EE 126 Probability and Random Processes
February 22, 2018 Binary Erasure Channel II

1 Reading

Capacity of the Binary Erasure Channel

This is the general setup of a digital communication system: we're sending a message over a noisy channel. In order to do so, we first compress the message into a bitstring (source coding), then add redundancy to deal with the noise in the channel (channel coding), and finally send the coded message over the channel itself. On the receiving end, we perform all of these steps in reverse.

Let's study the special case of the binary erasure channel (which erases the input with probability p). We're interested in the maximum number of bits that we can send per transmission without error.

We'll assume we have a message of length L and are encoding to a message of length n (as a buffer against erasures, n must be larger than L). If the message were to make it through the channel without erasures, the receiver would receive a string of n bits which provides L bits of information. Therefore, the ratio $R = L/n$ is known as the **rate** of the code (the number of bits of information about the source message per symbol the receiver receives). Intuitively, we'd like the rate to be higher, i.e. we'd like n to not be too much larger than L.

We additionally define $\mathcal{X} = \{0, 1\}$ to be the input alphabet, $\mathcal{Y} = \{0, 1, *\}$ to be the output alphabet, $f_n : \mathcal{X}^L \mapsto \mathcal{X}^n$ to be the encoding function, and $g_n : \mathcal{Y}^n \mapsto \mathcal{Y}^L$ to be the decoding function.

If $X^{(n)} = (X_1, ..., X_n)$ represents the n bits which are fed into the channel and $Y^{(n)} = (Y_1, ..., Y_n)$ represents the n bits that come out, then the **maximum probability of error** of the code is

$$P_e(n) = \max_{x \in \mathcal{X}^L} P\{g_n(Y^{(n)}) \neq x \mid X^{(n)} = f_n(x)\}$$

This is the maximum probability that the decoded message is different from the original message, where the maximum is taken over all choices of the input message.

The rate R is called *achievable* if for each positive integer n there exists an f_n and g_n which encode messages of length $L(n) = \lceil nR \rceil$ to messages of length n, such that $P_e(n) \to 0$ as $n \to \infty$. The largest achievable rate of the channel is called the channel **capacity**.

In the case of the BEC, there are $2^{L(n)}$ possible messages.

2 Lecture

BEC

Again, we're interested in understanding the capacity of the BEC. Recall: in a BEC, a one-bit input is erased with probability p, and goes through with probability $1 - p$. The domain is $\{0, 1\}$ and the range is $\{0, 1, *\}$.

We want to send a message m of certain length, and encode it ($X^{(n)}$) before sending it over a binary erasure channel. The BEC will spit out $Y^{(n)}$ before being decoded into $\hat{m}$. The hope is that $\hat{m}$ will be equal to the original message m.

We have $m \in \{0, 1\}$, $X^{(n)} = (X_1, ...X_n)$ for $X_i \in \{0, 1\}$, and $Y^{(n)} = (Y_1, ...Y_n)$ for $Y_i \in \{0, 1, *\}$. The probability of

error after using the channel n times is going to be the probability that $m \neq \hat{m}$:

$$P_e^{(n)} = \max_{m \in \{0,1\}} P(m \neq \hat{m})$$

This is our measure of quality. We'll be judged by our worst error.

Define **rate** as L/n (units: "bits per channel use", range: $[0, 1]$). Say that our message is of length L. We're using the channel n times, and we'll need to use it at least this much (because we have n bits to send). But we'll probably need to use it more, because it's screwing up on us. Hence the rate encodes the redundancy of what we're doing.

Define **capacity** as the max rate at which we can reliably send things, i.e. at which $P_e^{(n)} \to 0$ as $n \to \infty$ (the block length can be as big as we like, but the error should be going to zero). The capacity of the BEC(p) channel is $(1-p)$ bits per channel use.

Imagine we had a genie that told us exactly which np bits would be erased. (This is a more powerful genie than the one described in the previous lecture.) We get to know this *before* we start our transmission. In this scenario, we will just not transmit the soon-to-be-killed bits; we'll only send the $n(1-p)$ bits in the other locations. Now, see that even if we have this powerful genie, we can reliably send only $(1-p)$ bits [per channel use]!

Therefore, the capacity C of the BEC **must** be less than or equal to $(1-p)$ bits per channel use.

Random Coding Argument

We're not done yet. We still need to tangibly achieve the result the genie gave us (in order to show that it is indeed the tightest bound). To do so, we will use *Shannon's random coding based "achievability" scheme.*

Imagine a codebook that gives us a codeword for each message. It's arranged as a $2^L \times n$ grid, where each entry is generated via a fair coin toss and is therefore either 0 or 1. (There are 2^L messages and n channel usages.) Then assume the following setup:

1. The codebook $\mathcal{C}$ is shared by the encoder and the decoder.
2. The encoder takes message i ($i \in \{1, 2, ..., 2^L\}$) and transmits codeword C_i corresponding to the ith row of $\mathcal{C}$.
3. The channel randomly erases np of these bits. (We can assume this as per the laws of large numbers.)
4. WLOG, assume that the last np bits are erased. Now the final np columns of the codebook are gone.
5. The decoder looks for an exact match between the received ${Y'}^{n(1-p)}$ "good bits" and codewords in $\mathcal{C}'$. We know by definition we must have at least one match.

Now the only problem is if multiple messages have the same pattern. If there are two or more matches, we must declare failure. WLOG, assume that the first message is sent. Now

$$\begin{aligned} P_e^{(n)} &= P((C_2' = C_1') \cup (C_3' = C_1') \cup \ ... \ \cup (C_n' = C_1')) \\ &\leq \sum_{i=2}^{M} P(c_i' = c_1') \\ &= \sum_{i=2}^{M} 2^{-n(1-p)} \\ &< M \cdot 2^{-n(1-p)} \end{aligned}$$

Since $M = 2^L$,

$$P_e^{(n)} \leq 2^L 2^{-n(1-p)} = 2^{-n((1-p)-\frac{L}{n}))} \to 0 \text{ as } n \to \infty \text{ if } (1-p) - \frac{L}{n} > 0, \ \text{ i.e. if } \ \frac{L}{n} < (1-p)$$

Let the rate $R = \frac{L}{n}$ and the capacity $C = (1-p)$. Note that $R < C$ (if it were larger, there would be collisions and we would be hosed; we would always be declaring failure). If $R = C(1-\epsilon) \ \ \forall\ \epsilon > 0$,

$$P_e^{(n)} = 2^{-n(C-R)} = 2^{-nC\epsilon} \to 0 \ \text{ as } n \to \infty$$

Therefore we can get arbitrarily close to $C = (1-p)$, and our genie bound is tight. We have achieved a capacity of $(1-p)$ using Shannon's random coding scheme.

Example. Let $n = 10,000$ (we're sending $10,000$), $p = 0.5$ (the channel is going to erase half of our bits), and $\epsilon = 0.01$. We have that $R = C(1-\epsilon) = 0.49$. Therefore, we can send $0.49 \cdot 10,000 = 4900$ actual bits in $10,000$ channel uses, with $P_e \leq 2^{-(10,000)(0.5)(0.01)} = 2^{-50}$ which is negligible.

Note: Shannon's random coding scheme, while brilliant, is neither practical nor constructive. We need tractable codes with efficient computational complexity, which *coding theory* addresses.

Markov Chains

Here, we go from random variables to random processes. A random process is a probabilistic description of a *sequence* of random variables (instead of having one random variable, we have many!). Usually, this is associated with time.

There are many examples of this. For example, the number of students in Prof. Ramchandran's office hours at time t, where we quantize t to 10-minute intervals. Another example would be the number of students enrolled in EE 126 between January 15th and February 20th.

In general, we can describe a random process by describing the *joint distribution* of $(X_{t1}, X_{t2}, ..., X_{tn})$. However, this is not tractable. Therefore, a Markov process makes the simplifying assumption that "given the present, the future is independent of the past." In other words,

$$P(X_{n+1} = x_{n+1} \mid X_n = x_n, X_{n-1} = x_{n-1}, ..., X_1 = x_1) = P(X_{n+1} = x_{n+1} \mid X_n = x_n)$$

We will have states, and links going between them. A **finite Markov chain** assumes that we have a finite set of states $\mathcal{X} = \{1, ..., K\}$ and we're given a probabilistic distribution π_0 on $\mathcal{X}$, where $\pi_0(i) \geq 0$ and $\sum_{i \in \mathcal{X}} \pi_0(i) = 1$. We're also given the transition probability P_{ij} for all i, j in the state space. We need that the P_{ij}s are valid probabilities, i.e. $P_{ij} \geq 0$ and $\sum_j P_{ij} = 1$ for all i.

$$P(X_0 = i) = \pi_0(i),\ i \in \mathcal{X}$$

$$P(X_{n+1} = j \mid X_n = i, X_{n-1}, ..., X_0) = P_{ij} \text{ for all } i, j \in \mathcal{X}$$