EE376A - Information Theory
Midterm, Tuesday February 9th Solutions

Instructions:

- You have **two hours**, 6:30PM - 8:30PM
- The exam has 2 questions, totaling 100 points. There is an extra credit question for additional 20 points.
- Please start answering each question on a new page of the answer booklet.
- You are allowed to carry the textbook, your own notes and other course related material with you. Electronic reading devices [including kindles, laptops, ipads, etc.] are allowed, provided they are used solely for reading pdf files already stored on them and not for any other form of communication or information retrieval.
- You are required to provide a detailed explanation of how you arrived at your answers.
- You can use previous parts of a problem even if you did not solve them.
- As throughout the course, entropy (H) and Mutual Information (I) are specified in bits.
- log is taken in base 2.
- Throughout the exam 'prefix code' refers to a variable length code satisfying the prefix condition.
- Good Luck!

1. Coin Toss Experiment and Golomb Codes *(50 points)*

Kedar, Mikel and Naroa have been instructed to record the outcomes of a coin toss experiment. Consider the coin toss experiment $X_1, X_2, X_3, ...$ where X_i are i.i.d. $Bern(p)$ (probability of a H (head) is p), $p = 15/16$.

(a) (*5 points*) Kedar decides to use Huffman coding to represent the outcome of each coin toss separately. What is the resulting scheme? What compression rate does it achieve?

(b) (*10 points*) Mikel suggests he can do a better job by applying Huffman coding on blocks of r tosses. Will his scheme approach the optimum expected number of bits per description of source symbol (coin toss outcome) with increasing r? How does the space required to represent the codebook increase as we increase r?

(c) (*20 points*) Naroa suggests that, as the occurrence of T is so rare, we should just record the number of tosses it takes for each T to occur.
To be precise, if Y_k represents the number of trials until the k^{th} T occurred (inclusive), then Naroa records the sequence:

$$Z_k = Y_k - Y_{k-1}, \; k \geq 1, \tag{1}$$

where $Y_0 = 0$

i. What is the distribution of Z_k, i.e., $P(Z_k = j), j \in 1, 2, 3, ...$?
ii. Compute the entropy and expectation of Z_k.
iii. How does the ratio between the entropy and the expectation of Z_k compare to the entropy of X_k? Give an operational interpretation.

(d) (*15 points*) Consider the following scheme for encoding Z_k, which is a specific case of Golomb Coding. We are showing the first 10 codewords.

Z	Quotient	Remainder	Code
1	0	1	1 01
2	0	2	1 10
3	0	3	1 11
4	1	0	0 1 00
5	1	1	0 1 01
6	1	2	0 1 10
7	1	3	0 1 11
8	2	0	00 1 00
9	2	1	00 1 01
10	2	2	00 1 10

i. Can you guess the general coding scheme [Hint: We are considering quotient and remainder of Z with respect to 4 in this case, and concatenating them in a specific format to form the code for Z]?
ii. What is the decoding rule for this Golomb code?

iii. How can you efficiently encode and decode with this codebook?

Solution:

(a) As we have just 2 source symbols (H & T), the optimal Huffman code is $\{0,1\}$ for $\{H,T\}$. The compression rate (expected codelength) achieved is 1.

(b) We know that, using Shannon codes for extended codewords of length r, the expected number of bits per source symbol is:

$$H(X) \leq \ell_{avg} \leq H(X) + \frac{1}{r} \tag{2}$$

As Huffman coding is the optimal prefix code, thus it will always perform at least as good as the Shannon codes. Thus, as r increases, we will achieve optimal average codelength of $H(X) = H_b(1/16)$. As r increases, the number of codewords increases as 2^r. Also, the average size of the codewords, increases as $rH(X)$. Thus, effectively, the amount of space required to store the codebook increases exponentially in r.

(c) We observe that $Z_1, Z_2, ..., Z_k$, are i.i.d. (since the coin tosses are independent).

i. Z_k has geometric distribution. Specifically $P(Z_k = j) = p^{j-1}(1-p), j \in \{1,2,3,...\}$. To see this, lets consider Z_1 for ease first. For $Z_1 = j$, we should have the first $j-1$ tosses to be H, while the j^{th} toss should be a T. This results in: $P(Z_k = j) = p^{j-1}(1-p)$. Due to Z_k being i.i.d., we can generalize the results to Z_k.

ii. $H(Z_k) = \frac{H_b(p)}{(1-p)}$ (This is a simple symbol manipulation exercise).
As Z_k has geometric distribution, $E[Z_k] = \frac{1}{(1-p)}$.

iii. We observe that $H(X) = \frac{H(Z_k)}{E[Z_k]}$. Operationally, this implies that Naroa's scheme will achieve the entropy rate of the source if the Z sequence is optimally compressed. More specifically, $H(Z)$ bits will be needed per description of each component of the source Z, and each such component will on average account for $E[Z]$ symbols of the source X, so overall the scheme will require $H(Z)/E[Z]$ bits of description per representation of a symbol from the source X.

(d) As an aside, the aim of the question was to introduce Golomb Codes. Golomb codes are in fact optimal for sequences with geometric distribution, and are very simple to construct. We have presented a specific case of Golomb codes in our example.

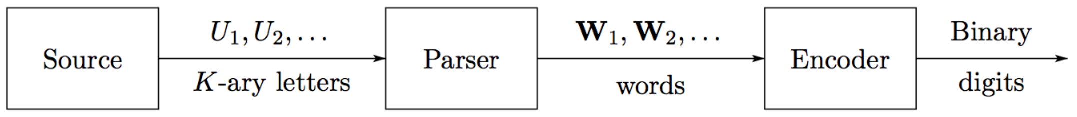
i. We are considering quotient of Z with respect to 4 in the unary form, and remainder in the binary form and concatenating them using a separator bit (the bit 1 in this case).

ii. It is easy to show that the code is prefix-free. Thus, the decoding can proceed by using the standard techniques of constructing a prefix-tree and using the tree for decoding. However, note that in this case, the codebook size itself is unbounded, thus even though this is a valid way of decoding, it is certainly not an efficient one (unless you find a different way of representing the prefix-tree). (Note: It is good if you directly come up with the efficient scheme, rather than the inefficient one stated above)

iii. There is, however, a better and a simpler way of decoding, which does not require us to store the codebook explicitly. We can parse the symbol stream until we get the bit 1, this gives us the quotient q. Now, we read the next 2 bits, which gives us the remainder r. Using q and r, we can decode $Z = 4 * q + r$. The good thing about Golomb codes is that, even though the codebook size is unbounded, we do not need to explicitly store the codebook, but can have a simple code which generates the codes. It is also easy to see that we can use Golomb codes along with Naroa's scheme to represent the outcomes of the original coin toss experiment.

2. Tunstall Codes *(50 points)*

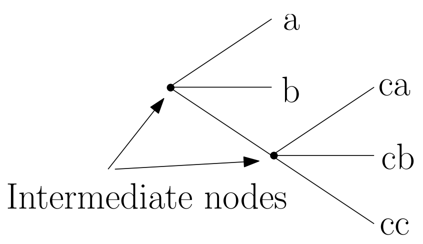
Consider an approach for compression of a memoryless K-ary source $U_1, U_2, \ldots$ distributed according to the discrete random variable U, with $|\mathcal{U}| = K$, illustrated in the following figure:



Instead of encoding the source symbols directly, we first parse them into words with the help of a dictionary of size M. A dictionary is a prefix-free collection of words $\boldsymbol{w}_1, \ldots, \boldsymbol{w}_M$, where each $\boldsymbol{w}_m$ is a sequence of source letters, with the property that any stream of source symbols can be constructed as a concatenation of words from the dictionary. Each dictionary entry is then represented by a binary codeword of the same constant length b. These binary representations of the dictionary words that make up the source stream comprise the encoder output.

Given a dictionary with words $\boldsymbol{w}_1, \ldots, \boldsymbol{w}_M$, the parser picks the longest prefix of the source sequence $U_1, U_2, \ldots$ that is in the dictionary (say, $U_1, \ldots, U_L$), and then continues in the same fashion with $U_{L+1}, U_{L+2}, \ldots$, etc.

Consider, for example, $\mathcal{U} = \{a, b, c\}$, and $\boldsymbol{w}_1 =$"a", $\boldsymbol{w}_2 =$"b", $\boldsymbol{w}_3 =$"ca", $\boldsymbol{w}_4 =$"cb", $\boldsymbol{w}_5 =$"cc". The tree associated with this dictionary is given by



(a) (*10 points*) Given a dictionary represented by binary codewords of fixed length b, what should the words of the dictionary satisfy such that the overall compression scheme is optimal (i.e., it achieves the entropy)? Specifically, what should M satisfy, and what should be the probabilities of the words that comprise the dictionary?

Consider the following Tunstall code, for $\mathcal{U} = \{a, b, c\}$, and $P_U(a) = 0.6$, $P_U(b) = 0.3$, and $P_U(c) = 0.1$:

Word	Codeword
b	000
c	001
ab	010
ac	011
aaa	100
aab	101
aac	110

(b) (*6 points*) Compute the entropy of the source $H(U)$.

(c) (*6 points*) Compute the entropy of the dictionary words, denoted by $H(W)$.

(d) (*8 points*) Compute the expected length of the dictionary words, denoted by $E[L_W]$.

(e) (*8 points*) What is the overall compression rate, i.e., the average number of bits of description per source symbol under this compression scheme? Compare this with the compression achieved if the dictionary words were given by "a", "'b" and "c", and the associated binary codewords were of length 2. Which scheme is better?

(f) (*12 points*) Given the dictionary in the above table, can you come up with a better assignment of codewords (not necessarily of the same length) for the same dictionary? The code should be prefix-free. Please construct the best such code. What is the overall compression achieved, i.e., the expected codeword length per source symbol in this case? How does it compare to that of the Tunstall code above? How does it compare to the entropy of the source? Explain.

Solution:

(a) Note that the codewords are of the same fixed length b. Such a code will achieve the entropy if the M words that compose the dictionary are uniformly distributed, that is, $P(w_i) = 1/M$, for $i = 1, 2, \ldots, M$, and if the number of words M equals 2^b. Specifically, note that in this case

$$H(W) = \log M = \log 2^b = b,$$

the length of the codewords.

(b) The entropy of the source is given by

$$\begin{aligned} H(U) &= \sum_u p(u) \log \frac{1}{p(u)} \\ &= 0.6 \times \log \frac{1}{0.6} + 0.3 \times \log \frac{1}{0.3} + 0.1 \times \log \frac{1}{0.1} \\ &= 1.2955 \end{aligned}$$

(c) To compute the entropy of the dictionary words, we first compute the probability of each of them, given by:

Word	Probability
b	0.3
c	0.1
ab	$0.6 \times 0.3 = 0.18$
ac	$0.6 \times 0.1 = 0.06$
aaa	$0.6 \times 0.6 \times 0.6 = 0.216$
aab	$0.6 \times 0.6 \times 0.3 = 0.108$
aac	$0.6 \times 0.6 \times 0.1 = 0.036$

Note that the probabilities sum up to 1.
The entropy of the dictionary words is given by

$$\begin{aligned}
H(W) &= \sum_w p(w) \log \frac{1}{p(w)} \\
&= 0.3 \times \log \frac{1}{0.3} + 0.1 \times \log \frac{1}{0.1} + 0.18 \times \log \frac{1}{0.18} + 0.06 \times \log \frac{1}{0.06} \\
&+ 0.216 \times \log \frac{1}{0.216} + 0.108 \times \log \frac{1}{0.108} + 0.036 \times \log \frac{1}{0.036} \\
&= 2.5391
\end{aligned}$$

(d) The expected length of the dictionary words is given by

$$\begin{aligned}
E[L_W] &= \sum_w p(w) l_w \\
&= 0.3 \times 1 + 0.1 \times 1 + 0.18 \times 2 + 0.06 \times 2 + 0.216 \times 3 + 0.108 \times 3 + 0.036 \times 3 \\
&= 1.96
\end{aligned}$$

(e) The overall compression rate is given by the expected length of the codewords (in this case 3), divided by the expected length of the dictionary words (in this case 1.96). That is, $3/1.96 = 1.5306$.
If the dictionary words were ”a”, ”b”, and ”c”, we would need codewords of length 2.
As expected, Tunstall performs better, since $1.5306 < 2$.

(f) The best such code is Huffman. One possible Huffman code in this case is the following:

Word	Probability	Codeword	Length of codeword
b	0.3	00	2
c	0.1	110	3
ab	0.18	100	3
ac	0.06	1110	4
aaa	0.216	01	2
aab	0.108	101	3
aac	0.036	1111	4

The expected length of the codewords in this case is given by

$$\begin{aligned}
E[L_{huffman}] &= \sum_{w} p(w) l_{huffman} \\
&= 0.3 \times 2 + 0.1 \times 3 + 0.18 \times 3 + 0.06 \times 4 + 0.216 \times 2 + 0.108 \times 3 + 0.036 \times 4 \\
&= 2.58,
\end{aligned}$$

which is smaller than 3, the codeword length of Tunstall.

Thus the expected number of bits per source symbol with Huffman is $2.58/1.96 = 1.3163$, less than that achieved with Tunstall (1.5306). This is expected, as Huffman is the best prefix-code, and thus it can not be worse than using Tunstall. Also, note that we are still above the entropy of the source ($H(U) = 1.2955$). This is also expected, as we can not do better than the entropy.

3. Mix of Problems (Extra credit: *20 points)*

(a) Consider a random variable X supported on $\{1, 2, 3, ...\}$ with $E[X] = 2$.

i. (*5 points*) Show that $H(X) \leq 2$.

ii. (*5 points*) What distribution of the random variable achieves $H(X) = 2$?

(b) Let $X_1, X_2, \ldots$ be an i.i.d. sequence of discrete random variables with entropy $H(X)$. Let

$$C_n(t) = \{x^n \in \mathcal{X}^n : p(x^n) \geq 2^{-nt}\}$$

denote the subset of n-sequences with probabilities $\geq 2^{-nt}$.

i. (*5 points*) Show that $|C_n(t)| \leq 2^{nt}$.

ii. (*5 points*) What is $\lim_{n\to\infty} P(X^n \in C_n(t))$? (distinguish between different values of t)

Solution:

(a) Consider $q(x) = \frac{1}{2^x}, x \in \{1, 2, 3, ...\}$. Now, for any distribution $p(x)$,

$$D(p||q) = \sum_x p(x) log\left(\frac{p(x)}{q(x)}\right) \geq 0 \tag{3}$$

On simplification, $D(p||q) = E[X] - H(X) \geq 0$, which gives the result. Also, as we know for $D(p||q)$, equality is obtained only when $p = q$, thus, $H(X) = 2$, when $p(x) = \frac{1}{2^x}, x \in \{1, 2, 3, ...\}$

One of the students gave the following solution, which is nice!
Consider the code assignment of $\{1, 01, 001, ...\}$ for the values $\{1, 2, 3...\}$. Observe that the code length ℓ_k corresponding to $X = k$ is also k.
Thus, the average codelength ℓ_{avg} of this code assignment would be:

$$\ell_{avg} = \sum_k \ell_k P(X = k) = \sum_k kP(X = k) = E[X] = 2 \tag{4}$$

Now, as we know ℓ_{avg} can never be lower than the entropy, this results in $H(X) \leq 2$, with equality if and only if the code lengths are the ideal length functions for the source, namely if and only if the source if geometric with parameter 1/2.

(b) i.

$$\begin{aligned}
1 &= \sum_{x^n \in \mathcal{X}^n} p(x^n) \\
&\geq \sum_{x^n \in \mathcal{C}^n(t)} p(x^n) \\
&\geq \sum_{x^n \in \mathcal{C}^n(t)} 2^{-nt}
\end{aligned}$$

$$\geq |\mathcal{C}^n(t)| 2^{-nt}.$$

Hence $|\mathcal{C}^n(t)| \leq 2^{nt}$, as we wanted to show.

ii. By AEP, for any $\epsilon > 0$, $P(A_\epsilon^{(n)}) \to 1$, where

$$A_\epsilon^{(n)} = \{x^n \in \mathcal{X}^n : 2^{-n(H+\epsilon)} \leq p(x^n) \leq 2^{-n(H-\epsilon)}\}.$$

If $t > H$ we note that for sufficiently small ϵ

$$C_n(t) \supset A_\epsilon^{(n)}$$

and hence $P(\{X^n \in C_n(t)\}) \to 1$.

On the other hand, for $t < H$, $C_n(t)$ does not intersect with $A_\epsilon^{(n)}$ and hence $P(\{X^n \in C_n(t)\}) \leq 1 - P(A_\epsilon^{(n)}) \to 0$.

Finally, for $t = H$, we have

$$\begin{aligned} C_n(t) &= \{x^n \in \mathcal{X}^n : p(x^n) \geq 2^{-nH}\} \\ &= \{x^n \in \mathcal{X}^n : -\frac{1}{n} \log p(x^n) \leq H\}. \end{aligned}$$

So

$$\begin{aligned} P(X^n \in C_n(t)) &= P(-\frac{1}{n} \log p(X^n) \leq H) \\ &= P(\frac{1}{n} \sum_{i=1}^n \log \frac{1}{p(X_i)} \leq H) \\ &= P(\frac{1}{n} \sum_{i=1}^n Y_i \leq E[Y]) \quad \text{(With } Y_i = \log \frac{1}{p(X_i)}) \\ &= P\big((\frac{1}{n} \sum_{i=1}^n Y_i - E[Y]) \leq 0\big) \\ &= P\big(\sqrt{n}(\frac{1}{n} \sum_{i=1}^n Y_i - E[Y]) \leq 0\big) \\ &= P\big(\frac{\sqrt{n}}{\text{Var}[Y]}(\frac{1}{n} \sum_{i=1}^n Y_i - E[Y]) \leq 0\big) \\ &\to \frac{1}{2}, \end{aligned}$$

where the last step follows from the Central Limit Theorem (CLT).

We gave full credit to anyone who got the $t > H$ and $t < H$ parts right.