18.200 Lecture notes April 17, 2024

Data compression & Shannon's Entropy Theorem

Lecturer: Peter Shor

In these notes, we want to consider some basic questions and notions in *information theory*. Roughly speaking, we would like to answer questions such as *how much information is contained in some piece of data*? One way to approach this question is to say that the data contains n bits of information (in average) if it can be coded by a binary sequence of length n (in average). This is useful both for data compression (how to make the data as compact as possible) and for data transmission. In these notes, we focus on the noiseless setting in which the bits of information are not modified (either reliably stored, or reliably transmitted); in this case, the more compact the code the better. Later on, we will consider the noisy setting in which the bits may be modified/corrupted (either randomly or by an adversary); this will lead to error-correcting codes that form the foundations of our digital life. Eg, how are bits stored in a CD/DVD to allow recovering the data after minor scratches, or how are digital transmissions performed to recover from minor interferences/failures?

After basic definitions, we discuss Shannon's *entropy theorem*, which is one of the founding results of the field of information theory. The codes that come from this theorem are not at all practical, and in the next set of lecture notes, we develop a practical coding scheme, known as Huffman coding.

1 Coding

Let S be a finite set of objects. A *coding function* for the set S is a function which associate a distinct binary sequence $f(s)$ to each element s in S. The binary sequence $f(S)$ is called code of S. Different objects might be encoded with binary sequences of different lengths. Here are lower bounds for the length of codes.

Lemma 1. *If S contains N elements then at least one of the codes has length greater or equal to $\lfloor \log_2(N) \rfloor$. If one consider the uniform distribution for elements in S then the codes have length at least $\log_2(N) - 2$ in expectation.*

Exercise: Prove Lemma 1. Remember that different elements of S may get encoded by binary strings of different lengths.

Example 1: coding permutations. Let $\mathbb{S}_n$ be the set of permutations of $\{1, 2, \ldots, n\}$. We now discuss a possible coding function f for the set $\mathbb{S}_n$. Recall that for any integer i, the binary representation of i has $\lceil \log_2(i+1) \rceil$ bits. Thus each number $i \in \{1, 2, \ldots, n\}$ can be represented uniquely by binary sequences of length exactly $\lceil \log_2(n+1) \rceil$: it suffice to take their binary representations and add a few 0 in front if necessary to get this length. Let $\pi \in \mathbb{S}_n$ be a permutation seen as a sequence of distinct numbers $\pi = \pi_1 \pi_2 \ldots \pi_n$. One can define $f(\pi)$ as the concatenation of the binary sequences (of length $\lceil \log_2(n) \rceil$) corresponding to each number $\pi_1 \pi_2 \ldots \pi_n$. Then the length of the code $f(\pi)$ is $n \lceil \log_2(n+1) \rceil \sim n \log_2(n)$. We can recover the permutation from the code:

if one has the code, it can cut it in subsequences of length $\log_2(n+1)$ each and then recover the numbers $\pi_1\pi_2\dots\pi_n$ making the permutation. Is it an efficient coding? Well according to Lemma 1 we cannot achieve codes shorter than $\log_2(n!) - 2$ in average. Moreover, $\log_2(n!) \sim n\log_2(n)$ (see Stirling's formula below). Therefore our coding function f has length as short as possible asymptotically.

Example 2: coding Dyck paths. Consider the set $\mathbb{D}_n$ of Dyck path of length $2n$. There is an easy way of coding a Dyck path $D \in \mathbb{D}_n$ by a binary sequence of length $2n$. Simply encode down steps by "0" and up steps by "1" this give a binary sequence $f(D)$ of length $2n$. Could we hope for shorter codes? Certainly it would be possible to get a code of length $2n-2$ because the first step is an up step and the last step is a down step, so these could be ignored. But could we do better than $2n + o(n)$? We have seen that the set $\mathbb{D}_n$ has cardinality $N = \frac{2n!}{n!(n+1)!}$. Using Stirling's formula, we derive that

$$\log_2(N) = \log_2(2n!) - \log_2(n!) - \log_2((n+1)!) = 2n + o(n).$$

Therefore, by Lemma 1 one cannot encode Dyck paths by codes of length less than $2n + o(n)$ on average. So our naive coding is asymptotically optimal. Observe that this also gives a way of coding plane trees or binary trees optimally.

Roughly speaking, we want to answer such questions as *how much information is contained in some piece of data*? One way to approach this question is to say that the data contains n bits of information (in average) if it can be coded by a binary sequence of length n (in average). So information theory is closely related to data compression.

Lemma 1 gives a bound on the expected length of a code for a uniform distribution. Shannon's entropy theorem will provide the best expected length of a code for more general distributions.

2 Some History

History of data compression. One of the earliest instances of widespread use of data compression came with telegraph code books, which were in widespread use at the beginning of the 20th Century. At this time, telegrams were quite expensive; the cost of a transatlantic telegram was around $1 per word, which would be equivalent to something like $30 today. This led to the development of telegraph code books, some of which can be found in Google Books. These books gave a long list of words which encoded phrases. Some of the codewords in these books convey quite a bit of information; in the Fourth edition of the ABC Code, for example, "Mirmidon" means "Lord High Chancellor has resigned" and "saturation" means "Are recovering salvage, but should bad weather set in the hull will not hold out long."[1]

Information theory. In 1948, Claude Shannon published a seminal paper "A Mathematical Theory of Communication" which founded the field of information theory[2]. In this paper, among other things, he set data compression on a firm mathematical ground. How did he do this? Well, he set up a model of random data, and managed to determine how much it could be compressed. This is what we will discuss now.

[1] *The ABC Universal Commercial Telegraph Code, Specially Adapted for the Use of Financiers Merchants, Shipowners, Brokers, Agents, Etc.* by W. Clauson-Thue, American Code Publishing Co., Fourth Edition (1899)

[2] Available on-line at http://dx.doi.org/10.1002/j.1538-7305.1948.tb01338.x

3 Random data and compression

First of all we need a model of data. We take our data to be a sequence of letters from a given alphabet A. Then we need some probabilistic setting. We will have a random *source* of letters. So we could say that we have random letters $X_1, X_2, X_3, \ldots$ from A. In these notes we will assume that we have a *first-order source*, that is, we assume that the random variables $X_1, X_2, X_3, \ldots$ are independent and identically distributed. So for any letter a in the alphabet A the probability $\mathbb{P}(X_n = a)$ is some constant p_a which does not depend on n or on the letters $X_1, X_2, \ldots, X_{n-1}$.

Shannon's theory actually carries out to more complicated models of sources (Markov chains of any order). These more complicated sources would be more realistic models of reality. However for simplicity, but we shall only consider first-order sources in these notes.[3]

Now we need to say what data compression means. We shall encode data by *binary sequences* (sequences of 0 and 1). A *coding function* ϕ for a set S of messages is simply a function which associates to each element $s \in S$ a distinct binary sequence $\phi(s)$. Now if the messages s in S have a certain probability distribution then the length L of the binary sequence $\phi(s)$ is a random variable. We are looking for codes such that the average length $\mathbb{E}(L)$ is as small as possible. In our context, the random messages will be the sequences $s = (X_1, X_2, \ldots, X_n)$ consisting of the first n letters coming out of the source. One way to encode these messages is to attribute distinct binary sequences of length $\lceil \log_2(|A|) \rceil$ to the letters in the alphabet A. Then the binary sequence $\phi(s)$ would be the concatenation of the codes of the letter, so that the length L of $\phi(s)$ would be $n\lceil \log_2(|A|) \rceil$. That's a perfectly valid coding function, leading to average length $\mathbb{E}(L) = n\lceil \log_2(|A|) \rceil$. Now the main question is: can we do better? How much better? This is what we discuss next.

4 Shannon's entropy Theorem

Consider an alphabet $A = \{a_1, \ldots, a_k\}$ and a first-order source X as above: the nth (random) letter is denoted X_n. For all $i \in \{1, \ldots, k\}$ we denote by p_i the probability of the letter a_i, that is, $\mathbb{P}(X_n = a_i) = p_i$. We define the *entropy* of the source X as

$$H(p) = -\sum_{i=1}^{k} p_i \log_2(p_i).$$

We often denote the entropy just by H, without emphasizing the dependence on p. The entropy $H(p)$ is a nonnegative number. It can also be shown that $H(p) \leq \log_2(k)$ by concavity of the logarithm. This upper bound is achieved when $p_1 = p_2 = \ldots = p_k = 1/k$. We will now discuss and prove (leaving a few details out) the following result of Shannon.

[3]Suppose for instance you are trying to compress English text. We might consider that we have some sample corpus of English text on hand (say, everything in the Library of Congress). Shannon considered a series of sources, each of which is a better approximation to English. The *first-order source* which emits a letter a with probability p_a which is proportional to its frequency in the text. The probability distribution of a sequence of n letters from this source is just n independent random variables where the letter a_j appears with some probability p_j. The *second-order source* is that where a letter is emitted with probability that depends only on the previous letter, and these probabilities are just the conditional probabilities that appear in the corpus (that is, the conditional probability of getting a 'u', given that the previous letter was a 'q', is derived by looking at the frequency of all letters that follow a 'q' in the corpus). In the third-order source, the probability of a letter depends only on the two previous letters, and so on. High-order sources would seem to give a pretty good approximation of English, and so it would seem that a compression method that works well on this class of sources would also work well on English text.

Theorem 1 (Shannon's entropy Theorem)**.** *Let X be a first order source with entropy H. Let ϕ be any coding function (in binary words) for the sequences $s = (X_1, X_2, \ldots, X_n)$ consisting of the first n letters coming out of the source. Then the length L of the code $\phi(s)$ is at least $H\,n$ on average, that is,*

$$\mathbb{E}(L) \geq H\,n - o(n),$$

where the "little o" notation means that the expression divided by n goes to zero as n goes to infinity. Moreover, there exists a coding function ϕ such that

$$\mathbb{E}(L) \leq H\,n + o(n).$$

So the entropy of the source tells you how much the messages coming out of it can be compressed. Another way of interpreting this theorem is to say that the amount of information coming out of the source is "H bits of informations per letters".

We will now give a sketch of the proof of Shannon's entropy Theorem. First, let's try to show that one cannot compress the source too much. We look at a sequence of n letters from the first-order source X, with the probability of letter a_i being p_i for all i in $[k]$.

First observe that the number of sequences of length n with exactly n_i letter a_i is

$$\binom{n}{n_1, n_2, \ldots, n_k} = \frac{n!}{n_1! n_2! \cdots n_k!},$$

and all these words have the same probability $p_1^{n_1} p_2^{n_2} \cdots p_k^{n_k}$. Now, if we have some number M of equally likely messages that must be sent, then in order to send them we need to use $\log M$ bits on average (see Lemma 1). So we need to send at least

$$\log_2 \binom{n}{n_1, n_2, \ldots, n_k}$$

bits. To approximate this, we can use Stirling's formula

$$n! \sim \sqrt{2\pi n} \left(\frac{n}{e}\right)^n.$$

It gives

$$\log_2(n!) = n \log_2(n) - n \log_2(e) + o(n).$$

Using this formula one obtains

$$\begin{aligned}
\log_2 \binom{n}{n_1, n_2, \ldots, n_k} &= \log_2(n!) - \sum_{i=1}^{k} \log_2(n_i!) \\
&= n \log_2(n) - \sum_{i=1}^{k} n_i \log_2(n_i) + o(n) \\
&= -\sum_{i=1}^{k} n_i \log_2(n_i/n) + o(n).
\end{aligned}$$

In terms of the entropy function, this can be rewritten as:

$$\log_2 \binom{n}{n_1, n_2, \ldots, n_k} = nH\left(\frac{n_1}{n}, \cdots, \frac{n_k}{n}\right) \pm o(n).$$

So we have a conditional lower bound on the length of the coding sequences

$$\mathbb{E}(L|n_1,\ldots,n_k) \geq -\sum_{i=1}^{k} n_i \log_2(n_i/n) - o(n). \tag{1}$$

In particular if $n_i = p_i\, n$ for all i one gets $\mathbb{E}(L|n_1,\ldots,n_k) \geq -\sum_{i=1}^{k} n\, p_i \log_2(p_i) - o(n) = n\, H - o(n)$.

Now we want to find what n_i is in general. The expectation of n_i is np_i. Moreover it can be shown that n_i is very concentrated around this expectation. Actually, using Chebyshev's inequality one can prove (do it!) that for any constant $\epsilon > 0$

$$\mathbb{P}\left(|n_i - np_i| \geq \epsilon n\right) \leq \frac{p_i(1-p_i)}{\epsilon^2 n}.$$

Now we fix a constant $\epsilon > 0$ and we consider two cases. We define a sequence of letters to be *ϵ-typical* if $|n_i - np_i| \leq \epsilon n$ for all i and *ϵ-atypical* otherwise. By the union bound, the above gives a bound on the probability to be ϵ-atypical:

$$\mathbb{P}(\epsilon\text{-atypical}) \leq \sum_{i=1}^{k} \frac{p_i(1-p_i)}{\epsilon^2 n} \leq \frac{1}{\epsilon^2 n}.$$

Moreover Equation (1) gives

$$\mathbb{E}(L|\epsilon\text{-typical}) \geq -n \sum_{i=1}^{k} (p_i - \epsilon)\log_2(p_i + \epsilon) + o(n).$$

We now use the linearity of expectation to bound the length of the coded message:

$$\begin{aligned} \mathbb{E}(L) &= \mathbb{E}(L|\epsilon{-}\text{typical})\mathbb{P}(\epsilon{-}\text{typical}) + \mathbb{E}(L|\epsilon{-}\text{atypical})\mathbb{P}(\epsilon{-}\text{atypical}) \\ &\geq (-n\sum_{i=1}^{k}(p_i-\epsilon)\log_2(p_i+\epsilon) + o(n)) \cdot (1 - \frac{1}{\epsilon^2 n}). \end{aligned}$$

Since one can take ϵ as small as one wants, this shows that

$$\mathbb{E}(L) \geq n\, H - o(n).$$

So we have proved the first part of the Shannon entropy theorem.

We now will show that one can do compression and get coded messages of length no more than Hn in average. We use again the linearity of expectation to bound the length of the coded messages:

$$\mathbb{E}(L) = \mathbb{E}(L|\epsilon{-}\text{typical})\mathbb{P}(\epsilon{-}\text{typical}) + \mathbb{E}(L|\epsilon{-}\text{atypical})\mathbb{P}(\epsilon{-}\text{atypical}).$$

Now, we need to analyze this expression. $\mathbb{P}(\epsilon{-}\text{atypical}) \leq \frac{c}{n}$, so as long as we don't make the output in the atypical case more than length Cn, we can ignore the second term, as this one will be constant while the first term will be linear. What we could do is use one bit to tell the receiver whether the output was typical and atypical. If it is atypical, we can send it without compression (thus sending $\log_2(k^n) = n\log_2(k)$), and if it typical, we can then compress it. The dominant

contribution to the expected length then occurs from typical outputs, because of the rarity of atypical ones.

How do we compress the source output if it's typical? One of the simplest ways theoretically (but this is not practical) is to calculate the number of typical outputs, and then assign a number (in binary representation) to each output. This compresses the source to $\log_2$ of the number of typical outputs. We will do this and get an upper bound of $nH + o(n)$ bits, where the "little o" notation means that the expression divided by n goes to zero as n goes to infinity.

How do we calculate the number of typical outputs? For each typical vector of numbers n_i, we have that the number of outputs is

$$\binom{n}{n_1, n_2, \ldots, n_k.}$$

So an upper bound on the total number of typical outputs is

$$\sum_{i=1}^{k} \sum_{n_i : np_i - \epsilon \le n_i \le np_i + \epsilon} \binom{n}{n_1, n_2, \ldots, n_k},$$

this is an upper bound as we haven't taken into account that $\sum_{i=1}^{n} n_i = n$. But we can even use an even cruder upper bound by upper bounding the number of terms in the summation by n^k (it is not necessary to use the improved $(2\epsilon n)^k$). Thus, we get that the number of typical outputs is

$$\le n^k \binom{n}{(p_1 \pm \epsilon)n, (p_2 \pm \epsilon)n, \ldots, (p_k \pm \epsilon)n},$$

where we can choose the $p_i \pm \epsilon$ to maximize the expression. Taking logs, we get that the number of bits required to send a typical output is at most

$$k \log_2 n + nH + cn\epsilon,$$

for some constant c. The first term is negligible for large n, and we can let ϵ go to zero as n goes to ∞ so as to get compression to $nH + o(n)$ bits.

In summary, Shannon's entropy theorem says that we need to transmit $H\,n$ bits and this is can be essentially achieved. We'll see next a much more practical way (Huffman codes) to do the compression. In its basic version, it does not quite achieve the Shannon bound but it comes very close to it. Furthermore, it can be improved by encoding longer blocks of symbols and this gives an alternate proof of Shannon's result.

18.200 Principles of Discrete Applied Mathematics

Spring 2024