Tradeoffs Choosing Codeword Lengths

The Kraft-McMillan inequalities imply that to make some codewords shorter, we will have to make others longer.

Example: The obvious binary encoding for eight symbols uses codewords that are all three bits long. This code is instantaneous, and satisfies the Kraft inequality, since:

$$\frac{1}{2^3}+\frac{1}{2^3}+\frac{1}{2^3}+\frac{1}{2^3}+\frac{1}{2^3}+\frac{1}{2^3}+\frac{1}{2^3}+\frac{1}{2^3} \;=\; 1$$

Suppose we want to encode the first symbol using only two bits. We'll have to make some other codewords longer – eg, we can encode two of the other symbols in four bits, and the remaining five symbols in three bits, since

$$\frac{1}{2^2}+\frac{1}{2^4}+\frac{1}{2^4}+\frac{1}{2^3}+\frac{1}{2^3}+\frac{1}{2^3}+\frac{1}{2^3}+\frac{1}{2^3} \;=\; 1$$

How should we choose among the possible codes?

Formalizing Which Codes are the Best: Probabilities for Source Symbols

We'd like to choose a code that uses short codewords for common symbols and long ones for rare symbols.

To formalize this, we need to assign each symbol in the source alphabet a probability. Symbols $s_1, \ldots, s_q$ will have probabilities written as $p_1, \ldots, p_q$. We assume that these probabilities don't change with time.

We also assume that symbols in the source sequence, X_1, X_2, ..., X_n, are *independent*:

$$P(X_1 = s_{i_1} \;\&\; X_2 = s_{i_2} \;\&\; \cdots \;\&\; X_n = s_{i_n}) \\ = \; p_{i_1} p_{i_2} \cdots p_{i_n}$$

These assumptions are really too restrictive in practice, but we'll ignore that for now.

Average Codeword Length

We define the *average codeword length* of a code with codewords of lengths $l_1, \ldots, l_q$ for symbols $s_1, \ldots, s_q$, whose probabilities are $p_1, \ldots, p_q$, to be

$$L \;=\; \sum_{i=1}^{q} p_i l_i$$

This is the average length of the codeword encoding a single source symbol. But since averaging is a linear operation, the average length of a coded message with n source symbols is just nL. For example, when $n = 3$:

$$\sum_{i_1=1}^{q}\sum_{i_2=1}^{q}\sum_{i_3=1}^{q} p_{i_1} p_{i_2} p_{i_3} (l_{i_1} + l_{i_2} + l_{i_3}) \\ = \sum_{i_1=1}^{q} p_{i_1} l_{i_1} + \sum_{i_2=1}^{q} p_{i_2} l_{i_2} + \sum_{i_3=1}^{q} p_{i_3} l_{i_3} \;=\; 3L$$

We aim to choose a code for which L is small.

Optimal Codes

We say a code is *optimal* for a given source (with given symbol probabilities) if its average length is at least as small as that of any other code.

There can be many optimal codes for the same source, all with the same average length.

The Kraft-McMillan inequalities imply that if there is an optimal code, there is also an optimal *instantaneous* code. More generally, for any uniquely decodable code with average length L, there is an instantaneous code with the same average length.

Do Optimal Codes Always Exist?

It's not obvious that an optimal code will always exist.

There are infinitely many possible codes. Perhaps there could be an infinite sequence of codes with average lengths of, say

10.1, 10.01, 10.001, 10.0001, ...

If these codes were better than any others, then there would be no optimal code — just codes closer and closer to the limiting average length of 10.

Jones & Jones prove that there are always optimal codes. I'll bypass this and prove that we can always construct an optimal code — from this it follows that they always exist.

Ways to Improve Instantaneous Codes

Suppose we have an instantaneous code for $s_1, \ldots, s_q$, with probabilities $p_1, \ldots, p_q$. Let l_i be the length of the codeword for s_i.

Under each of the following conditions, we can find a better instantaneous code, with smaller average length:

- If $p_1 < p_2$ and $l_1 < l_2$:

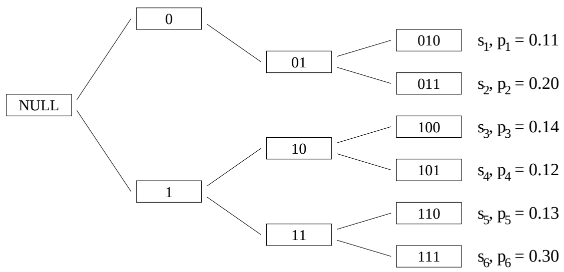
  Swap the codewords for s_1 and s_2.

- If there is a codeword of the form xby, with $x, y \in T^*$ and $b \in T$, but there are no codewords of the form $xb'z$, for $z \in T^*$ and $b' \in T$, with $b' \neq b$:

  Change all the codewords of the form xby to xy. (This will be an improvement if none of the p_i are zero, and doesn't make things worse in any case.)

The Improvements in Terms of Trees

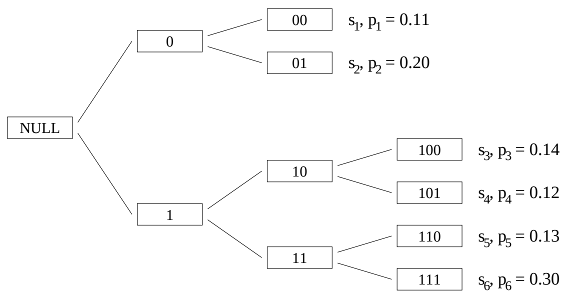
We can view these improvements in terms of the trees for the codes. Here's an example:



Two codewords have the form 01... but none have the form 00... (ie, there's only one branch out of the 0 node). We can therefore improve the code by deleting the surplus node.

Continuing to Improve the Example

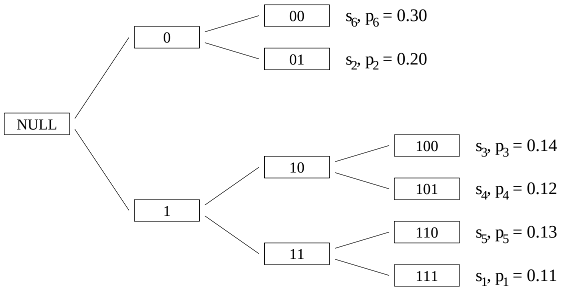
The result is the code shown below:



Now we note that s_6, with probability 0.30, has a longer codeword than s_1, which has probability 0.11. We can improve the code by swapping the codewords for these symbols.

The State After These Improvements

Here's the code after this improvement:



In general, after such improvements:

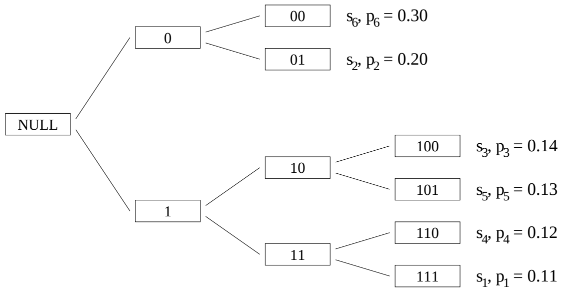
- The most improbable symbol will have a codeword of the longest length.
- There will be at least one other codeword of this length — otherwise the longest codeword would be a solitary branch.
- The second-most improbable symbol will also have a codeword of the longest length.

A Final Rearrangement

The codewords for the most improbable and second-most improbable symbols must have the same length. The most improbable symbol's codeword also has a "sibling" of the same length.

We can swap codewords to make this sibling be the codeword for the second-most improbable symbol.

For the example, the result is



Huffman Codes

We can use these insights about code trees to try to construct optimal codes.

We will prove later that the resulting *Huffman codes* are in fact optimal.

We'll concentrate on Huffman codes for a binary code alphabet. Non-binary Huffman codes are similar, but slightly messier.

Huffman Procedure for Binary Codes

Here's a recursive procedure to construct a Huffman code for a binary code alphabet:

procedure Huffman:

  inputs: symbols $s_1, \ldots, s_q$
          probabilities $p_1, \ldots, p_q$

  output: a code mapping $s_1, \ldots, s_q$ to codewords

  if $q = 2$:

    Return the code $s_1 \mapsto 0,\ s_2 \mapsto 1$.

  else

    Reorder $s_1, \ldots, s_q$ and $p_1, \ldots, p_q$ so that $p_1 \geq \cdots \geq p_q$.

    Create a new symbol s', with associated probability $p' = p_{q-1} + p_q$.

    Recursively call Huffman to find a code for $s_1, \ldots, s_{q-2}, s'$ with probabilities $p_1, \ldots, p_{q-2}, p'$. Let the codewords for $s_1, \ldots, s_{q-2}, s'$ in this code be $w_1, \ldots, w_{q-2}, w'$.

    Return the code
      $s_1 \mapsto w_1,\ \ldots,\ s_{q-2} \mapsto w_{q-2},\ s_{q-1} \mapsto w'0,\ s_q \mapsto w'1$.