Lecture 3: Lossless compression

ENGR 76 lecture notes — April 9, 2024

Ayfer Ozgur, Stanford University

1 Joint entropy

In this lecture, we will prove Shannon's source coding theorem which we stated in the previous lecture. To do that, we first introduce the notion of joint entropy. Recall that the entropy $H(X)$ of a discrete random variable X with alphabet $\mathcal{X}$ is defined as

$$H(X) \triangleq \sum_{x\in\mathcal{X}} P(X=x)\log\frac{1}{P(X=x)}.$$

For any pair of r.v.s X and Y (with alphabets $\mathcal{X}$ and $\mathcal{Y}$, respectively), we define their joint entropy as

$$H(X,Y) \triangleq \sum_{x\in\mathcal{X}}\sum_{y\in\mathcal{Y}} P(X=x,Y=y)\log\frac{1}{P(X=x,Y=y)}$$

Note that this definition can be interpreted as treating (X,Y) as a supersource. We go through all the values the pair of random variables (X,Y) can take together in $\mathcal{X}\times\mathcal{Y}$ and consider the corresponding probability $P(X=x,Y=y)$ for each outcome $(x,y)\in\mathcal{X}\times\mathcal{Y}$.

Example: Consider
$\mathcal{X}=\{1,\dots,M\}$, $P(X=i)=\frac{1}{M}$, $1\le i\le M$
$\mathcal{Y}=\{1,\dots,N\}$, $P(Y=j)=\frac{1}{N}$, $1\le j\le N$
X and Y are independent, i.e., $P(X=i,Y=j)=P(X=i)P(Y=j)=\frac{1}{MN}$ for $1\le i\le M$, $1\le j\le N$.

So the pair (X,Y) takes values in an $M\times N$ grid of points with each point having the same probability of $\frac{1}{MN}$. Now, using the definition of joint entropy above we get

$$\begin{aligned} H(X,Y) &= \log MN \\ &= \log M + \log N \\ &= H(X)+H(Y) \end{aligned}$$

The fact that $H(X,Y)=\log MN$ is consistent with our earlier observation that a source that takes MN different values with equal probabilities has entropy $\log MN$. The last equality makes use of the same fact; since X and Y are uniformly distributed over their respective alphabets, $H(X)=\log M$ and $H(Y)=\log N$.

Theorem. *For any pair of independent r.v.s X and Y, $H(X,Y)=H(X)+H(Y)$. More generally, $H(X,Y)\le H(X)+H(Y)$ with equality iff X and Y are independent.*

We will not prove the fact that $H(X,Y)\le H(X)+H(Y)$ for any two random variables X and Y, but we will verify that when the two random variables are independent their joint entropy is always the sum of their individual entropies. Recall that if two random variables are independent $P(X=x,Y=y)=P(X=$

$x)P(Y=y)$. Therefore,

$$\begin{aligned}
H(X,Y) &= \sum_{x\in\mathcal{X}}\sum_{y\in\mathcal{Y}} P(X=x,Y=y)\log\frac{1}{P(X=x,Y=y)} \\
&= \sum_{x\in\mathcal{X}}\sum_{y\in\mathcal{Y}} P(X=x)P(Y=y)\log\frac{1}{P(X=x)P(Y=y)} \\
&= \sum_{x\in\mathcal{X}}\sum_{y\in\mathcal{Y}} P(X=x)P(Y=y)\left(\log\frac{1}{P(X=x)}+\log\frac{1}{P(Y=y)}\right) \\
&= \sum_{x\in\mathcal{X}}\sum_{y\in\mathcal{Y}} P(X=x)P(Y=y)\log\frac{1}{P(X=x)} + \sum_{x\in\mathcal{X}}\sum_{y\in\mathcal{Y}} P(X=x)P(Y=y)\log\frac{1}{P(Y=y)} \\
&= \sum_{x\in\mathcal{X}} P(X=x)\log\frac{1}{P(X=x)}\left(\sum_{y\in\mathcal{Y}} P(Y=y)\right) + \sum_{y\in\mathcal{Y}}\log\frac{1}{P(Y=y)}\left(\sum_{x\in\mathcal{X}} P(X=x)\right) \\
&= \sum_{x\in\mathcal{X}} P(X=x)\log\frac{1}{P(X=x)} + \sum_{y\in\mathcal{Y}}\log\frac{1}{P(Y=y)} \\
&= H(X)+H(Y)
\end{aligned}$$

The definition of joint entropy and the above fact extends to any number of random variables: let $X_1, X_2, \ldots, X_n$ be independent random variables. Then

$$H(X_1, X_2, X_3, \ldots, X_n) = \sum_{i=1}^{n} H(X_i). \tag{1}$$

2 Representing information with bits

As you saw in lecture 1, much of the recent advances in information technology are built around digital systems which use bits for representing information. Towards the end of this class, you will see why that is in fact a really good idea. For now, we will develop methods for representing information using bits as efficiently as possible.

We will work with an example of representing DNA, with alphabet $\mathcal{X} = \{a, g, c, t\}$ where each of the letters represent a nucleotide base in DNA. At a simplified level, we can think of a DNA sequence as being a long string with each character being one of the 4 nucleotides in $\mathcal{X}$.

Question: Given a segment of DNA (say, a gene), that looks like $agccattagc\ldots ccgta$, how many bits do we need to represent it?

In the absence of any other information, one might go with a mapping $\{a \to 00, g \to 01, c \to 10, t \to 11\}$ which uses 2 bits per nucleotide. But if we know something about the statistics of the sequence, we can do better.

Suppose the DNA is from a species characterized by a r.v. X with probabilities of a, g, c, t as shown in the table below. We can then design a new code for this distribution with different codeword lengths for different nucleotides.

nucleotide	probability	codeword	codeword length
a	$\frac{1}{2}$	`0`	1
g	$\frac{1}{4}$	`10`	2
c	$\frac{1}{8}$	`110`	3
t	$\frac{1}{8}$	`111`	3

Intuition:

- We assign shorter codewords to more frequent nucleotides to decrease the average length of representation.
- We care about more than just handling a single nucleotide, instead our code should be able to work with long sequences (explained in more detail below).

Prefix-free code: A prefix-free code is a code where no codeword is a prefix of another codeword.

Example: The code in the table above is a prefix-free code. Note that the encoding and decoding for a prefix-free code can be done efficiently in real-time and on-the-fly by simply reading the sequence from left to right. As an example, suppose we encode *agcca* to `0101101100`. In order to decode this back, we can parse the bits from left to right, just looking for a valid codeword from the table. As soon as we encounter a codeword we can stop and decode since we know that no other codeword can have this codeword as a prefix. For example, if you see `10`, you know that this is a g, since none of the other codewords begin with a `10`. With this understanding, we can parse back to `0,10,110,110,0` and decode successfully.

Average code length: For a given source distribution and a code, we use $\bar{l}$ to denote the expected number of bits per source symbol.

Example: For this scheme, $\bar{l}$ (expected number of bits per nucleotide) is simply

$$\bar{l} = \frac{1}{2} \times 1 + \frac{1}{4} \times 2 + \frac{1}{8} \times 3 + \frac{1}{8} \times 3 = 1.75$$

Thus, $\bar{l} = 1.75$ bits/nucleotide which is better than the 2 bits/nucleotide achieved with the fixed length code.

Also note that, in this case the entropy of the source is

$$\begin{aligned} H(X) &= \frac{1}{2} \times \log 2 + \frac{1}{4} \times \log 4 + \frac{1}{8} \times \log 8 + \frac{1}{8} \times \log 8 \\ &= \frac{1}{2} \times 1 + \frac{1}{4} \times 2 + \frac{1}{8} \times 3 + \frac{1}{8} \times 3 \\ &= 1.75 \end{aligned}$$

This match between average length and entropy is not a coincidence and we will investigate this in more detail in the next lecture.

But, what about the following scheme:

nucleotide	probability	codeword	codeword length
a	$\frac{1}{2}$	`0`	1
g	$\frac{1}{4}$	`1`	1
c	$\frac{1}{8}$	`01`	2
t	$\frac{1}{8}$	`11`	2

Clearly, this scheme assigns different codewords to different nucleotides, and appears to be lossless. Furthermore,

$$\bar{l} = \frac{1}{2} \times 1 + \frac{1}{4} \times 1 + \frac{1}{8} \times 2 + \frac{1}{8} \times 2 = 1.25$$

which is better than the previous scheme. But, there's a catch here. As mentioned above, we are usually interested in encoding not just a single symbol, but a succession of symbols in a sequence. This scheme does not fare well in that scenario, for example, both *agt* and *cgg* have the same encoding `0111` which makes decoding impossible. With this intuition, we can more formally define "information-lossless" and restate the fact from above.

Uniquely decodable code: A code is uniquely decodable (UD) if every **sequence** of source symbols is mapped to a **distinct** bit representation.

Note that every prefix-free code is uniquely decodable since it can be uniquely decode using the procedure described earlier.