Shannon-Fano Coding

September 18, 2017

One of the first attempts to attain optimal lossless compression assuming a probabilistic model of the data source was the Shannon-Fano code. It is possible to show that the coding is non-optimal, however, it is a starting point for the discussion of the optimal algorithms to follow.

Assume that each letter in the alphabet occurs according to a fixed probability distribution. In particular, assume that the probability of a letter is the same regardless of where it appears in the input string.

The algorithm is as follows. Let the probability distribution of the letters be $(p_1, \dots, p_n)$. If there is only one letter, then we are done, and we encode the letter using 0 bits. Otherwise, we divide the probability vector into two halves of roughly equal probabilities. If any of the halves contains exactly one probability component, then we encode that using 1 bit. Otherwise, we set the Most Significant Bit of one half to 1, and the other half to 0. We repeat the algorithm on each of the halves.

The second stage of the algorithm decides the second bit in each of the output encodings. Thus the leading bits in the four quarters have bits set to 00, 01, 10 and 11.

The process repeats until each of the subsets are singletons. We illustrate the process with an example.

Example 1. If the probabilities of the letters are

$$\{0.35, 0.17, 0.17, 0.16, 0.15\},$$

then the Shannon Fano coding proceeds as follows.

The first stage divides the set into $\{0.35, 0.17\}$ and $\{0.17, 0.16, 0.15\}$. Every string in the first set has an encoding starting with 0, and those in the second, with 1.

In the second stage, the subsets are $\{0.35\}$, $\{0.17\}$, $\{0.17\}$, and $\{0.16, 0.15\}$. The first subset is encoded as 00, the second as 01 and the third as 11. The elements of the final subset have their leading bits set to 10.

In the final stage, the two-element set in the previous stage is split as $\{0.16\}$ and $\{0.15\}$, encoded 100 and 101 respectively.

The expected code length is

$$0.35 * 2 + 0.17 * 2 + 0.17 * 2 + (0.16 + 0.15) * 3 = 2.31.$$

This allocation is not optimal.

The following encoding produces a shorter code on average.

0.35	0
0.17	100
0.17	101
0.16	110
0.15	111

This code has expected length 2.3. $\Diamond$