Digital Communication Systems ECS 452

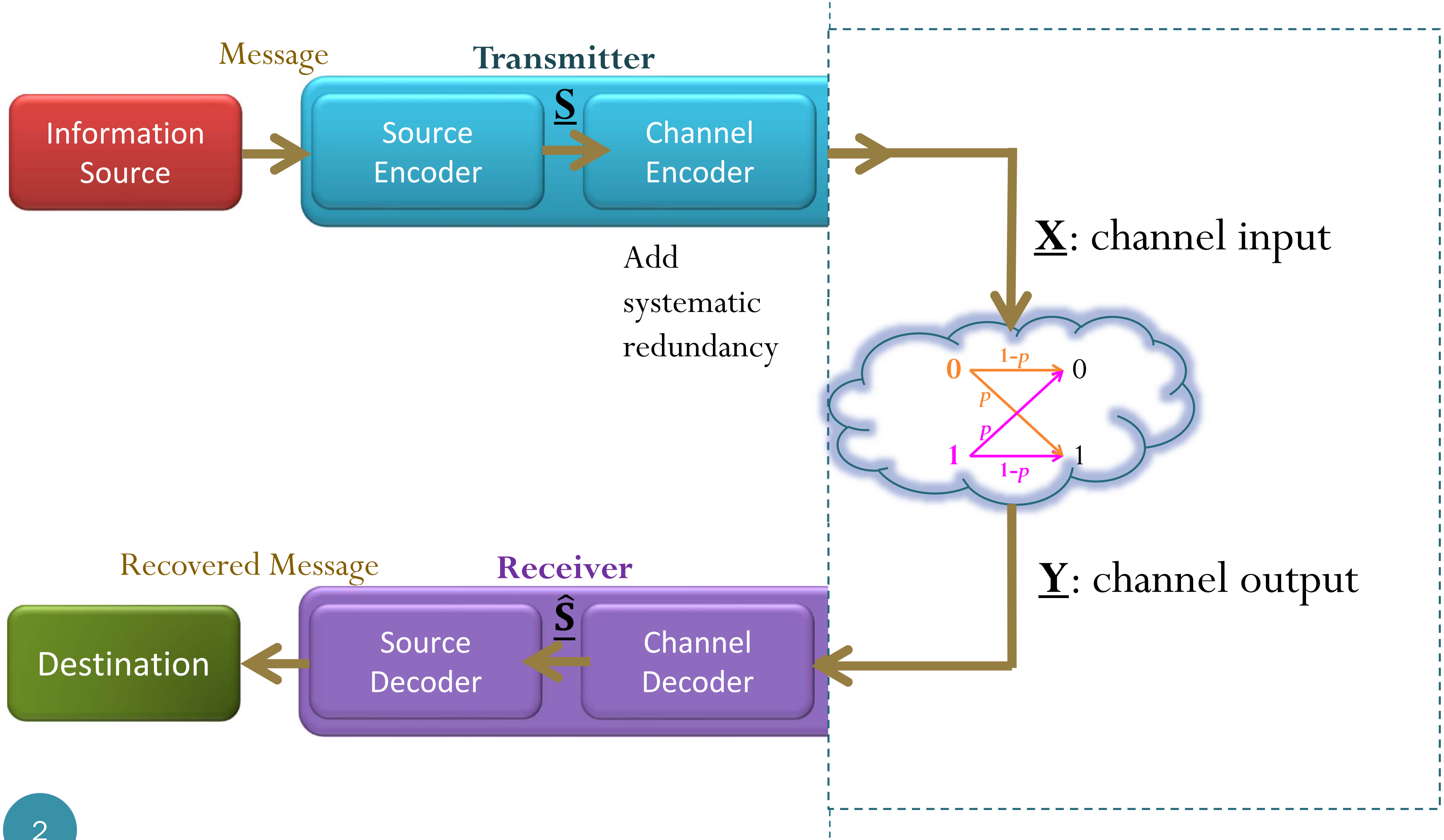
Asst. Prof. Dr. Prapun Suksompong

prapun@siit.tu.ac.th

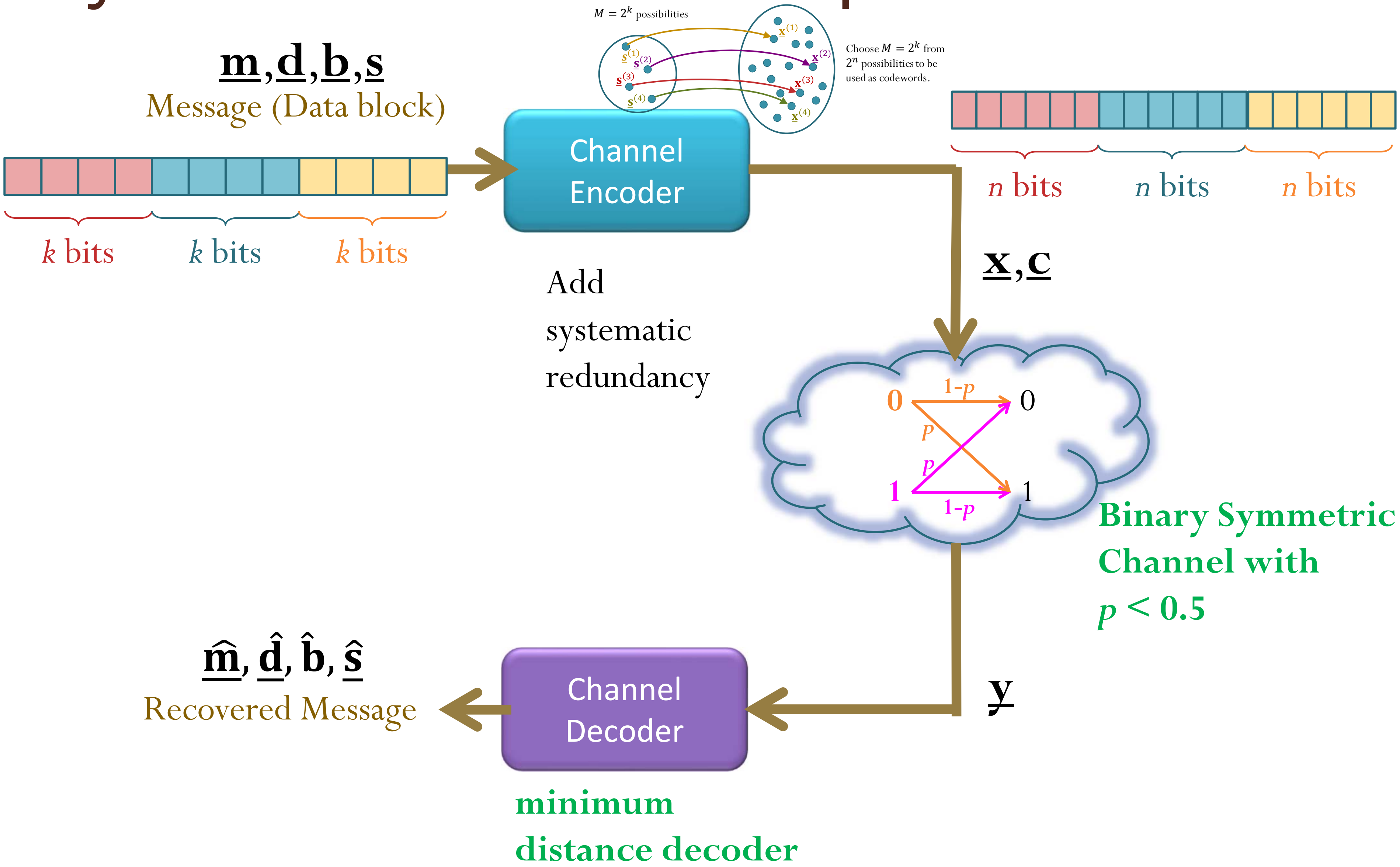
5. Channel Coding

Office Hours:
Check Google Calendar on the course website.
Dr.Prapun's Office:
6th floor of Sirindhralai building, BKD

Review: Channel Encoder and Decoder



System Model for Chapter 5



Vector Notation

$$\begin{pmatrix} v_1 \\ v_2 \\ \vdots \\ v_i \\ \vdots \\ v_n \end{pmatrix}$$

$\vec{\mathbf{0}}, \underline{\mathbf{0}}$: the zero vector (the all-zero vector)

$\vec{\mathbf{1}}, \underline{\mathbf{1}}$: the one vector (the all-one vector)

- $\vec{\mathbf{v}}$: column vector
- $\underline{\mathbf{r}}$: row vector $(r_1, r_2, \dots, r_i, \dots r_n)$
- Subscripts represent element indices inside individual vectors.
  - v_i and r_i refer to the i^{th} elements inside the vectors $\vec{\mathbf{v}}$ and $\underline{\mathbf{r}}$, respectively.
- When we have a list of vectors, we use superscripts in parentheses as indices of vectors.
  - $\vec{\mathbf{v}}^{(1)}, \vec{\mathbf{v}}^{(2)}, \dots, \vec{\mathbf{v}}^{(M)}$ is a list of M column vectors
  - $\underline{\mathbf{r}}^{(1)}, \underline{\mathbf{r}}^{(2)}, \dots, \underline{\mathbf{r}}^{(M)}$ is a list of M row vectors
  - $\vec{\mathbf{v}}^{(i)}$ and $\underline{\mathbf{r}}^{(i)}$ refer to the i^{th} vectors in the corresponding lists.

Channel Decoding

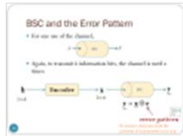
- Recall

MAP decoder is optimal → ML decoder is optimal → Min distance decoder is optimal

Codewords are equally likely

BSC with $p < 0.5$

1. **MAP decoder** is the optimal decoder.
2. When the codewords are equally-likely, the **ML decoder** the same as the MAP decoder; hence it is also **optimal**.
3. When the **crossover probability** of the BSC p is < 0.5,
   ML decoder is the same as the **minimum distance decoder**.



- In this chapter, we assume the use of **minimum distance decoder**.
  - $\hat{\underline{\mathbf{x}}}(\underline{\mathbf{y}}) = \arg\min_{\underline{\mathbf{x}}} d(\underline{\mathbf{x}}, \underline{\mathbf{y}})$
- Also, in this chapter, we will focus
  - less on probabilistic analysis,
  - but more on explicit codes.

Digital Communication Systems ECS 452

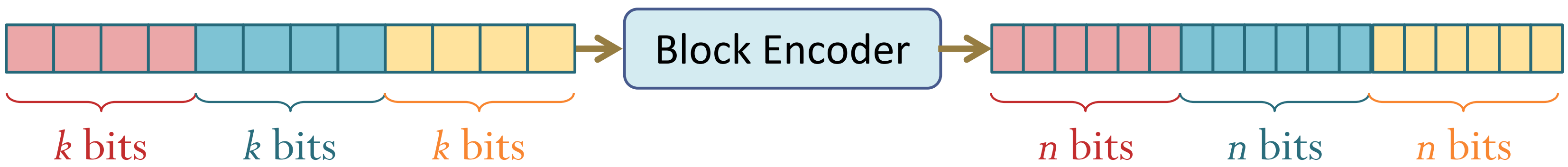
Asst. Prof. Dr. Prapun Suksompong

prapun@siit.tu.ac.th

5.1 Binary Linear Block Codes

Review: Block Encoding

- We mentioned the general form of channel coding over BSC.
- In particular, we looked at the general form of **block codes**.



Code length

"Dimension" of the code

- **(n,k) codes**: n-bit blocks are used to conveys k-info-bit blocks

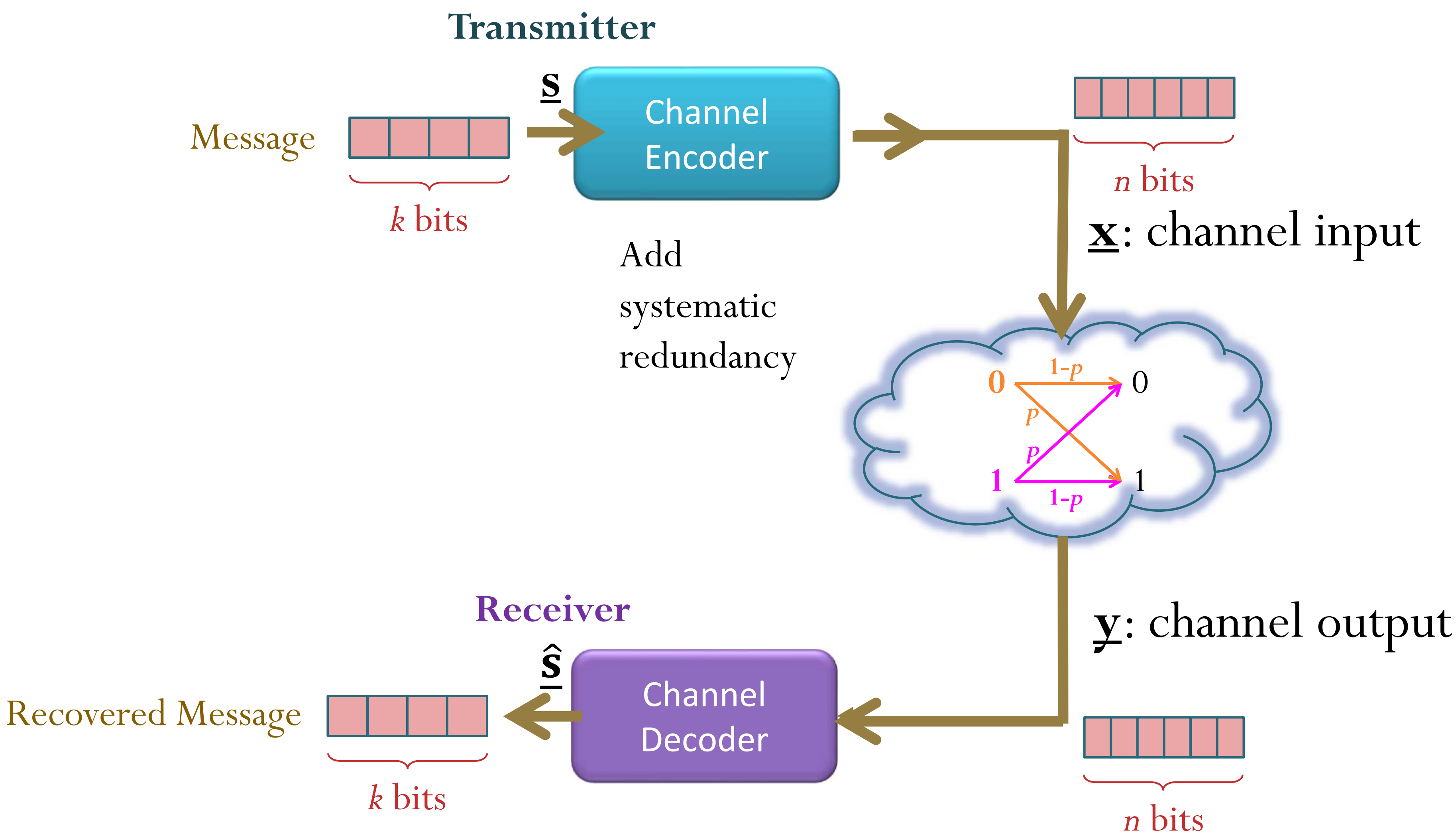
codewords "messages"

- **Assume $n > k$**
- **Rate**: $R = \frac{k}{n}$.

Max. achievable rate

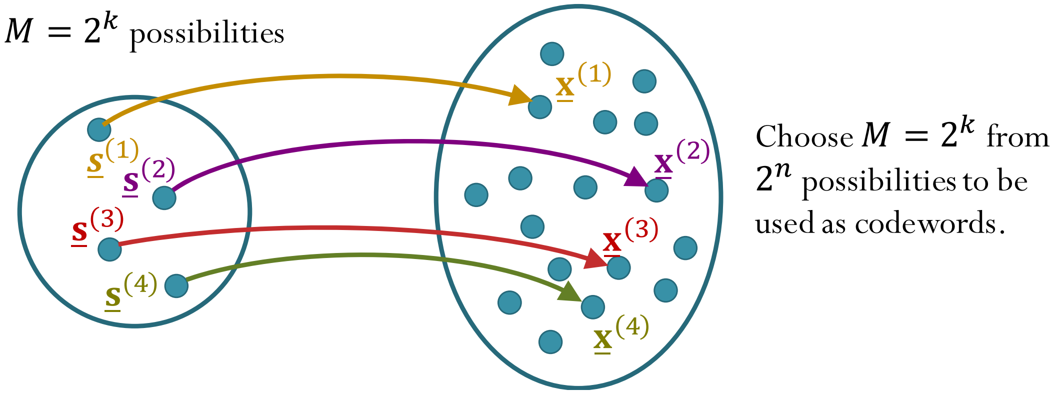
Recall that the capacity of BSC is $C = 1 - H(p)$.
For $p \in (0,1)$, we also have $C \in (0,1)$.
Achievable rate is < 1.

System Model for Section 5.1



$\mathcal{C}$

- $\mathcal{C}$ = the collection of all codewords for the code considered.
- Each n-bit block is selected from $\mathcal{C}$.
- The message (data block) has k bits, so there are 2^k possibilities.
- A reasonable code would not assign
  the same codeword to different messages.
- Therefore, there are 2^k (distinct) codewords in $\mathcal{C}$.



- Ex. Repetition code with $n = 3$

GF(2)

- The construction of the codes can be expressed in matrix form using the following definition of **addition** and **multiplication** of bits:

$$\begin{array}{c|cc} \oplus & 0 & 1 \\ \hline 0 & 0 & 1 \\ 1 & 1 & 0 \end{array} \qquad \begin{array}{c|cc} \bullet & 0 & 1 \\ \hline 0 & 0 & 0 \\ 1 & 0 & 1 \end{array}$$

- These are **modulo-2** addition and **modulo-2** multiplication, respectively.
- The operations are the same as the **exclusive-or** (**XOR**) operation and the **AND** operation.
  - We will simply call them addition and multiplication so that we can use a matrix formalism to define the code.
- The two-element set $\{0, 1\}$ together with this definition of addition and multiplication is a number system called a **finite field** or a **Galois field**, and is denoted by the label **GF(2)**.

Modulo operation

- The **modulo operation** finds the **remainder** after division of one number by another (sometimes called **modulus**).
- Given two positive numbers, a (the **dividend**) and n (the **divisor**),
- a **modulo** n (abbreviated as a **mod** n) is the remainder of the division of a by n.
- "83 mod 6" = 5
- "5 mod 2" = 1
  - In MATLAB, `mod(5,2) = 1`.
- **Congruence relation**
  - $5 \equiv 1 \pmod{2}$

quotient 13
divisor 6) 83 dividend
6
23
18
5 remainder

quotient 2
divisor 2) 5 dividend
4
1 remainder

GF(2) and modulo operation

- Normal addition and multiplication (for 0 and 1):

$+$	0	1
0	0	1
1	1	2

$\times$	0	1
0	0	0
1	0	1

- Addition and multiplication in GF(2):

$\oplus$	0	1
0	0	1
1	1	0

$\bullet$	0	1
0	0	0
1	0	1

GF(2)

- The construction of the codes can be expressed in matrix form using the following definition of addition and multiplication of bits:

$\oplus$	0	1
0	0	1
1	1	0

$\bullet$	0	1
0	0	0
1	0	1

- Note that

$$x \oplus 0 = x$$

$0 \oplus 0 = 0$
$1 \oplus 0 = 1$

$$x \oplus 1 = \overline{x}$$

$0 \oplus 1 = 1$
$1 \oplus 1 = 0$

$$x \oplus x = 0$$

$0 \oplus 0 = 0$
$1 \oplus 1 = 0$

The property above implies $\underbrace{-x}_{\text{By definition, “}-x\text{” is something that, when added with } x\text{, gives } 0.} = x$

- Extension: For vector and matrix, apply the operations to the elements the same way that addition and multiplication would normally apply (except that the calculations are all in GF(2)).

Examples

- Normal vector addition:

$$\begin{array}{r} [1 \quad -1 \quad 2 \quad 1] \\ [-2 \quad 3 \quad 0 \quad 1] \\ \hline = [-1 \quad 2 \quad 2 \quad 2] \end{array} +$$

- Vector addition in GF(2):

$$\begin{array}{r} [1 \quad 0 \quad 1 \quad 1] \\ [0 \quad 1 \quad 0 \quad 1] \\ \hline = [1 \quad 1 \quad 1 \quad 0] \end{array} \oplus$$

Alternatively, one can also apply normal vector addition first, then apply "mod 2" to each element:

$$\begin{array}{r} [1 \quad 0 \quad 1 \quad 1] \\ [0 \quad 1 \quad 0 \quad 1] \\ \hline = [1 \quad 1 \quad 1 \quad 2] \end{array} +$$

$$\Big\downarrow \text{mod } 2$$

$$[1 \quad 1 \quad 1 \quad 0]$$

Examples

- Normal matrix multiplication:

$$(7 \times (-2)) + (4 \times 3) + (3 \times (-7)) = -14 + 12 + (-21)$$

$$\begin{bmatrix} 7 & 4 & 3 \\ 2 & 5 & 6 \\ 1 & 8 & 9 \end{bmatrix} \begin{bmatrix} -2 & 4 \\ 3 & -8 \\ -7 & 6 \end{bmatrix} = \begin{bmatrix} -23 & 14 \\ -31 & 4 \\ -41 & -6 \end{bmatrix}$$

- Matrix multiplication in GF(2):

$$(1 \cdot 1) \oplus (0 \cdot 0) \oplus (1 \cdot 1) = 1 \oplus 0 \oplus 1$$

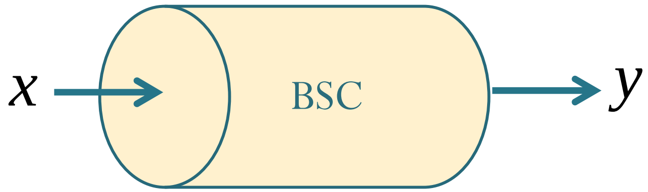
$$\begin{bmatrix} 1 & 0 & 1 \\ 0 & 0 & 1 \\ 1 & 1 & 1 \end{bmatrix} \begin{bmatrix} 1 & 1 \\ 0 & 1 \\ 1 & 0 \end{bmatrix} = \begin{bmatrix} 0 & 1 \\ 1 & 0 \\ 0 & 0 \end{bmatrix}$$

Alternatively, one can also apply normal matrix multiplication first, then apply "mod 2" to each element:

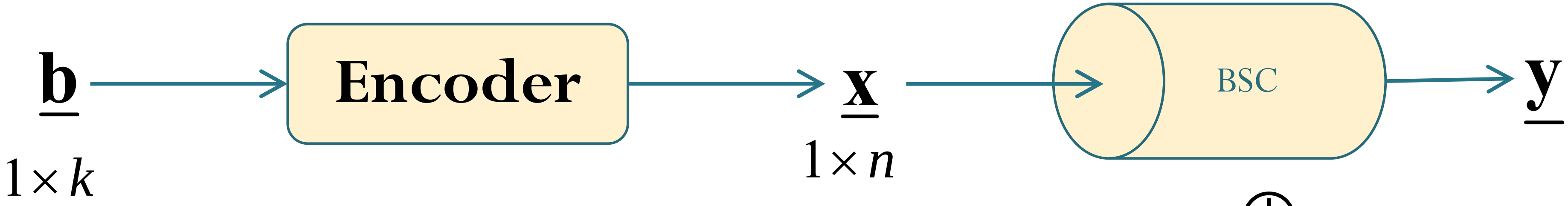
$$\begin{bmatrix} 1 & 0 & 1 \\ 0 & 0 & 1 \\ 1 & 1 & 1 \end{bmatrix} \begin{bmatrix} 1 & 1 \\ 0 & 1 \\ 1 & 0 \end{bmatrix} = \begin{bmatrix} 2 & 1 \\ 1 & 0 \\ 2 & 2 \end{bmatrix} \xrightarrow{\text{mod } 2} \begin{bmatrix} 0 & 1 \\ 1 & 0 \\ 0 & 0 \end{bmatrix}$$

BSC and the Error Pattern

- For one use of the channel,



- Again, to transmit k information bits, the channel is used n times.



$$\underline{\mathbf{y}} = \underline{\mathbf{x}} \oplus \underline{\mathbf{e}}$$

error pattern

Its nonzero elements mark the positions of transmission error in $\underline{\mathbf{y}}$

Additional Properties in GF(2)

- The following statements are equivalent
  1. $a \oplus b = c$
  2. $a \oplus c = b$
  3. $b \oplus c = a$

  Having one of these is the same as having all three of them.

- The following statements are equivalent
  1. $\underline{\mathbf{a}} \oplus \underline{\mathbf{b}} = \underline{\mathbf{c}}$
  2. $\underline{\mathbf{a}} \oplus \underline{\mathbf{c}} = \underline{\mathbf{b}}$
  3. $\underline{\mathbf{b}} \oplus \underline{\mathbf{c}} = \underline{\mathbf{a}}$

  Having one of these is the same as having all three of them.

- In particular, because $\underline{\mathbf{x}} \oplus \underline{\mathbf{e}} = \underline{\boldsymbol{y}}$, if we are given two quantities, we can find the third quantity by summing the other two.

Linear Block Codes

- Definition: $\mathcal{C}$ is a **(binary) linear (block) code** if and only if $\mathcal{C}$ forms a vector (sub)space (over GF(2)).

In case you forgot about the concept of vector space,…

  - Equivalently, this is the same as requiring that

$$\text{if } \underline{\mathbf{x}}^{(1)} \text{ and } \underline{\mathbf{x}}^{(2)} \in \mathcal{C}, \text{ then } \underline{\mathbf{x}}^{(1)} \oplus \underline{\mathbf{x}}^{(2)} \in \mathcal{C}.$$

  - Note that any (non-empty) linear code $\mathcal{C}$ must contain $\underline{\mathbf{0}}$.

- Ex. The code that we considered in **Problem 5 of HW4** is

$$\mathcal{C} = \{00000, 01000, 10001, 11111\}$$

Is it a linear code?

Ex. Checking Linearity

- $\mathcal{C} = \{00000, 01000, 10001, 11111\}$
- Step 1: Check that $\mathbf{0} \in \mathcal{C}$.
  - OK for this example.
- Step 2: Check that

$$\text{if } \underline{\mathbf{x}}^{(1)} \text{ and } \underline{\mathbf{x}}^{(2)} \in \mathcal{C}, \text{ then } \underline{\mathbf{x}}^{(1)} \oplus \underline{\mathbf{x}}^{(2)} \in \mathcal{C}.$$

$\oplus$	00000	01000	10001	11111
00000				
01000				
10001				
11111				

Ex. Checking Linearity

- We have checked that
  $\mathcal{C} = \{00000, 01000, 10001, 11111\}$
  is not linear.
- Change one codeword in $\mathcal{C}$ to make the code linear.

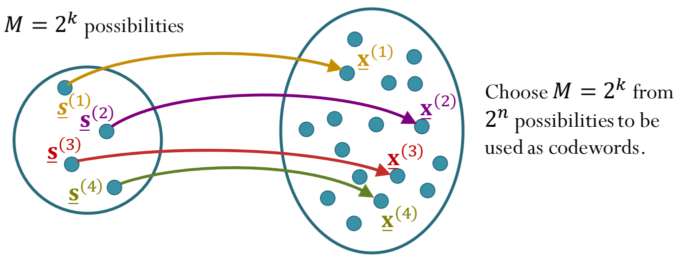
$\oplus$	00000			
00000				

Linear Block Codes: Motivation (1)

- Why linear block codes are popular?
- Recall: General block **encoding**
  - Characterized by its codebook.
    - The table that lists all the 2^k mapping from the k-bit info-block $\underline{\mathbf{s}}$ to the n-bit codeword $\underline{\mathbf{x}}$ is called the **codebook**.
    - The M info-blocks are denoted by $\underline{\mathbf{s}}^{(1)}, \underline{\mathbf{s}}^{(2)}, \ldots, \underline{\mathbf{s}}^{(M)}$. The corresponding M codewords are denoted by $\underline{\mathbf{x}}^{(1)}, \underline{\mathbf{x}}^{(2)}, \ldots, \underline{\mathbf{x}}^{(M)}$, respectively.

[See Section 3.5 of the lecture notes.]

index i	info-block $\underline{\mathbf{s}}$	codeword $\underline{\mathbf{x}}$
1	$\underline{\mathbf{s}}^{(1)} = 000\ldots0$	$\underline{\mathbf{x}}^{(1)} =$
2	$\underline{\mathbf{s}}^{(2)} = 000\ldots1$	$\underline{\mathbf{x}}^{(2)} =$
$\vdots$	$\vdots$	$\vdots$
M	$\underline{\mathbf{s}}^{(M)} = 111\ldots1$	$\underline{\mathbf{x}}^{(M)} =$



  - Can be realized by combinational/combinatorial circuit.
    - If lucky, can used K-map to simplify the circuit.

Linear Block Codes: Motivation (2)

- Why linear block codes are popular?
- Linear block encoding is the same as matrix multiplication.
  - See next slide.
  - The matrix replaces the table for the codebook.
  - The size of the matrix is only $k \times n$ bits.
    - Compare this against the table (codebook) of size $2^k \times (k+n)$ bits for general block encoding.
- Linearity $\Rightarrow$ easier implementation and analysis
- Performance of the class of linear block codes is similar to performance of the general class of block codes.
  - Can limit our study to the subclass of linear block codes without sacrificing system performance.

Example

- $\mathcal{C} = \{00000, 01000, 10001, 11001\}$
- Let

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

- Find $\underline{\mathbf{b}}\mathbf{G}$ when $\underline{\mathbf{b}} = [0\ 0]$.
- Find $\underline{\mathbf{b}}\mathbf{G}$ when $\underline{\mathbf{b}} = [0\ 1]$.
- Find $\underline{\mathbf{b}}\mathbf{G}$ when $\underline{\mathbf{b}} = [1\ 0]$.
- Find $\underline{\mathbf{b}}\mathbf{G}$ when $\underline{\mathbf{b}} = [1\ 1]$.

All possible two-bit vectors

Block Matrices

- A **block matrix** or a **partitioned matrix** is a matrix that is interpreted as having been broken into sections called **blocks** or **submatrices**.
- Examples:

$$\left(\underset{A}{\boxed{\begin{matrix}10 & 6\\ 9 & 7\end{matrix}}}\;\underset{B}{\boxed{\begin{matrix}6 & 4 & 3\\ 3 & 5 & 9\end{matrix}}}\right)$$

$$\left(\begin{matrix}\underset{C}{\boxed{\begin{matrix}2 & 2 & 5\\ 3 & 3 & 4\end{matrix}}} & \underset{D}{\boxed{\begin{matrix}10 & 2 & 10 & 2 & 5\\ 5 & 10 & 5 & 3 & 6\end{matrix}}}\\ \underset{E}{\boxed{\begin{matrix}3 & 3 & 4\\ 7 & 2 & 5\\ 8 & 3 & 6\end{matrix}}} & \underset{F}{\boxed{\begin{matrix}1 & 1 & 5 & 5 & 6\\ 3 & 10 & 6 & 10 & 3\\ 9 & 8 & 3 & 6 & 5\end{matrix}}}\end{matrix}\right)$$

Ex: Block Matrix Multiplications

$$\begin{pmatrix} \underbrace{\begin{matrix} 10 & 6 \\ 9 & 7 \end{matrix}}_{A} & \underbrace{\begin{matrix} 6 & 4 & 3 \\ 3 & 5 & 9 \end{matrix}}_{B} \end{pmatrix} \times \begin{pmatrix} \underbrace{\begin{matrix} 2 & 2 & 5 \\ 3 & 3 & 4 \end{matrix}}_{C} & \underbrace{\begin{matrix} 10 & 2 & 10 & 2 & 5 \\ 5 & 10 & 5 & 3 & 6 \end{matrix}}_{D} \\ \underbrace{\begin{matrix} 3 & 3 & 4 \\ 7 & 2 & 5 \\ 8 & 3 & 6 \end{matrix}}_{E} & \underbrace{\begin{matrix} 1 & 1 & 5 & 5 & 6 \\ 3 & 10 & 6 & 10 & 3 \\ 9 & 8 & 3 & 6 & 5 \end{matrix}}_{F} \end{pmatrix}$$

$$= \begin{pmatrix} \underbrace{\begin{matrix} 108 & 73 & 136 \\ 155 & 85 & 164 \end{matrix}}_{AC+BE} & \underbrace{\begin{matrix} 175 & 150 & 193 & 126 & 149 \\ 224 & 213 & 197 & 158 & 165 \end{matrix}}_{AD+BF} \end{pmatrix}$$

$$\underbrace{\begin{pmatrix} 10 & 6 & 6 & 4 & 3 \\ 9 & 7 & 3 & 5 & 9 \end{pmatrix}}_{X} \times \begin{pmatrix} \underbrace{\begin{matrix} 2 & 2 & 5 & 10 \\ 3 & 3 & 4 & 5 \\ 3 & 3 & 4 & 1 \\ 7 & 2 & 5 & 3 \\ 8 & 3 & 6 & 9 \end{matrix}}_{G} & \underbrace{\begin{matrix} 2 & 10 & 2 & 5 \\ 10 & 5 & 3 & 6 \\ 1 & 5 & 5 & 6 \\ 10 & 6 & 10 & 3 \\ 8 & 3 & 6 & 5 \end{matrix}}_{H} \end{pmatrix}$$

$$= \begin{pmatrix} \underbrace{\begin{matrix} 108 & 73 & 136 & 175 \\ 155 & 85 & 164 & 224 \end{matrix}}_{XG} & \underbrace{\begin{matrix} 150 & 193 & 126 & 149 \\ 213 & 197 & 158 & 165 \end{matrix}}_{XH} \end{pmatrix}$$

From $\underline{\mathbf{b}}$ to $\underline{\mathbf{x}}$

$$\underline{\mathbf{x}} = \underline{\mathbf{b}}\mathbf{G} = [b_1 \;\; b_2 \;\; \cdots \;\; b_k] \begin{bmatrix} \boxed{\quad \underline{\mathbf{g}}^{(1)} \quad} \\ \boxed{\quad \underline{\mathbf{g}}^{(2)} \quad} \\ \vdots \\ \boxed{\quad \underline{\mathbf{g}}^{(k)} \quad} \end{bmatrix}_{k \times n}$$

$$= b_1 \underline{\mathbf{g}}^{(1)} \oplus b_2 \underline{\mathbf{g}}^{(2)} \oplus \cdots \oplus b_k \underline{\mathbf{g}}^{(k)} = \sum_{j=1}^{k} b_j \underline{\mathbf{g}}^{(j)}$$

- Any codeword is simply a linear combination of the rows of **G**.
  - The weights are given by the bits in the message $\underline{\mathbf{b}}$

Linear Combination in GF(2)

- A **linear combination** is an expression constructed from a set of terms by multiplying each term by a constant (weight) and adding the results.
- For example, a linear combination of *x* and *y* would be any expression of the form $ax + by$, where a and b are constants.
- General expression:
$$c_1\underline{\mathbf{a}}^{(1)} + c_2\underline{\mathbf{a}}^{(2)} + \cdots + c_k\underline{\mathbf{a}}^{(k)}$$
- In GF(2), c_i is limited to being 0 or 1. So, a linear combination is simply a sum of a sub-collection of the vectors.

Linear Block Codes: Generator Matrix

For any linear code, there is a matrix $\mathbf{G} = \begin{bmatrix} \boxed{\underline{\mathbf{g}}^{(1)}} \\ \boxed{\underline{\mathbf{g}}^{(2)}} \\ \vdots \\ \boxed{\underline{\mathbf{g}}^{(k)}} \end{bmatrix}_{k \times n}$

called the **generator matrix**

such that, for any codeword $\underline{\mathbf{x}}$, there is a message vector $\underline{\mathbf{b}}$ which produces $\underline{\mathbf{x}}$ by

$$\boxed{\underline{\mathbf{x}} = \underline{\mathbf{b}}\mathbf{G}} = \underbrace{\sum_{j=1}^{k} b_j \underline{\mathbf{g}}^{(j)}}$$

mod-2 summation

Note:

(1) Any codeword can be expressed as a linear combination of the rows of $\mathbf{G}$

(2) $C = \{\underline{\mathbf{b}}\mathbf{G} : \underline{\mathbf{b}} \in \{0,1\}^k\}$

Note also that, given a matrix $\mathbf{G}$, the (block) code that is constructed by (2) is always linear.

Fact: If a code is generated by plugging in every possible $\underline{\mathbf{b}}$ into $\underline{\mathbf{x}} = \underline{\mathbf{b}}\mathbf{G}$, then the code will automatically be linear.

Proof

If $\mathbf{G}$ has k rows. Then, $\underline{\mathbf{b}}$ will have k bits. We can list them all as $\underline{\mathbf{b}}^{(1)}, \underline{\mathbf{b}}^{(2)}, \dots, \underline{\mathbf{b}}^{(2^k)}$. The corresponding codewords are

$$\underline{\mathbf{x}}^{(i)} = \underline{\mathbf{b}}^{(i)}\mathbf{G} \text{ for } i = 1, 2, \dots, 2^k.$$

Let's take two codewords, say, $\underline{\mathbf{x}}^{(i_1)}$ and $\underline{\mathbf{x}}^{(i_2)}$. By construction, $\underline{\mathbf{x}}^{(i_1)} = \underline{\mathbf{b}}^{(i_1)}\mathbf{G}$ and $\underline{\mathbf{x}}^{(i_2)} = \underline{\mathbf{b}}^{(i_2)}\mathbf{G}$. Now, consider the sum of these two codewords:

$$\underline{\mathbf{x}}^{(i_1)} \oplus \underline{\mathbf{x}}^{(i_2)} = \underline{\mathbf{b}}^{(i_1)}\mathbf{G} \oplus \underline{\mathbf{b}}^{(i_2)}\mathbf{G} = \left(\underline{\mathbf{b}}^{(i_1)} \oplus \underline{\mathbf{b}}^{(i_2)}\right)\mathbf{G}$$

Note that because we plug in ***every possible*** $\underline{\mathbf{b}}$ to create this code, we know that $\underline{\mathbf{b}}^{(i_1)} \oplus \underline{\mathbf{b}}^{(i_2)}$ should be one of these $\underline{\mathbf{b}}$. Let's suppose $\underline{\mathbf{b}}^{(i_1)} \oplus \underline{\mathbf{b}}^{(i_2)} = \underline{\mathbf{b}}^{(i_3)}$ for some $\underline{\mathbf{b}}^{(i_3)}$. This means

$$\underline{\mathbf{x}}^{(i_1)} \oplus \underline{\mathbf{x}}^{(i_2)} = \underline{\mathbf{b}}^{(i_3)}\mathbf{G}.$$

But, again, by construction, $\underline{\mathbf{b}}^{(i_3)}\mathbf{G}$ gives a codeword $\underline{\mathbf{x}}^{(i_3)}$ in this code. Because the sum of any two codewords is still a codeword, we conclude that the code is linear.

Linear Block Code: Example

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

- Find the codeword for the message $\underline{\mathbf{b}} = [1\ 0\ 0]$
- Find the codeword for the message $\underline{\mathbf{b}} = [0\ 1\ 1]$
- How many codewords do this code have?

Linear Block Code: Codebook

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

$$\underline{\mathbf{x}} = \underline{\mathbf{b}}\mathbf{G} = (b_1\ b_2\ b_3)\begin{pmatrix} 1 & 0 & 0 & 1 & 0 & 1 \\ 0 & 1 & 0 & 0 & 1 & 1 \\ 0 & 0 & 1 & 1 & 1 & 0 \end{pmatrix}$$
$$= (b_1, b_2, b_3, b_1 \oplus b_3, b_2 \oplus b_3, b_1 \oplus b_2)$$

$\underline{\mathbf{b}}$			$\underline{\mathbf{x}}$					
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
1	1	0	1	1	0	1	1	0
1	1	1	1	1	1	0	0	0

MATLAB: Codebook

```
G = [1 0 0 1 0 1; 0 1 0 0 1 1; 0 0 1 1 1 0];
[B C] = blockCodebook(G)
```

```
function [B C] = blockCodebook(G)
[k n] = size(G);
% All data words
B = dec2bin(0:2^k-1)-'0';
% All codewords
C = mod(B*G,2);
end
```

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

$\underline{\mathbf{b}}$			$\underline{\mathbf{x}}$					
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
1	1	0	1	1	0	1	1	0
1	1	1	1	1	1	0	0	0

Linear Block Code: Example

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

- Find the codeword for the message $\underline{\mathbf{b}} = [1\ 0\ 0\ 0]$
- Find the codeword for the message $\underline{\mathbf{b}} = [0\ 1\ 1\ 0]$
- How many codewords do this code have?

MATLAB: Codebook

```
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)
```

```
function [B C] = blockCodebook(G)
[k n] = size(G);
% All data words
B = dec2bin(0:2^k-1)-'0';
% All codewords
C = mod(B*G,2);
end
```

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

$\underline{\mathbf{b}}$				$\underline{\mathbf{x}}$						
0	0	0	0	0	0	0	0	0	0	0
0	0	0	1	1	0	1	0	1	0	1
0	0	1	0	0	0	1	0	1	1	0
0	0	1	1	1	0	0	0	0	1	1
0	1	0	0	1	0	0	1	1	0	0
0	1	0	1	0	0	1	1	0	0	1
0	1	1	0	1	0	1	1	0	1	0
0	1	1	1	0	0	0	1	1	1	1
1	0	0	0	1	1	1	0	0	0	0
1	0	0	1	0	1	0	0	1	0	1
1	0	1	0	1	1	0	0	1	1	0
1	0	1	1	0	1	1	0	0	1	1
1	1	0	0	0	1	1	1	1	0	0
1	1	0	1	1	1	0	1	0	0	1
1	1	1	0	0	1	0	1	0	1	0
1	1	1	1	1	1	1	1	1	1	1

Summary

Review: Linear Block Codes

- Given a list of codewords for a code $\mathcal{C}$, we can determine whether $\mathcal{C}$ is linear by
  - Definition: if $\underline{\mathbf{x}}^{(1)}$ and $\underline{\mathbf{x}}^{(2)} \in \mathcal{C}$, then $\underline{\mathbf{x}}^{(1)} \oplus \underline{\mathbf{x}}^{(2)} \in \mathcal{C}$
  - Shortcut:
    - First check that $\mathcal{C}$ must contain $\underline{\mathbf{0}}$.
    - Then, check only pairs of the non-zero codewords.
      - One check = three checks
- Codewords can be generated by a **generator matrix**
  - $\underline{\mathbf{x}} = \underline{\mathbf{b}}\mathbf{G} = \sum_{i=1}^{k} b_i \underline{\mathbf{g}}^{(i)}$ where $\underline{\mathbf{g}}^{(i)}$ is the i^{th} row of $\mathbf{G}$
- Codebook can be generated by
  - working **row-wise**: generating each codeword one-by-one, or
  - working **column-wise**: first, reading, from **G**, how each bit in the codeword is created from the bits in $\underline{\mathbf{b}}$; then, in the codebook, carry out the operations on columns $\underline{\mathbf{b}}$.

Linear Block Codes: Examples

- **Repetition code**: $\underline{\mathbf{x}} = [b \quad b \quad \cdots \quad b]$
  - $\mathbf{G} = [1 \quad 1 \quad \cdots \quad 1]$
  - $\underline{\mathbf{x}} = \underline{\mathbf{b}}\mathbf{G} = b\mathbf{G} = [b \quad b \quad \cdots \quad b]$
  - $R = \frac{k}{n} = \frac{1}{n}$

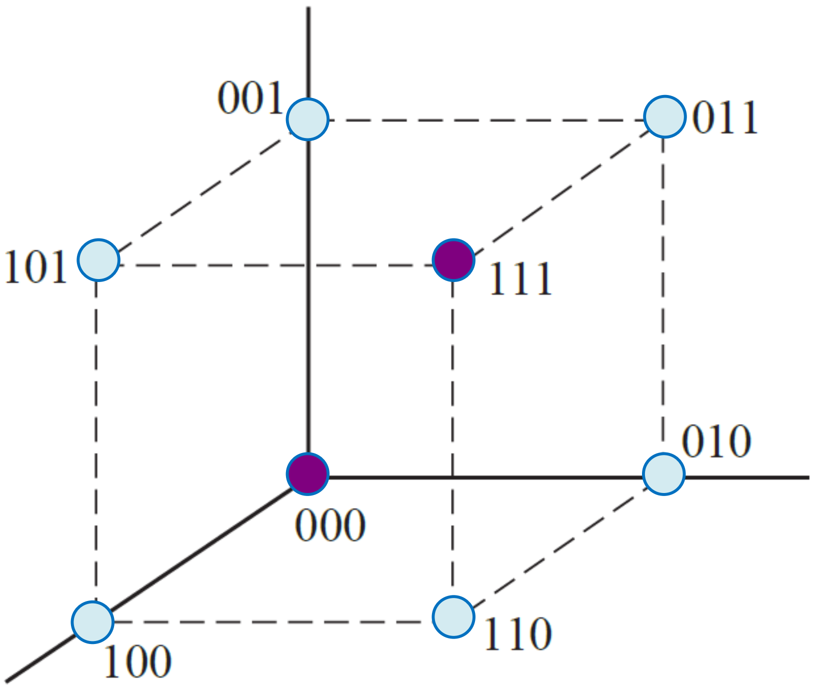
b	$\underline{\mathbf{x}}$		
0	0	0	0
1	1	1	1

- **Single-parity-check code**: $\underline{\mathbf{x}} = \left[\boxed{\underline{\mathbf{b}}}; \underbrace{\sum_{j=1}^{k} b_j}_{\text{parity bit}}\right]$
  - $\mathbf{G} = [\mathbf{I}_{k\times k}; \underline{\mathbf{1}}^T]$
  - $R = \frac{k}{n} = \frac{k}{k+1}$

$\underline{\mathbf{b}}$		$\underline{\mathbf{x}}$		
0	0	0	0	0
0	1	0	1	1
1	0	1	0	1
1	1	1	1	0

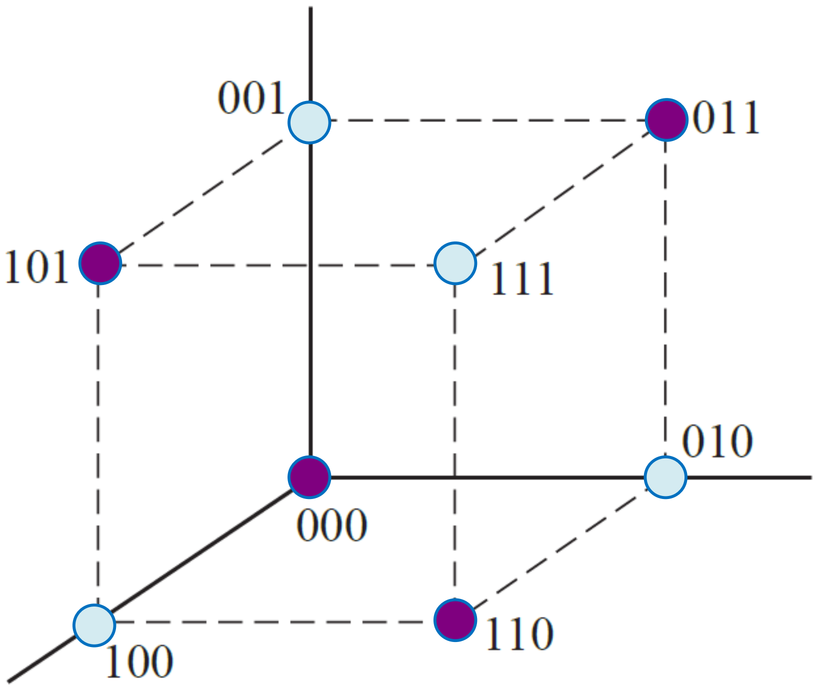
Vectors representing 3-bit codewords

Representing the codewords in the two examples on the previous slide as vectors:



Triple-repetition code

$$P(\mathcal{E}) = 1-(1-p)^3 - 3p(1-p)^2$$

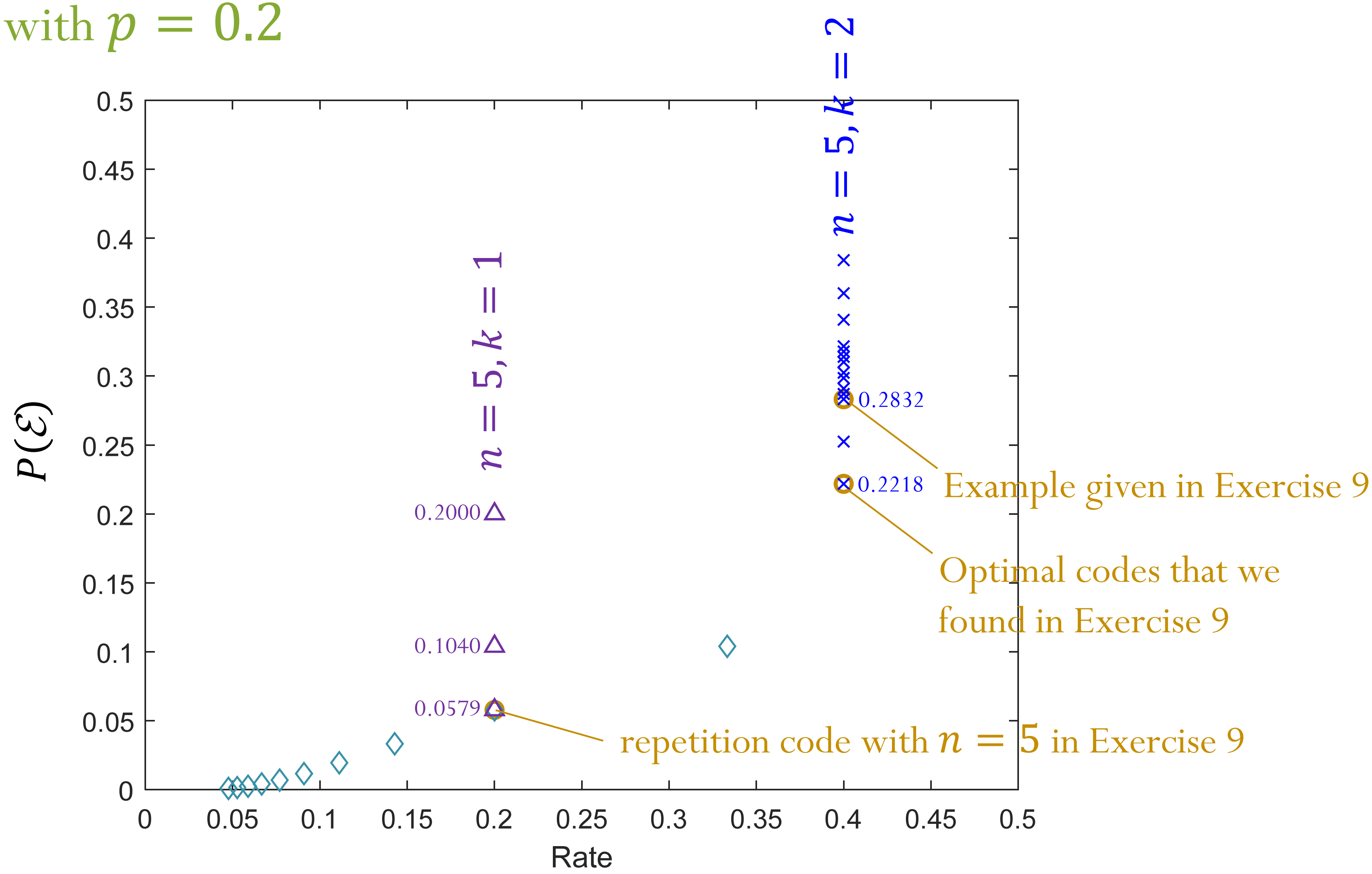


Single-Parity-check code

$$P(\mathcal{E}) = 1-(1-p)^3 - p(1-p)^2$$

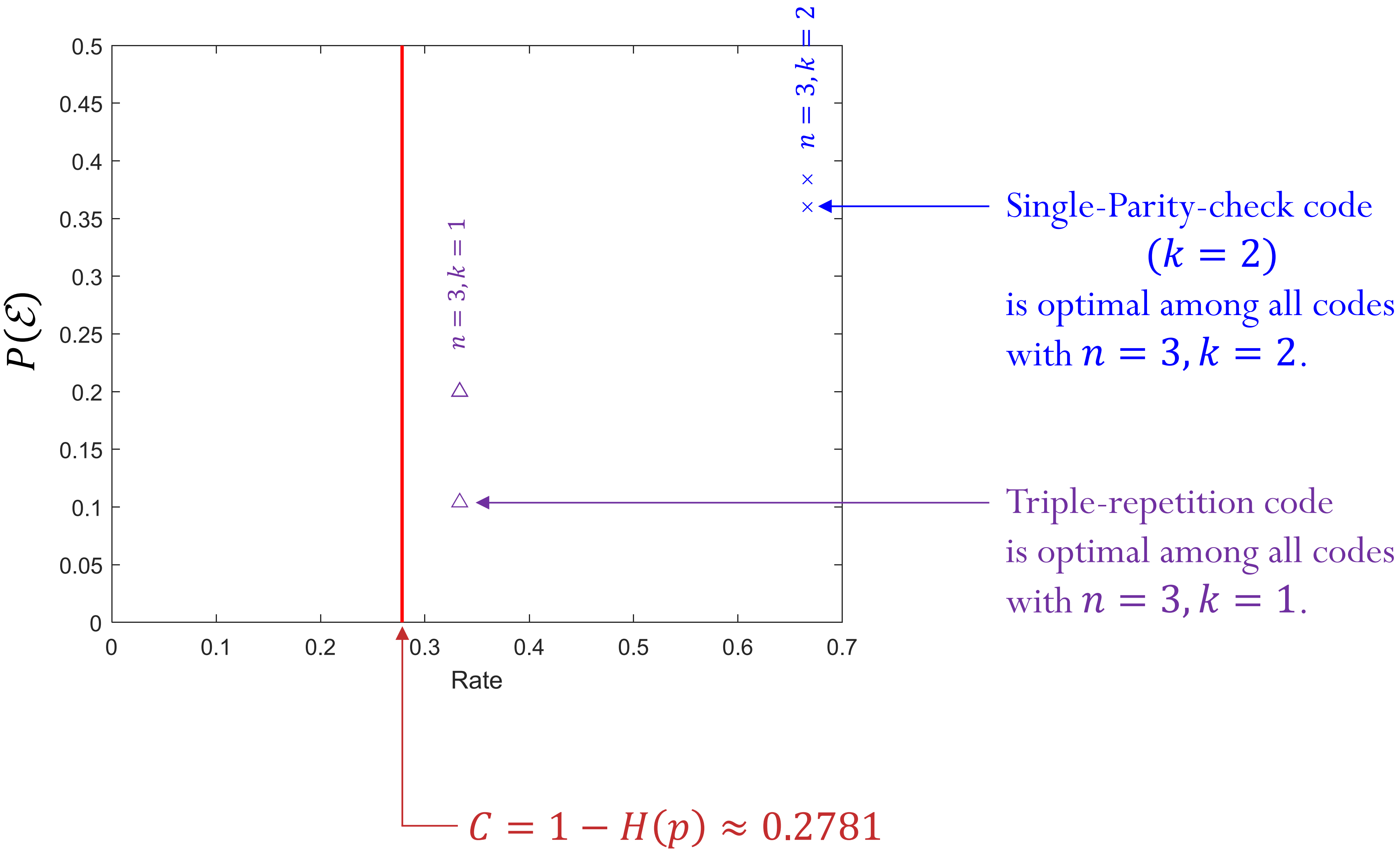
Recall: Achievable Performance

BSC with $p = 0.2$



Achievable Performance

BSC with $p = 0.2$



Related Idea:

Even Parity vs. Odd Parity

- Parity bit checking is used occasionally for transmitting ASCII characters, which have 7 bits, leaving the 8th bit as a **parity bit**.
- Two options:
  - **Even Parity**: Added bit ensures an <u>even</u> number of 1s in each codeword.
    - A: 1000001**0**
  - **Odd Parity**: Added bit ensures an <u>odd</u> number of 1s in each codeword.
    - A: 1000001**1**

Even Parity vs. Odd Parity

- Even parity and odd parity are properties of a codeword (a vector), not a bit.
- Note: The generator matrix $\mathbf{G} = [\mathbf{I}_{k\times k}; \underline{\mathbf{1}}^T]$ previously considered produces even parity codeword

$$\underline{\mathbf{x}} = \left[\boxed{\quad\underline{\mathbf{b}}\quad}; \sum_{j=1}^{k} b_j\right]$$

- Q: Consider a code that uses odd parity. Is it linear?

Error Control using Parity Bit

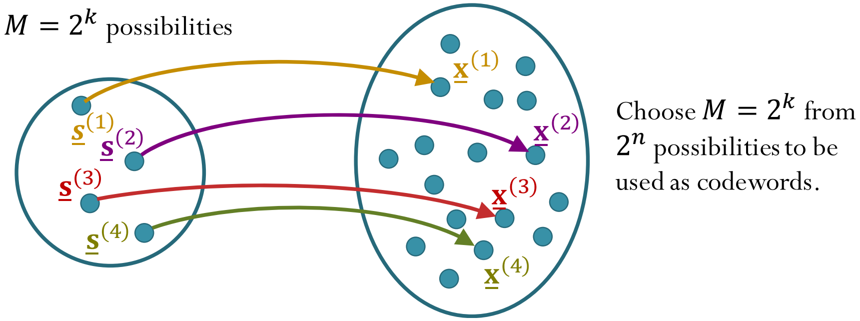
- If an odd number of bits (including the parity bit) are transmitted incorrectly, the parity will be incorrect, thus indicating that a parity error occurred in the transmission.
- Ex.
  - Suppose we use even parity.
  - Consider the codeword $\underline{\mathbf{x}}$ = 10000010 
- Suitable for *detecting* errors; *cannot correct* any errors

Two types of **error control**:

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

Error Detection

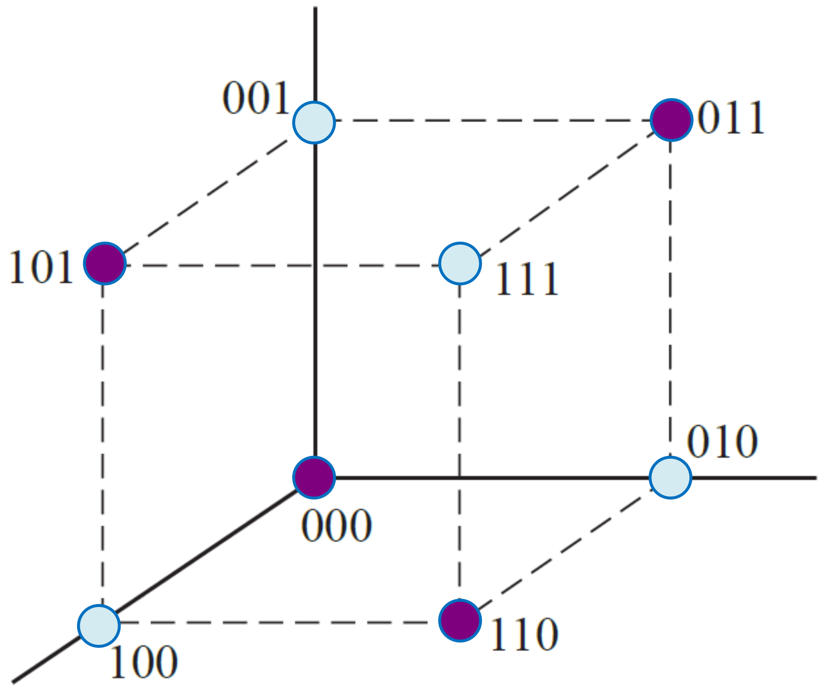
- **Error detection**: the determination of whether errors are present in a received word
  - usually by checking whether the received word is one of the valid codewords.



- 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.

- 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.

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

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.

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

$= 101110100$

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

$= 101110100\_\_\_\_\_\_$

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	

1	0	1	
1	1	0	
1	0	0	

$\underline{\boldsymbol{y}} = 100110100001111$

[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	_
_	_	_	

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 $\underline{\mathbf{e}}$.

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.

Review: 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. **Problem 5 of HW4**:

  Problem 5. 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	00000	01000	10001	11111
00000		1	2	5
01000			3	4
10001				3
11111				

$d_{\text{min}} = 1$

- 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
```

$d_{\min}$ 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

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

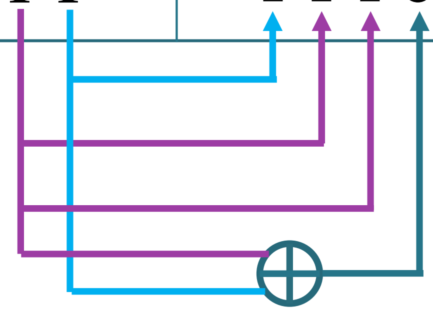
Example

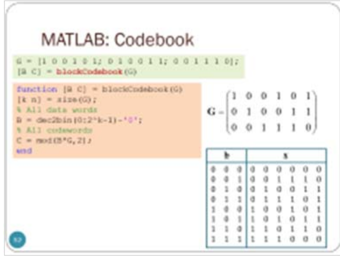
$$\mathbf{G} = \begin{bmatrix} 0 & 1 & 1 & 1 \\ 1 & 0 & 0 & 1 \end{bmatrix}$$

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

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

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





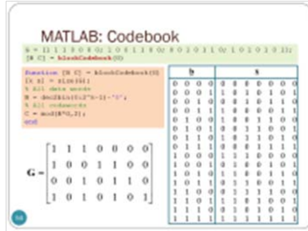
Example

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

$\underline{\mathbf{b}}$			$\underline{\mathbf{x}}$						$w(\underline{\mathbf{x}})$
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

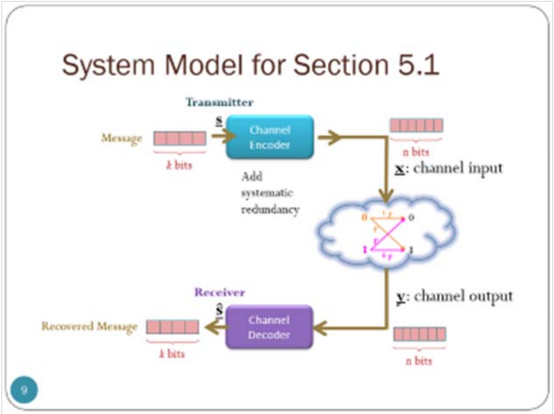


$$\mathbf{G} = \begin{bmatrix} 1 & 1 & 1 & 0 & 0 & 0 & 0 \\ 1 & 0 & 0 & 1 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 & 1 & 1 & 0 \\ 1 & 0 & 1 & 0 & 1 & 0 & 1 \end{bmatrix}$$

```
>> 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
```

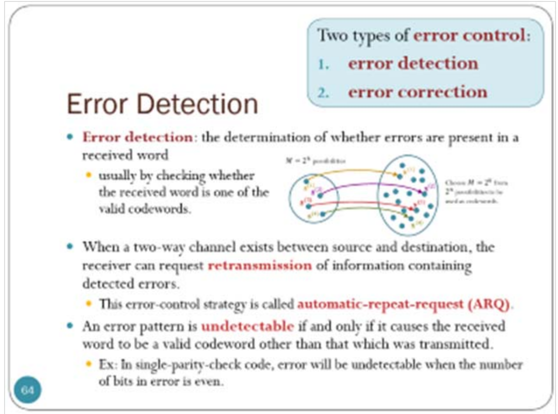
$\underline{\mathbf{b}}$	$\underline{\mathbf{x}}$	$w(\underline{\mathbf{x}})$
0 0 0 0	0 0 0 0 0 0 0	0
0 0 0 1	1 0 1 0 1 0 1	4
0 0 1 0	0 0 1 0 1 1 0	3
0 0 1 1	1 0 0 0 0 1 1	3
0 1 0 0	1 0 0 1 1 0 0	3
0 1 0 1	0 0 1 1 0 0 1	3
0 1 1 0	1 0 1 1 0 1 0	4
0 1 1 1	0 0 0 1 1 1 1	4
1 0 0 0	1 1 1 0 0 0 0	3
1 0 0 1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

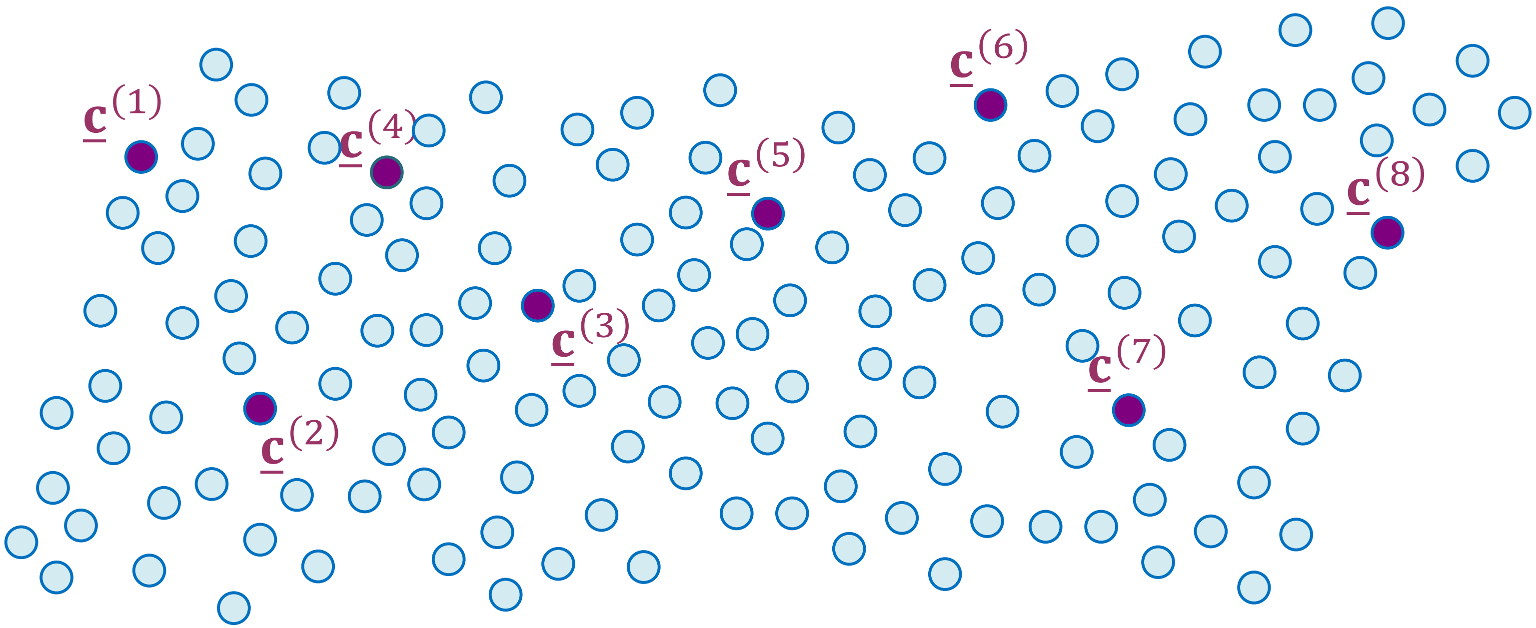
Visual Interpretation of $d_{\min}$



Recall: Codebook construction

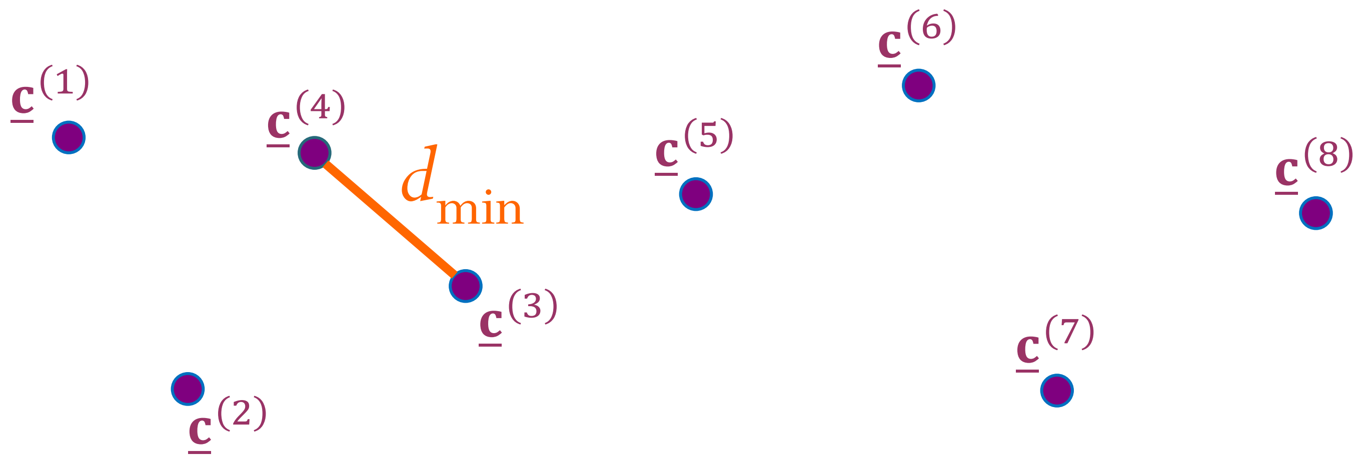
Choose $M = 2^k$ from 2^n possibilities to be used as codewords.





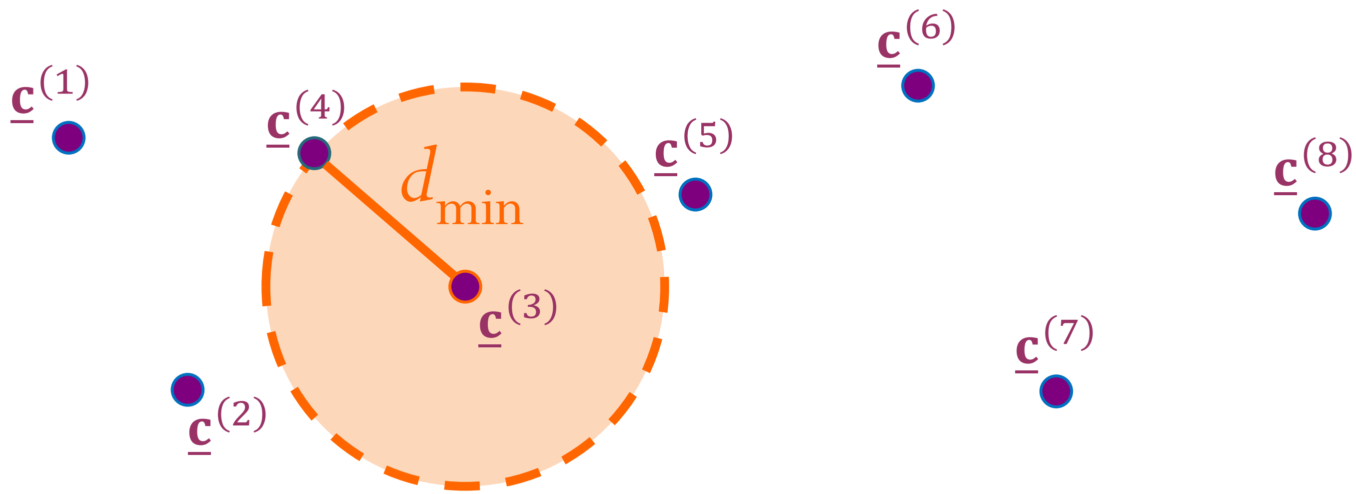
Visual Interpretation of d_{min}

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



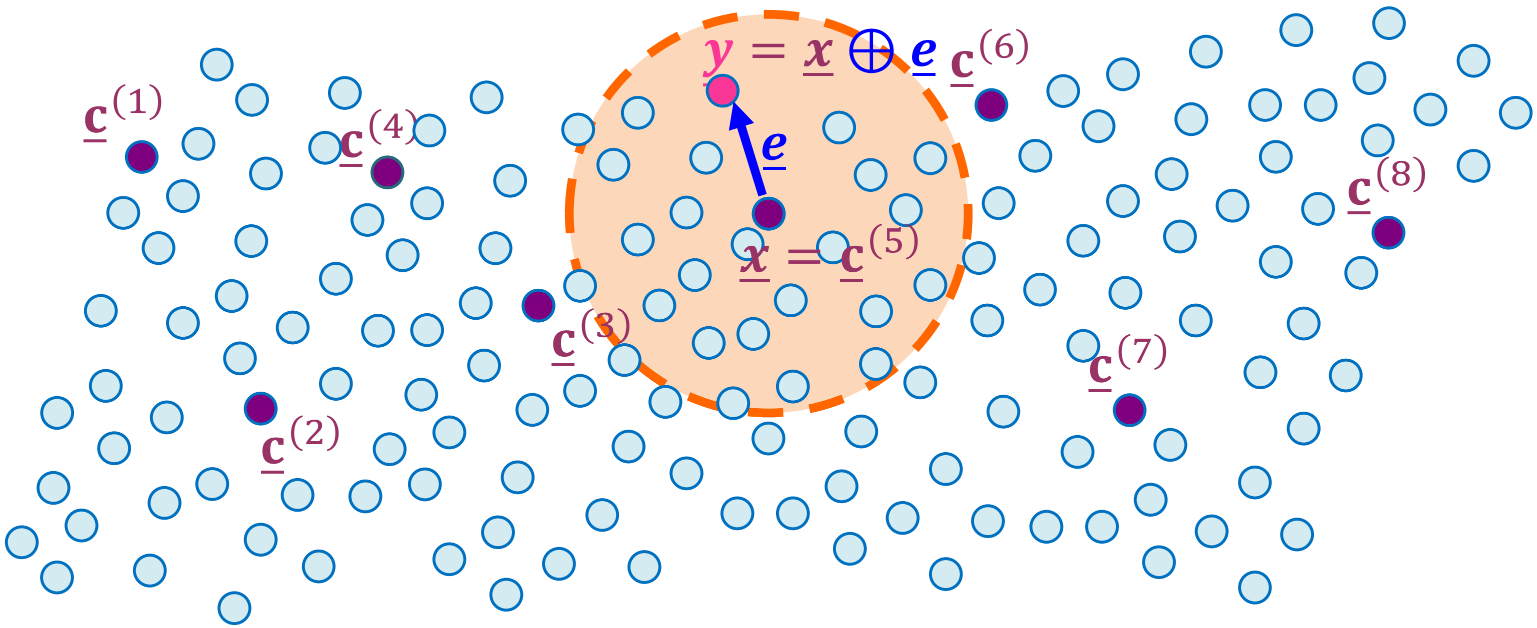
$d_{\min}$ and Error Detection

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

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

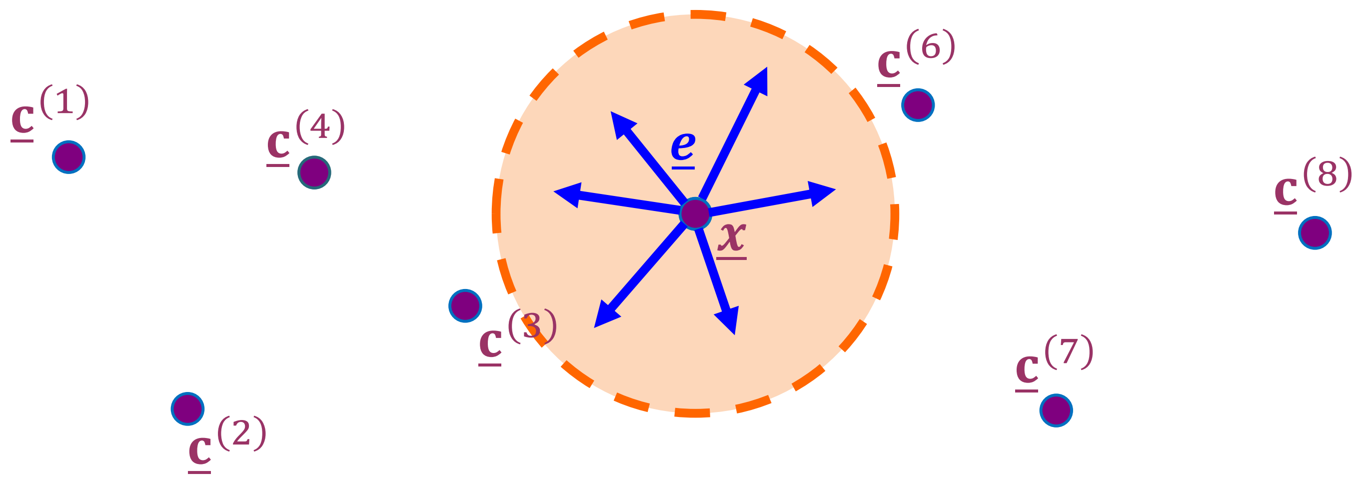
- The received vector $\underline{\boldsymbol{y}}$ can be calculated from

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



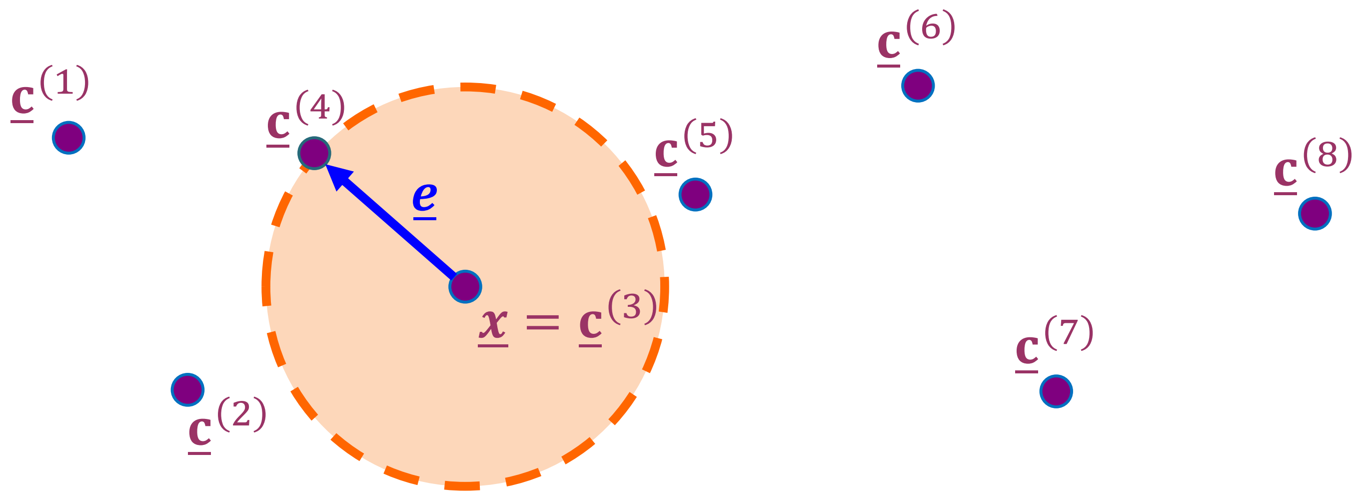
$d_{\min}$ and Error Detection

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



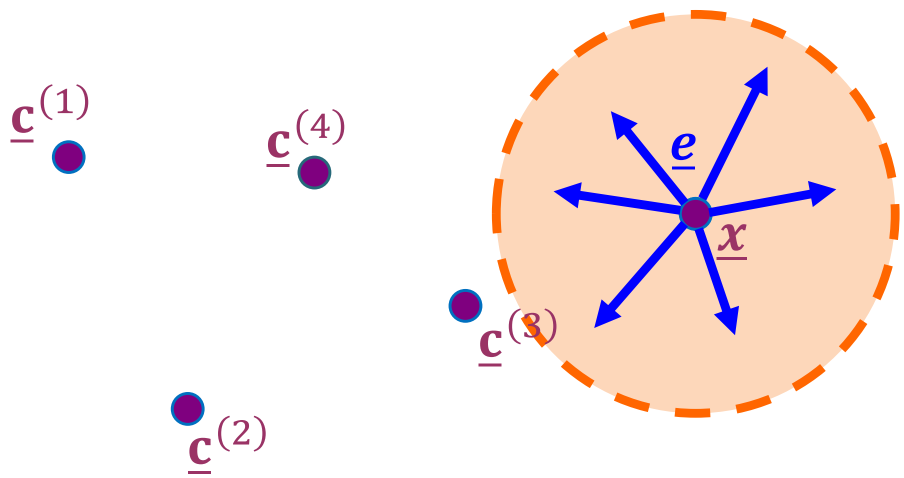
$d_{\min}$ and Error Detection

- 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.

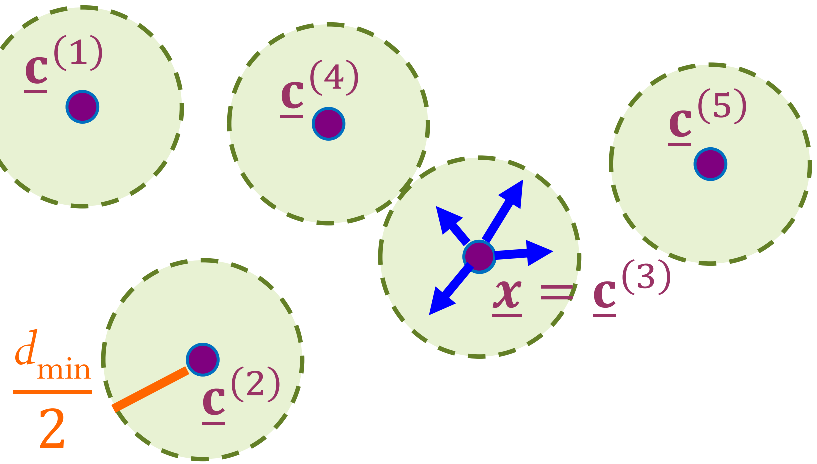


When $d_{\min} > w$, there is no way that w errors can change a valid codeword into another valid codeword.

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

$d_{\min}$ is an important quantity

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



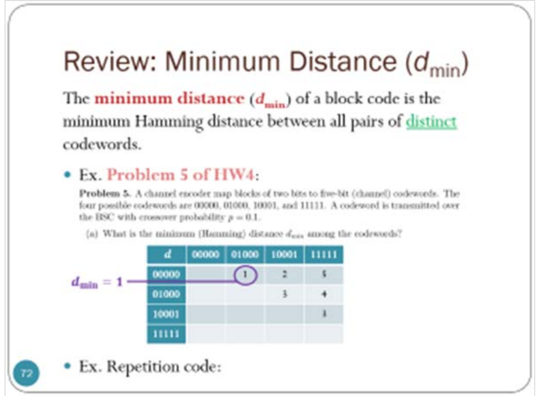
d_{min}: two important facts

- 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 $\leq d_{min}$-1.
  - correct all error patterns of weight w $\leq \left\lfloor \frac{d_{min}-1}{2} \right\rfloor$.

the floor function

Example



Review: 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. Problem 5 of HW4:

  Problem 5. 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?

$d_{\text{min}} = 1$

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

- Ex. Repetition code:

72

Repetition code with $n = 5$

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

Example

Consider the code

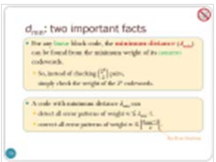
$$\mathcal{C} \in \{0000000000,\ 0000011111,\ 1111100000, \text{ and } 1111111111\}$$

- 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	$\underline{\mathbf{c}}^{(4)}$				

- $d_{\text{min}} =$

- It can detect (at most) ____ errors.



- It can correct (at most) ____ errors.