CS 229r Essential Coding Theory, Lecture 1 Jan 27, 2020

Lecture 1

Instructor: Madhu Sudan *Scribe: Amir Shanehsazzadeh*

1 Today

- Course Information
- Hamming's Problem (Toy)
- Solutions
- Model/Formation
- Limits

2 Course Information

Welcome to CS 229r: Essential Coding Theory! The Lecturer for this class is Professor Madhu Sudan. His Office Hours are Monday, Wednesday from 4:30-5:30 pm in MD 339. The TF is Chi-Ning Chou. The course website can be found at `http://madhu.seas.harvard.edu/courses/Spring2020/`. Here there is a detailed syllabus, a pointer to the course's unofficial textbook, and additional references. Make sure you have access to the class' Piazza and Canvas sites and that you sign up for scribing. Grading will be determined using Problem Sets (40%), a Final Project (30%), Participation (15%), and Scribing at least one lecture (15%). For lectures before spring break limit the number of scribes to 1. After spring break the limit will be raised to 2. **PSET0 is out and due this Friday!**

The topic of this course is error-correcting codes. We will proceed by constructing and studying codes, proving limits and bounds on the properties of these codes, and creating algorithms that efficiently encode and decode. Note that this is a mathematical course at the Graduate level, so there will be some level of hand-waving. Two significant papers motivate our studies. The first is Shannon '48 and the second Hamming '50. Both Shannon and Hamming were working at Bell Labs. Shannon was thinking about communication devices whereas Hamming was thinking about storage devices and the study information over time. One thing to note is that communication devices allow for a "resend."

3 Hamming's Problem and Framework

Today's lecture is based on Hamming '50. Suppose we have a storage device that can store 1000 bits. In 1 day this device admits at most 1 error.

$$0011011000\mathbf{0}011000 \mapsto 0011011001\mathbf{1}01100.$$

Can we detect a single error if our stored input is $m \in \{0,1\}^{1000}$? The answer is no. To prevent errors we must add some level of redundancy or "wiggle room." Specifically, we want to construct an Encoder and Detector such that

$$m \in \{0,1\}^{?} \to \text{Encoder} \to \{0,1\}^{1000} \to \text{Storage Device} \to \{0,1\}^{1000} \to \text{Detector} \to \begin{cases} \text{YES} & \text{No Errors} \\ \text{NO} & \text{1 Error} \\ \text{Arbitrary} & \text{Otherwise} \end{cases}.$$

4 Naive Solutions

The first solution is to take 500 bits and repeat them twice:

$$010110\cdots \to \text{Encoder} \to 001100111100\cdots.$$

Then our detector returns NO if any adjacent pair of bits differ and otherwise returns YES. The problem with this code is that its rate is $R = \frac{500}{1000} = \frac{1}{2}$.

The second solution is to take 999 bits and XOR them:

$$x_1, ..., x_{999} \to \text{Encoder} \to x_1, ..., x_{999}, \bigoplus_{i=1}^{999} x_i.$$

Our detector then returns the truth value of

$$\bigoplus_{i=1}^{999} x_i = \bigoplus_{i=1}^{999} \hat{x}_i.$$

If a single bit is flipped then the above expression is false, otherwise it is true. This is much better as the rate is $R = \frac{999}{1000}$, which is as good as we can get.

Now we want to be able to *detect* 2, 3, 4, ... errors. Something better though would be to *correct* 1, 2, 3, ... errors. Our framework now aims to find a decoder such that:

$$m \in \{0,1\}^? \to \text{Encoder} \to \{0,1\}^{1000} \to \text{Storage Device} \to \{0,1\}^{1000} \to \text{Decoder} \to \begin{cases} m & \text{At most 1 error} \\ \text{Arbitrary} & \text{Otherwise} \end{cases}.$$

For correcting 1 error we can repeat thrice instead of twice. More specifically we take 333 bits and map:

$$010110\cdots \to \text{Encoder} \to 000111000111111000\cdots.$$

Then if we take the majority vote of every group of three we get the correct bit. We have now achieved a rate of $R = \frac{333}{1000} \approx \frac{1}{3}$.

5 Hamming's Solution Version 1

The naive repetition approach above for correcting a single error is in fact not optimal. We can improve the rate of the code using a more complicated scheme. Hamming's starting solution was to break up sequences of bits into continuous length 4 sequences:

$$x_1x_2... \to (x_1, x_2, x_3, x_4), (x_5, x_6, x_7, x_8), ...$$

Next we map each 4-tuple to a 7-tuple:

$$x = (x_1, x_2, x_3, x_4) \in \mathbb{F}_2^4 \to y = (x_1, x_2, x_3, x_4, x_1 \oplus x_2 \oplus x_4, x_1 \oplus x_3 \oplus x_4, x_2 \oplus x_3 \oplus x_4) \in \mathbb{F}_2^7.$$

Here $\mathbb{F}_2$ is the field with 2 elements. Addition and multiplication are defined modulo 2.

Exercise 1. *Convince yourself (perhaps using casework) that Hamming's solution here works.*

Define the *generator matrix* $G \in \mathbb{F}_2^{4\times 7}$ of the code by:

$$G = \begin{bmatrix} 0 & 0 & 1 & 0 & 1 & 1 & 0 \\ 0 & 1 & 0 & 0 & 1 & 0 & 1 \\ 1 & 0 & 0 & 0 & 0 & 1 & 1 \\ 0 & 0 & 0 & 1 & 1 & 1 & 1 \end{bmatrix} \tag{1}$$

The encoding of a message x is $y = xG$. A generator matrix will always be "fat" and have more columns than rows. Now define $H \in \mathbb{F}_2^{7\times 3}$ as

$$H = \begin{bmatrix} 0 & 0 & 1 \\ 0 & 1 & 0 \\ 0 & 1 & 1 \\ 1 & 0 & 0 \\ 1 & 0 & 1 \\ 1 & 1 & 0 \\ 1 & 1 & 1 \end{bmatrix} \tag{2}$$

Note that each row H is unique and that $GH = 0$. It's easy to verify the latter observation computationally. Now the encoder E maps our message $m = (x_1, x_2, x_3, x_4) \in \mathbb{F}_2^4$ to $E(m) = (x_1, x_2, x_3, x_4) \cdot G$. Let $y = E(m) + e_i$ where $e_i \in \mathbb{F}_2^7$ is a vector with 0s everywhere and a 1 at the ith index. Then we claim that $y \cdot H = (i)$ in binary. Why is this true? We defined $E(m) = m \cdot G$. Then

$$y \cdot H = (E(m) + e_i) \cdot H = m \cdot G \cdot H + e_i \cdot H = e_i \cdot H = (i).$$

One thing to note is that if there are no errors then $y \cdot H = (0)$. What we have done is create a code that returns the index of the error if an error has occurred. In this code G maps a 4 bit object to a 7 bit object and H maps a 7 bit object to a 3 bit object. The rate is $R = \frac{4}{7}$.

6 Hamming's Solution Version 2

Must H map from 7 bits to 3 bits? No. Notice that $7 = 2^3 - 1$. In general we can construct a matrix H_ℓ that maps $2^\ell - 1$ bits to ℓ bits. This matrix H_ℓ is just the binary representation of the value of each row. We can write

$$H_\ell = \begin{bmatrix} 0 & 0 & 0 & \cdots & 0 & 0 & 1 \\ 0 & 0 & 0 & \cdots & 0 & 1 & 0 \\ & & & \vdots & & & \\ 1 & 1 & 1 & \cdots & 1 & 1 & 0 \\ 1 & 1 & 1 & \cdots & 1 & 1 & 1 \end{bmatrix} \tag{3}$$

The above matrix has ℓ columns. Note that $e_i \cdot H = (i) \neq (0)$.

Let $G \in \mathbb{F}_2^{k\times n}$ be full rank with $G \cdot H = 0$. Then the encoder $E_G : \mathbb{F}_2^k \to \mathbb{F}_2^n$ given by $m \mapsto m \cdot G$ corrects one error. The proof of this is identical to the previous proof. Note that

$$(m \cdot G + e_i) \cdot H = m \cdot G \cdot H + e_i \cdot H = e_i \cdot H = (i) \neq (0).$$

This again allows us to locate a potential error and recover m from $m \cdot G$ if G is full rank.

Why must G be full rank? This is for linear algebra purposes. The following exercise shows the importance of G being full rank.

Exercise 2. *Prove that for any $H \in \mathbb{F}_2^{n\times \ell}$ there exists a full rank $G \in \mathbb{F}_2^{(n-\ell)\times n}$ such that $G \cdot H = 0$. This is called the orthogonal space.*

If we take $\ell = 10$ then there exists $G : \mathbb{F}_2^{1013} \to \mathbb{F}_2^{1023}$ and $H \in \mathbb{F}_2^{1023\times 10}$ such that $G \cdot H = 0$. Is this the best we can do? We will see that the answer to this is yes.

7 1000 Bit Limit

What about the fact that our storage device only stores 1000 (and not 1023) bits? It turns out we can essentially remove the last 23 bits in the encoding. To see why this is true we can write out a $2^{1013} \times 1023$ matrix where each row is the encoding of one of the 2^{1013} different elements of $\mathbb{F}_2^{1013}$. Consider the last 23 columns. The values of the elements in these columns can be partitioned into 0, 1, 2, ..., $2^{23}-1$ and by the pigeonhole principle at least one of them must have $2^{1013}/2^{23} = 2^{990}$ elements. Let's suppose that the value 2 (this value is arbitrary) has at least 990 elements $x, y_1, y_2, ..., y_{989}$. Then the bucket with value 0 contains at least 990 elements, specifically they are $x-x, x-y_1, x-y-2, ..., x-y_{989}$. And so we are done.

Why can we not do better? Consider trying to make a code that maps $\mathbb{F}_2^{991}$ to $\mathbb{F}_2^{1000}$. Our code is also defined so that for $m \in \mathbb{F}_2^{991}$ and $x \in \mathbb{F}_2^{1000}$ we have $m \leftrightarrow x$ if $E(m) = x$ or $E(m) \oplus e_i = x$ for some i. If we view our code as a bipartite graph from a set of 2^{991} elements to a set of 2^{1000} elements then the number of edges is easily seen to be $2^{991}(1001) > 2^{1000}$ since $1001 > 2^9 = 512$. But then by the pigeonhole principle there must exist an $x \in \mathbb{F}_2^{1000}$ such that for $m_1, m_2 \in \mathbb{F}_2^{991}$ with $m_1 \neq m_2$ we have $E(m_1) = E(m_2) = x$. Thus our code is not injective.

8 Citation Aside

Interestingly enough, Shannon '48 cites Hamming '50 which cites Golay '49 which cites Shannon '48.

9 Detecting $\equiv$ Correcting

Thus far we have shown how to best detect and correct 1 error. What about detecting and correcting n errors? Hamming showed the amazing result that detecting $2t$ errors is the same as correcting t errors. The idea is to consider a message $m \in \{0,1\}^k$ that is encoded to $E(m) \in \{0,1\}^n$. For $x \in \mathbb{F}_2^n$ we will create the correspondence $m \leftrightarrow x$ if $x = E(m)$ + up to t-many errors.

For an arbitrary discrete alphabet Σ let us define the distance between $x = (x_1, ..., x_n), y = (y_1, ..., y_n) \in \Sigma^n$ as

$$\Delta(x,y) = \#\{i \mid x_i \neq y_i\}..$$

Hamming had a notion of a distance d code $E : \Sigma^k \to \Sigma^n$ (injective). Here the "code" corresponding to E is equal to $C = \{E(m) \mid m \in \Sigma^k\} \subset \Sigma^n$. Hamming's notion of distance is

$$\text{Distance}(C) := \min_{x,y \in C, x \neq y} \{\Delta(x,y)\}.$$

Exercise 3. *Prove that Hamming distance is in fact a metric.*

A code of distance d *detects* $d-1$ errors and *corrects* $\lfloor \frac{d-1}{2} \rfloor$ errors. This is clear geometrically. In our code space $C \subset \Sigma^n$ consider a point $c \in C$. Then a neighborhood of radius $d-1$ about C is disjoint with the rest of C allowing for the detection of $d-1$ errors. Now take two points $a, b \in C$ and take neighborhoods of radius $\lfloor \frac{d-1}{2} \rfloor$ about them. These neighborhoods are disjoint allowing for the correction of $\lfloor \frac{d-1}{2} \rfloor$ errors. See below for a beautiful artist's rendition.

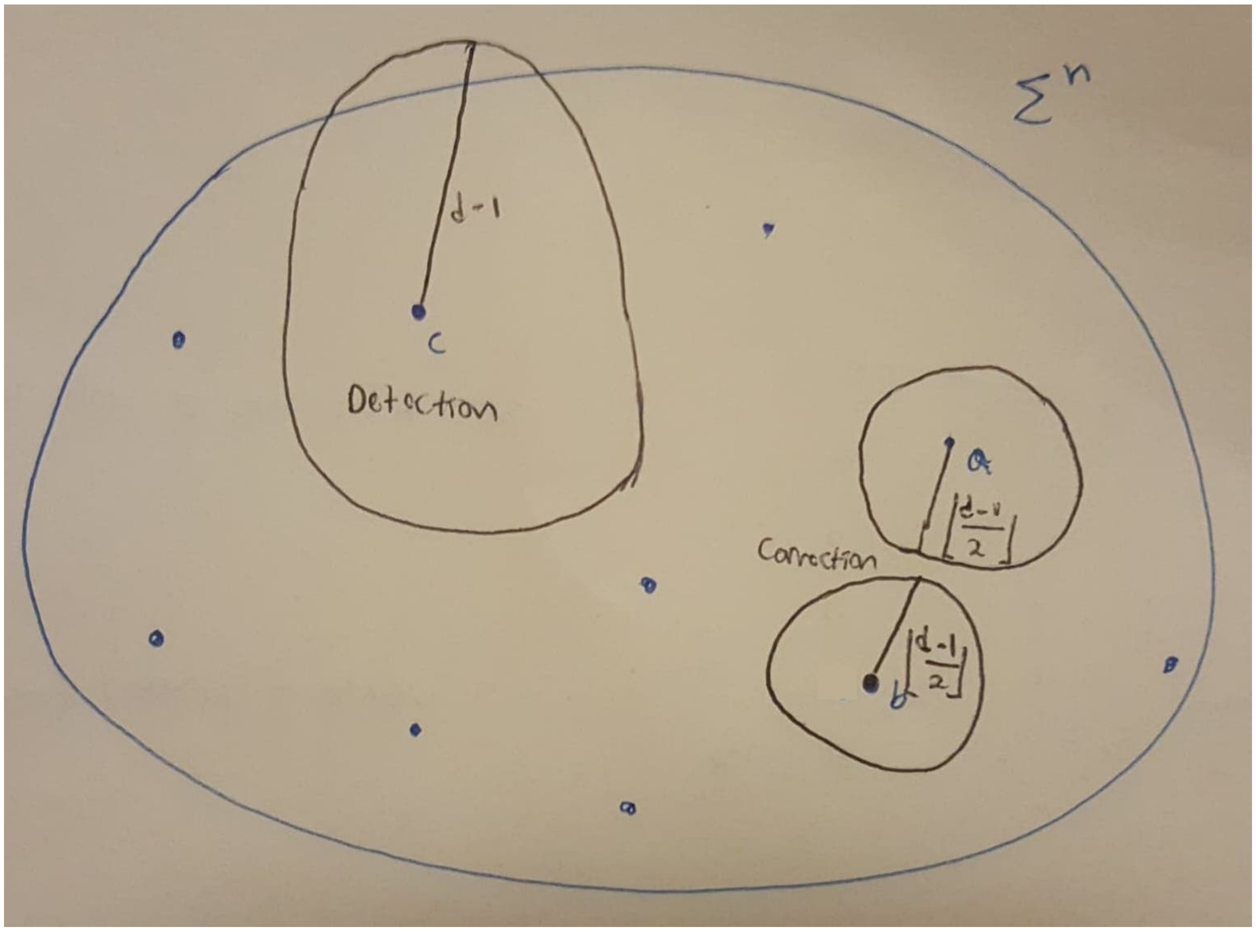
Σ^n
$d-1$
c
Detection
a
$\left\lfloor\frac{d-1}{2}\right\rfloor$
Correction
b
$\left\lfloor\frac{d-1}{2}\right\rfloor$

Figure 1: Geometric Intuition behind Hamming Distance