Handout 11 | Information Theory and Coding
Homework 5 | Oct. 16, 2023

PROBLEM 1. Assume $\{X_n\}_{-\infty}^{\infty}$ and $\{Y_n\}_{-\infty}^{\infty}$ are two i.i.d. processes (individually) with the same alphabet, with the same entropy rate $H(X_0) = H(Y_0) = 1$ and independent from each other. We construct two processes Z and W as follows:

- To construct the process Z, we flip a fair coin and depending on the result $\Theta \in \{0, 1\}$ we select one of the processes. In other words, $Z_n = \Theta X_n + (1 - \Theta)Y_n$.
- To construct the process W, we do the coin flip at every time n. In other words, at every time n we flip a coin and depending on the result $\Theta_n \in \{0, 1\}$ we select X_n or Y_n as follows $W_n = \Theta_n X_n + (1 - \Theta_n)Y_n$.

(a) Are Z and W stationary processes? Are they i.i.d. processes?

(b) Find the entropy rate of Z and W. How do they compare? When are they equal?
Recall that the entropy rate of the process U (if exists) is $\lim_{n\to\infty} \frac{1}{n} H(U_1, \cdots, U_n)$.

PROBLEM 2. We have shown in class that

$$\binom{n}{k} \leq 2^{n \mathrm{h}_2\left(\frac{k}{n}\right)}.$$

(a) Given $n \in \mathbb{N}_+$ and $n_1, n_2, \ldots, n_K \in \mathbb{N}$ such that $\sum_{i=1}^{n} n_i = n$, we define the quantity $\binom{n}{n_1 n_2 \ldots n_K} = \frac{n!}{n_1! n_2! \ldots n_K!}$.
Show that

$$\binom{n}{n_1 n_2 \ldots n_K} \leq 2^{n \mathrm{h}(p_1, p_2, \ldots, p_K)},$$

where $p_i = \frac{n_i}{n}$ and $\mathrm{h}(p_1, \ldots, p_K) = -\sum_{i=1}^{K} p_i \log(p_i)$.

Let $U_1, U_2, \ldots$ be the letters generated by a memoryless source with alphabet $\mathcal{U} = \{u_1, u_2, \ldots, u_K\}$, i.e., $U_1, U_2, \ldots$ are i.i.d. random variables taking values in the alphabet $\mathcal{U}$ according to the distribution $q = \{q_1, q_2, \ldots, q_K\}$.

(b) We want to compress this source without any idea about its distribution. Describe an optimal universal code that achieves this goal. Give a proof of its optimality.
Hint: Use the same idea as for the binary source case.

(c) What if the source is not i.i.d. Will your code still be optimal?

PROBLEM 3. Suppose $p_1, p_2, \ldots, p_K$ are probability distributions on an alphabet $\mathcal{U}$. Let $H_1, \ldots, H_K$ be the entropies of these distributions, and let $H = \max_k H_k$. Fix $\epsilon > 0$ and for each $n \geq 1$ consider the set

$$T(n, \epsilon) = \bigcup_k T(n, p_k, \epsilon)$$

where $T(n, p_k, \epsilon)$ is the set of ϵ-typical sequences of length n with respect to the distribution p_k, i.e., $T(n, p_k, \epsilon) = \left\{u^n \in \mathcal{U}^n : \forall_{u' \in \mathcal{U}} \left|\frac{1}{n} N_{u'}(u^n) - p_k(u')\right| < \epsilon p_k(u')\right\}$ where $N_{u'}(u^n)$ is the number of occurrences of u' in sequence u^n.

Suppose that $U_1, U_2, \ldots$ are i.i.d. with distribution p where p is one of $p_1, \ldots, p_K$.

(a) Show that $\lim_{n\to\infty} \Pr\big((U_1,\dots,U_n) \in T(n,\epsilon)\big) = 1$. (In particular for any $\delta > 0$, for n large enough $\Pr\big((U_1,\dots,U_n) \in T(n,\epsilon)\big) > 1-\delta$.)

(b) Show that for large enough n, $\frac{1}{n}\log|T(n,\epsilon)| < (1+\epsilon)H + \epsilon$.

(c) Fix $R > H$ and $\delta > 0$. Show that for n large enough there is a prefix-free code $c : \mathcal{U}^n \to \{0,1\}^*$ such that

$$\Pr\Big(\text{length}\big(c(U^n)\big) < nR\Big) > 1-\delta$$

whenever $U_1, U_2, \dots$ are i.i.d. with distribution p, where p is one of $p_1, \dots, p_K$.

Problem 4. Let the alphabet be $\mathcal{X} = \{a, b\}$. Consider the infinite sequence $X_1^\infty = ababababababab\dots$

(a) What is the compressibility of $\rho(X_1^\infty)$ using finite-state machines (FSM) as defined in class? Justify your answer.

(b) Design a specific FSM, call it M, with at most 4 states and as low a $\rho_{\mathrm{M}}(X_1^\infty)$ as possible. What compressibility do you get?

(c) Using only the result in point (a) but no specific calculations, what is the compressibility of X_1^∞ under the Lempel–Ziv algorithm, i.e., what is $\rho_{\mathrm{LZ}}(X_1^\infty)$?

(d) Re-derive your result from point (c) but this time by means of an explicit computation.