CS 811 Final Project

Ray Shulang Lei
200253624
Department of Computer Science
University of Regina
Regina, Saskatchewan, S4S0A2, Canada

April 25, 2011

Abstract

This project paper reviews Claude E. Shannon's information theory and the algorithmic information theory.

1 Introduction

2 Claude E. Shannon's information theory

2.1 The entropy of information

In information theory, the entropy of information means the measurement of the uncertainty of the information. The entropy H of an variable X is:

$$H(X) = E(I(X))$$

Where E is the expected value, and I is the information content of X. Let $x_i \in X$, and $p(x_i)$ be the probability mass function of X, then the entropy can be written as[1]:

$$H(X) = \sum_{i=1}^{n} p(x_i)I(x_i) = -\sum_{i=1}^{n} p(x_i)log_2 p(x_i)$$

Note that each $p(x_i)I(x_i)$ gives the expect value, where $I(x_i)$ is the average entropy of the specific event outcome. Here we assume the unit of entropy is bit, therefore the logarithm base is 2.

2.2 The fair coin tossing example

When tossing a coin, the outcomes are either head(x_h) or tail(x_t). Thus

$$X = \{x_h, x_t\}$$

Thus, before tossing the coin once, its entropy is:

$$H_{before}(X) = -(p(x_h)log_2p(x_h) + p(x_t)log_2p(x_t))$$

Assume this is a fair coin:

$$p(X = x_h) = 0.5, p(X = x_t) = 0.5$$

Therefore,

$$H_{before}(X) = -(0.5 \times log_20.5 + 0.5 \times log_20.5) = -(-0.5 - 0.5) = 1$$

Which means before toss the coin, the event would have 1 bit entropy.
After the coin has been tossed, let's say the result is tail, then its possible outcome now is only x_t:

$$X = \{x_t\}, p(X = x_t) = 1$$

And its entropy is now:

$$H_{after}(X) = -(1 \times log_21 + 1 \times log_21) = -(0 + 0) = 0$$

Thus, the entropy we have gained during the coin tossing is:

$$H(X) = H_{before} - H_{after} = 1 - 0 = 1$$

Which is 1 bit entropy. This means we need 1 bit entropy to store the each toss of a fair coin.
But what if the coin we toss is not fair?

2.3 The unfair coin tossing example

Assume we have an unfair coin that the chance it lands on its head is 0.8 and the chance it lands on its tail is 0.2:

$$p(X = x_h) = 0.8, p(X = x_t) = 0.2$$

Thus, before tossing the coin once, its entropy is:

$$H_{before}(X) = -(p(x_h)log_2p(x_h) + p(x_t)log_2p(x_t))$$

Which equals to:

$$H_{before}(X) = -(0.8 \times log_20.8 + 0.2 \times log_20.2)$$

$$\approx -(0.8 \times -0.321928094887362 + 0.2 \times -2.321928094887362)$$

$$= -(-0.25754247590989 - 0.464385618977472)$$

$$= 0.721928094887362$$

Again, the entropy after the coin has been tossed with a result of head is 0:

$$X = \{x_h\}, p(X = x_h) = 1$$

And its entropy is now:

$$H_{after}(X) = -(1 \times log_21 + 1 \times log_21) = -(0 + 0) = 0$$

Thus, the entropy we have gained during the unfair coin tossing is:

$$H(X) = H_{before} - H_{after} = 0.721928094887362 - 0 = 0.721928094887362$$

We can see that when the possibilities are not even, the entropy is 0.721928094887362 bit, which is less than 1 bit when the coin is fair.

2.4 The average information gain

The above two examples gives us how many information on average we would gain during a fair coin toss event and a unfair coin toss event.
Notice that these information gains are average information gains for each coin tosses considering both outcomes of heads and tails.

What is the average information gain for the coin toss event that the outcomes are heads only or tails only, then? Let's look at the following examples: We will look into I(X) where I is the information content of X.

$$I(X) = -log_2 p(x_i)$$

In the case of a fair coin toss with outcomes of heads only:

$$I(X) = -log_2 p(x_h), p(x_h) = 0.5$$

$$\implies I(X) = -log_2 \frac{1}{2} = 1$$

In the case of a fair coin toss with outcomes of tails only:

$$I(X) = -log_2 p(x_t), p(x_t) = 0.5$$

$$\implies I(X) = -log_2 \frac{1}{2} = 1$$

Thus 1 bit entropy is gained on average with each fair coin toss with the outcome of both head and tail.

In the case of the unfair coin toss with outcomes of heads only:

$$I(X) = -log_2 p(x_h), p(x_h) = 0.8$$

$$\implies I(X) = -log_2 \frac{4}{5} \approx 0.321928094887362$$

In the case of the unfair coin toss with outcomes of tails only:

$$I(X) = -log_2 p(x_t), p(x_t) = 0.2$$

$$\implies I(X) = -log_2 \frac{1}{5} \approx 2.321928094887362$$

When the outcome is head, we only gained 0.3 bit of information, where when the outcome is tail, we gained 2.3 bits of information. We discovered that more entropy is gained on average with each unfair coin toss event with the outcome of less common. We conclude that when the outcome of an event is less common, then it contains more information.

2.5 Exercise 1 - Rolling a fair die

What is the average entropy gain for the event of rolling a fair die? (in binary bits)

2.6 The Coding theory

After finished exercise 1, we can see that the average entropy gain for the event of rolling a fair die is about 2.58 bits. How many bits of storage we need to save this information? Let's try using two bits first:

$$00 - x_1, 01 - x_2, 10 - x_3, 11 - x_4, ?? - x_5, ?? - x_6$$

Apparently, with two bits of storage, we can only save 4 out of 6 outcomes. Let's increase to three bits:

$$000{-}x_1, 001{-}x_2, 010{-}x_3, 011{-}x_4, 100{-}x_5, 101{-}x_6, 110{-}unused, 111{-}unused$$

Now we can save all the outcomes. However, 110 and 111 are unused and wasted. This is becase with three bits storage, we have:

$$H(X) = -\sum_{1}^{8} \frac{1}{8} log_2 \frac{1}{8} = 3$$

Which is capable of saving 3 bits of entropy. For the event of rolling a fair die, only 2.58 bits of entropy are needed.
Let's look at an example of rolling a fair die 8 times with the outcomes of $x_2, x_4, x_5, x_3, x_6, x_4, x_3, x_1$ in sequence. Using the three bits code mapping above we can write this as:

$$001011100010101011010000$$

Now let's look at another coding for the 6 outcomes:

$$010 - x_1, 011 - x_2, 10 - x_3, 11 - x_4 000 - x_5, 001 - x_6,$$

And let's encode our example of rolling a fair die 8 times with the outcomes of $x_2, x_4, x_5, x_3, x_6, x_4, x_3, x_1$ in sequence:

$$01111000100011110010$$

We store 8 outcomes with 20 bits, compare to:

$$001011100010101011010000$$

where we only used 24 bits. This is possible because of the new code mapping:

$$010 - x_1, 011 - x_2, 10 - x_3, 11 - x_4 000 - x_5, 001 - x_6,$$

$$\begin{aligned} H(X) &= \frac{1}{6} \times 3 + \frac{1}{6} \times 3 + \frac{1}{6} \times 2 + \frac{1}{6} \times 2 + \frac{1}{6} \times 3 + \frac{1}{6} \times 3 \\ &= \frac{1}{2} + \frac{1}{2} + \frac{1}{3} + \frac{1}{3} + \frac{1}{2} + \frac{1}{2} \approx 2.67 \end{aligned}$$

Thus, we use about 2.67 bits for each outcome on average, instead of 3 bits. By measuring entropy, it's possible that we can compress data and exploit better coding. For example, in English, words being used more frequently codes with less letter than words being used rarely. Such as {'I', 'is', 'a'} compare to {'communism', 'encephalopathy'}

2.7 Exercise 2 - entropy of languages

Try to explain why languages like English is more effecient to be inputed by a keyboard than languages like Chinese using Shannon's information theory.

Assumptions:
A typical keyboard device has about 50 keys related to language inputs.
Each English letter is being used evenly, and here we only considering 26 English alphabets plus peroid and comma.
There are about 3000 commonly used Chinese characters and they are being used evenly.

3 Algorithmic information theory

We can see that Shannon's information theory look at information in a probablistic and statistic way. By doing exercise 2, we can also see that using entropy to encode a language is tedious and troublesome. Even calculating the entropy of a language is hard, considering the probabilitic usage of different letters or characters are not the same. In reality, English has an entropy of 4.03 bits per word and Chinese has 9.65 bits per character. Calculating these entropies requires tremendous effort.

Now let's look at a more descriptive and logical approach for the definition of information using Turing Machine's power.

3.1 Definition

4 Exercise Solutions

4.1 Exercise 1 Solution

First, we list the domain of the outcomes of rolling a fair die:

$$X = \{x_1, x_2, x_3, x_4, x_5, x_6\}$$

Second, we list the possibility distribution:

$$p(X = x_1) = \frac{1}{6}$$

$$p(X = x_2) = \frac{1}{6}$$

$$p(X = x_3) = \frac{1}{6}$$

$$p(X = x_4) = \frac{1}{6}$$

$$p(X = x_5) = \frac{1}{6}$$

$$p(X = x_6) = \frac{1}{6}$$

And then we calculate the entropy before we roll the die:

$$H_{before}(X) =$$

$$-p(x_1)log_2p(x_1) - p(x_2)log_2p(x_2)$$
$$-p(x_3)log_2p(x_3) - p(x_4)log_2p(x_4)$$
$$-p(x_5)log_2p(x_5) - p(x_6)log_2p(x_6)$$

$\Longrightarrow$

$$H_{before}(X) =$$

$$-\frac{1}{6}log_2\frac{1}{6} - \frac{1}{6}log_2\frac{1}{6}$$
$$-\frac{1}{6}log_2\frac{1}{6} - \frac{1}{6}log_2\frac{1}{6}$$
$$-\frac{1}{6}log_2\frac{1}{6} - \frac{1}{6}log_2\frac{1}{6}$$

$\Longrightarrow$

$$H_{before}(X) = -log_2\frac{1}{6}$$

$\Longrightarrow$

$$H_{before}(X) = 2.584962500721156$$

We also need to calculate the entropy after we roll the die. Let's say the outcome is x_i:

$$X = \{x_i\}$$

Then we figure the possibility distribution:

$$p(x_i) = 1$$

Then the entropy:

$$H_{after}(X) = -p(x_i)log_2p(x_i)$$
$$\Longrightarrow H_{after}(X) = -1log_21 = 0$$

Thus the entropy gain of rolling a die is:

$$H_{before}(X) - H_{after}(X) = 2.584962500721156 - 0 = 2.584962500721156$$

We say that the entropy gain for the event of rolling a fair die is about 2.58 bits.

4.2 Exercise 2 Solution

A typical keyboard device has about 50 keys related to language inputs. Thus it's average entropy for each key stroke is:

$$H(X) = -\sum_{i=1}^{50} p(x_i) log_2 p(x_i)$$

$$\implies H(X) = -\sum_{i=1}^{50} \frac{1}{50} log_2 \frac{1}{50}$$

$$= log_2 50 \approx 5.643856189774724$$

Which is about 5.6 bits entropy for each key stroke.
Now let's calculate the average entropy for each letter of English. Assuming their possibility distribution is even and considering 26 English alphabets plus peroid and comma:

$$H(X) = -\sum_{i=1}^{28} \frac{1}{28} log_2 \frac{1}{28}$$

$$= log_2 28 \approx 4.807354922057604$$

Thus we have about 4.8 bits entropy for each letter in English.
Storing each letter in English with 5.6 bits of entropy per key stroke is sufficient. Thus we can have one-to-one mapping between keys and letters and users do not need to remember any coding.

Let's look at entropy of each character of Chinese. Assuming there are about 3000 commonly used Chinese characters and their possibility distribution is even:

$$H(X) = -\sum_{i=1}^{3000} \frac{1}{3000} log_2 \frac{1}{3000}$$

$$= log_2 3000 \approx 11.55$$

Thus we have about 11.55 bits entropy for each character in Chinese.
Storing each letter in Chinese with 5.6 bits of entropy per key stroke is insufficient. Thus we need a coding schema containning more than three key

strokes for each Chinese character, and users have to remember the coding schema in order to use the keyboard as an input device. In fact, Chinese Pinyin is being use as a such schema, where Chinese Pinyin is very close to English compare to traditional Chinese.

References

[1] Fazlollah M. Reza, *An Introduction to Information Theory*. ISBN 0-486-68210-2, Dover Publications, Inc., New York, Dover Edition, 1994.