PHYS 7895: Quantum Information Theory Fall 2015

Lecture 2 28 August, 2015

Prof. Mark M. Wilde *Scribe: Kevin Valson Jacob*

1 Overview

In the last lecture we saw an overview of the course.

In this lecture we will discuss Shannon's theorem on data compression.

2 Data compression

Let us consider a scenario in which Alice (A) wants to send a message to Bob (B). Let us imagine that they are connected by a noiseless communication channel. This means that whatever message Alice sends through the channel, Bob receives exactly the same message. In other words, there is no error introduced by the channel. Mathematically, this is represented as the following conditional probability distribution:

$$p_{Y|X}(y|x) = \delta_{y,x}, \quad x, y \in \mathcal{X} \tag{1}$$

where $\mathcal{X}$ is the alphabet that Alice uses to send the information and x and y are symbols selected from the alphabet.

2.1 An example of data compression

Consider a simple scenario in which Alice's alphabet has only four symbols a, b, c, d. The symbols are drawn from an information source where different symbols occur with certain probabilities. For this example, let the probability distribution be as follows:

$$\Pr\{a\} = 1/2, \tag{2}$$
$$\Pr\{b\} = 1/8, \tag{3}$$
$$\Pr\{c\} = 1/4, \tag{4}$$
$$\Pr\{d\} = 1/8. \tag{5}$$

As the communication channel accepts only bits as inputs, Alice has to use a coding scheme to map her alphabet to bits. A naive code that Alice could use is the following two-bit code, wherein the mapping is as follows:

$$a \to 00, \quad b \to 01, \quad c \to 10, \quad d \to 11. \tag{6}$$

In this case, we see that each symbol uses two bits. That is to say, the message length is always two. However, this coding scheme does not take advantage of the skewed probability distribution

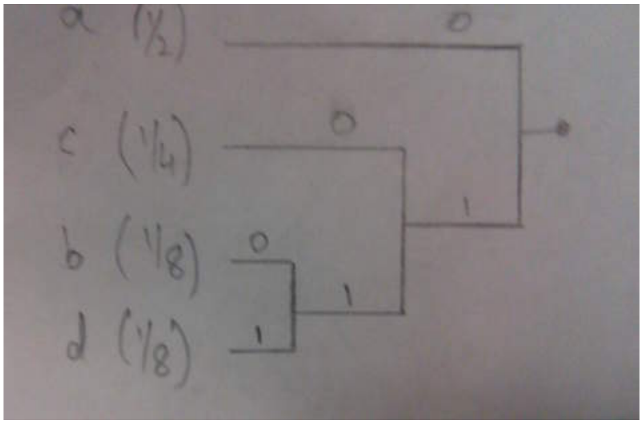

Figure 1: To find the Huffman code for each letter, arrange alphabets in the decreasing order of their probabilities. Draw lines connecting the letters as shown above and assign 0 or 1 as indicated. The Huffman code for each letter will be the binary string encountered as we move from the right to the letter.

of the symbols of the alphabet. A more advantageous coding scheme is a **Huffman code**. This scheme is shown in Figure 1. In this case, the mapping is done as

$$a \to 0, \quad b \to 110, \quad c \to 10, \quad d \to 111, \tag{7}$$

This mapping is uniquely decodable. For example, if Bob receives a binary string 0011010111010100010, then it represents the message *aabcdacaac*. In this case, the average length of a codeword is

$$\frac{1}{2}(1) + \frac{1}{8}(3) + \frac{1}{4}(2) + \frac{1}{8}(3) = \frac{7}{4} \text{ bits per symbol.} \tag{8}$$

This reduction in the length of a codeword is possible because the code exploits the skewedness of the probability distribution of the symbols of the information source. This possibility motivates data compression.

2.2 Measure of information content

We will now start using the term information in a precisely technical manner. The Huffman coding scheme suggests to us a measure of information content known as the 'surprisal'. That is to say, we are more surprised on finding a less probable letter, say 'b', rather than a more probable letter, say 'a'. This motivates us to define the measure of information content as follows:

$$i(x) \equiv \log\left(\frac{1}{p_X(x)}\right) = -\log_2\left(p_X(x)\right) \tag{9}$$

We note that $i(x)$ diverges when $p_X(x) = 0$. In order to avoid this problem, if $p_X(x_i) = 0$ for any x_i, then we will just say that x_i is not part of the code. The behavior of the function $i(x)$ is shown in Figure 2.

Information content as defined above has the desired property of **additivity**. If two random variables X_1 and X_2 are independent, then

$$p_{X_1,X_2}(x_1, x_2) = p_{X_1}(x_1)p_{X_2}(x_2). \tag{10}$$

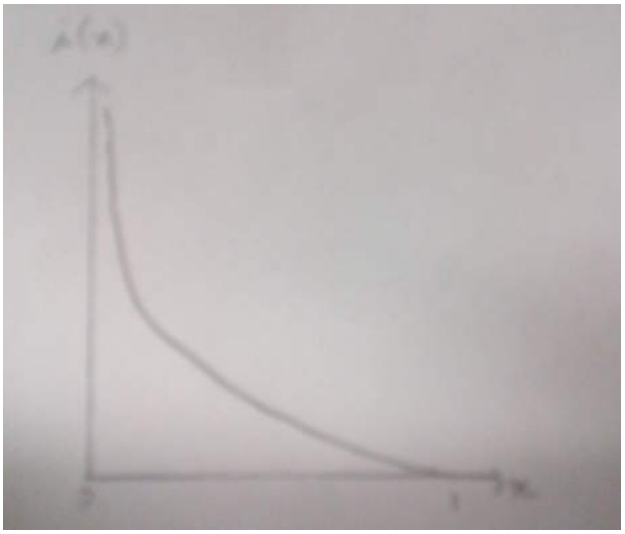

Figure 2: Behavior of ‘surprisal’

Therefore,

$$\begin{aligned} i\left(x_1, x_2\right) &= -\log\left(p_{X_1,X_2}\left(x_1, x_2\right)\right) && (11)\\ &= -\log\left(p_{X_1}\left(x_1\right) p_{X_2}\left(x_2\right)\right) && (12)\\ &= -\log\left(p_{X_1}\left(x_1\right)\right) - \log\left(p_{X_2}\left(x_2\right)\right) && (13)\\ &= i\left(x_1\right) + i\left(x_2\right). && (14) \end{aligned}$$

Thus we note that as the number of independent random variables grows, the information content grows additively.

We will now define the notion of ‘**entropy**’ as the expected ‘surprisal’. Mathematically,

$$H(X) = \sum_x p_X(x) i(x) = -\sum_x p_X(x) \log\left(p_X(x)\right) \tag{15}$$

Evaluating the entropy of the probability distribution in (5) gives

$$\begin{aligned} H(X) = -\frac{1}{2}\log\frac{1}{2} - \frac{1}{8}\log\frac{1}{8} - \frac{1}{4}\log\frac{1}{4} - \frac{1}{8}\log\frac{1}{8} \\ &= \frac{1}{2}(1) + \frac{1}{8}(3) + \frac{1}{4}(2) + \frac{1}{8}(3) && (16)\\ &= \frac{7}{4}. && (17) \end{aligned}$$

This is exactly the expected length of the code using the Huffman scheme (note that this is *not* a general phenomenon, but occurs for this example due to the fact each element of the probability distribution is an inverse power of two).

3 Shannon’s source coding theorem

The Huffman coding scheme motivates us to consider the notion of data compression. Shannon put forth the idea that we may relax the notion of perfect reproduction of the source so as to obtain

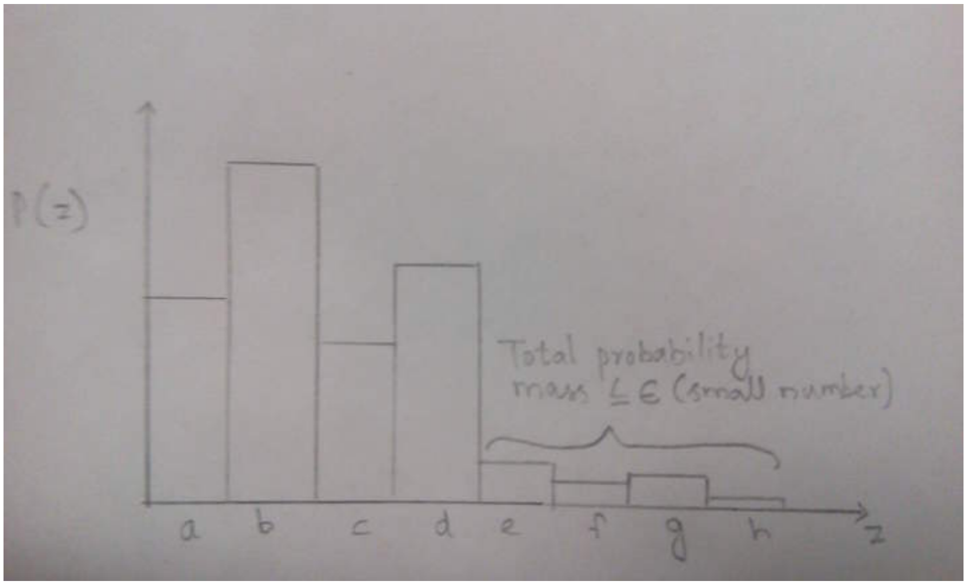

Figure 3:

'Almost lossless data compression'. This idea is as follows:

Suppose a random variable Z has a histogram as shown in Figure 3. The realizations a, b, c, and d together contribute a probability mass greater than $1 - \epsilon$, where $\epsilon \in (0, 1)$ and we think of it as a small positive number. Then the idea behind 'almost lossless data compression' is to ignore e, f, g, and h so as to save a bit in representing an information source faithfully. Thus we notice that if we allow a small failure probability ϵ in reproducing an information source, then we can save on the number of bits necessary to represent the information source.

In the above scenario, the probability distribution of the information was a bit particular, which allowed us to argue about how to save using compression if a small error probability is allowed. However, things are not so particular in the i.i.d. case. Consider a sequence of random variables $Z^n = Z_1 \cdots Z_n$. Then the sequence of realizations is $z^n = z_1 \cdots z_n$. Then, as a consequence of the law of large numbers, we will note that the total probability mass of a relative small number of realizations of Z^n will be greater than $1 - \epsilon$. This is illustrated in Figure 4.

Law of large numbers Let $Z_1 \cdots Z_n$ be a sequence of i.i.d. random variables. Let the expectation value of the random variable be μ. That is,

$$\mu = \sum_z z \; p_Z(z) \tag{18}$$

Then, for all tolerances $\delta > 0$ and error probabilities $\epsilon \in (0, 1)$, the following bound holds:

$$\Pr[|\bar{Z}^n - \mu| \leq \delta] \geq 1 - \epsilon, \tag{19}$$

where the sample average $\bar{Z}^n$ is defined as

$$\bar{Z}^n = \frac{1}{n} \sum_{i=1}^{n} Z_i \tag{20}$$

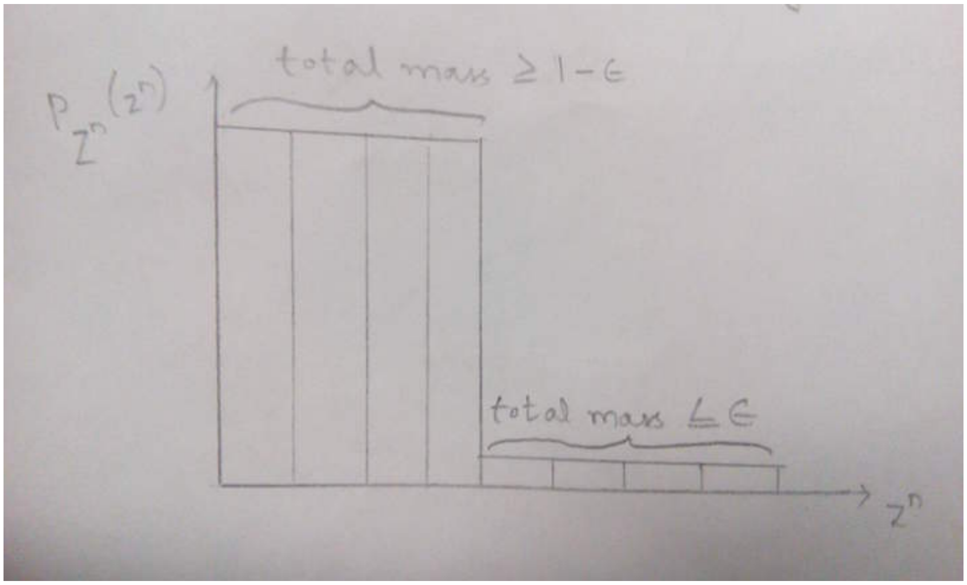

Figure 4:

Interpreting the law of large numbers, if someone picks ϵ and δ, then we can pick n large enough so as to achieve both the tolerance δ and the success probability $1-\epsilon$.

Let's return to data compression.

3.1 Shannon compression task

Define an (n, R, ϵ) code where n represents the length of the data sequence, R represents the rate of the code, and ϵ denotes the error probability. Consider an information source (Random variable X) which outputs sequence x^n drawn i.i.d. according to the distribution of the random sequence $X^n = X_1 \cdots X_n$. Alice, the sender of the information, encodes this using an encoding map

$$E : \mathcal{X}^n \to \{0,1\}^{nR}. \tag{21}$$

where $\{0,1\}^{nR}$ stands for binary sequences of length nR. This set has size 2^{nR}. We want $R \leq \log_2 |\mathcal{X}|$ as only then there is compression.

In the communication scheme, Alice transmits the codewords over nR uses of a noiseless classical channel. Bob decodes according to some decoding map $D : \{0,1\}^{nR} \to \mathcal{X}^n$. This is schematically depicted in Figure 5. The probability of error for an (n, R, ε) source code is

$$p(e) \equiv \Pr\{(D \circ E)(X^n) \neq X^n\} \leq \varepsilon. \tag{22}$$

The compression rate is defined as the number of noiseless channel bits transmitted over the channel divided by the number of source symbols, and is equal to R. We now focus on compression rate. We say that a compression rate is **achievable** if for $\epsilon \in [0,1), \delta > 0$, and sufficiently large n, there exists an $(n, R+\delta, \epsilon)$ compression code.

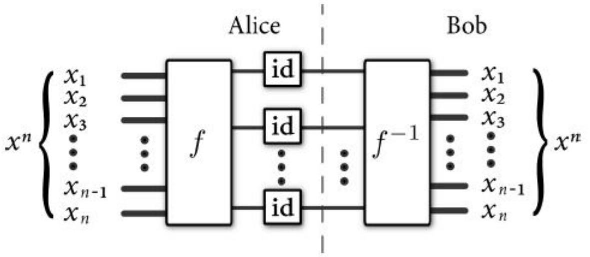

Figure 5: Schematic depiction of the communication between Alice and Bob

3.2 Shannon data compression theorem

The Shannon data compression theorem states that the entropy of the source is the lowest achievable rate of compression. That is,

$$\inf \{R : R \text{ is achievable for } X\} = H(X). \tag{23}$$

This theorem is proved by proving LHS $\geq$ RHS, and RHS $\geq$ LHS. We will prove the latter, which is the achievability part.

Define a typical set or a high probability set as T^n_δ where

$$T^n_\delta \equiv \left\{ x^n \in \mathcal{X}^n : \left| -\frac{\log_2 p_{X^n}(x^n)}{n} - H(X) \right| \leq \delta \right\} \tag{24}$$

We have

$$-\frac{1}{n}\log_2 p_{X^n}(x^n) = -\frac{1}{n}\log_2 \prod_{i=1}^{n} p_{X_i}(x_i) = \frac{1}{n}\sum_{i=1}^{n}(-\log_2 p_{X_i}(x_i)) \tag{25}$$

We can rewrite the probability that a random sequence X^n falls in the typical set as follows:

$$\Pr(X^n \in T^n_\delta) = \Pr\left(\left| \frac{1}{n}\sum_{i=1}^{n}(-\log_2 p_{X_i}(X_i)) - H(X) \right| \leq \delta \right) \tag{26}$$

Thus, from the law of large numbers, we can conclude that

$$\Pr(X^n \in T^n_\delta) \geq 1 - \epsilon \tag{27}$$

Using the definition of the typical set, we can find a bound on its size with the following argument:

$$1 = \sum_{x^n \in T^n_\delta} p_{X^n}(x^n) \geq \sum_{x^n \in T^n_\delta} 2^{-n[H(X)+\delta]} \tag{28}$$

$$= |T^n_\delta| 2^{-n[H(X)+\delta]} \tag{29}$$

This implies that

$$|T^n_\delta| \leq 2^{n[H(X)+\delta]} \tag{30}$$

Recalling that $|\mathcal{X}^n| = 2^{n \log |\mathcal{X}|}$, we note that the typical set is exponentially smaller than the total set.

Thus the proof guarantees that there exists a scheme that can compress at the rate of entropy in the asymptotic limit. What is the scheme? The encoder is an invertible mapping from the typical set to the set of binary sequences of size no larger than $2^{n[H(X)+\delta]}$. If a sequence emitted by the source is atypical, then map it to any binary sequence in the aforementioned set. The decoder then simply executes the inverse of the above encoding. The probability of making an error is equal to the probability that a random sequence is not typical, which can be made arbitrarily small with increasing n. The rate of the scheme is equal to $H(X) + \delta$. Since δ can be made arbitrarily small with increasing n, we have shown that the entropy is an achievable rate.