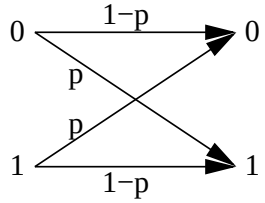
Shannon's Noisy Coding Theorem

Prof. Michel Goemans and Peter Shor

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 remarkable scratch-resistant); when you're downloading stuff 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? What we'll talk about today is a theoretical result on how much information can be transmitted over a noisy channel. The proof that it can be done that you will see today relies on an algorithm which is hopeless to run in practice because it would take way too long. In the next few classes, we'll talk about specific error correcting codes. First, we will discuss Hamming codes, which were the first error-correcting codes to be discovered. Next, we'll discuss BCH codes, which wre one of the first families of codes discovered that worked reasonably well in practice.

To study the theory of communication mathematically, you first need a mathematical model of a communication channel. For the next couple of weeks, we will be talking mainly about 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,



This is 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. Later in this lecture, I will define a more general type of channel, the memoryless channel, and give Shannon's theorem for a this type of channel.

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. 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 around $\binom{5}{3}p^3 = 10p^3$.

The problem with this redundant encoding is that it is inefficient. The longer the string of 0's or 1's you replace each original bit is, the smaller the probability of error, but the more bits you have

to send. Until 1948, most scientists believed this was true; in order to make the probability of error closer to 0, the amount of communication would have to go up indefinitely. In a groundbreaking paper in 1948, Claude Shannon showed that this was not true.

What Shannon showed was that every channel has a capacity. If you want to send data at a rate less than the capacity (generally expressed as bits per channel use ... a channel use for the binary symmetric channel is sending one bit through the channel), then you can reduce the error rate to zero by encoding blocks of $(C-\epsilon)n$ bits into a codeword consisting of n bits, and sending it through the channel. You can reduce the error to zero and make the rate arbitrarily close to C by choosing n large enough. If you want to send data at a rate 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.

2 The Binary Symmetric Channel

We would like to encode a message using this channel so that, after it has passed through the channel, a receiver can decode it and obtain the original message with high probability. For the binary symmetric channel, this capacity $C < 1$, since you're encoding bits into bits, and bits ... If, in a channel use, you could send one of (say) 26 symbols, then the capacity might be larger than 1.

3 Binary symmetric channels

How do we prove Shannon's theorem? What we will do is first prove the theorem (and thus derive the capacity) for the binary symmetric channel, and second sketch how things change when you have a larger class of channels.

For our proofs, we will assume that the coding works as follows: the sender takes a block of Rn bits (the message m), encodes it into some n bits (the codeword c) and send 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 $Prob(m \neq \tilde{m}) \leq \epsilon$. 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. Note that for the binary symmetric channel, we must have $R \leq 1$ (otherwise we would be encoding $Rn > n$ bits into n bits, which is impossible).

At this point, we need to define the Hamming distance $d_H(s,t)$ between two strings s and t. This is the number of places where the two strings differ. For example, $dH(10110100, 00111100) = 2$ since these two strings differ in the first bit and the fifth bit. We should also recall the definition of the binary entropy function $H(p) = -p\log_2 p - (1-p)\log_2(1-p)$.

Before we go into the technical details of the proof, let me give the intuition behind the proof. The idea is that we will send one of M codewords. If one of these codewords $c$is put into the channel, the output $\tilde{c}$ will very likely be in a ball of radius slightly more than pn in Hamming distance around the codeword. In order that with high probability we decode the output to the correct codeword, for each of these balls, we need to decode most of the codewords in this ball to the center of the ball, c. There are approximately $2^{H(p)n}$ words in each of these balls, and there are 2^n words altogether, so to make sure that the balls around the codewords are mostly disjoint,

we need that we have at most $2^{(1-H(p))n}$ codewords.

We will next show that if we choose $2^{(1-H(p)-\epsilon)n}$ codewords at random, then we can find a decoding algorithm which succeeds in decoding them with high probability.

We will assume that there are M messages, so if our input is Rn bits, then $M = 2^{Rn}$. We will also assume that each of these messages is chosen to be transmitted with probability $\frac{1}{M}$. We then have that the probability of error

$$Prob(m \neq \tilde{m}) = \frac{1}{M} \sum_{m=0}^{M-1} Prob(\tilde{m} \neq m | \text{message was } m)$$

It is easy to see from this that there must be at least $M/2$ messages such that the probability of error when each of these is sent is at most $\epsilon/2$.

Now, consider a codeword c. When we send it through the channel, the output is likely going to be a word near c, with the closer messages the more likely. Each possible output $\tilde{c}$ could be decoded to some message $\tilde{m}$, and we would like to make sure that when we put the codeword c corresponding to m into the channel, the sum of the probabilities of the channel outputs $\tilde{c}$ that decode to m is at least $1 - 2\epsilon$. That is, we want

$$\sum_{\tilde{c} \text{ decodes to} m} Prob(\tilde{c}|c) \geq 1 - 2\epsilon.$$

The probability that on channel input c, we get output $\tilde{c}$ is

$$Prob(\tilde{c}|c) = p^{d_H(c,\tilde{c})}(1-p)^{n-d_H(c,\tilde{c})}$$

By the law of large numbers, there is some δ so that with probability $1 - \epsilon$,

$$(p - \delta)n \leq d_H(c, \tilde{c}) \leq (p + \delta)n,$$

whre δ goes to 0 as n goes to ∞. Call this set of words the *typical outputs* on input c. Because with probability $1-\epsilon$ the output is typical, and we need to decode the output correctly with probability $1-2\epsilon$, we need to decode a typical output correctly with probability at least $1-3\epsilon$. Now, a typical output with highest probability is one which has Hamming distance $(p-\delta)n$ from the codeword. The probability of getting any particular one of these ouptuts is

$$\begin{aligned} p^{(p-\delta)n}(1-p)^{(1-p-\delta)n} &= 2^{((p-\delta)\log p + (1-p+\delta)\log(1-p))n} \\ &= 2^{(p\log p + (1-p)\log(1-p))n + \delta' n} \\ &= 2^{(-H(p)+\delta')n} \end{aligned}$$

so to make sure the probability of decoding this output is at least $1 - 2\epsilon$, we need

$$\frac{1-3\epsilon}{2^{(-H(p)+\delta')n}}$$

words that decode to the message m. There must therefore be at least $M/2$ messages which have this many words associated with them. Since no word can decode to more than one message, and there are 2^n words altogether, by the pigeonhole principle we must have at most

$$M/2(1-3\epsilon)2^{(H(p)-\delta')n} \leq 2^n$$

The 2 and the ϵ terms are tiny compared with the δ', so we can absorb them into δ', and we have

$$M \leq 2^{n(1-H(p)+\delta'')},$$

showing that if we want to transmit a message error-free using blocks of n bits, we must have $\log M = Rn \leq n(1-H(p)+\delta'')$. Thus, if we transmit bits at a rate $R \geq 1-H(p)$, for long enough messages we will inevitably introduce errors.

We will next show that we can indeed find a coding scheme which transmits messages almost error-free if $F \leq 1-H(p)$, showing that the capacity of the binary symmetric channel is $1-H(p)$.

We will show that if we choose random codewords, then these will asymptotically achieve Shannon's capacity formula $1-H(p)$ as the blocklength n goes to infinity. We do this by calculating the probability that a random codeword, after transmission through the channel, gets decoded to the wrong codeword.

We choose $2^{(1-H(p)-\epsilon)n}$ codewords at random and use these for our code.

Before we can compute the probability of successful decoding, we need to decide how the receiver should decode the codeword. The best decoding algorithm would be to find the codeword c' which has the highest probability of being taken by the channel into the received word $\tilde{c}$. (This is called the *maximum likelihood* decoding.) For the binary symmetric channel, this is easily seen to be the closest codeword to the received word $\tilde{c}$. However, we won't use this decoding for our calculations. We will use a decoding algorithm for which it is much easier to calculate the probabiltiy of a bad decoding.

Recall that if we take a codeword, and put it through the binary symmetric channel, then with high probability something like pn bits will be flipped. Thus, with probability $1-\epsilon$, the received word $\tilde{c}$ will have Hamming distance no more than $(p+\delta)n$ from the codeword c that was input into the channel, and we need not worry about the case when the more than $(p+\epsilon)n$ bits are flipped, as this is a low probability event.

Our decoding algorithm will be: if there is a unique codeword c' within distance $(p+\delta)n$ from the received word $\tilde{c}$, decode to c'. Otherwise, give up. We will show that this works with high probability.

What is the probability that our received word is not decoded properly? First, with probability $1-\epsilon$, the received word is close enough to the original codeword c. In this case, the only reason for the decoding to fail is if another codeword is also within $(p+\delta)n$ distance from the received codeword. We will compute the probability that our received word is decoded to a specific other codeword c'. Recall that the codewords were chosen at random. Thus, the probability that a specific codeword c' is within distance $(p+\delta)n$ from our received word is exactly the probability that a random binary string is within distance this of c'. This is just the ratio of the number of words close to c' to the total number of words, or approximately

$$\frac{2^{H(p+\delta)n}}{2^n}.$$

Now, if we have $M-1$ possible incorrect codewords, the total probability that one of them is within $(p+\delta)n$ of $\tilde{c}$ is at most

$$(M-1)\frac{2^{H(p+\delta)n}}{2^n} \leq M2^{-(1-H(p))n+\delta' n}.$$

If we choose $2^{(1-H(p))n} - \delta'' n$ codewords for $\delta'' > \delta'$, then the probability that any of them is too close to the received word $\tilde{c}$ is small, so the algorithm decodes correctly, and we are done.