**Spectral Graph Theory** Lecture 11

# Introduction to Coding Theory

*Daniel A. Spielman* October 7, 2009

## 11.1 Overview

In this lecture, I introduce basic concepts from combinatorial coding theory. We will view error-correcting codes from a worst-case perspective, focusing on minimum distance. We will examine Hamming codes, random linear codes, and Reed-Solomon codes, ignoring algorithmic issues.

Be warned that while this is a mathematically appealing treatment of the problem, it is not the most practical treatment. In particular, there are codes with poor minimum distance properties that are very useful in practice.

## 11.2 Coding

Error-correcting codes are used to compensate for noise and interference in communication. They are used in practically all digital transmission and data storage schemes. In this class, we will only consider the problem of storing or transmitting bits[1], or maybe symbols from some small discrete alphabet.

The only type of interference we will consider is the flipping of bits. Thus, `0101` may become `1101`, but not `010`. More noise means more bits are flipped.

In our model problem, a transmitter wants to send $m$ bits, which means that the transmitter's message is an element of $\{0,1\}^m$. But, if the transmitter wants the receiver to correctly receive the message in the presence of noise, the transmitter should not send the plain message. Rather, the transmitter will send $n > m$ bits, encoded in such a way that the receiver can figure out what the message was even if there is a little bit of noise.

A naive way of doing this would be for the transmitter to send every bit 3 times. If only 1 bit were flipped during transmission, then the receiver would be able to figure out which one it was. But, this is a very inefficient coding scheme. Much better approaches exist.

[1]Everything is bits. You think that's air you're breathing?

## 11.3 Hamming Codes

The first idea in coding theory was the parity bit. It allows one to detect one error. Let's say that the transmitter wants to send $b_1, \dots, b_m$. If the transmitter constructs

$$b_{m+1} = \sum_{i=1}^{m} b_i \mod 2, \tag{11.1}$$

and sends

$$b_1, \dots, b_{m+1},$$

then the receiver will be able to detect one error, as it would cause (11.1) to be violated. But, the receiver won't know where the error is, and so won't be able to figure out the correct message unless it request a retransmit. And, of course, the receiver wouldn't be able to detect 2 errors.

Hamming codes combine parity bits in an interesting way to enable the receiver to correct one error. Let's consider the first interesting Hamming code, which transmits 4-bit messages by sending 7 bits in such a way that any one error can be corrected. Note that this is much better than repeating every bit 3 times, which would require 12 bits.

For reasons that will be clear soon, I will let $b_3, b_5, b_6$, and $b_7$ be the bits that the transmitter would like to send. The parity bits will be chosen by the rules

$$\begin{aligned} b_4 &= b_5 + b_6 + b_7 \\ b_2 &= b_3 + b_6 + b_7 \\ b_1 &= b_3 + b_5 + b_7. \end{aligned}$$

All additions, of course, are modulo 2. The transmitter will send the *codeword* $b_1, \dots, b_7$.

If we write the bits as a vector, then we see that they satisfy the linear equations

$$\begin{pmatrix} 0 & 0 & 0 & 1 & 1 & 1 & 1 \\ 0 & 1 & 1 & 0 & 0 & 1 & 1 \\ 1 & 0 & 1 & 0 & 1 & 0 & 1 \end{pmatrix} \begin{pmatrix} b_1 \\ b_2 \\ b_3 \\ b_4 \\ b_5 \\ b_6 \\ b_7 \end{pmatrix} = \begin{pmatrix} 0 \\ 0 \\ 0 \end{pmatrix}.$$

For example, to transmit the message `1010`, we set

$$b_3 = 1, b_5 = 0, b_6 = 1, b_7 = 0,$$

and then compute

$$b_1 = 1, b_2 = 0, b_4 = 1.$$

Let's see what happens if some bit is flipped. Let the received transmission be $c_1, \dots, c_7$, and assume that $c_i = b_i$ for all $i$ except that $c_6 = 0$. This means that the parity check equations that involved the 6th bit will now fail to be satisfied, or

$$\begin{pmatrix} 0 & 0 & 0 & 1 & 1 & 1 & 1 \\ 0 & 1 & 1 & 0 & 0 & 1 & 1 \\ 1 & 0 & 1 & 0 & 1 & 0 & 1 \end{pmatrix} \boldsymbol{c} = \begin{pmatrix} 1 \\ 1 \\ 0 \end{pmatrix}.$$

Note that this is exactly the pattern of entries in the 6th column of the matrix. This will happen in general. If just one bit is flipped, and we multiply the received transmission by the matrix, the product will be the column of the matrix containing the flipped bit. As each column is different, we can tell which bit it was. To make this even easier, the columns have been arranged to be the binary representations of their index. For example, `110` is the binary representation of 6.

## 11.4 The asymptotic case

In the early years of coding theory, there were many papers published that contained special constructions of codes such as the Hamming code. But, as the number of bits to be transmitted became larger and larger, it became more and more difficult to find such exceptional codes. Thus, an asymptotic approach became reasonable.

We will view an error-correcting code as a mapping

$$C : \{0,1\}^m \to \{0,1\}^n,$$

for $n$ larger than $m$. Every string in the image of $C$ is called a *codeword.* We will also abuse notation by identifying $C$ with the set of codewords.

We define the *rate* of the code to be

$$r = \frac{m}{n}.$$

The rate of a code tells you how many bits of information you receive for each codeword bit. Of course, codes of higher rate are more efficient.

The *Hamming distance* between two words $\boldsymbol{c}^1$ and $\boldsymbol{c}^2$ is the number of bits in which they differ. It will be written

$$\text{dist}(\boldsymbol{c}^1, \boldsymbol{c}^2).$$

The *minimum distance* of a code is

$$d = \min_{\boldsymbol{c}^1 \neq \boldsymbol{c}^2 \in C} \text{dist}(\boldsymbol{c}^1, \boldsymbol{c}^2)$$

(here we have used $C$ to denote the set of codewords). It should be clear that if a code has large minimum distance then it is possible to correct many errors. In particular, it is possible to correct any number of errors less than $d/2$. To see why, let $\boldsymbol{c}$ be a codeword, and let $\boldsymbol{r}$ be the result of flipping $e < d/2$ bits of $\boldsymbol{c}$. As $\text{dist}(\boldsymbol{c}, \boldsymbol{r}) < d/2$, $\boldsymbol{c}$ will be the closest codeword to $\boldsymbol{r}$. This is because for every $\boldsymbol{c}^1 \neq \boldsymbol{c}$,

$$d \leq \text{dist}(\boldsymbol{c}^1, \boldsymbol{c}) \leq \text{dist}(\boldsymbol{c}^1, \boldsymbol{r}) + \text{dist}(\boldsymbol{r}, \boldsymbol{c}) < \text{dist}(\boldsymbol{c}^1, \boldsymbol{r})n + d/2 \quad \text{implies} \quad d/2 < \text{dist}(\boldsymbol{c}^1, \boldsymbol{r}).$$

So, large minimum distance is good.

The *minimum relative distance* of a code is

$$\delta = \frac{d}{n}.$$

It turns out that it is possible to keep both the rate and minimum relative distance of a code bounded below by constants, even as $n$ grows. To formalize this notation, we will talk about a sequence of codes instead of a particular code. A sequence of codes $C_1, C_2, C_3, \ldots$ is presumed to be a sequence of codes of increasing message lengths. Such a sequence is called *asymptotically good* if there are absolute constants $r$ and $\delta$ such that for all $i$,

$$r(C_i) \geq r \qquad \text{and} \qquad \delta(C_i) \geq \delta.$$

One of the early goals of coding theory was to construct asymptotically good sequences of codes.

## 11.5 Random Codes

We will now see that random linear codes are asymptotically good with high probability. We can define a random linear code in one of two ways. The first is to choose a rectangular $\{0,1\}$ matrix $M$ uniformly at random, and then set

$$C = \{\boldsymbol{c} : M\boldsymbol{c} = \boldsymbol{0}\}.$$

Instead, we will choose an $m$-by-$n$ matrix with independent uniformly chosen $\{0,1\}$ entries, and then set

$$C(\boldsymbol{b}) = M\boldsymbol{b}.$$

Thus, the code will map $m$ bits to $n$ bits. Let's call this code $C_M$. Such a code is called a *linear code*. Clearly, the rate of the code will be $m/n$.

Understanding the minimum distance of a linear code is simplified by the observation that

$$\text{dist}(\boldsymbol{c}^1, \boldsymbol{c}^2) = \text{dist}(\boldsymbol{0}, \boldsymbol{c}^1 - \boldsymbol{c}^2) = \text{dist}(\boldsymbol{0}, \boldsymbol{c}^1 + \boldsymbol{c}^2),$$

where we of course do addition and subtraction component-wise and modulo 2. The linear structure of the code guarantees that if $\boldsymbol{c}^1$ and $\boldsymbol{c}^2$ are in $C_M$, then $\boldsymbol{c}^1 + \boldsymbol{c}^2$ is as well. So, the minimum distance of $C_M$ is

$$\min_{\boldsymbol{0} \neq \boldsymbol{b} \in \{0,1\}^m} \text{dist}(\boldsymbol{0}, M\boldsymbol{b}) = \min_{\boldsymbol{0} \neq \boldsymbol{b} \in \{0,1\}^m} |M\boldsymbol{b}|,$$

where by $|\boldsymbol{c}|$ we mean the number of `1`s in $\boldsymbol{c}$. This is sometimes called the *weight* of $\boldsymbol{c}$.

Here's what we can say about its minimum distance of a random linear code.

**Lemma 11.5.1.** *Let $M$ be a random $m$-by-$n$ matrix. For any $d$, the probability that $C_M$ has minimum distance at least $d$ is at least*

$$1 - \frac{2^m}{2^n} \sum_{i=0}^{d} \binom{n}{d}.$$

*Proof.* It suffices to upper bound the probability that there is some non-zero $\boldsymbol{b} \in \{0,1\}^m$ for which

$$|M\boldsymbol{b}| \leq d.$$

To this end, fix some non-zero vector $\boldsymbol{b}$ in $\{0,1\}^m$. Each entry of $M\boldsymbol{b}$ is the inner product of a column of $M$ with $\boldsymbol{b}$. As each column of $M$ consists of random $\{0,1\}$ entries, each entry of $M\boldsymbol{b}$ is chosen uniformly from $\{0,1\}$. As the columns of $M$ are chosen independently, we see that $M\boldsymbol{b}$ is a uniform random vector in $\{0,1\}^n$. Thus, the probability that $|M\boldsymbol{b}|$ is at most $d$ is precisely

$$\frac{1}{2^n}\sum_{i=0}^{d}\binom{n}{d}.$$

As the probability that one of a number of events holds is at most the sum of the probabilities that each holds (the "union bound"),

$$\begin{aligned}\Pr_M\left[\exists \boldsymbol{b} \in \{0,1\}^m, \boldsymbol{b} \neq \boldsymbol{0} : |M\boldsymbol{b}| \leq d\right] &\leq \sum_{\boldsymbol{0}\neq\boldsymbol{b}\in\{0,1\}^m} \Pr_M\left[|M\boldsymbol{b}| \leq d\right] \\ &\leq (2^m - 1)\frac{1}{2^n}\sum_{i=0}^{d}\binom{n}{d}. \\ &\leq \frac{2^m}{2^n}\sum_{i=0}^{d}\binom{n}{d}.\end{aligned}$$

□

To see how this breaks down asymptotically, recall that for a constant $p$,

$$\binom{n}{pn} \approx 2^{nH(p)},$$

where

$$H(p) \overset{\text{def}}{=} -p\log_2 p - (1-p)\log_2(1-p)$$

is the *binary entropy function.* If you are not familiar with this, I recommend that you derive it from Stirling's formula. For our purposes $2^{nH(p)} \approx \sum_{i=0}^{pn}\binom{n}{i}$. Actually, we will just use the fact that for $\beta > H(p)$,

$$\frac{\sum_{i=0}^{pn}\binom{n}{i}}{2^{n\beta}} \to 0$$

as $n$ goes to infinity.

If we set $m = rn$ and $d = \delta n$, then Lemma 11.5.1 tells us that $C_M$ probably has rate $r$ and minimum relative distance $\delta$ if

$$\frac{2^{rn}}{2^n}2^{nH(\delta)} < 1,$$

which happens when

$$H(\delta) < 1 - r.$$

For any constant $r < 1$, we can find a $\delta$ for which $H(\delta) < 1 - r$, so there exist asymptotically good sequences of codes of every non-zero rate. This is called the Gilbert-Varshamov bound. It is still not known if binary codes exist whose relative minimum distance satisfies $H(\delta) > 1 - r$. This is a big open question in coding theory.

Of course, this does not tell us how to choose such a code in practice, to efficiently check if a given code has large minimum distance, or how to efficiently decode such a code.

## 11.6 Reed-Solomon Codes

Reed-Solomon Codes are one of the workhorses of coding theory. The are simple to describe, and easy to encode and decode.

However, Reed-Solomon Codes are not binary codes. Rather, they are codes whose symbols are elements of a finite field. If you don't know what a finite field is, don't worry (yet). For now, we will just consider prime fields, $F_p$. These are the numbers modulo a prime $p$. Recall that such numbers may be added, multiplied, and *divided.*

A message in a Reed-Solomon code over a field $F_p$ is identified with a polynomial of degree $m - 1$. That is, the message $f_1, \ldots, f_m$ is viewed as providing the coefficients of the polynomial

$$Q(x) = \sum_{i=0}^{m-1} f_{i+1} x^i.$$

A Reed-Solomon code is encoded by evaluating it over every element of the field. That is, the codeword is

$$Q(0), Q(1), Q(2), \ldots, Q(p-1).$$

Sometimes, it is evaluated at a subset of the field elements.

I will now show you that the minimum distance of such a Reed-Solomon code is $p - m$. We show this using the following standard fact from algebra, which I will re-derive if there is time:

**Lemma 11.6.1.** *Let $Q$ be a polynomial of degree at most $m - 1$ over a field $F_p$. If there exists distinct field elements $x_1, \ldots, x_m$ such that*

$$Q(x_i) = 0$$

*for all $i$, then $Q$ is identically zero.*

**Theorem 11.6.2.** *The minimum distance of the Reed-Solomon code is at least $p - m$.*

*Proof.* Let $Q^1$ and $Q^2$ be two different polynomials of degree at most $m - 1$. For a polynomial $Q$, let

$$E(Q) = (Q(0), Q(1), \ldots, Q(p))$$

be its encoding. If

$$\mathrm{dist}(E(Q^1), E(Q^2)) \le p - k,$$

then there exists field elements $x_1, \ldots, x_k$ such that

$$Q^1(x_j) = Q^2(x_j).$$

Now, consider the polynomial

$$Q^1(x) - Q^2(x).$$

It also has degree at most $m-1$, and it is zero at $k$ field elements. Lemma 11.6.1 tells us that if $k \geq m$, then $Q^1 - Q^2$ is exactly zero, which means that $Q^1 = Q^2$. Thus, for distinct $Q^1$ and $Q^2$, it must be the case that

$$\text{dist}(E(Q^1), E(Q^2)) > p - m.$$

$\square$

However, Reed-Solomon codes do not provide an asymptotically good family. If one represents each field element by $\log_2 p$ bits in the obvious way, then the code has length $p \log_2 p$, but can only correct at most $p$ errors. That said, one can find an asymptotically good family by encoding each field element with its own small error-correcting code.

Next lecture, we will see how to make asymptotically good codes out of expander graphs. In the following lecture, we will use good error-correcting codes to construct graphs.