Introduction to Information Theory

Alliot Nagle

01/17/2019

Problem Setup and Overview

Let's say that you want to devise a way to order pizza using the least amount of effort possible. For simplicity, the pizza parlor on the other side of town only offers the option for you to create a custom pizza, and you may only choose one option from each category provided in the table below.

Diameter	**Crust**	**Sauce**	**Toppings**
10"	regular	marinara	mushrooms
12"	thin	pesto	olives
16"	deep dish	alfredo	chicken
18"			sausage

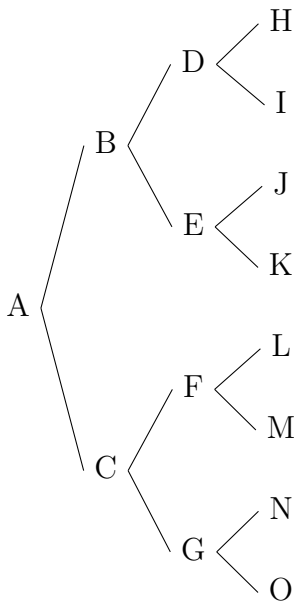
If you wanted to order your pizza that traditional way, you'd call the pizza parlor and you'd say something along the lines of "Hi, I'll have a 12-inch deep dish pizza with pesto and mushrooms." But this method isn't *efficient* because you're also providing *information* that doesn't directly relate to the pizza you want to order. Words like "Hi", "I'll", and "have", could be easily mitigated. If you wanted, you could pick up the phone and say "12-inch deep dish pizza, pesto, mushrooms," but you could *encode* this information in an even simpler way.

Let's take a step back, define some terms, and try to wrap our heads around the problem we are considering and the solution I want to propose. Everyone is familiar with the concept of *information*: it includes the ideas we communicate to each other, the data which traverses landlines and satellites before it reaches your computer, and the thematic elements and motifs embedded in music. Each of these are just examples in which information is communicated, transmitted, and conveyed. How that information is expressed depends on the way it is *encoded*, and all of these encodings depend on some set of *symbols*. For example, people communicate their thoughts and ideas using written text. But what is text? It's a collection of symbols which have meaning when the symbols are put together in a certain way. I am able to communicate to you through the words on this post by ordering a collection of symbols (the alphabet in this case) in a certain way to convey my thoughts. Music has it's own special form of encoding to communicate to the musician how to play the music, and the symbols inlcude pitches on the staff, time signatures, and clefs. Computers transmit data using a collection of binary digits.

Depending on context, we may want our encodings to be *efficient*. An efficient encoding is essentially an encoding which contains a maximal amount of information transmitted

or communicated per symbol. We will look at what an efficient encoding is in more detail later.

So, how is information defined so that we can determine the most efficient encoding for ordering your pizza? In 1948, a brilliant researcher named Claude Shannon determined how information can be measured, which marked the very beginning of the digital revolution. He coined the unit of measure for information the *bit*, not to be confused with *binary digit*. A bit is a continuous, real number and represents some amount of information. A binary digit is discrete, and can either be the value 0 or 1. Furthermore, binary digits may be used to encode information. For example, consider the graph below.



If you start at A and want to travel to M (let's assume that you don't possess the graph above while you are traveling); all you know is that you can only take a right or a left at each vertex. The result is a binary decision that can be represented with binary digits: a left turn is 0 and a right turn is 1. We can represent the series of turns needed to travel from A to M as 101. This interpretation is useful because our two symbols, 0 and 1, convey new information each time they're used. If directions were instead provided as *right left right*, we would need a total of nine symbols (r, i, g, h, t, l, e, f, and a symbol to represent space). Most of the information would be redundant or unnecessary.

Let's look in more detail to develop an intuition about the bit, and how it is a measure of information. Since each vertex requires a binary decision, there are a total of $2^3 = 8$ different paths to take. Which means that in order to travel from A to M without the use of the graph, you need $log_2(8) = 3$ bits of information. Using the binary digit encoding we devised above, this means that every binary digit in our sequence of 0s and 1s corresponds to one bit of information.

We can apply what we've just learned to our pizza problem. For example, since there are four choices of pizza diameters, two binary digits can be used to represent each diameter: 00 for 10", 01 for 12", 10 for 16", and 11 for 18." This same idea can be applied to the crusts, sauce, and toppings. The table below reflects a possible code for representing these different choices for pizza.

Diameter	Crust	Sauce	Toppings
10" (00)	regular (00)	marinara (00)	mushrooms (00)
12" (01)	thin (01)	pesto (01)	olives (01)
16" (10)	deep dish (10)	alfredo (10)	chicken (10)
18" (11)			sausage (11)

Since the same code is used to express the choices in each category, an ordering of the binary digits must be implied. The first two binary digits will represent the diameter, the next two binary digits represent the crust, etc. For example, 01101001 is the code for a 12" deep dish with alfredo sauce and olives. 00000000 is the code for a 10" regular crust with marinara sauce and mushrooms.

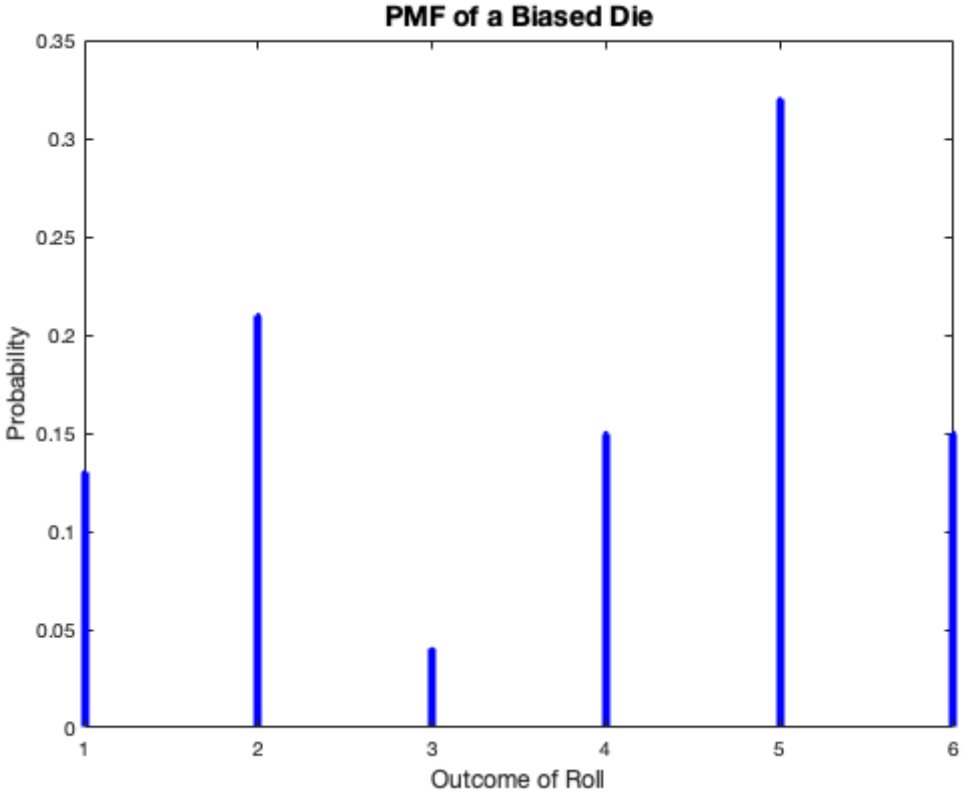
Unfortunately, you can't simply pick up the phone, call the pizza parlor, and say "10100110" and expect the employee to understand that you want a 16" deep dish with pesto and chicken." The problem is that they don't have a way to *decode* your message.

The figure below defines the basic system that is considered when problems in information theory are solved.

A *message* is generated ("12" pizza with regular crust..."), it is encoded (0100...) and then transmitted across the *channel*, which has some amount of noise β (i.e. white noise on the phone line). If the noise is left uncorrected, the noise may change the output in such a way that the *decoder* (the pizza parlor) converts the code into the wrong message. For the sake of simplicity, this introduction will ignore the impact of noise.

Surprise and Entropy

Now it's time to dig a little deeper, and we'll use what we learn to better understand why our encoding is a good one. Consider the probability mass function (pmf) for a biased die, shown in the figure below.



The *Shannon information* of an outcome for discrete variables is given by

$$h(x_i) = log_2\left(\frac{1}{p(x_i)}\right)$$

where $p(x_i)$ is probability of outcome x_i. The discrete random variable X may be used to represent these outcomes. In other words,

$$p(X) = \{p(x_1), p(x_2), p(x_3), ...\}$$

For the case of our biased die,

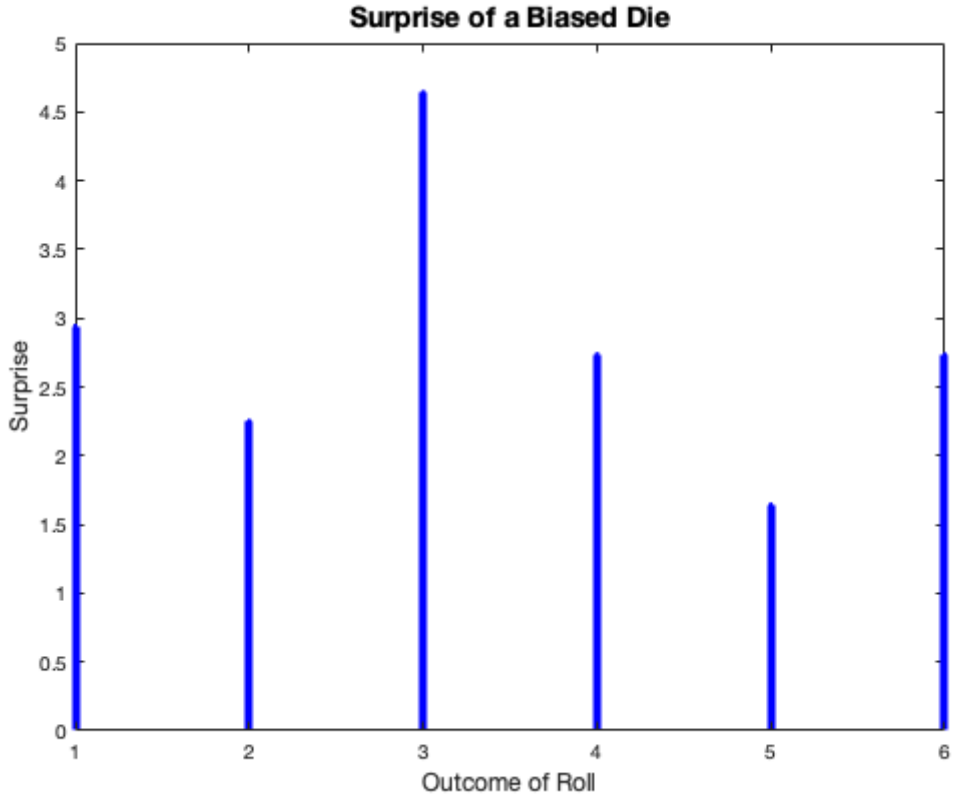
$$p(X) = \{p(1) = 0.13, p(2) = 0.21, p(3) = 0.04, p(4) = 0.15, p(5) = 0.32, p(6) = 0.15\}$$

According to the pmf above, a roll with an outcome of five is more likely than one with three. We can compare the Shannon information in both cases:

$$h(3) = log_2\left(\frac{1}{p(3)}\right) = 4.644 \text{ bits}$$

$$h(5) = log_2\left(\frac{1}{p(5)}\right) = 1.644 \text{ bits}$$

Shannon information is a measure of *surprise*. The results from the calculation above are intuitive: a roll of a three is much less likely than a five, so we are more surprised when we roll a three. The surprise associated with the outcomes of the biased die is shown in the next figure.



For a discrete random variable X with a pmf $p(X)$, the *entropy*, $H(X)$, represents the average surprise:

$$H(X) = \sum_{i=1}^{n} p(x_i) \frac{1}{p(x_i)} \text{ bits}$$

Our biased die has an entropy of

$$H(X) = \sum_{i=1}^{6} p(x_i) \log_2 \left(\frac{1}{p(x_i)} \right) = 2.8261 \text{ bits}$$

And a fair die (a die where the probability of rolling any of the six sides is the same) has an entropy of

$$H(X) = \sum_{i=1}^{6} p(x_i) \log_2 \left(\frac{1}{p(x_i)} \right) = \sum_{i=1}^{6} \frac{1}{6} \log_2 6 = 2.5850 \text{ bits}$$

What this means is that we will be more surprised by the outcome of our biased die than a fair die, on average. On one last note, these results can be used to find the number of equally probable outcomes. To demonstrate my point, consider the fair die.

$$H(X) = \sum_{i=1}^{6} \frac{1}{6} \log_2 (6) = \log_2 6$$

$$2^{H(X)} = 2^{\log_2 6}$$

$$2^{H(X)} = 6$$

In other words, a fair die has six equally probable outcomes. But for the biased die the results are different:

$$H(X) = 2.8261 \text{ bits}$$
$$2^{H(X)} = 7.0915 \text{ bits}$$

This means that rolling our biased die is equivalent to rolling a die with 7.0915 equally probable outcomes!

Source Coding Theorem

The pizza parlor has decided to help you in the creation of an efficient method for ordering pizza. You design a system with two buttons (one for 1 the other for 0) so that when you press a sequence of 1s and 0s, a message with your order is sent to the pizza parlor and decoded appropriately. We can use Shannon's *source coding theorem* to ensure that our encoding can communicate as much information over the channel per unit time as possible.

Consider the diagram above. For a discrete noiseless channel (noise β has no effect), the *channel capacity* is the maximum number of bits per second that the channel can communicate. Given a channel capacity $C\left[\frac{\text{bits}}{\text{s}}\right]$ and entropy $H\left[\frac{\text{bits}}{\text{symbol}}\right]$, Shannon's source coding theorem states that data may only be transmitted at a rate less than or equal to $C/H\left[\frac{\text{symbols}}{\text{s}}\right]$. The unit of symbol is referring to the symbols used in the original message (alphanumeric), not the symbols used for encoding.

Let's revisit the pizza problem. The table with the choices of ingredients is reproduced here.

Diameter	Crust	Sauce	Toppings
10"	regular	marinara	mushrooms
12"	thin	pesto	olives
16"	deep dish	alfredo	chicken
18"			sausage

For now, assume that each pizza is iid from a uniform distribution. There are $4 \cdot 3 \cdot 3 \cdot 4 = 144$ different possible pizzas from this table, so each pizza order has $log_2(144) = 7.1699$ bits of information. Under our assumption, the entropy for each choice of pizza can be found. Since each pizza is iid from a uniform distribution, the entropy will be the same as the number of bits of information needed to order each pizza.

$$H = \sum_{i=1}^{144} \frac{1}{144} \log_2 144 = 7.1699 \text{ bits/pizza}$$

Each option is represented as a list of alphanumeric symbols. Including a symbol for space, our options are constructed from a set of 30 symbols. On average, each pizza uses

27.5833 symbols (including space appended between words). Thus, the average entropy for all pizzas is

$$H = 7.1699 \text{ bits/choice} \cdot \frac{1}{27.5833} \text{ choice/symbol} = 0.2599 \text{ bit/symbol}$$

If we had a channel capacity of $C = 1$ bit/s, we could transmit a maximum of $C/H = \frac{1 \text{ bit/s}}{0.2599 \text{ bit/symbol}} = 3.8471$ symbols/s on average. Using our binary digit encoding, eight binary digits are used to represent an average of 27.5833 symbols per choice. From this information, the number of bits per binary digit can be determined:

$$\frac{0.2599 \text{ bits/symbol}}{8 \text{ binary digits}/27.5833 \text{ symbols}} = 0.8961 \text{ bit/binary digit}$$

Since a binary digit can only encode a maximum of one bit of information, a code which is maximally efficient would be able to transmit 1 bit/binary digit (a binary digit has two possibilities, so when the base-two log is taken of the number of possibilities, one bit is the result: $log_2(2) = 1$). So, although our code is not maximally efficient, 0.8961 bit/binary digits is still good!

Conclusion

The article has presented an introduction to the topics of information theory. Shannon's development of this field has laid the foundation of digital communication systems and compression methods that we rely on today. Information theory seems like a fascinating area of research and I'm interested in investigating how it may be applied to areas like machine learning or artificial intelligence, specifically natural language processing.

My knowledge on information theory came from the book *Information Theory: A Tutorial Introduction* by James Stone. This book has been my only reference in writing this article, and I found that the book does an excellent job of making intuitive sense of the math of information theory.