§ Coding Theory and Cryptography

Lecture 6

Warm up problem: Wrong Number. (Similar to page 17 and 5.7 of Ecco)

In a city called Five people are getting very upset. In Five, phone numbers are 5 digits long made with the numbers 0 to 4, but lately many phone calls were resulting in wrong numbers. After a correct number has been called, in transmission, one pair of adjacent digits gets swapped ("the switch bug"). For example the number ABCDE could be called but whoever is at ABDCE receives the phone call.

After a heated city hall discussion, going against all cultural beliefs the city has decided to add a sixth number to their phone system. This was decided even though the sixth digit will still be prone to swapping with the fifth digit. The scientists of Five are going to add a sixth digit called a *check digit*. After doing so, called numbers that experience "the switch bug" will result in a nonfunctioning number.

How can the check digit be chosen successfully?

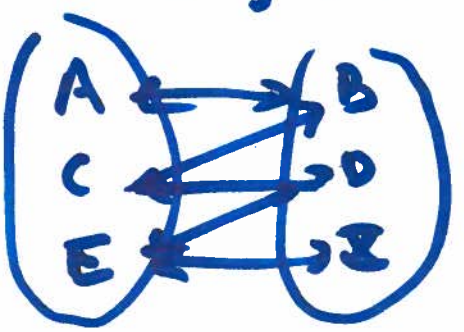
Coding Theory

Definition 1: A *codeword* is a word (or string) of digits or letters or other symbols.

Definition 2: A *code* is a collection of codewords.

Example 1: Find a solution to the Wrong Number problem:

Put the digits into 2 groups:



No digits get swapped within a group.

Pick $\underline{X}$ so that

$$\boxed{B + D + \underline{X} \equiv 0} \pmod 5$$

and suppose there is an error: $C \leftrightarrow D$

Case 1: $\boxed{B + C + \underline{X} \not\equiv 0} \pmod 5$

and in this case we report an error.

Case 2: $\boxed{B + C + \underline{X} \equiv 0} \pmod 5$

Subtracting we get $D - C \equiv 0 \pmod 5$

i.e. $C \equiv D \pmod 5$.

Since $C, D \in \{0, 1, 2, 3, 4\}$, then

$C \equiv D \pmod 5$ implies $C = D$, and

there is no error.

Note: In $\underline{A\,B\,C\,D\,E}\,\underline{X}$, we have

5 choices for A, 5 choices for B, ..., 5 choices for E

so 5^5 choices, but $\underline{X}$ is determined once

we choose A, B, C, D, E.

Note: In this solution to the wrong number problem, a code with 5^5 codewords was used.

Error Detection:

Definition 3: Given two codewords x, y of the same length the *Hamming distance* $H(x, y)$ is the number of places in which the components of the strings differ.

- if $x = 1010101010$, $y = 1111111111$ $\Longrightarrow$ $H(x,y) = 5$
- if $x = AB\$12$, $y = AC\#12$ $\Longrightarrow$ $H(x,y) = 2$
- if $x = 12345$, $y = 54321$ $\Longrightarrow$ $H(x,y) = 4$

Def: The set of all binary strings of length k, i.e. a string of 0's and 1's of length k, is denoted by B_k.

A code where each code word has length k is a collection of binary strings of length k.

Ex: If $k=3$, the binary strings of length k are

$$B_3 = \left\{ \begin{matrix} 0\,0\,0 \\ 0\,0\,1 \\ 0\,1\,0 \\ 0\,1\,1 \\ 1\,0\,0 \\ 1\,0\,1 \\ 1\,1\,0 \\ 1\,1\,1 \end{matrix} \right.$$

We can choose the Code to be

$$C = \left\{ \begin{matrix} c_1 = 0\,0\,1 \\ c_2 = 0\,1\,0 \\ c_3 = 1\,0\,0 \\ c_4 = 1\,1\,1 \end{matrix} \right.$$

Ex: What is the Hamming distance between pairs of code words?

$H(c_1, c_3) = 2$, $H(c_1, c_3) = 2$, $H(c_1, c_4) = 2$

$H(c_2, c_3) = 2$, $H(c_2, c_4) = 2$, $H(c_3, c_4) = 2$

Suppose a message is sent and one of the strings received is $011 = s_1$.

$H(s_1, c_1) = 1$

Therefore s_1 is not a legitimate code word! So we can detect the error.

Theorem: On the set of binary strings of length k, the Hamming distance is a distance function with the following properties:

(i). $H(x,y) \geq 0$ and $H(x,y) = 0$ if and only if $x = y$.

(ii) $H(x,y) = H(y,x)$

(iii). $H(x,y) \leq H(x,z) + H(z,y)$

proof:

(ii) $H(x,y)$ is the minimum # of bits that have to be changed to change x into y. We can also change x into y by first making $H(x,z)$ changes (changes x to z) and making $H(z,y)$ changes (changes z to y).

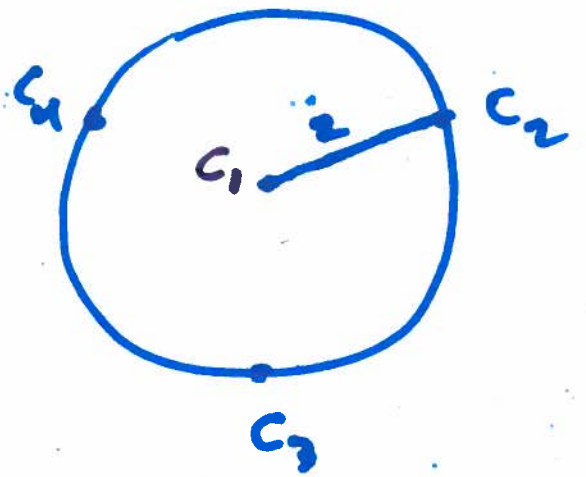
Therefore

$H(x,y) \leq H(x,z) + H(z,y)$

□

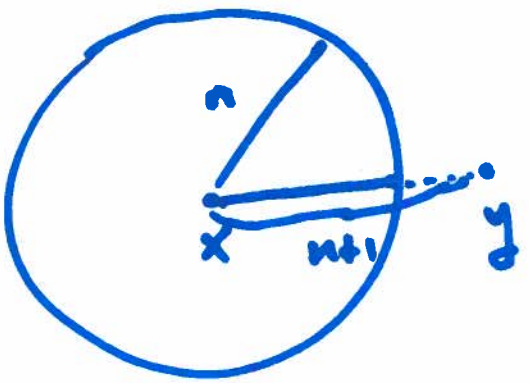
Ex: C: $\begin{cases} 001 = c_1 \\ 010 = c_2 \\ 100 = c_3 \\ 111 = c_4 \end{cases}$

$H(c_i, c_j) = 2$ if $i \neq j$.



If we receive a message with a string s_1 where $H(s_1, c_j) < 2$ then we can detect the error.

Theorem: If up to n characters of a codeword can be corrupted, and if the Hamming distance between every pair of codewords is at least $n+1$, then n errors can be detected.



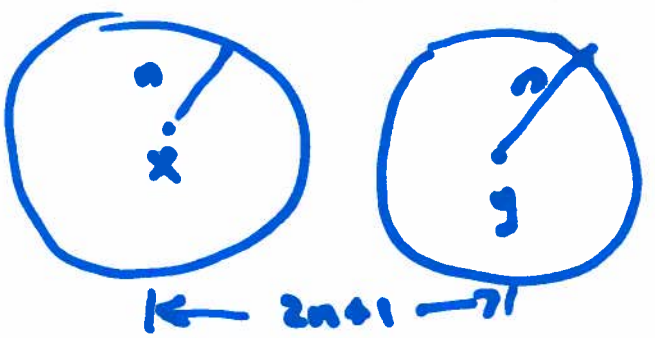
If x is any codeword any other codeword is at a distance at least n+1 from x.

Error Correction:

Suppose a codeword x (which we don't know) has been transmitted to us, and we receive a string y which may have been distorted by noise. It seems reasonable to decode y as that codeword x' (hope it is x) such that H(x',y) is a small as possible.

This is called nearest neighbour decoding this will maximise the decoders likelihood of correcting the error in some cases.

Theorem 2: If up to n characters of a codeword can be corrupted, and if the Hamming distance between every pair of codewords is at least $2n+1$, then n errors can be corrected.

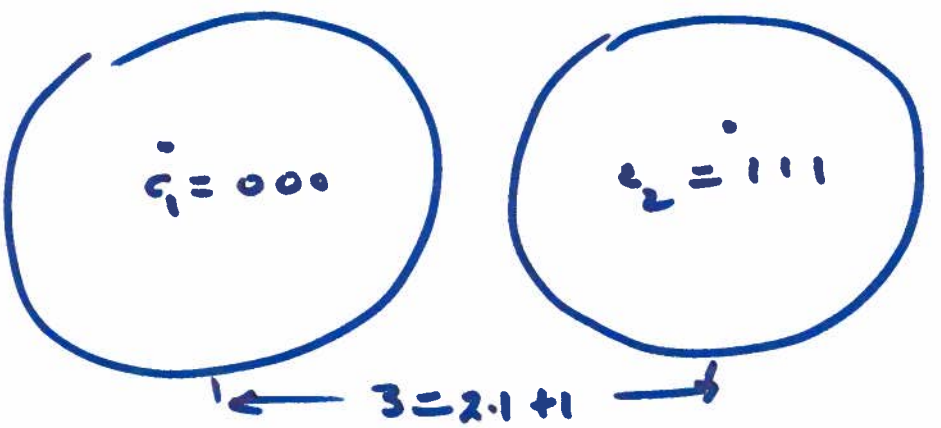


Ex: In the code

$$C: \begin{cases} c_1 = 000 \\ c_2 = 111 \end{cases}$$

$H(c_1, c_2) = 3 = 2 \cdot 1 + 1$

So we can correct up to 1 error.



The Hamming distance between the code words c_1 and c_2 is 3.

If we receive $\begin{cases} 001 \\ 010 \\ 100 \end{cases}$ we decode it as $c_1 = 000$

If we receive $\begin{cases} 011 \\ 101 \\ 110 \end{cases}$ we decode it as $c_2 = 111$.

Lecture 6 (Cont.)

· Coding Theory

(Scarlet, p. 22) **Error Detection**

Thm. 2.2.3 If up to n characters of a codeword can be corrupted, and if the Hamming distance between every pair of codewords is at least $n+1$, then up to n errors can be detected.

Converse also true:

Thm. 2.2.4 If up to n characters can be corrupted, and if the distance between some two of the codewords is smaller than $n+1$, then there is a corrupted message which cannot be detected.

Theorem 2.2.1 If up to n bits of a codeword can be corrupted, and if the distance between every pair of codewords is at least $2n+1$, then up to n errors can be corrected.

Converse is true:

Theorem 2.2.2 If up to n bits of a codeword can be corrupted, and if the distance between some two of the code words is smaller than $2n+1$, then there is a corrupted message which can not be corrected.

Error Detection:

- One card
- Visa or Mastercard
- ISBN
- UPC universal product code
- EAN European article #.

• Validating UPC + EAN

The secret is to use redundant data for error detection, usually done by means of a single digit called a check digit.

Fr "Java Examples in a Nutshell"

Check digit for ISBN is 5
for EAN is 3
for U.P.C. is 8

Ex1: Fr UPC:

6 [3] 6 [9] 2 [0] 9 [2] 3 [7] 1 [8]

starting with rightmost number, identify alternate digits.

. Add boxed digits: 3+9+0+2+7+8 = 29

. Add remaining digits ×3: 3(6+6+2+9+3+1) = 81

. Add the two numbers: 29+81 = 110

The U.P.C. is valid if the sum of the two numbers is divisible by 10.

Ex1: In general the UPC is a 12-digit number of the form

$a_{12}\ a_{11}\ a_{10}\ a_9\ a_8\ a_7\ a_6\ a_5\ a_4\ a_3\ a_2\ a_1$

when digits are labeled from the right

The UPC is accepted as being valid if the following number is divisible by 10:

$$3\cdot a_{12} + a_{11} + 3\cdot a_{10} + a_9 + 3\cdot a_8 + a_7 + 3\cdot a_6 + a_5 + 3\cdot a_4 + a_3 + 3\cdot a_2 + a_1$$

Ex2: EAN are validated in exactly the same way as UPC except that there are 13 digits instead of 12

$$a_{13} + 3\cdot a_{12} + a_{11} + 3\cdot a_{10} + \cdots + a_3 + 3\cdot a_2 + a_1$$

$$= 120$$

which is divisible by 10, so in Ex1 the EAN is valid.

Ex 3: The following UPC has its check digit missing. What should it be?

0-27242-40274 □

Let the check digit be x. Then the following # should be divisible by 10:

$$3\cdot 0+2+3\cdot 7+2+3\cdot 4+0+3\cdot 2+7+3\cdot 4+x$$

$$= 46+x$$

So the check digit is $x = 4$.

- **Validating the ISBN**

The 10-digit ISBN uses a modulo 11 check digit.

The check digit can be an X, denoting 10, as well as one of the digits 0 through 9.

Ex4: The ISBN number is

$a_{10}\, a_9\, a_8\, a_7\, a_6\, a_5\, a_4\, a_3\, a_2\, a_1$

the check digit a_1 must be chosen so that

$$10 \cdot a_{10} + 9 \cdot a_9 + 8 \cdot a_8 + 7 \cdot a_7 + 6 a_6 + 5 \cdot a_5 + 4 \cdot a_4 + 3 \cdot a_3 + 2 \cdot a_2 + a_1$$

is divisible by 11.

For Java book, we get

$$10 \cdot 1 + 9 \cdot 5 + 8 \cdot 6 + 7 \cdot 5 + 6 \cdot 9 + 5 \cdot 2 + 4 \cdot 3 + 3 \cdot 7 + 2 \cdot 1 + 5 = 242$$

and this is divisible by 11, so the ISBN is valid.

- S.I.N., Visa, Mastercard numbers (Called the IBM Scheme).

Suppose

S.I.N.: 324-217-694

VISA: 4002-1255-7001-69x

in both cases, the right-most digit is the check digit.

Ex 5: Validity of S.I.N.

Beginning with right-most digit identify alternate digits:

[3] 2 [4] 2 [1] 7 [6] 9 [4]

add the boxed digits $3+4+1+6+4 = 18$

Mult remaining $\times 2$: 4, 4, 14, 18

Add all digits of them: $4+4+1+4+1+8 = 22$

Add the two results $18+22 = 40$

Ex 5: The SIN is valid if the final result is divisible by 10.

Note: The same method is used to validate VISA and MASTERCARD.

Ex 6: Visa: Find the check digit x:

4002 - 1255 - 7001 - 069x

---

4 [0] 0 [2] 1 [2] 5 [5] 7 [0] 0 [1] 0 [6] 1 [x]

- Add the boxed digits: 0+2+2+5+0+1+6+x to get 16+x
- Double the remaining digits: 8, 4, 2, 10, 14, 0, 2, 18
- Add all those digits: 8+2+1+1+4+1+8 = 25
- Sum the results:

$$16 + x + 25 = 41 + x$$

To be valid, we need

$$41 + x \equiv 0 \pmod{10}$$

So that $x = 9$.

How good is an ISBN code?

The ISBN code has 9 information digits and one check digit for example:

8175257660

If the first nine digits of an ISBN are $abcdefghi$ then the tenth digit j is chosen so:

$$10a + 9b + 8c + 7d + 6e + 5f + 4g + 3h + 2i + j \equiv 0 \pmod{11}$$

When $j \equiv 10 \pmod{11}$, an X is used instead.

Example 4: The ISBN code can detect a single digit error: (as can UPC + IBM)

Suppose $a \to y$

Case 1: $10y + 9b + \cdots + j \not\equiv 0 \pmod{11}$, then report an error.

Case 2: $10y + 9b + \cdots + j \equiv 0 \pmod{11}$

Subtract: $10(y-a) \equiv 0 \pmod{11}$

Since 10 and 11 are relatively prime, by cancellation law:

$$y - a \equiv 0 \pmod{11}$$

i.e. $$y \equiv a \pmod{11}$$

Since $y, a \in \{0, 1, 2, \ldots 9\}$ then $y = a$, so there was no error.

Example 5: This means the ISBN code is a single error detecting code, so every pair of codewords has a Hamming distance of at least 2. For example:

$H(x,y) \geq 2$ for all code words

Since we can detect a single error.

Example 6: The ISBN code can detect when two adjacent symbols are interchanged (the switch bug):

Suppose $a_{10}\, a_9\, a_8\, a_7\, a_6\, a_5\, a_4\, a_3\, a_2\, a_1$ is valid ISBN, so the checksum is divisible by 11.

So

$$10 \cdot a_{10} + 9 \cdot a_9 + \cdots + 2a_2 + a_1 \equiv 0 \pmod{11}$$

Suppose a_n and a_m are switched

Before switch: [Some junk] $+ n \cdot a_n + m \cdot a_m \equiv 0 \pmod{11}$

After switch: [Some junk] $+ m \cdot a_n + n \cdot a_m \equiv 0 \pmod{11}$

So $(n-m)(a_n - a_m) \equiv 0 \pmod{11}$, So $a_n = a_m$.

Example 7: The ISBN code can correct a single digit error when the location of the error is known. For example:

Ecco's ISBN: $0-486-2961\,?-6$ $\quad ?=y$

$10 \cdot 0 + 9 \cdot 4 + 8 \cdot 8 + 7 \cdot 6 + 6 \cdot 2 + 5 \cdot 9 + 4 \cdot 6 + 3 \cdot 1 + 2y + 6$

$\equiv 2y + 1 \pmod{11}$, in $2y \equiv -1 \equiv 10 \pmod{11}$

11 & 2 are relatively prime, so $y \equiv 5 \pmod{11}$.

Example 8: Can The ISBN code correct a single digit error in general?

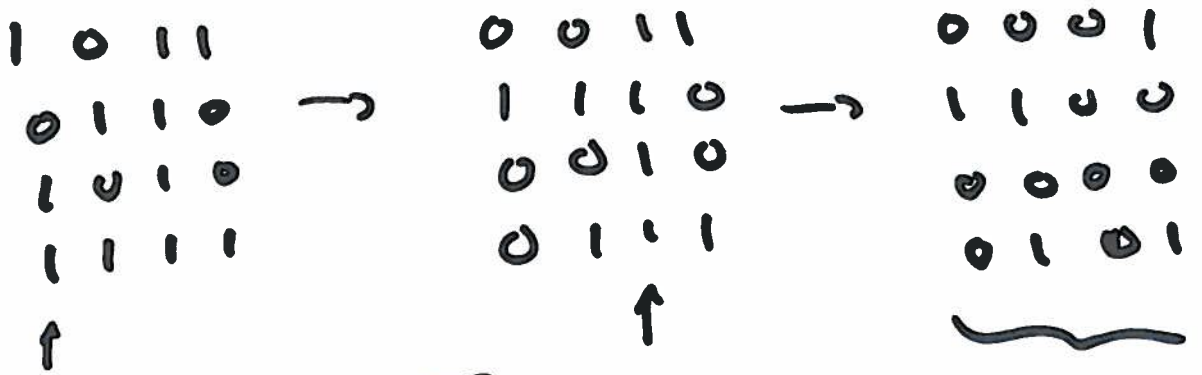
No! $H(x,y) \leq 2 < 2 \cdot 1 + 1 = 3$

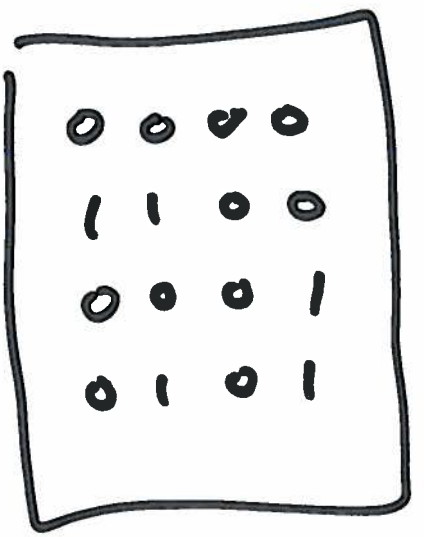
Note: The UPC, ISBN, IBM method will detect an error if a single digit is changed.

- The ISBN method will detect an error if any two digits are swapped.
- The UPC will detect an error if two adjacent digits are swapped provided the digits do not differ by 5.
- The IBM method will detect an error if two adjacent digits are swapped provided the digits are not 9 and 0.

Note: We may assume that 0000 is a code word, we can change the bits in 1 column without changing the relative differences between code words.

For example:





0000 is a codeword.

Ex: Suppose we have a code with two information bits, how many bits do we have to add to make a 1-error detecting code?

1-error correcting code?

0 0
0 1
1 0
1 1

} Given them information bits.

Can we add one more bit to each code word to get a 1-error detecting code? Can assume that 000 is a code word.

0 0 0
forced → 0 1 1
1 0 X ← what about?
1 1 0

Ex: Given info bits

0 0 x x
0 1 x x
1 0 x x
1 1 x x

Can we add 2 more bits to each code word to get an 1-error detecting code

Again:

0 0 0 0
0 1 0 1
1 1 0 0
1 0 0 1

} yes!

The code is

0 0 0 0
0 1 0 1
1 0 0 1
1 1 0 0

} 1-error detecting

Ex: Given info bits

0 0
0 1
1 0
1 1

Can we add 2 bits to get a 1-error correcting code?

need $d(x,y) \geq 3$

0 0 0 0
0 1 1 1
1 0 x x ← don't work.

If we add 3 bits:

0 0 0 0 0
0 1 0 1 1
1 0 1 0 1
1 1 0 1 0 X

doesn't work.

But,

0 0 0 0 0
0 1 0 1 1
1 0 1 1 1
1 1 1 0 0

} 1-error correcting. works!