Information Theory Introduction

EECS 126 (UC Berkeley)

Fall 2018

1 Information Measures

This note is about some basic concepts in information theory. We start by introducing some fundamental information measures. They are so called because, as the name suggests, they help us measure the amount of information. As we go forward, we will see why this is in more detail. Note, many proofs are omitted in this note. You will do many of these on homework and the rest are beyond the focus of this class. If you are interested in learning more, take EE229A.

1.1 Entropy

The entropy of a discrete random variable X which takes values in the alphabet $\mathcal{X}$ and is distributed according to $p_X : \mathcal{X} \to [0, 1]$ is defined by

$$H(X) := \sum_{x\in\mathcal{X}} p_X(x) \log \frac{1}{p_X(x)} = \mathbb{E}\left[\log \frac{1}{p_X(X)}\right] \tag{1}$$

where in the second equality we use the fact that $\mathbb{E}[f(x)] = \sum p_X(x)f(x)$ with $f(x) = \log \frac{1}{p_X(x)}$ [1]. The function $f(x) = \log \frac{1}{p_X(x)}$ is often explained in English as the 'surprise', 'information content' or 'uncertainty' associated with observing the value x, so the entropy can be understood as the **expected surprise** or **expected information content** or **expected uncertainty** in the random variable. Notice that $f(x)$ is larger when $p_X(x)$ is smaller. This is in agreement with the fact that we would be more surprised by/gain more information from/feel more uncertain about observing a value with low probability.

The terms surprise, uncertainty and information are all used often to describe entropy in words. You may pick whichever one helps you build intuition most easily as we go along. In this note, we choose to use the term 'information' for qualitative explanations.

Generally, the base of the logarithm is 2 and the entropy has units 'bits'. The expected number of bits needed to express the information in X is a natural choice to measure its uncertainty. It turns out that the expected number of bits required to describe the random variable is roughly the entropy $H(X)$.

Question: Why not use the variance to measure the uncertainty of a random variable?

Answer: We want a quantity that depends only on the distribution of X and not on the specific values it can take on. For example consider the random variables $Z \sim \mathit{Bernoulli}(p)$ and $Y = 1000Z$. $\text{var}(Z) = p(1 - p)$ while $\text{var}(Y) = 1000^2 p(1 - p)$. We see that the variance of Y is much larger, though both of them are Bernoulli with the same bias. We want a quantity that gives us the same uncertainty for both Y and Z as they both have the same bias. Our goal is to measure the uncertainty of a random variables or the information gained from observing it, without being concerned about what exactly the values are. The entropy satisfies this. It depends only on the distribution of X. It is also for this reason that it makes sense to speak about entropy for 'random variables' that may take values that are not numerical such as symbols in an alphabet.

1.1.1 Some Properties of Entropy

- Entropy is **non-negative**. This can be easily seen since $p_X(x) \in [0, 1]$, so $\frac{1}{p_X(x)} \geq 1 \implies \log \frac{1}{p_X(x)} \geq 0$.
- Since $f(x) = x \log x$ is a convex function in x, we know entropy is concave in $p_X(x)$, which is a very useful property when optimizing on entropy or justifying inequalities.

[1] We adopt the convention that $0 \log \frac{1}{0} = 0$ for values with $p_X(x) = 0$

- One such important result that can be shown using concavity is that $H(X) \leq \log_2|\mathcal{X}|$ (proof out of scope but its fun to try if you would like). This shows that for a fixed $\mathcal{X}$, entropy is maximized by the uniform distribution over $\mathcal{X}$. (It is a good exercise to verify that the entropy of a uniform distribution is $\log_2|\mathcal{X}|$).

1.2 Joint Entropy

The joint entropy of two random variables X and Y with alphabets $\mathcal{X}$ and $\mathcal{Y}$ and joint distribution $p_{X,Y}$ is

$$H(X,Y) := \sum_{x,y\in\mathcal{X}\times\mathcal{Y}} p_{X,Y}(x,y)\log\frac{1}{p_{X,Y}(x,y)} \tag{2}$$

Note that this is just a natural extension of (1) to (X,Y). This measures the information conveyed by X and Y. This can also be written as $H(X,Y) = \mathbb{E}\left[\log\frac{1}{p_{X,Y}(X,Y)}\right]$

1.3 Conditional Entropy

The conditional entropy of Y given X $(H(Y|X))$ is defined as

$$H(Y|X) := \sum_{x\in\mathcal{X}} p_X(x)\sum_{y\in\mathcal{Y}} p_{Y|X}(y|x)\log\frac{1}{p_{Y|X}(y|x)} = \sum_{x\in\mathcal{X}} p_X(x)H(Y|X=x) \tag{3}$$

The interpretation of this quantity is the information gained from Y given we have X, or in other words, the information in Y not already provided by X. In expectation form it can also be written as $\mathbb{E}\left[\log\frac{1}{p_{Y|X}(Y|X)}\right]$.

1.4 Some properties of Joint and Conditional Entropy

1. Conditional entropy can be related to the joint entropy and the entropy of one the individual random variables. Moving $p_X(x)$ into the inner sum, we get

$$\begin{aligned}
H(Y \mid X) &= \sum_x p_X(x)\sum_y p_{Y|X}(y|x)\log\frac{1}{p_{Y|X}(y|x)}\\
&= \sum_{x,y} p_X(x)p_{Y|X}(y|x)\log\frac{p_X(x)}{p_X(x)p_{Y|X}(y|x)}\\
&= \sum_{x,y} p_{X,Y}(x,y)\log\frac{p_X(x)}{p_{X,Y}(x,y)}\\
&= \sum_{x,y} p_{X,Y}(x,y)\log\frac{1}{p_{X,Y}(x,y)} - \sum_{x,y} p_{X,Y}(x,y)\log\frac{1}{p_X(x)}\\
&= H(X,Y) - \sum_x p_X(X)\log\frac{1}{p_X(x)}\cancelto{1}{\sum_y p_{Y|X}(y|x)}\\
&= H(X,Y) - H(X)
\end{aligned}$$

Rearranging we get the following theorem:

Theorem(*Chain rule of entropy*):

$$H(X,Y) = H(X) + H(Y|X) \tag{4}$$

This agrees with the natural argument that the information obtained from both X and Y is the information from X + the information from Y not already provided by X. Since $H(X,Y) = H(Y,X)$, the same set of steps also leads to $H(X,Y) = H(Y) + H(X|Y)$.

2. In general,
$$H(Y|X) \leq H(Y) \tag{5}$$
Given X, the information from Y can only be reduced. This makes sense as X may provide some information about Y when they are dependent. It intuitively follows that the information from Y given X is equal to the information from Y i.e $H(Y|X) = H(Y)$ iff X and Y are independent. It is left as an exercise to verify this from the definition of conditional entropy.

3. Combining (4) and (5), we also get
$$H(X,Y) \leq H(X) + H(Y) \tag{6}$$
with equality iff X, Y are independent.

1.5 Mutual Information

As mentioned earlier, if X and Y are not independent, they provide some information about each other. The mutual information $I(X;Y)$ measures the information about one gained from the other. It is defined as -

$$\begin{aligned} I(X;Y) &:= H(X) - H(X|Y) && (7) \\ &= H(X) + H(Y) - H(X,Y) && (8) \\ &= H(Y) - H(Y|X) && (9) \end{aligned}$$

The equation above in English says it is the information from Y, after taking out the information that is not given by X. Note that it is symmetric ($I(X;Y) = I(Y;X)$). Also, from inequality (5), we see that mutual information is non-negative and is 0 if and only if X and Y are independent. Figure 1 gives a nice pictorial representation of the relationship between $H(X,Y)$, $H(X)$, $H(Y)$, $H(X|Y)$, $H(Y|X)$ and $I(X;Y)$ (Note this pictorial representation only really makes sense between two variables).

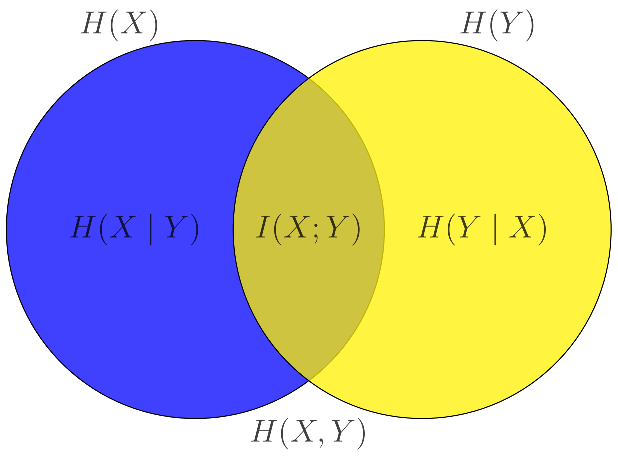


Figure 1: Venn diagram representing relationships between the various quantities (Go bears!)

The figure above brings out a nice analogy with sets. The joint entropy is the analog of the union while the mutual information is analogous to the intersection. $H(Y|X)$ is analogous to $X \cup Y \setminus X$ (note the similarity in the formula $H(X,Y) - H(X) = H(Y|X)$). Lastly, the chain rule of entropy is the analog of the inclusion exclusion principle in this analogy.

2 Source Coding, AEP, Typical Sets and Compression

We mentioned at the beginning that the average number of bits required to convey the information in a random variable is the entropy. Lets see what this means a little more formally.

2.1 Source Coding

A **source coding** scheme is a mapping from the symbols x in the source alphabet $\mathcal{X}$ to bit strings. The goal of source coding is to send the most compressed sequence of bits for which it is possible to recover the symbols. Clearly this will have something to do with the entropy which we've been referring to as the average number of bits of information in a random variable.

Let the function $l(x)$ be the length of the binary string description of x. $l(X)$ is then a random variable representing the length of the description for X. Extending this to a sequence of symbols (think of a random text file for example), the binary description of this sequence has length $l(X_1, X_2, ...X_n)$. The average bits per symbol for this sequence of symbols is then $\frac{l(X_1,X_2,...X_n)}{n}$. The following theorem regarding how small this average could be (best achievable compression) was stated by Shannon in 1948:

Source Coding Theorem:

- For an i.i.d sequence $X_1, ..., X_n$ and an arbitrarily small $\epsilon > 0$, there is a source coding scheme for which
$$\lim_{n\to\infty} \mathbb{E}[\frac{1}{n} l(X_1, X_2, ...X_n)] \le H(X) + \epsilon \tag{10}$$
(bits per symbol) such that the sequence $X_1, \ldots, X_n$ can be recovered from the encoding with a high probability $(1 - \epsilon)$.
- Conversely, there is no source coding that can achieve an average less than $H(X)$ bits per symbol in expectation.

In words, the theorem above tells us that any compression above an average of $H(X)$ bits per symbol is achievable and you can get arbitrarily close to $H(X)$. The entropy $H(X)$ is a fundamental limit (asymptotically) on the number of bits per symbol to which a sequence can be compressed under any source coding scheme. To understand why this is we introduce the AEP and the notion of a typical set.

2.2 Typical Sets and the Asymptotic Equipartition Property

Our goal for compression as we stated before is to minimize the expected average number of bits per symbol used. If we wanted to assign a bit string to every n length string in $\mathcal{X}^n$, we would end up needing n length bit strings. How can we do better?

Well, we want to be able to assign shorter bit strings to the most probable (more 'typical') sequences to reduce the expected length of the code. It turns out that as n grows large, all the probability concentrates on a much (exponentially) smaller subset of sequences in $\mathcal{X}^n$. This set is called the typical set and a formal definition of it will follow shortly. First lets look at an example to try and develop a notion of a typical set.

Example: Biased Coin Flips
Consider a situation where you are flipping n coins with a probability of p of coming up heads. What is the probability of seeing a 'typical' sequence?

Answer: First we would need to ask ourselves what it means to be a typical sequence. A reasonable starting point might be to say it is one in which there are np heads and $n(1-p)$ tails, the expected number of heads and tails. What is the probability of seeing such a typical sequence? It is

$$\begin{aligned}
&p^{np} \cdot (1-p)^{n(1-p)} \\
&= 2^{\log(p^{np} \cdot (1-p)^{n(1-p)})} \\
&= 2^{n\cdot(p \log p + (1-p)\log(1-p))} \\
&= 2^{-nH(p)}
\end{aligned}$$

This example gives us an idea for a possible definition of a typical set for a sequence of random variables $(X_1, \cdots, X_n)$. It would be a set of sequences which have a probability $\approx 2^{-nH(X)} = 2^{\mathbb{E}[\log p_X(X)^n]}$. Formally, this condition is stated by defining the ϵ-typical set $A_\epsilon^{(n)}$, which is the set of sequences in $\mathcal{X}^n$ which fit the condition:

$$2^{-n(H(X)+\epsilon)} \leq p_{X^n}(x_1, x_2, \ldots, x_n) \leq 2^{-n(H(X)-\epsilon)} \tag{11}$$

It can be shown that the size of this set is $\approx 2^{nH(X)}$. Compare this to the size of the set of all possible sequences $= |\mathcal{X}|^n = 2^{n \log |\mathcal{X}|}$. $\log |\mathcal{X}|$ is the entropy of a uniform distribution over $\mathcal{X}$, so we know $H(X) \leq \log |\mathcal{X}|$. This means the size of the typical set is **exponentially smaller** than the size of all possible sequences $(x_1, \cdots, x_n)$. Let us see this explicitly with another example with coins.

Example: Biased Coin Flips Again
You flip 100 coins with probability 1/4 of coming up heads. What is the size of the sample space? What is the size of a typical set? How do they compare?

Answer: In this case the number of possible sequences is 2^{100}. $H(X) = \frac{1}{4}\log 4 + \frac{3}{4}\log 4/3 = 0.5 + 0.311 = 0.811$. So the size of the typical set is $\approx 2^{81}$. $\frac{2^{81}}{2^{100}} = \frac{1}{2^{19}}$, which means the size of typical set is 1/524288 of the the size of the sample space - much smaller!

We know that the sequences in the typical set contain more of the probability mass, but how much? It turns out as n grows large the typical set contains almost all of it. This is formally expressed through the Asymptotic Equipartition Property (AEP) which is an analogue of the law of large numbers in information theory.[2]

Theorem(*Asymptotic Equipartition Property*):
If $X_1, X_2, \ldots, X_n \sim p_{X^n}$ i.i.d [3], then

$$-\frac{1}{n}\log_2 p_{X^n}(X_1, X_2 \cdots X_n) \to H(X) \text{ i.p.} \tag{12}$$

where p_{X^n} is shorthand for the joint pmf $p_{X_1,\cdots,X_n}$. This means for any $\epsilon > 0$,

$$\mathbb{P}\left(|-\frac{1}{n}\log_2 p_{X^n}(X^n) - H(X)| > \epsilon\right) \to 0 \text{ as } n \to \infty \tag{13}$$

Doing some math to write out the absolute value and get rid of the log, we get

$$\mathbb{P}\left(2^{-n(H(X)+\epsilon)} < p_{X^n}(X^n) < 2^{-n(H(X)-\epsilon)}\right) \to 1 \tag{14}$$

But the event above is exactly the ϵ-typical set! What this tells us is that sequences in the (much smaller) typical set contain virtually all the probability mass. Thus there is a set of sequences which is exponentially smaller in size that has probability 1 (asymptotically).

This concept is at the heart of compression. The fact that the typical set has probability approaching 1 as n grows large means that we only really need to care about encoding the sequences in the typical set. The number of bits required to encode a set of size S is $\lceil \log_2 S \rceil$, so the number required to assign a code to each sequence in the typical set is $\leq n(H(X)+\epsilon)$. The expected average length of the bitstring for the sequence is thus $\leq H(X)+\epsilon$ and ϵ can be made arbitrarily small which explains the achievability part of the Source Coding Theorem - a sequence of symbols can be compressed to a binary string with an average of $H(X)$ bits per symbol. This further reinforces the interpretation of the entropy as the average information content of a random variable.

[2]This comes directly from applying the law of large numbers to the random variable $\log \frac{1}{p_X(X)}$. Though it is true in the almost sure sense, it is more useful to work with it in the weak sense.

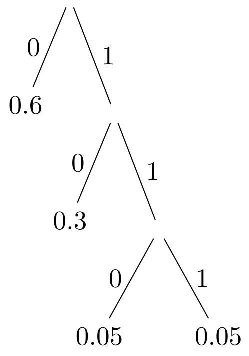
[3]$X_1 ... X_n$ do not need to be i.i.d. For a more general version, see the Shannon–McMillan–Breiman theorem

3 Huffman Coding

In this section, we introduce Huffman Coding, which is the optimal scheme to code a random variable X in a way that is prefix free.

3.1 Prefix Free Codes

A source coding scheme is a **prefix code** (i.e. prefix free) if no codeword is a prefix of another codeword. Equivalently, a code is a prefix code if it can be represented as a binary tree where the leaves correspond to codewords. The codewords themselves are then determined by the path from the root to the leaf.



Here is an example of a code for a random variable that takes on $m = 4$ values with probabilities $\mathbf{p} = (0.6, 0.3, 0.05, 0.05)$ and codewords $\mathbf{w} = (0, 10, 110, 111)$ with lengths $\mathbf{l} = (1, 2, 3, 3)$. It's expected length is $0.6 \cdot 1 + 0.3 \cdot 2 + 0.05 \cdot 3 + 0.05 \cdot 3 = 1.5$ bits. The property of being prefix free is important because it allows us to uniquely decode a sequence of symbols. For example, if a code was $\mathbf{w} = (0, 01, 010, 10)$ and I received 010, it could have either of been (010), $(0, 10)$ or $(01, 0)$. The issue is that 0 is a prefix of 01 which is a prefix of 010.

3.2 The Algorithm

```
1: procedure HUFFMAN($p_1, \cdots, p_n$)
2:     Input: The probabilities $p_1, ..., p_n$ of each symbol
3:     Output: The encoding binary tree
4:     Let Q be a priority queue of nodes prioritized by $p_i$
5:     for $i = 1$ to $n$ do
6:         add leaf node i to Q
7:     while size of Q is not 1 do
8:         i = pop minimum probability node from Q
9:         j = pop minimum probability node from Q
10:        create a new node x
11:        set x.right $\leftarrow$ i, x.left $\leftarrow$ j and $p_x \leftarrow p_i + p_j$
12:        add x to Q with priority $p_x$
```

The Huffman algorithm repeatedly combines the lowest probability nodes and builds a binary tree in a bottom-up fashion. This means that it is putting the lowest probability nodes at the bottom of tree and the higher probability nodes are closer to the root. If we encode the symbols by denoting going left in the tree by 0 and going right by 1, we get the encoding for each symbol by tracing the sequence of 0s and 1s from the root to the corresponding leaf node for that symbol.The Huffman algorithm thus greedily assigns longer strings to the lowest probability symbols while assigning shorter strings to the higher probability symbols. This is ideal as we are looking to reduce the expected codeword length $\sum p_i l_i$.

Example
Consider the symbols a, b, c, d occurring with the same probabilities as given in the example in section 3.1 $(0.6, 0.3, 0.05, 0.05)$. We know the tree should look like the one shown. Following the algorithm, we first combine node c and node d to make an internal node (let's label it i_1) with probability $0.05 + 0.05 = 0.1$, whose left and right nodes are c and d. Now the two lowest probability nodes are b and i_1. We now combine these to form another internal node (i_2) with probability $0.3 + 0.1 = 0.4$. Now we have only two nodes left (a and i_2) which we combine at the root to give us the tree we expected. The code words for a, b, c, d are thus $(0, 10, 110, 111)$. The expected length of this code as calculated in 3.1 is 1.5. Lets compare this to the entropy $-0.6 \log 0.6 - 0.3 \log 0.3 - 0.1 \log 0.05 = 1.395$. We notice that the Huffman code length is slightly more. It can be shown that it will be between $H(X)$ and $H(X) + 1$ in general.

3.3 A Word on the Optimality of Huffman Codes

Huffman Codes can be shown to be optimal among all uniquely decodable codes that assign code words one symbol at a time (see Cover and Thomas for proof). However, for such codes we can only show that the expected code length is between $H(X)$ and $H(X) + 1$ for each symbol, which means that the expected average length over n symbols is between $\frac{nH(X)}{n}$ and $\frac{n(H(X)+1)}{n}$, so this is also between $H(X)$ and $H(X)+1$. You may notice then that this does not manage to get arbitrarily close to $H(X)$ which the Source Coding Theorem tells us is possible. There are more optimal codes that encode entire sequences and not just one symbol at a time. Arithmetic codes are an example of this which manage to achieve the Shannon limit. This however is beyond the scope of our class.

References

[1] Thomas M. Cover and Joy A. Thomas. *Elements of Information Theory.* Wiley Interscience. Second Edition.