Sophomore College
Mathematics of the Information Age
Entropy

Let's recall the fundamental notions of information and entropy. To repeat from the first class, Shannon's emphasis is on *selecting* a given message from a collection of possible messages. The set of all possible messages is called the source, and one might say that 'information' (before transmission, at least) is less about what you *do* say than it is about what you *can* say. There are various legends about Shannon's choice of the term 'entropy' to describe the average information of a source. The one that's repeated most often is John von Neumann's advice to 'Call it entropy. No one knows what entropy is, so if you call it that you will win any argument.' This little pointer on wrangling notwithstanding, there are many similarities in form and substance between Shannon's definition of entropy and the term as it is used in physics, especially in thermodynamics and statistical mechanics.

For the definitions: Consider a source consisting of N symbols $s_1, \ldots, s_N$, each of which has an associated probability $p_1, \ldots, p_N$. The *information* of the message s_n is then

$$I(s_n) = \log \frac{1}{p_n}$$

in bits. Remember that we always use logs to base 2.[1] The *entropy* is the average information of the source, that is, it is the weighted average of the information of each of the symbols according to their respective probabilities:

$$H = \sum_{n=1}^{N} p_n I(s_n) = \sum_{n=1}^{N} p_n \log \frac{1}{p_n}.$$

The units of H are bits/symbol.

So that's why logs add ... Here's a nice bit of intuition bundling up information, logs, and old, familiar identities. Suppose that a message A occurs with probability p_1 and another message B occurs with probability p_2. What is the information associated with receiving *both* messages? Intuitively, we feel that the informations should *add*, that is, the information associated with receiving both messages (our surprise at receiving both) should be the sum of the information associated with each *provided they are independent*. You wouldn't say 'It's raining outside' and 'The sidewalk is wet' are independent messages, but you would say that 'It's raining outside' and 'I

[1] To compute, you can relate the log base 2 to the log base 10 (for example) by the formula

$$\log_2 x = \frac{\log_{10} x}{\log_{10} 2} = \frac{\log_{10} x}{.3010}.$$

play the trombone' are independent. One fundamental idea in probability is that the probability of two independent events occuring is the product of the probabilities of each. (Think of coin tossing, or rolling a die, for common examples.)

Thus, as we've written it, the probabilities of both messages occuring is $p_1 p_2$, and the information associated with that is

$$I(A \text{ and } B) = \log \frac{1}{p_1 p_2}.$$

But, we all know, the log of a product is the sum of the logs, and hence

$$I(A \text{ and } B) = \log \frac{1}{p_1 p_2} = \log \frac{1}{p_1} + \log \frac{1}{p_2} = I(A) + I(B).$$

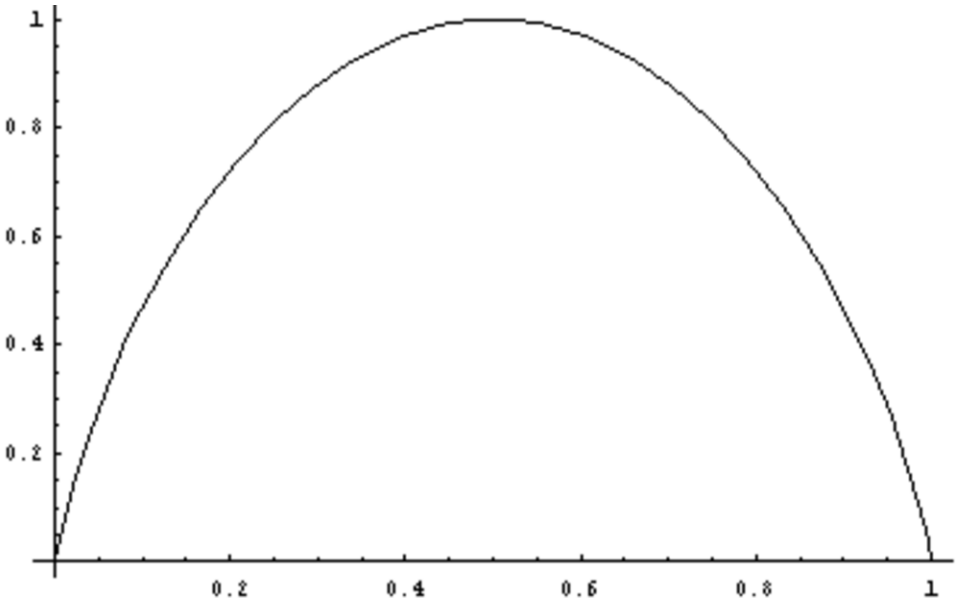
Turning this around, if you ever wanted an intuitive reason *why* the log of a product should be the sum of the logs, think of this – that the 'information of the sum is the sum of the informations'.

An important special case of entropy

Let's take a special case of an entropy calculation. Suppose we have a source consisting of just two messages A and B. Suppose A occurs with probability p. Since only A or B can occur, and one of them must occur, the probability that B occurs is then $1 - p$. Thus the entropy of the source is

$$H(\{A, B\}) = p \log \frac{1}{p} + (1 - p) \log \frac{1}{1 - p}$$

How does this look as a function of p, for $0 < p < 1$, and for the limiting cases $p = 0$ and $p = 1$? Here's a graph:



At $p = 0$ or $p = 1$ the entropy is zero, as it should be, and the entropy is a maximum at $p = 1/2$, also as it should be. The maximum value is 1. These facts can be verified with some calculus – I won't do that.

More on the meaning of entropy after we look briefly at a few sample problems that put entropy to work.

Fool's gold Consider the following 'weighing problem'.[2] You have 27 apparently identical gold coins. One of them is false and lighter but is virtually indistinguishable form the others. You also have a balance with two pans, but without weights. Accordingly, any measurement will tell you if the loaded pans weight the same or, if not, which weighs more. How many measurements are needed to find the false coin?

- Before solving the problem, find a lower bound for the number of weighings.

Here's an information theoretic way of analyzing the problem. Recall that we have 26 genuine gold coins, one lighter, fake, gold coin and a balance for comparing weights. The question is how many weighings are necessary to find the fake.

A weighing – putting some coins on one pan of the balance and other coins on the other pan – produces one of three equally likely outcomes:

1. The pans balance
2. The left pan is heavier
3. The right pan is heavier

The information associated with any weighing is thus $\log 3$ bits, and so n weighings give $n \log 3$ bits. Furthermore, the amount of information we need to select one of 27 items is $\log 27$ bits. Thus we want to choose n so that

$$n \log 3 \geq \log 27 = 3 \log 3, \quad \text{so} \quad n \geq 3.$$

This analysis suggests we should be able to find the fake coin with three weighings.

Here's a way to weigh. Divide the coins into three piles of 9 coins each and compare two piles. If the scale balances then the fake coin is in the remaining pile of 9 coins. If one pan is heavier, then the other pan has the fake coin. In any event, with one weighing we now can locate the fake coin among a group of nine coins. Within that group, weigh three of those coins against another three. The result of this weighing isolates the fake coin in a group of three. In this last group, weigh two of the coins again each other. That weighing will determine which coin is fake.[3]

Place your bets Here's an example of applying Shannon's entropy to a two message source where the probabilities are not equal.[4] Suppose we have a roulette wheel with 32 numbers. (Real roulette wheels have more, but $32 = 2^5$ so our calculations will be easier.) You play the game by betting on one of the 32 numbers. You win if your number comes up and you lose if it doesn't. In a fair game all numbers are equally likely to occur.

If we consider the source consisting of the 32 statements 'The ball landed in number n', for n going from 1 to 32, then all of those statements are equally likely and the information associated with each is

$$I = \log \frac{1}{\frac{1}{32}} = \log 32 = 5.$$

[2]This problem comes from *A Diary on Information Theory*, by A. Rényi.

[3]Note that another approach in the second step might be lucky. Namely, in a group of nine coins, one of which is known to be bad, weigh four against four. If the scale balances then the left over coin is the fake, and we've found it with two weighings. But if the scale doesn't balance then we're left with a group of four coins, and it will take two more weighings to find fake coin.

[4]This example is from *Silicon Dreams* by Robert W, Lucky.

(It takes 5 bits to code the number.)

But you don't care about the particular number, really. You want to know if you won or lost. You are interested in the two message source

$$\begin{aligned} W &= \text{You have won} \\ L &= \text{You have lost} \end{aligned}$$

What is the entropy of this source? The probability of A occuring is $1/32$ and the probability of B occuring is $31/32 = 1 - 1/32$. So the entropy of the source is

$$\begin{aligned} H &= \frac{1}{32}\log 32 + \frac{31}{32}\log\frac{32}{31} \\ &= .15625 + 0.04437 = .20062 \end{aligned}$$

That's a considerably smaller number than 5; L is a considerably more likely message. This tells us that, on average, you should be able to get the news that you won or lost in about 0.2 bits.

What does that mean, 0.2 bits? Doesn't it take at least one bit for each, say 1 for 'You have won' and 0 for 'You have lost'? Remember, the notions of '1/32 probability of winning' and '31/32 probability of losing' only make sense with the backdrop of playing the game many times. Probabilities are statements on percentages of occurences *in the long run.* Thus the question, really, is: If you have a *series* of wins and losses, *e.g.* $LLLLLWLLLLLLWWLLLL\ldots$, how could you code your record so that, on average, it takes something around 0.2 bits per symbol?

Here's one method. We track the won/loss record in 8 event blocks. It's reasonable to expect that 8 events in a row will be losses. As we look at the record we ask a yes/no question: 'Are the next 8 events all losses?'. If the answer is yes we let 0 represent that block of 8 losses in a row. So every time that this actually happens in the total record we will have resolved 8 outcomes with a single bit, or a single outcome (in that block) with 1/8 of a bit. If the answer to the yes/no question is 'no', that is, if the next 8 events were not all losses then we have to record what they were.

Here's the scheme:

- 0 means the next 8 results are $LLLLLLL$;
- $1XXXXXXXX$ mean the 8 X bits specify the actual record. So, for example, 100010000 means the next 8 results are $LLLWLLLL$
- So, starting the record at the beginning, we might have, say, 0 0 100010000 0, meaning that the first 16 events are losses, the next 8 are LLLWLLLL, the next 8 are losses, and so on.

How well does this do? Look at a block of eight win-loss outcomes. The probability of a single loss is 31/32 and the probability of 8 straight losses is $(31/32)^8 = .776$ (the events are independent, so the probabilities multiply), and it takes 1 bit to specify this. The probability that the 8 results are *not losses* is $1 - .776 = .224$, and, according to our scheme it takes 9 bits to give these results. Thus, on average, the number of bits needed to specify 8 results are

$$.776 \times 1 + .224 \times 9 = 2.792.$$

Finally, the average number of bits *per result* is therefore

$$2.792/8 = .349.$$

That's much closer to the .2 bits given by calculating the entropy.

Let me raise two questions:

- Can we be more clever and get the average number of bits down further – and can we even be so sneaky as to break the entropy of 0.2 bits per symbol?
- Is this just a game, or what?

By way of example, let me address the second question first.

An example: Run length coding Don't think that the strategy of coding blocks of repeated symbols in this way is an idle exercise. There are many examples of messages when there are few symbols, and probabilities strongly favor one over the other thus indicating that substantial blocks of a particular symbol will occur often. The sort of coding scheme that makes use of this. much as we did above, is called *run length coding*. Here's an example of how run length coding can be used to code a simple image efficiently. I say 'simple' because we'll look at a 1-bit image – just black or white, no shades of gray.

The idea behind run length coding, and using it to compress the digital representation of this image, is to code by blocks, rather than by individual symbols. For example

- $0X$, where X is a binary number, means that the next X pixels are black.
- $1Y$ means that the next Y pixels are white.

There are long blocks of a single pixel brightness, black or white, and coding blocks can result in a considerable savings of bits over coding each pixel with its own value.

You shouldn't think that 1-bit, black-and-white images are uncommon or not useful to consider. You wouldn't want then for photos – we certainly didn't do the palm tree justice – but text files are precisely 1-bit images. A page of text has letters (made up of black pixels) and white space in between and around the letters (white pixels) Fax machines, for example, treat test documents as images – they don't recognize and record the symbols on the page as characters, only as black or white pixels. Run length coding is a standard way of compressing black-and-white images for fax transmission.[5]

[5] Actually, fax machines combine run length coding with Huffman coding, a topic we'll discuss later.

Run length coding can also be applied to grayscale images. In that case one codes for a run of pixels all of the same brightness. Typical grayscale images have enough repetition that the compression ratio is about 1.5 to 1, whereas for black-and-white images it may be as high as 10 to 1.

How low can you go: The Noiseless Source Coding Theorem

Now let's consider the first question I raised in connection with the roulette problem. I'll phrase it like this: Is there a limit to how clever we can be in the coding? The answer is yes – the entropy of the source sets a lower bound for how efficient the coding of a message can be. This is a really striking demonstration of how natural entropy is.[6] The precise result is called Shannon's *Noiseless Source Coding Theorem*.

Before stating and proving the result, we need a few clarifications on coding. For simplicity we work exclusively with binary codes. Coded messages are then represented by binary numbers, *i.e.* by strings of 0's and 1's, but it's helpful to break this down. The *source symbols* (uncoded) are the things to be sent. (The symbols 'stand for something', as in coding the letters of the (usual) alphabet, or brightness of pixels, *etc.*) The 0 and 1 are the *code alphabet*.[7] Each source symbol is encoded as a specified strings of 0's and 1's – call these the *coded source symbols*.[8]

The main property we want of any code is that it be *uniquely decodable*. This is what you think it is: It should be possible to parse any codeword unambiguously into source symbols. For example coding symbols s_1, s_2, s_3 and s_4 via

$$\begin{aligned} s_1 &= 0 \\ s_2 &= 01 \\ s_3 &= 11 \\ s_4 &= 00 \end{aligned}$$

is *not* a uniquely decodable code. The received message 0011 could be either $s_3 s_4$ or $s_1 s_1 s_3$. We'll look at the finer points of this later, but we will *always* assume that a code is uniquely decodable.

Finally, let's talk about entropy, coding, frequency counts and probabilities. The elements of the source have associated probabilities p_i. The entropy is defined in terms of these probabilities – coding hasn't happened yet. A message is formed by selecting source symbols according to their respective probabilities. Now code the source, and hence code the message. Suppose we have a message using a total of M coded source symbols. If the i'th source symbol occurs m_i times in the message (its frequency count) then the probability associated with the i'th coded source symbol is the fraction of times it does occur, *i.e.*

$$p_i = \frac{m_i}{M}.$$

The entropy of the source is

$$H(S) = \sum_i p_i \log \frac{1}{p_i} = \sum_i \frac{m_i}{M} \log \frac{M}{m_i}.$$

[6] 'Natural' if you go through the hard work we're about to go through. It's a valuable exercise in understanding the significance of a definition and how it applies. It's mathematics.

[7] Thus we consider *radix* 2 codes. If the symbols in the code are drawn from an alphabet of r elements then he code is referred to as *radix* r.

[8] I'm not guaranteeing that this terminology is absolutely standard. We could make this more mathematically formal by defining a *code* to be a mapping from a 'source domain' to a set of binary numbers.

Next, let A_M be the average of the lengths of all the coded source symbols in the message; so A_M is a weighted average of the lengths of the individual coded source symbols, weighted according to their respective probabilities. Of course there are different ways of coding the source, and efficient coding of a message means making A_M small. A code is *optimal* if A_M is as small as possible. How small can it be? One version of Shannon's answer is:

> **Noiseless Source Coding Theorem** Let S be a source and let $H(S)$ be the entropy of S. In the notation above,
>
> $$A_M \geq H(S).$$

This is called the noiseless source coding theorem because it deals with codings of a source before any transmission (no noise has yet been introduced that might corrupt the message).

We're going to prove this by mathematical induction on M. To recall the principle of induction, we: (1) Establish that the statement is true for $M = 1$, and (2) Assume that the statement is true for all numbers $\leq M$ (the induction hypothesis) and deduce that it is also true for $M + 1$. The principle of mathematical induction says that if we can do this then the statement is true for *all* natural numbers.[9]

The first step is to verify the statement of the theorem for $M = 1$. When $M = 1$ there is only one message and the entropy $H = 0$. Any coding of the single message is at least one bit long (and an optimal coding would be exactly one bit, of course), so $A_1 \geq 1 > H$. We're on our way.

The induction hypothesis is that

$$A_M \geq H$$

for an optimal coding of a message with M coded source symbols. Suppose we have a source S with $M + 1$ and an optimal coding for S. From this new source we're going to come up with two sources S_0 and S_1 each with fewer than M coded source symbols, and we'll apply the induction hypothesis to S_0 and S_1.

Split S into classes:

$$\begin{aligned} S_0 &= \text{all source symbols in } S \text{ whose coded source symbol starts with a 0} \\ S_1 &= \text{all source symbols in } S \text{ whose coded source symbol starts with a 1} \end{aligned}$$

Suppose that the frequency count of the i'th coded source symbol in S is m_i. Then the number of coded source symbols in the coding of S_0 and S_1, respectively, are computed by

$$M_0 = \sum_{S_0} m_i, \quad \text{and} \quad M_1 = \sum_{S_1} m_i,$$

where the sums go over the coded source symbols in S which are in S_0 and S_1. Now $M_0 + M_1 = M + 1$, and neither M_1 nor M_2 can be zero. (If, say, $M_0 = 0$ then no coded source symbols of S start with a 0, *i.e* all coded source symbols start with a 1. But then this leading 1 is a wasted bit in coding the source symbols, contrary to our assumption that the coding is optimal.) Thus we can say that

$$1 \leq M_1, M_2 \leq M,$$

and we'll be set up to apply the induction hypothesis.

Now drop the first bit from the coded source symbols coming from S_0 and S_1, *i.e.* consider S_0 and S_1 as two new sources with this coding. Then, first of all, the average of the lengths of

[9] See me if you want practice with induction.

the coded source symbols in S is given by the weighted average of the lengths of the coded source symbols in S_0 and S_1 *plus* 1 – adding the one comes from the extra bit that's been dropped, namely

$$A_{M+1} = 1 + \frac{M_0}{M+1}\sum_{S_0}\frac{m_i}{M_0}\text{length}_i + \frac{M_1}{M+1}\sum_{S_1}\frac{m_i}{M_1}\text{length}_i.$$

The sums on the right hand side are larger than if we use an *optimal* code for S_0 and S_1, and for an optimal code we can apply the induction hypothesis to bound the average length by the entropy. That is,

$$\begin{aligned} A_{M+1} &\geq 1 + \frac{M_0}{M+1}A_{M_0} + \frac{M_1}{M+1}A_{M_2} \\ &\geq \frac{M_0}{M+1}\sum_{S_0}\frac{m_i}{M_0}\log\frac{M_0}{m_i} + \frac{M_1}{M+1}\sum_{S_1}\frac{m_i}{M_1}\log\frac{M_1}{m_i}. \end{aligned}$$

Now several algebraic miracles occur:

$$\begin{aligned} &1 + \frac{M_0}{M+1}\sum_{S_0}\frac{m_i}{M_0}\log\frac{M_0}{m_i} + \frac{M_1}{M+1}\sum_{S_1}\frac{m_i}{M_1}\log\frac{M_1}{m_i} = \\ &1 + \sum_{S_0}\frac{m_i}{M+1}\log\frac{M_0}{m_i} + \sum_{S_1}\frac{m_i}{M+1}\log\frac{M_1}{m_i} \\ &= 1 + \sum_{S_0}\frac{m_i}{M+1}\log\frac{1}{m_i} + \sum_{S_1}\frac{m_i}{M+1}\log\frac{1}{m_i} + \frac{M_0}{M+1}\log M_0 + \frac{M_1}{M+1}\log M_1 \\ &= \sum_{S}\frac{m_i}{M+1}\log\frac{M+1}{m_i} + 1 + \log\frac{1}{M+1} + \frac{M_0}{M+1}\log M_0 + \frac{M_1}{M+1}\log M_1 \\ = \quad &H(S) + 1 + \log\frac{1}{M+1} + \frac{M_0}{M+1}\log M_0 + \frac{M_1}{M+1}\log M_1 \end{aligned}$$

To complete the proof of the theorem, that $A_{M+1} \geq H(S)$, we have to show that

$$1 + \log\frac{1}{M+1} + \frac{M_0}{M+1}\log M_0 + \frac{M_1}{M+1}\log M_1 \geq 0.$$

Write

$$x = \frac{M_0}{M+1}, \quad \text{and hence} \quad 1 - x = \frac{M_1}{M+1},$$

since $M_0 + M_1 = M + 1$. Then what we need is the inequality

$$1 + x\log x + (1-x)\log(1-x) \geq 0, \quad \text{for} \quad 0 < x < 1.$$

This is a calculus problem – the same one as finding the maximum of the entropy of a two-symbol source. The function has a unique minimum when $x = 1/2$ and it's value there is zero.

The induction is complete and so is the proof of Shannon's theorem.