

Huffman Coding

We consider the problem of coding a set of symbols by binary strings. Let the set of symbols be denoted by $\{s_i\}$ and their probabilities (of occurring or being transmitted say) are $\{p_i\}$. We require an efficient code that is instantaneously decodable. To illustrate the latter suppose we code s_1 as 0 and s_2 as 00. Then if we see 00 we cannot decide if this means s_1s_1 or s_2 ; it is not instantaneously decodable. To satisfy the required property no code should be a precursor of another. For example we could code the symbols above as $s_1 \rightarrow 0$ and $s_2 \rightarrow 10$. To be efficient we want the more probable symbols to have the shorter code and this can be measured by the average code length per symbol:

$$L = \sum_i l_i p_i$$

where l_i is the code length of symbol s_i . (In the example above $l_1 = 1, l_2 = 2$). An algorithm that achieves this is the Huffman coding algorithm.

The Huffman algorithm

1. Combine the two least probable symbols to one composite symbol the probability of this being the sum of the the probabilities of the two symbols. If there is more than one choice of the two least probable symbols then it may be made arbitrarily.
2. Continue this process until we have just two symbols and arbitrarily code one of them as 0 and the other 1.
3. Now trace back the process so that if say a symbol s comes from just one symbol t then the code of t is the same as the code for s . However if s came from t combined with u then we arbitrarily add 0 at the end of code for s to form the code for t and 1 at the end of s to form the code for u .

This is illustrated in examples below.

1. Table 1

s_1	0.4	→	0.4	→	0.4	→	0.4	→	0.4
s_2	0.3	→	0.3	→	0.3	→	0.3	→	0.6
s_3	0.1	→	0.1	→	0.1	→	0.3	↑	
s_4	0.1	→	0.1	→	0.2	↑			
s_5	0.06	→	0.1	↑					
s_6	0.04	↑							

Reversing the process:

s_1	0	←	0	←	0	←	0	←	0
s_2	10	←	10	←	10	←	10	←	1
s_3	110	←	110	←	110	←	11	↓	
s_4	1110	←	1110	←	111	↓			
s_5	11110	←	1111	↓					
s_6	11111	↓							

The average code length per symbol is

$$L = 1 \times 0.4 + 2 \times 0.3 + 3 \times 0.1 + 4 \times 0.1 + 5 \times 0.06 + 5 \times 0.04 = 2.2$$

2. A smaller example is the obvious case of two symbols with probabilities 0.9 and 0.1 which are coded as 0 and 1 respectively and the average code length is $L = 1 \times 0.9 + 1 \times 0.1 = 1$. To improve this we can use the *second extension* where *pairs* of symbols are coded. In this example the possible pairs are $s_1s_1, s_1s_2, s_2s_1, s_2s_2$ and the symbols are assumed to occur independently so that the probability of a pair is the product of the single probabilities. That is

$$p(s_i s_j) = p(s_i) p(s_j)$$

The second extension calculation is:

Table 2

s_1s_1	0.81	→	0.81	→	0.81
s_1s_2	0.09	→	0.09	→	0.19
s_2s_1	0.09	→	0.1	↑	
s_2s_2	0.01	↑			

Reversing the process

s_1s_1	0	←	0	←	0
s_1s_2	10	←	10	←	1
s_2s_1	110	←	11	↓	
s_2s_2	111	↓			

The average code length per *two symbols* sent is : $0.81 + 0.18 + 0.27 + 0.03 = 1.29$ whereas the average code length per symbol is $L = 1.29/2 = 0.645$ and the efficiency of the code has improved.

Clearly the extension process can be extended to coding triplets, quartets and in fact to strings of any length.

There are in general some theorems that measure this process. These involve the *entropy* of the source which is defined as

$$H(S) = - \sum_i p_i \log_2(p_i)$$

Note that the logarithm is to base 2 which is conventional in this work and

$$\log_2(x) = \frac{\ln(x)}{\ln(2)}$$

In the two examples above the entropies of the source are

$$H(S) = -(0.4\log_2(0.4) + 0.3\log_2(0.3) + 0.1\log_2(0.1) + 0.1\log_2(0.1) + 0.06\log_2(0.06) + 0.04\log_2(0.04)) = 2.14353$$

and

$$H(S) = -(0.9\log_2(0.9) + 0.1\log_2(0.1)) = 0.4689956$$

The important theorems are

1.

$$H(S) \leq L \leq H(S) + 1$$

The corresponding theorem for n-extensions is

2.

$$H(S) \leq \frac{L_n}{n} < H(S) + \frac{1}{n}$$

where L_n is the average code length of the n-th extension code.

This shows that $H(S)$ is the best average code length that can be attained and that

$$\lim_{n \rightarrow \infty} \frac{L_n}{n} = H(S)$$

but note that the theorem does not prove that there will always be an improvement for every higher extension, for some examples the code length is unchanged, but for large enough n we will have the improvement.

Proofs

To establish these theorems we may make use of the following result. Suppose we have two sets of n non-negative numbers $\{x_i\}$ and $\{y_i\}$ such that

$$\sum_{i=1}^n x_i = \sum_{i=1}^n y_i = 1, \quad x_i \geq 0, y_i \geq 0$$

then

$$\sum_{i=1}^n x_i (\log_2(y_i) - \log_2(x_i)) = \frac{1}{\ln(2)} \sum_{i=1}^n x_i \ln\left(\frac{y_i}{x_i}\right)$$

But we also have in general

$$\ln(x) \leq x - 1$$

so that

$$\sum_{i=1}^n x_i \ln\left(\frac{y_i}{x_i}\right) \leq \frac{1}{\ln(2)} \sum_{i=1}^n x_i \left(\frac{y_i}{x_i} - 1\right) = \frac{1}{\ln(2)} \sum_{i=1}^n (y_i - x_i) = 0$$

so that

$$-\sum_{i=1}^n x_i \log_2(x_i) \leq -\sum_{i=1}^n x_i \log_2(y_i)$$

We will use this identifying x_i with the probabilities p_i and y_i with

$$2^{-l_i}$$

where l_i are the code lengths. It is easy to see that at each stage of the reversal process in coding with the Huffman algorithm we have

$$\sum_i 2^{-l_i} = 1 \tag{1}$$

so that this definition satisfies the constraints of the theorem above. This may be established by induction. At the final stage the symbols are coded as 0 and 1 so that the code lengths are 1 and equation (1) holds. At each subsequent stage, if we assume that equation (1) holds, then at the next stage all the code lengths remain the same except one, l_j say, which is replaced by two codes each of length $l_j + 1$. Thus the *change* in equation 1 is

$$2 \frac{1}{2^{l_j+1}} - \frac{1}{2^{l_j}} = 0$$

We may conclude, by induction, that equation 1 holds at every stage including the final stage. With this interpretation

$$-\sum_i p_i \log_2 p_i = H(S) \leq -\sum_i p_i \log_2 2^{-l_i} = \sum_i p_i l_i \log_2(2) = \sum_i p_i l_i = L$$

and this proves that the entropy is the best code length per symbol that can be attained. Thus $l_i \approx -\log_2 p_i$ and if we choose l_i to be the unique integer in the range

$$-\log_2 p_i \leq l_i < -\log_2 p_i + 1$$

so that if we multiply by p_i and sum we obtain

$$H(S) \leq L < H(S) + \sum_i p_i = H(S) + 1$$

This is the first theorem above and is known as Shannon's first theorem. To establish the second theorem all we need to do is prove that the entropy of the n th extension is given by $nH(S)$ so that

$$nH(S) \leq L_n < nH(S) + 1$$

To illustrate that this is true consider a second extension entropy of the form

$$\begin{aligned} -\sum_{i,j} p_i p_j \log_2(p_i p_j) &= -\sum_{i,j} p_i p_j \log_2(p_i) - \sum_{i,j} p_i p_j \log_2(p_j) = \\ &H(S) \sum_j p_j + H(S) \sum_i p_i = 2H(S) \end{aligned}$$

The result can easily be extended for a general n extension.

Examples

1. Find a Huffman code for three symbols with associated probabilities 0.625, 0.25 and 0.125.
2. For the same system as in question 1, find the second extension and compare the code length per symbol. Verify that the entropy of the second extension is twice the entropy of the source.
3. Two symbols with associated probabilities 0.6 and 0.4 are coded as 0 and 1 respectively. Find the second extension Huffman code and show that the code length per symbol is unchanged. What is the code length per symbol for the 3rd extension?
4. In the proofs of the measure we use the result $\ln(x) \leq x - 1$. Explain why this is true?