CS 229r Information Theory in Computer Science Feb 19, 2019

Lecture 7

Instructor: Madhu Sudan *Scribe: Jane Ahn*

1 Overview

1.1 Outline

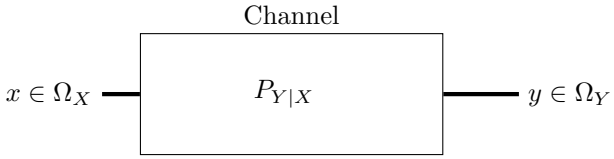
1. Converse Coding Theorems
2. Efficiency in Coding
3. Linear Coding & Linear Compression

1.2 Administrative Things

1. We have a new TF! His name is Noah Golowich
2. Problem set 1 solutions are out
3. Problem set 2 is out
4. No Office Hours for Madhu this Thursday
5. Mitali has Office Hours Tuesday (2/19) at 5pm and Thursday (2/21) at 5pm

2 Channel Coding

Our goal is to recover the original message that has passed through a noisy channel, where we also have an encoding function $E_n : \{0,1\}^{Rn} \to \Omega_X^n$ and a decoding function $D_n : \Omega_Y^n \to \{0,1\}^{Rn}$. A channel looks like this:



If you feed in $x \in \Omega_X$, you get $y \in \Omega_Y$. Every channel is specified by the conditional probability distribution $P_{Y|X}$. Define a channel's capacity to be the following:

$$Cap = \sup_R \{\lim_{\varepsilon \to 0} \lim_{n \to \infty} \{\text{communication of } Rn \text{ bits is possible with } \varepsilon\text{-error during } n \text{ uses of channel}\}\}$$

The capacity is the supremum of all rates R as error goes to zero and the string has sufficient bits. Capacity tells you how well/efficiently you can communicate. Now if $E_n(m)_i \sim P_X$ i.i.d. over m, i, (we proved last time that) these channels have bounded capacity. What we proved last time was for every $\varepsilon > 0$, we can achieve $R \geq \sup_{P_X}\{I(X;Y)\} - \varepsilon$. This means the capacity can get arbitrarily close to the information Y gives about X. This implies that $\sup_{P_X}\{I(X;Y)\}$ is a lower bound for the capacity. We now define

$$C_0 \triangleq \sup_{P_X}\{I(X;Y)\}$$

for notational convenience, and prove the following claim:

Claim 1.

$$Cap = \sup_{P_x}\{I(X;Y)\}$$

2.1 Upper Bounding Capacity

We won't prove the following theorem in class, but it gives a really strong bound:

Theorem 2. *For a BSC(p), for all $\varepsilon > 0$ there exists $\delta > 0$ such that if rate $R > C_0 + \varepsilon$, then $Pr[\text{decoding error}] \geq 1 - exp(-\delta n)$.*

This implies that if the rate is too high, most of the time you get the wrong answer. We will prove the following theorem instead:

Theorem 3. *For all $P_{Y|X}$ (channels) and for all $\varepsilon > 0$, there exists $\delta > 0$ such that if rate $R > C_o + \varepsilon$, then $Pr[\text{decoding failure}] \geq \delta$*

Proof. We want to use the following understanding of the process.

Note that the above process is a Markov chain, which means that if we fix any state, the future is independent of the past. For example, if we fix Y^n, $\hat{m}$ is independent of m, X^n.

Define $\delta = Pr[m \neq \hat{m}]$. We wish to show that $\delta > 0$. In order to understand the probability that $m \neq \hat{m}$, we look at the entropy

$$H(m) = nR$$

since $m \in \{0,1\}^{nR}$. Now by definition of mutual information, we have

$$H(m) = H(m|\hat{m}) + I(\hat{m};m)$$

By the data processing inequality, we see that the information $\hat{m}$ gives about m cannot be larger than the information Y_n gives about m. Similarly, then this amount is at most the amount of information X^n gives about Y^n. Hence,

$$I(\hat{m};m) \leq I(Y^n;X^n) = \sum_{i=1}^{n} I(Y_i; X^n, Y_1, \cdots, Y_{i-1})$$

Now consider the Markov chain

$$(X^n; Y_1, \cdots, Y_{n-1}) \quad \longrightarrow \quad X_i \quad \longrightarrow \quad Y_i$$

Conditioning on X_i, we know that the left and right are independent. Now since each Y_i is independent of Y_j for $j \neq i$, we get that $I(Y_i; X^n, Y_1, \cdots, Y_{i-1}) = I(Y_i; X_i)$. Hence, we upper bound $I(\hat{m};m)$ by

$$I(\hat{m};m) \leq I(Y^n;X^n) = \sum_{i=1}^{n} I(Y_i; X^n, Y_1, \cdots, Y_{i-1}) \leq \sum_{i=1}^{n} I(Y_i; X_i)$$

and further, for any i, $I(Y_i; X_i) \leq C_0$, so

$$I(\hat{m};m) \leq I(Y^n;X^n) = \sum_{i=1}^{n} I(Y_i; X^n, Y_1, \cdots, Y_{i-1}) \leq \sum_{i=1}^{n} I(Y_i; X_i) \leq nC_0$$

Now we consider $H(m|\hat{m})$ using Fano's inequality. We get

$$H(m|\hat{m}) \leq H(1_{m\neq\hat{m}}) + Pr[m \neq \hat{m}] \log |\{0,1\}^{nR}| \leq 1 + \delta nR$$

where $1_{m\neq\hat{m}}$ is the indicator variable for the event $m \neq \hat{m}$. Hence, we get

$$nR \leq 1 + \delta nR + nC_0 \implies \delta nR \geq n(R - C_0) - 1 \geq \varepsilon n - 1 \implies \delta \geq \frac{\varepsilon}{R} - \frac{1}{n}$$

□

3 Efficiency in Coding

Once we start looking at algorithm efficiency, what we have is pretty bad. There are three stages of efficiency:

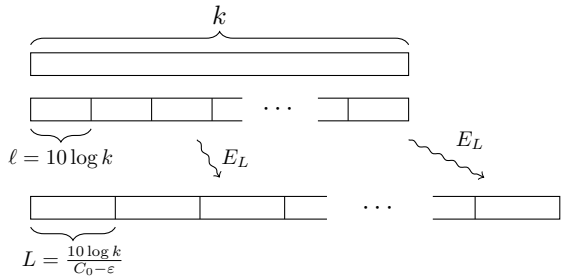
1. Preprocessing: Designing E_n and D_n
2. Encoding Complexity: How much work does it take to compute the encoding given a message
3. Decoding Complexity: How much work does it take to compute the decoding given an encoding

How long does it take to compute E_n? We look at each of the stages:

1. There are 2^{Rn} possible messages; this is the size of the encoding (since if it's completely random we can't compress it anymore): this is the space complexity and the randomized time complexity
2. Encoding still has 2^{Rn} space since we have to look up which encoding to use; time is poly(n)
3. Decoding is still 2^{Rn} space, 2^{Rn} deterministic time complexity (once we have the decoding everything is deterministic), since decoding will go through all possible encodings

Obviously, this isn't that great. But in the next couple lectures we'll develop something that will give us poly time. Note we've assumed so far that probability of error is exponentially low, but this is something we can leverage (we just need probability of error going to 0, not necessarily decreasing exponentially). We can do this by dividing the long block into small chunks and encoding each chunk separately. Moreover, from now on, we are solely going to focus on binary symmetric channels.

Given k bits, we chop it up into lengths of $10 \log k$. We apply Shannon's methodology (which is the breaking of the large block into smaller chunks) independently to each block. Let $l = 10 \log k$ and $L = \frac{10 \log k}{C_0 - \varepsilon}$.



1. The preprocessing cost and space is now $exp(L) = poly(k)$. There is still randomness here.
2. Encoding needs a table where you have to look things up, and this takes now poly time in k (since it is exp in $10 \log k$).
3. Decoding is similar; do brute force on each block - which string is most likely to have produced encoding? Decoding also takes $poly(k)$ now (since it is exp in $10 \log k$).

The error probability is probability of existence of a block that was decoded incorrectly. By the union bound.

$$Pr[\text{there exists a block decoded incorrectly}] \leq k \cdot Pr[\text{block is decoded incorrectly}] \leq k \cdot \frac{1}{k^2} = \frac{1}{k}$$

The inequality holds assuming the condition 10 was large enough. The upper bound value which goes to 0. People probably knew this a long time ago, and we implicitly uses it all the time. But formalization took longer!

However, there are problems with the current solution.

1. The running time of the decoding function is at least $\frac{1}{\text{error probability}}$.
2. There is also a lower bound on how small each of these small blocks can be. Each block should be big enough so that probability of bit flip is "detectable" and so you can check for errors. This is related to the divergence, and you get that the length of block $\geq \frac{1}{\varepsilon^2}$ for $\varepsilon = C_0 - R$. Any smaller and you can't distinguish between different error rates. So the running time is also at least $2^{\frac{1}{\varepsilon^2}}$.

Hence, running time of decoder is at least

$$\max\left\{\frac{1}{\text{error probability}}, 2^{\frac{1}{\varepsilon^2}}\right\}$$

The first problem was resolved by a technique called "concatenated codes" by Forney in 1966. The rough idea is you don't want to take the union bound (effectively worrying about single corruption), but you put in extra encoding before you encode each block separately. You worry about δ fraction corruption (using Chernoff bounds) instead. This probability is exponentially small in k, so error probability is exponential while runtime is polynomial. For the second problem, we had no idea how to get around the $2^{\frac{1}{\varepsilon^2}}$ until 2008 (formally proved in 2013), which uses Polar Codes.

4 Linear Coding $\Leftarrow$ Linear Compression

Suppose the encoding map is linear, i.e. $E_n : \mathbb{F}_2^k \to \mathbb{F}_2^n$. So far, all encoding functions were random, mapping strings of length k to a random string of length n. We get a random linear encoding by picking a random matrix.

Claim 4. *Random Linear Encoding achieves capacity.*

Exercise 5. *Prove the above claim. Could possibly be a problem on the next problem set.*

The nice thing about linear encoding maps is that we only need polynomial space. We only require a $k \times n$ matrix rather than a table of 2^k entries. So it's much more space-efficient. Note that we still have $E_m[\text{decoding incorrectly}] \leq \textit{small}$, so for all messages, probability of decoding incorrectly is small. Further, error detection is easy. Given a message we want to decode, first check whether there is a message for which the string I have could be an encoding of. This can be done with just standard linear algebra. Finally, it is much more likely to be injective.

Proposition 6. *For every full rank matrix $G \in \mathbb{F}_2^{n \times k}$, there exists a full rank $H \in \mathbb{F}_2^{m \times n}$ where $m = n - k$ such that $HG = 0$, i.e. every row of H is orthogonal to every column of G.*

Exercise 7. *Prove the above proposition.*

Exercise 8. *For $x \in \mathbb{F}_2^n$, show that $Hx = 0$ iff there exists m such that $x = Gm$.*

What does H do? H is actually compressing the error...why?

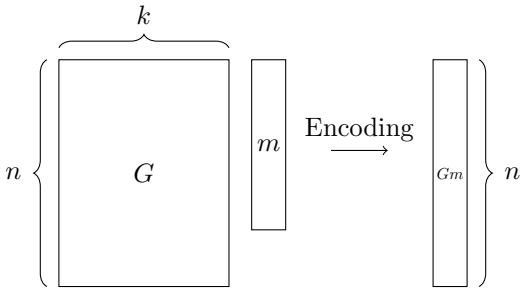
4.1 Efficient Linear Compression for Bern$(p)^n$

Definition 9. *A pair (H, D) is an **efficient linear compression** for Bern$(p)^n$ if*

1. *$H \in \mathbb{F}_2^{m \times n}$, $D \in \mathbb{F}_2^{n \times m}$, and $m \leq (H(p) + \varepsilon)n$*
2. *We have Comp: $\mathbb{F}_2^n \to \mathbb{F}_2^m$ is a linear map given by H and $D : \mathbb{F}_2^m \to \mathbb{F}_2^n$ is efficient; for $Z \in \mathbb{F}_2^n$, Comp$(Z) = HZ$.*
3. *$Pr_{Z \sim Bern(p)^n}[D(HZ) \neq Z] \leq \delta$ for some constant δ*

Given (H, D) we will now construct a good code:

1. Let $G \in \mathbb{F}_2^{(n-m) \times n}$ be the orthogonal matrix, so $HG = 0$.
2. Define encoding as Encoding$(m) = Gm$.



Now how do we decode this? Suppose we have $Gm+Z$ (your encoded message plus Bernoulli independent noise, denoted by Z). Note that recovering m is the same as recovering Z. To get Z, we multiply by H to get

$$H(Gm + Z) = HGm + HZ = HZ$$

and apply the decompressor on HZ. You will usually get Z (this happens with probability $1 - \delta$).

Everything in error correction boils down to compressing the error. The challenge from now on is to compress n Bern(p) bits to $(H(p) + \varepsilon)n$ bits. The decoding time is *poly* $\left(\frac{n}{\varepsilon}\right)$.