Homework 5: Information Theory & Compression

UC Irvine CS177: Applications of Probability in Computer Science

Due on November 22, 2019 at 11:59pm

Question 1: (25 points)

You would like to survey students at UCI to determine the mean time t (in hours) that they sleep each night. Assume you will independently sample n students from the population, and let X_i be the number of hours reported by student i. The mean of their answers is then

$$M_n = \frac{1}{n}\sum_{i=1}^{n} X_i.$$

You guess that all students have sleep times between 4 and 10 hours, and thus assume that $\text{Var}(X_i) \leq (10-4)^2/12 = 3.0$, the variance of a uniform distribution over that range.

a) How large should n be so that Chebyshev's inequality guarantees that the estimate M_n is within 30 minutes (1/2 hour) of the true t, with probability at least 0.99?

After learning about the central limit theorem, you assume that the number of surveyed students n is sufficiently large that the distribution of M_n is normal. The Python command `scipy.stats.norm.ppf` may be helpful when answering this question.

b) Assuming the distribution of M_n is normal, how large should n be so that the estimate M_n is within 30 minutes (1/2 hour) of the true t, with probability at least 0.99?

c) Briefly discuss how and why the answers obtained using the central limit theorem differ from those obtained using Chebyshev's inequality.

Question 2: (15 points)

We revisit a probabilistic model for a fault diagnosis problem from an earlier homework. The class variable C represents the health of a disk drive: $C = 0$ means it is operating normally, and $C = 1$ means it is in a failed state. When the drive is running it continuously monitors itself using a temperature and shock sensor, and records two binary features, X and Y. $X = 1$ if the drive has been subject to shock (e.g., dropped), and $X = 0$ otherwise. $Y = 1$ if the drive temperature has ever been above 70°C, and $Y = 0$ otherwise. The following table defines the joint probability mass function of these three random variables:

x	y	c	$p_{XYC}(x, y, c)$
0	0	0	0.1
0	1	0	0.2
1	0	0	0.2
1	1	0	0.1
0	0	1	0.0
0	1	1	0.1
1	0	1	0.05
1	1	1	0.25

For the questions below, provide the numerical value of your answer, as well as equations showing how you calculated that answer.

a) Compute $H(X)$ and $H(Y)$, the entropies of the sensor variables. Which variable has more uncertainty?

b) Compute the mutual information $I(X; C)$ and $I(Y; C)$ between each feature and the drive health C.

c) Which single diagnostic feature, temperature or shock, has the strongest dependence on the drive health C? Justify your answer.

Question 3: (20 points)

Now suppose now we want to transmit the state of the drive in question 2 over a network. The drive can be in one of $2^3 = 8$ different states $S \in \{1, \ldots, 8\}$ depending on the values of X, Y, and C. For example, $S = 1$ if $(X = 0, Y = 0, C = 0)$ and $S = 2$ if $(X = 0, Y = 1, C = 0)$. These states are listed in the table above along with their probabilities.

a) If we use a fixed length code for S, what is the expected codeword length?

b) Construct a Huffman code for S based on the probabilities from question 2. Derive the code graphically using the technique described in lecture, where a binary tree is constructed by recursively merging the lowest probability symbols.

c) What is the expected codeword length for the Huffman code from part (b)?

d) Calculate and report the entropy $H(S)$ of the hard drive state. Briefly discuss the relationship between the entropy and the expected Huffman code length.

Question 4: (40 points)

In this problem, you will build Huffman codes for two real datasets. We have provided a Python implementation of the Huffman coding algorithm, which uses a priority queue (or heap queue) to efficiently track the least frequent symbols as the binary tree is constructed.

We first consider the text data contained in the file `theTortoiseAndTheHare.txt`. The single text string is a classic fable about a tortoise and a hare.

a) Complete the implementation of the `getEntropy()` *function in Python. Compute and report the entropy of the empirical distribution of characters in* `theTortoiseAndTheHare.txt`.

b) Using the provided coding algorithm, build a Huffman code for the text data, and encode it as a binary string. What is the average number of coded bits per text character? Discuss the relationship between this average code length and the entropy computed in part (a).

c) Complete the implementation of the Huffman `decode()` *function in Python. To verify the correctness of your decoder, encode a couple of short text strings, and verify that the decoder outputs match the encoder inputs.*

d) The demonstration code includes a 55-bit encoding of a text message. Report the output of `decode()` *when it is applied to this message.*

We now consider the dataset of web traffic in `sessionLengths.npy`. The 100 integers are the number of web sessions over various lengths recorded over a few months for an ICS server. The first number is the number of sessions of length 1, the second number is the number of sessions of length 2, and the 100^{th} number is the number of sessions of length 100. (For this problem, we ignore the infrequent sessions of longer length.)

e) Compute and report the entropy of the empirical distribution of web session lengths.

f) Using the provided code, build the Huffman tree for the web traffic data. The demonstration code includes a 49-bit encoding of a series of web sessions. Report the output of `decode()` *when it is applied to this encoded data.*

g) For the web traffic Huffman code, compute and report the expected codeword length according to the following formula:

$$E[l(x)] = \sum_{x=1}^{100} p_X(x) l(x).$$

Here $l(x)$ is the number of bits for the codeword corresponding to x, and $p_X(x)$ is the empirical distribution of web session lengths. Discuss the relationship between this expected codeword length and the entropy computed in part (e).