1 Background and introduction

Throughout this talk, $\mathcal{X}$ will be an arbitrary finite set. X will be a random variable valued in $\mathcal{X}$, and for $x \in X$, we denote the point probability at x by $p(x) := \Pr(X = x)$ (and analogously for $\mathcal{Y}, Y$, and $p(y)$). Our main object of study will be the *entropy* of X, defined by

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

Here, and for the rest of the talk, log will denote the base-2 logarithm, though this doesn't really matter, and a different choice will simply scale everything by the same constant. Note that we can also write the entropy as

$$H(X) = \mathbb{E}_X \left[\log \frac{1}{p(X)} \right] = -\mathbb{E}_X [\log p(X)],$$

which will be a useful interpretation at times.

The first notion of the information contained in X, proposed by Hartley, was simply $\log|\mathcal{X}|$. This is not totally crazy, for the following reason: suppose we first sample X (distributed on $\mathcal{X}$) and then Y (distributed on $\mathcal{Y}$), which is independent of X. According to this definition, the amount of information we get in total is

$$\log|\mathcal{X} \times \mathcal{Y}| = \log|\mathcal{X}| + \log|\mathcal{Y}|,$$

which is the sum of the amount of information we got from X and Y. Of course, the problem with Hartley's definition is that it doesn't take the distribution of X into account: intuitively, a variable that takes on one variable 99% of the time and another 1% of the time contains less information than one that is uniform on these two outcomes. This is because we are much more likely to correclty predict the first variable, so we learn less new information by seeing an outcome of it.

To fix this, let's begin by thinking about how much information content is delivered by observing an outcome $x \in \mathcal{X}$ when we sample X. I claim that $-\log p(x)$ is a good measure of this information content. To see why, suppose we are trying to guess a secret number between 1 and 8 by asking yes/no questions. The standard strategy for this is binary search, where in each question we seek to cut the search space in half. In that case, every outcome we get to a question we ask has probability 1/2, and we will be done in 3 moves. Thus, the total amount of information content we get is

$$-\log \frac{1}{2} - \log \frac{1}{2} - \log \frac{1}{2} = 3,$$

which makes sense since $\log 8 = 3$. Now, suppose that we take the alternate strategy of asking, "Is it 1? Is it 2?" etc. In the first question, we have a 7/8 chance of getting the answer *no* and a 1/8 chance of getting the answer *yes*. Assuming we first saw a *no*, our probabilities for the second question are 6/7 for *no* and 1/7 for yes. In general, suppose we

finally get a *yes* when there are n possibilities left. Then the total amount of information content we get is

$$-\log\frac{7}{8}-\log\frac{6}{7}-\cdots-\log\frac{n}{n+1}-\log\frac{1}{n}=-\log\left(\frac{7}{8}\cdot\frac{6}{7}\cdots\frac{n}{n+1}\cdot\frac{1}{n}\right)=-\log\frac{1}{8}=3.$$

Moreover, you can convince yourself that no matter what guessing strategy we employ, and no matter what sequence of outcomes we observe, we will always gain a total of 3 bits of information content from these outcomes. This suggests that $-\log p(x)$ is really a worthwhile notion of information content. With this interpretation, we see that $H(X)=\mathbb{E}_X[-\log p(X)]$ is precisely the expected amount of information content we gain from one sample of X, which is a good measure of the information contained in X.

It is also worth mentioning that one can write down a few axioms we might want a function that measures information to have, and then prove that these axioms uniquely determine the function up to scaling. The scaling corresponds to the fact that we can choose a different base for the log (or, equivalently, a different unit of information content than the bit), and this unique function is exactly the entropy.

2 Shannon's coding theorems

Before defining all the relevant terms, we will (informally) state Shannon's three main "coding theorems". All of them are very different and deal with different setups, but all can be interpreted as saying that there is some quantity associated to a random variable that controls how much information it contains, and, moreover, this quantity is the *same* in all three cases: its entropy.

Theorem 1 (Noiseless coding theorem, informal)**.** *If X is encoded as a collection of binary strings which can be uniquely decoded, then the average string length is at least $H(X)$. Moreover, every X can be encoded with average string length at most $H(X)+1$.*

Theorem 2 (Noisy coding theorem, informal)**.** *Let $X_1,\ldots,X_n$ be iid copies of X. The vector $(X_1,\ldots,X_n)$ can be encoded in $nH(X)$ bits with arbitrarily small probability of error. However, if we encode in fewer than $nH(X)$ bits, then the probability of error tends to 1 as $n\to\infty$.*

Theorem 3 (Channel coding theorem, informal)**.** *Let W be a "noisy channel", which takes an input from $\mathcal{X}$ and (randomly) ouputs an element of $\mathcal{Y}$. There exists a number $C=C(W)$, called the channel capacity, with the following properties. First, we can always communicate across W with negligible error probability if we send fewer than C bits of information per bit sent over W. Second, if we try to transmit more than C bits of information per bit sent, then the error probability tends to 1 no matter how we do this. Finally, if the noise in W is gotten by "adding" a random variable X, then $C(W)=1-H(X)$. In other words, exactly $H(X)$ bits of information are destroyed by this noise.*

3 Noiseless coding

Definition 1. Let $\{0,1\}^*$ denote the set of all finite binary strings. A map $f : \mathcal{X} \to \{0,1\}^*$ is called a *prefix code* if for every $x \neq x' \in \mathcal{X}$, $f(x)$ is not a prefix of $f(x')$.

Example. The set $\{0, 10, 11\}$ forms a prefix code, as does $\{00, 01, 101, 111\}$. The set $\{0, 01, 10\}$ does not, since 0 is a prefix of 01.

The reason prefix codes are useful is the following. Suppose we keep getting samples from $\mathcal{X}$, and we transmit them one after another according to a prefix code $\mathcal{X} \to \{0,1\}^*$. Then the resulting long binary string can be uniquely decomposed into codewords by the prefix-free property, and therefore the sequence of elements of $\mathcal{X}$ can be uniquely determined. For instance if we use the prefix code $\{a, b, c\} \mapsto \{0, 10, 11\}$ and we see the sequence

$$0010111110011,$$

then we can decompose this as

0	0	10	11	11	10	0	11

and read off the message *aabccbac*.

The basic property of prefix codes is the following, which says that there must be a tradeoff in the lengths of the codewords.

Proposition 1 (Kraft's inequality)**.** *Let $f : \mathcal{X} \to \{0,1\}^*$ be a prefix code, and let $\ell(x)$ denote the length of the binary string $f(x)$. Then*

$$\sum_{x \in \mathcal{X}} 2^{-\ell(x)} \leq 1$$

Proof. Consider a uniformly random infinite binary string S. By the prefix-free property, at most one codeword $f(x)$ can be a prefix of S. Thus,

$$\Pr(\exists x \in \mathcal{X} \text{ s.t. } f(x) \text{ is a prefix of } S) = \sum_{x \in \mathcal{X}} \Pr(f(x) \text{ is a prefix of } S) = \sum_{x \in \mathcal{X}} 2^{-\ell(x)},$$

where the final equality follows since a given binary string of length ℓ will be a prefix of S with probability exactly $2^{-\ell}$. However, as the left-hand side is some probability, it is upper-bounded by 1, giving the desired bound. □

Using this, we can state and prove one direction of Shannon's noiseless coding theorem.

Theorem 4 (Noiseless coding theorem)**.** *Let X be a random variable on a state space $\mathcal{X}$, and let $f : \mathcal{X} \to \{0,1\}^*$ be a prefix code. Let $\ell(x)$ denote the length of $f(x)$. Then*

$$\mathbb{E}_X[\ell(X)] \geq H(X).$$

Proof. We write

$$\begin{aligned} H(X) - \mathbb{E}_X[\ell(X)] &= \mathbb{E}_X\left[\log \frac{1}{p(X)} - \ell(X)\right] \\ &= \mathbb{E}_X\left[\log \frac{1}{p(X)2^{\ell(X)}}\right] \\ &\leq \log \mathbb{E}_X\left[\frac{1}{p(X)2^{\ell(X)}}\right] \\ &= \log \sum_{x \in \mathcal{X}} 2^{-\ell(x)} \\ &\leq 0, \end{aligned}$$

where we use Jensen's inequality applied to the (concave) logarithm function, and then Kraft's inequality in the final step. □

This theorem says that if we wish to noiselessly encode X as a prefix code, we must use at least $H(X)$ bits on average. It is also known that such an encoding is always possible using at most $H(X) + 1$ bits on average, meaning that $H(X)$ is essentially exactly the threshold of feasibility. Shannon's original proof of this $H(X) + 1$ encoding was in a certain sense suboptimal, and the more modern way of doing this uses what are called Huffman codes, which are provably optimal in a certain sense. However, I'll skip over all these feasibility results for this talk.

It is also worth mentioning that there is a weaker notion than that of being a prefix code, which is known as being *uniquely decodable*. This notion is exactly what you think it is according to the above discussion: no matter what sequence in $\mathcal{X}$ we wish to send, the resulting bit sequence can be uniquely decomposed as above. Uniquely decodable codes also satisfy Kraft's inequality (though here it usually called the Kraft–McMillan inequality), but it is a bit harder to prove in this more general setting. But because of this, Shannon's noiseless coding theorem holds for them as well, which in particular implies that there is no real advantage to using uniquely decodable codes over prefix codes.

4 Noisy coding

In this case, the compression we want to do is of a different sort. Now, we let $n \to \infty$, and wish to map $\mathcal{X}^n \to \{0,1\}^k$ for some "small" k. If we want to have no chance of error, then this map must be injective, and we basically can't do anything. In the previous section, we dealt with this problem by not mapping into strings of a fixed length, and we simply ensured that the *average* length was short. Here, however, we really want all lengths to be short. To deal with the fact that this is impossible, we will allow ourselves some small chance of error.

More formally, we will consider an *encoder* $E : \mathcal{X}^n \to \{0,1\}^k$ and a *decoder* $D : \{0,1\}^k \to \mathcal{X}^n$. We are interested in getting a small error probability, defined by

$$p_e = \Pr(D(E(X_1, \ldots, X_n)) \neq (X_1, \ldots, X_n)),$$

where $X_1, \dots, X_n$ are iid copies of X.

Theorem 5 (Noisy coding theorem)**.** *For every $\varepsilon > 0$, the following holds. If $k = n(H(X) + \varepsilon)$, then there exist an encoder $E : \mathcal{X}^n \to \{0,1\}^k$ and a decoder $D\{0,1\}^k \to \mathcal{X}^n$ such that $p_e \to 0$ as $n \to \infty$. However, if $k = n(H(X) - \varepsilon)$, then $p_e \to 1$ no matter what encoder and decoder we choose.*

In other words, as $n \to \infty$, the threshold for encoding X is exactly $H(X)$ bits: any arbitrarily larger slack than this and we can encode with vanishing error probability, but any smaller and we are essentially guaranteed to get errors no matter what we do.

Proof sketch. For every $(x_1, \dots, x_n) \in \mathcal{X}^n$, let $p(x_1, \dots, x_n)$ denote the probability $\Pr(X_1 = x_1, \dots, X_n = x_n)$, and consider the random variable $Y_n = -\log p(X_1, \dots, X_n)$. Observe that

$$\mathbb{E}[Y_n] = \mathbb{E}[-\log p(X_1, \dots, X_n)] = H(X_1, \dots, X_n) = nH(X).$$

Moreover, by independence, we can write

$$Y_n = -\log p(X_1, \dots, X_n) = -\log[p(X_1) \cdots p(X_n)] = \sum_{i=1}^{n} -\log p(X_i).$$

Thus, we see that Y_n is the sum of n iid random variables, $-\log p(X_i)$, each of which has mean $\mathbb{E}[-\log p(X)] = H(X)$. Thus, by the weak law of large numbers, we know that for any $\delta > 0$,

$$\Pr\left(\left|\frac{1}{n}Y_n - H(X)\right| > \delta\right) \overset{n\to\infty}{\longrightarrow} 0.$$

Equivalently, suppose we define the *typical set* $T_{n,\delta} \subseteq \mathcal{X}^n$ by

$$\begin{aligned} T_{n,\delta} &= \left\{(x_1, \dots, x_n) \in \mathcal{X}^n : 2^{-n(H(X)+\delta)} \le p(x_1, \dots, x_n) \le 2^{-n(H(X)-\delta)}\right\} \\ &= \left\{(x_1, \dots, x_n) \in \mathcal{X}^n : H(X) - \delta \le -\frac{1}{n}\log p(x_1, \dots, x_n) \le H(X) + \delta\right\}. \end{aligned}$$

Then what we've found is that this typical set is indeed typical, in the sense that

$$\Pr((X_1, \dots, X_n) \in T_{n,\delta}) \overset{n\to\infty}{\longrightarrow} 1.$$

Moreover, we see that the elements of $T_{n,\delta}$ are "almost" equiprobable, in the sense that each $(x_1, \dots, x_n) \in T_{n,\delta}$ has probability between $2^{-n(H(X)+\delta)}$ and $2^{-n(H(X)-\delta)}$; this is known as the *asymptotic equipartition property*. From these bounds, it is also simple to compute that $|T_{n,\delta}| \approx 2^{nH(X)}$, for an appropriate notion of $\approx$.

Using this, it is easy to prove the theorem. First, to find an encoding that works and uses $k = n(H(X)+\varepsilon)$ bits, we pick an arbitrary injection $T_{n,\delta} \hookrightarrow \{0,1\}^k$, and extend it arbitrarily to a map $\mathcal{X}^n \to \{0,1\}^k$. With probability tending to 1, when we sample $(X_1, \dots, X_n)$, we will fall into the typical set, so our encoding will be injective on the inputs we are likely to see, meaning that we'll be able to decode; the only way an error can happen is if we land outside

the typical set, which happens with vanishing probability. For the reverse direction, we see by the pigeonhole principle that no matter how we map $T_{n,\delta} \to \{0,1\}^k$ where $k = n(H(X) - \varepsilon)$, we will get many collisions. But all elements of $T_{n,\delta}$ are "essentially" equiprobable, so we will encounter one of these collisions with high probability, meaning that we will not be able to decode without a high probability of error. □

5 Channel coding

Definition 2. A *channel* $W : \mathcal{X} \to \mathcal{Y}$ is a random function from $\mathcal{X}$ to $\mathcal{Y}$. More precisely, W is a collection of probability distributions on $\mathcal{Y}$, one for every $x \in \mathcal{X}$, denoted $p(y \mid x)$. If $x \in \mathcal{X}$ is the input to the channel, then the outupt is distributed according to $p(\cdot \mid x)$.

Example. The *binary erasure channel (BEC)* with probability p is the channel with $\mathcal{X} = \{0,1\}$ and $\mathcal{Y} = \{0,1,?\}$, and transition probabilities given by

$$p(? \mid 0) = p(? \mid 1) = p, p(0 \mid 0) = p(1 \mid 1) = 1 - p, p(0 \mid 1) = p(1 \mid 0) = 0.$$

In other words, each symbol is erased (replaced with ?) with probability p, and kept unchanged with probability $1 - p$. There is no chance of confusing the symbols.

Example. The *binary symmetric channel (BSC)* with probability p is the channel with $\mathcal{X} = \mathcal{Y} = \{0,1\}$, and transition probabilities

$$p(0 \mid 0) = p(1 \mid 1) = 1 - p, p(0 \mid 1) = p(1 \mid 0) = p.$$

In other words, every symbol is swapped with the other one with probability p, and kept unchanged with probability $1 - p$. Note that the BSC can equivalently be thought of as adding to the input a random variable $X \sim \mathrm{Ber}(p)$, where addition is done mod 2.

In order to communicate across a noisy channel, we will apply an *error-correcting code*. Namely, we wish to pick an encoder $E : \mathcal{X}^k \to \mathcal{X}^n$ and a decoder $D : \mathcal{Y}^n \to \mathcal{X}^k$, so that when we apply the channel $W^n : \mathcal{X}^n \to \mathcal{Y}^n$ (where each coordinate is run through W independently), we can decode with low probability of error. Namely, we define

$$p_e = \max_{x \in \mathcal{X}^k} \Pr(D(W(E(x))) \neq x),$$

where the probability is over the randomness in W. The *rate* of such an encoding is defined to be k/n, which measures the number of symbols of information we can send per symbol transmitted.

Theorem 6 (Channel coding theorem)**.** *For every channel W, there is a number $C = C(W)$ (called the* channel capacity*) such that the following holds. For every $R < C$, and $\varepsilon > 0$, there is an encoding algorithm with rate $\geq R$ and $p_e < \varepsilon$, for n sufficiently large. However, for every $R > C$, every encoding algorithm has $p_e > 1 - \varepsilon$ for all n sufficiently large.*

Shannon's proof of this theorem was one of the earliest uses of the *probabilistic method*. Rather than exhibiting an encoding algorithm that achieves this, he instead picked a code at random, and proved that with positive probability, it has the desired properties.

Rather than prove the theorem in full generality (which is somewhat difficult), we will sketch a proof technique that demonstrates a connection to the noisy source coding discussed in the previous section. Suppose for now that $\mathcal{X} = \mathbb{F}_2$ is the finite field with two elements, and that the channel W is the BSC with parameter p. In other words, W works by sampling a random variable $X \sim \text{Ber}(p)$ and then adding it to the input (where addition is done in the field $\mathbb{F}_2$). Suppose additionally that we restrict our attention to *linear* codes, where the functions $E : \mathbb{F}_2^k \to \mathbb{F}_2^n$ and $D : \mathbb{F}_2^n \to \mathbb{F}_2^k$ are required to be linear maps between the relevant vector spaces over $\mathbb{F}_2$. If we run an input $u = (u_1, \dots, u_n)$ through the channel W, the output will be

$$(Y_1, \dots, Y_n) = (u_1, \dots, u_n) + (X_1, \dots, X_n),$$

where $X_1, \dots, X_n$ are iid Bernoulli random variables. Since we assume that the encoding function E is linear, there exists some map $P : \mathbb{F}_2^n \to \mathbb{F}_2^{n-k}$ whose kernel is the image of E. Applying P to Y, we see that

$$P(Y_1, \dots, Y_n) = P(X_1, \dots, X_n),$$

since $P(u_1, \dots, u_n) = 0$ by the choice of P. Now, if we can recover $(X_1, \dots, X_n)$ from its image under P in $\mathbb{F}_2^{n-k}$, then we can also recover the original message. So the channel coding problem has turned into a compression problem, of compressing $(X_1, \dots, X_n)$ into $n-k$ bits. Other than the (important) fact that we are restricting ourselves to linear maps, this is precisely the situation we dealt with in the previous section, and we know that the threshold for possibility is when $n - k = nH(X)$, or equivalently when the rate k/n is $1 - H(X)$. In other words, assuming everything done in the previous section works with linear maps, we have essentially proven the channel coding theorem for this special case, and identified the capacity as $1 - H(X)$. In other words, in this channel, where a random noise drawn from X is added to every input symbol, exactly $H(X)$ bits of information are destroyed per bit sent.

6 Polar codes

Since Shannon's proof of the channel coding theorem uses a random code, the main question left open by Shannon's work was how to explicitly construct the codes that he proved exist. Despite a lot of very beautiful work that gave rates very close to the channel capacity C, it remained open for 60 years how to actually achieve the capacity. These were finally constructed by Arıkan in 2009, in a shockingly simple and elegant construction. I'll sketch the construction of how these *polar codes* can be used to achieve capacity for source coding, namely how to encode a sequence of n iid Bernoulli random variables into essentially $nH(p)$ bits with vanishing error probability. Since this compression will be linear, this will also solve the corresponding channel coding problem above; however, it is important to note that this construction is actually substantially more general, and works for essentially every channel.

To start, let X_1, X_2 be two independent $\mathrm{Ber}(p)$ random variables, and consider the new variables

$$U_1 = X_1 + X_2 \qquad U_2 = X_2.$$

Note that since this simple linear transformation is invertible, we have that

$$H(U_1, U_2) = H(X_1, X_2) = 2H(p).$$

Moreover, we can write

$$H(U_1, U_2) = H(U_1) + H(U_2 \mid U_1),$$

by an important property of entropy called the *chain rule*; essentially, this says that the information learned from the pair (U_1, U_2) is the same as the information gotten by first learning U_1 and then, knowing this, learning U_2.

However, note that U_1 is distributed as a Bernoulli random variable with parameter $q = 2p(1-p)$. For $p \neq \frac{1}{2}$, q is strictly closer to $\frac{1}{2}$ than p is. Therefore, $H(U_1) > H(X_1)$. Moreover, by the chain rule and the conservation of entropy, we thus learn that $H(U_2 \mid U_1) < H(X_2)$. In other words, we started with a pair (X_1, X_2) where the information was equally distributed in the two coordinates, and we've generated a new pair where the first coordinate has more information than the second.

Now, given $n = 2^m$ variables $X_1, \ldots, X_n$ we can recursively apply this operation. Namely, we can define the matrix

$$G_n = \begin{pmatrix} 1 & 1 \\ 0 & 1 \end{pmatrix}^{\otimes m}$$

and then the vector $U = G_n X$. Since G_n is invertible, we have that

$$nH(p) = H(X_1, \ldots, X_n) = H(U_1) + H(U_2 \mid U_1) + \cdots + H(U_n \mid U_1, \ldots, U_{n-1}).$$

Moreover, by recursing the above argument and using the martingale convergence theorem, one can show that almost all the terms on the right-hand side converge to 0 or 1. Therefore, there must be exactly $nH(p)$ many of them that converge to 1. Then we may encode the vector $(X_1, \ldots, X_n)$ by only keeping those coordinates of U whose conditional entropy is close to 1, and by doing so we keep essentially all the information of X. One can then use this fact to show that decoding can be done with negligible error probability, which makes intuitive sense because essentially all the information has been kept.