Lecture 16:

Equivalent Codes & Systematic Forms

November 9, 2005

Reminder: Dealing with Long Blocks

- Recall that Shannon's second theorem tells us that for any noisy channel, there is some code which allows us to achieve error free transmission at a rate up to the capacity.
- However, this might require us to encode our message in very long blocks. Why?
- Intuitively it is because we need to add just the right fraction of redundancy; too little and we won't be able to correct the erorrs, too much and we won't achieve the full channel capacity.
- For many real world situations, the block sizes used are thousands of bits, e.g. $K = 1024$ or $K = 4096$.

Potential Problems with Large Blocks

- Using very large blocks could potentially cause some serious practical problems with storage/retrieval of codewords.
- In particular, if we are encoding blocks of K bits, our code will have 2^K codewords. For $K \approx 1000$ this is a huge number!
- How could we even store all the codewords?
- How could we retrieve (look up) the N bit codeword corresponding to a given K bit message?
- How could we check if a given block of N bits is a valid codeword or a forbidden encoding?
- Last class, we saw how to solve all these problems by representing the codes *mathematically* and using the magic of *linear algebra*.
- The valid codewords formed a subspace in the vector space of the finite field Z_2^N.

Generator and Parity Check Matrices

- Using aritmetic mod 2, we could represent a code using either a set of basis functions or a set of constraint (check) equations, represented a generator matrix G or a parity check matrix H.
- Every linear combination basis vectors is a valid codeword & all valid codewords are spanned by the basis; similarly all valid codewords satisfy every check equation & any bitstring which satisfies all equations is a valid codeword.
- The rows of the generator matrix form a basis for the subspace of valid codes; we could encode a source message $\mathbf{s}$ into its transmission $\mathbf{t}$ by simple matrix multiplication: $\mathbf{t} = \mathbf{s}G$.
- The rows of the parity check matrix H form a basis for the complement of the code subspace and represent check equations that must be satisfied by every valid codeword. That is, all codewords $\mathbf{v}$ are orthogonal to all rows of H (lie in the null space of H), meaning that $\mathbf{v}H^T = \vec{0}$.

Repetition Codes and Single Parity-Check Codes

- An $[N,1]$ repetition code has the following generator matrix:

$$\begin{bmatrix} 1 & 1 & 1 & 1 \end{bmatrix} \quad \text{for N=4}$$

Here is a parity-check matrix for this code:

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

- One generator matrix for the $[N, N-1]$ single parity-check code is the following:

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

Here is the parity-check matrix for this code:

$$\begin{bmatrix} 1 & 1 & 1 & 1 \end{bmatrix}$$

Manipulating the Parity-Check Matrix

- There are usually many parity-check matrices for a given code. We can get one such matrix from another using the following "elementary row operations", which don't alter the solutions to the equations the parity-check matrix represents.
  - Swapping two rows.
  - Adding a row to a different row.

Ex: This parity-check matrix for the $[5,2]$ code:

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

can be transformed into this alternative:

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

Manipulating the Generator Matrix

- We can apply the same elementary row operations to a generator matrix for a code, in order to produce another generator matrix, since these operations just convert one set of basis vectors to another.

Example: Here is a generator matrix for the $[5,2]$ code we have been looking at:

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

Here is another generator matrix, found by adding the first row to the second:

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

Note: These manipulations leave the set of codewords unchanged, but they *don't* leave the way we encode messages by computing $\mathbf{t} = \mathbf{s}G$ unchanged!

Equivalent Codes

- Two codes are said to be *equivalent* if the codewords of one are just the codewords of the other with the order of symbols permuted.
- Permuting the order of the columns of a generator matrix will produce a generator matrix for an equivalent code, and similarly for a parity-check matrix.
- **Example:** Here is a generator matrix for the $[5,2]$ code we have been looking at:

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

- We can get an equivalent code using the following generator matrix obtained by moving the last column to the middle:

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

Generator & Parity Matrices In Systematic Form

- Using elementary row operations and column permutations, we can convert any generator matrix to a generator matrix for an equivalent code that is is *systematic form*, in which the left end of the matrix is the identity matrix.
- Similarly, we can convert to the systematic form for a parity-check matrix, which has an identity matrix in the right end.
- For the $[5, 2]$ code, only permutations are needed. The generator matrix can be permuted by swapping columns 1 and 3:

$$\begin{bmatrix} 0\ 0\ 1\ 1\ 1 \\ 1\ 1\ 0\ 0\ 1 \end{bmatrix} \Rightarrow \begin{bmatrix} 1\ 0\ 0\ 1\ 1 \\ 0\ 1\ 1\ 0\ 1 \end{bmatrix}$$

- When we use a systematic generator matrix to encode a block $\mathbf{s}$ as $\mathbf{t} = \mathbf{s}G$, the first K bits will be the same as those in $\mathbf{s}$. The remaining $N - K$ bits can be seen as "check bits".

Relationship b/w Generator & Parity Matrices

- If G and H are generator and parity-check matrices for $\mathcal{C}$, then for every $\mathbf{s}$, we must have $(\mathbf{s}G)H^T = \vec{0}$ — since we should only generate valid codewords. It follows that

$$GH^T = \vec{0}$$

- Furthermore, any H with $N - K$ independent rows that satisfies this is a valid parity-check matrix for $\mathcal{C}$.
- Suppose G is in systematic form, so for some P,

$$G \ = \ [I_K \mid P]$$

- Then we can find a parity-check matrix for $\mathcal{C}$ in systematic form as follows:

$$H \ = \ [-P^T \mid I_{N-K}]$$

  since $GH^T = -I_K P + P I_{N-K} = \vec{0}$.
  (Note that in Z_2, $-P^T = P^T$.)

More on Hamming Distance

- Recall that the Hamming distance, $d(\mathbf{u}, \mathbf{v})$, of two codewords $\mathbf{u}$ and $\mathbf{v}$ is the number of positions where $\mathbf{u}$ and $\mathbf{v}$ have different symbols.
- This is a proper distance, which satisfies the *triangle inequality*:

$$d(\mathbf{u}, \mathbf{w}) \ \leq \ d(\mathbf{u}, \mathbf{v}) + d(\mathbf{v}, \mathbf{w})$$

- Here's a picture showing why:

```
u : 0 1 1 0 0 1 1 0 1 1 1 0
                - - - - - -
v : 0 1 1 0 0 1 0 1 0 0 0 1
          - - -         - -
w : 0 1 1 1 1 0 0 1 0 0 1 0
```

  Here, $d(\mathbf{u}, \mathbf{v}) = 6$, $d(\mathbf{u}, \mathbf{v} = 5)$, and $d(\mathbf{u}, \mathbf{w}) = 7$.

Minimum Distance and Decoding

- A code's *minimum distance* is the minimum of $d(\mathbf{u}, \mathbf{v})$ over all distinct codewords $\mathbf{u}$ and $\mathbf{v}$.
- If the minimum distance is at least $2t+1$, a nearest neighbor decoder will always decode correctly when there are t or fewer errors.
- Here's why: Suppose the code has distance $d \geq 2t + 1$.
  If $\mathbf{u}$ is sent and $\mathbf{v}$ is received, having no more than t errors, then
  $-d(\mathbf{u}, \mathbf{v}) \leq t$.
  $-d(\mathbf{u}, \mathbf{u}') \geq d$ for any codeword $\mathbf{u}' \neq \mathbf{u}$.
  From the triangle inequality:

$$d(\mathbf{u}, \mathbf{u}') \leq d(\mathbf{u}, \mathbf{v}) + d(\mathbf{v}, \mathbf{u}')$$

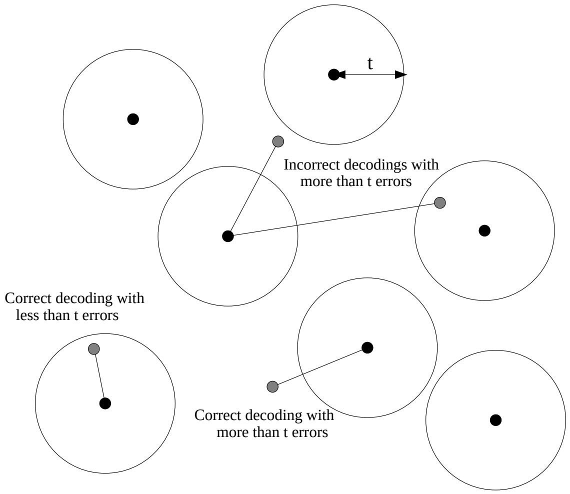
- It follows that

$$d(\mathbf{v}, \mathbf{u}') \geq d(\mathbf{u}, \mathbf{u}') - d(\mathbf{u}, \mathbf{v}) \geq d - t \geq (2t+1) - t \geq t + 1$$

  The decoder will therefore decode correctly to $\mathbf{u}$, at distance t, rather than to some other $\mathbf{u}'$.

A Picture of Distance and Decoding 12

- Here's a picture of codewords (black dots) for a code with minimum distance $2t+1$, showing how some transmissions are decoded:



Minimum Distance for Linear Codes 13

- To find the minimum distance for a code with 2^K codewords, we will in general have to look at all $2^K(2^K-1)/2$ pairs of codewords.
- But there's a short-cut for linear codes...
- Suppose two distinct codewords $\mathbf{u}$ and $\mathbf{v}$ are a distance d apart. Then the codeword $\mathbf{u}-\mathbf{v}$ will have d non-zero elements. The number of non-zero elements in a codeword is called its *weight*.
- Conversely, if a non-zero codeword $\mathbf{u}$ has weight d, then the minimum distance for the code is at least d, since $\vec{0}$ is a codeword, and $d(\mathbf{u},\vec{0})$ is equal to the weight of $\mathbf{u}$.
- So the minimum distance of a linear code is equal to the minimum weight of the 2^K-1 non-zero codewords. (This is useful for small codes, but when K is large, finding the minimum distance is difficult in general.)

Examples of Minimum Distance and Error Correction for Linear Codes 14

- Recall the $[5,2]$ code with the following codewords:

$$00000 \;\; 00111 \;\; 11001 \;\; 11110$$

- The three non-zero codewords have weights of 3, 3, and 4. This code therefore has minimum distance 3, and thus can correct any single error since ($2t+1=3$ for $t=1$).
- The single-parity-check code with $N=4$ has these codewords:

$$0000 \;\; 0011 \;\; 0101 \;\; 01101001 \;\; 1010 \;\; 1100 \;\; 1111$$

- The smallest weight of a non-zero codeword above is 2, so this is the minimum distance of this code. This is too small to guarantee correction of even one error. (Though the presence of a single error can be *detected*.)

Finding Minimum Distance From a Check Matrix 15

- We can find the minimum distance of a linear code from a parity-check matrix for it, H.
- The minimum distance is equal to the smallest number of linearly-dependent columns of H.
- Why? A vector $\mathbf{u}$ is a codeword iff $\mathbf{u}H^T=\vec{0}$. If d columns of H are linearly dependent, let $\mathbf{u}$ have 1s in those positions, and 0s elsewhere. This $\mathbf{u}$ is a codeword of weight d. And if there were any codeword of weight less than d, the 1s in that codeword would identify a set of less than d linearly-dependent columns of H.
- Special cases:
  - If H has a column of all zeros, then $d=1$.
  - If H has two identical columns, then $d \leq 2$.
  - For binary codes, if all columns are distinct and non-zero, then $d \geq 3$.