George Verghese
verghese@mit.edu

6.02
11/30/10

Rec. 20

Information, entropy, coding, compression

Data source puts out a sequence of messages

$$\boxed{X} \longrightarrow X[1], X[2], X[3], \dots$$

Assume these are independent, but are all distributed exactly as the random quantity X is:

X takes one of N possible values,

$$x_1, \dots, x_N$$

with respective probabilities

$$p_1, \dots, p_N .$$

Shannon 1948

Definition: The amount of information conveyed or revealed by announcing the outcome

$$X = x_i \quad \text{is} \quad \log_2\left(\frac{1}{p_i}\right) \underline{\text{bits}} .$$

This can also be taken as a measure of the uncertainty associated with this outcome prior to its being announced.

$\Longrightarrow$ More information is revealed (or more uncertainty is removed) when a less probable outcome is announced.

Caution: 'bit' has so far denoted simply a binary digit; but now 'bit'

$\longrightarrow$

can mean a unit of information. These are *distinct* notions, though there are connections (as we'll see).

What's the information revealed by announcing $X[1] = x_7$ and $X[2] = x_{13}$?

$$\Rightarrow \log_2\left(\frac{1}{p_7 \cdot p_{13}}\right) \text{ bits}$$

$$= \log_2\left(\frac{1}{p_7}\right) + \log_2\left(\frac{1}{p_{13}}\right) \text{ bits}$$

i.e., *information* is *additive* over *independent* experiments (a consequence of our using a log in the definition, and of the fact that *probabilities* of *independent* events multiply).

$\boxed{\text{Definition}}$: The *entropy* of the random quantity X is the *expected* (or average, or mean) *information*, or *average uncertainty*, over all possible outcomes, i.e.,

$$\sum_{i=1}^{N} p_i \log_2\left(\frac{1}{p_i}\right) \leftarrow \text{This sum is denoted by } H(X).$$

Unit of entropy is still *bits*

Comments: 1. If we ever need to differentiate $H(X)$ with respect to p_i, it's useful to recall that $\log_2(1/p_i) = \dfrac{\ln(1/p_i)}{\ln 2}$.

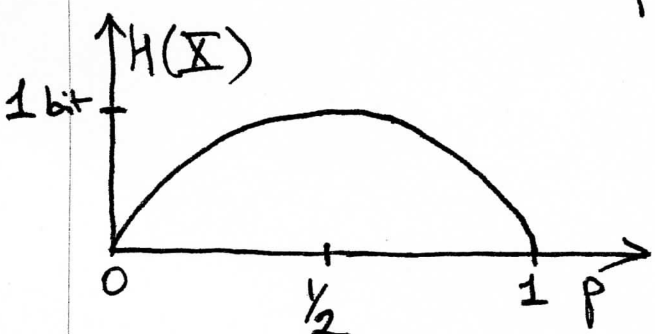
2. A term with $p_i = 0$ contributes $0 \log_2(1/0)$ to $H(X)$ — what sense to

make of this? — use L'Hôpital's rule, $\lim_{p \to 0} \left(p \log_2 \frac{1}{p} \right) = 0$. So such terms contribute nothing to $H(X)$.

At the other extreme, if $p_i = 1$ for some i, then $p_j = 0$ for $j \neq i$, and $H(X) = 0$ — there's no uncertainty, and no information to be revealed!

<u>Example</u>: Coin toss, $X = \begin{cases} \text{Heads with probability } p, \\ \text{Tails} \quad " \quad " \quad 1-p. \end{cases}$

$$H(X) = p \log_2 \frac{1}{p} + (1-p) \log_2 \frac{1}{1-p}$$



(can write $\log_2 \frac{1}{p}$ as $-\log_2 p$ if you prefer).

$H(X)$ turns out to be symmetric about $p = \frac{1}{2}$ (fair coin) and to have its maximum there, 1 bit at the max.

Until this point of the course, we would have said it takes 1 bit (i.e., binary digit!) to represent the outcome of a coin toss. But the above results suggest that the expected information in a coin toss is 1 bit (of information) only when $p = \frac{1}{2}$. As pointed out at the top of p.6 of Lec 22 notes, if $p = 0.999$, then $H(X) = .0114$ bits. For this case, the coin almost always (around 999 times out of 1000) comes up Heads, so the average uncertainty or expected information is very small. For 1000 independent tosses of this coin, the associated entropy $H(\underbrace{X^{1000}}) = 1000 \times .0114 = 11.4$ bits.

shorthand for '1000 independent realizations of the random quantity X'

Since information and entropy are additive over indep. trials

Is this saying that we may not need 1000 binary digits (e.g. '1' = Heads, '0' = Tails) to represent the outcome of 1000 tosses, but only something closer ~~to~~ $\underbrace{\text{11 binary digits}}_{\text{(on average)}}$? Yes! Why?

Very roughly:

The most typical sequences are those that have 999 Heads in 1000 tosses, so for these we only need to indicate where the single Tail occurs — 10 binary digits can do that (because they allow counting up to 1024). We can use other, longer codewords to represent all other non-typical outcomes, but since they have such low probability, it turns out they don't increase the average codeword length too much. (We'll say more about this shortly, in connection with Shannon's source coding theorem.)

Note: We noted for the coin that $H(X)$ was maximum when the two outcomes were equally likely. This is true more generally:

If there are N possible outcomes, $H(X)$ is maximized when all outcomes have equal probability $\Rightarrow p_i = \frac{1}{N}$, $H(X) = \log_2 N$. So if we have 2^k

$\rightarrow$

equally likely outcomes, then: $H(X) = k$ bits.

Prob 3 of Lec 22 Notes:

"X is an $\underbrace{\text{unknown}}$ $\underbrace{\text{8-bit}}$ binary number"

i.e., 8 binary digits !!

Interpret as: all 2^8 possible binary numbers are equally likely (who knew one word could say so much?!!). But then $H(X) = 8$ bits of information.

So in this case of 2^k equally likely possibilities, the two meanings of 'bits' coincide.

"You are given another 8-bit binary number Y, and are told the Hamming distance between X and Y is 1. How many bits of information about X have you been given?"

Again, this is binary digit!

Ans: X is now known to agree with Y in all-but-one position, so X is now one of $8 = 2^3$ equally likely possibilities, i.e., the entropy after the revelation is just 3 bits, reduced from 8 bits. So we have been given 5 bits of information about X.

The notion of entropy is central to communication theory and information theory. We shall only explore the role it plays as a benchmark/bound for the efficiency of binary codes for representing the outcomes associated with a random quantity X.

A <u>binary code</u> for X maps each outcome to a binary string. <u>e.g.</u>, take $N = 4$, so $X = x_1$ or x_2 or x_3 or x_4, assume probabilities

$$p_1 = 0.5,\ p_2 = 0.3,\ p_3 = 0.15,\ p_4 = 0.05.$$

A possible code is

$$x_1 \to 0,\quad x_2 \to 10,\quad x_3 \to 110,\quad x_4 \to 111$$

This is a <u>variable-length</u> code, i.e., the codewords have different lengths (unlike in a block code).

It is also a 'self-punctuating' code, in the sense that you don't need any punctuation marks (spaces or commas or...) to separate out the individual codewords in a string of them, e.g.

$$\underbrace{10}_{x_2}\underbrace{0}_{x_1}\underbrace{110}_{x_3}\underbrace{10}_{x_2}\underbrace{111}_{x_4}\underbrace{0}_{x_1}\underbrace{10}_{x_2}$$

this is the unique way to break it up; we didn't need help doing it

Why did this happen? Because no codeword is a <u>prefix</u> for another, i.e., the initial few bits of a codeword never yield another codeword — we say the code is <u>prefix-free</u> (many books just say 'prefix code').

And how is this code any better than simple binary counting of the outcomes, i.e.,

$$x_1 \to 00,\quad x_2 \to 01,\quad x_3 \to 10,\quad x_4 \to 11 \ ?$$

(in binary counting)

Well, the length of the codewords here is 2, but for the variable-length code above the <u>expected length</u>

$$L = 0.5\times 1 + 0.3\times 2 + 0.15\times 3 + 0.05\times 3$$
$$= \underline{1.7} \text{ binary digits}$$

So the variable-length code does a better job of compression (on average) into the shortest binary string.

What has $H(X)$ got to do with this?

Here $H(X) = .5 \log_2 2 + .3 \log_2\left(\frac{10}{3}\right) + .15 \log_2\left(\frac{100}{15}\right)$
$+ .05 \log_2(20) = \underline{\underline{1.648 \text{ bits}}}$

So $\boxed{L \geq H(X)}$ in this case.

The above inequality turns out to always be true for the expected code-word length of any prefix-free code.

Algorithm is in lecture notes, not repeated here.

→ Huffman coding yields a prefix-free code of minimum L, and satisfying $\boxed{H(X) + 1 \geq L \geq H(X)}$ — (*)

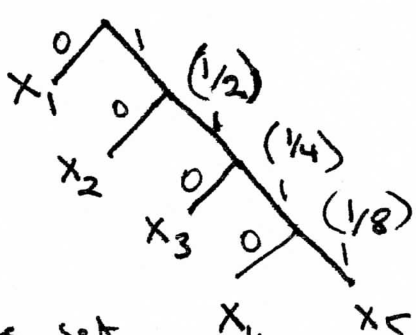
When the p_i are powers of $\frac{1}{2}$, the Huffman code actually attains the lower limit.

e.g. $N = 5$, $p_1 = \frac{1}{2}$, $p_2 = \frac{1}{4}$, $p_3 = \frac{1}{8}$, $p_4 = p_5 = \frac{1}{16}$

Then $H(X) = \frac{1}{2}\log_2 2 + \frac{1}{4}\log_2 4 + \frac{1}{8}\log_2 8 + \frac{2}{16}\log_2 16$
$= \frac{1}{2} + \frac{1}{2} + \frac{3}{8} + \frac{1}{2} = \underline{\frac{15}{8}}$ bits of entropy.

Huffman code:

$x_1 \to 0$, $x_2 \to 10$, $x_3 \to 110$,
$x_4 \to 1110$, $x_5 \to 1111$



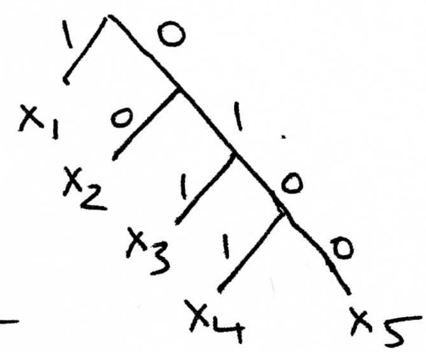
if we label every downward left edge 0 and downward right edge 1. (The codeword is the set of edge labels from 'root' to 'leaf'.)

$L = \frac{1}{2}\cdot 1 + \frac{1}{4}\cdot 2 + \frac{1}{8}\cdot 3 + \frac{2}{16}\cdot 4 = \frac{15}{8}$ bits, as expected.
(binary digits!)

(Simple binary counting with a fixed-length code would have needed $\underline{3}$ binary digits, whereas Huffman manages < 2.)

<u>Note</u>: The assignment of 0 to left and 1 to right on the Huffman tree is not needed. We can make the opposite assignment at any node, if desired. So here's another Huffman code (but equivalent) from the same tree as on p. 7.

$x_1 \to 1$, $x_2 \to 00$,
$x_3 \to 011$, $x_4 \to 0101$,
$x_5 \to 0100$.

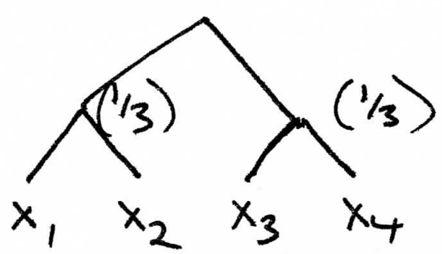


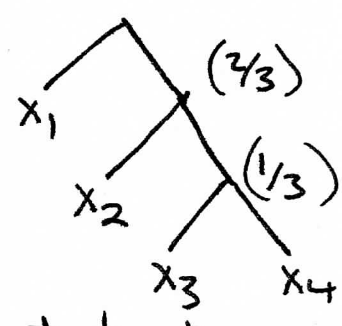
(The prefix-free property is still guaranteed by the tree structure, with the symbols sitting at the leaves of a binary tree.)

<u>e.g.</u> A more interesting non-uniqueness:

$$N = 4, \quad p_1 = p_2 = \frac{1}{3}, \quad p_3 = \frac{1}{4}, \quad p_4 = \frac{1}{12}$$

yields two Huffman-tree possibilities, depending on what choices are made when ties arise:





(and other structures equivalent to these two).

For the first case, $L = \frac{1}{3}\cdot 2 + \frac{1}{3}\cdot 2 + \frac{1}{4}\cdot 2 + \frac{1}{12}\cdot 2$
$= 2$, and for the second case

$L = \frac{1}{3}\cdot 1 + \frac{1}{3}\cdot 2 + \frac{1}{4}\cdot 3 + \frac{1}{12}\cdot 3 = 2$ again, as would have to be true.

How can we come closer to the lower bound on L in Eq(*), p. 7? Answer: Instead of coding symbol-by-symbol on the output of the data source on p. 1, <u>code</u> <u>blocks</u> <u>of symbols</u>, i.e., take the possible values that $(X[1], X[2], \ldots, X[M])$ can take, namely

$$\left.\begin{array}{ll} (x_1, x_1, \ldots, x_1), & \text{probability } p_1^M \\ (x_1, x_1, \ldots, x_2), & \text{probability } p_1^{M-1} p_2 \\ \vdots & \vdots \\ (x_1, x_1, \ldots, x_N), & \text{probability } p_1^{M-1} p_N \\ \vdots & \vdots \\ (x_N, x_N, \ldots, x_N), & \text{probability } p_N^M \end{array}\right\}$$

$(N)^M$ different 'super symbols', i.e., symbols for the block outputs.

Now do a Huffman code construction for this, to get a code of expected length L_M binary digits, satisfying

$$M H(X) + 1 \geq L_M \geq \underbrace{M H(X)}$$

this is the entropy associated with M independent realizations of the random quantity X.

Then the expected codeword length <u>per symbol</u> is $L = \frac{L_M}{M}$, so

$$\boxed{H(X) + \frac{1}{M} \geq L \geq H(X)}$$

This is the essence of the 'source coding theorem'

Shannon 1948

The larger M is, the closer we are guaranteed to approach the lower bound.