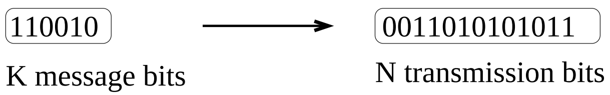
Lecture 15:

Linear Codes

November 1, 2006

Codes for Blocks of Symbols   1

- In error-correcting coding, we transmit a block of K message symbols by *encoding* it as a block of N transmission symbols.
- A *code* $\mathcal{C} \subseteq \mathcal{A}_X^N$ is a subset of all possible blocks of length N.
- The elements of $\mathcal{C}$ are called the *codewords*. These are the only blocks that we ever transmit.
- Normally we design the code to have the same number of codewords as possible message blocks (2^K for binary messages) and define a mapping between message blocks and codewords.
- The *rate* of $\mathcal{C}$ is $R = \log_2 |\mathcal{C}|/N$, which is K/N for binary channels.



Shannon's Noisy Coding Theorem   2

- Shannon's noisy coding theorem states that:

  For any channel with capacity C, any desired error probability, $\epsilon > 0$, and any transmission rate, $R < C$, there exists a code with some length N having rate at least R such that the probability of error when decoding this code by maximum likelihood is less than ϵ.
- In other words: We can transmit at a rate arbitrarily close to the channel capacity with arbitrarily small probability of error.
- The converse is also true: We *cannot* transmit with arbitrarily small error probability at a rate greater than the channel capacity. (see BSC example at the end of last class)
- We could always chose to transmit beyond the capacity, but not with vanishly small error – our best possible error rate would still be finite.

Long Blocks Might be Needed   3

- Shannon's second theorem (above) 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?
- Today, we'll see how to solve all these problems by representing the codes *mathematically* and using the magic of *linear algebra*.

The Finite Field Z_2

- From now on, we will consider only at binary channels, whose input and output alphabets are both $\{0, 1\}$.
- We will look at the symbols 0 and 1 as elements of Z_2, the integers considered modulo 2.
- Z_2 (also called F_2 or $GF(2)$) is the smallest example of a "field" — a collection of "numbers" that behave like real and complex numbers. Specifically, in a field:
  - Addition and multiplication are defined. They are commutative and associative. Multiplication is distributive over addition.
  - There are numbers called 0 and 1, such that $z + 0 = z$ and $z \cdot 1 = z$ for all z.
  - Subtraction and division (except by 0) can be done, and these operations are the inverses of addition and multiplication.

Arithmetic in Z_2

- Addition and multiplication in Z_2 are defined as follows:

$$\begin{array}{ll} 0+0=0 & 0\cdot 0=0 \\ 0+1=1 & 0\cdot 1=0 \\ 1+0=1 & 1\cdot 0=0 \\ 1+1=0 & 1\cdot 1=1 \end{array}$$

- This can also be seen as arithmetic modulo 2, in which we always take the remainder of the result after dividing by 2.
- Viewed as logical operations, addition is the same as 'exclusive-or', and multiplication is the same as 'and'.

  Note: In Z_2, $-a = a$, and hence $a - b = a + b$.

Vector Spaces Over Z_2

- Just as we can define vectors over the reals, we can define vectors over any other field, including over Z_2. We get to add such vectors, and multiply them by a scalar from the field.
- We can think of these vectors as N-tuples of field elements. For instance, with vectors of length five over Z_2:

$$\begin{aligned} (1,0,0,1,1) + (0,1,0,0,1) &= (1,1,0,1,0) \\ 1 \cdot (1,0,0,1,1) &= (1,0,0,1,1) \\ 0 \cdot (1,0,0,1,1) &= (0,0,0,0,0) \end{aligned}$$

- Most properties of real vector spaces hold for vectors over Z_2 — eg, the existence of basis vectors.
- We refer to the vector space of all N-tuples from Z_2 as Z_2^N; these are all bitstrings of length N. We will use boldface letters such as $\mathbf{u}$ and $\mathbf{v}$ to refer to such vectors.

Linear Codes — 8

- We can view Z_2^N as the input and output alphabet of the Nth extension of a binary channel.
- A code, $\mathcal{C}$, for this extension of the channel is a subset of Z_2^N.
- $\mathcal{C}$ is a *linear code* if the following condition holds:

  If $\mathbf{u}$ and $\mathbf{v}$ are codewords of $\mathcal{C}$, then $\mathbf{u}+\mathbf{v}$ is also a codeword.

  In other words, $\mathcal{C}$ must be a subspace of Z_2^N.
- Notice that since $\mathbf{u}+\mathbf{u}=\vec{0}$, the all-zero codeword must be in $\mathcal{C}$.

Note: For non-binary codes, we need a second condition, namely that if $\mathbf{u}$ is a codeword of $\mathcal{C}$ and z is in the field, then $z\mathbf{u}$ is also a codeword.

Linear Codes From Basis Vectors — 9

- We can construct a linear code by choosing K linearly-independent *basis vectors* from Z_2^N.
- We'll call the basis vectors $\mathbf{u}_1, \ldots, \mathbf{u}_K$. We define the set of codewords to be all those vectors that can be written in the form

  $$a_1\mathbf{u}_1 + a_2\mathbf{u}_2 + \cdots + a_K\mathbf{u}_K$$

  where $a_1, \ldots, a_K$ are elements of Z_2.
- The codewords obtained with different $a_1, \ldots, a_K$ are all different. (Otherwise $\mathbf{u}_1, \ldots, \mathbf{u}_K$ wouldn't be linearly-independent.)
- There are therefore 2^K codewords. We can encode a block consisting of K symbols, $a_1, \ldots, a_k$, from Z_2 as a codeword of length N using the formula above.
- This is called an $[N, K]$ code. (MacKay's book uses (N, K), but that has another meaning in other books.)

Linear Codes From Linear Equations — 10

- Another way to define a linear code for Z_2^N is to provide a set of simultaneous equations that must be satisfied for $\mathbf{v}$ to be a codeword.
- These equations have the form $\mathbf{c} \cdot \mathbf{v} = 0$, ie

  $$c_1v_1 + c_2v_2 + \cdots + c_Nv_N \;=\; 0$$
- The set of solutions is a linear code because $\mathbf{c}\cdot\mathbf{u} = 0$ and $\mathbf{c}\cdot\mathbf{v}=0$ implies $\mathbf{c}\cdot(\mathbf{u}+\mathbf{v}) = 0$.
- If we have $N-K$ such equations, and they are independent, the code will have 2^K codewords.
- The basis representation and the constraint equation representations are equivalent: we can always convert from one to the other. (In linear algebra terms, we can either specify a basis for the codeword subspace or a basis for its complement null space.)
- If K is close to N, it is more compact to specify the constraint equations; if K is close to 0, it is more compact to specify the basis.

The Repetition Codes Over Z_2 — 11

- A repetition code for Z_2^N has only two codewords — one has all 0s, the other all 1s.
- This is a linear $[N, 1]$ code, with $(1, \ldots, 1)$ as the basis vector.
- The code is also defined by the following $N-1$ equations satisfied by a codeword $\mathbf{v}$:

  $$v_1 + v_2 = 0, \quad v_2 + v_3 = 0, \quad \cdots, \quad v_{N-1} + v_N = 0$$
- Each of these equations has two solutions, $\{0, 0\}$ and $\{1, 1\}$. But the only solutions which satisfy them all are all 0s or all 1s.

110 → 111111110000

K=3 message bits → N=12 transmission bits

The Single Parity-Check Codes 12

- An $[N, N-1]$ code over Z_2 can be defined by the following single equation satisfied by a codeword $\mathbf{v}$:

$$v_1 + v_2 + \cdots + v_N \;=\; 0$$

In other words, the *parity* of all the bits in a codeword must be even.

- This code can also be defined using $N-1$ basis vectors.
One choice of basis vectors when $N = 5$ is as follows:

$$\begin{array}{c}(1,0,0,0,1)\\(0,1,0,0,1)\\(0,0,1,0,1)\\(0,0,0,1,1)\end{array}$$

110010 → 1100101

K=6 message bits → N=7 transmission bits

A $[5, 2]$ Binary Code 13

- Recall the following code from lecture 13 (page 12):

$$\{\ 00000,\quad 00111,\quad 11001,\quad 11110\ \}$$

- Is this a linear code?
We need to check that all sums of codewords are also codewords:

$$\begin{array}{c}00111 + 11001 = 11110\\00111 + 11110 = 11001\\11001 + 11110 = 00111\end{array}$$

- We can generate this code using 00111 and 11001 as basis vectors.
We then get the four codewords as follows:

$$\begin{array}{c}0 \cdot 00111 + 0 \cdot 11001 = 00000\\0 \cdot 00111 + 1 \cdot 11001 = 11001\\1 \cdot 00111 + 0 \cdot 11001 = 00111\\1 \cdot 00111 + 1 \cdot 11001 = 11110\end{array}$$

Generator Matrices 14

- We can arrange a set of basis vectors for a linear code in a *generator matrix*, each row of which is a basis vector.
- A generator matrix for an $[N, K]$ code has K rows and N columns.
- We can use a generator matrix for an $[N, K]$ code to encode a block of K message bits as a block of N bits to send through the channel.
- We regard the K message bits as a row vector, $\mathbf{s}$, and multiply by the generator matrix, G, to produce the channel input, $\mathbf{t}$:

$$\mathbf{t} \;=\; \mathbf{s}G$$

- If the rows of G are linearly independent, each distinct message $\mathbf{s}$ will produce a different channel encoding $\mathbf{t}$, and every $\mathbf{t}$ that is a valid codeword will be produced by some $\mathbf{s}$.
- Note: Almost all codes have more than one generator matrix.

Example: Generator Matrix for the $[5, 2]$ Code 15

- Here's a generator matrix for the $[5, 2]$ code looked at earlier:

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

- Encoding the message block $(1, 1)$ using the generator matrix above:

$$\mathbf{s}G \;=\; \mathbf{t}$$

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

Parity-Check Matrices 16

- Suppose we have specified an $[N, K]$ code by a set of $M = N - K$ equations satisfied by any codeword, $\mathbf{v}$:

$$\begin{aligned} c_{1,1}\, v_1 + c_{1,2}\, v_2 + \cdots + c_{1,N}\, v_N &= 0 \\ c_{2,1}\, v_1 + c_{2,2}\, v_2 + \cdots + c_{2,N}\, v_N &= 0 \\ &\vdots \\ c_{M,1}\, v_1 + c_{M,2}\, v_2 + \cdots + c_{M,N}\, v_N &= 0 \end{aligned}$$

- We can arrange the coefficients in these equations in a *parity-check matrix*, as follows:

$$\begin{bmatrix} c_{1,1} & c_{1,2} & \cdots & c_{1,N} \\ c_{2,1} & c_{2,2} & \cdots & c_{2,N} \\ & & \vdots & \\ c_{M,1} & c_{M,2} & \cdots & c_{M,N} \end{bmatrix}$$

- If $\mathcal{C}$ has parity-check matrix H, we can check whether $\mathbf{v}$ is in $\mathcal{C}$ by seeing whether $\mathbf{v}H^T = \vec{0}$.

  Note: Almost all codes have more than one parity-check matrix.

Example: Parity Check Matrix for the $[5, 2]$ Code 17

- Here is one parity-check matrix for the $[5, 2]$ code used earlier:

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

- We see that 11001 is a codeword as follows:

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

- But 10011 isn't a codeword, since

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

Repetition Codes and Single Parity-Check Codes 18

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

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

  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:

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

  Here is the parity-check matrix for this code:

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