Digital Communication Systems
ECS 452

Asst. Prof. Dr. Prapun Suksompong

prapun@siit.tu.ac.th

5. Channel Coding

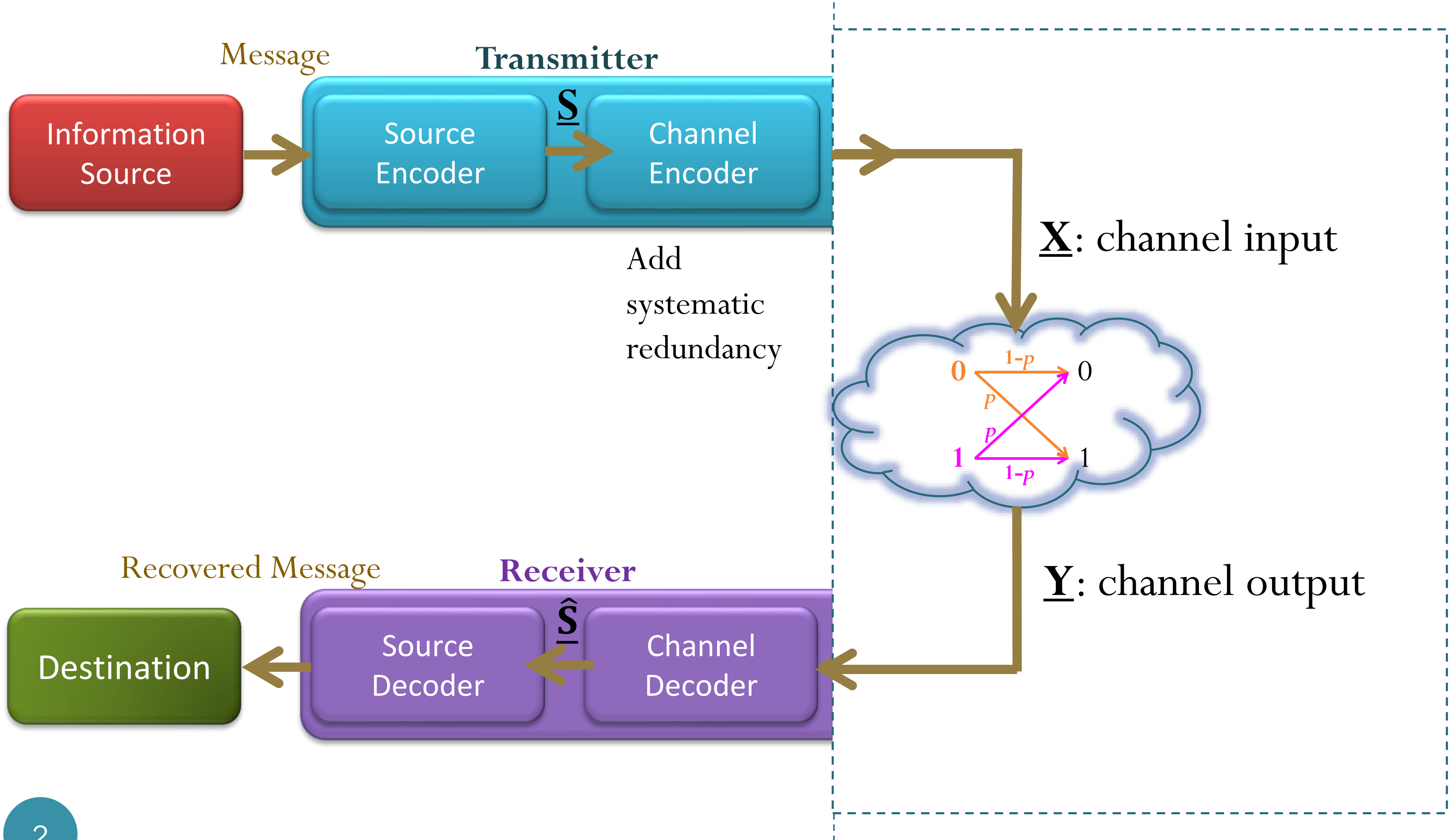
Office Hours:

Check Google Calendar on the course website.

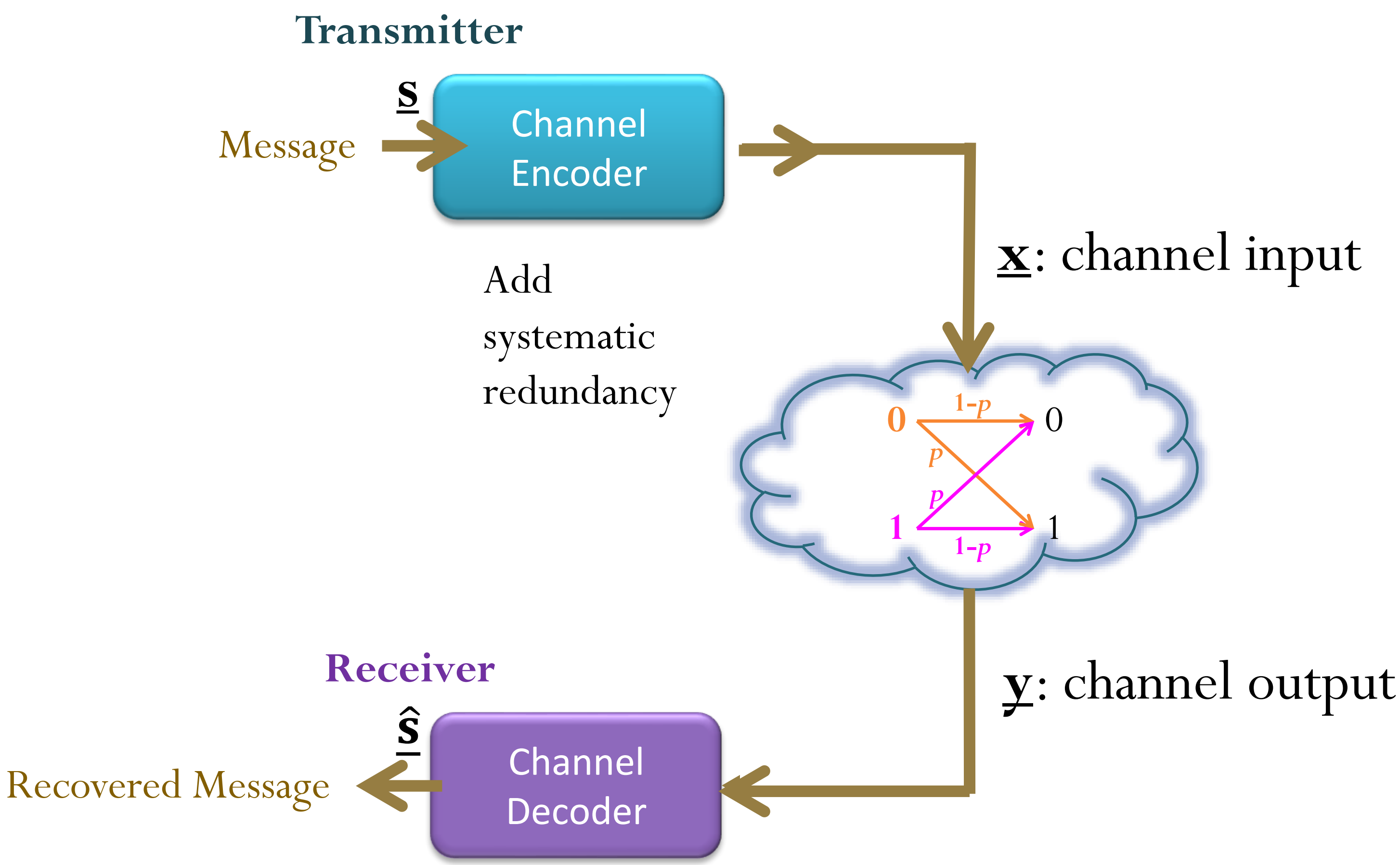
Dr.Prapun's Office:

6th floor of Sirindhralai building, BKD

Review: Channel Encoder and Decoder



System Model for Chapter 5



Vector Notation

- $\vec{\mathbf{v}}$: column vector $\begin{pmatrix} v_1 \\ v_2 \\ \vdots \\ v_i \\ \vdots \\ v_n \end{pmatrix}$
- $\underline{\mathbf{r}}$: row vector $(r_1, r_2, \dots, r_i, \dots r_n)$

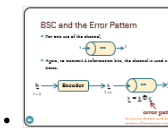
$\vec{0}, \underline{0}$: the zero vector (the all-zero vector)

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

- 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.

Review: Channel Decoding

- Recall
  1. The **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

Office Hours:

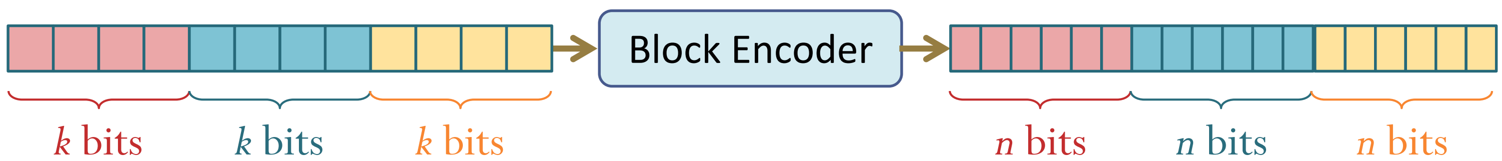
Check Google Calendar on the course website.

Dr.Prapun's Office:

6th floor of Sirindhralai building, BKD

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

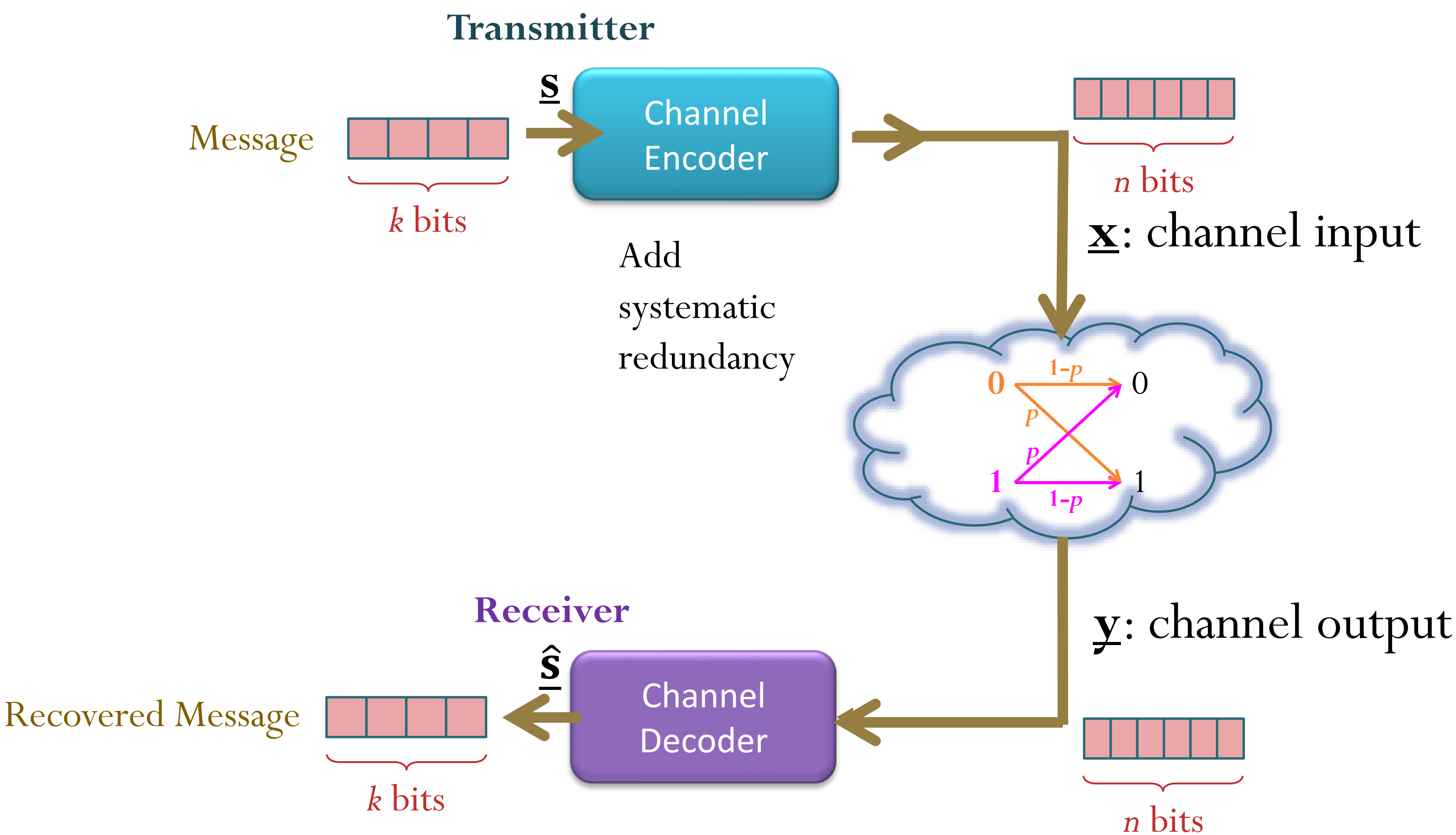
"messages"

- **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:

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

$\cdot$	0	1
0	0	0
1	0	1

- 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**),
- $\boldsymbol{a}$ **modulo** $\boldsymbol{n}$ (abbreviated as $\boldsymbol{a}$ **mod** $\boldsymbol{n}$) 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

×	0	1
0	0	0
1	0	1

- Addition and multiplication in GF(2):

⊕	0	1
0	0	1
1	1	0

•	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

$$x \oplus 0 = x$$

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

$$x \oplus x = 0$$

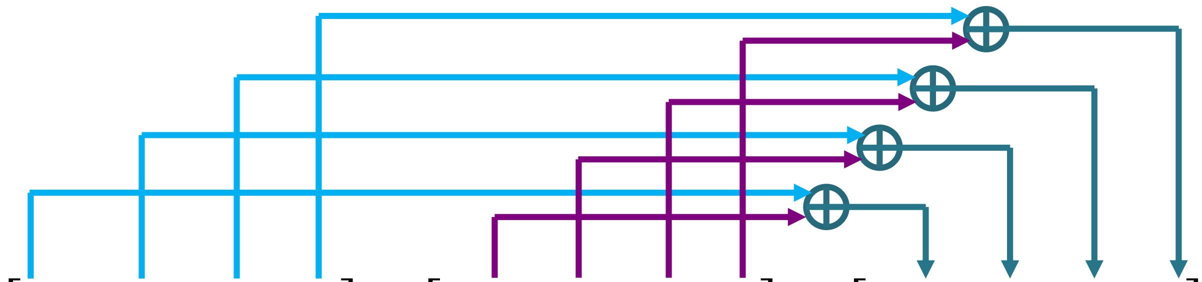
The above property implies $\underbrace{-x}_{} = x$

By definition, "-*x*" is something that, when added with *x*, gives 0.

- 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)).

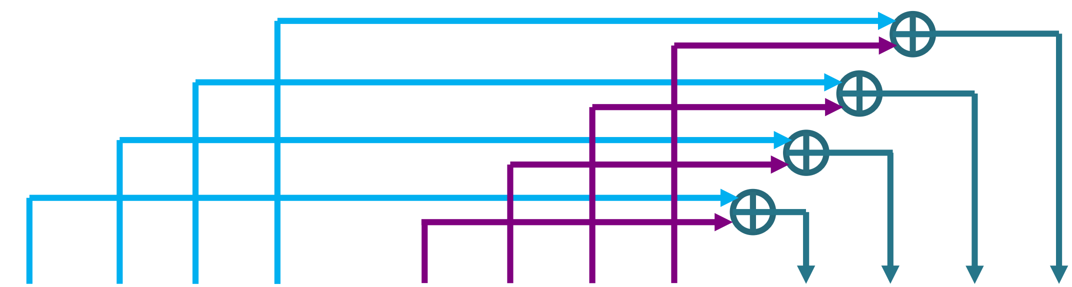
Examples

- Normal vector addition:



$$[1 \quad -1 \quad 2 \quad 1] + [-2 \quad 3 \quad 0 \quad 1] = [-1 \quad 2 \quad 2 \quad 2]$$

- Vector addition in GF(2):



$$[1 \quad 0 \quad 1 \quad 1] \oplus [0 \quad 1 \quad 0 \quad 1] = [1 \quad 1 \quad 1 \quad 0]$$

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

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

$$= [1 \quad 1 \quad 1 \quad 2] \xrightarrow{\text{mod } 2} [1 \quad 1 \quad 1 \quad 0]$$

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):

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

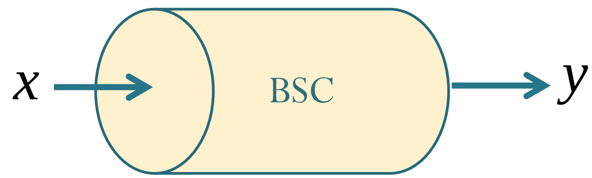
$$\begin{bmatrix} 1 & 0 & 1 \\ 0 & 0 & 1 \\ 1 & 1 & 1 \end{bmatrix} \begin{bmatrix} 1 & 1 \\ 0 & 1 \\ 1 & 0 \end{bmatrix} = \begin{bmatrix} 0 & 1 \\ 1 & 0 \\ 0 & 0 \end{bmatrix}$$

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

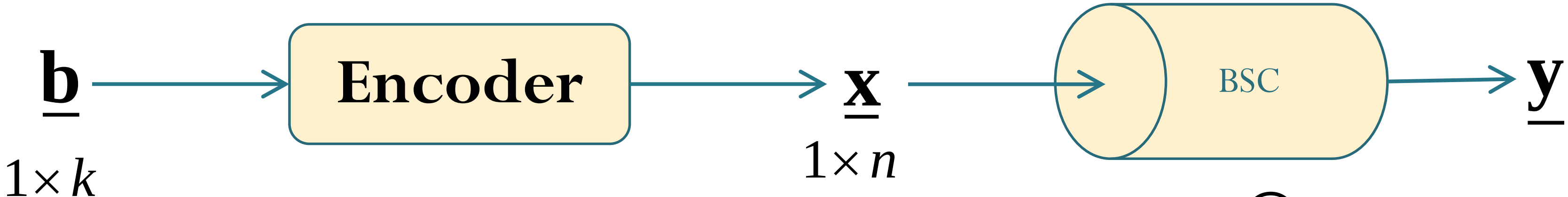
$$\begin{bmatrix} 1 & 0 & 1 \\ 0 & 0 & 1 \\ 1 & 1 & 1 \end{bmatrix} \begin{bmatrix} 1 & 1 \\ 0 & 1 \\ 1 & 0 \end{bmatrix} = \begin{bmatrix} 2 & 1 \\ 1 & 0 \\ 2 & 2 \end{bmatrix} \xrightarrow{\text{mod } 2} \begin{bmatrix} 0 & 1 \\ 1 & 0 \\ 0 & 0 \end{bmatrix}$$

BSC and the Error Pattern

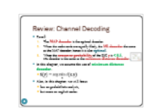
- For one use of the channel,



- 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 $\underline{\mathbf{y}}$

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 HW3** is

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

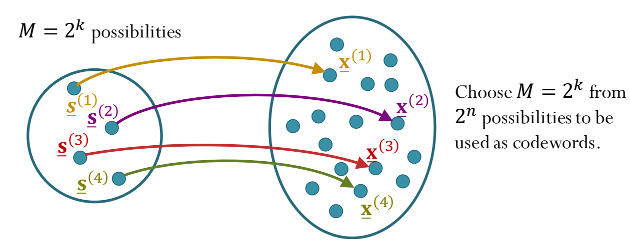
Is it a linear code?

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 $\underline{\mathbf{s}}^{(1)}, \underline{\mathbf{s}}^{(2)}, \dots, \underline{\mathbf{s}}^{(M)}$. The corresponding M codewords are denoted by $\underline{\mathbf{x}}^{(1)}, \underline{\mathbf{x}}^{(2)}, \dots, \underline{\mathbf{x}}^{(M)}$, respectively.

[See p. 51 in Ch. 3 of the lecture notes.]

index i	info-block $\underline{\mathbf{s}}$	codeword $\underline{\mathbf{x}}$
1	$\underline{\mathbf{s}}^{(1)} = 000\dots0$	$\underline{\mathbf{x}}^{(1)} =$
2	$\underline{\mathbf{s}}^{(2)} = 000\dots1$	$\underline{\mathbf{x}}^{(2)} =$
$\vdots$	$\vdots$	$\vdots$
M	$\underline{\mathbf{s}}^{(M)} = 111\dots1$	$\underline{\mathbf{x}}^{(M)} =$



  - 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.

Linear Block Codes: Generator Matrix

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

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

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

mod-2 summation

Note:

(1) Any codeword can be expressed as a linear combination of the rows of **G**

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

Note also that, given a matrix **G**, the (block) code that is constructed by (2) is always linear.

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 $\underline{\mathbf{b}} = [0\ 1\ 1]$

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 $\underline{\mathbf{b}} = [0\ 1\ 1\ 0]$

Linear Block Codes: Examples

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

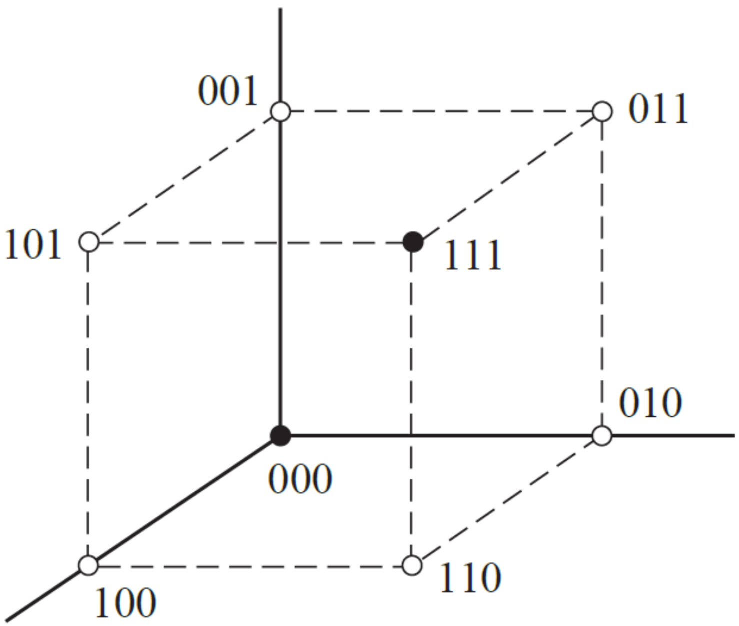
b	$\underline{\mathbf{x}}$		
0	0	0	0
1	1	1	1

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

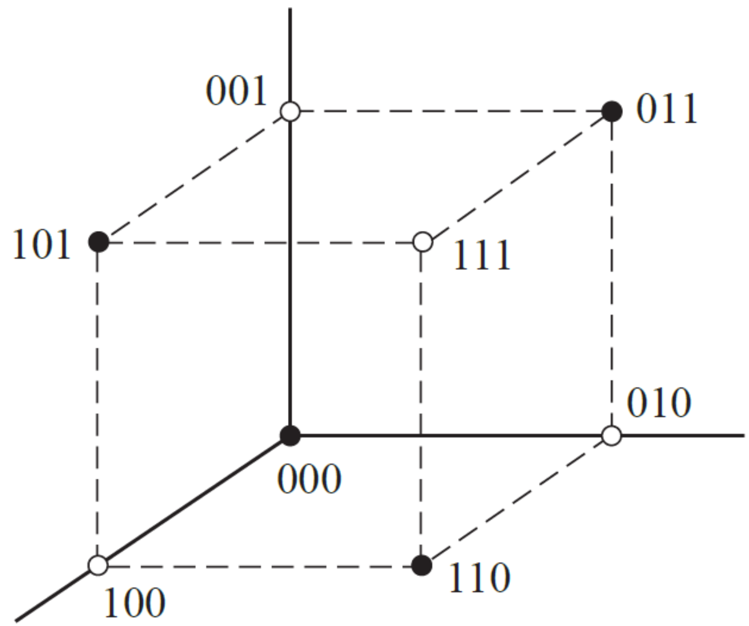
b		$\underline{\mathbf{x}}$		
0	0	0	0	0
0	1	0	1	1
1	0	1	0	1
1	1	1	1	0

Vectors representing 3-bit codewords

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



Triple-repetition code



Parity-check code

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 even number of 1s in each codeword.
    - A: 10000010
  - **Odd Parity**: Added bit ensures an odd number of 1s in each codeword.
    - A: 10000011

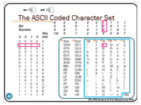
Even Parity vs. Odd Parity

- 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

US UK

The ASCII Coded Character Set

(American Standard Code for Information Interchange)

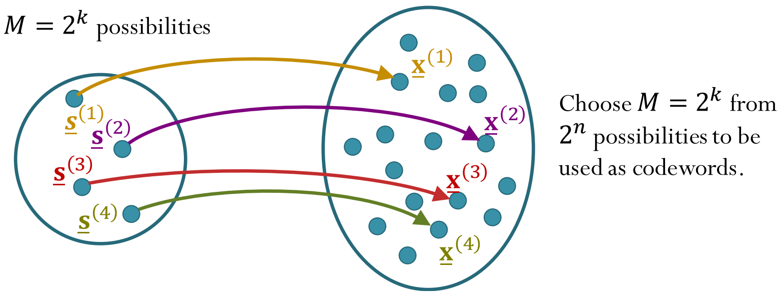
Bit Number					6	0	0	0	0	1	1	1	1
					5	0	0	1	1	0	0	1	1
					4	0	1	0	1	0	1	0	1
				Hex	1st	0	1	2	3	4	5	6	7
3	2	1	0	2nd									
0	0	0	0	0		0 NUL	16 DLE	32 SP	48 0	64 @	80 P	96 `	112 p
0	0	0	1	1		SOH	DC1	!	1	A	Q	a	q
0	0	1	0	2		STX	DC2	“	2	B	R	b	r
0	0	1	1	3		ETX	DC3	#	3	C	S	c	s
0	1	0	0	4		EOT	DC4	$	4	D	T	d	t
0	1	0	1	5		ENQ	NAK	%	5	E	U	e	u
0	1	1	0	6		ACK	SYN	&	6	F	V	f	v
0	1	1	1	7		BEL	ETB	'	7	G	W	g	w
1	0	0	0	8		BS	CAN	(	8	H	X	h	x
1	0	0	1	9		HT	EM	)	9	I	Y	i	y
1	0	1	0	A		LF	SUB	*	:	J	Z	j	z
1	0	1	1	B		VT	ESC	+	;	K	[	k	{
1	1	0	0	C		FF	FS	,	<	L	\	l	|
1	1	0	1	D		CR	GS	-	=	M	]	m	}
1	1	1	0	E		SO	RS	.	>	N	^	n	~
1	1	1	1	F		SI	US	/	?	O	—	o	DEL

[The ARRL Handbook for Radio Communications 2013]

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.

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	

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

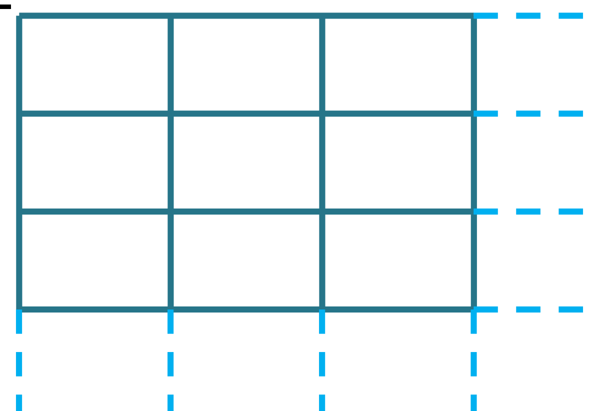
$= 101110100$

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

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

1	0	1
1	1	0
1	0	0

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



[Carlson & Crilly, p 594]

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

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?

  $d_{min} = 1$

$\underline{x}$ \ $\underline{x}$	00000	01000	10001	11111
00000		1	2	5
01000			3	4
10001				3
11111				

- Ex. Repetition code:

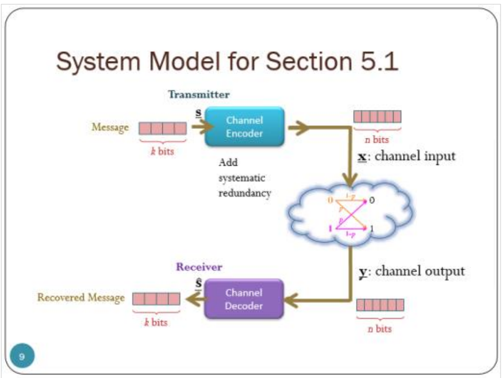
d_{min}: two important facts

- For any linear block code, the **minimum distance** (d_{min}) can be found from 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 $w \leq \left\lfloor \frac{d_{min}-1}{2} \right\rfloor$.

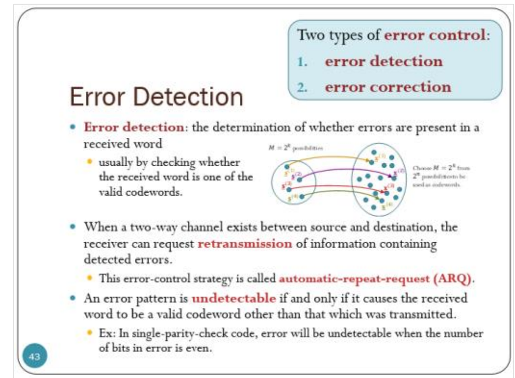
the floor function

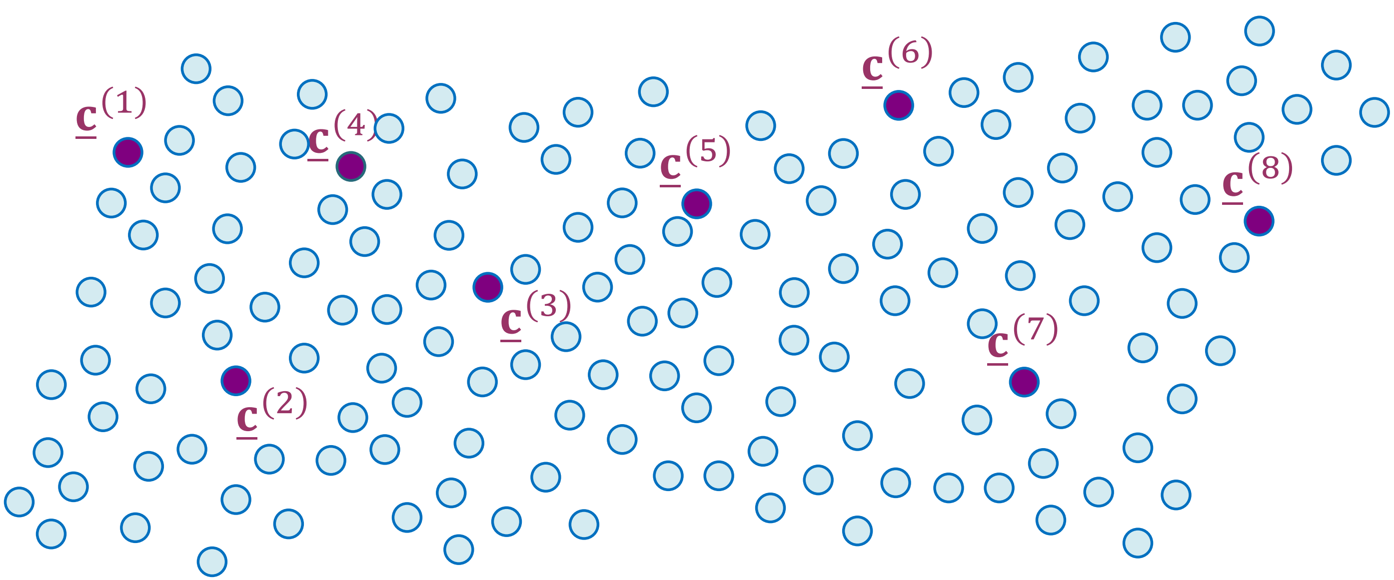
d_{min} is an important quantity



Recall: Codebook construction

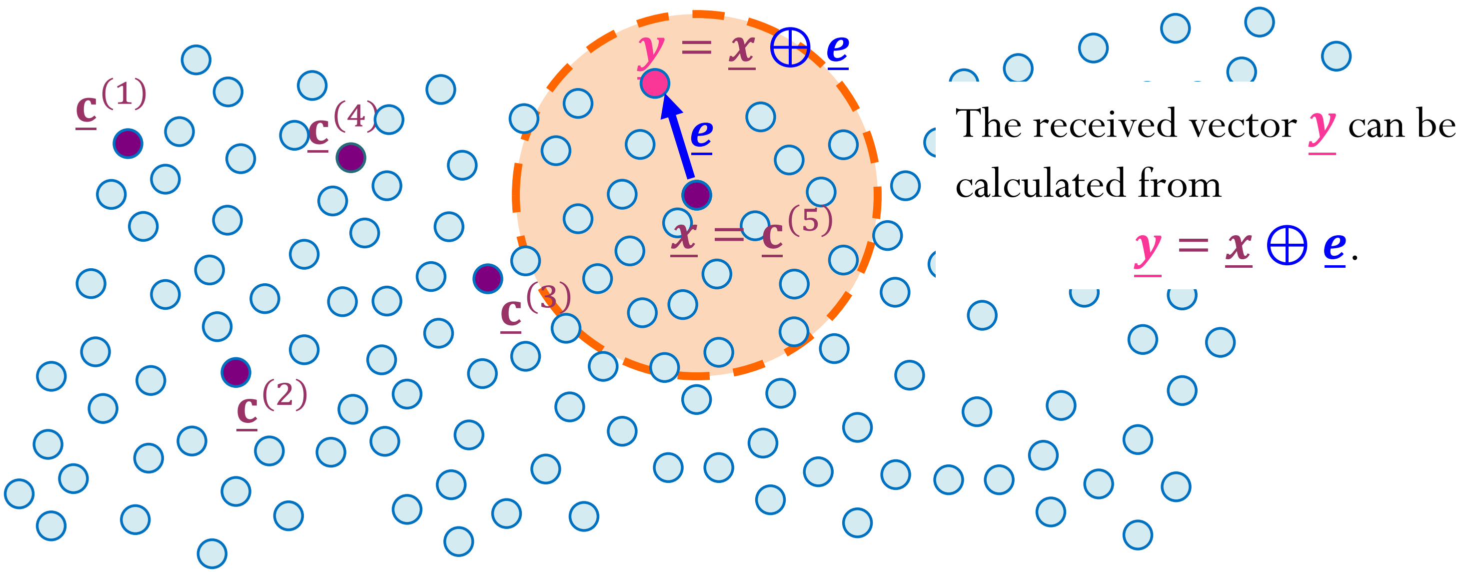
Choose $M = 2^k$ from 2^n possibilities to be used as codewords.





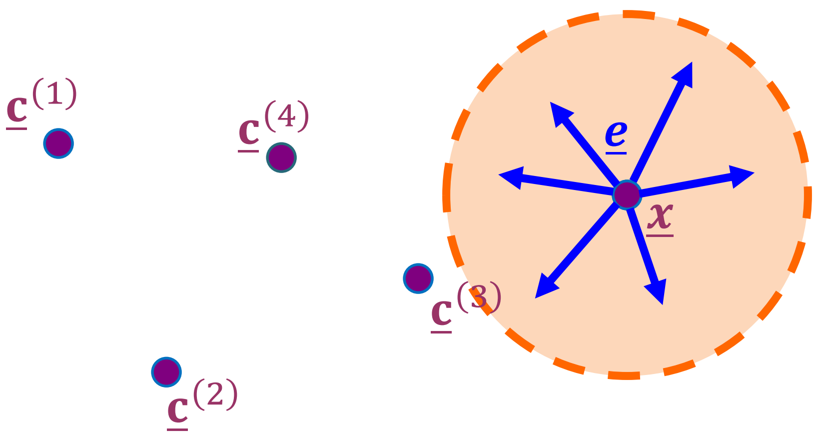
$d_{\min}$ is an important quantity

- 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.



$d_{\min}$ is an important quantity

- 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.

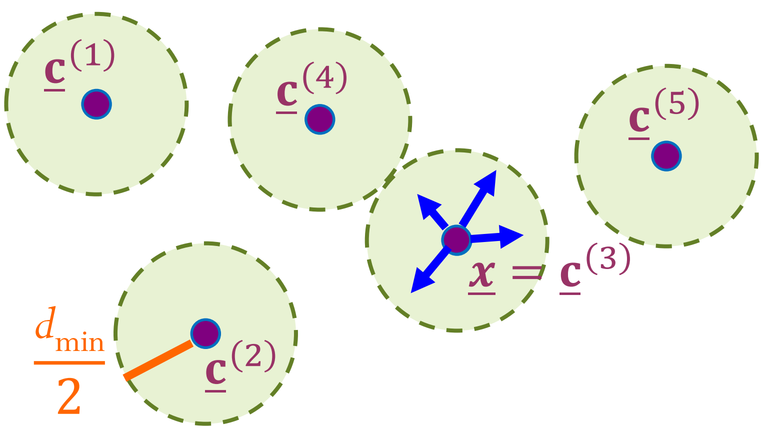


When $d_{\min} > w$, there is no way that w errors can change a valid codeword into another valid codeword.

When $d_{\min} = w$, 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.



Example

Consider the code

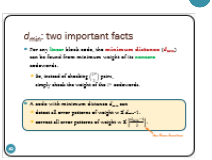
$$\mathcal{C} \in \{0000000000,\ 0000011111,\ 1111100000,\ \text{and}\ 1111111111\}$$

- 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.