Lecture 3:

Lecture notes posted online each week.

Recall Defn *Hamming distance:* for words $\overline{x} = x_1 \dots x_n, \overline{y} = y_1 \dots y_n$ of the same length over the same alphabet,

$$d(\overline{x}, \overline{y}) = |\{1 \leq i \leq n : x_i \neq y_i\}|$$

i.e., $d(\overline{x}, \overline{y})$ is the number of of places at which x and y differ.

Recall Proposition: $d(\overline{x}, \overline{y})$ is a metric, i,e.,

(0) $d(\overline{x}, \overline{y}) \geq 0$

(1) $d(\overline{x}, \overline{y}) = 0$ iff $\overline{x} = \overline{y}$

(2) $d(\overline{x}, \overline{y}) = d(\overline{y}, \overline{x})$

(3) $d(\overline{x}, \overline{z}) \leq d(\overline{x}, \overline{y}) + d(\overline{y}, \overline{z})$ (triangle inequality)

Proof: 0, 1 and 2 are obvious.

(3) (there is a simple proof in the text). Alternatively

$d(\overline{x}, \overline{y})$ can be written in terms of the distance function for words of length 1:

$$d(\overline{x}, \overline{y}) = \sum_{i=1}^{n} d(x_i, y_i)$$

where

$$d(x_i, y_i) = \begin{array}{ll} 0 & x_i = y_i \\ 1 & x_i \neq y_i \end{array}$$

Prove (3) first for length $n = 1$:

Case 1: If $x = z$, then LHS is 0 and so LHS $\leq$ RHS.

Case 2: If $x \neq z$, then LHS is 1 and either $x \neq y$ or $y \neq z$. So, RHS is at least 1. So, LHS $\leq$ RHS.

For general n:

– By the case, $n = 1$, for each i, $d(x_i, z_i) \leq d(x_i, y_i) + d(y_i, z_i)$. So,

$$d(\overline{x}, \overline{z}) = \sum_{i=1}^{n} d(x_i, z_i) \leq \sum_{i=1}^{n} d(x_i, y_i) + d(y_i, z_i) = \sum_{i=1}^{n} d(x_i, y_i) + \sum_{i=1}^{n} d(y_i, z_i)$$

$$= d(\overline{x}, \overline{y}) + d(\overline{y}, \overline{z})$$

QED

Q: Why is it called the triangle inequality?

A: Analogue with euclidean distance in the plane: picture

Defn: *Minimum Distance of a code C*:

$$d(C) = \min(d(\overline{c}, \overline{c}') : \overline{c}, \overline{c}' \in C, \overline{c} \neq \overline{c}')$$

Notation: (n, M, d)-*code:* a code of length n, size M and minimum distance d

– The larger M is, the more distinct messages can be transmitted.

– The larger $d(C)$ is, the more "spread out" are the codewords and so the more errors can be corrected.

If we want to emphasize the alphabet size q, then we write $(n, M, d)_q$.

Examples: Recall the codes:

2-repetition code $(n, M, d) = (2, 2, 2)$

$C = \{aa, bb\}$ $(n, M, d) = (2, 2, 2)$

3-repetition code $(n, M, d) = (3, 2, 3)$

n-repetition code $(n, M, d) = (n, 2, n)$

C_1: (2,4,1)

C_2: (3,4,2)

C_3: (5,4,3)

C_4 (the ternary 6-repetition code): $(6, 3, 6)_3$

For C_2:

	000	011	101	110
000	0			
011	2	0		
101	2	2	0	
110	2	2	2	0

For C_3:

	00000	01101	10110	11011
00000	0			
01101	3	0		
10110	3	4	0	
11011	4	3	3	0

To assess the goodness of a code, you need to specify an encoder and a decoder (or decoding algorithm).

The encoder is simply a one-to-one mapping from all words of some length k into the codewords.

Defn *Nearest-neighbor (minimum distance) decoder (NND)*:

Given a code C and received word $\overline{x}$ find the codeword $\overline{c} \in C$ closest, in Hamming distance, to $\overline{x}$. Decode $\overline{x}$ to $\overline{c}$.

If $\overline{x}$ is a codeword, then $\overline{c} = \overline{x}$. Otherwise, it does its best to find the transmitted codeword.

Note: Recall that a decoder performs two functions:

– (i) given received word, finds the most likely transmitted codeword

– (ii) inverts the encoder

Here, we are ignoring (ii), since it is much easier than (i), and the concept of NND has only to do with the code, not the encoder.

Two types of NND:

Complete NND: If there is more than one closest codeword $\overline{c}$, decode to an arbitrary such $\overline{c}$.

Incomplete NND : If there is more than one closest codeword, declare an error.

Q: Why would you choose Incomplete?

A: For very noisy channels, Complete may be too risky.

We usually assume Incomplete.

Example C_3:

Assume 11011 was transmitted.

received word	# channel errors	dist to closest codeword	decoded word (Inc)	decoded word (Comp)
11011	0	0	11011	11011
11111	1	1	11011	11011
10111	2	1	10110	10110
11000	2	2	?	00000 or 11011

Table showing the distance between received word (rows) and each codeword (colums).

	00000	10110	01101	11011
11011	4	3	3	0
11111	5	2	2	1
10111	4	1	3	2
11000	2	3	3	2

Note: decoding succeeded when at most 1 channel error is made; this is related to $d(C_3) = 3$ as follows.

Defn: Let $v \in \mathbb{N}$ (the natural numbers, 1, 2, . . .

A code C is *v-error-correcting* if, using INND, the decoder corrects all occurrences of v or fewer errors in any codeword.

Theorem 1.9(ii): A code C is v-error-correcting iff $d(C) \geq 2v+1$.

Note: "iff" means if and only if.

Defn: *Hamming Ball:* Let $\overline{x} \in F^n$ and $r \in \mathbb{N}$.

$$B_r(\overline{x}) = \{\overline{y} \in F^n : \ d(\overline{x}, \overline{y}) \leq r\}.$$

Analogue with Euclidean ball.

Proof of Theorem (*If*):

We first claim that the Hamming balls of radius v centered at codewords of C are pairwise disjoint. Suppose not. If $\overline{c}, \overline{c}' \in C$, $\overline{c} \neq \overline{c}'$ and $B_v(\overline{c}) \cap B_v(\overline{c}') \neq \emptyset$, there exists some $\overline{x} \in B_v(\overline{c}) \cap B_v(\overline{c}')$. Then

$$d(\overline{c}, \overline{c}') \leq d(\overline{c}, \overline{x}) + d(\overline{x}, \overline{c}') \leq 2v.$$

a contradiction

Let $\overline{c}$ be transmitted codeword and $\overline{x}$ be received. Assume that at most v errors were made, i.e., $d(\overline{c}, \overline{x}) \leq v$. Then $\overline{x} \in B_v(\overline{c})$ and for any other $\overline{c}' \in C$, $\overline{x} \notin B_v(\overline{c}')$. Thus, $\overline{c}$ is the closest codeword to $\overline{x}$ and so NND will decode $\overline{x}$ to $\overline{c}$.

Only if: HW1.

Equivalent statement:

$$d(C) \geq 2v + 1 \text{ iff } v \leq \lfloor \frac{d(C) - 1}{2} \rfloor.$$

where $\lfloor r \rfloor$ is the greatest integer less than or equal to r (e.g., $\lfloor 2.5 \rfloor = 2$.

So, to correct 1 error, you need $d(C) \geq 3$; to correct 2 errors you need $d(C) \geq 5$.

Examples:

– 3-repetition code is 1-error-correcting.

– C_3 is 1-error correcting.

– 5-repetition code is 2-error-correcting.

Defn: Let $u \in \mathbb{N}$. A code C is *u-error-detecting* if whenever at least one but at most u errors are made, the received word is *not* a codeword.

Why is this called "error-detecting?"

Theorem 1.9(i): A code C is u-error-detecting iff $d(C) \geq u + 1$.

Proof (*If*): if at most u errors are made, then the received word cannot be a codeword, and so the error can be detected.

(*Only If*): HW1.

Equivalent statement:

$$d(C) \geq u + 1 \text{ iff } d(C) - 1 \geq u$$

So, to detect 1 error, you need $d(C) \geq 2$; to detect 2 errors you need $d(C) \geq 3$.

Lecture 4:

Recall: If $d(C) \geq 2v+1$, then C can correct up to v errors (using NND decoder).

If $d(C) \geq u+1$, then C can detect up to u errors (declares an error if received word is not a codeword and returns the received word if it is a codeword).

Different decoding modes: Can use a code (i.e., a set of codewords) as either error-correcting or error detecting.

Example: if $d(C) = 3$, then C can detect (up to) 2 errors OR correct (up to) 1 error.

But it must decide what it wants to do in advance.

Example: 3-repetition code (in this case, NND = majority vote):

Assume 111 was transmitted:

received word	# channel errors	Correction mode	Detection mode
111	0	111	111
110	1	111	?
100	2	000	?
000	3	000	000

Note: if 2 errors are made in a codeword, then the detector will detect that there is an error, but it cannot in general say how many errors.

Can do the same for C_3.

Q: If $d(C) = 3$, does there exist a *hybrid*-decoder:

– if (exactly) 2 errors are made, decoder is guaranteed to detect;

AND

– if (exactly) 1 error is made, decoder is guaranteed to correct.

A: No: Suppose 110 were received.

If 111 were transmitted (so 1 error), then decoder must decode to 111.

But if 000 were transmitted (so 2 errors), then decoder should declare an error.

Thus, decoder must do two different things if 110 is received!

Example: if $d(C) = 4$, then C can

1. (error detection) detect (up to) 3 errors OR
2. (error correction) correct (up to) 1 error OR
3. (hybrid decoder) simultaneously (guaranteed!) detect 2 errors and correct 1 error

But it must decide what it wants to do in advance.

Hybrid Decoder:

Let $\overline{x}$ be received word and $\overline{c}'$ be some nearest neighbour codeword.

— If $d(\overline{x}, \overline{c}') \leq 1$, decode to $\overline{c}'$ (in this case, there can be only one such $\overline{c}'$)

— If $d(\overline{x}, \overline{c}') \geq 2$, declare an error.

Proof that this works:

Let $\overline{c}$ be transmitted word , $\overline{x}$ received word, and $\overline{c}'$ a nearest neighbour codeword to $\overline{x}$.

Case 1: 1 channel error.

Then $d(\overline{x}, \overline{c}) = 1$ and so $d(\overline{x}, \overline{c}') \leq 1$. Thus, decoder decodes to $\overline{c}'$. And

$$d(\overline{c}, \overline{c}') \leq d(\overline{c}, \overline{x}) + d(\overline{x}, \overline{c}') \leq 2.$$

Since $d(C) = 4$, $\overline{c} = \overline{c}'$.

So, decoder correctly corrects 1 error.

Case 2: 2 channel errors.

Then $d(\overline{x}, \overline{c}) = 2$.

If $d(\overline{x}, \overline{c}') \leq 1$, then

$$d(\overline{c}, \overline{c}') \leq d(\overline{c}, \overline{x}) + d(\overline{x}, \overline{c}') \leq 3.$$

Since $d(C) = 4$, $\overline{c} = \overline{c}'$. But impossible since $d(\overline{x}, \overline{c}') \neq d(\overline{x}, \overline{c})$.

Thus, $d(\overline{x}, \overline{c}') \geq 2$, and decoder correctly detects an error. $\square$

Can we do any better if $d(C) = 4$? No.

Example: $C = \{0000, 1111\}$ (4-repetition code).

You cannot simultaneously guarantee: correction of 1 error and detection of 3 errors

– Suppose 0001 were received.

– 0000 could have been sent (in which case decoder should correct to 0000)

– 1111 could have been sent (in which case decoder should detect an error) $\square$

Defns: 1. the *weight* $w(\overline{x})$ of a binary word $\overline{x}$ is the number of 1's it contains.

2. for binary words $\overline{x}, \overline{y}$, define $\overline{x} \oplus \overline{y} := \overline{z}$ where $z_i = x_i \oplus y_i$ and $\oplus$ is *mod-2 addition* (or *exclusive or*).

Useful fact: $d(\overline{x}, \overline{y}) = w(\overline{x} \oplus \overline{y})$ because $x_i \neq y_i$ iff $x_i \oplus y_i = 1$.

Another useful fact: Let A and B be finite sets and $f : A \to B$ be a function.

– If f is 1-1 (i.e., whenever $a, a' \in A$ and $a \neq a'$, then $f(a) \neq f(a')$), then $|A| \leq |B|$.

– If f is onto (i.e., for all $b \in B$, there exists $a \in A$ s.t. $f(a) = b$), then $|A| \geq |B|$.

– If f is 1-1 and onto, then $|A| = |B|$.

For fixed n and q, for an $(n, M, d)_q$ code, there is a trade-off (fight) between M and d.

Defn: $A_q(n, d)$ is the largest M such that there is an $(n, M, d)_q$ code.

For fixed n and q, as d increases, one would expect that $A_q(n, d)$ decreases.

A major coding problem: determine $A_q(n, d)$.

Clearly, $A_q(n, 1) = q^n$

Proposition: $A_q(n, n) = q$

Proof: If C is code of length n, then $d(C) = n$ iff all codewords differ in all positions. Thus, the map from C to F, defined by

$$f(x_1 x_2 \dots x_n) = x_1$$

is 1-1. Thus, $A_q(n, n) \leq q$. But the q-ary n-repetition code contains exactly q codewords. So, $A_q(n, n) = q$.

In HW1, you will find: $A_2(n, n-1)$ and $A_2(n, 2)$.

Some other specific values (or ranges) of $A_q(n, d)$ are given in Table 2.4 in the text.

But in general $A_q(n, d)$ is hard to find. In fact, it is Unsolved! Some more recent partial results will be posted on the website.