Entropy and Compression

Tim Thornton

1 Entropy

The concept of entropy is fundamental to information theory. In 1948, Claude Shannon introduced entropy as a measure of the amount of choice in a set of events, where only the probabilities of the events are known [3]. Entropy can be understood in many ways, such as a measure of uncertainty, or a measure of the amount of information gained when learning the outcome of a set of events. Each interpretation is insightful to specific applications, such as compression and coding.

Formally, for a discrete random variable $X$, entropy is the expected value of the information content of $X$. Thus, the entropy $H$ of $X$ is defined as $H(X) = \mathbb{E}[I(X)]$, if $I(X)$ is the information content. The information content of an event is simply the negative logarithm of the probability of the event, $I(p) = -\log_b p$. This definition of information content has the benefit of being additive [3]. If $p$ and $q$ are two independent events, then the information content of the sequence $(p, q)$ is $I(p) + I(q)$. Thus, entropy is defined to be:

$$H(X) = \mathbb{E}[I(X)] = \mathbb{E}[-\log_b P_r(X)] = -\sum_{x \in X} P_r(X = x)\log_b P_r(X = x)$$

The base $b$ of the logarithm is related to the unit of measurement. If bits are used as a unit of measurement, as is natural for information theory, then the base of the logarithm is 2. This definition of entropy provides a measure of the uncertainty of a random variable. The more uncertain the outcome of the random variable, the larger the entropy of the variable. Further, if $A$ and $B$ are two independent events, then entropy of the sequence $C = (A, B)$ is $H(C) = H(A) + H(B)$. This can be extended inductively to any finite sequence of independent events [2].

As an example, consider two biased coins - the first with probability of Heads = 3/4, and the second with probability of Heads = 7/8. Intuitively, the outcome of the first coin is less certain, so we should expect it to have larger entropy. Indeed, the entropy of the first is 0.8113, while the entropy of the second is 0.5436. Since these are binary random variables, their entropy can be found with the following expanded definition of entropy:

$$H(p) = -[p \log_2 p + (1 - p) \log_2 (1 - p)]$$

This follows immediately from the definition of entropy. Further, as expected, entropy is maximized when all possible outcomes of a random variable are equally likely [3]. This is expected since there is the greatest possible amount of uncertainty when all events are equally likely. In general, for a random variable $X$ with $n$ equally likely outcomes each with probability $1/n$, the entropy is:

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

On the other hand, if an event is certain, then the entropy is 0. In this case, no knowledge is gained when learning the outcome of the event.
2 Extraction

A first application of entropy is as the approximate number of unbiased independent bits that can be
extracted from a value of a random variable [2]. If \( \text{Ext}(\cdot) \) is an extraction function, then for any value
of a random variable \( X \), \( \text{Ext}(X) \) produces a sequence of bits \( y \), such that each sequence of length \( |y| \) is
equally likely to be produced (each has probability \( 1/2^{|y|} \)). As an example, the extraction function for a
uniform random variable with 4 values (0-3) could simply encode each value as a number written in binary,
as shown below.

<table>
<thead>
<tr>
<th>Input</th>
<th>0 1 2 3</th>
</tr>
</thead>
<tbody>
<tr>
<td>Output</td>
<td>00 01 10 11</td>
</tr>
</tbody>
</table>

Each sequence output by the extractor would have length two, and each sequence of length two is
equally likely to be generated. Further, each bit in any sequence produced is independent and unbiased.

In general, creating an extraction scheme for uniform random variables is trivial. If the random variable
has only one value, output nothing. If there are \( 2^n \) values, encode each as its binary number (this uses
\( \log_2 n \) many bits which is equal to the entropy of the random variable). If the number of values is not a
power of two, encode the first \( 2^n \) many values (for the largest \( n \) such that \( 2^n \) is less than the number of
values) with the previous scheme, and recurse on the remaining values. For example, for a random variable
with 6 equally likely values, the extraction would be defined as:

<table>
<thead>
<tr>
<th>Input</th>
<th>0 1 2 3 4 5</th>
</tr>
</thead>
<tbody>
<tr>
<td>Output</td>
<td>00 01 10 11 0 1</td>
</tr>
</tbody>
</table>

If a biased random variable is used instead, the process is similar. Consider \( n \) flips of a biased coin,
which lands on Heads with probability \( p \). If the \( n \) flips results in \( j \) heads, then there are \( \binom{n}{j} \) many
possibilities, each with probability \( p^j(1-p)^{n-j} \). Since these possibilities all have the same probability, the
extraction function can be constructed using the same scheme that was used for unbiased random variables
by mapping each sequence of \( j \) heads to a number in the range \( [0, \binom{n}{j} - 1] \).

Formally, as shown by Mitzenmacher and Upfal [2], for sufficiently large \( n \) and any \( \delta > 0 \), the expected
number of bits produced is \( (1 - \delta) n H(p) \).

3 Compression

The concept of entropy naturally arises when discussing compression. In this domain, entropy can be
viewed as the average number of bits needed to encode each symbol of a string of symbols \( s_1, ..., s_n \), each
appearing with frequency \( p_1, ..., p_n \). This follows from the Source Coding Theorem [3], which states:

1. The average bits per symbol needed to encode each value of a random variable is greater than or
   equal to the entropy of the random variable

2. If a sufficiently large sequence of values of a random variable are encoded, there is a prefix code for
   the random variable that uses an average number of bits per symbol arbitrarily close to the entropy
   of the random variable

Huffman codes are an encoding strategy that provides encodings with an average number of bits per
symbol between \( H \) and \( H + 1 \). Constructing Huffman codes requires a simple recursive scheme that creates
a tree starting from the leaves:

1. Use each symbol as a leaf of the tree
2. Repeatedly select the two least frequent nodes and construct a new node with the two selected nodes as children. Systematically label each branch with a ’0’ or ’1’.

3. Construct the codewords for each symbol by reading the branches from the root to each leaf.

Huffman codes are an optimal variable-length prefix code for lossless compression. However, the second assertion of the Source Coding Theorem says that for large strings, the upper bound of \((H + 1)\) of Huffman Codes is not optimal.

Block coding is a strategy that encodes groups of a chosen size \(m\) of symbols rather than each individual symbol. For example, a block code with \(m = 2\) would encode pairs of symbols. Taking \(m\) to be arbitrarily large and encoding with the same Huffman encoding scheme as before, the upper bound on the average bits/symbol can be brought arbitrarily close to the entropy \(H\) \[3\].

4 Extensions

There are many useful extensions on the concept of entropy, with many applications themselves. For example, a natural question that may arise is how many bits are required on average to encode a sample of an alphabet with distribution \(P\) using an encoding optimized for the same alphabet over distribution \(Q\). For instance, if the distribution of an alphabet is found by analyzing Book A, how many bits are required to compress Book B (which may have a different distribution of characters). Cross entropy provides a measure for this, with the following definition (for some alphabet \(A\)) \[1\]:

\[
H(P, Q) = -\sum_{i \in A} P(i) \log_2 Q(i)
\]

Subtracting the entropy of \(P\) from the cross entropy of \(P\) and \(Q\) gives the number of extra bits required on average to encode a sample of \(P\) using an encoding of \(Q\). This measurement is called Kullback-Leibler divergence \[1\], and is defined as:

\[
D_{KL}(P \mid Q) = \sum_{i \in A} P(i) \log_2 \frac{P(i)}{Q(i)}
\]

This can be used as a rough (non-symmetric) measure of the similarity of two probability distributions.

References

