

Solutions to Homework Set #7

1. **Minimax regret data compression and channel capacity.** First consider universal data compression with respect to two source distributions. Let the alphabet $V = \{1, e, 0\}$ and let $p_1(v)$ put mass $1 - \alpha$ on $v = 1$ and mass α on $v = e$. Let $p_2(v)$ put mass $1 - \alpha$ on 0 and mass α on $v = e$.

We assign word lengths to V according to $l(v) = \log \frac{1}{p(v)}$, the ideal codeword length with respect to a cleverly chosen probability mass function $p(v)$. The worst case excess description length (above the entropy of the true distribution) is

$$\max_i \left(E_{p_i} \log \frac{1}{p(V)} - E_{p_i} \log \frac{1}{p_i(V)} \right) = \max_i D(p_i \parallel p). \quad (1)$$

Thus the minimax regret is $R^* = \min_p \max_i D(p_i \parallel p)$.

- (a) Find R^* .
- (b) Find the $p(v)$ achieving R^* .
- (c) Compare R^* to the capacity of the binary erasure channel

$$\begin{bmatrix} 1 - \alpha & \alpha & 0 \\ 0 & \alpha & 1 - \alpha \end{bmatrix}$$

and comment.

Solution: Minimax regret data compression and channel capacity.

- (a) The whole trick to this problem is to employ the duality between universal data-compression and channel capacity. Although we don't immediately know how to solve the data-compression problem, we turn it into a channel-capacity problem whose answer we already know.

We know that if we wish to compress a source that is drawn from either of two possible distributions \mathbf{p}_1 or \mathbf{p}_2 , but we don't know which, then the minimax regret, R , that we can achieve using a universal data-compression scheme is equivalent to the capacity of a channel whose channel-transition matrix has rows which are precisely the probability distributions \mathbf{p}_1 and \mathbf{p}_2 .

In this case, $\mathbf{p}_1 = (1 - \alpha, \alpha, 0)$ and $\mathbf{p}_2 = (0, \alpha, 1 - \alpha)$, yielding the channel-transition matrix

$$\begin{bmatrix} 1 - \alpha & \alpha & 0 \\ 0 & \alpha & 1 - \alpha \end{bmatrix}$$

which we recognize immediately as a binary erasure channel with erasure probability α and hence capacity $1 - \alpha$. Thus, the minimax regret R is $1 - \alpha$.

- (b) We know that the $p(v)$ achieving R will be the center of the smallest relative-entropy ball that contains the \mathbf{p}_i . In the dual problem, involving the erasure channel, the center of this ball is the distribution induced on the channel output when we send according to the input distribution that achieves capacity. For the erasure channel, the input distribution that achieves capacity is $(\frac{1}{2}, \frac{1}{2})$, which induces a distribution of $(\frac{(1-\alpha)}{2}, \alpha, \frac{(1-\alpha)}{2})$ on the output. Thus, $p(v) = (\frac{(1-\alpha)}{2}, \alpha, \frac{(1-\alpha)}{2})$.
- (c) As indicated above, the minimax regret D^* is equal to the capacity of the channel.

2. **Arithmetic coding.** Let X_i be binary stationary Markov with transition matrix $\begin{bmatrix} \frac{1}{3} & \frac{2}{3} \\ \frac{2}{3} & \frac{1}{3} \end{bmatrix}$.

- (a) Find $F(01110) = \Pr\{.X_1X_2X_3X_4X_5 < .01110\}$.
- (b) How many bits $.F_1F_2 \dots$ can be known for sure if it is not known how $X = 01110$ continues?

Solution: *Arithmetic coding.*

- (a) The stationary distribution of this Markov chain is $(1/2, 1/2)$, and hence $p(0) = p(1) = \frac{1}{2}$. $p(01) = p(10) = 1/3$, $p(00) = p(11) = 1/6$, $p(100) = 3/32$, etc. We start with the equation

$$F(x^n) = \sum_{k=1}^n p(X^{k-1}0)x_k \tag{2}$$

$$= p(0)0 + p(00)1 + p(010)1 + p(0110)1 + p(01110)0 + \dots \tag{3}$$

$$= \frac{1}{2}0 + \frac{1}{2} \frac{1}{3}1 + \frac{1}{2} \frac{2}{3} \frac{2}{3}1 + \frac{1}{2} \frac{2}{3} \frac{1}{3} \frac{2}{3}1 + \frac{1}{2} \frac{2}{3} \frac{1}{3} \frac{2}{3} \frac{2}{3}0 + \dots \tag{4}$$

$$= \frac{25}{54} \tag{5}$$

$$= 0.0111011\dots \tag{6}$$

- (b) Since the next term in this series is $4/243 > 1/64$, we cannot know the 6th bit until we know the next input symbol. Since the next term is less than $1/32$, we know the 5th bit for sure.

3. **Lempel-Ziv.**

- (a) Continue the Lempel-Ziv parsing of the sequence 0,00,001,00000011010111.
- (b) Give a sequence for which the number of phrases in the LZ parsing grows as fast as possible.
- (c) Give a sequence for which the number of phrases in the LZ parsing grows as slowly as possible.

Solution: *Lempel-Ziv.*

- (a) The Lempel-Ziv parsing is: 0, 00, 001, 000, 0001, 1, 01, 011, 1

- (b) The sequence is: 0, 1, 00, 01, 10, 11, 000, 001, ... concatenating all binary strings of length 1,2,3, etc. This is the sequence where the phrases are as short as possible.
- (c) Clearly the constant sequence will do: 1, 11, 111, 1111, ...

4. **Another idealized version of Lempel-Ziv coding.** An idealized version of LZ was shown to be optimal: The encoder and decoder both have available to them the “infinite past” generated by the process, \dots, X_{-1}, X_0 , and the encoder describes the string (X_1, X_2, \dots, X_n) by telling the decoder the position R_n in the past of the first recurrence of that string. This takes roughly $\log R_n + 2 \log \log R_n$ bits.

Now consider the following variant: Instead of describing R_n , the encoder describes R_{n-1} plus the last symbol X_n . From these two the decoder can reconstruct the string (X_1, X_2, \dots, X_n) .

- (a) What is the number of bits per symbol used in this case to encode (X_1, X_2, \dots, X_n) ?
- (b) Modify the proof given in the text to show that this version is also asymptotically optimal, namely that the expected number of bits-per-symbol converges to the entropy rate.

Solution: *Another idealized version of Lempel-Ziv coding.*

In this version of LZ coding, the encoder and decoder both have available to them the infinite past generated by the process, \dots, X_{-1}, X_0 , and the encoder describes the string $X_1^n = (X_1, X_2, \dots, X_n)$ by telling the decoder the position R_{n-1} in the past of the first recurrence of the string X_1^{n-1} , plus the last symbol X_n .

- (a) Let A be the alphabet of the process, and $|A|$ denote its size. Then the number of bits it takes to represent R_{n-1} is roughly $\log R_{n-1} + C \log \log R_{n-1}$, where C is a constant independent of n . To represent X_n , it takes $\lceil \log |A| \rceil$ bits, so the overall number of bits per symbol used for the whole string X_1^n is

$$\frac{L_n(X_1^n)}{n} = \frac{\log R_{n-1} + C \log \log R_{n-1} + \lceil \log |A| \rceil}{n}.$$

- (b) To prove that this description is asymptotically optimal it suffices to show that

$$\limsup_{n \rightarrow \infty} \mathbf{E} \left(\frac{L_n}{n} \right) \leq H, \tag{7}$$

and the optimality will follow since we know that the reverse inequality also holds, by Shannon’s Noiseless Coding Theorem.

For the last term in L_n it is immediate that

$$\frac{\lceil \log |A| \rceil}{n} \rightarrow 0, \tag{8}$$

as $n \rightarrow \infty$. Now notice that we always have $R_{n-1} \leq R_n$, so for the first two terms in L_n ,

$$\mathbf{E} \left(\frac{\log R_{n-1} + C \log \log R_{n-1}}{n} \right) \leq \mathbf{E} \left(\frac{\log R_n + C \log \log R_n}{n} \right), \tag{9}$$

and as was shown in class, the above right-hand-side is asymptotically bounded above by H ,

$$\limsup_{n \rightarrow \infty} \mathbf{E} \left(\frac{\log R_n + C \log \log R_n}{n} \right) \leq H. \quad (10)$$

Combining (8) with (9) and (10), yields (7), as claimed.

5. **Tunstall Coding:** The normal setting for source coding maps a symbol (or a block of symbols) from a finite alphabet onto a variable length string. An example of such a code is the Huffman code, which is the optimal (minimal expected length) mapping from a set of symbols to a prefix free set of codewords. Now consider the dual problem of variable-to-fixed length codes, where we map a variable length sequence of source symbols into a fixed length binary (or D -ary) representation. A variable-to-fixed length code for an i.i.d. sequence of random variables $X_1, X_2, \dots, X_n, X_i \sim p(x), x \in \mathcal{X} = \{0, 1, \dots, m-1\}$ is defined by a prefix-free set of phrases $A_D \subset \mathcal{X}^*$, where \mathcal{X}^* is the set of finite length strings of symbols of \mathcal{X} , and $|A_D| = D$. Given any sequence X_1, X_2, \dots, X_n , the string is parsed into phrases from A_D (unique because of the prefix free property of A_D), and represented by a sequence of symbols from a D -ary alphabet. Define the efficiency of this coding scheme by

$$R(A_D) = \frac{\log D}{EL(A_D)} \quad (11)$$

where $EL(A_D)$ is the expected length of a phrase from A_D .

Prove that $R(A_D) \geq H(X)$.

Solution: *Tunstall Coding:*

We will argue that if $R(A_D) < H(X)$, then it is possible to construct a uniquely decodable code with average length less than the entropy. Consider a long sequence of i.i.d. random variables $X_1, X_2, \dots, X_n \sim \mathbf{p}$. We can parse this sequence into phrases using the prefix-free set A_D , and these phrases are independent and identically distributed, with the distribution induced by \mathbf{p} on the tree. Thus, using renewal theory, since the expected phrase length is $EL(A_D)$, the number of phrases in the block of length n is $\approx n/EL(A_D)$. These phrases can be described with $\log D$ bits each, so that the total description length is $\approx \log D(n/EL(A_D))$. If $R(A_D) < H$, then the total description length is less than nH and we have a contradiction to the fundamental theorem of source coding.

However, making the above argument precise raises issues for which we have two different solutions:

- The algorithm above does not describe how to handle a sequence of random variables that is a prefix of an element of A_D . For example, after parsing a block of length n into phrases from A_D , we might be left with a few symbols of X that are not long enough to make a phrase. We can imagine that these symbols are sent uncompressed, and the overhead is small (the set A_D is finite, and so the maximal length of a phrase in A_D is finite, and so the maximum length of the residue is bounded).

- We could extend the fundamental theorem of source coding directly to the variable to variable case, i.e., we can prove that for any mapping $F : \mathcal{X}^* \rightarrow \{0,1\}^*$, that $EL(A_D)H(X) \leq EL(C)$, where $EL(C)$ is the average length of the binary codewords induced by the mapping. An example of such a mapping for $\mathcal{X} = \{a, b, c\}$ is the mapping $aa \rightarrow 000, ab \rightarrow 001, ac \rightarrow 010, b \rightarrow 011, c \rightarrow 1$. It is easy to see that the above result is a “special” case of such a mapping, where we assume that we can find a code length of $\log D$ for all the elements of A_D .

Let Y be the random variable whose values are the elements of A_D , and whose distribution is induced by the distribution of X , i.e., if the phrase is ab , the probability $\Pr(Y = 'ab') = p_a p_b$. It is easy to see that Y is well defined random variable with total probability 1.

Now the code C is a code for the random variable Y , and by the standard source coding theorem, we have $EL(C) \geq H(Y)$. We will now prove that $H(Y) = EL(A_D)H(X)$, which will complete the proof.

There are many ways to prove this result, which is the Wald equation for entropies. We will prove it directly, using a summation over the leaves of a tree. Let $L(X_1, X_2, \dots, X_n)$ be the stopping defined by set A_D , i.e., $L(X_1, X_2, \dots, X_n) = l$ if the sequence of length l at the beginning of X is in A_D . Choose n larger than the maximal length in A_D , and let $Y = X_1^L$ be the first phrase in the parsing of X_1, X_2, \dots, X_n . Then

$$nH(X) = H(X_1, X_2, \dots, X_n) \quad (12)$$

$$= H(X_1^L, L, X_{L+1}^n) \quad (13)$$

$$= H(X_1^L) + H(L|X_1^L) + H(X_{L+1}^n|L, X_1^L) \quad (14)$$

Now since L is fixed given X_1^L , $H(L|X_1^L) = 0$. Also, X_{L+1}^n is independent of X_1^L given L , and we can write

$$H(X_{L+1}^n|L) = \sum_{l=1}^n \Pr(L = l)H(X_{l+1}^n) \quad (15)$$

$$= \sum_{l=1}^n \Pr(L = l)(n - l)H(X) \quad (16)$$

$$= (n - EL)H(X) \quad (17)$$

Substituting this in the equation above, we get

$$nH(X) = H(X_1^L) + (n - EL)H(X) \quad (18)$$

or

$$H(Y) = H(X_1^L) = ELH(X) \quad (19)$$

To prove the required result, we only need to verify that $H(Y) \leq \log D$, which follows directly from the fact that the range of Y is limited to D values.

6. **Optimal Integer Codes.** Consider a variation of the simple integer encoding described in Lemma 13.5.1. To encode the integer k : represent $\lceil \log(k+1) \rceil$ in unary, followed by the binary representation of k using $\lceil \log(k+1) \rceil$ bits, the first bit of which is 1, i.e.,

$$C_1(k) = \underbrace{00\dots 0}_{\lceil \log(k+1) \rceil \text{ 0's}} \underbrace{1xx\dots x}_{k \text{ in binary}} \quad (20)$$

It is easy to see that the length of this representation is $2\lceil \log(k+1) \rceil$.

- (a) Argue that this representation is prefix free.
- (b) Assume that the codeword 1 is used to represent the symbol “0”, and the other codewords are used to represent the integers $1, 2, \dots$. For what distribution on the integers is this encoding optimal?

Solution: *Optimal Integer Codes*

- (a) To show that this representation is prefix free, it is easy to see that any codeword that starts with k 0’s followed by a 1 cannot be a prefix of any other codeword that starts with more than k 0’s. Therefore no codeword of length $2k$ is a prefix of a longer codeword.

Since the binary representations of the integers between 2^{k-1} and $2^k - 1$ (all the integers whose binary representation is k bits) are distinct, the codewords of length $2k$ are distinct, and the code is prefix free.

- (b) It is easiest to construct the code explicitly for small x . Given the codeword lengths, the code is optimal if the input distribution is equal to $2^{-l(x)}$.

x	$c(x)$	$l(x)$	$p^*(x)$
0	1	1	0.5
1	01	2	0.25
2	0010	4	0.0625
3	0011	4	0.0625
4	000100	6	0.015625
5	000101	6	0.015625
6	000110	6	0.015625
7	000111	6	0.015625
8	00001000	8	0.00390625
	\vdots		

In general, the codeword for all integers between 2^{k-1} and $2^k - 1$ will have length $2k$ bits. The optimal probability distribution that corresponds to this would therefore be

$$p^*(x) = 2^{-2k} \quad \text{for } 2^{k-1} \leq x < 2^k \quad (21)$$

To show that this is a real distribution, we need to show that $\sum_x p^*(x) = 1$. Since there are 2^{k-1} integers between 2^{k-1} and 2^k ,

$$\sum_{x=1}^{\infty} p^*(x) = \sum_{k=1}^{\infty} 2^{k-1} 2^{-2k} \quad (22)$$

$$= \frac{1}{2} \sum_{k=1}^{\infty} 2^{-k} \quad (23)$$

$$= \frac{1}{2} \quad (24)$$

and adding the probability for $x = 0$, we get

$$\sum_{x=0}^{\infty} p^*(x) = 1. \quad (25)$$

We can also calculate the entropy of this distribution (which is equal to the average codeword length) as

$$H(p^*) = \frac{1}{2} \log 2 + \sum_{k=1}^{\infty} 2k 2^{k-1} 2^{-2k} \quad (26)$$

$$= \frac{1}{2} + \sum_{k=1}^{\infty} k 2^{-k} \quad (27)$$

$$= \frac{1}{2} + 2 \quad (28)$$

$$= 2.5 \quad (29)$$