ÉCOLE POLYTECHNIQUE FÉDÉRALE DE LAUSANNE

School of Computer and Communication Sciences

Handout 12 — Information Theory and Coding

Solutions to Homework 5 — Oct. 25, 2022

PROBLEM 1.

(a) It is easy to check that W is an i.i.d. process but Z is not. As W is i.i.d. it is also stationary. We want to show that Z is also stationary. To show this, it is sufficient to prove that the distribution of the process does not change by shift in the time domain.

$$\begin{aligned}
&p_Z(Z_m = a_m, Z_{m+1} = a_{m+1}, \cdots, Z_{m+r} = a_{m+r})\\
&= \frac{1}{2}p_X(X_m = a_m, X_{m+1} = a_{m+1}, \cdots, X_{m+r} = a_{m+r})\\
&+ \frac{1}{2}p_Y(Y_m = a_m, Y_{m+1} = a_{m+1}, \cdots, Y_{m+r} = a_{m+r})\\
&= \frac{1}{2}p_X(X_{m+s} = a_m, X_{m+s+1} = a_{m+1}, \cdots, X_{m+s+r} = a_{m+r})\\
&+ \frac{1}{2}p_Y(Y_{m+s} = a_m, Y_{m+s+1} = a_{m+1}, \cdots, Y_{m+s+r} = a_{m+r})\\
&= p_Z(Z_{m+s} = a_m, Z_{m+s+1} = a_{m+1}, \cdots, Z_{m+s+r} = a_{m+r}),
\end{aligned}$$

where we used the stationarity of the X and Y processes. This shows the invariance of the distribution with respect to the arbitrary shift s in time which implies stationarity.

(b) For the Z process we have

$$\begin{aligned}
H(Z) &= \lim_{n\to\infty} \frac{1}{n} H(Z_1, \cdots, Z_n)\\
&= \lim_{n\to\infty} \frac{1}{n} H(Z_1, \cdots, Z_n \mid \Theta)\\
&= \frac{1}{2}H(X_0) + \frac{1}{2}H(Y_0) = 1.
\end{aligned}$$

W process is an i.i.d process with the distribution $p_W(a) = \frac{1}{2}p_X(a) + \frac{1}{2}p_Y(a)$. From concavity of the entropy, it is easy to see that $H(W) = H(W_0) \geq \frac{1}{2}H(X_0) + \frac{1}{2}H(Y_0) = 1$. Hence, the entropy rate of W is greater than the entropy rate of Z and the equality holds if and only if X_0 and Y_0 have the same probability distribution function.

PROBLEM 2.

(a) Let $p_i = \frac{n_i}{n}$. Then

$$\begin{aligned}
1 = (p_1 + p_2 + \cdots + p_K)^n &\overset{(a)}{=} \sum_{\substack{n_1, n_2, \ldots, n_K\\ \text{s.t. } \sum n_i = n}} \binom{n}{n_1 n_2 \ldots n_K} p_1^{n_1} p_2^{n_2} \ldots p_K^{n_K}\\
&\geq \binom{n}{n_1 n_2 \ldots n_K} p_1^{n_1} p_2^{n_2} \ldots p_K^{n_K}\\
&\geq \binom{n}{n_1 n_2 \ldots n_K} 2^{n_1 \log(p_1) + n_2 \log(p_2) + \cdots + n_K \log(p_K)}\\
&\geq \binom{n}{n_1 n_2 \ldots n_K} 2^{-n\mathrm{h}(p_1, p_2, \ldots, p_K)},
\end{aligned}$$

where (a) is the binomial expansion. This proves our claim.

(b) For a random sequence of length n, $U_1U_2\dots U_n$, we encode the number of occurrences of the first $(K-1)$ letters, denoted $N_1, N_2, \dots, N_{K-1}$, since we get the last letter for free ($N_K = n - N_1 - N_2 - \dots - N_{K-1}$). For each letter we need at most $\lceil \log(n+1) \rceil$ bits. Now that we know the number of times each letter appeared in the sequence we need to encode the index of this specific sequence among all sequences having the same numbers of letter occurrences $(N_1, \dots, N_{K-1})$. Since there are $\binom{n}{N_1N_2\dots N_K}$ of those sequences then we need at most $\left\lceil \log\left(\binom{n}{N_1N_2\dots N_K}\right)\right\rceil$ bits.

Hence the total length L of the codeword is

$$L = (K-1)\lceil \log(n+1) \rceil + \left\lceil \log\left(\binom{n}{N_1N_2\dots N_K}\right)\right\rceil.$$

The expected length is

$$\begin{aligned}
\mathbb{E}(L) &= (K-1)\lceil \log(n+1) \rceil + \mathbb{E}\left(\left\lceil \log\left(\binom{n}{N_1N_2\dots N_K}\right)\right\rceil\right) \\
&\leq (K-1)\left(\log(n+1)+1\right) + \mathbb{E}\left(\log\left(\binom{n}{N_1N_2\dots N_K}\right)\right) + 1 \\
&\overset{(a)}{\leq} (K-1)\log(n+1) + K + \mathbb{E}\left(n\mathrm{h}\left(\frac{N_1}{n}, \frac{N_2}{n}, \dots, \frac{N_K}{n}\right)\right) \\
&\overset{(b)}{\leq} (K-1)\log(n+1) + K + n\mathrm{h}\left(\mathbb{E}\left(\frac{N_1}{n}, \frac{N_2}{n}, \dots, \frac{N_K}{n}\right)\right)
\end{aligned}$$

where (a) is due to the first part of the exercise and (b) is due to Jensen's inequality.

For a random sequence $U_1U_2\dots U_n$, the number of occurrences of a particular letter u_i is a random variables that can be written as the sum of indicator functions which take the value 1 with probability q_i

$$N_i = \sum_{j=1}^{n} 1_{\{U_j=u_i\}}.$$

Hence

$$\mathbb{E}\left(\frac{N_i}{n}\right) = \frac{\sum_{j=1}^{n}\mathbb{E}\left(1_{\{U_j=u_i\}}\right)}{n} = q_i.$$

Therefore, the expected codeword length per letter is

$$\frac{1}{n}\mathbb{E}(L) \leq (K-1)\frac{\log(n+1)}{n} + \frac{K}{n} + \mathrm{h}\left(q_1, q_2, \dots, q_K\right).$$

This shows that $\lim_{n\to\infty}\frac{1}{n}\mathbb{E}(L) \leq \mathrm{h}(q_1, q_2, \dots, q_K) = \mathrm{H}(U)$. Since the source is i.i.d then $\mathrm{H}(U) = \lim_{n\to\infty}\mathrm{H}(U_1U_2\dots U_n)$. This proves the optimality of this compression code for i.i.d sources.

(c) If the source is not i.i.d then $\mathrm{H}(U) \neq \lim_{n\to\infty}\mathrm{H}(U_1U_2\dots U_n)$. Hence, the code is not necessarily optimal.

Problem 3.

(a) We have for all k, n and ϵ, $P\big((U_1,\dots,U_n)\in T(n,p_k,\epsilon)\big) \le P\big((U_1,\dots,U_n)\in T(n,\epsilon)\big)$ as $T(n,\epsilon) \supseteq T(n,p_k,\epsilon)$. This implies that for any $\epsilon > 0$, with k and p such that $p_k = p$, we have

$$\begin{aligned}\lim_{n\to\infty} \Pr\big((U_1,\dots,U_n)\in T(n,p_k,\epsilon)\big) &\le \lim_{n\to\infty} \Pr\big((U_1,\dots,U_n)\in T(n,\epsilon)\big)\\ 1 &\le \lim_{n\to\infty} \Pr\big((U_1,\dots,U_n)\in T(n,\epsilon)\big).\end{aligned}$$

where the second line is due to the property of typical sets.

As we also have $\lim_{n\to\infty}\Pr\big((U_1,\dots,U_n)\in T(n,\epsilon)\big) \le 1$, with these inequalities we prove the statement.

(b) For typical sets, we know that $|T(n,p_k,\epsilon)| \le 2^{(1+\epsilon)H_k n} \le 2^{(1+\epsilon)Hn}$. Hence, we obtain the following upper bound.

$$|T(n,\epsilon)| = \left|\bigcup_k T(n,p_k,\epsilon)\right| \le \sum_k |T(n,p_k,\epsilon)| \le K2^{(1+\epsilon)Hn}.$$

By taking logartihm and dividing by n the above expression, we have

$$\frac{1}{n}\log|T(n,\epsilon)| \le (1+\epsilon)H + \frac{\log K}{n}.$$

This implies that for any $n \ge \log K/\epsilon$ we have

$$\frac{1}{n}\log|T(n,\epsilon)| \le (1+\epsilon)H + \epsilon.$$

(c) Let us use the construction of prefix-free code for typical set given in the lectures. First, take an injective function $f_{\epsilon,n} : T(n,\epsilon) \to \{0,1\}^{\lceil n(1+\epsilon)H+n\epsilon\rceil}$, this function exists for large enough n due to our result in (b). Now take another injective function $g_n : \mathcal{U}^n \to \{0,1\}^{\lceil n\log|\mathcal{U}|\rceil}$. We define $c_{\epsilon,n}(x)$ as $0||f_{\epsilon,n}(x)$ if $x \in T(n,\epsilon)$ and $1||g_n$ otherwise, where $||$ is the concatenation operator.

We have that

$$\begin{aligned}\Pr\Big(U^n \in T(n,\epsilon)\Big) &= \Pr\Big(\text{length}\big(c_{\epsilon,n}(U^n)\big) = \lceil n(1+\epsilon)H + n\epsilon\rceil\Big)\\ &\le \Pr\Big(\text{length}\big(c_{\epsilon,n}(U^n)\big) \le n(1+\epsilon)H + n\epsilon + 1\Big).\end{aligned}$$

From (a) we know that there exists an $n_a(\epsilon,\delta)$ such that $1-\delta < \Pr\Big(U^n \in T(n,\epsilon)\Big)$ for all $n \ge n_a(\epsilon,\delta)$. From (b) we require $n \ge \log K/\epsilon = n_b(K,\epsilon)$. To get the form required in the problem statement, we need that :

$$n\big((1+\epsilon)H + \epsilon + 1/n\big) < nR$$

Since $1/n \le \epsilon$ for $n \ge n_b(K,\epsilon)$, the following inequality will also work

$$n\big((1+\epsilon)H + 2\epsilon\big) < nR.$$

The above inequality satisfied by choosing an appropriate ϵ $\big($i.e., $0 \le \epsilon < \frac{R-H}{H+2}\big)$.

Therefore, for a code $c_{\epsilon,n}$ constructed as above and ϵ chosen small enough, we have

$$\begin{aligned}\Pr\Big(\text{length}\big(c_{\epsilon^*,n}(U^n)\big) < nR\Big) &\geq \Pr\Big(\text{length}\big(c_{\epsilon,n}(U^n)\big) \leq n(1+\epsilon)H + n\epsilon + 1\Big)\\ &\geq \Pr\Big(U^n \in T(n,\epsilon)\Big)\\ &> 1-\delta\end{aligned}$$

for all $n \geq \max\big\{n_a(\epsilon,\delta), n_b(K,\epsilon)\big\}$.

Problem 4.

(a) We have $\rho(X_1^\infty) = 0$. We show this by showing that $\rho(X_1^\infty) \leq \delta$ for any $\delta > 0$. To see the last statement, build an invertible FSM which "recognizes" a string of type "ab...ab" for a particular even length, call it L, and outputs lets say "0" at the end of this string and returns to the starting state. Hence this machine will output an infinite string of "0" when the input is X_1^∞. From each state (including the starting state) of the chain which recognizes the special string make an edge back to the starting state in the case the next input is not the correct one. The output for each such edge is $1 + \lceil \log L \rceil$ bits long, the first bit is 1 to indicate that it is not the special path and on the next $\lceil \log L \rceil$ bits we give the index of the state (in binary representation) from which the return edge is drawn. This machine is clearly lossless and has a compressibility of $1/L$ for the desired sequence.

(b) A machine as described above will have $\rho_M(X_1^\infty) = 1/4$. In fact, one cannot do better than this. Consider a cycle, when from a given state we get back to the same state. During such a cycle we have to output at least one symbol, because the machine has to be information lossless. In an L state machine we eventually create such a cycle within at most L steps. This means that we output at least one symbol for every L input symbols, so $\rho_M(X_1^\infty) \geq 1/L$.

(c) We have $\rho_{\text{LZ}} = 0$ since compressibility is non-negative and we know that the compressibility of LZ is at least as good as that of any FSM, i.e., we know that $\rho_{\text{LZ}}(X_1^\infty) \leq \rho(X_1^\infty)$.

(d) The dictionary increases by 1 every time and has size 2 in the beginning. Hence, if we look at lets say c steps of the algorithm then we need in total

$$\sum_{i=1}^{c} \lceil \log(1+i) \rceil \leq c \log(2(c+1))$$

bits to describe the output.

What are the words which we are using. Note that the parsing is a, b, ab, aba, ba, bab,... Note that in average at most every second step the length of the used dictionary word increases by 1, i.e., we have a linear increase in the used dictionary words. Therefore, if we compute the total length which we have parsed after c steps, this length increases like the square of c.

It follows that the ratio of the total number of bits used divided by the total length described behaves like $1/c$, i.e., it tends to 0.