6.440 Essential Coding Theory Feb 6, 2008

Lecture 1

Lecturer: Madhu Sudan *Scribe: Shira Mitchell*

1 Administrative

Lecturer: Madhu Sudan, 32-G640, madhu@mit.edu
TA: Swastik Kopparty, 32-G636, swastik@mit.edu
website: http://people.csail.mit.edu/madhu/ST08 (which stands for "spring term 2008)
To do:

- Fill up signup sheet to be turned in now, first day of class
- Sign up for scribing, read instructions on course website
- Get added to mailing list by e-mailing course staff
- Look at problem set 1, due in one week
- TF office hours today, Wed. 5-7pm

H level credit, but no ECs, no TQEs, (but anything is possible by petition if you get on your knees and ask)
To pass:

- turn in all psets (3-4)
- do a project - there will be a sign up soon, project will involve reading a paper, understanding it, and trying to make progress on the results. Projects will be done in pairs, you are not allowed to work alone.
- scribe one lecture
- participate actively in lectures

2 Error Correcting Codes [Hamming and Shannon]

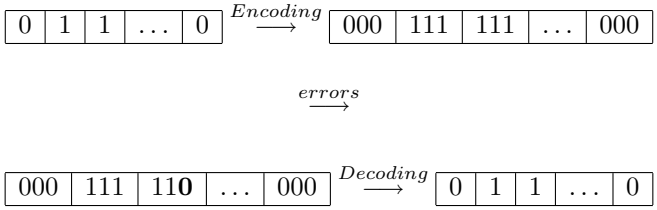
This course is about Error Correcting Codes, and we begin with two seminal papers:

- 1948 - Shannon's paper (next lecture) [2]
- 1950 - Hamming's paper (today) [1]

Today we will discuss Hamming's Problem. Hamming studied magnetic storage devices. He wanted to build (out of magnetic tapes) a reliable storage medium where data was stored in blocks of size 63 (this is a nice number, we will see why later). When you try to read information from this device, bits may be corrupted, i.e. flipped (from 0 to 1, or 1 to 0). Let us consider the case that at most 1 bit in every block of 63 bits may be corrupted. How can we store the information so that all is not lost? We must design an encoding of the message to a codeword with enough redundancy so that we can recover the original sequence from the received word by decoding. (In Hamming's problem about storage we still say "'received word"') There are several possible solutions:

2.1 Naive Solution

To encode a message we repeat every bit 3 times to create a codeword. We consider codewords of length 63 bits, so |message| = 21, which is much less than 63. The received word is a corrupted version of the codeword with at most 1 bit flipped. To decode: in every triple of bits in the received word, we take a majority vote to determine the bit of the message.



Good news - We can efficiently (polynomial-time computable) encode and decode a 21 bit message as a 63 bit codeword with up to 1 bit of error.
Bad news - the *rate* of the code is bad. What does this mean?

Definition 1 *Rate* $\equiv \frac{\textit{message length}}{\textit{codeword length}} \equiv \frac{k}{n}$.

In our case the message length ($\equiv k$) is 21 and the codeword length ($\equiv n$) is 63, so the Rate $= \frac{21}{63} = \frac{1}{3}$. This is not so great! Can we do better? Can we do better while still having encoding and decoding easy?

2.2 Hamming Solution - 1

Break the message into 4 bit chunks. Encode each chuck $m = \square\square\square\square \in \{0,1\}^4$ as follows:
$m \to mG$ where

$$G = \left(\begin{array}{cccc|ccc} 1 & 0 & 0 & 0 & 0 & 1 & 1 \\ 0 & 1 & 0 & 0 & 1 & 0 & 1 \\ 0 & 0 & 1 & 0 & 1 & 1 & 0 \\ 0 & 0 & 0 & 1 & 1 & 1 & 1 \end{array}\right)$$

is called the *Generator matrix*.
Note: the multiplication is over F_2, i.e. modulo 2.
Here are some nice (and surprising) properties:
Claim 1: $\forall m_1 \neq m_2 \in \{0,1\}^4$ any two distinct message chunks, m_1G and m_2G (which are 7 bit strings) differ in ≥ 3 coordinates.
We will prove this later.
Claim 2: This is a sufficient condition to correct any 1 bit flip error
We will also prove this later.
Now we look to another neat property to assist with decoding.
Claim 3: The received word x is either a codeword (i.e. of the form: mG) or of the form: mG + 1 bit flip.
We now consider the following matrix H, known as the *Parity check matrix*:

$$H = \begin{pmatrix} 0 & 0 & 1 \\ 0 & 1 & 0 \\ 0 & 1 & 1 \\ 1 & 0 & 0 \\ 1 & 0 & 1 \\ 1 & 1 & 0 \\ 1 & 1 & 1 \end{pmatrix}$$

Note that xH is a 3 bit vector, which represents some number between 0 and 7.
Claim 4: xH is 0 if no bit was flipped, and otherwise xH will be the coordinate of the flipped bit in x (the 7 bit received word).
We will also not prove this, stare at it on your own.
So how good is this? In this case the message length ($\equiv k$) is 36 (9 chunks of length 4) and the codeword length ($\equiv n$) is 63 (9 chunks of length 7), so the Rate $= \frac{36}{63} = \frac{4}{7}$.
Can we do better? YES!

2.3 Hamming Solution - 2

Hamming proved that there exists a 57 x 63 matrix G and a 63 x 6 matrix H such that:

1. $\forall m_1 \neq m_2 \in \{0,1\}^57$, m_1G and m_2G (which are 63 bit strings) differ in ≥ 3 coordinates.

2. $\forall x$, x is either a codeword (no bit was flipped), in which case xH is 0, or one bit was flipped and xH is the coordinate of the flipped bit in x (the 63 bit received word). Once you know the flipped bit, you can recover the message.

Conclusion: We can acheive Rate $= \frac{57}{63}$.
Hamming showed that this is optimal when you fix 63 as the length of the codeword and at most one bit is flipped. Note that we are not yet discussing asymptotics.
Hamming's paper includes:

- Construction of Error Correcting Code
- Method for encoding and decoding
- Proof and investigation into optimality and limits (beautiful model to prove optimality)
- What can sometimes be overlooked but is very important: Complexity of (algorithms for) decoding

Note: Shannon [2] has all the same features with slightly different models and emphasis as we will see next week.

What is not in Hamming's paper but is very important as well, are the many applications of codes. This course is motivated not only by engineering (computers today would not exist without codes) but by mathematics. Error correcting codes are fundamental in Complexity Theory.

2.4 Hamming's Notions

First now present some definitions.

2.4.1 Hamming Distance

Let Σ be some finite set we call the *alphabet*. In our earlier example $\Sigma = \{0, 1\}$, we could also consider bytes where $\Sigma = \{0, 1\}^8$, or $\Sigma =$ the English letters.

Let Σ^n be the ambient space, which represents the set of n letter words over the alphabet Σ.

Definition 2 *The Hamming distance $\Delta(x, y)$ between $x, y \in \Sigma^n$ is the number of coordinates i where $x_i \neq y_i$, i.e. $\Delta(x, y) = |\{i | x_i \neq y_i\}|$.*

The Hamming distance is a metric since it is easy to verify that:

1. $\Delta(x, y) = 0 \Leftrightarrow x = y$
2. $\Delta(x, y) = \Delta(y, x)$
3. $\Delta(x, z) \leq \Delta(x, y) + \Delta(y, z)$ (triangle inequality)

Thus, we can use geometric intuition to think about Hamming distance.

2.4.2 Codes

Definition 3 *For Hamming, an Error Correcting Code is a set of code words $C \subseteq \Sigma^n$.*

We can also consider codes as the image of an encoding function $\mathbf{E} : \Sigma^k \to \Sigma^n$ that is injective. If $C = \text{Image}(\mathbf{E}) = \{\mathbf{E}(x) | x \in \Sigma^k\}$ then $|C| = |\Sigma^k| = |\Sigma|^k$. Previously we defined Rate as $\frac{\text{message length}}{\text{codeword length}} \equiv \frac{k}{n}$ where k is the message

length. If you do not know the encoding function **E** and its domain, then you define $\text{Rate}(C) \equiv \frac{\log_{|\Sigma|}|C|}{n}$.

A crucial property of a code is the *distance*.

Definition 4 *The distance of C, written $\Delta(C)$, is the minimum Hamming distance between pairs of different code words in C, i.e. $\Delta(C) = min_{x \neq y \in C}\{\Delta(x,y)\}$*

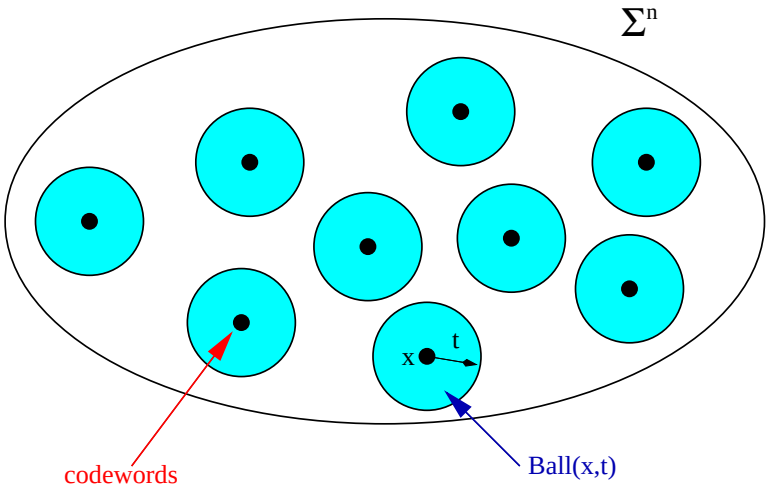
We want codes with large distance, and we will prove that codes of too large a distance cannot exist.

Definition 5 *Informally: A code is t-error correcting if any sequence of $\leq t$ symbol errors can be corrected by a (possibly inefficient) method.*

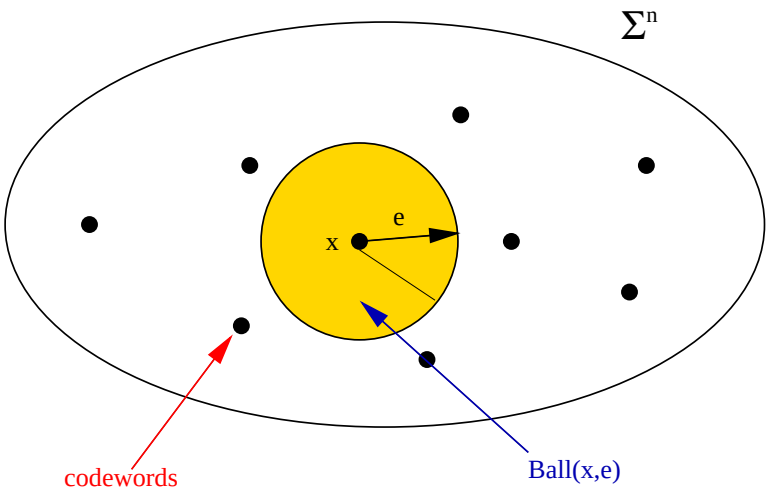
Definition 6 *Informally: A code is e-error detecting if whenever $1 \leq |\{\text{symbol errors}\}| \leq e$, it can be detected by a (possibly inefficient) method that errors have occured. You may not be able to tell where the errors occured.*

To formalize these notions we define:
$\text{Ball}(x,t) = \{y \in \Sigma^n | \Delta(x,y) \leq t\}$, the ball of radius t centered at x.

Definition 7 *Formally: A code C is t-error correcting if $\forall x \neq y \in C$, $Ball(x,t) \cap Ball(y,t) = \emptyset$.*



Definition 8 *Formally: A code is e-error detecting if $\forall x \in C$, $Ball(x,e) \cap C = \{x\}$.*



Hamming proved the following:

Proposition 9 *C is t-error correcting $\Leftrightarrow$ C is $2t$-error detecting $\Leftrightarrow$ $\Delta(C) \geq 2t+1$.*

We did the case where $t = 1$, so we are interested in finding codes C with distance $\Delta(C) = 3$.

The following propositions will help us see that the code in **Hamming Solution - 2** is optimal:

Proposition 10 *For $\Sigma = \{0,1\}$ and $x \in \Sigma^n$, $|Ball(x,t)| = \sum_{i=0}^{t} \binom{n}{i} \equiv Vol(n,t)$*

Proof A vector y is in Ball(x,t) if the number of coordinates of y that differ from x is at most t. Since $\Sigma = \{0,1\}$, they can only differ in one way, namely by being the opposite bit (0 if the bit of x is 1 and 1 if the bit of x is 0). Thus, to count the number of ways to be in the ball, we choose i of the n coordinates for i from 0 to t. We define Vol(n,t) to be the number of points in the ball Ball(x,t). ■

Proposition 11 *For $\Sigma = \{0,1\}$, if C is t-error correcting, then $|C| \leq \frac{2^n}{Vol(n,t)}$.*

Proof If the code C is t-error correcting, then $\forall x \neq y \in C$, Ball$(x,t) \cap$ Ball$(y,t) = \emptyset$, namely, the balls do not intersect. Thus, $|C| *$ Vol$(n,t) \leq$ Vol(Ambient Space). Note that Vol(Ambient Space) $= 2^n$, so dividing gives the result. ■

By the above propositions we can conclude that Rate $= \frac{57}{63}$ is optimal in our example. What about the case of more errors? What if the number of errors grows in proportion to the message length and is not constant? In this course, we will consider codes with an encoding function $\mathbf{E} : \Sigma^{Rn} \rightarrow \Sigma^n$.

3 Overview of Class, Final Notes

We will follow the topics in Hamming's paper outlined above as well as the additional topic of applications of codes to other areas of mathematics and computer science. We will not follow any specific text, but there are many of them available (see website). Most importantly, we will have fun!!

References

[1] Richard W. Hamming. Error Detecting and Error Correcting Codes. *Bell System Technical Journal*, 29:147-160, April 1950.

[2] Claude E. Shannon. A mathematical theory of communication. *Bell System Technical Journal*, 27:379-423, 623-656, 1948.