CS 225: Assignment #2

Due on Thursday, Jaunary 28, 2016

Prof. Wim van Dam

Chad Spensky

1 Question 1

In this question we are given that $\sum_{i=1}^{m} 2^{-l_i} = 1 - \delta$, and asked to prove something interesting about $L(C) - H(p)$ in terms of δ. Trivially, we can say that $L(C) - H(p) > 0$, by the definition given in class and Kraft's. Note that δ is essentially dictating the depth of our binary tree, the greater δ is, the larger our lengths, and thus, the deeper our tree is. In fact, δ is the fraction of the tree, or "bottom line" of this tree/carpet, that we are **not** using, thus restricting our possible encodings. A similar proof the original, Kraft's proof, to show that the optimal encoding is produced by $l_i = -\log(\frac{p_i - \delta}{m})$.

We can re-write summation as

$$\sum_{i=1}^{m}(2^{-l_i} + \delta/m) = 1$$

and confirm that the equality still holds.

Thus, our difference is:

$$\begin{aligned}
L(C) - H(p) \geq \sum_{i=1}^{m}[p_i - \log(\frac{-p_i\delta}{m}) + p_i \log(p_i)] &= \sum_{i=1}^{m}[-p_i(\log(\frac{p_i - \delta}{m}) + \log(p_i))] \\
&= \sum_{i=1}^{m}[-p_i \log(\frac{p_i(p_i - \delta)}{m})] \\
&= E_p[-\log(\frac{p_i(p_i - \delta)}{m})] \\
&= H(\frac{p_i(p_i - \delta)}{m})
\end{aligned}$$

The best encoding, 2^{-l_i} must be at least $\frac{1-\delta}{m}$.

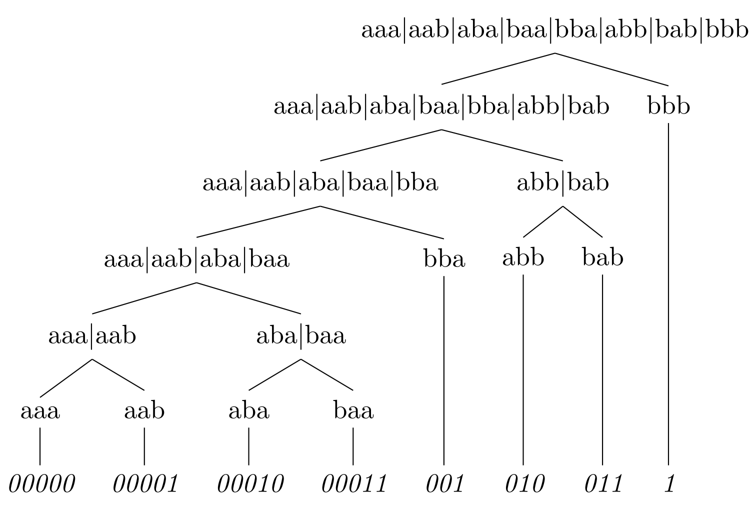
2 Question 2

This question asked us to find the Huffman Coding of X^3 where $\mathcal{X} = \{a, b\}, p(a) = .25, p(b) = .75$. First, we need to know the probability of every string happening. For example, $p(aaa) = p(a) * p(a) * p(a) = .25^3 = 0.015625$ and $p(aab) = p(a) * p(a) * p(b) = .25^2 * .75 = 0.046875$.

The table below outlines all of these probabilities:

$x \in \mathcal{X}^3$	$Pr(x)$
aaa	0.015625
aab	0.046875
aba	0.046875
baa	0.046875
abb	0.140625
bab	0.140625
bba	0.140625
bbb	0.421875
aaa|aab	0.0625
aba|baa	0.09375
aaa|aab|aba|baa	0.15625
abb|bab	0.28125
aaa|aab|aba|baa|bba	0.296875
aaa|aab|aba|baa|bba|abb|bab	0.578125

The tree of the Huffman code can be seen below:



The expected code length per letter is thus:

$$\begin{aligned}
L(C) &= \sum_{x \in \mathcal{X}} p(x) l(x) \\
&= 0.015625 * 5 + 3(0.046875 * 5) + 3(0.140625 * 3) + 0.421875 * 1 \\
&= 2.46875 \text{ bits}
\end{aligned}$$

3 Question 3

This question asks for the Shannon?Fano?Elias coding of the same data presented in Question 2. First, we must compute $\bar{F}(x), \forall x \in X^3$. Recall that:

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

We present the result for each element in X^3 below, with their respective binary representation.

$x \in \mathcal{X}^3$	**Pr(x)**	$\bar{F}(x)$	**binary**$(\bar{F}(x))$	$\lceil -\log_2 p(x) \rceil$	$c(x)$
aaa	0.015625	0.0078125	0.**000000**1	6	000000
aab	0.046875	0.0390625	0.**00001**01	5	00001
aba	0.046875	0.0859375	0.**00010**11	5	00010
baa	0.046875	0.1328125	0.**00100**01	5	00100
abb	0.140625	0.2265625	0.**001**1101	3	001
bab	0.140625	0.3671875	0.**010**1111	3	010
bba	0.140625	0.5078125	0.**100**0001	3	100
bbb	0.421875	0.7890625	0.**11**00101	2	11

Figure 1: Table outlining the SFE coding of the given alphabet.

The expected code length per letter is thus:

$$\begin{aligned} L(C) &= \sum_{x \in \mathcal{X}} p(x) l(x) \\ &= 0.015625 * 6 + 3(0.046875 * 5) + 3(0.140625 * 3) + 0.421875 * 2 \\ &= 2.90625 \text{ bits} \end{aligned}$$

4 Question 4

In this question we are asked to provide the tightest upper-bound of the Lempel-Ziv compression algorithm, i.e. given a string of length n, what is the maximum length of it's encoded bit string?

For an upper bound, we must assume the worse-case scenario for the encoding scheme. In this case, is that every possible combination of bits appears for a each given encoded length (i.e. both 1 and 0 are used to encode the patterns of length 1, $11, 10, 00, 01$ all appear for the patterns of length 2 that are encoded, etc.).

For a 2 bit string, the maximum would be 01 or (0,0) (0,1), the length of the indices would be 1 bit, thus the length of the LV string would be $2 * (1 + 1) = 4$ bits.

For a 4 bit string, a maximum would be 0100 or (0,0) (0,1) (0,0), the length of the indices would be 2 bit, thus the length of the LV string would be $3 * (2 + 1) =$ 9) bits.

We will now try to construct the worst-case string. As stated previously, it would ideally enumerate every possible encoded value at every length. Thus, we will construct the string using binary counting as follows: Note that the LZ

λ	0	1	00	01	10	11	000	001	010	011	100	101	110	111
0	1	0	1	2	3	0	1	2	3	4	5	6	7	8

dictionary words and encoding are then simply:

0_1	1_2	00_3	01_4	10_5	11_6	000_7	001_8	010_9	011_{10}	100_{11}	101_{12}	110_{13}	111_{14}
(0,0)	(0,1)	(1,0)	(1,1)	(2,0)	(2,0)	(3,0)	(3,1)	(4,0)	(4,1)	(5,0)	(5,1)	(6,0)	(6,1)

Note that the length of the string is 34 bits, the minimum length needed to encode the indices is $\lceil \log_2(6) \rceil = 3$, and the LV length is $14 * (3 + 1) = 56$.

Notice that in our worst case scenario, we are simply enumerating a complete binary tree, and to enumerate the number of nodes, i.e. LV dictionaries. The number of bits that can be encoded by a complete binary tree of depth d (we ignore depth 0, as this is implicit in LV) is:

$$\sum_{i=1}^{d} 2^{2d-1} = \sum_{i=1}^{d} 4 * 2^d * \frac{1}{2} = 2 * (2^{d+1} - 1) - 2^0 = 2^{d+2} - 2 = n$$

since the numbers of nodes at each depth is 2^d and the number of bits per node at that level is 2^{d-1}. Thus, the size of the tree, as a function of n is

$$d = \log_2(n + 2) - 2$$

the numbers of nodes in a binary tree, i.e. number of LV dictionaries, of depth d is $\sum_{i=1}^{d} 2^d = 2^{d+1} - 2$, thus the number of dictionaries is:

$$(2^{\log_2(n+2)-1} - 2) = \frac{1}{2}(n+2) - 2) = \frac{n}{2} - 1$$

Note that the indices at each level of the tree, all point to previous level, thus the maximum index value required is the number of nodes of a tree of depth $d-1$, thus when we multiply the number of bits required to encode this number, by the number of dictionaries, we get our bound:

$$\begin{aligned}
& (\log_2(n+2) - 2 - 2^{\log_2(n+2)-2}) * (\frac{n}{2} - 1) \\
= & (\log_2(n+2) - 2 - \frac{1}{4}(n+2)) * (\frac{n}{2} - 1) \\
= & (\log_2(n+2) - \frac{5}{2} - \frac{1}{4}n) * (\frac{n}{2} - 1)
\end{aligned}$$

Thus our worst-case is roughly bound by $O(n^2)$. This bound could be reduced by accounting for the fact not every dictionary on the bottom level of the tree will be utilized, however the overall boudn of n^2 would still apply.