CSC 310, Fall 2011 — Solutions to Theory Assignment #2

Question 1 (50 marks): Suppose that the designer of a data compression system uses a model for a source of binary symbols, X_1, X_2, ..., X_n, in which these symbols are independent given the probability, p_1, of symbol 1. (Of course, the probability of symbol 0 is $p_0 = 1 - p_1$.) The designer doesn't know p_1, but uses a prior model in which p_1 is either 0, 1, or 1/2, with $P(p_1 = 0) = P(p_1 = 1) = 1/4$ and $P(p_1 = 1/2) = 1/2$.

Consider two ways of encoding such a sequence of length n. (We assume that n is fixed, and known to both the encoder and decoder.) These are:

> **Method A:** Estimate p_1 by examining the whole sequence of n symbols (with the estimate being either 0, 1, or 1/2), then transmit the estimated p_1 using an optimal code based on the prior probabilities, and finally transmit X_1, ..., X_n using an optimal code based on p_1 (and the assumption that the symbols are independent).
>
> **Method B:** Transmit each of X_1, ..., X_n using a method such as arithmetic coding that compresses nearly down to the entropy, basing the transmission of X_i on the predictive distribution of X_i given X_1, ..., X_{i-1}. These predictive probabilities are found by summing the predictive probabilties based on each possible value of p_1 times the posterior probability of that value of p_1 given X_1, ..., X_{i-1} (which is the prior probability when $i = 1$).

Answer the following, justifying your answers:

a) When using Method A, what is the best way to estimate p_1 — that is, the way that results in the smallest number of bits being transmitted on average?

If X_1, ..., X_n consists of some 0s and some 1s, we must choose $p_1 = 1/2$, since otherwise sending some of the X_i symbols will take an infinite number of bits, since they will have probability 0.

If X_1, ..., X_n are all 1s, then we can't use $p_1 = 0$, but we could use either $p_1 = 1$ or $p_1 = 1/2$. Encoding X_1, ..., X_n takes n bits if we use $p_1 = 1/2$, but it takes 0 bits if we use $p_1 = 1$, since then $\log_2(1/p_1) = 0$. To this, we have to add the number of bits used to send p_1. The optimal code for p_1 encodes $p_1 = 1/2$ as one bit (say 0) and $p_1 = 0$ and $p_1 = 1$ as two bits (say 10 and 11). The total number of bits sent if we choose $p_1 = 1/2$ is therefore $n + 1$, whereas the total number of bits sent if we choose to use $p_1 = 1$ is 2. It is therefore better to use $p_1 = 1$ when all the X_i are 1 and n is greater than 1. When n is equal to 1, $p_1 = 1$ and $p_1 = 1/2$ are equally good choices.

When X_1, ..., X_n are all 0s, the result is reversed, with $p_1 = 0$ being best, unless $n = 1$, in which case $p_1 = 1/2$ is equally good.

b) When using Method A, and the estimation method from part (a) above, how many bits (total) are transmitted when $X_1, \ldots, X_n$ has n_0 0s and n_1 1s (with $n_0 + n_1 = n$)?

As implied by the answer to part (a), the total number of bits is 2 if $n_0 = 0$ or $n_1 = 0$, and otherwise is $n + 1$.

c) Derive a simple way of finding the predictive distribution for X_i given $X_1, \ldots, X_{i-1}$ that is needed to implement Method B.

The predictive distribution for X_1 is $P(X_1 = 1) = 1/2$, due to the symmetry of the prior.

For $i > 1$, the posterior distribution of p_1 given $X_1, \ldots, X_{i-1}$ will be $P(p_1 = 0) = 0$, $P(p_1 = 1) = 0$, and $P(p_1 = 1/2) = 1$ if both 0 and 1 appear at least once in $X_1, \ldots, X_{i-1}$. If all of $X_1, \ldots, X_{i-1}$ are 1s, then $P(p_1 = 0) = 0$, and from the odds form of Bayes' Rule, we get

$$\begin{aligned} \frac{P(p_1 = 1)}{P(p_1 = 1/2)} &= \frac{P(X_1 = 1, \ldots, X_{i-1} = 1 \mid p_1 = 1)}{P(X_1 = 1, \ldots, X_{i-1} = 1 \mid p_1 = 1/2)} \, \frac{P(p_1 = 1)}{P(p_1 = 1/2)} \\ &= \frac{1 \times (1/4)}{(1/2)^{i-1} \times (1/2)} = 2^{i-2} \end{aligned}$$

From this we get that $P(p_1 = 1/2) = 1/(1+2^{i-1})$ and $P(p_1 = 1) = 2^{i-1}/(1+2^{i-1})$. The predictive distribution for X_i is therefore

$$P(X_i = 1 \mid X_1 = 1, \ldots, X_{i-1} = 1) = (1/2) \frac{1}{1+2^{i-2}} + (1) \frac{2^{i-2}}{1+2^{i-2}} = \frac{1}{2} \frac{1+2^{i-1}}{1+2^{i-2}}$$

Similarly,

$$P(X_i = 0 \mid X_1 = 0, \ldots, X_{i-1} = 0) = \frac{1}{2} \frac{1+2^{i-1}}{1+2^{i-2}}$$

d) How many bits are transmitted using Method B when $X_1, \ldots, X_n$ are all symbol 1? Express your answer in as simple a form as possible. In this and later parts of this question, assume that with the encoding method used, the number of bits that are transmitted as a result of encoding a symbol having probability p is $\log_2(1/p)$ (which need not be an integer).

Encoding X_1 takes 1 bit. Adding to this the number of bits for each following 1 symbol, using the predictive probabilities from part (c), we get that the total number of bits is

$$\begin{aligned} 1 + \sum_{i=2}^{n} \log_2 \left[2 \frac{1+2^{i-2}}{1+2^{i-1}} \right] &= n + \log_2 \left[\prod_{i=2}^{n} \frac{1+2^{i-2}}{1+2^{i-1}} \right] \\ &= n + \log_2 \left[\frac{2}{1+2^{n-1}} \right] \\ &= n + 1 + \log_2 \left[\frac{2^{-(n-1)}}{1+2^{-(n-1)}} \right] \\ &= 2 - \log_2(1+2^{-(n-1)}) \end{aligned}$$

e) How many bits are transmitted using Method B when $X_1, \ldots, X_{n-1}$ are all symbol 1 and X_n is symbol 0? Express your answer in as simple a form as possible.

(Assumes $n \geq 2$, as I intended.)

From part(d), we see that the number of bits to encode the first $n-1$ symbols, all 1s, will be

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

Using the predictive probability from part(c), we see that the number of bits to encode the final 0 symbol will be

$$\begin{aligned} -\log_2\left[1 - \frac{1}{2}\,\frac{1+2^{n-1}}{1+2^{n-2}}\right] &= -\log_2\left[\frac{1}{2}\,\frac{1}{1+2^{n-2}}\right] \\ &= 1 + \log_2(1+2^{n-2}) \\ &= n - 1 + \log_2(1+2^{-(n-2)}) \end{aligned}$$

Adding these gives a total of $n+1$ bits.

f) How many bits are transmitted using Method B when X_1 is symbol 0 and $X_2, \ldots, X_n$ are all symbol 1? Express your answer in as simple a form as possible.

The first 0 symbol takes 1 bit to encode. The predictive probability that the second symbol is 1 given that the first is 0, from part (c), is 1/4, so the second symbol takes 2 bits to encode. The remaining $n-2$ symbols will be encoded with predictive probabilities of 1/2, since at this point only $p_1 = 1/2$ is possible, and hence will take 1 bit each. The total number of bits is therefore $n+1$.

(This is of course what is expected — since the distribution of $X_1, \ldots, X_n$ is exchangeable, the answers to parts (e) and (f) should be the same.)

g) Discuss which of Method A and Method B is better in terms of compression.

Using exchangeability, and proceeding as in part (f), we we see that Method B will encode any sequence with at least one 0 and at least one 1 in $n+1$ bits. From part (b), we see that this is the same as for Method A, so there is no difference between the methods in this case.

When the sequence is all 0s or all 1s, we see from part (b) that Method A takes 2 bits, but from part (d) we see that Method B takes $2 - \log_2(1+2^{-(n-1)})$ bits. So Method B is slightly better.

This inefficiency of Method A can be explained by the fact that it does not use all possible output sequences. It will never say that $p_1 = 1/2$ and then send a sequence of all 0s or all 1s. This is a general deficiency of methods that first estimate probabilities, then send them, and then send the message using a code based on these fixed probabilities.

Question 2 (50 marks): Recall that for each context the PPM method keeps track of which symbols have been seen before, along with the count of how many times each such symbol has been seen, and transmits a symbol that has been seen before using a probability for it that is proportional to its count. A fixed count of 1 is allocated to an "escape" symbol, that is used when a symbol not seen before needs to be transmitted, using a lower-order context. In this question, we will consider using this scheme when the maximum order is 0, so that symbols are modeled as being independent (not depending on any preceding symbols). In this case, the order 0 context will contain all symbols that have been seen before, and the escape from this context is to the order -1 context, in which all symbols have fixed counts 1, and hence are all equally likely.

Assume that an encoding method such as arithmetic coding is used, so that the number of bits that are transmitted as a result of encoding a symbol having probability p is $\log_2(1/p)$ (which need not be an integer).

a) Describe two deficiencies in this scheme that lead to a code tree in which some nodes have only one child — equivalently, these deficiencies result in the arithmetic coding interval from 0 to 1 having portions that can never contain the final transmitted message.

First, a count of 1 is allocated to the "escape" symbol even in contexts (other than the order -1 context) where all symbols occur, and where "escape" will therefore never be used.

Second, when we escape to a lower-order context to encode a symbol, we use the counts in that context, possibly including non-zero counts for symbols that appear in the higher-order contexts we escaped from. These symbols are not possible for this encoding operation, since if such a symbol were what is to be encoded, it would have been encoded in a higher-order context.

b) Show that this PPM scheme of maximum order 0 produces exchangeable codes — that is, show that the number of bits in which a message is encoded remains the same if the order of symbols in the message is permuted.

Every symbol that occurs in the message will be encoded the first time it occurs, say at position i (counting from 1), by sending "escape" and then the symbol encoded with equal probabilities from the order -1 context. The first occurrence of a symbol at position i therefore takes $\log_2(1+i-1) = \log_2(i)$ bits for the "escape" plus $\log_2(I)$ bits for the symbol. At a later occurrence of the symbol, say at position j, the number of bits to encode it will be $\log_2((1+j-1)/c) = \log_2(j) - \log_2(c)$, where c is the count of how many times this symbol has occurred in positions before j.

We therefore see that whatever is sent at position i, a term of $\log_2(i)$ is added to the total number of bits sent. Furthermore, for every symbol that occurs $C > 0$ times in the message, a term of $\log_2(I)$ bits is added to the total, and the total is decreased by

$$\sum_{c=1}^{C-1} \log_2(c)$$

None of these terms depends on the order of symbols — only on the counts of how many times they occur — so the code is exchangeable.

c) Consider a modified scheme in which the count for the escape symbol is not fixed at 1, but instead is equal to $(I-m)/I$, where I is the size of the alphabet and m is the number of distinct symbols seen previously (initially 0), and also, after a symbol is encoded in the order -1 context, its count in the order -1 context is decreased to 0 (so that the probability in the order -1 context of a symbol not previously seen is $1/(I-m)$, rather than $1/I$). Show that this modified scheme never encodes a message in more bits than the original scheme, and characterize the set of messages that the modified scheme encodes in fewer bits than the original scheme.

When sending "escape" and a new symbol, at position i, the original scheme uses $\log_2(1+i-1) = \log_2(i)$ bits for the escape, and $\log_2(I)$ bits for the symbol in the order -1 context, for a total of $\log_2(i) + \log_2(I) = \log_2(iI)$ bits. The modified scheme uses

$$\log_2 \left[\frac{(I-m)/I + i - 1}{(I-m)/I} \right]$$

bits for the escape and $\log_2(I-m)$ bits for the symbol in the order -1 context, so the total number of bits sent is

$$\log_2 \left[\frac{(I-m)/I + i - 1}{(I-m)/I} \right] \; + \; \log_2(I-m) \; = \; \log_2(iI - m)$$

which is the same as the number for the original scheme when $m = 0$, and is less when $m > 0$.

When sending a symbol that has been seen before, at position i, the original scheme uses $\log_2((1 + i - 1)/c) = \log_2(i/c)$ bits, where c is the number of times the symbol has been seen before, whereas the modified scheme uses $\log_2(((I - m)/I + i - 1)/c) = \log_2((i - m/I)/c)$ bits, which is less than for the original scheme.

d) Show that the modified scheme of part (c) does *not* produce an exchangeable code.

Here is an example demonstrating this:

Using the results from part (c), we see that the number of bits used by the modified scheme to encode AAB,

$$\log_2(I) \; + \; \log_2((2 - 1/I)/1) \; + \; \log_2(3I - 1)$$

whereas the number of bits to encode ABA is

$$\log_2(I) \; + \; \log_2(2I - 1) \; + \; \log_2((3 - 2/I)/1)$$

Considering the case where $I = 2$, the first expression above is $\log_2(2) + \log_2(3/2) + \log_2(5) = \log_2(15)$ whereas the second is $\log_2(2) + \log_2(3) + \log_2(2) = \log_2(12)$.

e) [5 Mark Bonus] Find a further modification of the modified scheme that does produce an exchangeable code, while still retaining whatever you consider to be its advantages. Do you think being exchangeable makes it better?

We get an exchangeable method if we add $1/I$ to the count of every symbol in the order 0 context, while keeping the count for the "escape" at $(I - m)/I$. The total count when encoding at position i is then $(I - m)/I + (i - 1) + m(1/I) = i$, which since it doesn't depend on which symbols came before, will lead to the method being exchangeable.

This modification retains the advantage that the escape probability goes to zero when there is no more need for it (since all symbols have been seen). Exchangeability ought to be good when the symbols to be encoded actually are independent given the symbol probabilities. However, this is rarely the case. It's conceivable that some form of non-exchangeability could actually be desirable when the symbols aren't independent, but I think you have to be rather optimistic to hope for this when the non-exchangeability is "accidental" rather than being designed to model your source of non-independent symbols.