Digital Communication Systems EES 452

Asst. Prof. Dr. Prapun Suksompong

prapun@siit.tu.ac.th

5.1 Binary Linear Block Codes

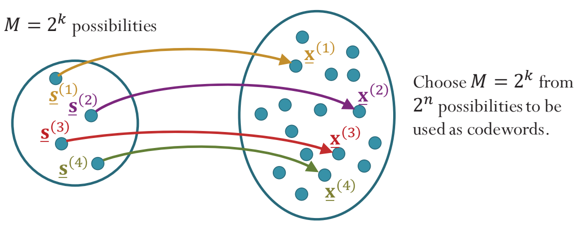
Error Control, Error Detection, Error Correction

Two types of **error control**:

1. **error detection**
2. **error correction**

Error Detection

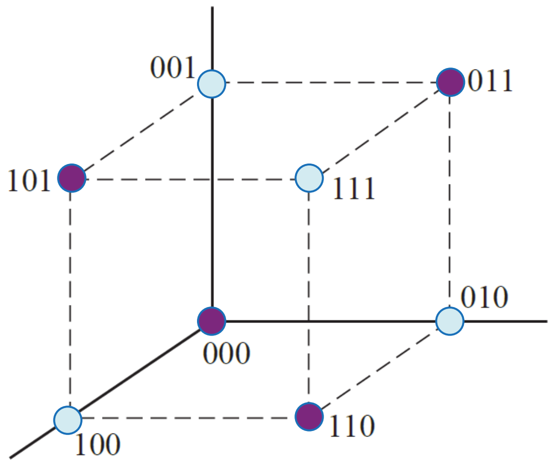
- **Error detection**: the determination of whether errors are present in a received word
  - usually by checking whether the received word is one of the valid codewords.



- When a two-way channel exists between source and destination, the receiver can request **retransmission** of information containing detected errors.
  - This error-control strategy is called **automatic-repeat-request (ARQ)**.
- An error pattern is **undetectable** if and only if it causes the received word to be a valid codeword other than that which was transmitted.
  - Ex: In single-parity-check code, error will be undetectable when the number of bits in error is even.

Example: (3,2) Single-parity-check code

- If we receive 001, 111, 010, or 100, we know that something went wrong in the transmission.
- Suppose we transmitted 101 but the error pattern is 110.
  - The received vector is 011
  - 011 is still a valid codeword.
  - The error is undetectable.



Error Correction

- In **FEC** (**forward error correction**) system, when the decoder detects error, the arithmetic or algebraic **structure** of the code is used to determine which of the valid codewords was transmitted.

- It is possible for a detectable error pattern to cause the decoder to select a codeword other than that which was actually transmitted. The decoder is then said to have committed a **decoding error**.

Square array for error correction by parity checking.

$\underline{\mathbf{b}} = [b_1, b_2, \ldots, b_9]$

- The codeword is formed by arranging k message bits in a square array whose rows *and* columns are checked by $2\sqrt{k}$ parity bits.
- A transmission error in one message bit causes a row and column parity failure with the error at the intersection, so single errors can be corrected.

b_1	b_2	b_3	p_1
b_4	b_5	b_6	p_2
b_7	b_8	b_9	p_3
p_4	p_5	p_6	

$\underline{\mathbf{x}} = [b_1, b_2, \ldots, b_9, p_1, p_2, \ldots, p_6]$

[Carlson & Crilly, p 594]

Example: square array

- $k = 9$
- $2\sqrt{9} = 6$ parity bits.

b_1	b_2	b_3	p_1
b_4	b_5	b_6	p_2
b_7	b_8	b_9	p_3
p_4	p_5	p_6	

$\underline{\mathbf{b}} = [b_1, b_2, \ldots, b_9]$

$= 101110100$

$\underline{\mathbf{x}} = [b_1, b_2, \ldots, b_9, p_1, p_2, \ldots, p_6]$

$= 101110100$_ _ _ _ _ _

1	0	1	
1	1	0	
1	0	0	

$\underline{y} = 100110100001111$

[Carlson & Crilly, p 594]

Review: Even Parity

- A binary vector (or a collection of 1s and 0s) has **even parity** if and only if the number of 1s in there is even.
  - Suppose we are given the values of all the bits except one bit.
    - We can force the vector to have even parity by setting the value of the remaining bit to be the sum of the other bits.

Single-parity-check code

[1 0 1 1 0 _]

Square array

1	0	1	_
0	1	1	_
0	0	1	_
_	_	_	

Digital Communication Systems EES 452

Asst. Prof. Dr. Prapun Suksompong

prapun@siit.tu.ac.th

5.1 Binary Linear Block Codes

Introduction to Minimum Distance

Minimum Distance ($d_{\min}$)

The **minimum distance** ($d_{\min}$) of a block code is the minimum Hamming distance between all pairs of distinct codewords.

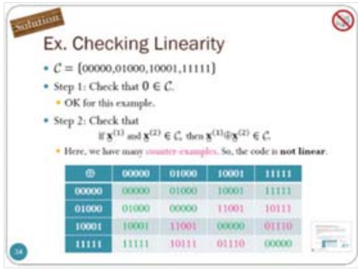
- Ex.:

A channel encoder map blocks of two bits to five-bit (channel) codewords. The four possible codewords are 00000, 01000, 10001, and 11111. A codeword is transmitted over the BSC with crossover probability $p = 0.1$.

(a) What is the minimum (Hamming) distance d_{min} among the codewords?

$d_{\min} = 1$

d	00000	01000	10001	11111
00000		1	2	5
01000			3	4
10001				3
11111				



- Ex. Repetition code:

MATLAB: Distance Matrix and $d_{\min}$

```
function D = distAll(C)

M = size(C,1);
D = zeros(M,M);
for i = 1:M-1
    for j = (i+1):M
        D(i,j) = sum(mod(C(i,:)+C(j,:),2));
    end
end
D = D+D';
```

This can be used to find $d_{\min}$ for all block codes. There is no assumption about linearity of the code. Soon, we will see that we can simplify the calculation when the code is known to be linear.

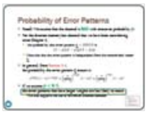
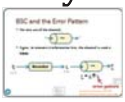
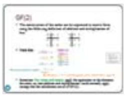
```
function dmin = dmin_block(C)
D = distAll(C);
Dn0 = D(D>0);
dmin = min(Dn0);
```

```
>> C=[0 0 0 0 0; 0 1 0 0 0; ...
      1 0 0 0 1; 1 1 1 1 1];
>> distAll(C)
ans =
     0     1     2     5
     1     0     3     4
     2     3     0     3
     5     4     3     0

>> dmin = dmin_block(C)
dmin =
     1
```

Weight and Distance

- The **weight** of a vector is the number of nonzero coordinates in the vector.
  - The weight of a vector $\underline{\mathbf{x}}$ is commonly written as $w(\underline{\mathbf{x}})$.
  - Ex. $w(010111) =$
  - For BSC with cross-over probability $p < 0.5$, error pattern with smaller weights (less #1s) are more likely to occur.
- The **Hamming distance** between two n-bit blocks is the number of coordinates in which the two blocks differ.
  - Ex. $d(010111,011011) =$
  - Note:
    - The Hamming distance between any two vectors equals the weight of their sum.
    - The Hamming distance between the transmitted codeword $\underline{\mathbf{x}}$ and the received vector $\underline{\mathbf{y}}$ is the same as the weight of the corresponding error pattern $\underline{\mathbf{e}}$.


$d_{\min}$ for linear block code

- For any linear block code, the **minimum distance** ($d_{\min}$) can be found from the minimum weight of its nonzero codewords.
  - So, instead of checking $\binom{2^k}{2}$ pairs, simply check the weight of the 2^k codewords.

```
function dmin =  dmin_linear(C)
w = sum(C,2);
w = w([w>0]);
dmin = min(w);
```

Proof

Because the code is linear, for any two distinct codewords $\underline{\mathbf{c}}^{(1)}$ and $\underline{\mathbf{c}}^{(2)}$, we know that $\underline{\mathbf{c}}^{(1)} \oplus \underline{\mathbf{c}}^{(2)} \in \mathcal{C}$; that is $\underline{\mathbf{c}}^{(1)} \oplus \underline{\mathbf{c}}^{(2)} = \underline{\mathbf{c}}$ for some nonzero $\underline{\mathbf{c}} \in \mathcal{C}$. Therefore,

$$d\left(\underline{\mathbf{c}}^{(1)}, \underline{\mathbf{c}}^{(2)}\right) = w\left(\underline{\mathbf{c}}^{(1)} \oplus \underline{\mathbf{c}}^{(2)}\right) = w\left(\underline{\mathbf{c}}\right) \text{ for some nonzero } \underline{\mathbf{c}} \in \mathcal{C}.$$

This implies

$$\min_{\substack{\underline{\mathbf{c}}^{(1)}, \underline{\mathbf{c}}^{(2)} \in \mathcal{C} \\ \underline{\mathbf{c}}^{(1)} \neq \underline{\mathbf{c}}^{(2)}}} d\left(\underline{\mathbf{c}}^{(1)}, \underline{\mathbf{c}}^{(2)}\right) \geq \min_{\substack{\underline{\mathbf{c}} \in \mathcal{C} \\ \underline{\mathbf{c}} \neq \underline{\mathbf{0}}}} w\left(\underline{\mathbf{c}}\right).$$

Note that inequality is used here because we did not show that $\underline{\mathbf{c}}^{(1)} \oplus \underline{\mathbf{c}}^{(2)}$ can produce all possible nonzero $\underline{\mathbf{c}} \in \mathcal{C}$.

$$\min_{\substack{\underline{\mathbf{c}}^{(1)}, \underline{\mathbf{c}}^{(2)} \in \mathcal{C} \\ \underline{\mathbf{c}}^{(1)} \neq \underline{\mathbf{c}}^{(2)}}} d\left(\underline{\mathbf{c}}^{(1)}, \underline{\mathbf{c}}^{(2)}\right) = \min_{\substack{\underline{\mathbf{c}} \in \mathcal{C} \\ \underline{\mathbf{c}} \neq \underline{\mathbf{0}}}} w\left(\underline{\mathbf{c}}\right)$$

Next, for any nonzero $\underline{\mathbf{c}} \in \mathcal{C}$, note that

$$d\left(\underline{\mathbf{c}}, \underline{\mathbf{0}}\right) = w\left(\underline{\mathbf{c}} \oplus \underline{\mathbf{0}}\right) = w\left(\underline{\mathbf{c}}\right).$$

Note that $\underline{\mathbf{c}}, \underline{\mathbf{0}}$ is just one possible pair of two distinct codewords. This implies

$$\min_{\substack{\underline{\mathbf{c}}^{(1)}, \underline{\mathbf{c}}^{(2)} \in \mathcal{C} \\ \underline{\mathbf{c}}^{(1)} \neq \underline{\mathbf{c}}^{(2)}}} d\left(\underline{\mathbf{c}}^{(1)}, \underline{\mathbf{c}}^{(2)}\right) \leq \min_{\substack{\underline{\mathbf{c}} \in \mathcal{C} \\ \underline{\mathbf{c}} \neq \underline{\mathbf{0}}}} w\left(\underline{\mathbf{c}}\right).$$

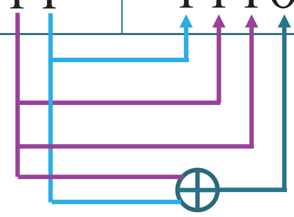
Example

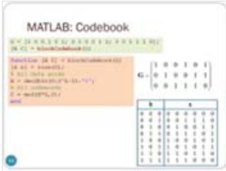
$$\mathbf{G} = \begin{bmatrix} 0 & 1 & 1 & 1 \\ 1 & 0 & 0 & 1 \end{bmatrix}$$

$$\underline{\mathbf{x}} = \underline{\mathbf{b}}\mathbf{G} = \begin{bmatrix} b_1 & b_2 \end{bmatrix} \begin{bmatrix} 0 & 1 & 1 & 1 \\ 1 & 0 & 0 & 1 \end{bmatrix}$$

$$= \begin{bmatrix} b_2 & b_1 & b_1 & b_1 \oplus b_2 \end{bmatrix}$$

$\underline{\mathbf{b}}$	$\underline{\mathbf{x}}$
00	0000
01	1001
10	0111
11	1110



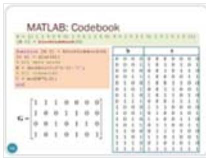


Example

$$\mathbf{G} = \begin{pmatrix} 1 & 0 & 0 & 1 & 0 & 1 \\ 0 & 1 & 0 & 0 & 1 & 1 \\ 0 & 0 & 1 & 1 & 1 & 0 \end{pmatrix}$$

$\underline{\mathbf{b}}$			$\underline{\mathbf{x}}$						$w(\underline{\mathbf{x}})$
0	0	0	0	0	0	0	0	0	0
0	0	1	0	0	1	1	1	0	
0	1	0	0	1	0	0	1	1	
0	1	1	0	1	1	1	0	1	
1	0	0	1	0	0	1	0	1	
1	0	1	1	0	1	0	1	1	4
1	1	0	1	1	0	1	1	0	4
1	1	1	1	1	1	0	0	0	3

```
>> G = [1 0 0 1 0 1; 0 1 0 0 1 1; 0 0 1 1 1 0];
>> [B C] = blockCodebook(G);
>> dmin =  dmin_block(C)
dmin =
     3
>> dmin =  dmin_linear(C)
dmin =
     3
```

Example

$$\mathbf{G} = \begin{bmatrix} 1 & 1 & 1 & 0 & 0 & 0 & 0 \\ 1 & 0 & 0 & 1 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 & 1 & 1 & 0 \\ 1 & 0 & 1 & 0 & 1 & 0 & 1 \end{bmatrix}$$

```
>> G = [1 1 1 0 0 0 0; 1 0 0 1 1 0 0;...
        0 0 1 0 1 1 0; 1 0 1 0 1 0 1];
>> [B C] = blockCodebook(G);
>> dmin =  dmin_linear(C)
dmin =
     3
>> dmin =  dmin_block(C)
dmin =
     3
```

$\underline{\mathbf{b}}$				$\underline{\mathbf{x}}$							$w(\underline{\mathbf{x}})$
0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	1	1	0	1	0	1	0	1	4
0	0	1	0	0	0	1	0	1	1	0	3
0	0	1	1	1	0	0	0	0	1	1	3
0	1	0	0	1	0	0	1	1	0	0	3
0	1	0	1	0	0	1	1	0	0	1	3
0	1	1	0	1	0	1	1	0	1	0	4
0	1	1	1	0	0	0	1	1	1	1	4
1	0	0	0	1	1	1	0	0	0	0	3
1	0	0	1	0	1	0	0	1	0	1	3
1	0	1	0	1	1	0	0	1	1	0	4
1	0	1	1	0	1	1	0	0	1	1	4
1	1	0	0	0	1	1	1	1	0	0	4
1	1	0	1	1	1	0	1	0	0	1	4
1	1	1	0	0	1	0	1	0	1	0	3
1	1	1	1	1	1	1	1	1	1	1	7

Digital Communication Systems
EES 452

Asst. Prof. Dr. Prapun Suksompong
prapun@siit.tu.ac.th

5.1 Binary Linear Block Codes

Probability of Error Patterns and Minimum Distance Decoder

Probability of Error Patterns

- Recall: We assume that the channel is **BSC** with crossover probability $\boldsymbol{p}$.
- For the discrete memoryless channel that we have been considering since Chapter 3,
  - the probability that error pattern $\underline{\mathbf{e}} = 00101$ is
    $$(1-p)(1-p)p(1-p)p.$$
  - Note also that the error pattern is independent from the transmitted vector $\underline{\mathbf{x}}$
- In general, from Section 3.4,
  the probability the error pattern $\underline{\mathbf{e}}$ occurs is
  $$p^{d(\underline{\mathbf{x}},\underline{\mathbf{y}})}(1-p)^{n-d(\underline{\mathbf{x}},\underline{\mathbf{y}})} = \left(\frac{p}{1-p}\right)^{d(\underline{\mathbf{x}},\underline{\mathbf{y}})}(1-p)^n = \left(\frac{p}{1-p}\right)^{w(\underline{\mathbf{e}})}(1-p)^n$$
- If we assume $\boldsymbol{p < 0.5}$,
  the error patterns that have larger weights are less likely to occur.
  - This also supports the use of minimum distance decoder.

Digital Communication Systems EES 452

Asst. Prof. Dr. Prapun Suksompong
prapun@siit.tu.ac.th

5.1 Binary Linear Block Codes

Properties of $d_{\min}$

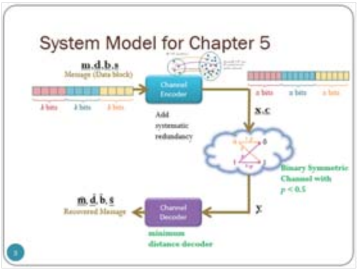
$d_{\min}$: two important facts

- For any linear block code, the **minimum distance** ($d_{\min}$) can be found from the minimum weight of its nonzero codewords.
  - So, instead of checking $\binom{2^k}{2}$ pairs, simply check the weight of the 2^k codewords.

- A code with minimum distance $d_{\min}$ can
  - detect all error patterns of weight w ≤ $d_{\min}$-1.
  - correct all error patterns of weight w ≤ $\left\lfloor \frac{d_{\min}-1}{2} \right\rfloor$.

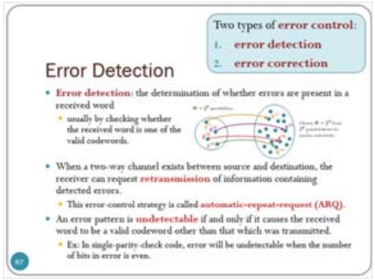
the floor function

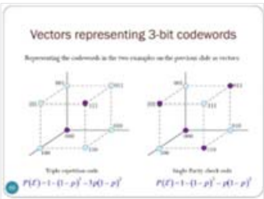
Visual Interpretation of $d_{\min}$

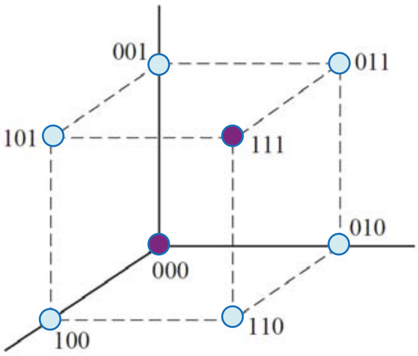


Recall: Codebook construction

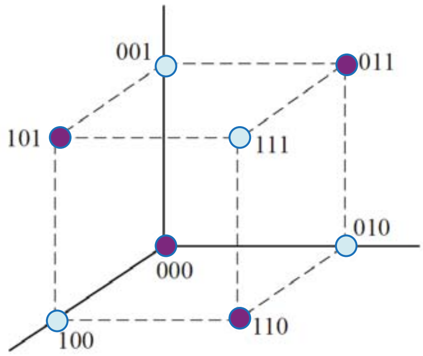
Choose $M = 2^k$ from 2^n possibilities to be used as codewords.





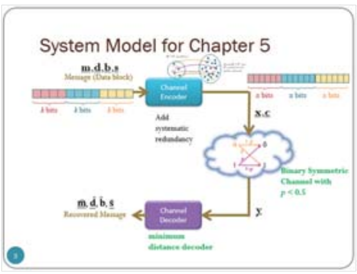


Triple-repetition code



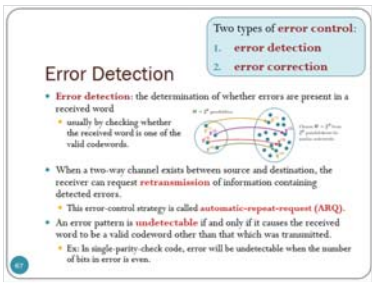
Single-Parity-check code

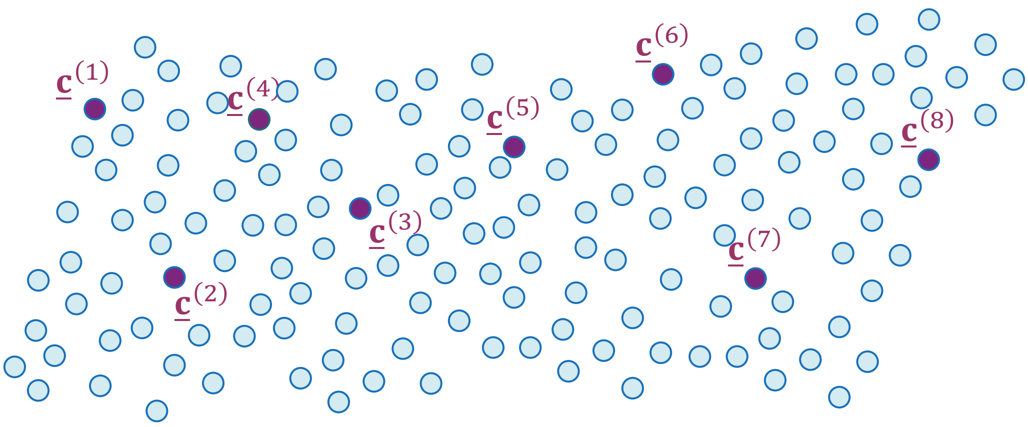
Visual Interpretation of $d_{\min}$



Recall: Codebook construction

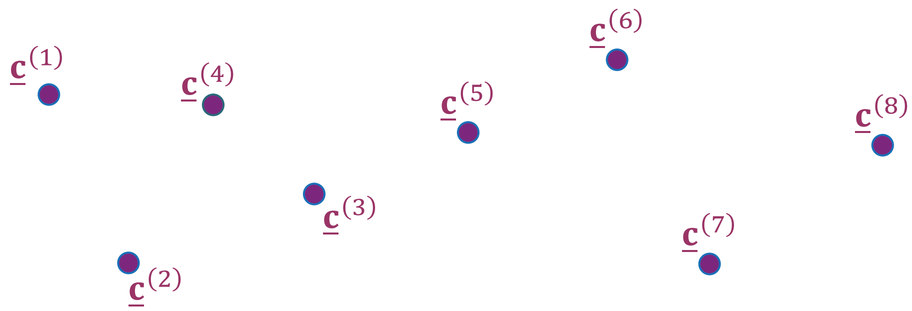
Choose $M = 2^k$ from 2^n possibilities to be used as codewords.





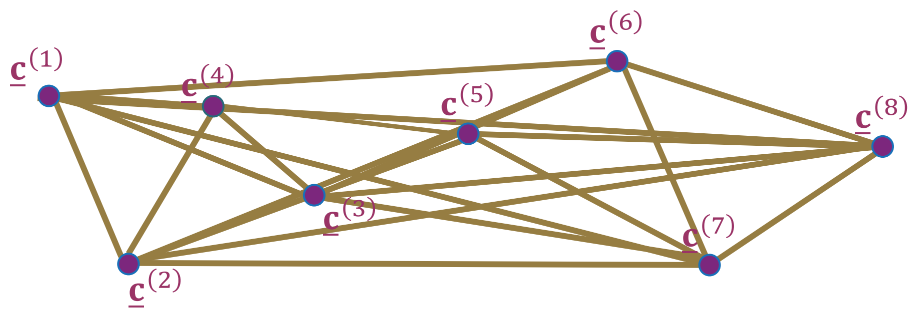
Visual Interpretation of d_{min}

- Consider all the (valid) codewords (in the codebook).



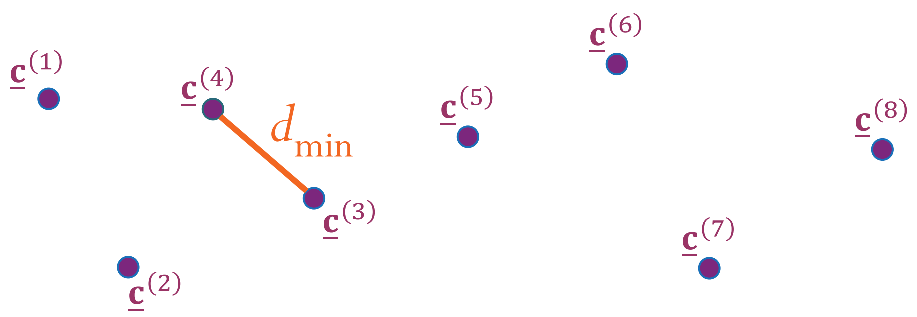
Visual Interpretation of d_{min}

- Consider all the (valid) codewords (in the codebook).
- We can find the distances between them.



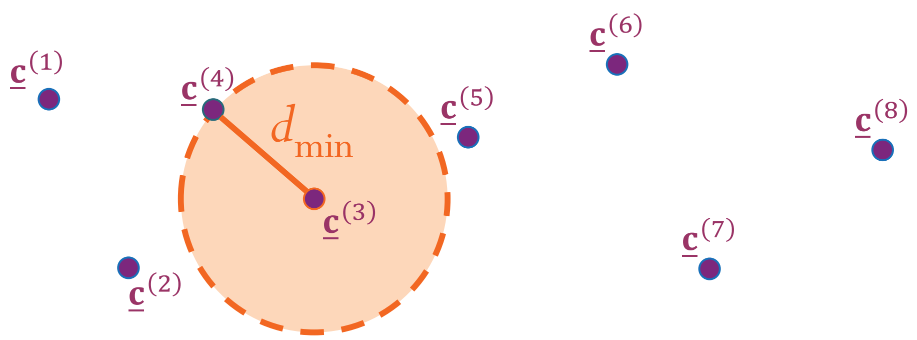
Visual Interpretation of $d_{\min}$

- Consider all the (valid) codewords (in the codebook).
- We can find the distances between them.
- We can then find $d_{\min}$.



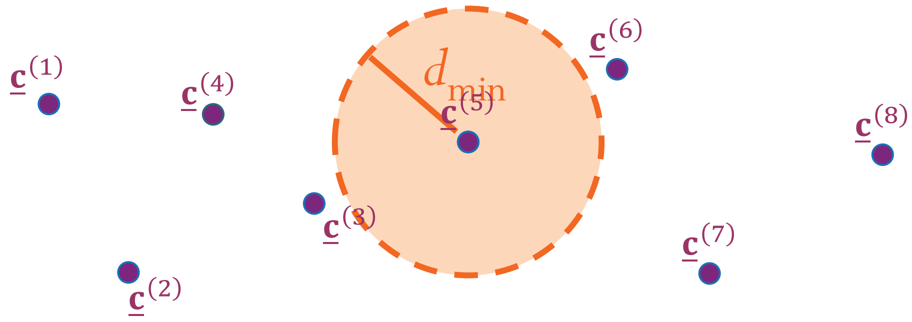
Visual Interpretation of $d_{\min}$

- When we draw a circle (sphere, hypersphere) of radius $d_{\min}$ around any codeword, we know that there can not be another codeword inside this circle.
- The closest codeword is at least $d_{\min}$ away.



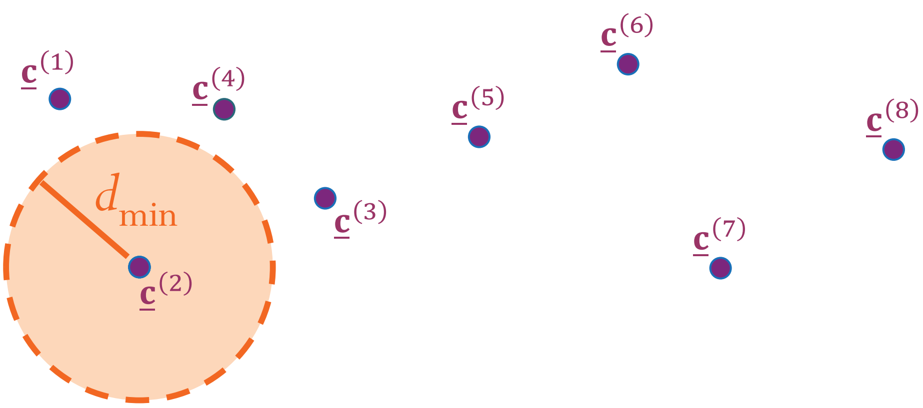
Visual Interpretation of $d_{\min}$

- When we draw a circle (sphere, hypersphere) of radius $d_{\min}$ around any codeword, we know that there can not be another codeword inside this circle.
- The closest codeword is at least $d_{\min}$ away.



Visual Interpretation of $d_{\min}$

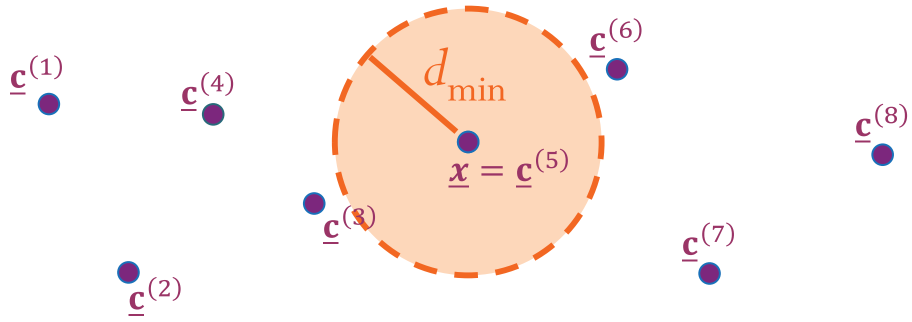
- When we draw a circle (sphere, hypersphere) of radius $d_{\min}$ around any codeword, we know that there can not be another codeword inside this circle.
- The closest codeword is at least $d_{\min}$ away.



$d_{\min}$ and Error Detection

- Suppose codeword $\underline{\mathbf{c}}^{(5)}$ is chosen to be transmitted; that is

$$\underline{\boldsymbol{x}} = \underline{\mathbf{c}}^{(5)}.$$



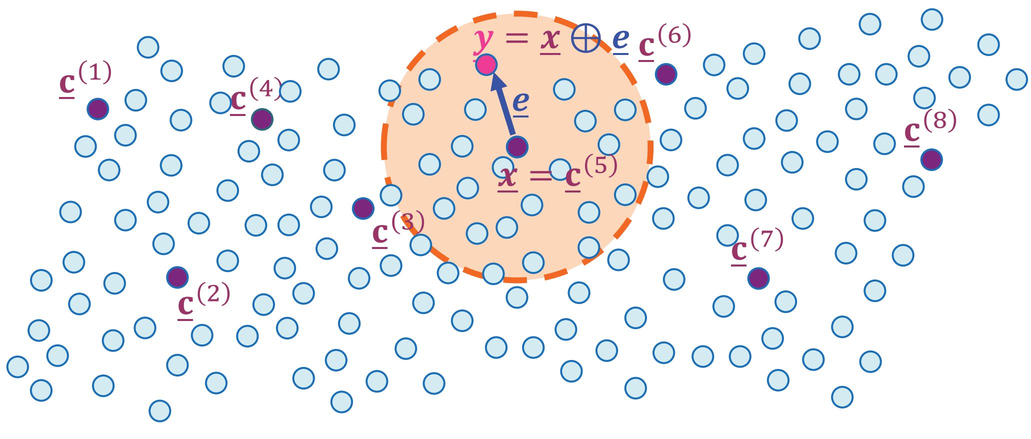
$d_{\min}$ and Error Detection

- Suppose codeword $\underline{\mathbf{c}}^{(5)}$ is chosen to be transmitted; that is

$$\underline{\boldsymbol{x}} = \underline{\mathbf{c}}^{(5)}.$$

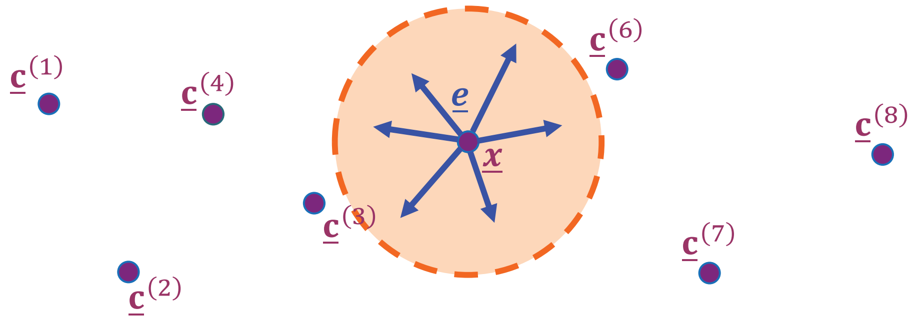
- The received vector $\underline{\boldsymbol{y}}$ can be calculated from

$$\underline{\boldsymbol{y}} = \underline{\boldsymbol{x}} \oplus \underline{\boldsymbol{e}}.$$



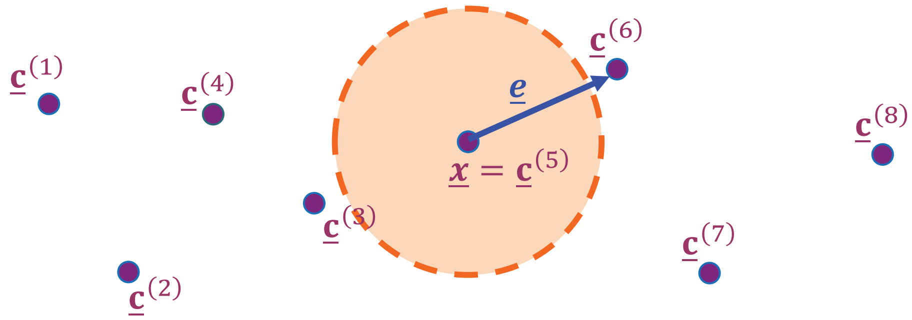
d_{min} and Error Detection

- When $d_{\text{min}} > w$, there is no way that w errors can change a valid codeword into another valid codeword.



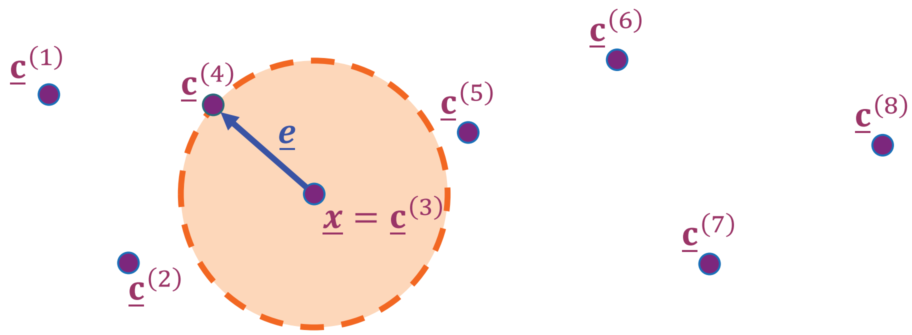
d_{min} and Error Detection

- When $d_{\text{min}} < w$, it is possible that w errors can change a valid codeword into another valid codeword.



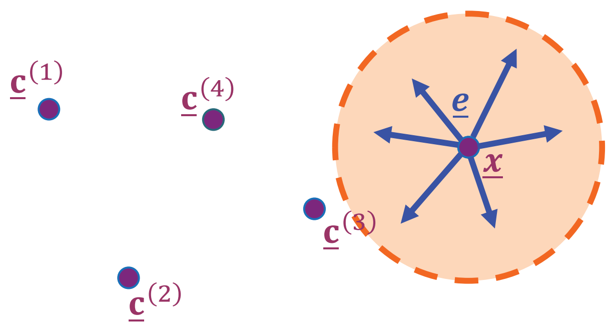
$d_{\min}$ and Error Detection

- For some codewords,
  when $d_{\min} = w$, it is possible that w errors can change a valid codeword into another valid codeword.



$d_{\min}$ and Error Detection

- To be able to **detect** *all* w-bit errors, we need $d_{\min} \geq w + 1$.
  - With such a code there is no way that w errors can change a valid codeword into another valid codeword.
  - When the receiver observes an illegal codeword, it can tell that a transmission error has occurred.

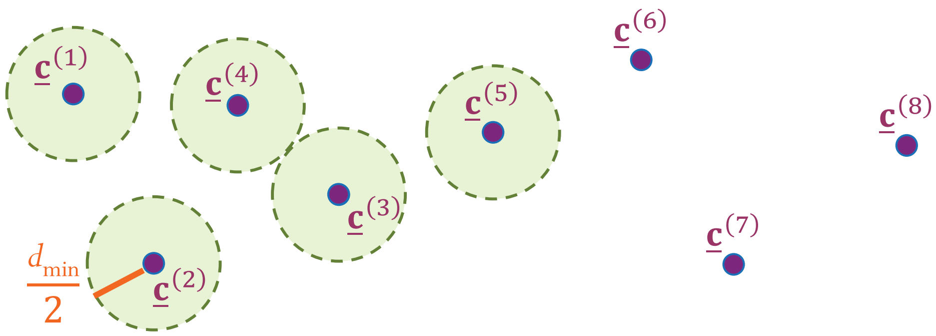


When $d_{\min} > w$, there is no way that w errors can change a valid codeword into another valid codeword.

When $d_{\min} \leq w$, it is possible that w errors can change a valid codeword into another valid codeword.

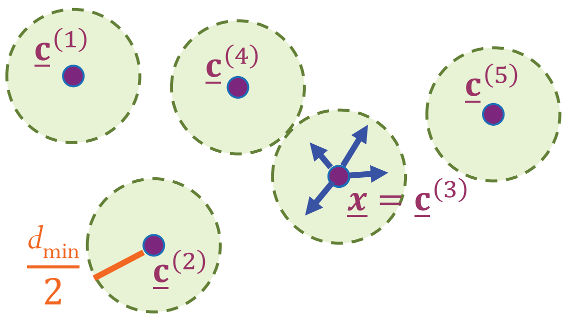
d_{min} and Error Correction

- To be able to **correct** *all* w-bit errors, we need $d_{\text{min}} \geq 2w + 1$.
  - This way, the legal codewords are so far apart that even with w changes, the original codeword is still ***closer*** than any other codeword.



d_{min} is an important quantity

- To be able to **correct** *all* w-bit errors, we need $d_{\text{min}} \geq 2w + 1$.
  - This way, the legal codewords are so far apart that even with w changes, the original codeword is still ***closer*** than any other codeword.



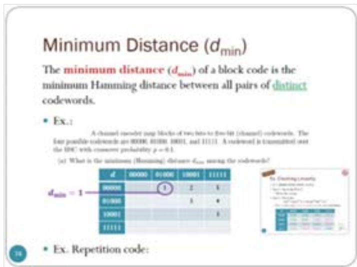
$d_{\min}$: two important facts

- For any linear block code, the **minimum distance** ($d_{\min}$) can be found from the minimum weight of its nonzero codewords.
  - So, instead of checking $\binom{2^k}{2}$ pairs, simply check the weight of the 2^k codewords.

- A code with minimum distance $d_{\min}$ can
  - detect all error patterns of weight $w \le d_{\min}-1$.
  - correct all error patterns of weight $w \le \left\lfloor \frac{d_{\min}-1}{2} \right\rfloor$.

the floor function

Example



Repetition code with $n = 5$

- We have seen that it has $d_{\min} = 5$.
- It can detect (at most) ____ errors.
- It can correct (at most) ____ errors.

Example

Consider the code

$$\mathcal{C} \in \{0000000000,\ 0000011111,\ 1111100000,\ \text{and } 1111111111\}$$

- Is it a linear code?

	$\oplus$	$\underline{\mathbf{c}}^{(1)}$	$\underline{\mathbf{c}}^{(2)}$	$\underline{\mathbf{c}}^{(3)}$	$\underline{\mathbf{c}}^{(4)}$
0000000000	$\underline{\mathbf{c}}^{(1)}$				
0000011111	$\underline{\mathbf{c}}^{(2)}$				
1111100000	$\underline{\mathbf{c}}^{(3)}$				
1111111111	$\underline{\mathbf{c}}^{(4)}$				

- $d_{\min} =$

- It can detect (at most) ____ errors.

- It can correct (at most) ____ errors.