

SECTION 2

Error Correcting Codes

1. Communication Errors

In daily communications, errors can and do occur.
Words are misspelled.
Wrong letters are typed.
Data is damaged, like by scratches on a CD.

3. Error Correcting Codes

Suppose we wish to encode a half byte (4 bits) so that if an error occurs in transmitting this data we could detect the error.

One simple minded way would be to just repeat the 4 bits.

For example, we could encode $m = 1011$ as

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

where the two rows are identical.

5. An Error Correcting Code

Goal: Use the parity check bit

- to **detect** errors and also
- to **correct** them.

Our Code is called the **Tic-Tac-Toe Code**

2. Consequences

Usually the errors can be identified and corrected.

Like when Google says “Did you mean . . .”

But even a one-letter change can have serious consequences.

Like when **NOT** is changed to **NOW**.

4. Finding the Error

If the message $M = \begin{bmatrix} 1 & 0 & 1 & 1 \\ 1 & 0 & 1 & 1 \end{bmatrix}$, is sent over a communications channel and the message

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

is received, we know that an error occurred in the third bit.

But we don't know whether the third bit was originally a 0 and the first line is wrong, or whether the third bit was originally a 1 and the second line is wrong.

You can **find** the error— you just can't **fix** it.

6. The Tic-tac-toe Code

Store the original message $m = b_1b_2b_3b_4$ in a 3 by 3 tic-tac-toe board:

$$M = \left[\begin{array}{cc|c} b_1 & b_2 & d_1 \\ b_3 & b_4 & d_2 \\ c_1 & c_2 & c_3 \end{array} \right].$$

- d_1 is the parity check bit for row 1
- d_2 is the parity check bit for row 2
- c_1 is the parity check bit for column 1
- c_2 is the parity check bit for column 2
- c_3 in the lower right hand corner (row 3, col 3) is the parity check bit for row 3 or for column 3. You get the same answer either way. Why?

7. Example

The message $m = 1011$ is encoded:

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

Note: the numbers of 1's in each column and each row is always even.

If the receiver gets a message with an odd number of 1's in a row or column, she knows an error occurred somewhere in that row or column.

8. Example Cont'd

Suppose she receives

$$M' = \begin{bmatrix} 1 & 0 & 1 \\ 0 & 1 & 0 \\ 0 & 1 & 1 \end{bmatrix}.$$

- Both the second row and the first column contain a single 1.
- Since the number 1 is odd, an error must have occurred in
row 2, column 1.
- Not only have we detected an error,
- but we know where it occurred!

9. Correcting the Error

error in row 2

1	0	1
0	1	0
0	1	1

error in
col 1

Change the 0 in row 2, column 1 to a 1.

We thus recover the correct transmitted message M .

To obtain the original four-bit message m , simply peel off the 5 parity check bits from M :

$$m = 1\ 0\ 1\ 1$$

11. Exercise 2

Decode the following messages using the Tic-tac-toe code:

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

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

10. Exercise 1

Encode the following messages in the Tic-tac-toe code:

0011

0110

1111.

12. Question

Using the Tic-tac-toe code, can you detect two errors?

Are you able to correct both errors?

Given examples to support your answers.