Fall 2013

18.310 Homework 12 Solutions

Due Wednesday December 4th at 6PM

---

1. Suppose you are encoding a source that emits one of three letters: a with probability $\frac{1}{2}$, b with probability $\frac{1}{3}$ and c with probability $\frac{1}{6}$.

   (a) What is the Shannon bound on the best encoding of n letters from this source.
   Solution. The expected length per letter is at least

   $$H(p) = -\frac{1}{2}\log_2\frac{1}{2} - \frac{1}{3}\log_2\frac{1}{3} - \frac{1}{6}\log_2\frac{1}{6} = 1.4591\cdots$$

   (b) Use the Huffman algorithm to find an optimal prefix code for encoding this source. What is the number of bits used per letter?
   Solution. To get the optimal prefix encoding, we take the two least frequent letters, b and c, and replace them by a letter α with $p_\alpha = 0.5$. The optimum for this new alphabet is (for example) $a \to 0$ and $\alpha \to 1$, therefore we get as optimum for the original alphabet

   $$a \to 0b \to 10c \to 11$$

   The expected length per letter transmitted is

   $$1 \cdot \frac{1}{2} + 2 \cdot \frac{1}{3} + 2 \cdot \frac{1}{6} = 1.5.$$

2. Now, consider the same source as in problem (1), but the new 9-letter "alphabet" consisting of all pairs of letters, so aa would have probability $\frac{1}{4}$, ab would have probability $\frac{1}{6}$, etc.

   (a) What is the Shannon bound on the best encoding of n "letters" from this source.
   Solution. The entropy of this new probability distribution p' on $A' = A \times A$ (where $A = \{a, b, c\}$) is

   $$\begin{aligned} H(p') &= -\sum (i,j) \in A \times A p_i p_j \log_2(p_i p_j) \\ &= -\sum_{i \in A} p_i \log_2(p_i) - \sum_{j \in A} p_j \log_2(p_j) \\ &= 2H(p) = 2.9182\cdots. \end{aligned}$$

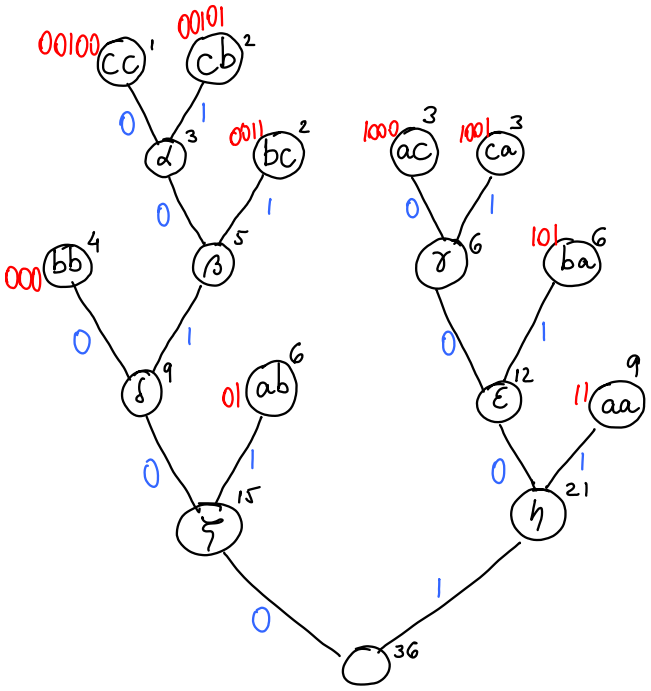
   This was expected by Shannon's theorem as generating independently $n/2$ 'letters' from A' (with the associated probabilities) gives the same probabilistic source as generating n letters from A. Shannon's bound for n letters is thus $2.9182n + o(n)$.

(b) Use the Huffman algorithm to find an optimal prefix code for encoding this source. What is the number of bits used per "letter'? Per letter of the original source?

Solution. Our new alphabet A' and associated probabilities (in non-increasing order) are:

aa	$1/4 =$	$9/36$
ab	$1/6 =$	$6/36$
ba	$1/6 =$	$6/36$
bb	$1/9 =$	$4/36$
ac	$1/12 =$	$3/36$
ca	$1/12 =$	$3/36$
bc	$1/18 =$	$2/36$
cb	$1/18 =$	$2/36$
cc	$1/36 =$	$1/36$

The best prefix code is given below (drawn upside down, as it was built by repeatedly combining the two least likely letters).



The Huffman code (with probabilities shown as well) is (there are several optimal ones):

aa	$9/36$	11
ab	$6/36$	01
ba	$6/36$	101
bb	$4/36$	000
ac	$3/36$	1000
ca	$3/36$	1001
bc	$2/36$	0011
cb	$2/36$	00101
cc	$1/36$	00100

This gives an encoding with an expected number of bits per 'letter' of A' of

$$\frac{1}{36}(9\cdot 2+6\cdot 2+6\cdot 3+4\cdot 3+3\cdot 4+3\cdot 4+2\cdot 4+2\cdot 5+1\cdot 5)=\frac{107}{36}=2.9722\cdots,$$

or 1.4861 per letter of A.

3. Level-Ziv encoding will be covered on Monday.

   (a) Suppose you encode n digits from the sequence

   $$12345678910111213141516171819202122\cdots$$

   obtained by concatenating all natural numbers. Approximately how many bits will this take to encode using Lempel-Ziv? By approximately, we mean that we care only about the asymptotic growth as n gets large.

   Solution. Each new number will become a new dictionary phrase. So, the i^{th} phrase will require $4+\lceil\log_2 i\rceil$ bits (since a decimal digit requires 4 bits) in its encoding. Now, in the first n digits, there are approximately $O(\frac{n}{\log_{10} n})$ phrases. So, the total number of bits required is approximately

   $$\begin{aligned}&\sum_{i=1}^{\frac{n}{\log_{10} n}} 4+\lceil\log_2 i\rceil\\ &\leq \sum_{i=1}^{\frac{n}{\log_{10} n}} 5+\log_2 i\\ &\leq \frac{5n}{\log_{10} n}+\log_2\left[\left(\frac{n}{\log_{10} n}\right)!\right]\end{aligned}$$

   Using big-O notation and Sterling's approximation, the last line comes out to be about $O(\frac{n}{\log_{10} n}\log_2\frac{n}{\log_{10} n})$, which is equivalent to $O(n)$ (roughly a constant times n), so this string does not compress very well using Lempel-Ziv.

   (b) Suppose you encode n bits from the sequence

   $$01010101010101010101\cdots$$

obtained by alternating 0's and 1's. Approximately how many bits will this take to encode using Lempel-Ziv?

Solution. Breaking up the first few phrases, we see the pattern:

0
1
01
010
10
101
0101
01010
1010
10101.

So, the i^{th} phrase takes approximately $\frac{i}{2}$ digits (this could be proved formally by induction). Therefore, the first n digits contains about $2\sqrt{n}$ phrases. Hence, the total number of bits required is about

$$\sum_{i=1}^{2\sqrt{n}} 1 + \lceil \log_2 i \rceil$$

Using the same analysis as above, we see that this is bounded above by $4\sqrt{n}+\log_2(2\sqrt{n})!$ and below by $2\sqrt{n} + \log_2(2\sqrt{n})!$, so in big-O notation, it is $O(\sqrt{n}\log n)$, i.e., roughly a constant times $\sqrt{n}\log n$.

18.310 Principles of Discrete Applied Mathematics
Fall 2013