Entropy Presentation

Swarthmore College Jaron Shrock

February 25, 2018

1 Introduction

We've now seen the idea of Entropy crop up in several different places. Today, I'll summarize those and delve into an extension of the idea in information theory and coding.

2 Where We've Seen It

It's remarkable how much our understanding of entropy has grown. We started with the vague idea of state function, invoked to explain why heat will always flow from a warm body to a cold body.

2.1 Thermodynamics

We first defined entropy with the assertion: "There exists an additive state function known as the entorpy S that can never decrease in an isolated system."

$$\text{Entropy} \equiv S(E,V,N) \quad \text{and} \quad \Delta S \leftarrow 0. \tag{1}$$

We went on to use this state function to define several thermodynamic quantities:

$$\frac{1}{T} \equiv \left(\frac{\partial S}{\partial E}\right)_{V,N}, \quad \frac{P}{T} \equiv \left(\frac{\partial S}{\partial V}\right)_{E,N}, \quad \frac{\mu}{T} \equiv \left(\frac{\partial S}{\partial N}\right)_{E,V}. \tag{2}$$

In the case where a process is quasistatic, we can extend these relationships to show that

$$dS = \frac{dQ}{T}. \tag{3}$$

We also saw the entropy state function crop up in the fundamental Thermodynamic relation:

$$dE = TdS - PdV + \mu dN. \tag{4}$$

2.2 Statistical Mechanics

When looking at systems statistically, we saw entropy creep up again, this time as a measure of the number of accessible microstates for a given system. One formulation of this is given by the Boltzmann relation:

$$S = k \ln \Omega. \tag{5}$$

When we are given a probability distribution rather than explicit layout of the microstates for a system, an equivalent form is given in the Gibbs definition:

$$S = -k \sum_{N} -P_s \ln P_s. \tag{6}$$

Remarkably, these different formulations all agree, and we have identified the entropy quantity in each as the same.

3 Entropy and Information

Another area where we encounter entropy is in the process of encoding information. To get a handle on the role entropy plays in this process, let's first identify a problem in information transformation: loss and redundancy. In order to most efficiently transmit some amount of information, we want to communicate as little as possible while still ensuring that all information is transmitted. If we wish to transmit a six letter word electronically, we could encode images of each letter in sequential order, or use morse code, or use some string of bits to convey the information. The first method would have a lot of redundancy, and convey way more information that is necessary to interpret the message, while the second and third compress the information.

3.1 A quick Experiment

Imagine that we want to generate a sequence of random letters independently. Is there a limit to how few bits we can use to encode the information? YES! In the mid 1950's, Shannon exactly quantified the answer using information entropy:

$$H \equiv -\sum_s P_s \log_2 P_s. \tag{7}$$

This means that the smallest average number of bits needed to represent the output is given by H. In other words, it is the average length, in bits, of the string of code attached to a given letter. Sayood gives a good introduction to the idea with a little thought experiment:

Imagine we have a four letter alphabet, and we want to encode a message in bits. Call the letters A_1, A_2, A_3, and A_4. If we assume that each letter has an equal likelihood of being used, then we can encode each letter using two bits. The image below shows an algorithm that could be used for such a scheme.

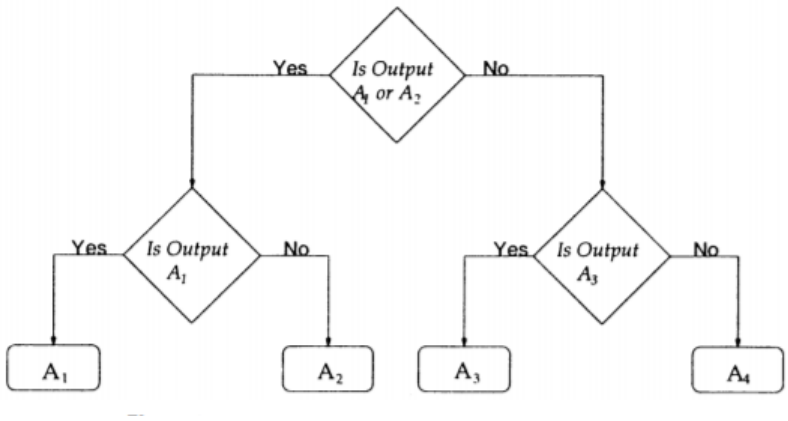

Figure 1: An Algorithm for encoding a four letter alphabet. Yes and no are the bits, and it takes two bits to encode each letter. Taken from *Data Compression* by Khalid Sayood.

The entropy for this system is $-4 \times (.5) \log_2(.5) = 2$ which is the average number of bits needed to encode a letter.

3.2 Another quick Experiment

What can we do to improve this number? Perhaps we can make the probabilities of different letters different. After all, if we choose a random letter in a random English word, some letters are more likely to appear than others. Using $P(A_1) = 1/2$, $P(A_2) = 1/4$, and $P(A_3) = P(A_4) = 1/8$, the algorithm is shown in the figure below. The information entropy for this process is 1.75 bits–an improvement over given each letter an equal probability

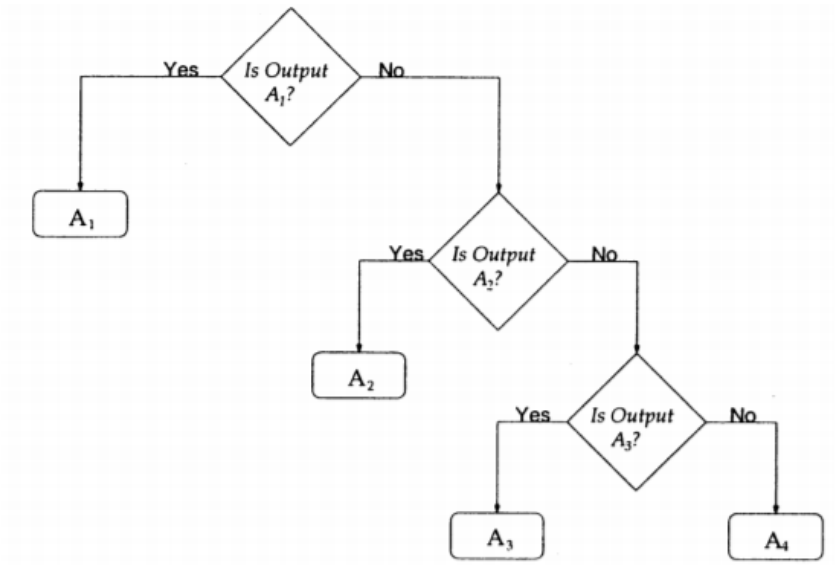

Figure 2: A decision tree for nonequal probabilities. Taken from *Data Compression* by Khalid Sayood

3.3 Huffman Coding

Using a similar idea, Huffman developed a strategy for encoding information based on three rules:

a. The codes corresponding to high probability letters could not be longer than the codes corresponding to lower probability letters

b. The two lowest probability letters have to have code words of the same length

c. The two lowest probability letters have codes that are the same except for the last bit.

In other words, each 'yes' bit represents a letter with the most likely letter corresponding to a the first bit as 'yes', the second most likely corresponding to a first bit 'no' and a second bit 'yes', and so on. A schematic for five letters is shown below.

The Huffman scheme is one of the most efficient methods for encoding random letters. On average, the redundancy is only one bit bigger than the information entropy. And in the limit of large messages, both converge to very small values.

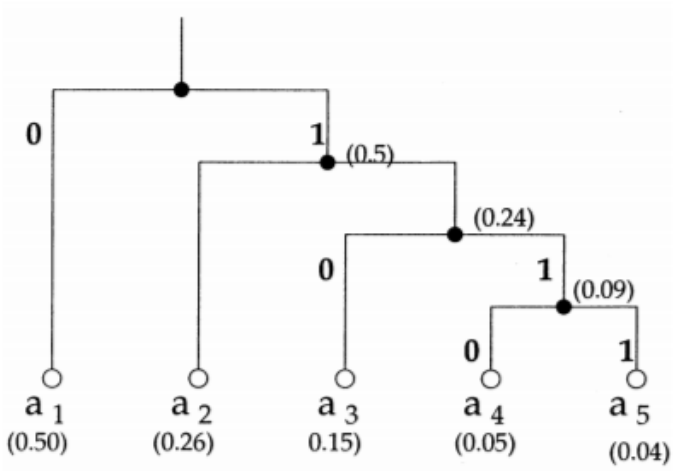

Figure 3: The Huffman Coding Scheme. Taken from *Data Compression* by Khalid Sayood.

3.4 Developing even better codes

Of course, all these methods hinge on the idea that the individual letters chosen independently of each other. In reality, the pieces of most information we wish to convey are not independent. A u will be much more likely to occur after at q, etc. Using these dependencies, we can construct probabilistic models which capture the dependent nature of language.

The Rabanni Jones reading gives a good description of how these models work. But I'll provide a cursory introduction. The most common model is the *Markov* model, in which the probability of a given letter being used depends on the preceding text. In the first order model, the probability is only dependent on the last letter used, and the entropy is given by

$$H(S) = \sum_S \sum_S p(s_i, s_j) \log_2 \left(\frac{1}{p(s_i|s_j)} \right), \tag{8}$$

where S is the number of characters in the alphabet, and the s_n are specific letters. With more specific probability models, fewer bits are needed to encode letters and words, and the information entropy will decrease. Both the Sayood and Rabanni Jones readings give much more detailed treatments of this topic if you're interested.