Instructions

Collaboration: Collaboration is allowed, but limit yourselves to groups of size at most four.

References: In general, try not to run to reference material to answer questions. Try to think about the problem to see if you can solve it without consulting any external sources. If this fails, you may look up any reference material.

Writeup: You must write the solutions by yourselves. Cite all references and collaborators. Explain why you needed to consult any of the references, if you did consult any.

Problems

   (a) What is the maximum possible length (expressed in big-Oh notation) of the Lempel-Ziv compression of an $n$-bit string? (Prove the upper bound, and display a string whose compression matches your bound to within constant factors.)
   (b) What is the shortest length (again in big-Oh notation) of the Lempel-Ziv compression of an $n$-bit string? (Prove the lower bound, and display a string whose compression matches your bound to within constant factors.)
   (c) Suggest a variation of Lempel-Ziv that would achieve a minimum compression length (as defined in Part (b)) of $\text{poly log } n$.

2. Fixed-length Encoding. In the lectures we focussed on zero-error compression methods, i.e., where the decompressor always recovered the original message, and used the expected length of the encoding as the performance measure. The goal of this exercise is to show that if one prefers to work with worst-case encoding length, one can do so, at the expense of an exponentially small probability of error.
   (a) State a precise theorem that captures the ability to compress using a fixed-length encoding, at rates arbitrarily close to the entropy of the source, for an i.i.d. source over a finite alphabet, with an exponentially small probability of decoding error.
   (b) Prove your theorem above.

(Note that an aim of this exercise is to test your ability to express yourselves mathematically. Watch out for imprecision, inaccuracy, and weaknesses in the theorem statement in Part (a).)
3. Typical sets and divergence. Recall that a set \( S \in \Omega^n \) is \((\varepsilon, n)\)-typical for a distribution \( P \) supported on \( \Omega \) if the probability under \( P^n \) of every element in the set is in the interval 
\[ (1/|S|^{1+\varepsilon}, 1/|S|^{1-\varepsilon}) \]
and the probability of \( S \) under \( P^n \) is at least \( 1 - \varepsilon \). For distributions \( P, Q \) supported on \( \Omega \) prove that for every \( \varepsilon > 0 \) and for sufficiently large \( n \), there is an \((\varepsilon, n)\)-typical set \( S \) for \( Q \) such that \( \Pr_{Z \sim P^n}[Z \in S] \leq 2^{-(1-\varepsilon)D(Q||P)\cdot n} \).

4. Compressing Hidden Markov Models and Learning Parity with Noise. The “Learning Parity with Noise” (LPN) problem is a classic in learning theory and asks to solve the following problem. We are given black box access to a source \( L = L_S \) for some (unknown) \( S \subseteq [k] \). Every time we press button on the black box it outputs a pair \((v, b)\) where \( v \sim \text{Uniform}([0,1]^k) \) and \( b = \bigoplus_{i \in S} v_i \oplus \eta \) where \( \eta \sim \text{Bern}(p) \). Each time the button is pressed the box outputs a sample from this distribution independently. The goal of the LPN problem is to to discover the secret \( S \) after sampling from \( L \) polynomially many times (in \( k \)) with running time also polynomial in \( k \). It is believed by many that such an algorithm does not exist and that any algorithm must take time exponential in \( k \) (even if it is allowed exponentially many samples).

(a) For every \( p < 1/2 \) show that there is an exponential time algorithm that samples \( L \) polynomially many times and produces the secret \( S \) with high probability.

(b) Describe an \( O(k) \)-state hidden markov model \( M \) that generates samples according to \( L \). I.e., the output sequence of the model \( M \) can be divided into blocks of length \( k+1 \) where each block has distribution identical to that of the black-box \( L \).

(c) Determine the entropy rate \( H(M) \) of the source \( M \).

(d) Show that if a polynomial time universal compression algorithm for HMMs can compress \( n \) length sequences from this source to an expected length of \((H(M) + \varepsilon) \cdot n \) in time \( \text{poly}(k, 1/\varepsilon) \) for some \( n = k^{O(1)} \) then there is a polynomial time algorithm to solve LPN. (Conclude that universal compression algorithms must take time that is exponential in \( k \) or exponential in \( 1/\varepsilon \).)

5. Markov Chains. Recall that a \( k \)-state Markovian source is given by a \( k \times k \) stochastic matrix, whose entries specify the transition probability, i.e. \( M_{ij} = \Pr[Z_2 = j|Z_1 = i] \).

(a) Given as input a matrix \( M \) describing a Markovian chain that produces a state sequence \( Z_1, Z_2, \ldots \), describe an algorithm to compute its entropy rate, i.e., \( \lim_{n \to \infty} H(Z_n|Z_1, \ldots, Z_{n-1}) \).

(You may assume standard linear algebra.)

(b) The entropy rate of a dog looking for a bone: A dog walks on the integers, moving one step in its current direction on typical time steps, while pausing to dig for a bone on some steps and reversing directions occasionally. Specifically, if the dog is at integer \( i \) at a given moment of time and \( i = 0 \text{(mod 3)} \) then the dog pauses with probability .2 and continues w.p. .8 (in its current direction). If \( i \neq 0 \text{(mod 3)} \) then the dog continues in its current direction with probability .9 and reverses directions with probability 0.1. Let \( X_1, X_2, \ldots, X_t, \ldots \) describe the sequence of locations of the dog over time. (Is this sequence a Markov chain?) Use your algorithm from Part (a) to compute the entropy rate of this sequence, i.e., the quantity \( \lim_{t \to \infty} \{H(X_t|X_1, \ldots, X_{t-1})\} \).

CS 229r Information Theory in CS-2