Categorical Cross Entropy

John Purcell

November 29, 2021

1 Units of Information

Can we measure a quantity of information?

Whether we consider written text, verbally transmitted messages, or some other form of information, clearly some entities convey more information than others. An elaborate description is more informative (and more surprising) than a "yes or no" answer.

For a message to contain information at all, it must be somewhat surprising. An endlessly repeated symbol does not convey any information because it is completely predictable, and not surprising. If someone asks you questions and you always answer "yes" no matter what the question, it becomes apparent that rather than giving them information, you are simply blanking them; refusing to answer truthfully.

The smallest possible unit of information is in fact the infamous "yes or no" answer. This can be thought of as a binary variable: an entity that can take one of two possible states ("yes", or "no", in this case.) We can say that a binary variable encodes or contains 1 bit of information.

If you ask a question that can be answered with "yes" or "no" and a colleague answers that question with "yes" or "no", they have transmitted one bit of information to you.

In general, we can choose to measure information in bits. Other possible units also exist, but here we will consider only bits.

2 Symbol Spaces

When we transmit messages, we use symbols to encode data. These symbols may be written characters, sounds, multi-coloured lights or whatever. The set of possible symbols that we have decided to use to encode a message is known as the "symbol space".

Suppose we use a symbol space with a size of four. In other words, we decide to transmit messages using four possible symbols, e.g. A, B, C or D.

How many bits of information can one of these symbols encode?

First let us assume the symbols occur with equal probability in your messages. Every message is as likely to contain a B as a D, etc.

Clearly we could encode these symbols in Unicode and transmit that, but since we have only four possible symbols to transmit, that would be very inefficient. We do not need multiple bytes for each symbol. We can greatly "compress" our messages by using an alternative encoding scheme.

In this simple case, one such way to figure out how many bits we need per symbol encoded is to map the symbols to binary numbers.

A: 00

B: 01

C: 10

D: 11

Then it becomes obvious that each symbol actually requires a minimum of two bits to encode it, and so conveys or contains two bits of information. If further bits are used to encode symbols, these bits would really be superfluous.

A message consisting of a string of 20 of these symbols would then require an average of $20 \times 2 = 40$ bits to encode it. If such a message is received, an average of 20 bits of information would be received.

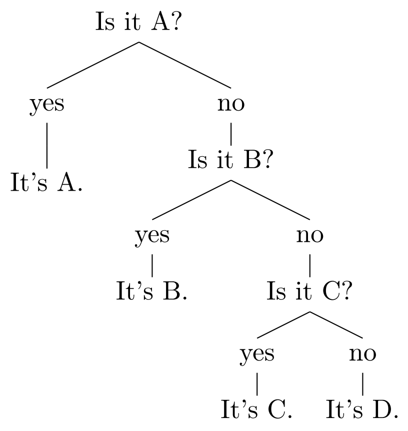
Another way to think about the problem is to imagine playing the following game. A collaborator constructs a message and you attempt to guess the message, one symbol at a time. You do this by asking questions that can be answered with only "yes" or "no".

Observe that every time you receive a "yes" or "no" answer, you receive one bit of information. Therefore by figuring out what questions to ask so as to determine each symbol as efficiently as possible, we also figure out how many bits it takes to optimally encode such a message, and so we determine how many bits of information are really contained in such a message.

The message could actually be stored as a series of 0's and 1's that represent answers to some predetermined question strategy that specify the message.

We *could* ask these questions to determine each symbol: "Is it A?", (if not) "Is it B?", (if not) "Is it C?".

The sequence of questions could be represented by the following tree diagram.



What is the average number of bits required to encode one symbol via this method? In other words, what is the average number of questions that has to be asked to determine a single symbol?

It requires one bit (one question) to specify 'A', two bits to specify 'B', and three to specify a 'C' or 'D'.

We will denote an unspecified symbol by X. We can think of X as representing a variable that can take the value A, B, C or D.

The average number of bits required to encode or transmit a symbol, we call the *Shannon entropy*, or just *entropy*, denoted by $H(X)$. So we say the entropy of the variable X is $H(X)$.

To find the average number of bits per symbol here, since each symbol is equally probable, we can add up the number of bits required to specify each symbol and divide by the number of symbols.

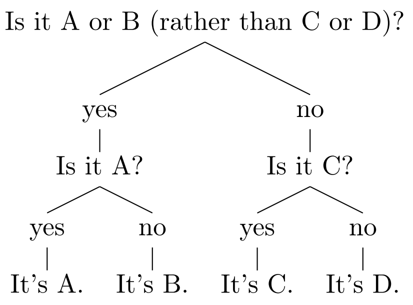
$$H(X) = \frac{1+2+3+3}{4} = 2.25$$

Equivalently, we can multiply the probability of each symbol by the number of yes-or-no questions (and therefore bits) required to determine each symbol.

$$H(X) = (0.25 \times 1) + (0.25 \times 2) + (0.25 \times 3) + (0.25 \times 3) = 2.25$$

However, this conclusion is **incorrect**, because we haven't asked an optimal set of questions. We can encode the symbols more efficiently, as we've already seen. We already know that the number of bits required per symbol is actually 2, not 2.25.

A better way to approach determining each symbol is to repeatedly divide the symbol space in two, and ask a yes-or-no question about the two halves, like this:



It now requires 2 answers (2 bits) to determine each symbol, so:

$$H(X) = (0.25 \times 2) + (0.25 \times 2) + (0.25 \times 2) + (0.25 \times 2) = 2$$

The entropy of the variable X (the average number of bits of information per symbol) is in fact 2 bits.

How many bits of information are encoded per symbol, on average, if the symbol space has eight equally-probable symbols, ABCDEFGH?

We can subdivide this into halves three times before getting down to individual symbols: ABCD-EFGH, AB-CD-EF-GH, A-B-C-D-E-F-G-H. A question tree for these symbols would have three levels of questions instead of two levels as above. The number of questions we would have to ask to determine any particular symbol would be 3 questions, and so each symbol would convey 3 bits of information.

In general, for N equally probable symbols, we can determine the number of bits per symbol by taking the binary log of N.

$$H(X) = \log_2(N)$$

(For N equally probable symbols)

3 Unequally Probable Symbols

The entropy of an "event" (such as transmitting a symbol or flipping a coin) or a variable depends on the probabilities of the symbols in the symbol space.

Remember, information is *surprise*, so entropy is a measure of average surprise per symbol.

Suppose a message is written down in a book using only two symbols: AB. Nearly all of the book consists of the symbol 'A', while very occasionally a B is found. Clearly most of the book is not surprising at all. It's actually very repetitive. To transmit the contents of this book via answers to a set of mutually-agreed questions, it would be very suboptimal to keep asking "Is it A?", "Is it A?", "Is it A?" ... since nearly every symbol *is* 'A'.

Instead we could ask something like "Are the next ten characters all A's?". Only when we receive the answer "no" do we need to ask further questions to find out which ones are 'B', and this might only happen once or twice in the entire book.

Even if the book should be 1000 pages long, I might encode the entire thing using this sentence, for example: "156,345 A's followed by a B followed by 834,324 A's".

While this encoding is not optimal, clearly the low entropy (lack of surprise/repetitive nature) of the book's text has allowed us to compress all 1000 pages massively.

We now turn to the question of how to determine the information associated with a particular symbol and the average information per symbol (entropy) in cases like this, where the symbols in the symbol space are not equally likely to be encountered in messages.

Imagine that a message is encoded using the four symbols ABCD, and these symbols are found with the following probabilities in messages:

A: P(A)=0.125

B: P(B)=0.375

C: P(C)=0.25

D: P(D)=0.25

'A' occurs in messages with a probability of 0.125; or in other words, on average every 1 in 8 symbols is an 'A'.

How much information does the symbol 'A' actually encode? To answer this question, we only need to know how often 'A' occurs in messages; it's this that determines how surprising an 'A' is, and so this is what determines how much information an 'A' conveys.

The question is really the same as asking how much information an 'A' encodes if the 'A' is part of an 8-symbol set of 8 equally-probable symbols, and we know that the answer to that is 3 bits — since in that case it would take 3 yes-no answers to specify any of the characters, including 'A'.

The probabilities and quantities of other symbols are irrelevant to answering this question. Only the probability of 'A' occurring is relevant to calculating the information encoded by an 'A'.

In general, the quantity of information $I(S)$ (measured in bits) associated with a symbol S occuring with probability $P(S)$ is given by the following equation.

$$I(S) = \log_2(\frac{1}{P(S)}) = -\log_2(P(S))$$

We can now determine how much information (in bits) is associated with each of the symbols in our 4-symbol space, ABCD, given the above probabilities for each symbol.

$I(A) = -\log_2(0.125) = 3$

$I(B) = -\log_2(0.375) = 1.415$

$I(C) = -\log_2(0.25) = 2$

$I(D) = -\log_2(0.25) = 2$

To determine the entropy of a variable or event X which takes the values of the above symbols with the stated probabilities, or loosely speaking, to determine the entropy of a message encoded using these symbols, we need to calculate the *average* quantity of information conveyed by a symbol in this encoding scheme.

To do this, we add up each of these quantities of information multiplied by the probability of that symbol actually occurring.

So the general formula for entropy is:

$$H(X) = \sum_i -P(x_i)\log_2(P(x_i))$$

Where X is a variable that can take the values x_1, x_2, x_3 ...

Applying that to our particular problem here:

$$H(X) = (0.125 \times 3) + (0.375 \times 1.415) + (0.25 \times 2) + (0.25 \times 2) = 1.9$$

We can see that an average of only 1.9 bits of information is now conveyed per symbol, using four symbols that occur with the stated probabilities. It's possible to use the unequal probabilities of the symbols to guess which symbol will occur next with slightly more reliablity than if the symbols were equally probable. This reduces your average degree of surprise, and so has to be said to reduce the amount of information encoded per symbol.

An optimal compression scheme could in theory compresses messages encoded using these four symbols with these probabilities so that only 1.9 bits of data on average were needed per symbol of the original message.

Various estimates have been made of the entropy of the English language — or in other words, the average amount of information gained when you receive a single letter of a message written in English. Often this is said to be around 1.14 bits, which suggests that arbitrary messages written in English could be compressed so that they require only 1.14 bits of data per character on average. In practice this compression limit has not been reached by practical compression techniques.

4 Cross Entropy

Suppose we devise an optimal encoding for a symbol space based on an assessment of the probabilities of the symbols in that space, but then we send messages with the symbols having a different probability distribution.

This is extremely likely to happen. For instance, suppose we manage to devise an optimal encoding for English text so that on average only 1.14 bits are used per character. Then we send messages using this encoding strategy. It's extremely unlikely that the symbols in our actual messages will occur with the exact same probability as the probabilities that we assessed by examining some large representative sample of English text. This means that even with an optimal encoding strategy based on a large sample of text, it is still likely to take a greater number of bits in practice to send messages.

Consider a set of 4 symbols that occur with the following probabilities in some large sample of messages:

A: 0.125

B: 0.125

C: 0.25

D: 0.5

We will represent these probabilities as Q(X). So $Q(A) = 0.125$, etc.

Using this information we calculate the information contained in each symbol as following:

A: 3 bits

B: 3 bits

C: 2 bits

D: 1 bit

However, suppose when we send a message, our symbols actually occur with the following probabilities:

A: 0.25

B: 0.5

C: 0.125

D: 0.125

We will represent these probabilities as $P(X)$, so $P(A) = 0.25$, etc.

The average number of bits per symbol in our message is then given by the following formula:

$$H(P,Q) = -\sum_i P(x_i)\log_2(Q(x_i))$$

We calculate this to be:

$$(0.25 \times 3) + (0.5 \times 3) + (0.125 \times 2) + (0.125 \times 1) = 2.625 \text{ bits}$$

This is known as the cross entropy betweeen the probability distributions P and Q.

It can be thought of as measuring the average number of bits required to transmit, encode or identify a symbol if symbols that occur with probability P (their true probability distribution) are optimally encoded using the theoretical probability distribution Q.

We can use cross entropy to compare two probability distributions. The higher the cross-entropy, the worse the match between the two distributions.

5 Applying Cross Entropy to Neural Networks

We can use cross entropy to evaluate the output of a neural network. Suppose we have a neural network which is trained to recognise the handwritten characters A, B, C or D. We present the network with an image and the network produces the following output:

$$\begin{pmatrix} 0.25 \\ 0.125 \\ 0.125 \\ 0.5 \end{pmatrix}$$

We can interpret this result as meaning that the network thinks there is a 0.25 probability that the presented image represents 'A', a 0.125 probability that it represents a 'B', and so on.

In reality, the image definitely represents one particular character. Let's suppose the image presented is an image of a handwritten 'B'. Then the true or "expected" probability distribution looks like this:

$$\begin{pmatrix} 0 \\ 1 \\ 0 \\ 0 \end{pmatrix}$$

This indicates that there is a zero probability the image is an 'A', a probability of 1 that it is a 'B', and a probability of zero that it is a C, or a D.

We can calculate the cross entropy of the two distributions. Since $P(X)$ is zero except for the value corresponding to 'B', $-P(X)\log_2(Q(X))$ is zero for all values except the value corresponding to 'B'. We need only calculate the value for 'B', which then reduces to $-\log_2(0.125) = 3$, since $P(B)$ is 1.

If the calculated probability distribution had matched the actual (true) probability distribution, we would instead have arrived at the value $-log_2(1) = 0$

When working with neural networks it's common to use the term *categorical cross entropy*, to differentiate from the formula needed when working with a binary classifier neural network (one that has only two outputs from the final layer). The latter special case of the cross entropy formula is known as *binary cross entropy*.

6 Summary

The information content in bits of a symbol given probability distribution $P(X)$ is:

$$\boxed{I(x_i) = -log_2(x_i)}$$

The entropy of a variable X with probability distribution $P(X)$ is:

$$\boxed{H(X) = -\sum_i P(x_i)\log_2(P(x_i))}$$

The cross entropy of the distribution $Q(X)$ relative to the distribution $P(X)$ is:

$$\boxed{H(P(X), Q(X)) = -\sum_i P(x_i)\log_2(Q(x_i))}$$