How Long are Optimal Codewords?

Consider a source with 2^k symbols, each with probability $1/2^k$.

In the binary Huffman code for this source, all codewords are k bits long.

Consider a source with $2^j + 2^k$ symbols.
2^j of the symbols have probabilities of $1/2^{j+1}$.
The other 2^k have probabilities of $1/2^{k+1}$.

In the binary Huffman code for this source, symbols with probability $1/2^{j+1}$ have codewords of length $j+1$, while symbols with probability $1/2^{k+1}$ have codewords of length $k+1$.

Example: For $j = 1$ and $k = 2$:

Probability:	1/4	1/4	1/8	1/8	1/8	1/8
Codeword:	00	01	100	101	110	111

A Conjecture

From these examples, we can make a vague conjecture:

> A symbol with probability $1/2^k$ "ought" to be encoded in k bits. More generally, a symbol with probability p ought to be encoded in $-\log_2 p$ bits.

This conjecture works for the examples on the previous slide — the Huffman codes have codewords of the "right" length.

In general, however, Huffman codes *don't* have codewords of exactly the "right" length. For one thing, $\log_2 p$ is usually not an integer.

But might we get the "right" lengths by encoding blocks of symbols?

The Entropy of a Source

To formalize this conjecture, we'll first define the (binary) *entropy* of a source, $\mathcal{S}$, having q symbols with probabilities $p_1, \ldots, p_q$:

$$H(\mathcal{S}) \;=\; \sum_{i=1}^{q} p_i \log_2(1/p_i)$$

We define $p_i \log_2(1/p_i)$ to be zero if p_i is zero.

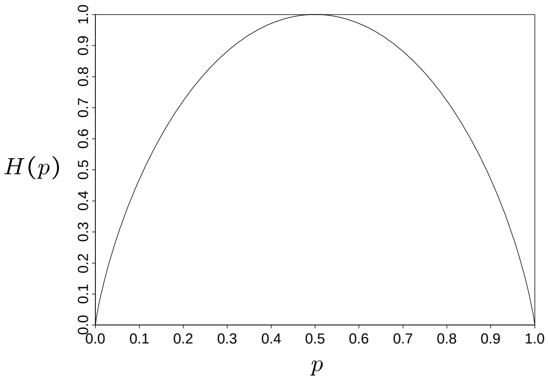
The r-ary entropy, $H_r(\mathcal{S})$, has the same form, but with logs to base r. The r-ary entropy is just in different units: $H_r(\mathcal{S}) = H(\mathcal{S}) / \log_2 r$, since $\log_r x = \log_2 x / \log_2 r$.

(Note: From now on, logs will be assumed to be to base 2 if no base is specified.)

We hope the entropy will turn out to be *the average number of bits per symbol in the most efficient encoding of the source.*

Entropy of a Binary Source

For a binary source, with symbol probabilities p and $1 - p$, the entropy as a function of p looks like this:



$H(0.1) = 0.469$, so we hope to compress a binary source with symbol probabilities of 0.1 and 0.9 by more than a factor of two. We obviously can't do that if we encode symbols one at a time.

Extensions of a Source

We formalize the notion of encoding symbols in blocks by defining the n-th *extension* of a source, $\mathcal{S}$, written $\mathcal{S}^n$.

If the source alphabet for $\mathcal{S}$ has q symbols, the source alphabet for $\mathcal{S}^n$ will have q^n symbols — all possible blocks of n symbols from S.

If the probabilities for symbols in $\mathcal{S}$ are $p_1, \ldots, p_q$, the probabilities for blocks in $\mathcal{S}^n$ are found by multiplying the p_i for all the symbols in the block. (This is appropriate when symbols are independent.)

Entropy of an Extension

We now prove that $H(\mathcal{S}^n) = nH(\mathcal{S})$:

$$
\begin{aligned}
H(\mathcal{S}^n) &= \sum_{i_1=1}^{q} \cdots \sum_{i_n=1}^{q} p_{i_1} \cdots p_{i_n} \log\left(\frac{1}{p_{i_1} \cdots p_{i_n}}\right) \\
&= \sum_{i_1=1}^{q} \cdots \sum_{i_n=1}^{q} p_{i_1} \cdots p_{i_n} \sum_{a=1}^{n} \log\left(\frac{1}{p_{i_a}}\right) \\
&= \sum_{a=1}^{n} \sum_{i_1=1}^{q} \cdots \sum_{i_n=1}^{q} p_{i_1} \cdots p_{i_n} \log\left(\frac{1}{p_{i_a}}\right) \\
&= \sum_{a=1}^{n} \sum_{i_a=1}^{q} \sum_{i_k \text{ for } k\neq a} p_{i_1} \cdots p_{i_n} \log\left(\frac{1}{p_{i_a}}\right) \\
&= \sum_{a=1}^{n} \sum_{i_a=1}^{q} p_{i_a} \log\left(\frac{1}{p_{i_a}}\right) \\
&\qquad\qquad \times \sum_{i_k \text{ for } k\neq a} p_{i_1} \cdots p_{i_{a-1}} p_{i_{a+1}} \cdots p_{i_n} \\
&= \sum_{a=1}^{n} \sum_{i_a=1}^{q} p_{i_a} \log\left(\frac{1}{p_{i_a}}\right) \;=\; nH(\mathcal{S})
\end{aligned}
$$

Another way of looking at it: The expectation of a sum is the sum of expectations.

We Can't Compress to Less Than the Entropy

We will prove that *any* uniquely decodable binary code for a source $\mathcal{S}$ must have average length of at least $H(\mathcal{S})$.

Any uniquely decodable r-ary code will have average length at least $H_r(\mathcal{S}) = H(\mathcal{S}) / \log_2 r$.

Applying this to the n-th extension, we see that the average length, L_n, will be at least $H(\mathcal{S}^n) = nH(\mathcal{S})$, and hence $L_n/n \geq H(\mathcal{S})$. So we can't compress below the entropy by encoding symbols in blocks.

Shannon's Noiseless Coding Theorem:

However, by using extensions of the source, we *can* compress arbitrarily close to the entropy!

Formally:

For any desired average length per symbol, R, that is greater than the r-ary entropy, $H_r(\mathcal{S})$, there is a value of n for which a uniquely decodable r-ary code for $\mathcal{S}^n$ exists that has average length less than nR.