18.325 Applied Extremal Combinatorics	Fall Term, 1996-97

Lecture 8 : October 7, 1996

Lecturer: Daniel A. Spielman *Scribe: Rocco Servedio*

1 Introduction

This is the first lecture on error-correcting codes. We begin with some motivation for error-correcting codes and an example of a simple family of error-correcting codes. We then define our communication model, a binary symmetric channel, and prove upper and lower asymptotic bounds on the communication rate that can be achieved in this model. Finally, we define random linear codes and prove some bounds on the minumum relative distance between code words.

A few preliminary definitions are indispensible.

Definition 1 *The* (Hamming) distance $d(x_1, x_2)$ *between two bit vectors is the number of positions at which they disagree.*

Definition 2 *A* binary code *is an injection* C *from* $\{0,1\}^m$ *(the* message space*) to* $\{0,1\}^n$*. An element* $y \in \{0,1\}^n$ *is called a* codeword *if* $y = C(x)$ *for some message* x*.*

2 Hamming Codes

The field of error-correcting codes really began back in the early days of computing when programs and data were submitted to computers on punch cards. A single mistake on a card could cause the programmer to waste days finding and fixing the error. It became pretty clear that a coding system enabling the machine to detect (and ideally, fix) errors would be a good thing.

One simple fix is to use a *parity bit.* If the information on a card is a bit sequence $b_1, \ldots, b_k$, then the parity bit for that card would be $b_0 = b_1 + \cdots + b_k \bmod 2$. By checking the value of the parity bit against the sum of the other bits, the machine can tell if one bit has been flipped.

This is a start, but it leaves something to be desired; a human operator must still look through the card to find out which bit is in error. Hamming, working at Bell Labs in the 1940s, figured out a method whereby if a single bit were flipped, the machine itself would be able to identify the flipped bit (and hence fix it).

We illustrate Hamming's technique for the case $k = 3$. Code words are of length $2^k - 1$ and the message space is $\{0,1\}^{2^k-1-k}$; each code word consists of $2^k - 1 - k$ bits of information and k "check bits". For the case $k = 3$, the check bits are b_1, b_2 and b_4 and the information bits are b_3, b_5, b_6 and b_7. If we are given values for the information bits b_3, b_5, b_6 and b_7, then the values of the check bits are such that the following equations are satisfied:

$$b_1 + b_3 + b_5 + b_7 = 0 \quad (1)$$

$$b_2 + b_3 + b_6 + b_7 = 0 \quad (2)$$

$$b_4 + b_5 + b_6 + b_7 = 0. \quad (3)$$

In general there are k equations, which are determined by the binary representations of the numbers 1 to $2^k - 1$. The case $k = 3$ is shown below:

b_1	b_2	b_3	b_4	b_5	b_6	b_7
1	0	1	0	1	0	1
0	1	1	0	0	1	1
0	0	0	1	1	1	1

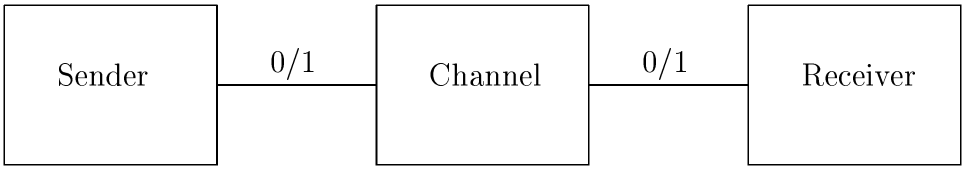
Now we have a very desirable state of affairs. Suppose that bit b_i is flipped. Let S_i be the set of bit positions whose values are 1 in the binary expression of i. Then equation j will be unsatisfied iff $j \in S_i$. For instance, consider what happens if b_6 is flipped in the $k = 3$ example above. Equations (2) and (3) become unsatisfied; this is as we expect, since the binary representation of 6 is 110. Conversely, by looking at which equations are unsatisfied, we can easily determine which bit position has been flipped, and can consequently reconstruct the original code word.

So we've seen that under this encoding system, each code word is of length $2^k - 1$. Since the message space is $\{0,1\}^{2^k-1-k}$, there are 2^{2^k-1-k} code words. For any code word W, there are 2^k bit strings of length $2^k - 1$ which will be mapped to W by the decoding described above (these are simply W itself and the $2^k - 1$ bit strings whose Hamming distance from W is one). So the number of bit strings of length $2^k - 1$ which get mapped to some code word is $2^{2^k-1-k} \cdot 2^k = 2^{2^k-1}$. This means that every bit string gets mapped to some code word, or equivalently, that the $(2^k - 1)$-dimensional hypercube can be completely partitioned into disjoint balls of radius 1 centered at the Hamming code words.

In general, if a hypercube can be partitioned into disjoint balls of radius t, then we say that the centers of the balls form a *perfect t-error-correcting code*, since every bit string will be mapped to some code word whose distance from it is at most t. Clearly, in addition to the perfect 1-error-correcting code described above, one can construct a perfect 0-error-correcting code and a perfect $(n/2)$-error-correcting-code for a hypercube of dimension n. Later in the semester, we'll prove that these are in fact the only infinite families of perfect t-error-correcting codes for any t.

3 Binary Symmetric Channels

Shannon provided a mathematically sound foundation for the theory of error-correcting codes in his important early paper [1].



Our model of communication is called a *binary symmetric channel*. Messages are transmitted as strings of bits. Each bit which the sender puts into the channel is flipped with probability p on its way to the receiver; this probability is independent for each bit.

This is one of the simplest of all reasonable models. Other models exist in which real numbers are transmitted instead of binary values, the channel has a memory (so probabilities are not independent), etc. However, codes which are designed for a binary symmetric channel seem to perform well in the real world, so this is the model we'll use.

A natural question is how much information can be transmitted through a binary symmetric channel for a given value of p. To answer this, we need the following definitions.

Definition 3 *Let X be a random variable which takes value s_i with probability p_i,*

$1 \le i \le n$. Then the entropy of X is defined to be

$$H(X) = -\sum_{i=1}^{n} p_i \log p_i.$$

Intuitively, the entropy of X measures the amount of information which you don't know (but would like to know) about X. Some important properties of $H(X)$ are listed below:

1. If X is an unbiased random bit, then $H(X) = 1$.
2. For two random variables X and Y, $H(X,Y) \le H(X) + H(Y)$, with equality holding iff X and Y are independent.
3. If X is a string of k independent unbiased random bits, then $H(X) = k$.
4. If the random variable X assumes n distinct values $s_1, \ldots, s_n$ with probabilities $p_1, \ldots, p_n$, then $H(X)$ is maximized if $p_i = 1/n$ for all i. '

Definition 4 *The* channel capacity *$C(p)$ of the binary symmetric channel with parameter p is $C(p) = 1 - H(p)$.*

In the definition above, $H(p)$ is the entropy of the random variable X which is 1 with probability p and 0 with probability $1-p$. It's worth noting at this point that by Stirling's formula, $\binom{n}{pn} = 2^{nH(p)+O(logn)}$.

Definition 5 *If $C : \{0,1\}^m \to \{0,1\}^n$ is a code, then the* communication rate *r of code C is m/n.*

Intuitively, this measures the "density" of information which is being communicated. In the Hamming code example from earlier with $k = 3$, the communication rate is $4/7$.

Definition 6 *Given a probability distribution D over the message space $\{0,1\}^m$ and a decoding algorithm A which maps $\{0,1\}^n$ into the set of all codewords $C(\{0,1\}^m)$, the* error rate *of code C is the probability under D that a message x gets decoded into something other than its correct codeword $C(x)$.*

With these definitions, we can state and prove the following theorem, due to Shannon, which gives a relationship between communication rates and channel capacity.

Theorem 1

1. *If $r > C(p)$, then there is no binary code which has communication rate r.*
2. *For any $r < C(p)$ and for any $\epsilon > 0$, for sufficiently large n there is a binary code with codewords of length n which has communication rate $\geq r$ and error rate $\leq \epsilon$.*

Proof:

We give an informal proof of (1) first. One way to view a binary symmetric channel with parameter p is that it adds $H(p)n$ bits of information to an n-bit message. If $r > C(p)$, then the number of bits of information which the receiver gets from the sender upon decoding is $m = rn > (1 - H(p))n$. However, after decoding the receiver knows both the message that was sent and the errors that were added by the channel. Since the errors added by the channel are independent of the message that was sent, the total amount of information which the receiver gets is $> (1 - H(p))n + H(p)n$ – but this is impossible, since the receiver only received a bit string of length n.

Now for (2). We assume a uniform probability distribution on messages $x \in M$, and we assume that an "ideal decoding algorithm" (IDA) is being used by the receiver which does the following: given a message from the channel, the algorithm maps it to the nearest (Hamming distance) code word. Let $\mathcal{C}$ be the set of all codes which have communication rate r, and let $ch(y)$ denote the output of the channel on input y (note that $ch(\cdot)$ is a probabilistic function). Then the overall probability that a random code word is decoded correctly by a random code with communication rate r is given by

$$E_{C \in \mathcal{C}}[\Pr_{x \in M}[IDA(ch(C(x))) = C(x)].$$

Using Fubini's theorem (i.e. reversing the order of summation), this equals

$$E_{x \in M}[\Pr_{C \in \mathcal{C}}[IDA(ch(C(x))) = C(x)].$$

Since we are assuming a uniform distribution on messages, this equals

$$\Pr_{C \in \mathcal{C}}[IDA(ch(C(x))) = C(x)]$$

for any message x.

Given a code C, we have that $IDA(ch(C(x))) = C(x)$ if and only if $C(x)$ is closer to $ch(C(x))$ than any other $C(y)$ is. Choose p_0 such that $r < C(p_0) < C(p)$. Since $p_0 > p$, it is clear that as n grows large $\Pr[d(ch(C(x)), C(x)) \geq p_0 n]$ grows arbitrarily small (this is easy to show using Chernoff bounds or Chebychev's theorem). So to show that $IDA(ch(C(x))) = C(x)$ with high probability, we need only show that with high probability there exists no other code word $C(y)$ whose distance from $ch(C(x))$ is $p_0 n$ or less.

$B_{p_0 n}(ch(C(x)))$, the ball of radius $p_0 n$ around $ch(C(x))$, contains $\binom{n}{0} + \binom{n}{1} + \cdots + \binom{n}{p_0 n}$ elements. This sum is dominated by the last term, so for simplicity's sake we say that the ball contains $\binom{n}{p_0 n}$ elements. Since there are $2^m = 2^{rn}$ codewords, we obtain

$$\Pr_{C \in \mathcal{C}}[C(\{0,1\}^m) \cap B_{p_0 n}(ch(C(x))) = \emptyset] = \left(1 - \frac{\binom{n}{p_0 n}}{2^n}\right) \cdots \left(1 - \frac{\binom{n}{p_0 n}}{2^n - 2^{rn} + 1}\right).$$

Since $r < C(p) < 1$, for large n we have that the above expression can be approximated by

$$\left(1 - \frac{\binom{n}{p_0 n}}{2^n}\right)^{2^{rn}}$$

Since $r < 1 - H(p_0)$, we have that $2^{rn} < 2^{n - nH(p_0)}$ and consequently for large n, $2^{rn} < \frac{2^n}{\binom{n}{p_0 n}}$. This in turn implies that the quantity above approaches 1 for large n.

So we've shown that

$$E_{C \in \mathcal{C}}[\Pr_{x \in M}[IDA(ch(C(x))) = C(x)]$$

approaches 1 as n grows large. Since a random variable must achieve its expected value at least once, it follows that there must be some binary code with communication rate r which has an arbitrarily low error rate. ■

We note in closing this section that since we've been assuming a uniform distribution over messages and a perfect decoding algorithm, the quantity which mattered most to us has been the minimum distance between code words. If this minimum distance is d, then we can correct up to $(d-1)/2$ errors, so a "good code" for us has been a code which maximizes d. However, this isn't necessarily how things work in the real world. There actually exist codes called *turbocodes* which have a few pairs of codewords which are very close (and hence a small value for d) but which perform extremely well in practice. It's not known how to analyze their behavior, though.

4 Random Linear Codes

As we've seen, we're interested in codes for which the minimum distance between two code words is as large as possible. In this section, we'll see that a random linear code does pretty well by this measure.

Definition 7 *A* linear code *is a code* $C : \{0,1\}^{rn} \to \{0,1\}^n$ *whose codewords* $C(\{0,1\}^{rn})$ *form a vector space over* $\{0,1\}$. *A* generator matrix *for a linear code* C *is a* rn *by* n *matrix whose rows are a basis for the set of codewords.*

So if C is a linear code, then the set of codewords of C is exactly the set of all linear combinations of the rows of some generator matrix.

Definition 8 *The* weight *of a codeword is the number of 1's which appear in it. The* minimum relative distance δ *of a code is*

$$\frac{\min_{x \neq y \in C(\{0,1\}^{rn})} d(x,y)}{n}.$$

For a linear code, $d(C(x), C(y)) = d(C(x-y), 0)$, so the minimum weight across all codewords is equal to δn.

A *random linear code* is a linear code which is obtained from a random generator matrix. It turns out that a random linear code will, with high probability, have a high minimum relative distance; we see this in the proof of the following theorem.

Theorem 2 *(Gilbert-Varshamov Bound) If* $r < C(\delta)$*, then there exists a binary linear code with communication rate* r *and minimum relative distance* δ*.*

Proof: We consider a random linear code. Codewords correspond to subsets of rows; since there are rn rows, it follows that there are at most 2^{rn} codewords (fewer if the rows are not linearly independent). So a random element of $\{0,1\}^n$ is a codeword with probability at most $2^{-(1-r)n}$. Since the number of codewords of weight $\leq \delta n$ is bounded by $2^{nH(\delta)}$, it follows that

$$\Pr[\text{some codeword has weight } \leq \delta n] \leq 2^{-(1-r)n} \cdot 2^{nH(\delta)}.$$

If $1-r > H(\delta)$, then the right side of the above bound is less than 1, and consequently there must exist some linear code in which every code word has weight at least δn. ∎

Definition 9 *A family of codes $C_1, \ldots, C_i$, where code C_i has communication rate r_i and relative minimum distance δ_i, is called* asymptotically good *if*

$$\lim_{i\to\infty} r_i > 0 \text{ and } \lim_{i\to\infty} \delta_i > 0.$$

We've just seen that a family of random linear codes is asymptotically good. Interestingly, explicit constructions for asymptotically good families of linear codes were unknown until the early 1970's. If we consider the Hamming code parameterized by k, we see that $r_k = \frac{2^k-1-k}{2^k-1}$ approaches 1 and $\delta_k = \frac{1}{2^k-1}$ approaches 0.

We now turn to some upper bounds on r as a function of δ. The simplest of these, the Hamming bound, is based simply on volume considerations. If δ is the minimum relative distance between codewords, then we can put disjoint balls of radius $\delta/2$ around each of the 2^{rn} codewords. Since each of these balls contains (approximately) $2^{nH(\delta/2)}$ points, we obtain the following inequality:

$$2^{rn} \cdot 2^{nH(\delta/2)} \leq 2^n.$$

which implies that $r \leq 1 - H(\delta/2)$.

This bound is in fact quite weak. For $\delta = \frac{1}{2}$, the Hamming bound only tells us that $r < 1 - H(1/4) \approx 0.188$, leaving open the possibility that a set of $2^{0.188n}$ codewords could have minimum relative distance $\frac{1}{2}$. However, it is in fact impossible for a set to contain more than n codewords and still have minimum relative distance $\frac{1}{2}$.

The Plotkin bound is somewhat stronger. The proof proceeds by counting $\sum_{i<j} d(w_i, w_j)$, the sum of all pairwise distances between code words, in two different ways. Since the minimum distance between codewords is δn, we have that

$$\sum_{i<j} d(w_i, w_j) \geq \binom{2^{rn}}{2} \delta n.$$

Now consider a 2^{rn} by n matrix whose rows are the codewords. Let a_i be the fraction of rows which have a 0 in the i-th position (i.e. in column i). A contribution to the total pairwise distance is made whenever two words differ in position i; consequently, the bits in the i-th bit position contribute $a_i(1-a_i)2^{2rn}$ to the total. Summing over columns, we find that

$$\sum_{i<j} d(w_i, w_j) = \sum_{i=1}^{n} a_i(1-a_i)2^{2rn}.$$

The right hand side of the above equation is maximized when $a_i = \frac{1}{2}$. Combining these relations, we have that

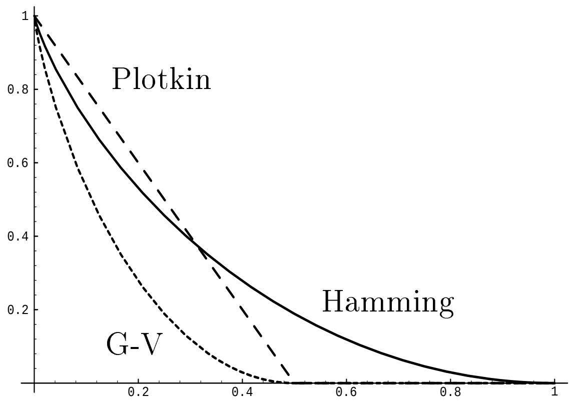
$$\frac{n2^{2rn}}{4} \geq \binom{2^{rn}}{2}\delta n = \frac{2^{rn} \cdot (2^{rn} - 1)\delta n}{2}.$$

Simplifying, we find that $\delta \geq 2^{rn}(\delta - \frac{1}{2})$. Now if $\delta > 1/2$, we can divide through by $\delta - \frac{1}{2}$ and obtain

$$\frac{\delta}{\delta - \frac{1}{2}} > 2^{rn}.$$

So if δ is a constant greater than $\frac{1}{2}$, then 2^{rn} is bounded above by a constant, which is as it should be. Furthermore, we have that r approaches 0 as n grows large for fixed $\delta > \frac{1}{2}$.

It turns out that the Plotkin bound at $\delta = \frac{1}{2}$ can be extended linearly back to $\delta = 0$. The following diagram shows all three bounds.



5 References

[1] C. E. Shannon, A mathematical theory of communication, *Bell Syst. Tech. J.*, **27** (1948), pp. 379-423.