Rec. 20

Information, entropy, coding, compression

Data source puts out a sequence of messages

\[ X \rightarrow X[1], X[2], X[3], \ldots \]

Assume these are independent, but are all distributed as the random quantity \( X \) is:

\( X \) takes one of \( N \) possible values, \( x_1, \ldots, x_N \)

with respective probabilities \( p_1, \ldots, p_N \).

Shannon 1948

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

\( I = x_i \) is \( \log_2 \left( \frac{1}{p_i} \right) \) bits.

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

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".
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).

**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)\).
\]

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

2. A term with \( p_i = 0 \) contributes \( 0 \log_2(1/0) \) to \( H(X) \) — what sense to
make this? — use L'Hôpital's rule,
\[ \lim_{p \to 0} (p \log_2 \frac{1}{p}) = 0. \] 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!

Example: Coin toss, \( X = \) Heads with probability \( p \),
\( \) Tails \( \) \( \) \( 1 - p \).
\[ H(X) = p \log_2 \frac{1}{p} + (1-p) \log_2 \frac{1}{1-p} \]

(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 (in formation) only when
\( p = \frac{1}{2} \). As pointed out at the top of p.6 of Lec 22
notes, if \( p = 0.999 \), then \( H(X) = 0.0114 \) 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

\[ H(X^{1000}) = 1000 \times 0.0114 = 11.4 \text{ bits}. \]

Since information and entropy are additive over independent 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 11 binary digits? 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, larger 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:

\[ p_i = \frac{1}{N}, \quad H(X) = \log_2 N. \]

So if we have \( 2^k \)
The notion of entropy is central to information theory. We shall only explore the role of the Shannon entropy associated with a random quantity $X$ in this context. The entropy $H(X)$ is defined to measure the uncertainty or information content of $X$.

The equality $H(X) = 8$ bits means that after the specification of $X$, we have been given 8 bits of information about $X$.

As for $H(Y)$ and $H(Y|X)$, we have been given 8 bits of information about $Y$, and are told that the Hamming distance between $X$ and $Y$ is $d$. How likely is it that two messages $X$ and $Y$ are equally likely given the same 8-bit binary number $X$ is an unknown 8-bit binary number, i.e., 8 binary digits.

In this case, $H(X)$ is $8$ bits, and $H(Y)$ is $8$ bits. So in this case, $H(X)$ is 8 bits.
A binary code for $X$ maps each outcome to a binary string. e.g., 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:

$x_1 \rightarrow 0$, $x_2 \rightarrow 10$, $x_3 \rightarrow 110$, $x_4 \rightarrow 111$

This is a variable-length 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.:

$x_2 x_1 x_3 x_2 x_4 x_1 x_2$

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 prefix for another, i.e., the initial few bits of a codeword never yield another codeword — we say the code is prefix-free (many books just say 'prefix code').

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

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

Well, the length of the codewords here is 2, but for the variable-length code above the expected length $L = 0.5 \times 1 + 0.3 \times 2 + 0.15 \times 3 + 0.05 \times 3$

$= 1.7$ 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 \( H(X) = 0.5 \log_2 2 + 0.3 \log_2 \left( \frac{10}{3} \right) + 0.15 \log_2 \left( \frac{100}{15} \right) + 0.05 \log_2 20 = 1.648 \) bits

So \( L \geq H(X) \) in this case.

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

Huffman coding yields a code of minimum prefix-free length, and satisfying

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

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

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

Huffman code:
\[
x_1 \rightarrow 0, \quad x_2 \rightarrow 10, \quad x_3 \rightarrow 110, \\
x_4 \rightarrow 1110, \quad x_5 \rightarrow 1111
\]

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")

\[
L = \frac{1}{2} \cdot 1 + \frac{1}{4} \cdot 2 + \frac{1}{8} \cdot 3 + \frac{3}{16} \cdot 4 = \frac{15}{8} \text{ bits, as expected.}
\]
(Simple binary counting with a fixed-length code would have needed 3 binary digits, whereas Huffman manages < 2.)

Note: 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.

\[ x_1 \rightarrow 1, \ x_2 \rightarrow 00, \]
\[ x_3 \rightarrow 011, \ x_4 \rightarrow 0101, \]
\[ x_5 \rightarrow 0100. \]

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

E.g.: A more interesting non-uniqueness:

\[ N = 4 , \ P_1 = P_2 = \frac{1}{3} , \ P_3 = \frac{1}{4} , \ P_4 = \frac{1}{12} \]

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

(\[ \frac{1}{3} \] and other structures equivalent to these two).

For the first case, \[ L = \frac{1}{3}.2 + \frac{1}{3}.2 + \frac{1}{4}.2 + \frac{1}{12} = 2 \], and for the second case
\[ L = \frac{1}{3}.1 + \frac{1}{3}.2 + \frac{1}{4}.3 + \frac{1}{12}.3 = 2 \] again, as would have to be true.
How can we come closer to the lower bound on $L$ in Eq. (x)? Answer: Instead of coding symbol-by-symbol on the output of the data source on p. 10, code blocks of symbols, i.e., take the possible values that $(X[1], X[2], \ldots, X[M])$ can take, namely:

- $(X_1, X_1, \ldots, X_1)$, probability $p_1^M$,
- $(X_1, X_1, \ldots, X_2)$, probability $p_1^{M-1}p_2$,
- \[ \vdots \]
- $(X_1, X_2, \ldots, X_2)$, probability $p_2^{M-1}p_2$,
- \[ \vdots \]
- $(X_N, X_N, \ldots, X_N)$, probability $p_N^M$.

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

$$MH(X) + 1 \geq L_M \geq MH(X)$$

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

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

$$H(X) + 1 \geq L \geq H(X)$$

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

This is the essence of the "source coding theorem," Shannon, 1948.