Huffman Encoding

Here are some notes on Huffman's original paper on his famous codes. To keep this document more self contained, here is a rough sketch of Huffman's goal. Make sure you read this somewhere that can do Latex with markdown. Pandoc can compile it locally.

- We have N messages $(1, 2, 3, \ldots, N)$ to encode;
- Define $L(N)$ as the length of message N's code;
- Define $P(N)$ as the probability of message N occurring;
- Define the average message length as

$$\sum_{k=1}^{N} L(N)P(N)$$

- Huffman develops a code that minimizes this sum.

Notes on Derivation of Properties

Huffman derives various properties of an optimal encoding method mostly by contradiction. The statements are, "if a code is optimal, then it has property P." Assume that an optimal code does *not* have property P, then show that this makes the resulting code non-optimal somehow. Thus an optimal code must have property P. The other type is showing that an optimal code would be equivalent to some code that Huffman proposes. It uses the same argument, but doesn't end in a contradiction per se.

Ordering of Lengths

"For an optimum code, the length of a given message code can never be less than the length of a more probable message code. If this requirement were not met, then a reduction in average message length could be obtained by interchanging the codes."

Symbolically, if $P(k) > P(j)$, then $L(k) \leq L(j)$. Assume to the contrary that we have an optimal code where $P(k) > P(j)$ but $L(k) > L(j)$. Then, say that the average sum of this code is

$$avg_1 = S + P(k)L(k) + P(j)L(j),$$

where S is the sum of the average terms that do not include $P(k)$, $P(j)$, $L(k)$, or $L(j)$. Now, note that $P(k) - P(j) > 0$ and $L(k) - L(j) > 0$, so

$$(P(k) - P(j))(L(k) - L(j)) > 0.$$

Expanding, we get that

$$P(k)L(k) + P(j)L(j) - (P(k)L(j) + P(j)L(k)) > 0.$$

Adding the mixed term to both sides shows that

$$P(k)L(k) + P(j)L(j) > P(k)L(j) + P(j)L(k).$$

Construct an encoding method that swaps these two codes. Its average is

$$avg_2 = S + P(k)L(j) + P(j)L(k) < S + P(k)L(k) + P(j)L(j) = avg_1.$$

Therefore the original code is non-optimal, and so every optimal encoding method satisfies this property.

Later, "if there are several messages with the same probability, then ... the codes for these messages may be interchanged in any way without affecting the average code length."

Symbolically, if $P(k) = P(j) = P$, then $PL(k) + PL(j) = PL(j) + PL(k)$. This follows directly from the commutative law of addition.

Identical $L(N) - 1$ Prefixes

"[I]n an optimum code, it is necessary that at least two (and no more than D) of the codes with length $L(N)$ have identical prefixes of order $L(N) - 1$."

The "no more than D" part isn't immediately obvious from the argument. If we have messages of length $L(N)$ that have identical prefixes of order $L(N) - 1$, then the only thing different about them is their final digit. But we only have $D \geq 2$ such digits, so there can only be at most D such messages. Otherwise we would run out of digits to put into the final place.

All Possible Prefixes Must be Used

"[A]ll possible sequences of $L(N) - 1$ digits must be used either as message codes, or must have one of their prefixes used as message codes."

This one actually comes some something more general. Every combination of the D digits that is shorter than $L(N)$ must be used as a code. Symbolically, if we assume that $P(N) > 0$ and there is some unused code of length $l < L(N)$, then $lP(N) < L(N)P(N)$. This follows directly from $P(N) > 0$. So no such unused code can exist in an optimal code.

Generalization

Wikipedia says that "for n greater than 2, not all sets of source words can properly form an n-ary tree for Huffman coding." I don't think that's true. It may be true if at every level you must combine n messages into one, but Huffman states that it is only after the first step that n messages must be combined into one. So, expanding Huffman's argument, here is a proof that a Huffman code can be constructed for any reasonable set of messages with any reasonable number of digits.

Let $D \geq 2$ be the number of digits used for codes and $N \geq 2$ be the number of messages we want to encode. By "reasonable," we mean that $D, N \geq 2$. If $N = 1$, then that single message is already the root, so we only consider $N \geq 2$.

To prove that a D-ary Huffman tree is constructable, we must prove that there is a method to group messages into one root node. In each iteration of the algorithm, we must combine some number of messages together into one. At the first level, we may choose any number between two and D. After that, to meet a restriction on optimal codes, we must choose D messages. Joining D messages into one reduces the number of messages by $D - 1$. If N_1 is the number of messages after the first grouping, then to build the tree up to one root message, there has to exist some integer k such that $N_1 - k(D-1) = 1$. This is equivalent to

$$N_1 - 1 = k(D-1)$$

or

$$N_1 \equiv 1 \pmod{D-1}.$$

If we group n_0 messages in the first step, $N_1 = N - n_0 + 1$, so we must have that $(N - n_0 + 1) \equiv 1 \pmod{D-1}$, or

$$N \equiv n_0 \pmod{D-1}.$$

So to determine if we can construct a Huffman code for N messages, we must find an integer n_0 such that $N \equiv n_0 \pmod{D-1}$ where $2 \leq n_0 \leq D$. We will show that exactly one such an integer always exists. By the properties of integer division, there exists exactly one integer b such that $b \equiv (N-2) \pmod{D-1}$ and $0 \leq b \leq D-2$. From this we see that

$$(b+2) \equiv N \pmod{D-1}$$

and $2 \leq b \leq D$. Let $n_0 = b + 2$ and we get that n_0 exists for any combination of N and D. Note that $b = (N-2) \bmod (D-1)$, so

$$n_0 = 2 + (N-2) \bmod (D-1).$$

Now we will show that n_0 is unique. We have that $n_0 = 2 + b$. If any other integer n_0' has the same properties as n_0, then from the same integer division argument as for n_0, $n_0' = 2 + b = n_0$. Thus n_0 is a unique integer.

Therefore, given any number of digits and messages, we can always construct a Huffman tree with D nodes at each level, and $2 \leq n_0 \leq D$ nodes at the final level. This would be the description of a D-ary tree.

Edge Cases

There are a few edge cases that might give pause.

- $N = 1$
  - If $N = 1$, then the single message is already the root node.
- $D = 1$
  - We excluded this case for a good reason. First, we can't do integer division by $D - 1 = 0$. Next, a code in the Huffman system cannot be the prefix of another. If we only have 1 as a code, we have to give every message a string of ones. Is 111 one message, two, or three?
- $D = 2$
  - If $D = 2$, then $n_0 = 2 + (N - 2) \bmod 1 = 2$, and we recover the binary method.
- $2 \leq N \leq D$
  - If $2 \leq N \leq D$, then $0 \leq N-2 < D-1$, so $(N-2) \bmod (D-1) = N-2$. Thus $n_0 = 2 + N - 2 = N$.

Usefulness of non-binary codes

In general, the more digits allowed in a code, the less lower the overall average message length will be shorter[1]. But in the field of computer communication, this doesn't directly translate to shorter messages. Most computers are built around binary, meaning that any messages sent in some other base system will have to be encoded into binary. So cranking up the digits used might not be helpful all the time.

But if we could calculate the length of a base D number in binary, then we could take the average length of a message with D digits and maybe see what this length would be in binary. I'm thinking that if avg_D is the average length with D digits, and if $L(n_D)$ gives the length of a base D number in binary, then we could say the length in binary might be $L(\lceil avg_D \rceil)$. But I'm not sure. It's an interesting question.

[1] I'm not sure how to prove this, but I bet that someone could.