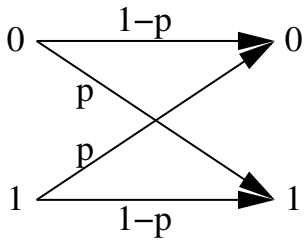
Error Correction and Shannon's Noisy Coding Theorem

Lecturer: Ankur Moitra

1 Channel Coding

Suppose that we have some information that we want to transmit over a noisy channel. Nowadays, this happens all the time: when you're talking on a cell phone, and there is interference from radio waves from other devices; when you're playing a CD (on a good CD player, CD's are remarkably scratch-resistant); when you're downloading files from the Internet. You'd like to make sure that the information gets through intact, even though the line is noisy. How can we do this? Here we will discuss a theoretical result on how much information can be transmitted over a noisy channel. The proof given below that it can be done relies on an algorithm which is hopeless to run in practice because it would take way too long (both for encoding and for decoding). In fact, even writing down the code would take up far too much space. After discussing Shannon's noisy coding theorem, we will introduce some more practical error-correcting codes.

To study the theory of communication mathematically, we first need a mathematical model of a communication channel. Here we focus on one specific channel: the binary symmetric channel. This is a channel where you input a bit, 0 or 1, and with probability $1 - p$ is passes through the channel intact, and with probability p it gets flipped to the other parity. That is, If we have a



message with n bits, the law of large numbers (or Chebyshev) tell us to expect close to pn errors.

This is called a binary channel because the input and output are both bits, and symmetric because the probability of an error is the same for an input of 0 and an input of 1. More generally, a channel is a particular (probabilistic) model for how what you transmit on the channel is mapped to what someone else receives. We could define a general family of channels called the memoryless channel, and give Shannon's theorem for this type of channel, and we won't do this.

1.1 The Repetition Code

How can you transmit information reliably, when you have a noisy channel? There's an obvious way to encode for transmission to ensure that a gets through, which has probably been known for centuries: you just send several copies of the message. This is called a repetition code. In this

scenario, what that means is sending many copies of each bit. Suppose that you replaced each 1 in the original message with five 1's, and each 0 with five 0's. Suppose the probability of error, p, is relatively small. To make an error in decoding the redundant message, at least three copies of each bit would have to be flipped, so the probability of error in a bit decreases from p to

$$\binom{5}{3}(1-p)^2p^3 + \binom{5}{4}(1-p)^1p^4 + \binom{5}{5}p^5.$$

More generally, if you repeat each bit ℓ times and ℓ is odd, then there would need to be more than $\lceil \ell/2 \rceil$ errors in a block in order for decoding by taking the majority vote to fail.

The problem with the repetition code is that it is inefficient. For a fixed length k string of 0's and 1's if we replace each bit with ℓ copies, then the probability of error decreases with ℓ. But the trouble is that if we want a fixed probability p of error, say $p = 1/3$, independent of what k is then the number of times we need to repeat each bit needs to increase with k. A straightforward application of the union bound and the Chernoff bound shows that if we repeat each bit

$$\ell = \Theta(\log k)$$

times then the probability of error can be made to be at most $1/3$. In fact, by choosing the constant hidden in the $\Theta(\cdot)$ term one can make the probability of error be any inverse polynomial in k. This bound is roughly tight, and if we want to keep the probability of error in the repetition code below some fixed constant, then we really would need to repeat each bit a number of times that grows with the number of bits to be encoded. Until 1948, most scientists believed this was inevitable no matter what scheme you chose to encode messages. In a groundbreaking paper in 1948, Claude Shannon showed that this was not true.

1.2 The Capacity

What Shannon showed was that every channel has a *capacity* C. In the case of the binary symmetric channel with $p \leq \frac{1}{2}$, it will turn out to be particularly simple formula

$$C = 1 - H(p)$$

where $H(p)$ is the binary entropy function, i.e., $H(p) = -p\log_2 p - (1-p)\log_2(1-p)$. When discussing the noiseless setting, we had already defined $H(.)$ for a probability distribution p on A. The $H(p)$ defined here corresponds to $H((p, 1-p))$ from the noiseless coding notes (corresponding to the setting with two outcomes, one with probability p, the other with probability $1-p$); this shouldn't create any confusion.

You should think about the capacity as representing for each use of the channel, how many bits of information you can convey to the receiver. In other words, it will be possible to send a $k = (C - \epsilon)n$ bit message by first encoding it into a codeword of n bits, sending it through the channel, and the receiver will be able to decode the original message with a minuscule probability of failure. In fact, not only will the probability of failure not go to one (as we would be worried about in the case of repetition codes if we were to choose ℓ to be a fixed constant) but it will go to zero very quickly as n increases.

This bound is sharp. If you want to send data at a rate strictly larger than the capacity, your error rate will be close to 1. That is, no matter what your encoding, the message decoded by the

receiver will differ from the message encoded by the sender with high probability, and as the length of the message increases, this probability goes to 1.

For the binary symmetric channel with probability $p \leq 0.5$, the capacity is $1 - H(p)$. Observe that the capacity is $1 - H(0) = 1$ if $p = 0$; this was expected since $p = 0$ means no noise. The capacity $1 - H(p)$ decreases as p increases. Also, for $p = 0.5$, the capacity is $1 - H(0.5) = 0$; again, this was expected as one cannot recover anything when the output bits are uniformly distributed.

Here is the notation for the encoding/decoding we will be using. The sender takes a block of Rn bits (the message m), encodes it into some n bits (the codeword c) and sends it through the channel. The receiver gets the output of the channel (the received word $\tilde{c}$ — this is still n bits) and decodes it into a putative message $\tilde{m}$ (Rn bits). We will say that this process is successful when $m = \tilde{m}$, and we would like to make the failure rate small; that is, we want $\mathbb{P}(m \neq \tilde{m}) \leq \epsilon$ no matter what m is. Here R is the ratio of the number of bits in the message to the number of bits we encoded, or the *rate* of the code. We would like the rate to be as large as possible. Note that for any binary channel, we must have $R \leq 1$ (otherwise we would be encoding $Rn > n$ bits into n bits, which is impossible).

2 The Binary Symmetric Channel

We won't state Shannon's theorem formally in its full generality (for memoryless channels), but focus on the binary symmetric channel.

Definition 1. A (k, n)-*encoding function* is a function $\text{Enc} : \{0,1\}^k \to \{0,1\}^n$. A (k, n)-*decoding function* is a function $\text{Dec} : \{0,1\}^n \to \{0,1\}^k$. Call a pair of (k, n) encoding and decoding functions an *encoding scheme*. The *rate* of the encoding scheme is precisely k/n.

The rate of an encoding scheme is a measure of how efficiently it uses the channel, with larger being better. For a fixed n, and a fixed rate $0 < R \leq 1$ (so $k = Rn$; formally, you would round up), what we want to consider is the *best* (Rn, n)-encoding scheme. By this, we mean the encoding scheme that minimizes the probability of decoding incorrectly, for any input message: $\max_{m \in \{0,1\}^{Rn}} \mathbb{P}(\tilde{m} \neq m)$ is as small as possible. As such, define λ_n^* to be this error probability of the *best* (Rn, n)-encoding scheme:

$$\lambda^*(R, n) := \min_{(Rn,n)\text{-encoding scheme}} \max_{m \in \{0,1\}^{Rn}} \mathbb{P}(\tilde{m} \neq m).$$

Now we're ready to formally state Shannon's theorem.

Theorem 1. *Consider the binary symmetric channel with parameter p.*

1. *For any $R > 1 - H(p)$, $\lambda^*(R, n) \to 1$ as $n \to \infty$. (In other words, no matter how clever we are with the coding, the probability of decoding correctly goes to zero.)*
2. *For any $R < 1 - H(p)$, $\lambda^*(R, n) \to 0$ as $n \to \infty$. (In other words, there exist good encoding schemes with this choice of R, and we can make the probability of decoding incorrectly as small as we like, as long as we allow n to be large enough.)*

This states that the capacity of the binary symmetric channel is precisely $1 - H(p)$.

We will need the notion of the *Hamming distance*. The Hamming distance $d_H(s,t)$ between two binary strings s and t of the same length is the number of places where the two strings differ. For example, $d_H(10110100, 00111100) = 2$ since these two strings differ in the first bit and the fifth bit.

We will not formally prove Shannon's theorem but will give the main ideas of the proof. The derivation will have some common features with Shannon's noiseless coding theorem. Recall from there that if one considers random messages of length n with two symbols having probability p and $1-p$, there are $2^{H(p)n}$ *typical* messages (to be more precise, there are $2^{H(p)n+o(n)}$ messages, each arising with probability $2^{-H(p)n-o(n)}$). We will drop these $o(n)$ for simplicity.

We now give the intuition behind the proof of the first part of the theorem. The idea is that we will send one of $M = 2^{Rn}$ codewords. If one of these codewords c is put into the channel, the output $\tilde{c}$ will very likely be in a very thin "ring" around the codeword of radius about pn (measuring distance using the Hamming distance). Analogously to the setting of the noiseless coding theorem, there are approximately $2^{H(p)n}$ words in each of these rings. In order that with high probability we decode the output to the correct codeword, for each of these rings, we need to decode most of the codewords in this ring to the center of the ring, c. Since there are 2^n words altogether, so to make sure that the rings are mostly disjoint, we need to have at most $2^{(1-H(p))n}$ codewords (again, by the pigeonhole principle). If we wanted to formalize this, we would need to replace terms like 'typical', 'most' and 'approximately' with precise events that happen with probability close to 1.

Now for the second part of the theorem. The proof will be *nonconstructive*; it will only show that a good encoding exists, but because it uses the probabilistic method, it won't give us an explicit encoding scheme. The proof will proceed in two main steps:

1. Pick codewords uniformly at random, but twice as many as we will eventually need. It is shown that such a random code is "good on average", in the sense that upon sending a random codeword, it is very likely to be decoded correctly. This is not the end of the story however, since we need to show that every possible codeword can be correctly decoded, with a high probability.

2. After a derandomization step that picks the collection of codewords that is best on average, the "worst half" of the codewords are thrown away, yielding a collection of codewords where every code has a large probability of being correctly decoded.

Let $\mathcal{C} = \{c_1, c_2, \ldots, c_M\}$ be some arbitrary collection of distinct codewords (we'll call this a *codebook*). If we send c_i through the channel, the output is a random code $\tilde{c}_i$. As already discussed, the Hamming distance between c_i and $\tilde{c}_i$ is very likely to be close to np. Choose $\gamma > 0$ as small as possible such that $\mathbb{P}(|d_H(c_i, \tilde{c}_i) - np| > \gamma n) < \epsilon/2$, and define

$$\text{Ring}(c_i) = \{c : |d_H(c_i, c) - np| \leq \gamma n\}.$$

So $\tilde{c}_i \in \text{Ring}(c_i)$ with probability at least $1-\epsilon/2$. Now let $E_i(\mathcal{C})$ denote the event that $\tilde{c}_i \in \text{Ring}(c_i)$, but $\tilde{c}_i \notin \text{Ring}(c_j)$ for any $j \neq i$. In this case, upon receiving $\tilde{c}_i$ we can be very confident about decoding it to c_i. If we receive a code that is not in a unique ring, we won't even try to decode it, we just give up. So what we eventually want is a codebook where $\mathbb{P}(E_i(\mathcal{C}))$ is large for every i. Let $\lambda_i(\mathcal{C}) = 1 - \mathbb{P}(E_i(\mathcal{C}))$, the probability that this event does not occur; we want these to all be small.

Now let's choose $\mathcal{C}$ randomly. Let $M = 2 \cdot 2^{Rn}$ (this is twice as big as we want in the end), and let X_i be a uniformly chosen binary string in $\{0,1\}^n$. Let $\mathcal{C} = \{X_1, \ldots, X_M\}$. Suppose now we pick

a codeword i uniformly at random from $\{1, 2, \ldots, M\}$, and send it through the channel. What is the probability that we cannot decode it, i.e., that the event $E_i(\mathcal{C})$ does not occur?

For a fixed $\mathcal{C}'$ and j, we know that the probability is $\lambda_j(\mathcal{C}')$. What we are asking for is the *average* value of this probability, over our random choice of $\mathcal{C}$ and i. One can show the following lemma.

Lemma 1. *For $\mathcal{C}$, i chosen randomly as described,*

$$\mathbb{E}(\lambda_i(\mathcal{C})) \leq \epsilon, \qquad \textit{if } n \textit{ is large enough.}$$

(Note: the expectation is taken over both the random choice of $\mathcal{C}$ and the random choice of i.)

Let us rewrite this conclusion, $\mathbb{E}(\lambda_i(\mathcal{C}))$ written out explicitly as an average:

$$\frac{1}{\#\text{possible codebooks}} \sum_{\text{possible codebooks } \mathcal{C}} \frac{1}{M} \sum_{i=1}^{M} \lambda_i(\mathcal{C}) \leq \epsilon.$$

There must be some choice of codebook $\mathcal{C}^*$ which does better than average, i.e., for which

$$\frac{1}{M} \sum_{i=1}^{M} \lambda_i(\mathcal{C}^*) \leq \epsilon.$$

So we have found a codebook $\mathcal{C}^* = \{c_1, c_2, \ldots, c_M\}$ that is "good on average"; if we send a random codeword through the channel, we're very likely to be able to decode it. But this doesn't mean that this is true for every codeword; it's possible that $\lambda_j(\mathcal{C}^*)$ is still large for some choices of j.

So let $\mathcal{C}'$ be the codebook consisting of just the best half of the codewords in $\mathcal{C}^*$: those with the smallest values of $\lambda_i(\mathcal{C}^*)$. Then $\lambda_i(\mathcal{C}^*) \leq 2\epsilon$ for all $c_i \in \mathcal{C}'$; for if not, then at least half the codewords in $\mathcal{C}^*$ have $\lambda_i(\mathcal{C}^*) > 2\epsilon$, and we would have

$$\frac{1}{M} \sum_{i=1}^{M} \lambda_i(\mathcal{C}^*) > \frac{1}{M} \cdot (M/2) 2\epsilon = \epsilon,$$

a contradiction; this is Markov's inequality.

So $\mathcal{C}'$ provides a good code (where the probability of an error is at most 2ϵ, where we can choose ϵ as small as we like) with 2^{Rn} codewords, and so it can be used as an (Rn, n)-encoding scheme.

18.200 Principles of Discrete Applied Mathematics

Spring 2024