A Short Tutorial
on
Single-bit Error Correction using Hamming Code

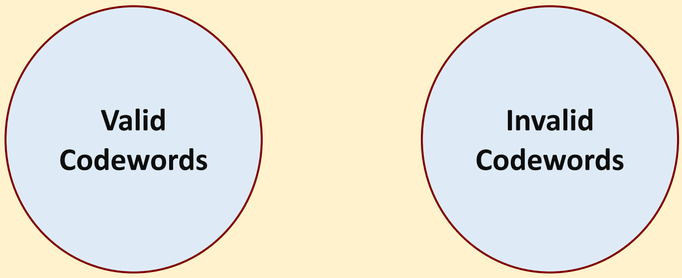
1

Error Detection and Correction

- **Basic concept:**
  - Extra bits are added to the data bits that we want to represent, to get *codewords*.
  - Codewords are divided into two categories: *valid codewords* and *invalid codewords*.
  - A bit error changes a *valid codeword* to an *invalid codeword*.
- **Definition:**
  - The *distance* between two codewords C_i and C_j, denoted by *dist* (C_i, C_j), denotes the number of bit positions in which the codewords differ.
  - Example: distance between codewords 01001 and 11100 is 3.

2

2



- We can check whether a given codeword is *valid* or *invalid*.
- For detecting single bit error, the distance between any pair of valid codewords must be *at least 2*.
  - For detecting *k* errors, the distance must be at least (*k*+1).
- For *single error correction*, the distance must be *at least 3*.

Parity Code for Error Detection

- The parity of a given binary word is defined by whether the number of 1's is *odd* or *even*.
  - We can add an extra bit, called *parity bit*, to make the number if 1's in every valid codeword *even* (say).
  - An example for 3-bit words is shown.
  - Any odd number of bit errors can be detected.

3-bit Binary Word			Parity Bit
0	0	0	**0**
0	0	1	**1**
0	1	0	**1**
0	1	1	**0**
1	0	0	**1**
1	0	1	**0**
1	1	0	**0**
1	1	1	**1**

Hamming Code for Single Error Correction

- A code with minimum distance of $2k+1$ can detect up to k bit errors.
- **Basic principle of Hamming code**:
  - To each group of m information bits, k parity bits are added to form a $(m+k)$-bit code.
  - Each of the $(m+k)$ bit locations in a codeword is assigned a *value*.
    - 1 to the MSB, 2 to the second MSB, …, (m+k) to the LSB.
  - k must satisfy the inequality: $2^k \geq m + k + 1$ (e.g. *for m=4, k will be 3*)
  - The parity check bits are assigned position numbers that are powers of 2 (that is, 1, 2, 4, …).
  - The parity check bits are computed based on some well-defined formula.

5

5

Example: Hamming Code for BCD (m=4)

	b1	b2	b3	b4	b5	b6	b7
Digit	p1	p2	m1	p3	m2	m3	m4
0	0	0	0	0	0	0	0
1	1	1	0	1	0	0	1
2	0	1	0	1	0	1	0
3	1	0	0	0	0	1	1
4	1	0	0	1	1	0	0
5	0	1	0	0	1	0	1
6	1	1	0	0	1	1	0
7	0	0	0	1	1	1	1
8	1	1	1	0	0	0	0
9	0	0	1	1	0	0	1

$p1 = m1 \oplus m2 \oplus m4$ (1, 3, 5, 7)

$p2 = m1 \oplus m3 \oplus m4$ (2, 3, 6, 7)

$p3 = m2 \oplus m3 \oplus m4$ (4, 5, 6, 7)

	1	2	3	4	5	6	7
p3	0	0	0	1	1	1	1
p2	0	1	1	0	0	1	1
p1	1	0	1	0	1	0	1

Can be extended to 4-bit binary code as well.

6

6

How to correct errors?

- Calculate the three check bits:

  $c1 = b1 \oplus b3 \oplus b5 \oplus b7$

  $c2 = b2 \oplus b3 \oplus b6 \oplus b7$

  $c3 = b4 \oplus b5 \oplus b6 \oplus b7$
- If $c_3c_2c_1 = 000$, there is *no error*.
- Else, the *bit position of the error* is given by $c_3c_2c_1$.

7

7

Important

- *Try it out by hand on a couple of examples before starting with the assignment.*

8