Linear Error-Correcting Codes

Lecturer: Michel Goemans

1 Linear Error Correcting Codes

We wish to transmit a message (represented in bits) to a destination through a channel reliably. Errors may occur during transmission and we hope to detect and correct these transmitted errors. In order to do so, we must introduce some redundancy in our transmitted message.

We saw in the last lecture that Shannon's Noisy Coding Theorem tells us that if the redundancy is above a certain level (constrained by the noise of the channel), then we can drive the probability of error (detecting the wrong message sent) to zero with increasing blocklength. The proof was based on a random code which is not very practical. For easy encoding and decoding, we need structure in our encoding/decoding functions. We will study a class of codes called *linear block codes*, because their structure offers several advantages. Linearity will allow an easier analysis of the error correcting ability of a code. Furthermore, the use of matrices to encode/decode messages is much easier than a random codebook, and the code can be concisely described.

Error detection means that the code can detect errors but we don't know where the errors are in a received sequence. *Error correction* means we know where the errors are and can correct the bit positions in error. If a code can correct up to t errors, then if the sequence contains more than t errors, the decoder may decode to the wrong information sequence.

All our messages, codewords, and received messages will be sequences with binary coefficients from the binary field. Let e_i be the vector with 1 in the i-th position and zero in all the other positions, so $e_3 = (00100\ldots00)$. For linear codes, encoding is a linear transformation c that maps a message m to a codeword $c(m)$. Block codes means all codewords $c(m)$ have the same length, which we denote as n. Let k be the length of information bits that we encode. Then there are $n-k$ redundancy bits in each codeword, called *parity check bits*, which are left for error correction. The (n, k) specify the parameters (codeword length and parity check bit length) for a block code. Let $\tilde{c}$ be the received vector or string of n bits.

The *Hamming distance* d_H between any two bit strings is the number of positions in which the 2 strings differ. For example, the Hamming distance d_H between the codewords $c_1 = (101101)$ and $c_2 = (100110)$ is 3. Denote d^* to be the minimum Hamming distance between any two distinct codewords of a code C as

$$d^* = d_{\min} = \min_{c_i \neq c_j} d_H(c_i, c_j). \tag{1}$$

A code with minimum distance d^* between codewords can detect $d^* - 1$ errors and can correct $\frac{d^*-1}{2}$ errors with nearest neighbor decoding. Thus, to correct one error, the minimum distance of the code must be at least 3.

A *linear* code means that the encoding function c is linear (over $\mathbb{Z}_2$). If messages m and m' have respective codewords $c(m)$ and $c(m')$, then $c(m + m') = c(m) + c(m')$ is the codeword for the

message $(m + m')$ (where addition is modulo 2). Thus, the all-zero sequence must be a codeword and must correspond to the message consisting of k 0's.

The (Hamming) *weight* $w(s)$ of a binary string s is defined as the sum of its non-zero entries s. A linear code is completely defined by all the codewords for messages of weight 1. For example, to encode 01101, we simply add the codewords for e_2, e_3, and e_5. Thus,

$$c(01101) = c(01000) + c(00100) + c(00001) = c(e_2) + c(e_3) + c(e_5).$$

The e_i's, $1 \leq i \leq k$, form the basis for the message space, and the codewords for the e_i completely describe the linear code. Thus, there is no need for a codebook with 2^k entries. All we need is a matrix G that gives the linear transformation for all weight one messages into codewords; G is a $k \times n$ matrix and row i is the codeword coresponding to e_i. This matrix G is called the *generating matrix*. Given G, a message m then gets encoded as $c = mG$.

We can assume that the generating matrix G takes a particular form. Indeed, if we perform elementary row operations on G (over Z_2, this means that we add a row to another row), we do not change the set of codewords in our code. We can even permute columns without fundamentally changing the properties of our code. Therefore, using Gaussian elimination, we can assume that G takes the simpler form:

$$G = [I_{k\times k}\ S]\,, \tag{2}$$

where S is $k \times (n - k)$.

For a linear code, the definition of the minimum distance d^* can be simplified. Indeed, for any two codewords c_i and c_j, we have that $d_H(c_i, c_j) = d_H(c_i - c_j, 0)$ and $c_i - c_j$ is a codeword. Thus, for a linear code, we have

$$d^* = \min_{c_i \in C, c_i \neq 0} w(c_i),$$

where as defined above, $w(c)$ denotes the Hamming weight.

Suppose our matrix G (in the form (2)) generates a single error correcting code. Then there are no codewords of weight 1 or 2, and the parity check portion S of G must satify the following 2 conditions:

- The weight of each row in S must be at least 2, since each row of the I part has weight exactly one.
- No two rows of S are identical. Otherwise, summing those two rows will give a codeword of weight 2.

These two conditions ensure that the minimum weight of any codeword is at least 3 (since summing 3 or more rows will give a codeword of weight at least 3 in the identity part). Thus, they are sufficient for defining a single error correcting code. It is also easy to find and correct that single error. In the next section we show that a Hamming code is one such 1-error correcting code.

2 Hamming Code

The Hamming code is a single error correction linear block code with $(n, k) = (2^m - 1, 2^m - 1 - m)$, where $m = n - k$ is the number of check bits in the codeword. The simplest non-trivial code is for $m = 3$, which is the $(7, 4)$ Hamming code.

The generator matrix of the Hamming code has dimension $k \times n$ and is of the form

$$G = [I_{k\times k}\ S]\,.$$

For this Hamming code, S has one row for each possible m-bit string with weight at least 2. There are exactly $k = 2^m - m - 1$ such strings. Any Hamming code is a 1-error correcting code as the two conditions above are satisfied.

Example. The $(7, 4)$ Hamming code is given by the generating matrix:

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

We know how to encode with any linear code. The codeword generated by message m is simply $c = mG$. But how about decoding? If no errors were introduced, the first k bits constitute the message. But errors might have been introduced. When there is only one error, decoding is easy as we show now.

For every generator matrix $G = [I\ S]$, there is a parity-check matrix H defined by

$$H = \begin{bmatrix} S \\ I \end{bmatrix}.$$

Since S has size $k \times (n - k)$, we have that the identity matrix is of size $(n - k) \times (n - k)$, and H has size $n \times (n - k)$. The important observation is that $GH = S + S = 0$, and therefore $c = mG$ satisfies that $cH = mGH = 0$. This means that if there an error and we have received $\tilde{c} = c + e_i$, we can find the error e_i by computing the *syndrome* $\tilde{c}H$ as this is supposed to be $cH + e_iH = e_iH$. Thus, $\tilde{c}H$ must be one of the rows of H and the index of the row indicates which bit has been corrupted. Once we know which bit was corrupted, we can recover the message.

Example: A $(7, 4)$ Hamming Code with the following generator matrix (the last 3 columns of each row must be at least 2 and no two are identical)

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

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

Suppose we have the received string $\tilde{c} = (1110111)$. Then the syndrome is

$$\tilde{c}H = (111)$$

The string (111) corresponds to the 4th row of H, so we know that an error occurred in the 4th bit position. The correct received string should be (1111111) and the information bits sent are (1111), which are the first four bits in the received sequence. If $\tilde{c}H = (000)$, then no errors occurred and the information bits would be (1110).

3 Perfect Codes

Given that a code (linear or non-linear) has length n codewords, an error can occur in any of the n positions or no error has occurred. The number of check bits must be able to tell us this. Thus, for an (n, k) code, $2^{n-k} \geq n+1$. If this bound is achieved with equality, $n = 2^{n-k}-1$, then the code is a *perfect code*. For a single error correcting code with $n-k$ check bits, the number of information bits encoded cannot exceed $[2^{n-k}-1-(n-k)]$. For codes of which $(n, k) = (2^{n-k}-1, 2^{n-k}-1-(n-k))$, note that every one of the 2^n possible received sequences is within 1 bit of some codeword. These types of codes are called *perfect single error correcting codes*. Every binary Hamming code is a perfect code.

A code (non necessarily linear) of length n can correct t errors if every string of length n is within Hamming distance of t to at most one codeword. The minimum distance between codewords is $2t+1$. This t-error correcting code is perfect if every string of length n is within Hamming distance t of exactly one codeword. In other words, there is no space left in the codeword space after it is partitioned into the decoding spheres for each each codeword. Perfect codes which correct more than one error are very difficult to find.

18.310 Principles of Discrete Applied Mathematics
Fall 2013