrac{1}{4}) + 3(\tfrac{1}{8}) + 3(\tfrac{1}{8})]\ N_{\{a,b,c,d\}} = 1.75\ N_{\{a,b,c,d\}}$$

Moreover, this string of 0's and 1's really does look entirely random; following the probabilities rightwards and down the tree diagram makes that pretty clear: First, the probability of a "0" or ("10", "110", "111") is 50/50, so the leftmost symbol has a 50/50 chance of being a zero or a one. If it was a zero, then the next symbol starts the same probabalities all over. If a one, then we have an even chance between "10" or ("110","111"); again, a 50/50 chance of a zero or one in the next position - and so on.

Calculating the conditional probabilities looking to the left, the preceding digits, is tricker but gives the same result. The trick is to start by calculating the probability that a given 0 or 1 came from an a, b, c, or d. These probabilities are

$$p(0 \text{ from an } a) = 1(\tfrac{1}{2}) / [1(\tfrac{1}{2}) + 2(\tfrac{1}{4}) + 3(\tfrac{1}{8}) + 3(\tfrac{1}{8})]$$

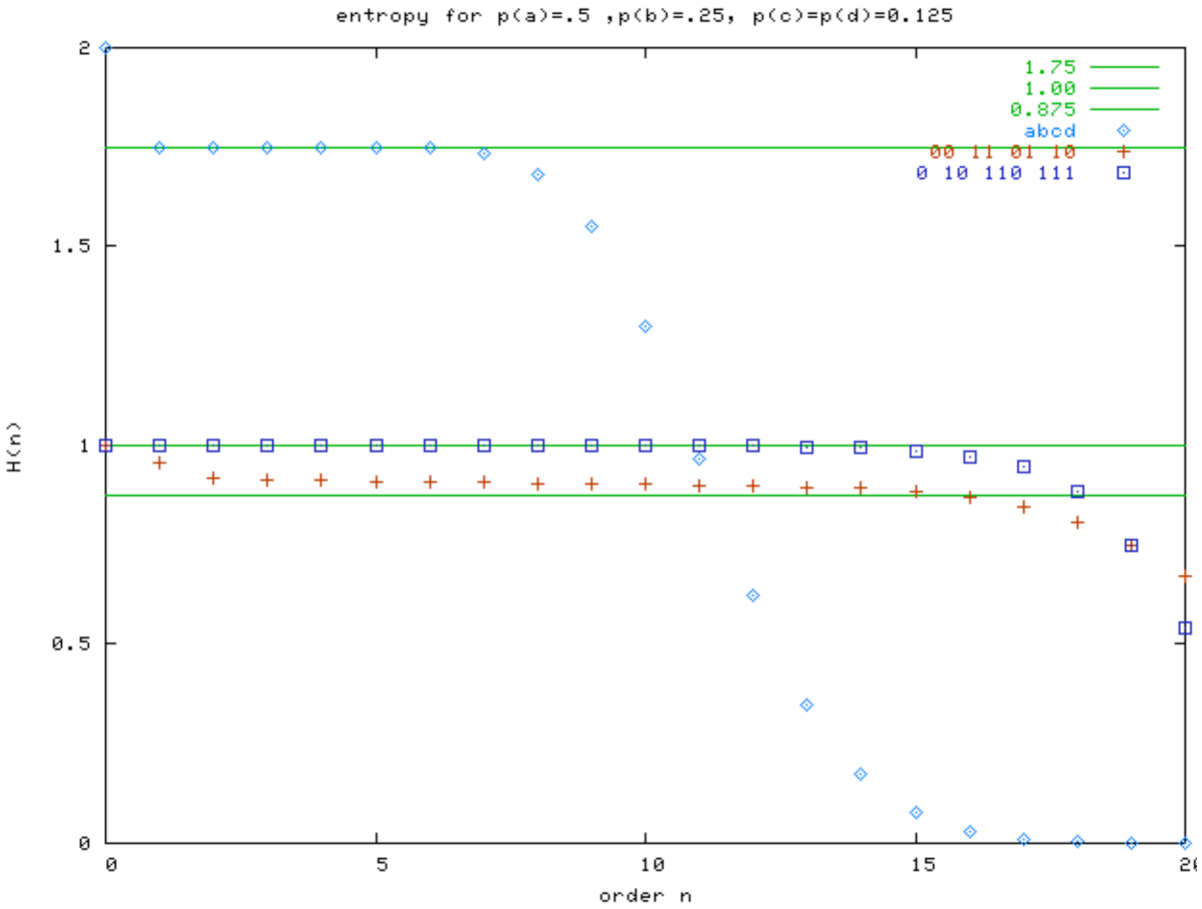
$$p(0 \text{ or } 1 \text{ from } b) = 2(\tfrac{1}{4}) / [1(\tfrac{1}{2}) + 2(\tfrac{1}{4}) + 3(\tfrac{1}{8}) + 3(\tfrac{1}{8})]$$

and so on. I'll leave the rest as an exercise.

- **2 iii. S_N = (*a a b a c a* ...) or (0 1 0 0 1 0 1 ...) or ...**

  To see what this looks like numerically, I've written a perl program.

  Turns out this isn't too hard to do, though the convergence is a bit tricky - if we pick too high an order for H_N, then we need some pretty long strings.
  Here's the picture of the results; the perl code is in "stream.pl"; pasting it here didn't do a good job keeping the text formatting.



Workspace