4.1 Problems from Texts

Do problems 6.1, 6.5, 6.7, 6.8 in Yeung.

4.2 Additional Problems

Problem 1: Types and type classes

Let the alphabet of a source be $X = a, b, c$ and consider all strings of symbols of length 4 from the source.

- (a) List the elements $P_4$, i.e., all the 4-types.
- (b) For each 4-type $P \in P_4$, give a string of length 4 from the source that has this type.
- (c) For each 4-type $P \in P_4$, determine the cardinality $|T(P)|$ of the corresponding type class $T(P)$.

Problem 2: Number of types

Show that the exact number of types of a length $n$ sequences of discrete random variables is exactly

$$\binom{n + |X| - 1}{|X| - 1}$$

(note that this is the same problem as in the handout on the method of types, problem 1 on page 39).

Problem 3. Matlab, the entropy of images and the entropy of text, and source coding.

Images, 3a Consider a probability distribution $p(X)$ over a $32 \times 32$ pixel image, where $X$ is a $32 \times 32$ random matrix. Assume, for this discussion, that there are 8 bit pixels which means that in total there are a large number $256^{32 \times 32} = 10^{2466}$ possible images that are representable (this is more than the number of atoms in the universe which is estimated to be about $10^{78}$, and certainly more than the entire human population has viewed throughout the course of its existence).\(^1\)

When $p(X)$ is a uniform distribution, then when we sample from such a distribution, the images would look entirely random — the likelihood of ever encountering an “real” image (say of birds, bees, trees, or industrial manufacturing plants), one we would likely see in our world, is exceedingly small. You can create such images in matlab with the following command imagesc(rand(32,32)). Note that the probability of seeing the image that you see is the same as that of seeing an image of yourself, when generating such image samples.

Since all the images are equally likely, the entropy of such a distribution is $H = 8192$ (why?).\(^2\)

---

\(^1\)While the matlab function surely uses more than 8 bits, we describe it in terms of 8 bits just for simplicity. In fact there are be many more possible images when the random number generator uses more than 8 bits.

\(^2\)Assume in this problem that rand() is a real and not a pseudo random number generator.
Your goal is to design a Matlab program which 1) generates random $32 \times 32$ pixel images, 2) has as low entropy as possible, and 3) still appears to the human perceptual system as random pixels (much like the Matlab command `imagesc(rand(32,32))` appears). Argue why the entropy for your scheme must be low. Include in your HW a fair number of samples from your distribution. Also include the Matlab code, and make sure to comment the code well. What can you say about the compressibility of such images as compared to `imagesc(rand(32,32))`?

Note that one solution you might imagine would have the distribution generate over a small handful of random looking images and just choose randomly between one of that small handful, and that would be low entropy. But this doesn’t really capture the spirit of the problem. The goal is to produce a random process that, compared to the number of images that could be generated if each bit was chosen by a fair coin, the entropy should be *much* less, but it should still seem (to a human observer) that the images are for all intents and purposes random. I.e., what would it take for the program to generate essentially an enumerable, for practical purposes, number of images, but still be as low entropy as possible? Also, compute the entropy of your distribution.

The last part of the question is to imagine that you had a random image generator that generates scenes from the natural world (trees, mountains, etc.). Would you think the entropy of the underlying distribution would be low or high relative to 8192? Explain why?

What to turn in:

1. Answer why the uniform distribution has entropy $H = 8192$.
2. The Matlab code to generate random looking images.
3. A few sample images that you have generated from your distribution.
4. A few sample images from the uniform distribution (as a comparison).
5. Compute the entropy of your distribution.
6. Consider $p_n(x)$, the natural world generator, what would you guess as the entropy of $p_n(x)$?

**Text, 3b** This is the same problem, but here you are to generate text strings. Assume that each string is of length 256, and that you have an alphabet size of 27 (capital letters ‘A’ through ‘Z’ plus space). The uniform distribution has entropy $256 \log 27$. Your goal is to produce a distribution that has much lower entropy but that looks to a Human just as random as the uniform distribution.

What to turn in:

1. The Matlab code to generate random looking strings.
2. A few sample strings from your distribution.
3. A few sample strings from the uniform distribution.
4. How might you estimate the entropy of real English text?

**Problem 4:** Shannon’s source coding theorem. In class, we proved using the method of types that if $R > H$, the simple encoder based on enumerating lexicographically the set that were type-good was such that it had exponentially decreasing error with $n$ as $n \to \infty$. We proved a very weak “converse” to this theorem which states for this encoder, that if $R \leq H$ that the error of this particular encoder would quickly go to unity as $n$ gets large.

Your problem here is to prove a more general form of the converse. Your goal is to prove that for any encoder that compresses a source with a rate $R < H$ (strict) that the probability of error goes to unity. Note that you do not have to prove this from scratch. Rather, the problem being asked is for you to search the texts, the Internet, and the information theory literature (e.g., IEEE Transactions on Information Theory) for a proof, and then re-state it and re-write it in your own words and using as similar a notation as the one that we used in class and in the book.
**Problem 5:** Inclusion/Exclusion: The handout posted on the web proving the general inclusion/exclusion formula was not as general as it could be since it used the cardinality measure rather than a general signed measure. Starting from the handout, modify the proof so that it applies for any signed additive measure $\mu(\cdot)$ (this is not hard).

**Problem 6:** Can correlated sensors be better than uncorrelated ones? In class, we considered the following two possible settings, where $X$ is a random source that is communicated to you in two possible ways, either via two separate looks $(Z_1, Z_2)$, or two dependent looks $(W_1, W_2)$. The top row of the following picture describes our setting:

![Diagram of communication channels](image)

The question we discussed in class was which of $I(X; Z_1, Z_2)$ and $I(X; W_1, W_2)$ would be larger, the thinking being that the former, $(Z_1, Z_2)$ would be better when striving to discover information about $X$ since there are separate looks at $X$, or they are independent. Assume in that:

$$I(Z_1; X) = I(Z_2; X) = I(W_1; X) = I(W_2; X)$$

Also assume all random variables have the same domain size (equal cardinality) so that there is no cardinality based capacity advantage of one over the other.

Assume moreover that all variables are binary, and that $X$ is a fair coin flip ($p(X = 1) = p(X = 0) = 0.5$). You have two questions:

6a: The second row of the figure above shows a $Z(\delta)$ channel and a $BSC(\epsilon)$ channel ($BSC = \text{binary symmetric channel}$). In this problem, you are to find values of $\delta$ and $\epsilon$ such that $I(W_1, W_2; X) > I(Z_1, Z_2; X)$, where for the edges marked as “C” in the top row you are to use a $Z$-channel and for the edges marked as “D” in the top row you are to use a BSC. Once you have found your values of $\delta$ and $\epsilon$, compute $I(X; W_1, W_2)$ to verify that it is negative as expected. In this case, the $\delta$ and $\epsilon$ you find should be the same for all channels in both left and right top-row figures.

6b: While it is possible to show that correlated sensors can have more information about a source than independent looks, what about the capacity? I.e., what is the maximum capacity of the two schemes $I(X; Z_1, Z_2)$ and $I(X; W_1, W_2)$. In this case, you are to maximize over $\delta$ and $\epsilon$ (in this case the $\delta$ and $\epsilon$ may be allowed to be different for each channel, the only constraint is the type of channels, Z-channels for the top portion and BSC for the lower). What is the capacity of each scheme and in this case which would you prefer?