Garrett ©2004

quiz 04.1 Solution

(1) A source X emits words with probabilities as follows: there are 5 words with probability 1/10, 5 with probability 1/12, and 5 with probability 1/60. Estimate the optimal average word length achievable by any uniquely-decodable binary encoding of X.

One approach to this would be to compute the Huffman encoding and (which we know to be optimal) and from that compute the optimal average word length. But this is maybe too tedious. Instead, let's invoke the Noiseless Coding Theorem, which asserts that the average word length (precisely, the expected value of the random variable which measure word length) of an optimal binary (uniquely decodable) code is between $H(X)$ and $H(X)+1$, where $H(X)$ is the entropy of the source X. The entropy only depends upon the list of probabilities. Thus, the entropy of this source is (using the definition of entropy of a list of probabilities)

$$H(X) = -5 \cdot (1/10) \cdot \log_2(1/10) - 5 \cdot (1/12) \cdot \log_2(1/12) - 5 \cdot (1/60) \cdot \log_2(1/60)$$

$$\approx 3.64693930571154$$

Thus, we have the bound for average word length of optimal code

$$3.64693930571154 \leq \text{ optimal code avg word length } \leq 3.64693930571154 + 1$$

(2) Given a source X which emits source words $w_1, w_2, w_3, w_4, w_5, w_6$ with respective probabilities 1/4, 1/5, 1/6, 1/6, 1/6, 1/20, determine a Huffman encoding for this source.

It's hard to typeset the **tree** which would be the optimal representation of this computation, but I'll try to emulate the tree construction in words: Huffman encoding takes a list of probabilities and builds a **binary tree** by repeatedly taking the two smallest probabilities and creating a **parent** node for them, assigning the sum of their probabilities to that parent node. One of the two nodes is the **left child** and the other is the **right child**. (Here it doesn't matter which.) The process is repeated, each time looking at the list of nodes which have no parent. When the tree is complete, that is, when there is a single node (the **root node**) without a parent, the process goes back down the tree recursively to assign codewords, as follows: the root node is assigned the empty string '' as codeword. The left child of a node with codeword 'w' is assigned codeword 'w0' and the right child is assigned 'w1'. This process percolates down the tree, giving the Huffman encoding.

The two smallest probabilities are 1/20 and 1/6 which get combined to 13/60. List of probabilities is now (13/60, 1/6, 1/6, 1/5, 1/4). The two smallest probabilities are 1/6 and 1/6 which get combined to 1/3. List of probabilities is now (1/3, 1/5, 13/60, 1/4). The two smallest probabilities are 1/5 and 13/60 which get combined to 5/12. List of probabilities is now (5/12, 1/4, 1/3). The two smallest probabilities are 1/4 and 1/3 which get combined to 7/12. List of probabilities is now (7/12, 5/12). 5/12 child of 1='' gets codeword '0'. 1/5 child of 5/12='0' gets codeword '00'. 13/60 child of 5/12='0' gets codeword '01'. 1/20 child of 13/60='01' gets codeword '010'. 1/6 child of 13/60='01' gets codeword '011'. 7/12 child of 1='' gets codeword '1'. 1/4 child of 7/12='1' gets codeword '10'. 1/3 child of 7/12='1' gets codeword '11'. 1/6 child of 1/3='11' gets codeword '110'. 1/6 child of 1/3='11' gets codeword '111'. The list of probabilities and codewords is (1/4='10', 1/5='00', 1/6='011', 1/6='110', 1/6='111', 1/20='010').

(3) A three-word message is encoded by 'a'='1000011', 'b'='0100101', 'c'='0010110', 'd'='0001111', 'e'='1100110', and 'f'='1010101', 'g'='1001100'. The message is sent across a noisy channel, and you receive
'100101110101111000101'. Using minimum-distance decoding, what was the original message?

The message is to be construed as three 7-bit words in a row, each of which is a mangled form of one of the codewords. We are to decode each mangled 7-bit word by finding the codeword which differs from it by the least number of bits. That is, we are to find the codeword closest to the received word, as measured by the Hamming distance, which by definition counts the number of differing bits in two binary strings. Before embarking upon such a task, a person should note (by a one-time pre-computation!) that the codewords themselves have Hamming distances as little as 3 from each other. Thus, hoping for an unambiguous decoding, we'd insist upon codewords of Hamming distance just 0 or 1 from the received words.

Here is a simple (though not so efficient) algorithm for a decoding. For each received word, start by comparing it to all the codewords, from left to right. As soon as a codeword differs by 2 or more bits from the received word, drop that codeword from the list. Thus, in this case, 1001011 is closest (differing only at 3 bit) to 1000011 = 'a', 1010111 is closest (differing only at 5 bit) to 1010101 = 'f', 1000101 is closest (differing only at 2 bit) to 1010101 = 'f'. Thus, the decoding is **'aff'**.

(4) A source emits codewords 1000, 0011, 1110 with equal probabilities. What is the *rate* of this code? In a channel with bit error 1/12, what is the probability of an *undetected* error in a single word?

The rate is the log-base-2 of the number of codewords, divided by their length. Here, the rate is

$$\log_2(3)/4 \approx 0.396240625180289$$

For an error to be undetected, it must be that one of the codewords is mangled into another one of the codewords. The probability of this happening depends only upon the Hamming distances apart (which are 2, 3, 3) which is the number of bit flips necessary to mangle the one word into the other. Notice also the symmetry, that the probability of a word w_1 turning into w_2 is the same as the probability of w_2 turning into w_1, which gives rise to the factor of 2 in the formula just below, and correspondingly cuts down the number of summands there by a factor of 2. Also, the probability of each word being sent is 1/3, so the probability of undetected error is

$$2 \times \frac{1}{3}(P(1000 \text{ to } 0011) + P(1000 \text{ to } 1110) + P(0011 \text{ to } 1110))$$

$$= \frac{2}{3} \cdot (P(3 \text{ specific bits flipped}) + P(2 \text{ specific bits flipped}) + P(3 \text{ specific bits flipped}))$$

$$= \frac{2}{3}((1/12)^3(1-1/12) + (1/12)^2(1-1/12)^2 + (1/12)^3(1-1/12)) \approx 0.00459747942386831$$