Notes on Entropy and Information

Blake Stacey

February 16, 2007

Chapter 1

Introduction

These notes cover the topics covered in the Science After Sunclipse seminar series on entropy and information, led by Ben Allen, Eric Downes and Blake Stacey in the early months of 2007. The mailing list for the seminar is `diyu@mit.edu` (as in Do-It-Yourself University).

Chapter 2

Shannon Theory

Our first topic is Claude Shannon's classic 1948 article, *The Mathematical Theory of Communication.* Ben Allen leads the discussion (16 February 2007), selecting one key theorem for close study.

The problem before us is to discover the best way of transmitting information. Shannon considered different types of communication methods, ways of sending a message from hither to yon. We first look at *discrete noiseless* transmissions, in which the message arrives at the destination without error, and in which the message is constructed out of a limited set of symbols (like an alphabet). Shannon proved that there exists a *bound* for how well this can be done.

Define a random variable X. This is an object which can take a set of different values x_i with probabilities p_i. When we send a message, we *sample* the random variable X several times and transmit the values we observe. For example, we could be looking at a traffic light, observing whether it is red, green or yellow, and communicating the results of our successive observations. We shall consider the situation where we *collect* our sequence of N observations and transmit them all in one block.

To obtain an intuitive notion of Shannon's proof, we invoke the law of large numbers. If the total N is large, then the distribution of x_i will reflect their associated probabilites p_i. (We expect to see far fewer instances of x than of e, and fewer orange lights than red or green.) How many times will each x_i occur?

$$\langle x_i \rangle = N p_i. \tag{2.1}$$

Can we say with what probability a typical message will arise by this random choice process?

$$p(\text{typical sequence}) = p_1^{Np_1} p_2^{Np_2} \cdots p_m^{Np_m} \equiv P. \tag{2.2}$$

Here, we say that we have m symbols in our alphabet. If we take the logarithm of this probability, we find that

$$\lg P = N \sum_i p_i \lg p_i, \tag{2.3}$$

and we compute a representative quantity per symbol,

$$H = -\frac{\lg P}{N}, \tag{2.4}$$

which we call the *entropy*:

$$\boxed{H = -\sum_i p_i \lg p_i.} \tag{2.5}$$

Theorem 2.0.1. *Given $\epsilon > 0$ and $\delta > 0$ we can find N_0 such that the sequences of length $N > N_0$ fall into two classes: first, a set whose probability is less than ϵ, and second, sequences whose probability satisfy*

$$\left| \frac{log P^{-1}}{N} - H \right| < \delta. \tag{2.6}$$

The second class constitute the "typical sequences".

$$-\delta < \frac{\lg P^{-1}}{N} - H < \delta \tag{2.7}$$

$$N(H-\delta) < \lg P^{-1} < N(H+\delta) \tag{2.8}$$

$$2^{-N(H-\delta)} < P < 2^{-N(H+\delta)}. \tag{2.9}$$

We see that there are *many more sequences* in class 1, but they are *not very likely.* When we weight by probability, the sequences in the second class are predominant. We can therefore choose an encoding scheme which is optimized for the messages of the second class without worrying about inefficiences relevant for the first class.

The number of sequences in class 2 is given roughly by the inverse of the probability, or 2^{NH}. We can represent all of them by a string of NH bits. Since the sequences of the first class are much less likely, we can use a less efficient coding scheme without concern for the penalties. For example, with m symbols we could use a coding scheme which sends messages of length $N \lg m$ bits.

For example, suppose that X takes the value 0 with probability $\frac{1}{8}$ and the value 1 with probability $\frac{7}{8}$. An inefficient code could transmit a message of N symbols in N bits (naturally). What about an efficient coding? First, we compute the entropy:

$$H = -\sum_i p_i \lg p_i \tag{2.10}$$

$$= -\left(\frac{1}{8}\lg\frac{1}{8} - \frac{7}{8}\lg\frac{7}{8}\right) \tag{2.11}$$

$$\approx 0.544. \tag{2.12}$$

We see that each symbol transmitted down the channel contains just over one-half a bit of information!

To do a second example, we consider the following symbols and probabilities:

$$\begin{cases} A & \frac{1}{2} \\ B & \frac{1}{4} \\ C & \frac{1}{8} \\ D & \frac{1}{8} \end{cases} \tag{2.13}$$

A naïve code assigns two bits to each symbol. For example:

$$\begin{cases} A & \to 00 \\ B & \to 01 \\ C & \to 10 \\ D & \to 11 \end{cases} \tag{2.14}$$

What about a *smart* coding? Because A is the most common symbol, it should receive the *shortest representation,* a string of one bit. B should have a representation in two bits, and C and D should both be encoded as three bits. With a little cleverness, we can arrange this so we are not confused, and we never encounter ambiguity in decoding our messages. This leads us into *prefix-free encoding,* in which no code word is the beginning of any other code word. To wit:

$$\begin{cases} A & \to 1 \\ B & \to 01 \\ C & \to 001 \\ D & \to 000 \end{cases} \tag{2.15}$$

Prefix-free codes are nice, because we can encode as we go along. We could start encoding the message $ABBCDAAA\ldots$ in the following way:

$$ABBCDAAA\ldots \to 10101001\ldots$$

It can be proven that we do not gain efficiency by dropping the prefix-free requirement: being confusing does not help!

Discuss 2.0.2. Can a code be prefix-free, suffix-free and still decodable? What are the conditions for being able to divide an entire message uniquely into code groups?

Discuss 2.0.3. DNA plays tricks like this! It has start codons and stop codons which delimit genetic messages, and *frame-shift mutations* are a real concern. What can we say about the coding of DNA?

There is another way of looking at this issue, an approach with *starts* with prefix-free encoding and arrives at a nice inequality. (The final result holds for all uniquely decodable schemes, but our proof applies to the prefix-free case.) Start with a binary alphabet, and let $n_1, \ldots, n_k$ be the lengths of codewords in a prefix-free code $\mathcal{C}$. Then

$$\sum_{i=1}^{k} 2^{-n_i} \leq 1. \tag{2.16}$$

The converse to this statement is perhaps even cooler: if we have a set of symbols which satisfy this condition, then we know we can construct a prefix-free code to represent them!

So, consider a code $\mathcal{C}$ with binary codewords w_i whose lengths are given by n_i. For messages of length N, how many messages have w_i as a prefix? Briefly put, 2^{N-n_i}. Because the sets of messages beginning with different prefixes w_i and w_j are non-overlapping, we can add:

$$\sum_{i} 2^{N-n_i} \leq 2^N \tag{2.17}$$

Dividing both sides by 2^N yields the relation we sought to prove, the *Kraft inequality*.