Optimality of Huffman Codes

Theorem Huffman coding is optimal; that is, if C* is a Huffman code and C' is any other uniquely decodable code, L(C*) ≤ L(C').

This is true not only for binary alphabet, but also for a generic D-ary alphabet.

It is important to remember that there are many optimal codes: inverting all the bits or exchanging two codewords of the same length will give another optimal code. The Huffman procedure constructs one such optimal code.

Shannon-Fano-Elias Coding

We showed that the codeword lengths $l(x) = \lceil \log 1/p(x) \rceil$ satisfy the Kraft inequality and can therefore be used to construct a uniquely decodable code for the source.

Now we describe a simple constructive procedure that uses the cumulative distribution function to allot codewords.

Without loss of generality, we can take $X = \{1, 2, \ldots, m\}$. Assume that $p(x) > 0$ for all x. The cumulative distribution function $F(x)$ is defined as:

$$F(x) = \sum_{a \le x} p(a)$$

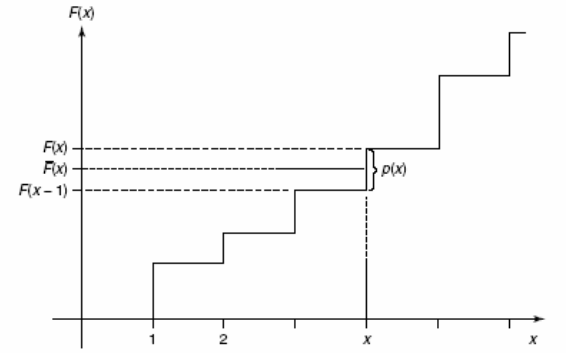
Consider the modified cumulative distribution function

$$\overline{F}(x) = \sum_{a < x} p(a) + \frac{1}{2} p(x)$$

Shannon-Fano-Elias Coding

Since the random variable is discrete, the cumulative distribution function consists of steps of size *p(x)*.

The value of the function *F(x)* is the midpoint of the step corresponding to *x*.



Shannon-Fano-Elias Coding

The value of $\overline{F}(x)$ can be used as a code for *x*. But, in general, *F(x)* is a real number expressible only by an infinite number of bits. So it is not efficient to use the exact value of $\overline{F}(x)$ as a code for *x*.

Assume that we truncate *F(x)* to *l(x)* bits (denoted by $\lfloor \overline{F}(x) \rfloor_{l(x)}$). Thus, we use the first *l(x)* bits of *F(x)* as a code for *x*. By definition of rounding off, we have

$$\overline{F}(x) - \lfloor \overline{F}(x) \rfloor_{l(x)} < \frac{1}{2^{l(x)}}$$

if $l(x) = \left\lceil \log \frac{1}{p(x)} \right\rceil + 1$

.

$$\frac{1}{2^{l(x)}} < \frac{p(x)}{2} = \overline{F}(x) - F(x-1)$$

Shannon-Fano-Elias Coding

Therefore, $\lfloor F(x) \rfloor_{l(x)}$ lies within the steo corresponding to x. Thus l(x) bits suffice to describe x.

In addition to requiring that the codeword identify the corresponding symbol, we also require the set of codewords to be prefix-free.

To check whether the code is prefix-free, we consider each codeword $z_1 z_2 \cdots z_l$ to represent not a point but the interval $[0.z_1 z_2 \cdots z_l , 0.z_1 z_2 \cdots z_l + 1/2^l]$.

The code is prefix-free if and only if the intervals corresponding to codewords are disjoint.

Shannon-Fano-Elias Coding

We now verify that the code above is prefix-free. The interval corresponding to any codeword has length $2^{-l(x)}$, which is less than half the height of the step corresponding to x.

The lower end of the interval is in the lower half of the step. Thus, the upper end of the interval lies below the top of the step, and the interval corresponding to any codeword lies entirely within the step corresponding to that symbol in the cumulative distribution function.

Therefore, the intervals corresponding to different codewords are disjoint and the code is prefix-free.

Note that this procedure does not require the symbols to be ordered in terms of probability.

Expected Length

Since we use $l(x) = \left\lceil \log \frac{1}{p(x)} \right\rceil + 1$ bits to represent x, the expected length of this code is

$$L = \sum_x p(x)\left(\left\lceil \log \frac{1}{p(x)} \right\rceil + 1\right) < H(X) + 2$$

Example 1

We first consider an example where all the probabilities are dyadic

x	$p(x)$	$F(x)$	$\overline{F}(x)$	$\overline{F}(x)$ in Binary	$l(x) = \left\lceil \log \frac{1}{p(x)} \right\rceil + 1$	Codeword
1	0.25	0.25	0.125	0.001	3	001
2	0.5	0.75	0.5	0.10	2	10
3	0.125	0.875	0.8125	0.1101	4	1101
4	0.125	1.0	0.9375	0.1111	4	1111

In this case, the average codeword length is 2.75 bits and the entropy is 1.75 bits. The Huffman code for this case achieves the entropy bound.

There is some inefficiency: for example, the last bit of the last two codewords can be omitted. But if we remove the last bit from all the codewords, the code is no longer prefix-free.

Construction of S-F-E Codes

In this case, since the distribution is not dyadic, the representation of *F(x)* in binary may have an infinite number of bits. We denote 0.01010101 . . . by $0.\overline{01}$.

x	$p(x)$	$F(x)$	$\overline{F}(x)$	$\overline{F}(x)$ in Binary	$l(x) = \left\lceil \log \frac{1}{p(x)} \right\rceil + 1$	Codeword
1	0.25	0.25	0.125	0.001	3	001
2	0.25	0.5	0.375	0.011	3	011
3	0.2	0.7	0.6	$0.1\overline{0011}$	4	1001
4	0.15	0.85	0.775	$0.110\overline{0011}$	4	1100
5	0.15	1.0	0.925	$0.111\overline{0110}$	4	1110

The above code is 1.2 bits longer on the average than the Huffman code for this source

Competitive Optimality

Let l(x) be the codeword lengths associated with the Shannon code, and let l(x) be the codeword lengths associated with any other uniquely decodable code. Then

$$\Pr(l(X) \geq l'(X) + c \leq \frac{1}{2^{c-1}}$$

For example, the probability that *l'(X)* is 5 or more bits shorter than *l(X)* is less than 1/16 .

Hence, no other code can do much better than the Shannon code most of the time.