George Verghese
verghese@mit.edu

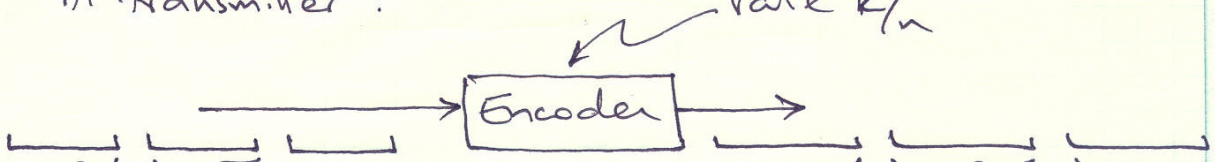
6.02
9/30/10

Rec. 7

Linear block codes over $\mathbb{F}_2$

'Galois field of 2 elements'

At transmitter:



Blocks of k bits — messages, 2^k possibilities for each message block

Blocks of n bits, $n > k$ — codewords, only 2^k of them, out of 2^n possibilities, to create 'distance' between valid codewords — permitting errors to be recognized (since one bit error in a codeword will then result in a nonsense word, not another codeword), and maybe corrected!

Aside on $\mathbb{F}_2$:

Only 0, 1, ×, ⊕ ('exclusive OR')

× : regular multiplication

Ex OR: $\begin{cases} 0 \oplus 0 = 0 \\ 0 \oplus 1 = 1 \\ 1 \oplus 0 = 1 \\ 1 \oplus 1 = 0 \end{cases}$ Equivalent to working with integers mod 2

Natural distance measure for such blocks of 0's and 1's (or vectors) is the Hamming distance:

$D(v, w)$ = # positions in which they differ
= minimum number of bit-flips (components in) of one of the vectors required to make it equal to the other.

(Two (0,1) blocks of equal length, e.g., two n-bit codewords)

e.g. $D([1\ 0\ 1\ 1],[1\ 1\ 0\ 1]) = 2$

$D([0\ 1\ 1\ 0],[0\ 0\ 0\ 0]) = 2$

(denote the zero-vector by $\mathbf{0}$.)

Another useful notion is the weight of an $\mathbb{F}_2$-vector:

$W(v)$ = # 1's in the vector v.

It's easy to show that

(*) $D(v,w) = W(v-w) = W(v \oplus w)$

How is '$-$' defined in $\mathbb{F}_2$? → Think mod-2:

$0-0=0$
$0-1=1$ (because $0 = 1 \oplus 1$)
$1-0=1$
$1-1=0$

So $-$ is the same as $\oplus$!

So $v-w$ will have a 1 in each bit position where v and w differ, 0 elsewhere.

These 3 conditions show $D(\cdot,\cdot)$ is a "metric" for $\mathbb{F}_2$-vectors

$D(v,v) = 0$; $D(v,w) = D(w,v)$; and

$D(v,w) \le D(v,u) + D(u,w)$ ← "triangle inequality"

Proof of this last claim (Prob. 1 of Lec 6 notes):
The minimum number of bit-flips to convert v to w cannot exceed the (minimum) number of bit-flips needed to first convert v to u, and then u to w.

If you know how matrix multiplication works, here's a good way to think about how an encoder generates a linear code:

$$\underbrace{[1\ 0\ 1\ 1]}_{\substack{k(=4)\ \text{bits,}\\ \text{message block}}} \underbrace{\begin{bmatrix} 1&0&0&0&1&1&0\\ 0&1&0&0&1&0&1\\ 0&0&1&0&0&1&1\\ 0&0&0&1&1&1&1 \end{bmatrix}}_{\text{"Generator" matrix } G \text{ for specific linear code } \mathcal{L}_G} = \underbrace{[1\ 0\ 1\ 1\ 0\ 0\ 1]}_{\substack{n(=7)\ \text{bits,}\\ \text{codeword block}}}$$

- This works the same as regular matrix multiplication, except that all the additions are $\oplus$, i.e., in $\mathbb{F}_2$.
- 2 ways to think of matrix multiplication:

(1) The j-th entry of the n-vector on the right is the inner product of the k-vector on the left with the j-th column of G.

e.g., the 3rd entry on the right is (3rd entry of codeword)

$$\underbrace{[1\ 0\ 1\ 1]}_{\text{message vector}} \begin{bmatrix} 0 \\ 0 \\ 1 \\ 0 \end{bmatrix} = (1\times 0) \oplus (0\times 0) \oplus (1\times 1) \oplus (1\times 0) = 1$$

↑ 3rd column of G

(2) Alternatively, the row vector on the right (the codeword) is a linear combination of the rows of G, with weights given by the entries of the row vector on the left (the message):

$$\begin{aligned} &(1\times[1\ 0\ 0\ 0\ 1\ 1\ 0]) \\ \oplus&(0\times[0\ 1\ 0\ 0\ 1\ 0\ 1]) \\ \oplus&(1\times[0\ 0\ 1\ 0\ 0\ 1\ 1]) \\ \oplus&(1\times[0\ 0\ 0\ 1\ 1\ 1\ 1]) \\ =\ &\underbrace{[1\ 0\ 1\ 1\ 0\ 0\ 1]}_{\text{codeword associated with message } [1\ 0\ 1\ 1]} \end{aligned}$$

Either way, we see that this is a linear code by the first definition of 'linear code' in the lecture notes: 'Each codeword bit is a linear combination of message bits'.

The second (equivalent) definition in the notes is just a superposition statement, given the above linearity: Since the sum of any two (of the 2^k) messages is again a valid message, the sum of any two codewords is a valid codeword.

(We will use the word code to mean either the mapping from messages to codewords, or just the set of codewords — you can infer the sense from the context.)

From the characterization of a linear code at the bottom of p.③, we have the following conclusion:

The zero vector ⓪ must be in any linear code $\mathcal{L}$.

Why? Because for any $v \in \mathcal{L}$ (i.e., any codeword v), $v \oplus v$ must be in $\mathcal{L}$, but $v \oplus v = ⓪$.

The key measure of error detection/correction capability of a code $\mathcal{C}$ is the minimum Hamming distance d between all distinct pairs of codewords:

$$d(\mathcal{C}) = \min_{v,w \in \mathcal{C},\, v \neq w} D(v,w)$$

For a linear code $\mathcal{L}$, this minimum Hamming distance is easy to find:

$$d(\mathcal{L}) = \min_{v \in \mathcal{L},\, v \neq ⓪} W(v)$$

(W ↑ weight)

← Involves a search over only $2^k - 1$ objects, not $\binom{2^k-1}{2}$.

i.e., it's the minimum weight among all nonzero codewords.

Why? Suppose v and w are two distinct codewords in $\mathcal{L}$ that are at the minimum Hamming distance, i.e., $D(v,w) = d(\mathcal{L})$. From (*) on p.②,

$D(v,w) = W(\underbrace{v \oplus w}_{\text{Some codeword, since } v \text{ and } w \text{ come from a linear code.}})$

Just for fun! Optional

A small detour to tackle Prob. 5 of Lec 6 notes:

Note that $v \oplus w$ will have a 1 in any particular bit position precisely when either v or w — but not both — has a 1 in that position. It follows that

(weight of vector)

$$W(v \oplus w) = W(v) + W(w) - 2q,$$

where q is the number of bit positions in which both v and w have a 1.

It also follows that the sum of two even-weight or two odd-weight codewords has even weight, while the sum of an even-weight and an odd-weight vector has odd weight.

The claim in Prob. 5 of Lec 6 notes: A linear block code has either only even-weight codewords or has an equal number of even-weight and odd-weight codewords.

Proof: Suppose there is at least one odd-weight codeword, c. List all the distinct even-weight codewords: $v_0 = 0, v_1, v_2, \ldots, v_{E-1}$, if there are E even codewords altogether. Then

$$c \oplus v_j, \quad j = 0, 1, \ldots, E-1$$

must all be codewords, since the code is linear; and all these codewords will have odd weight. Since $v_j \neq v_k$ for $j \neq k$, it must be that $c \oplus v_j \neq c \oplus v_k$ for $j \neq k$ (otherwise adding c to both sides would give $0 \oplus v_j = 0 \oplus v_k$, a contradiction), so we must have at least E distinct odd-weight vectors, i.e. codewords, too. There actually cannot be any further odd-weight vectors.

Because if there was one, say c', then $c \oplus c'$ would be even-weight, $= v_j$ for some j. →

But $c \oplus c' = v_j$ implies $\underbrace{c \oplus c}_{0} \oplus c' = c \oplus v_j$,

or $c' = c \oplus v_j$, but this is an odd-weight vector we already had, not a new one. The proof is done! (Perhaps you can find a shorter and more elegant proof — in that case, please let me know!)

I'll leave you to read up on rectangular codes from the Lec. 6 notes (and it will come up in Lab as well).

Hamming Codes

Main idea: add multiple parity bits to the message block (at the transmitter), each based on satisfying a distinct parity relation involving a subset of message bits. At the receiver, run corresponding parity checks, and design things so that failures of these parity checks will pinpoint (and allow correction of) single-bit errors.

Suppose we use q parity bits for a k-bit message, so $n = k + q$. Denote the message bits by $M_1, \ldots, M_k$, and the parity bits by $P_1, \ldots, P_q$. (n: codeword size)

With each parity bit P_j, associate a parity relation R_j that is the $\oplus$ sum of P_j and some selected subset of message bits — then pick P_j at the transmitter to ensure $R_j = 0$.

e.g., if $R_1 = P_1 \oplus M_1 \oplus M_2 \oplus M_4$ (**)

then when $M_1, M_2, M_4 = 1, 0, 0$ respectively, we choose $P_1 = 1$ at the transmit end; and when $M_1, M_2, M_4 = 1, 1, 0$, choose $P_1 = 0$.

To test the received message, the receiver evaluates these same relations, but using the

received rather than transmitted bits, of course. So at the receiving end we call these relations parity checks or syndromes, denoted by $S_1, \ldots, S_q$.

e.g., for the specific parity relation R_1 on p. (6), (**), we have the parity check

$$S_1 = P_1' \oplus M_1' \oplus M_2' \oplus M_4'$$

where we use the primes $'$ to indicate that bit-errors can cause these bits to differ from what was transmitted; but $M_i' = M_i$ and $P_j' = P_j$ if no errors.

So whereas the P_j at the transmitter were chosen to ensure $R_j = 0$, now S_j can be 1 — and will be 1 if there is a single-bit error affecting one of the bits that comprise it.

Hamming's clever design of the codeword allows the binary number $S_q \ldots S_1$ to directly mark the location in the code word where a single-bit error occurred (and in the absence of error this number is 0). For this, choose the code as follows:

e.g. Suppose we decide to use 4 parity bits, then $S_4 S_3 S_2 S_1$ can count from 0 to 16 (with 0 indicating no error). So we can have $n = 15$, $q = 4$, $k = 11$. Arrange the codeword with P_1, P_2, P_3, P_4 sitting in positions $2^0, 2^1, 2^2, 2^3$, and the message bits in the remaining positions:

1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
P_1	P_2	M_1	P_3	M_2	M_3	M_4	P_4	M_5	M_6	M_7	M_8	M_9	M_{10}	M_{11}

Keeping in mind that the binary position of a single bit error is to be located via the binary number $S_4 S_3 S_2 S_1$, we see that the relation defining S_1 — and R_1 — should involve the codeword bits that have a 1 in the least significant bit of

the binary representation of their position in the codeword. Hence

$$R_1 = P_1 \oplus M_1 \oplus M_2 \oplus M_4 \oplus M_5 \oplus M_7 \oplus M_9 \oplus M_{11}.$$

Similarly, S_2 and R_2 should involve the codeword bits that have a 1 in the next significant bit of the binary representation of their position in the codeword. Hence

$$R_2 = P_2 \oplus M_1 \oplus M_3 \oplus M_4 \oplus M_6 \oplus M_7 \oplus M_{10} \oplus M_{11}.$$

And proceeding in the same way,

$$R_3 = P_3 \oplus M_2 \oplus M_3 \oplus M_4 \oplus M_8 \oplus M_9 \oplus M_{10} \oplus M_{11}$$

and

$$R_4 = P_4 \oplus M_5 \oplus M_6 \oplus M_7 \oplus M_8 \oplus M_9 \oplus M_{10} \oplus M_{11}.$$

This completely specifies the Hamming (15, 11) code!

Remaining questions:

1. Is the Hamming code linear? Yes, because each codeword bit is a linear combination of message bits — because each P_j can be expressed as a linear combination of message bits. This is because P_j is picked to make $R_j = 0$, so adding P_j to both sides of the parity relation gives $P_j \oplus R_j = P_j$ on the left, and $\underbrace{P_j \oplus P_j}_{=0} \oplus$ sum of message bits on the right, i.e., expresses P_j as a sum of message bits.

   e.g., from (**) on p. 6, $P_1 = M_1 \oplus M_2 \oplus M_4$.

2. What is the minimum distance of the Hamming code? Must be ≥ 3, since single-error correction is possible. And a nonzero vector of weight 3 is provided by $M_1 = 1$, all other $M_i = 0$, so $P_1 = 1$, $P_2 = 1$. Hence minimum distance = 3.

Can also display the generator matrix G for any Hamming code; G for the (7, 4) Hamming, rearranged to have message bits first.

"Enough already!!"