- How to measure Information
- Self Information
- Instantaneouse Cod
- Joint Entropy
- Chain Rule for Entropy
- Prefix Code
- Shannon Fado Coding
- Mutual Information

How to measure information?

A good measure on information content should base on the degree of its uncertainty. From probabilistic viewpoint, if an event is less likely to happen, it should carry more information when it occurs, because it is more uncertain that the event would happen.

In addition, it is reasonable to have additivity for information measure, i.e., the degree-of-uncertainty of a joint event should equal the sum of the degree-of-uncertainty of each individual event. Moreover, a small change in event probability should only yield a small variation in event uncertainty.

How to measure information?

For example, two events respectively with probabilities 0:20001 and 0:19999 should reasonably possess comparable information content.
As it turns out, the only measure satisfying these demands is the self-information dened as the logarithm of the reciprocal of the event probability .
It is then legitimateto adopt the entropy—the expected value of the self-information—as a measure of information.

How to measure information?

In the case of data transmission over noisy channel, the concern is dierent from that for data storage (or error-free transmission).
The sender wishes to transmit to the receiver a sequence of pre-defined information symbols under an acceptable symbol error rate. Code redundancies are therefore added to combat the noise.
For example, one may employ the three-times repetition code (i.e., transmit 111 for information symbol 1 and send 000 for information symbol 0), and apply the majority law at the receiver end so that one-bit error can be recovered.

Definition

A code is called a prefix code or an instantaneous code if no codeword is a prefix of any other codeword. An instantaneous code can be decoded without reference to future codewords since the end of a codeword is immediately recognizable.
Hence, for an instantaneous code, the symbol x_i can be decoded as soon as we come to the end of the codeword corresponding to it. We need not wait to see the codewords that come later.

Information Theory

The initial questions treated by information theory lay in the areas of data compression and transmission. The answers are quantities such as entropy and mutual information, which are functions of the probability distributions that underlie the process of communication. A few definitions will aid the initial discussion.
. The entropy of a random variable X with a probability mass function p(x) is defined by $H(X) = -\sum xp(x)log2p(x)$.

We use logarithms to base 2. The entropy will then be measured in bits. The entropy is a measure of the average uncertainty in the random variable. It is the number of bits on average required to describe the random variable.

- How to measure Information
- Self Information
- Instantaneouse Cod
- Joint Entropy
- Chain Rule for Entropy
- Prefix Code
- Shannon Fado Coding
- Mutual Information

Definition

A code is called a prefix code or an instantaneous code if no codeword is a prefix of any other codeword. An instantaneous code can be decoded without reference to future codewords since the end of a codeword is immediately recognizable.

Hence, for an instantaneous code, the symbol x_i can be decoded as soon as we come to the end of the codeword corresponding to it. We need not wait to see the codewords that come later.

Self-information

Let E be an event with probability Pr(E), and let I(E) represent the amount of information you gain when you learn that E has occurred (or equivalently, the amount of uncertainty you lose after learning that E has happened). Then a natural question to ask is “what properties should I(E) have?”
The answer to the question may vary person by person. Here are some common properties that I(E), which is called the self-information, is reasonably expected to have.

Instantaneous Code

An instantaneous code is a selfpunctuating code; we can look down the sequence of code symbols and add the commas to separate the codewords without looking at later symbols. For example, the binary string 01011111010 produced by the code is parsed as 0,10,111,110,10.

In computing the entropy, we adopt the convention that

$$0 log 0 = 0$$

which can be justified by continuity since $x log x - 0$ as $x - 0$.

Joint Entropy

By its definition, joint entropy is commutative; i.e., $H(X,Y) = H(Y,X)$.
The conditional entropy can be thought of in terms of a channel whose input is the random variable X and whose output is the random variable Y .
$H(X|Y)$ is then called the equivocation and corresponds to the uncertainty in the channel input from the receiver's point-of-view.

Joint Entropy

- The Joint Entrropy $H(X,Y)$ of a pair of discrete random variables (X,Y) with a joint probability $p(x,y)$ is defined as

$$H(X,Y) = -\sum_{x}\sum_{y} p(x,y)\,\log p(x,y)$$

- This can also be expressed as

$$H(X,Y) = -E[\,\log p(x,y)]$$

Chain rule for entropy

$$H(X,Y) = H(X) + H(Y|X)$$

mutual information

For two random variables X and Y , the mutual information between X and Y is the reduction in the uncertainty of Y due to the knowledge of X (or vice versa)

uniquely decodable

In practice, the encoder often needs to encode a sequence of source symbols, which results in a concatenated sequence of codewords. If any concatenation of codewords can also be unambiguously reconstructed without punctuation, then the code is said to be uniquely decodable. Note that a non-singular code is not necessarily uniquely decodable. For example, consider a code for source symbols $\{A,B,C,D,E,F\}$ given by
code of A = 0;
code of B = 1;
code of C = 00;
code of D = 01;
code of E = 10;
code of F = 11:
The above code is clearly non-singular; it is however not uniquely decodable because the codeword sequence, 01, can be reconstructed as AB or D.

Classification of variable-length codes

- Prefix Codes
- Uniquely decodable codes
- Non-singular codes

A ***prefix code*** is also named an instantaneous code because the codeword sequence can be decoded instantaneously without the reference to future codewords in the same sequence. Note that a uniquely decodable code is not necessarily prefix-free and may not be decoded instantaneously.
For any prefix code, the average codeword length is no less than entropy.

Consider a random variable X taking values in the set X = 1, 2, 3, 4, 5 with probabilities 0.25, 0.25, 0.2, 0.15, 0.15, respectively. We expect the optimal binary code for X to have the longest codewords assigned to the symbols 4 and 5. These two lengths must be equal, since otherwise we can delete a bit from the longer codeword and still have a prefix code, but with a shorter expected length.

In general, we can construct a code in which the two longest codewords differ only in the last bit. For this code, we can combine the symbols 4 and 5 into a single source symbol, with a probability assignment 0.30.
Proceeding this way, combining the two least likely symbols into one symbol until we are finally left with only one symbol, and then assigning codewords to the symbols, we obtain the following table:

Huffmans idea

Instead of using a fixed-length code for each symbol, Huffmans idea is to represent a frequently occurring character in a source with a shorter code and to represent a less frequently occurring one with a longer code.
So for a text source of symbols with different frequencies, the total number of bits in this way of representation is, hopefully, significantly reduced. That is to say, the number of bits required for each symbol on average is reduced.

Frequency of occurrence:
E A O T J Q X
5 5 5 3 3 2 1
Suppose we find a code that follows Huffmans approach. For example, the most frequently occurring symbol E and A are assigned the shortest 2-bit codeword, and the lest frequently occurring symbol X is given a longer 4-bit codeword, and so on, as below:
E A O T J Q X
10 11 000 010 011 0010 0011
Then the total number of bits required to encode string ‘EEETTJX is only $2+2+2+3+3+3+4 = 19$ (bits). This is significantly fewer than $8 \times 7 = 56$ bits when using the normal 8-bit ASCII code.

1. Constructing a frequency table sorted in descending order.
2. Building a binary tree
Carrying out iterations until completion of a complete binary tree:
(a) Merge the last two items (which have the minimum frequencies) of the frequency table to form a new combined item with a sum frequency of the two.
(b) Insert the combined item and update the frequency table.

3. Deriving Huffman tree Starting at the root, trace down to every leaf; mark 0 for a left branch and 1 for a right branch.
4. Generating Huffman code:
Collecting the 0s and 1s for each path from the root to a leaf and assigning a 0-1 codeword for each symbol.

Huffmans idea

Instead of using a fixed-length code for each symbol, Huffmans idea is to represent a frequently occurring character in a source with a shorter code and to represent a less frequently occurring one with a longer code.
So for a text source of symbols with different frequencies, the total number of bits in this way of representation is, hopefully, significantly reduced. That is to say, the number of bits required for each symbol on average is reduced.

Frequency of occurrence:

E	A	O	T	J	Q	X
5	5	5	3	3	2	1

Suppose we find a code that follows Huffmans approach. For example, the most frequently occurring symbol E and A are assigned the shortest 2-bit codeword, and the lest frequently occurring symbol X is given a longer 4-bit codeword, and so on, as below:

E	A	O	T	J	Q	X
10	11	000	010	011	0010	0011

Then the total number of bits required to encode string ‘EEETTJX is only $2+2+2+3+3+3+4=19$ (bits). This is significantly fewer than $8\times 7=56$ bits when using the normal 8-bit ASCII code.

1. Constructing a frequency table sorted in descending order.
2. Building a binary tree
Carrying out iterations until completion of a complete binary tree:
(a) Merge the last two items (which have the minimum frequencies) of the frequency table to form a new combined item with a sum frequency of the two.
(b) Insert the combined item and update the frequency table.

3. Deriving Huffman tree Starting at the root, trace down to every leaf; mark 0 for a left branch and 1 for a right branch.
4. Generating Huffman code:
Collecting the 0s and 1s for each path from the root to a leaf and assigning a 0-1 codeword for each symbol.

Decoding algorithm

The decoding process is based on the same Huffman tree. This involves the following types of operations:

- We read the coded message bit by bit. Starting from the root, we follow the bit value to traverse one edge down the the tree.
- If the current bit is 0 we move to the left child, otherwise, to the right child.
- We repeat this process until we reach a leaf. If we reach a leaf, we will decode one character and re-start the traversal from the root.
- Repeat this read-move procedure until the end of the message.

Example : Given a Huffman-coded message,
111000100101111000001001000111011100000110110101,
what is the decoded message?