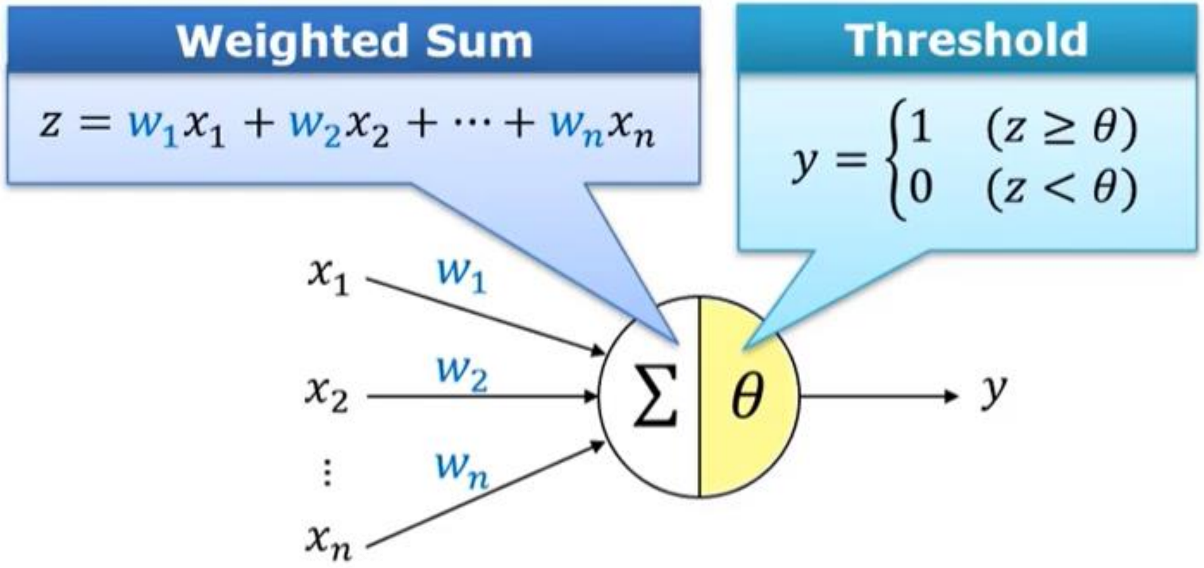
- **Key Steps for Perceptron Learning**
  - How to detect misclassified points
  - How to update a plane
- **Perceptron Learning Rule**
  - Graphical Explanation
  - Unified Learning Rule
  - Learning Rate

Recap: Perceptron vs MP Neuron



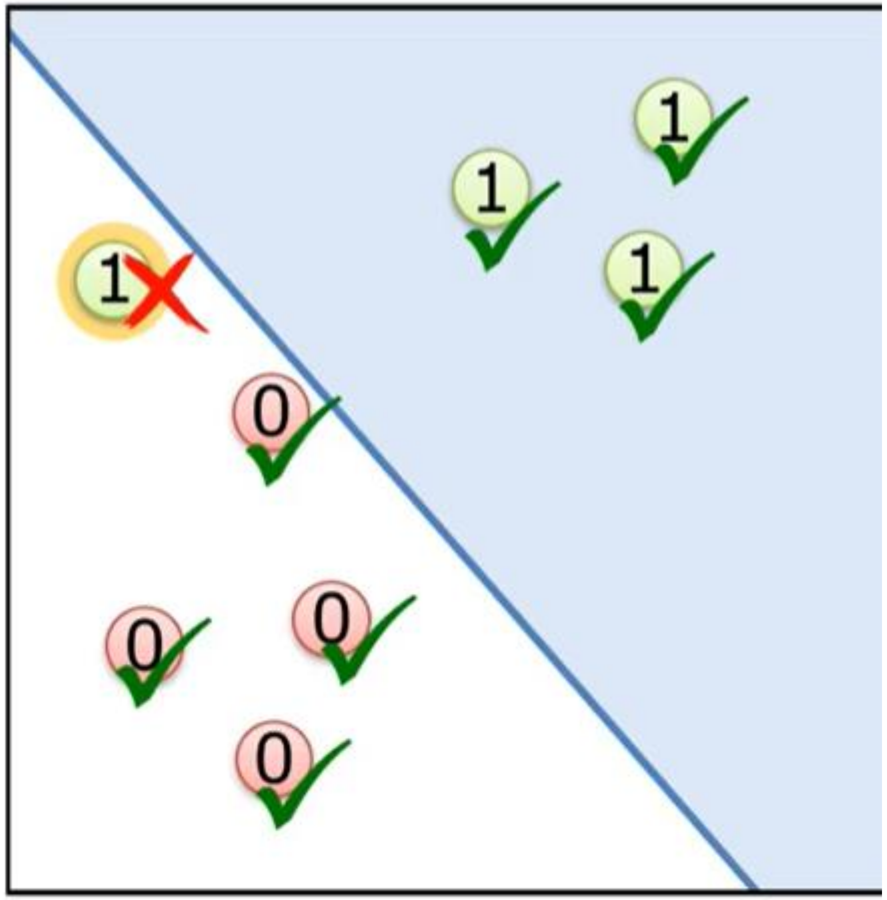
	Perceptron	MP Neuron
Inputs	Real numbers	Binary (0 or 1)
Weights	Each input carries a weight (which can be learned)	All inputs are equally important.
Threshold	Can be learned automatically	Manually set by users

Key Steps for Perceptron Learning

Initially, choose $(\boldsymbol{w}, b)$ at random to create a plane $\boldsymbol{w} \cdot \boldsymbol{x} + b = 0$

↓

Detect if there exists any misclassified node



Key Steps for Perceptron Learning

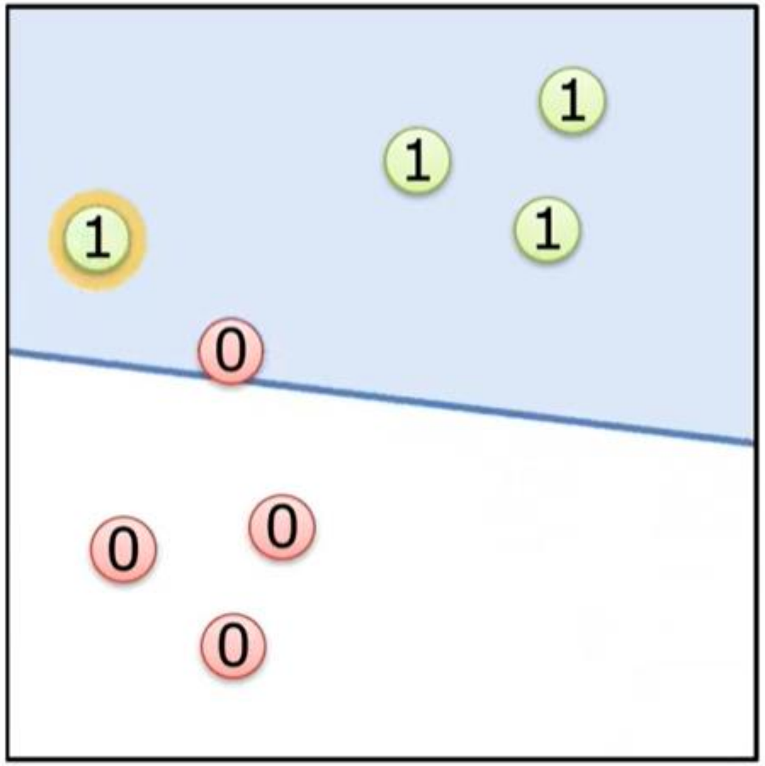
Initially, choose $(\boldsymbol{w}, b)$ at random to create a plane $\boldsymbol{w} \cdot \boldsymbol{x} + b = 0$

↓

Detect if there exists any misclassified node

↓ Y

Update the plane by adjusting $(\boldsymbol{w}, b)$



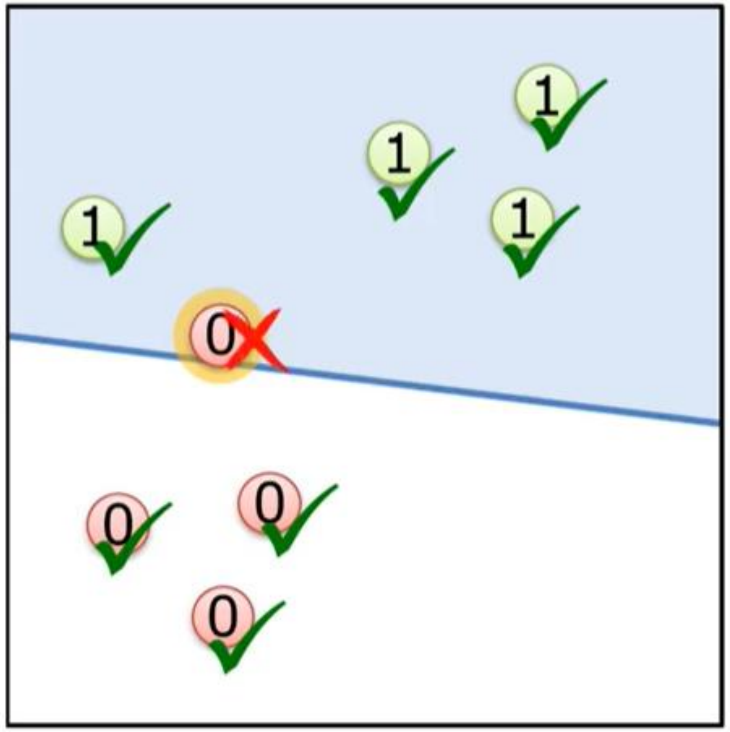
Key Steps for Perceptron Learning

Initially, choose $(\boldsymbol{w}, b)$ at random to create a plane $\boldsymbol{w} \cdot \boldsymbol{x} + b = 0$

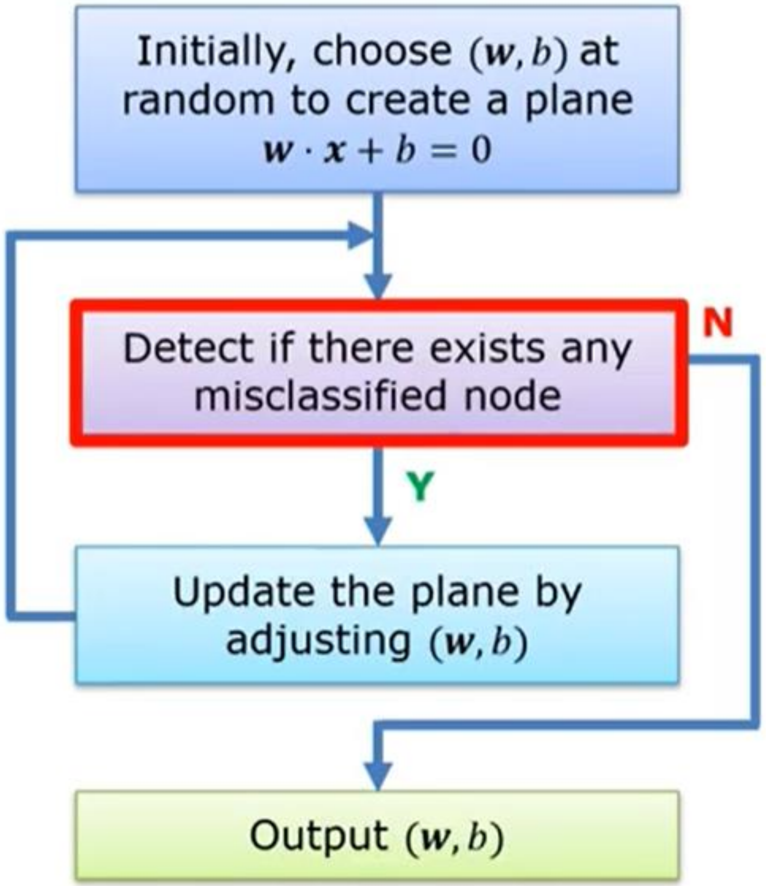
Detect if there exists any misclassified node

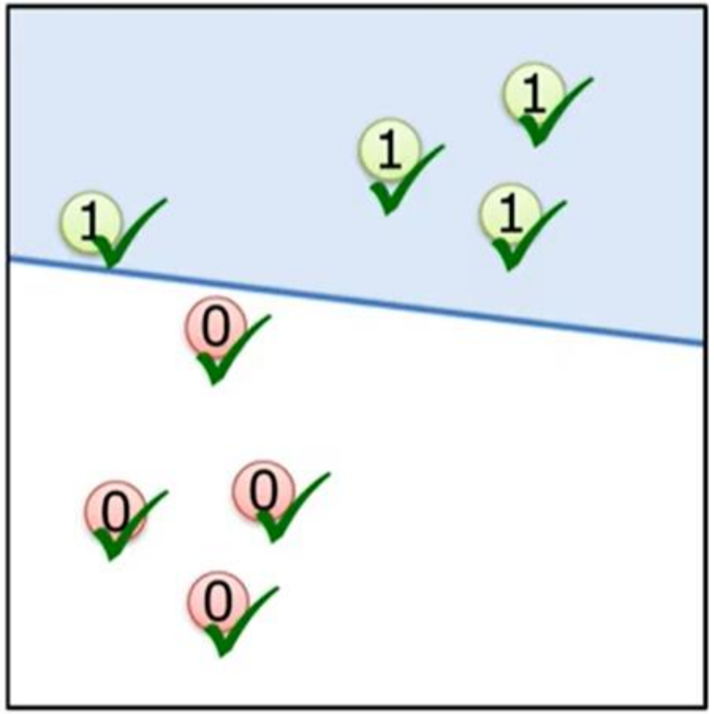
Y

Update the plane by adjusting $(\boldsymbol{w}, b)$

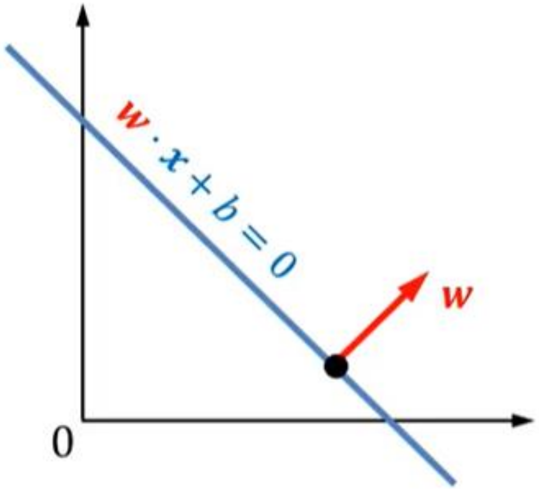


Key Steps for Perceptron Learning

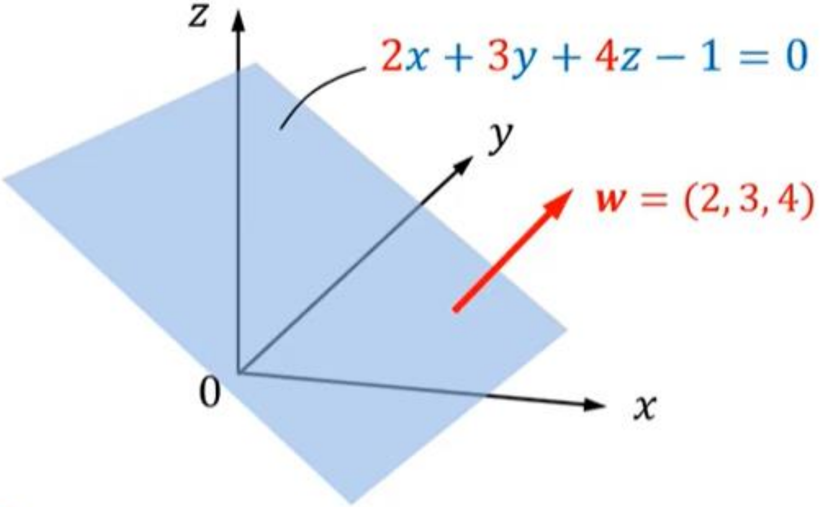




Normal Vector



Example in 3D



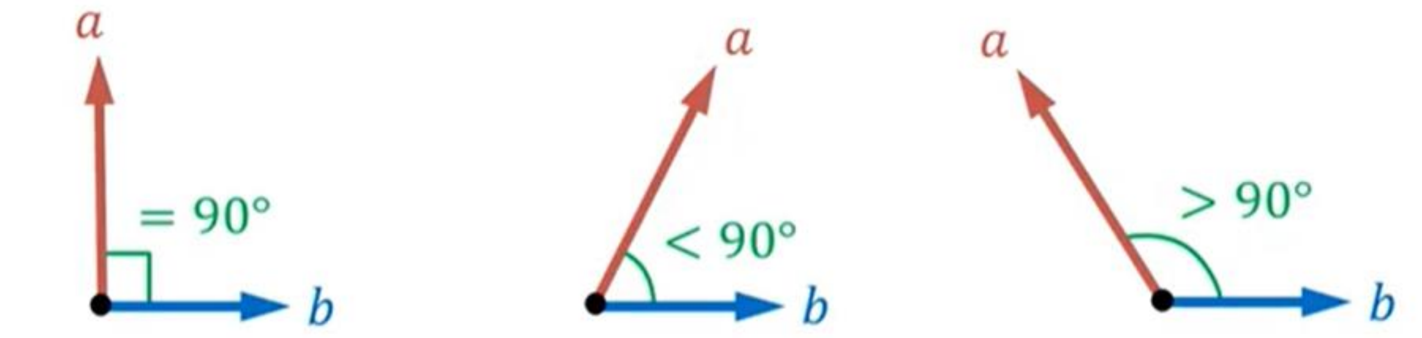
For the plane equation
$\boldsymbol{w} \cdot \boldsymbol{x} + b = 0$,
$\boldsymbol{w}$ is a normal vector,
which is perpendicular
to the plane.

$$2x + 3y + 4z - 1 = 0$$

$$\boldsymbol{w} = (2, 3, 4) \qquad \boldsymbol{x} = \begin{bmatrix} x \\ y \\ z \end{bmatrix} \qquad b = -1$$

$$\boldsymbol{w} \cdot \boldsymbol{x} + b = 0$$

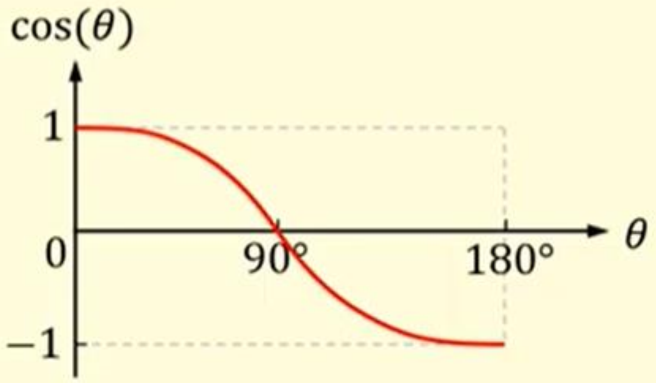
Dot Product & Angle btw Vectors



angle $(a, b) = 90°$ $\Leftrightarrow$ $a \cdot b = 0$

angle $(a, b) < 90°$ $\Leftrightarrow$ $a \cdot b > 0$

angle $(a, b) > 90°$ $\Leftrightarrow$ $a \cdot b < 0$



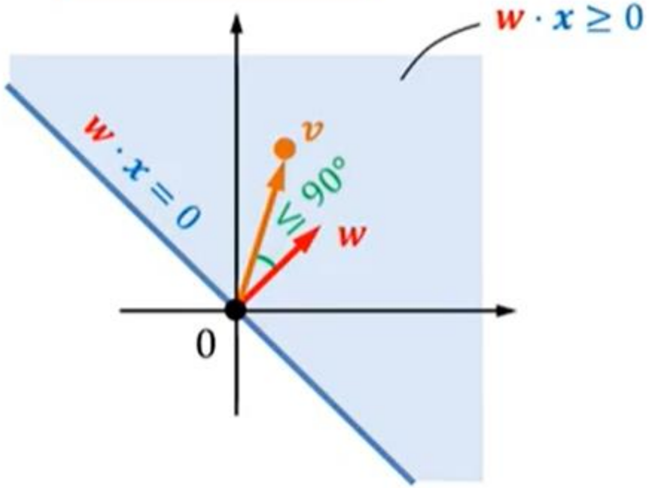
$$a \cdot b = |a| \cdot |b| \cdot \boxed{\cos\theta}$$

$$\cos\theta \begin{cases} > 0 & (0 < \theta < 90°) \\ = 0 & (\theta = 90°) \\ < 0 & (90° < \theta < 180°) \end{cases}$$

Determine if a point is inside a region

$$\boldsymbol{w} \cdot \boldsymbol{x} + b = 0 \quad \text{where} \quad \boldsymbol{w} = [w_1, w_2] \quad \boldsymbol{x} = \begin{bmatrix} x_1 \\ x_2 \end{bmatrix}$$

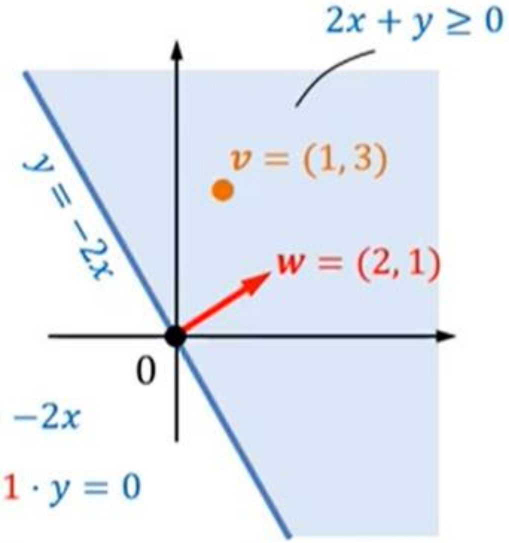
When $b = 0$



If v is inside the region, $\boldsymbol{w} \cdot \boldsymbol{v} \geq 0$

If v is outside the region, $\boldsymbol{w} \cdot \boldsymbol{v} < 0$

Example



$y = -2x$

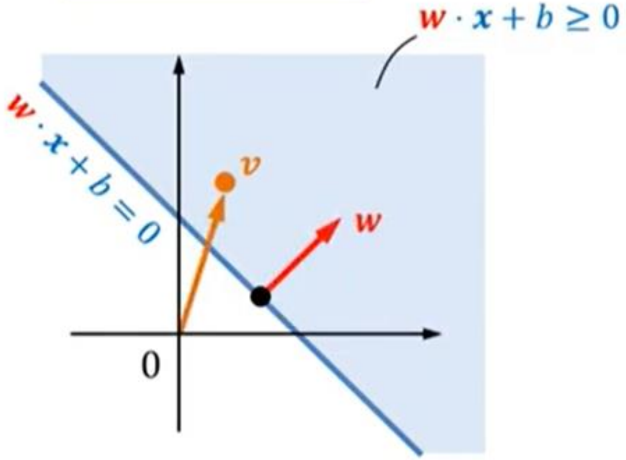
$2 \cdot x + 1 \cdot y = 0$

$\boldsymbol{w} = (2, 1) \quad v = (1, 3)$

$\Rightarrow \boldsymbol{w} \cdot v = (2, 1) \cdot (1, 3)$

$= 2 \times 1 + 1 \times 3 = 5 \geq 0$

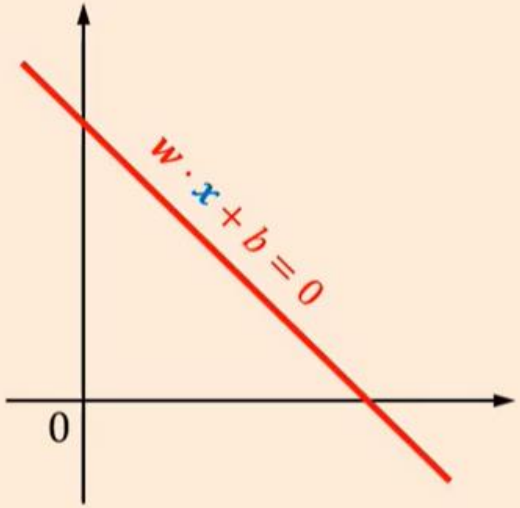
When $b \neq 0$



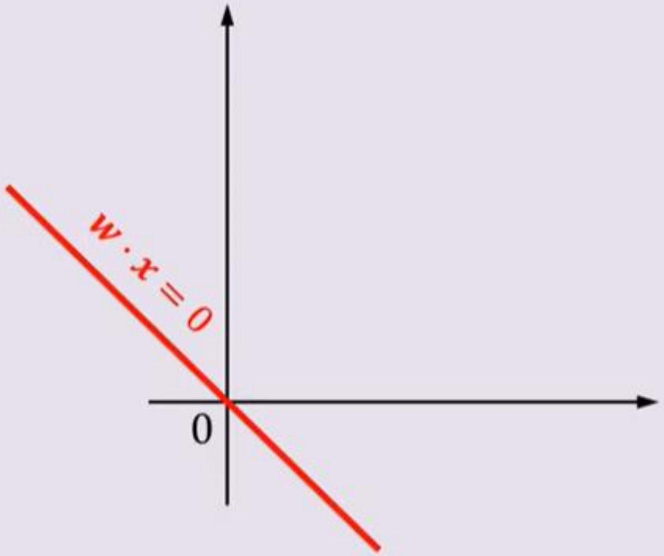
If v is inside the region, $\boldsymbol{w} \cdot v + b \geq 0$

If v is outside the region, $\boldsymbol{w} \cdot v + b < 0$

Affine vs. Linear Transformation



$\boldsymbol{x} \in \mathbf{R}^n$



$\boldsymbol{x} \in \mathbf{R}^{n+1}$

Affine Transformation in R^n

Let $\boldsymbol{w} = [w_1, \cdots, w_n]$ $\quad \boldsymbol{x} = [x_1, \cdots, x_n]^T$

$$\boldsymbol{w} \cdot \boldsymbol{x} + b = 0$$

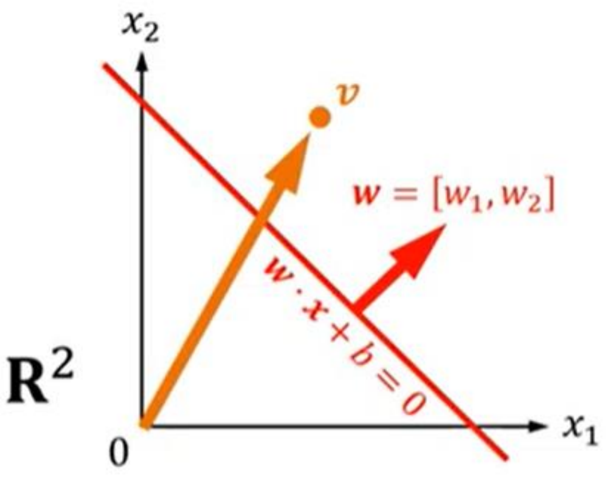
$\Longleftrightarrow$

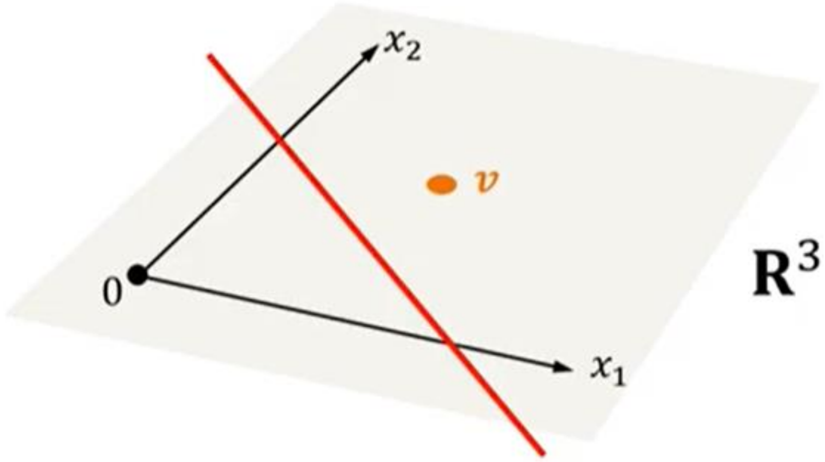
Linear Transformation in R^{n+1}

Let $\boldsymbol{w} = [w_1, \cdots, w_n, b]$ $\quad \boldsymbol{x} = [x_1, \cdots, x_n, x_{n+1}]^T$

$$\begin{cases} \boldsymbol{w} \cdot \boldsymbol{x} = 0 \\ x_{n+1} = 1 \end{cases}$$

From Affine to Linear





Affine Transformation in R^2

Let $\boldsymbol{w} = [w_1, w_2]$ $\quad \boldsymbol{x} = [x_1, x_2]^T$

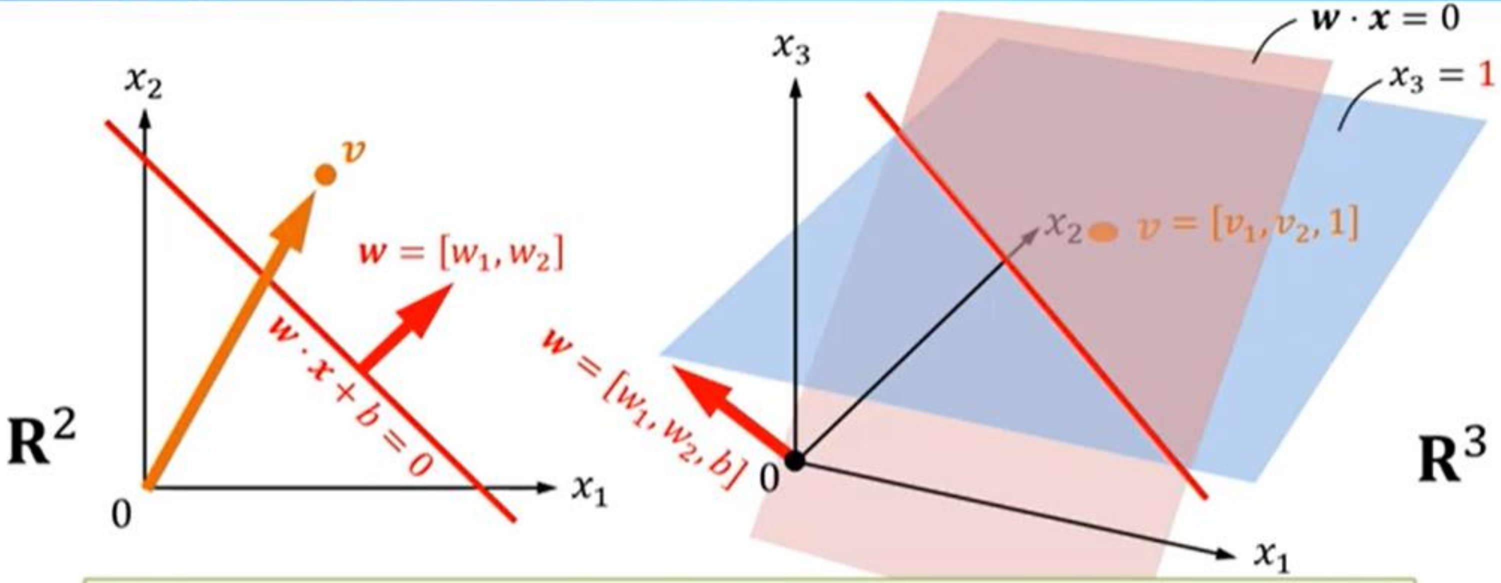
$$\boldsymbol{w} \cdot \boldsymbol{x} + b = 0$$

Linear Transformation in R^3

Let $\boldsymbol{w} = [w_1, w_2, b]$ $\quad \boldsymbol{x} = [x_1, x_2, x_3]^T$

$$\begin{cases} \boldsymbol{w} \cdot \boldsymbol{x} = 0 \\ x_3 = 1 \end{cases}$$

From Affine to Linear



- v is above the red line in 2D $\Leftrightarrow$ v is below the red plane in 3D
- v is below the red line in 2D $\Leftrightarrow$ v is above the red plane in 3D

Affine Transformation in $\mathbf{R}^2$

Let $\boldsymbol{w} = [w_1, w_2]$ $\quad \boldsymbol{x} = [x_1, x_2]^T$

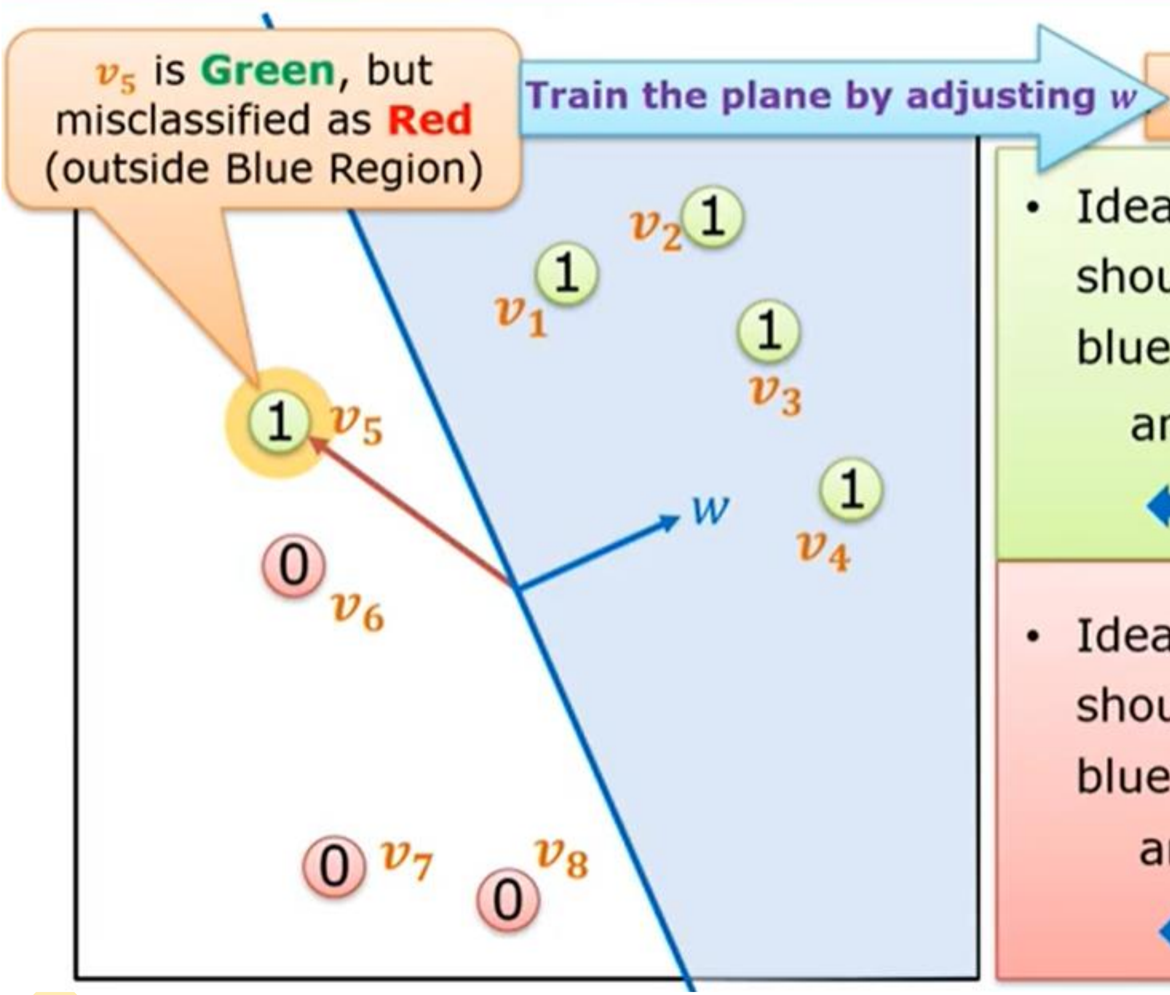
$$\boldsymbol{w} \cdot \boldsymbol{x} + b = 0$$

Linear Transformation in $\mathbf{R}^3$

Let $\boldsymbol{w} = [w_1, w_2, b]$ $\quad \boldsymbol{x} = [x_1, x_2, x_3]^T$

$$\begin{cases} \boldsymbol{w} \cdot \boldsymbol{x} = 0 \\ x_3 = 1 \end{cases}$$

Graphical Interpretation



a plane $w \cdot x = 0$

- Ideally, **green point** v should be inside the blue region, satisfying

  angle $(w, v) \leq 90°$

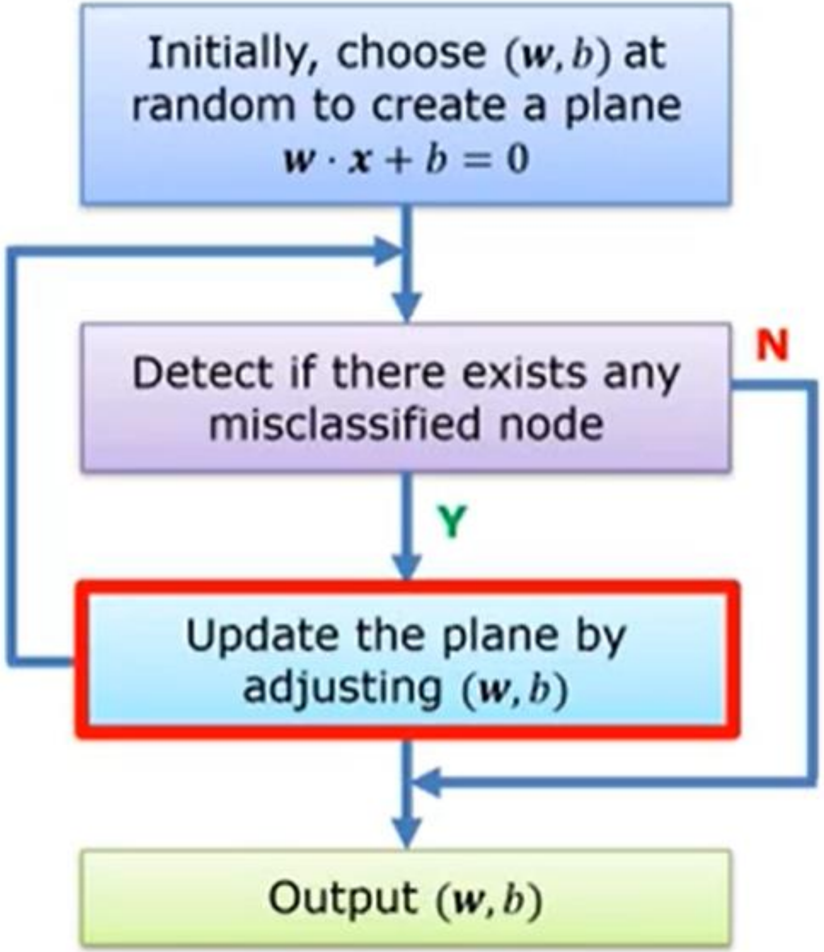
  $\Leftrightarrow \quad w \cdot v \geq 0$

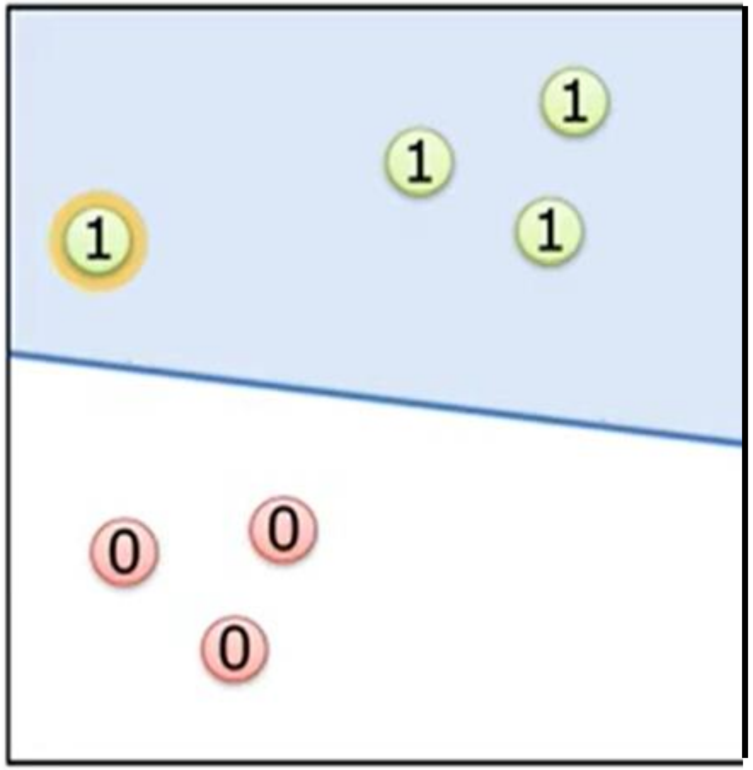
- Ideally, **red point** v should be outside the blue region, satisfying

  angle $(w, v) > 90°$

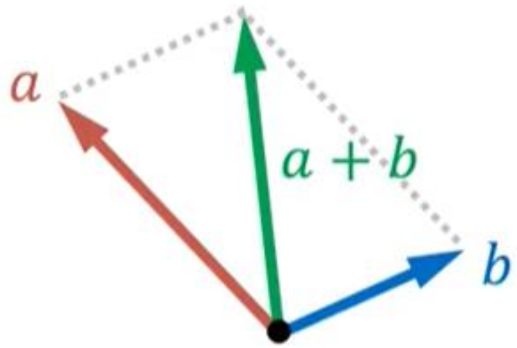
  $\Leftrightarrow \quad w \cdot v < 0$

Key Steps for Perceptron Learning

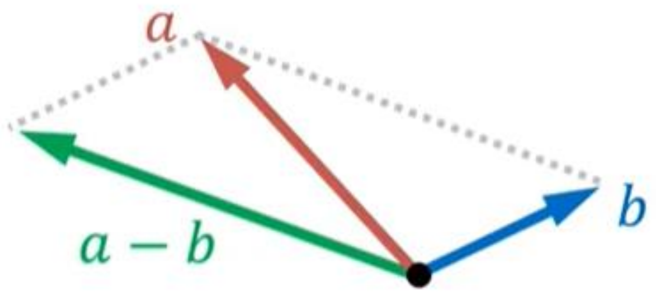




Vector Addition and Subtraction

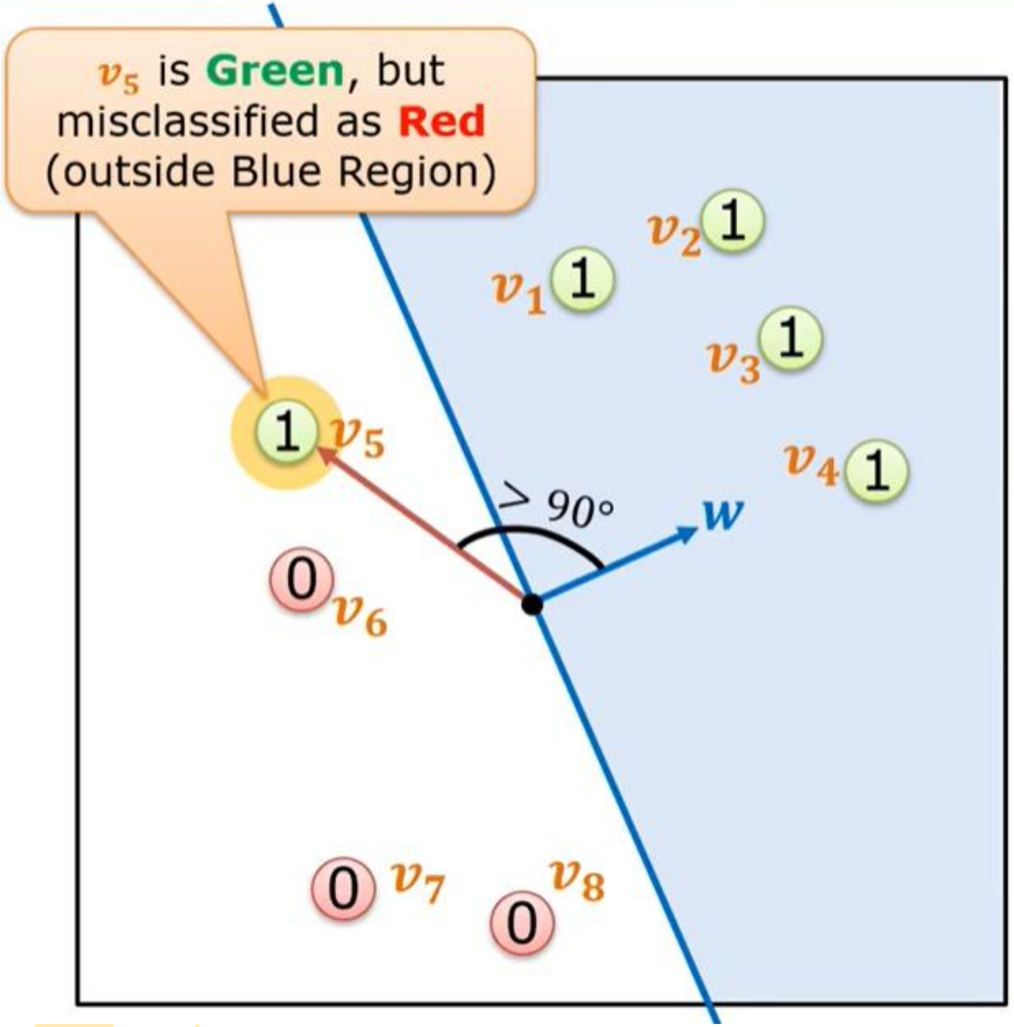


$a + b$



$a - b$

Update w to Learn $w \cdot x = 0$



Learn w for the plane:

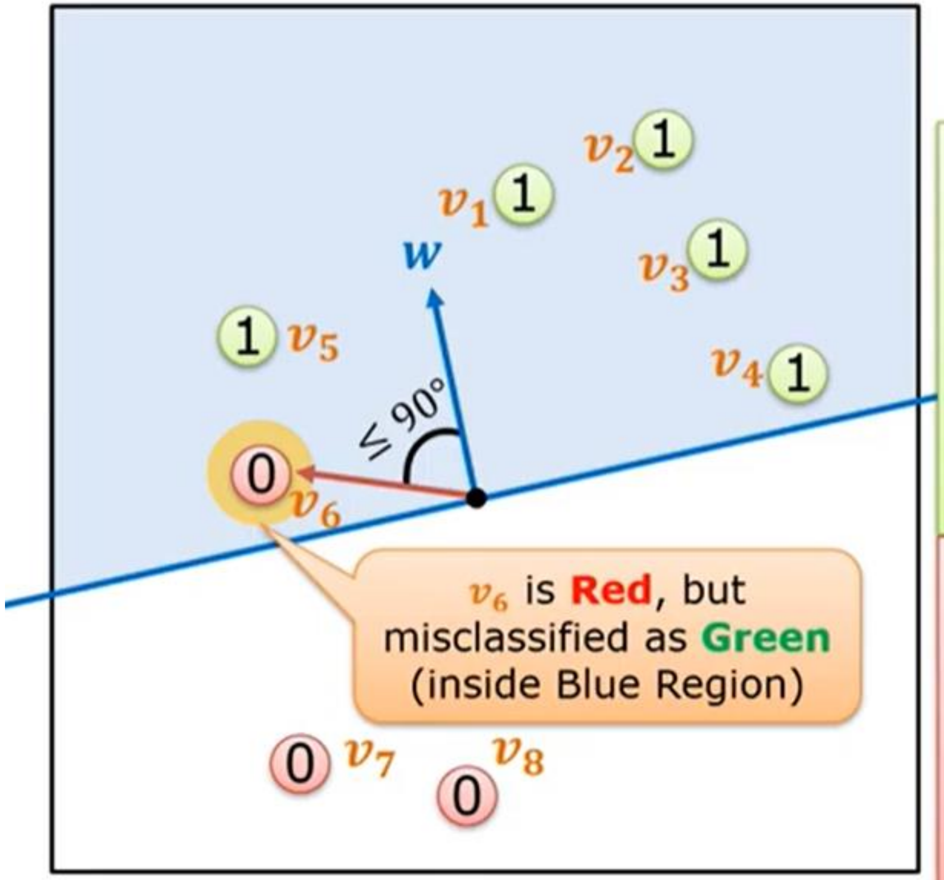
$$w \cdot x = 0$$

For any misclassified v,

- If angle $(w, v) > 90°$ (i.e., v is 1 but misclassified as 0),

$$w_{new} = w_{old} + v$$

Update w to Learn $w \cdot x = 0$



Learn w for the plane:
$w \cdot x = 0$

For any misclassified v,

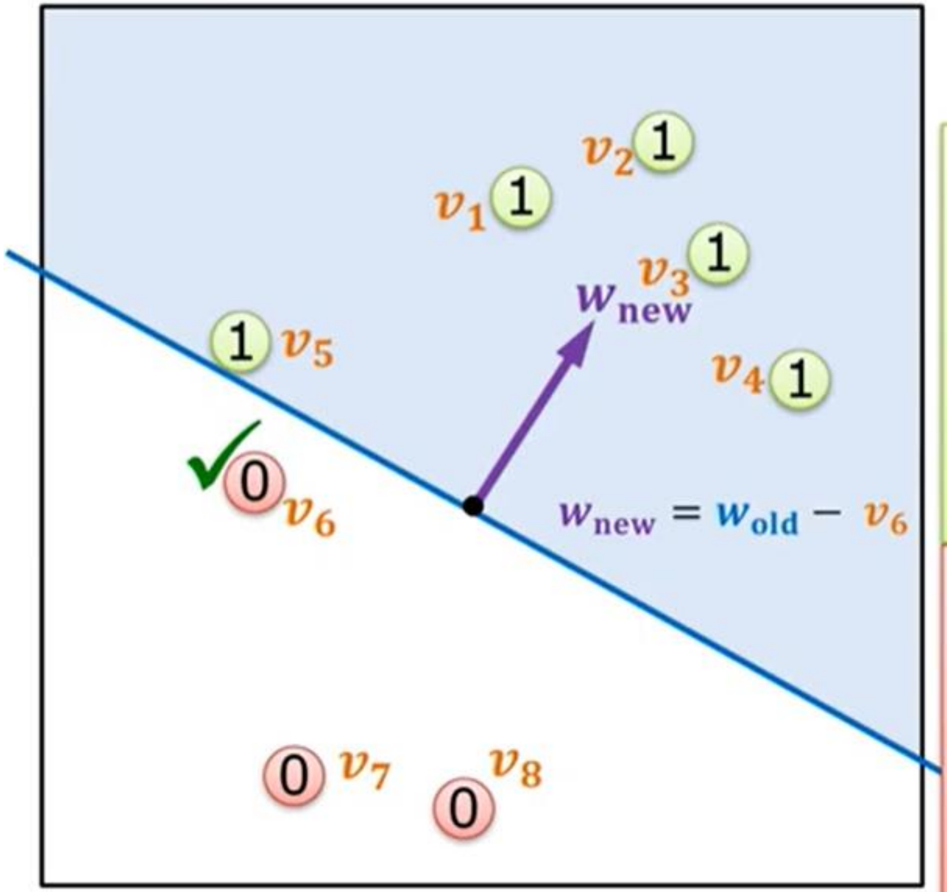
- If angle $(w, v) > 90°$ (i.e., v is 1 but misclassified as 0),

$$w_{\text{new}} = w_{\text{old}} + v$$

- If angle $(w, v) \leq 90°$ (i.e., v is 0 but misclassified as 1),

$$w_{\text{new}} = w_{\text{old}} - v$$

Update w to Learn $w \cdot x = 0$



Learn $\boldsymbol{w}$ for the plane:

$$\boldsymbol{w} \cdot \boldsymbol{x} = 0$$

For any misclassified v,

- If angle $(w, v) > 90°$ (i.e., v is 1 but misclassified as 0),

$$w_{new} = w_{old} + v$$

- If angle $(w, v) \leq 90°$ (i.e., v is 0 but misclassified as 1),

$$w_{new} = w_{old} - v$$

Learning Rule for Perceptron

- Initialise w randomly;
- **while** (there exists a misclassified input v)
- **if** (v is 1 and misclassified as 0) // angle$(w, v) > 90°$
- $w_{new} = w_{old} + v$
- **if** (v is 0 and misclassified as 1) // angle$(w, v) \leq 90°$
- $w_{new} = w_{old} - v$
- **end**

$$(1 - 0) = (\text{actual} - \text{predicted}) = (0 - 1)$$

$$w_{new} = w_{old} + 1 \times v \qquad\qquad w_{new} = w_{old} + (-1) \times v$$

Unified Learning Rule

- Initialise w randomly;
- **while** (there exists a misclassified input v)

$$\boxed{w_{\mathrm{new}} = w_{\mathrm{old}} + (y(v) - \hat{y}(v)) \times v}$$

- **end**

$$w_{\mathrm{new}} = w_{\mathrm{old}} + (\text{actual} - \text{predicted}) \times v$$

$y(v)$ $\quad$ $\hat{y}(v)$

Learning Rate

- Initialise w randomly;
- **while** (there exists a misclassified input v)

$$w \leftarrow w + \alpha \times (y(v) - \hat{y}(v)) \times v$$

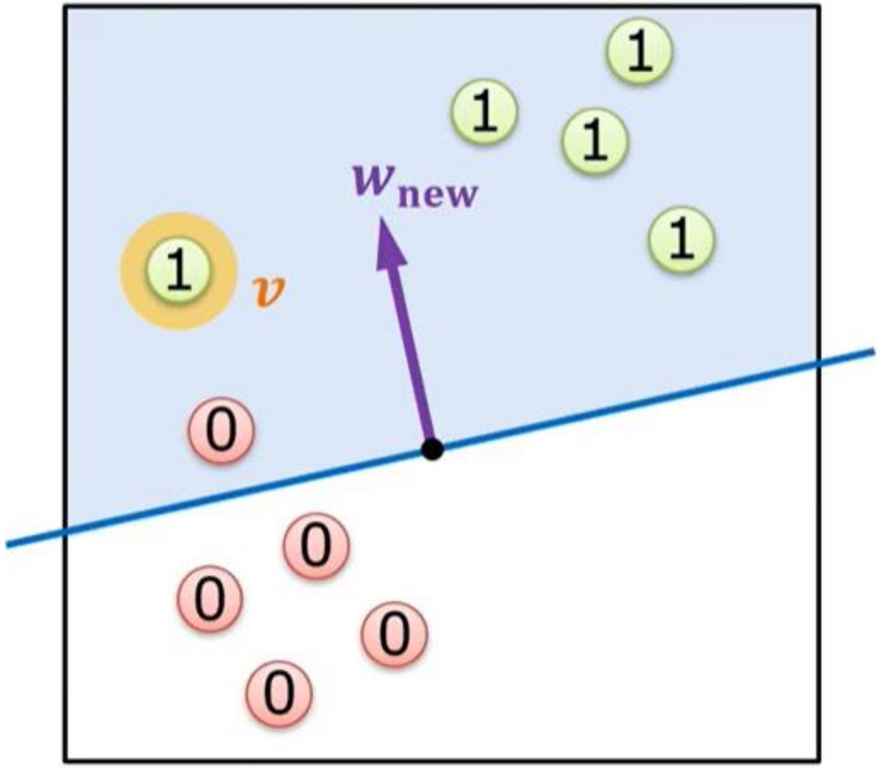
learning rate $(0 < \alpha < 1)$

- **end**

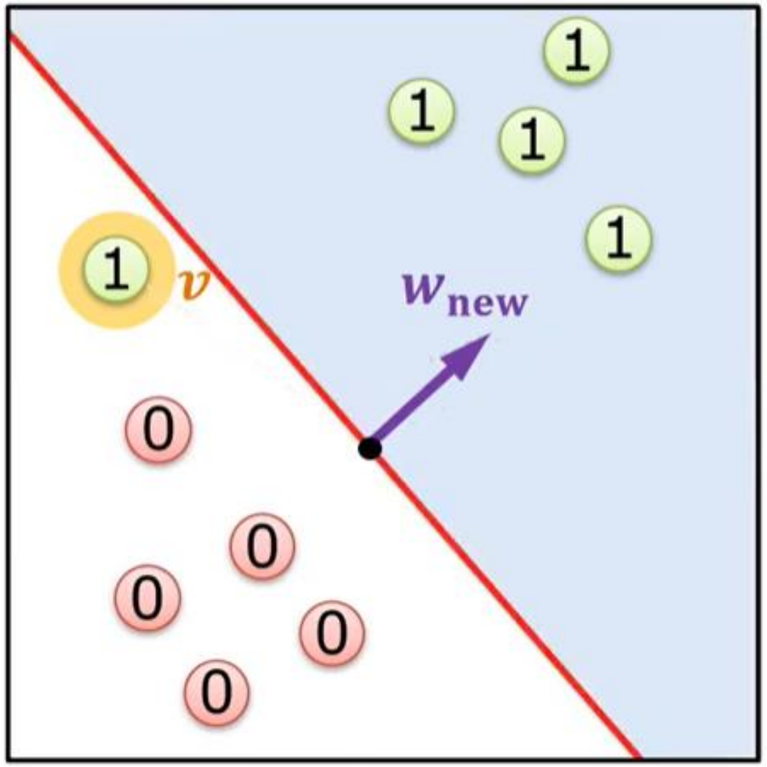
- Learning rate α is a configurable hyperparameter, which determines the step size at each iteration in the training.
- α is a speed-accuracy trade-off:
  - Smaller α requires more training iterations (epochs), leading to slow rate of convergence.
  - Larger α results in rapid changes that may make the learning jump over (overshoot) the optima.

Effects of Learning Rate

$$w_{\text{new}} = w_{\text{old}} + \alpha \times v$$



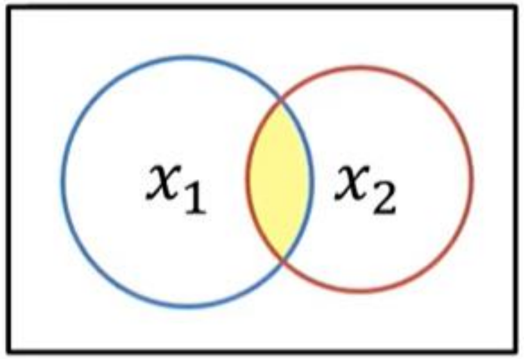
$\alpha = 1$



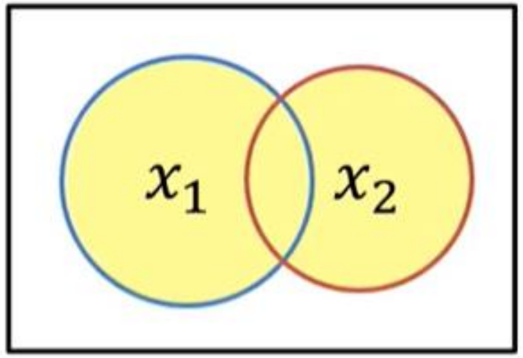
$\alpha = 0.1$

- **Why XOR is linear inseparable?**
- **Remedies for XOR Problem**
  - Replace Threshold Function
  - Use Multi-Layer Perceptron (MLP)
- **Complex Decision Boundaries through MLP Composition**

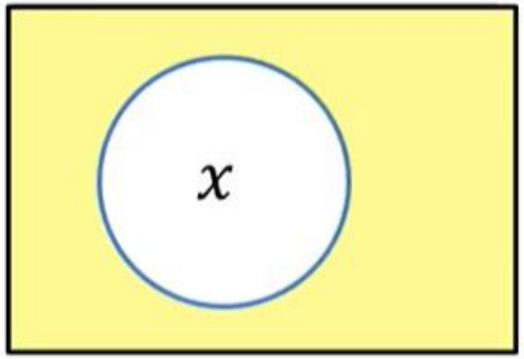
Logic Gates as Venn Diagrams



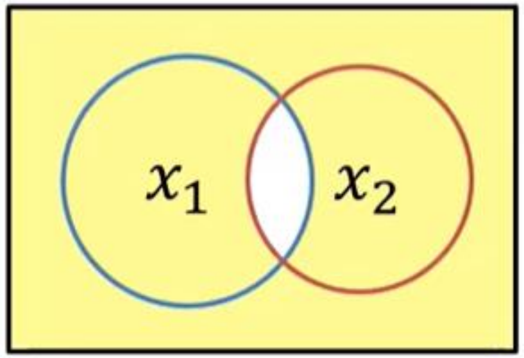
x_1 **AND** x_2



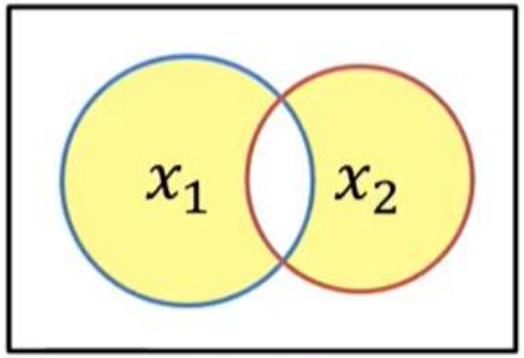
x_1 **OR** x_2



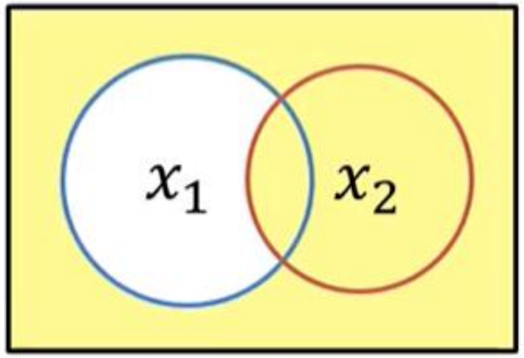
NOT x



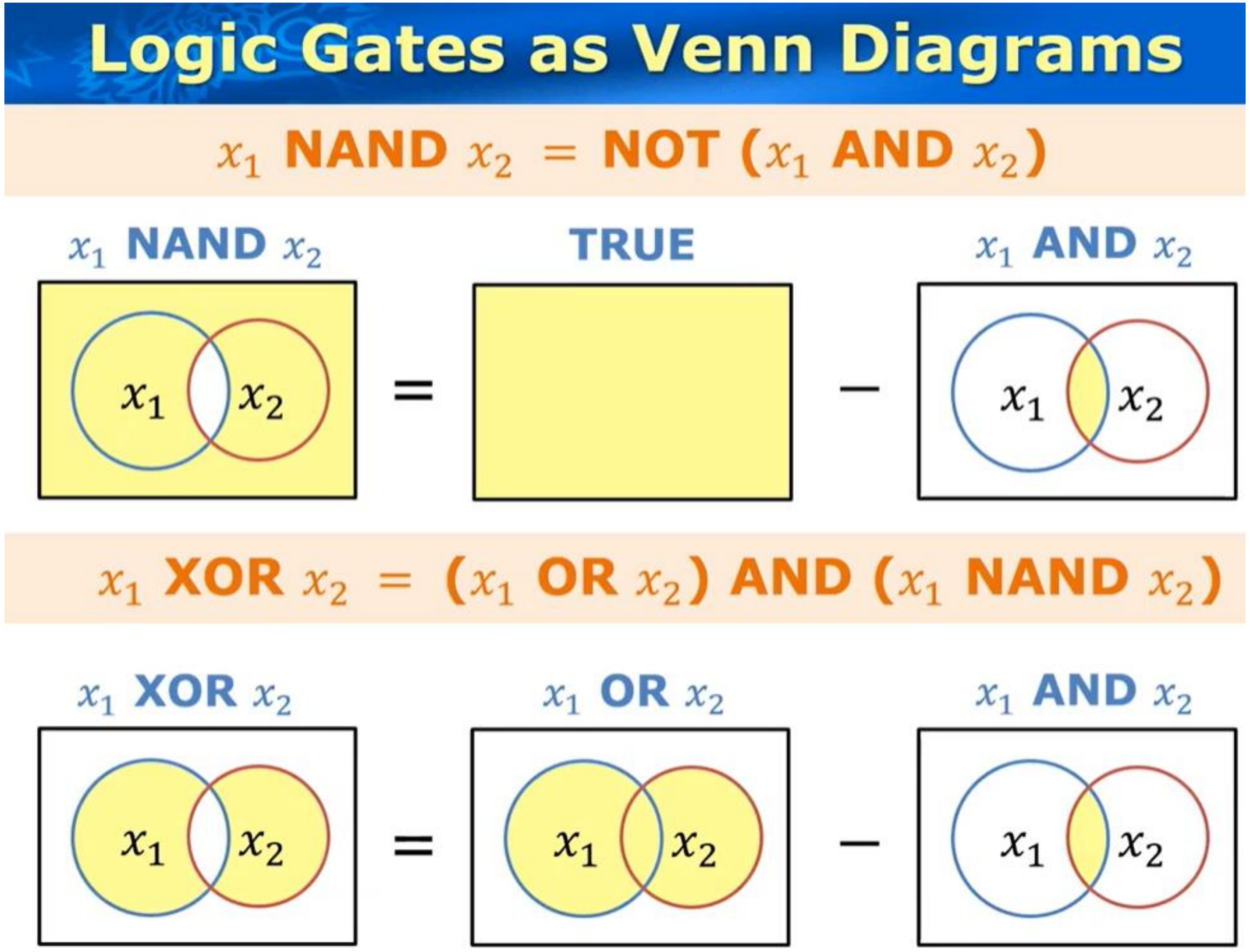
x_1 **NAND** x_2

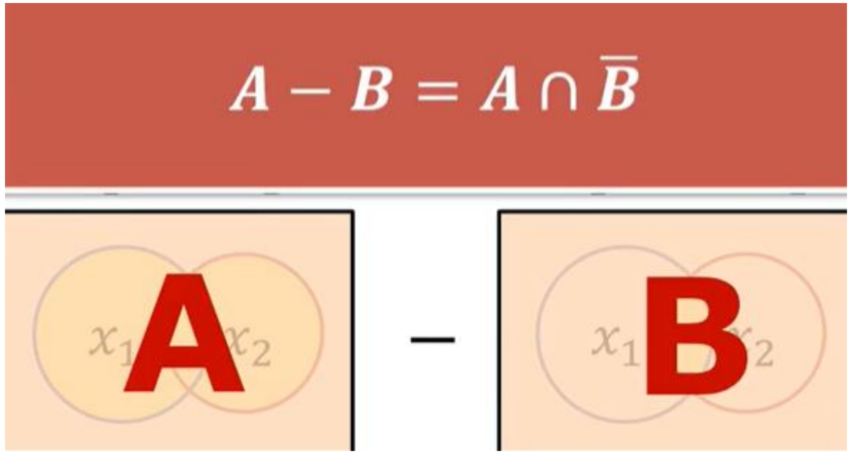


x_1 **XOR** x_2

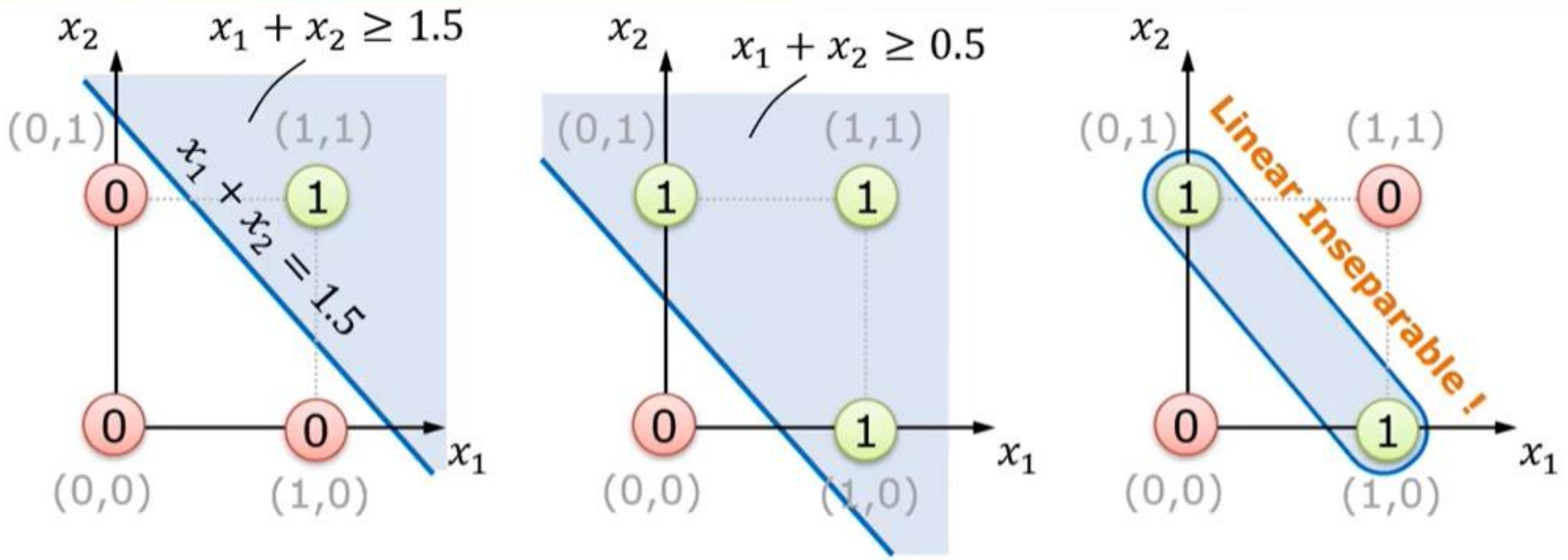


x_1 **IMPLY** x_2


Logic Gates as Venn Diagrams
x1 NAND x2 = NOT (x1 AND x2)
x1 NAND x2
TRUE
x1 AND x2
x1
x2
=
−
x1 XOR x2 = (x1 OR x2) AND (x1 NAND x2)
x1 XOR x2
x1 OR x2
x1 AND x2


A − B = A ∩ B̄
A
−
B

Linear Separability



x_1 AND x_2

x_1	x_2	y
1	1	1
0	1	0
1	0	0
0	0	0

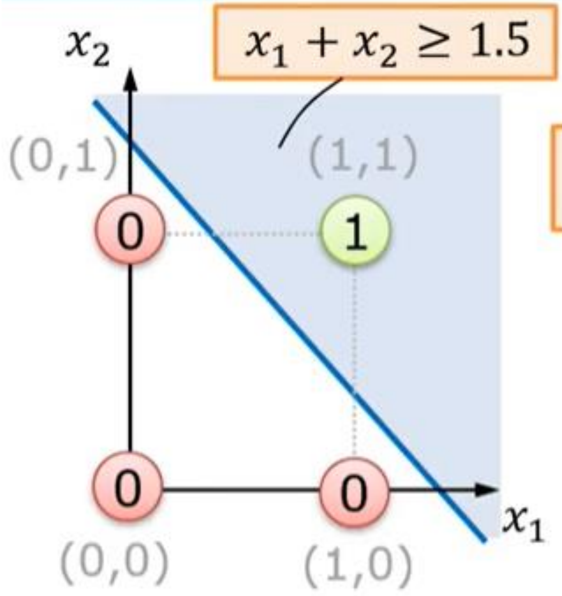
x_1 OR x_2

x_1	x_2	y
1	1	1
0	1	1
1	0	1
0	0	0

x_1 XOR x_2

x_1	x_2	y
1	1	0
0	1	1
1	0	1
0	0	0

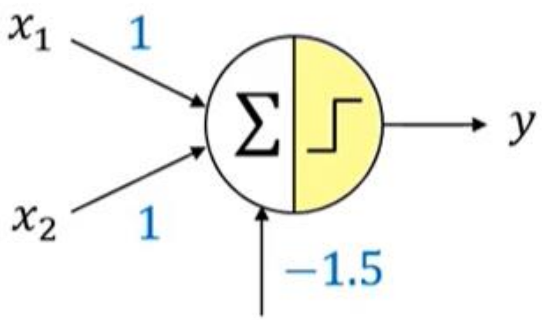
Perceptron for AND Gate



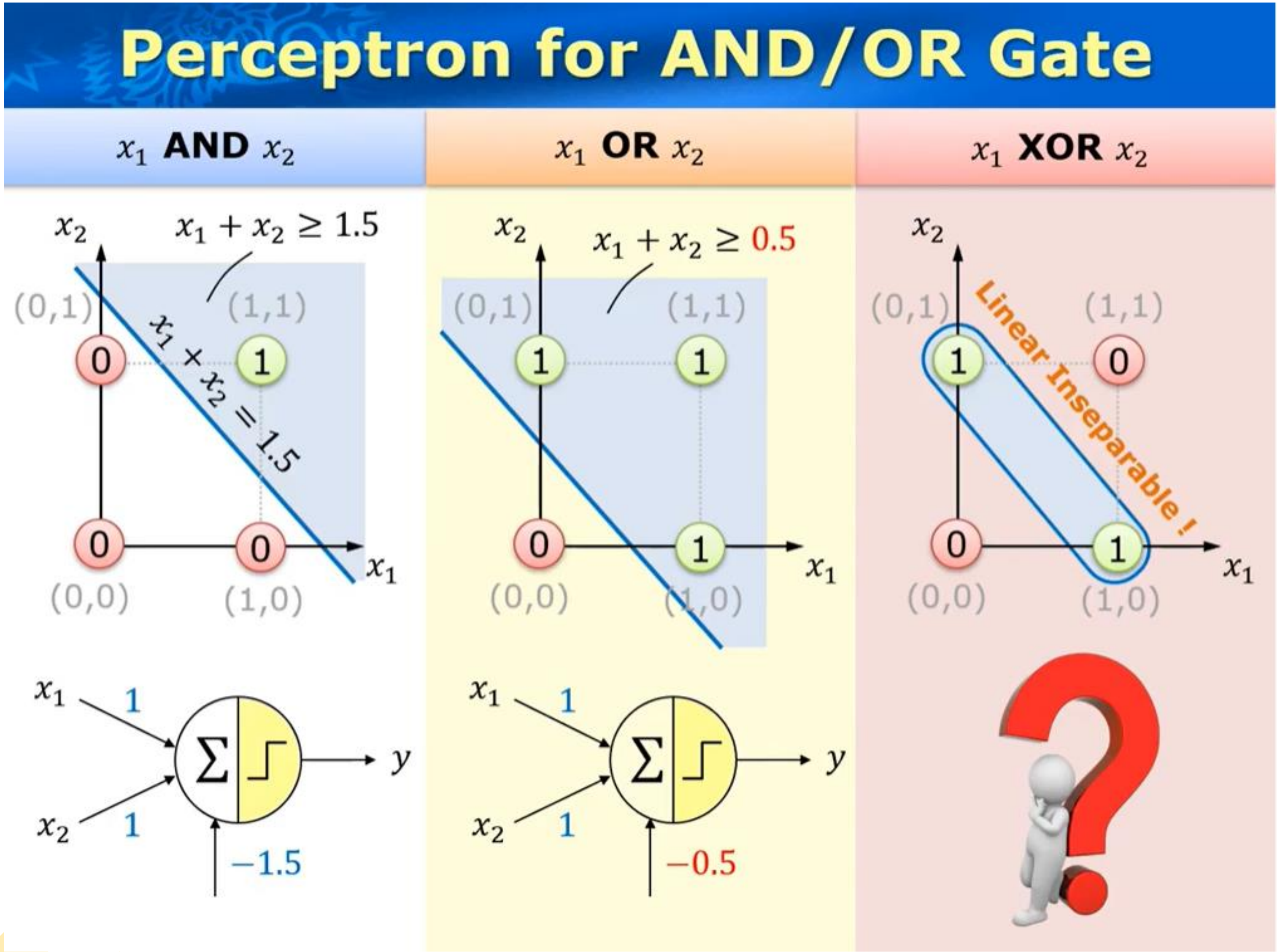
x_1 AND x_2

$$1 \cdot x_1 + 1 \cdot x_2 - 1.5 \geq 0$$

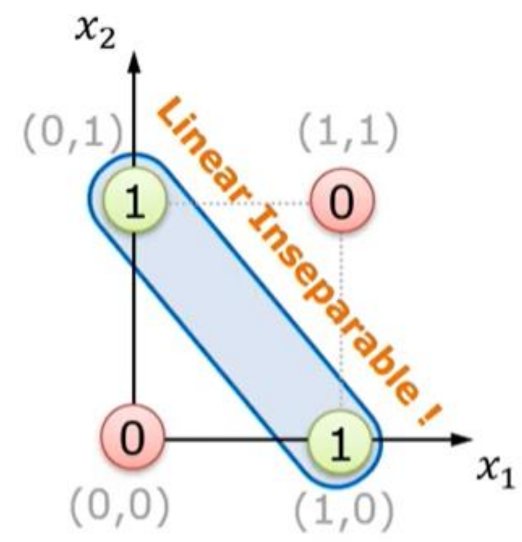
$$w_1 x_1 + w_2 x_2 + b \geq 0 \Rightarrow (w_1, w_2, b) = (1, 1, -1.5)$$



$$y = \begin{cases} 1 & \text{if } w_1 x_1 + w_2 x_2 + b \geq 0 \\ 0 & \text{otherwise} \end{cases}$$


Perceptron for AND/OR Gate
x1 AND x2
x1 OR x2
x1 XOR x2
x2
x1 + x2 ≥ 1.5
x1 + x2 = 1.5
(0,1)
(1,1)
(0,0)
(1,0)
x1
x2
1
1
−1.5
y
x1 + x2 ≥ 0.5
−0.5
Linear Inseparable !

XOR is Not Linear Separable



x_1	x_2	y	
0	0	0	$w_1 \cdot 0 + w_2 \cdot 0 + b < 0$
0	1	1	$w_1 \cdot 0 + w_2 \cdot 1 + b \geq 0$
1	1	0	$w_1 \cdot 1 + w_2 \cdot 1 + b < 0$
1	0	1	$w_1 \cdot 1 + w_2 \cdot 0 + b \geq 0$

$$y = \begin{cases} 1 & \text{if } w_1 x_1 + w_2 x_2 + b \geq 0 \\ 0 & \text{if } w_1 x_1 + w_2 x_2 + b < 0 \end{cases}$$

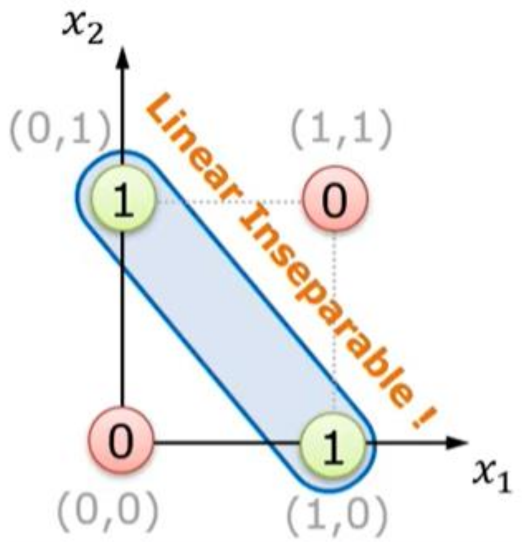
$$\begin{cases} \boxed{b < 0} \\ w_2 + b \geq 0 \\ w_1 + w_2 + b < 0 \\ \boxed{w_1 + b \geq 0} \end{cases}$$

$w_2 + b \geq 0$, $w_1 + w_2 + b < 0$ → $\boxed{w_1 < 0}$

Contradiction!

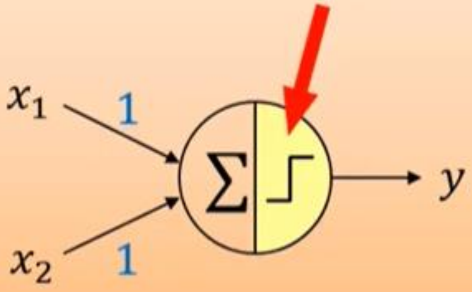
No Solution for such $(w_1,\ w_2,\ b)$!

Remedies for XOR Problem



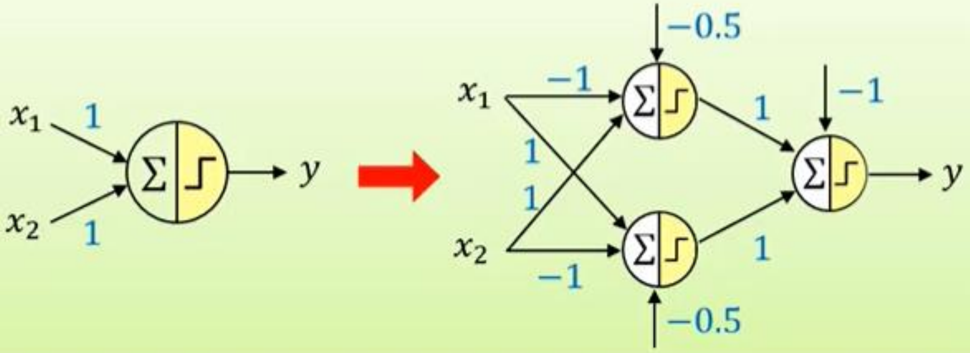
Treatment Method 1

Replace the existing threshold function with a more powerful function.



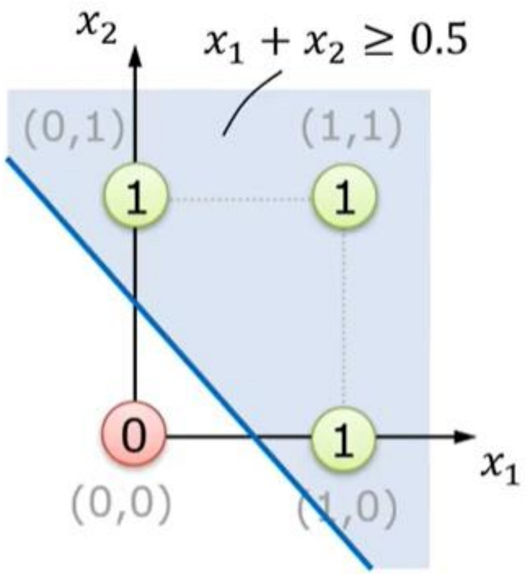
Treatment Method 2

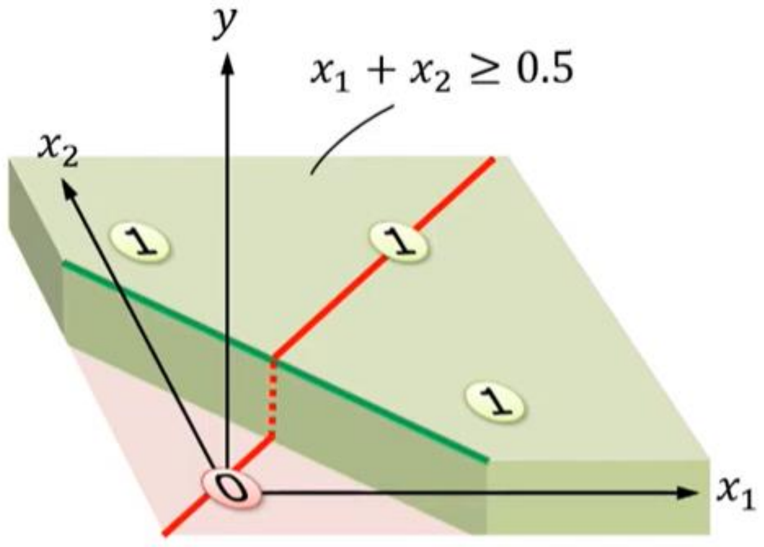
Increase the number of layers while keeping each unit using a threshold function.

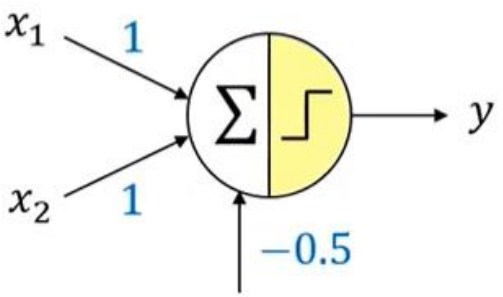


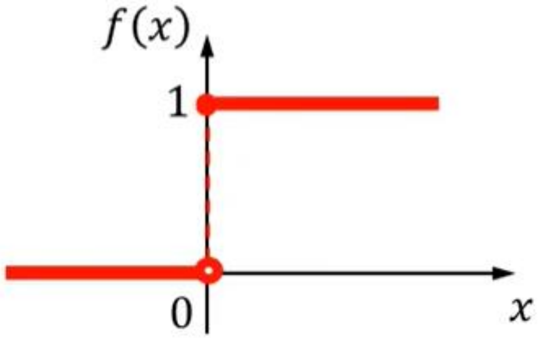
Recap: Threshold Function for OR Gate

x_1 OR x_2

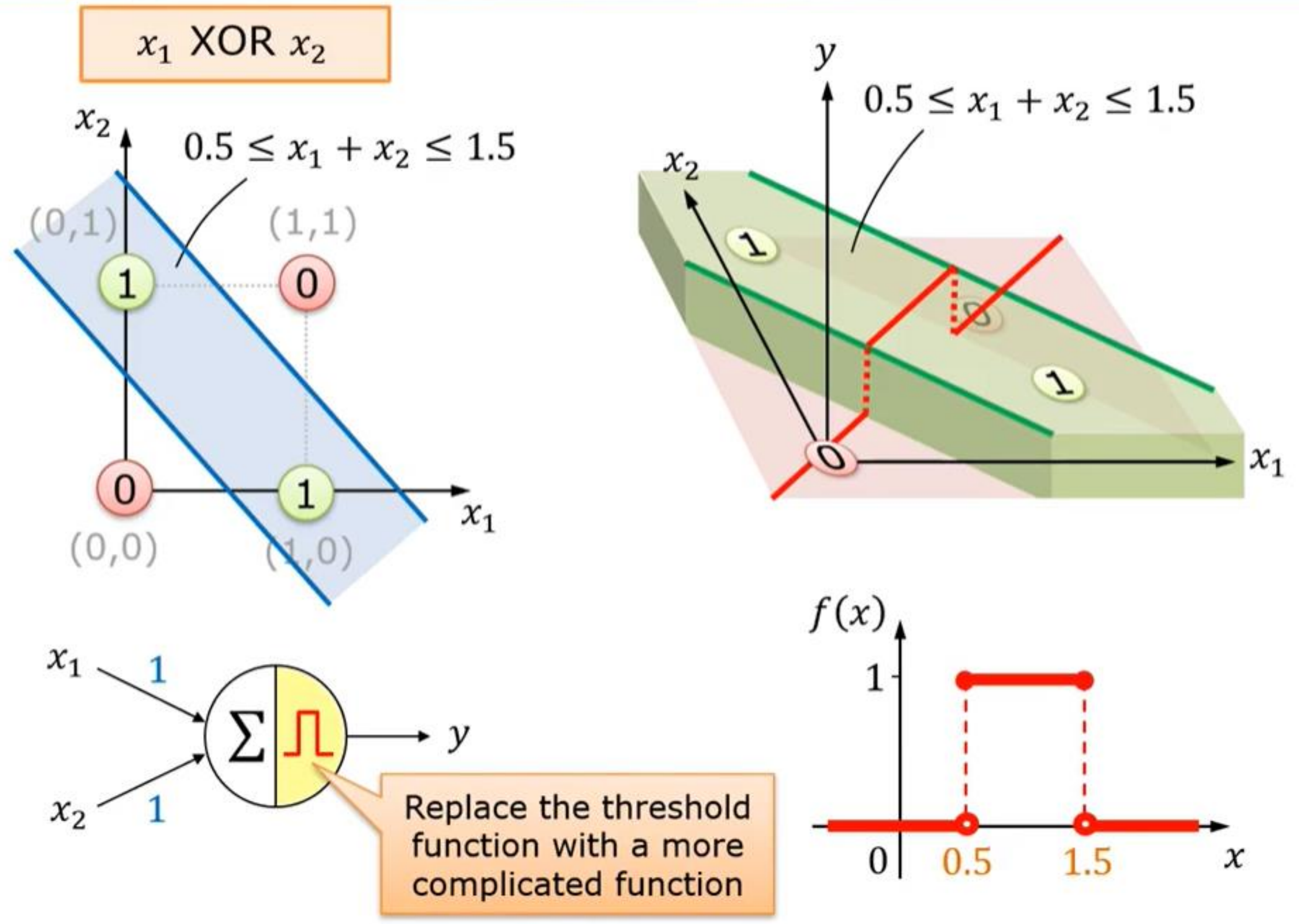




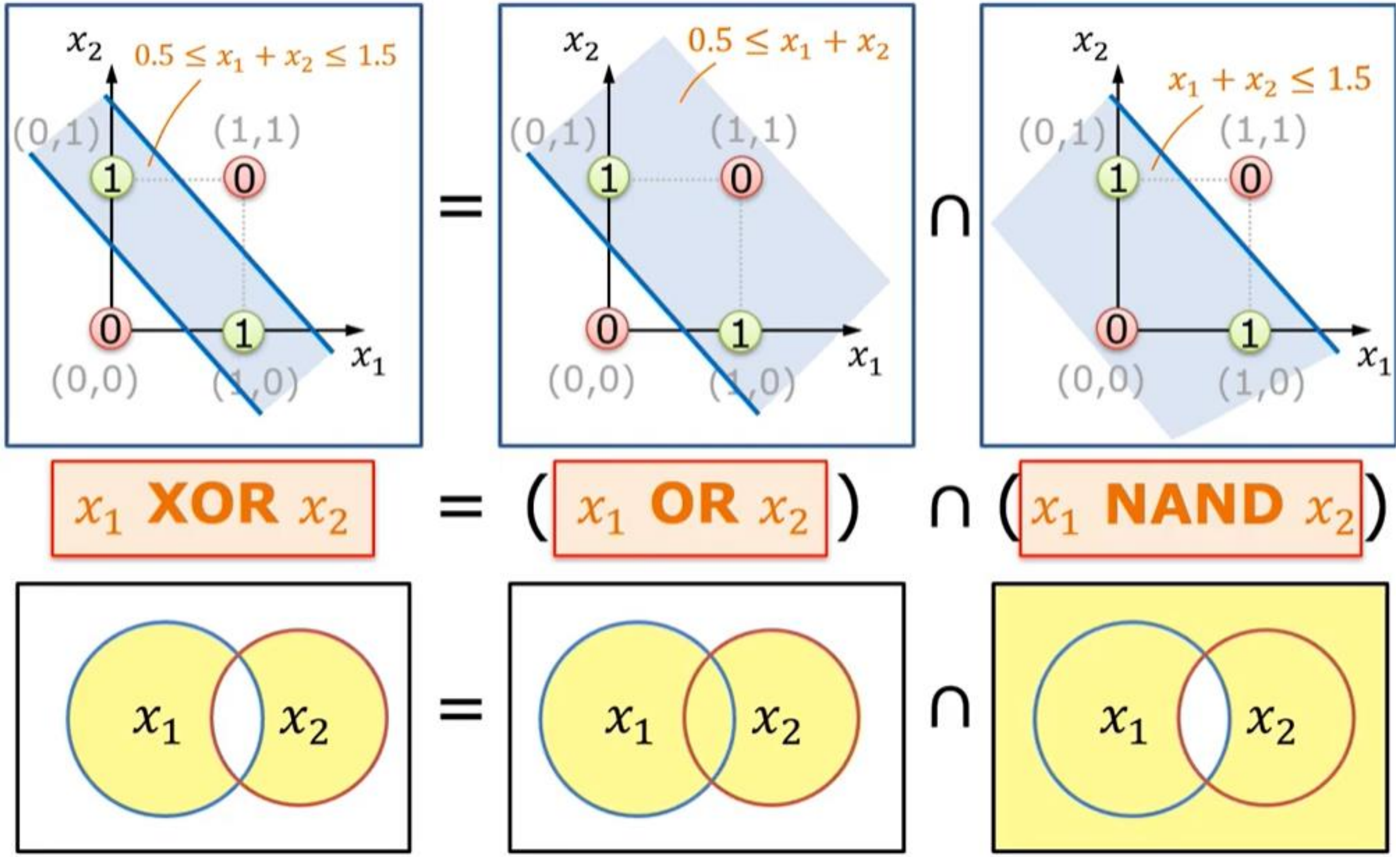




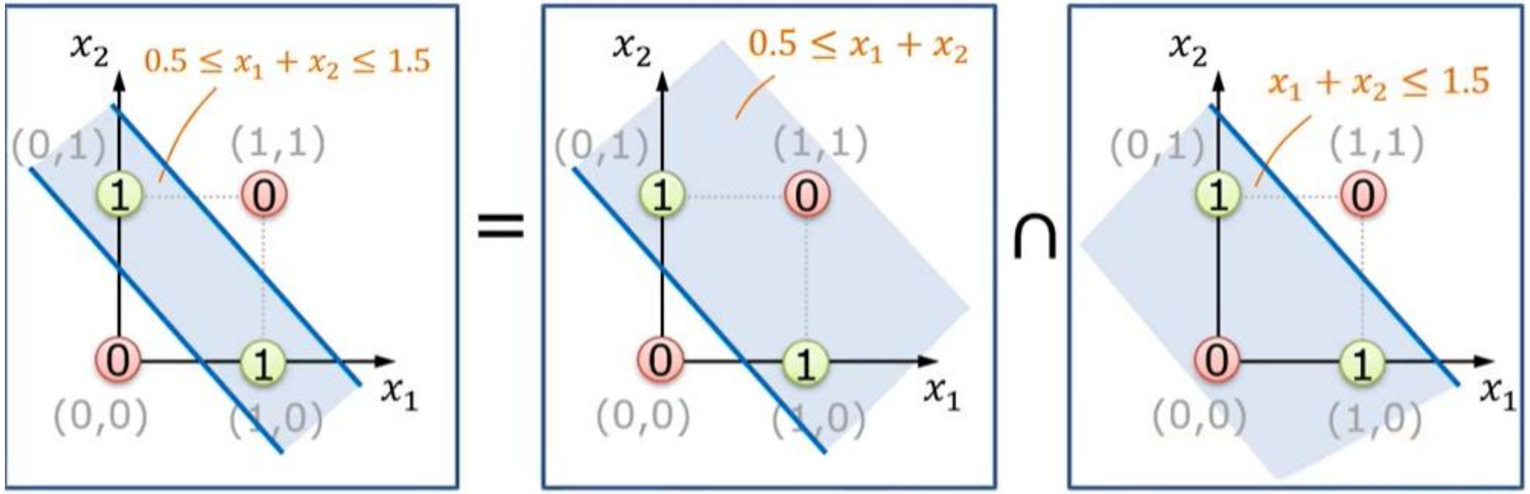
Method 1 – Replace Threshold Function



Method 2 – Increase Layers of Perceptron



Method 2 – Increase Layers of Perceptron

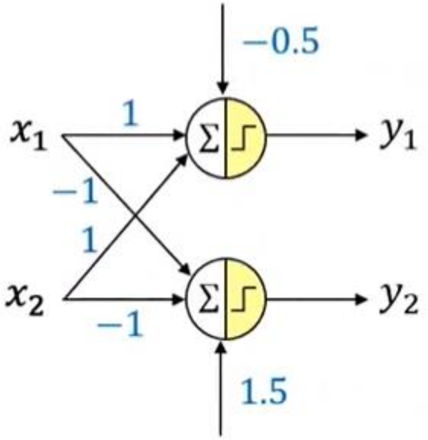


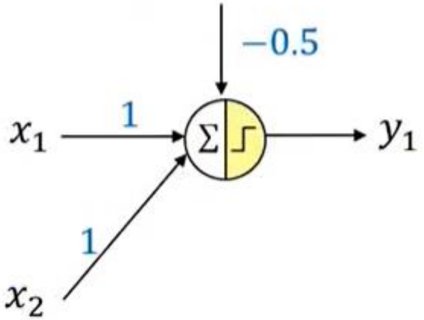
$x_1 + x_2 - 0.5 \ge 0$

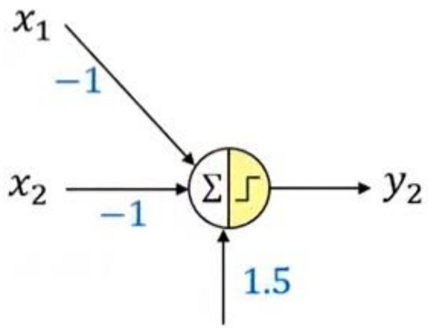
$\therefore (w_1, w_2, b) = (1, 1, -0.5)$

$-x_1 - x_2 + 1.5 \ge 0$

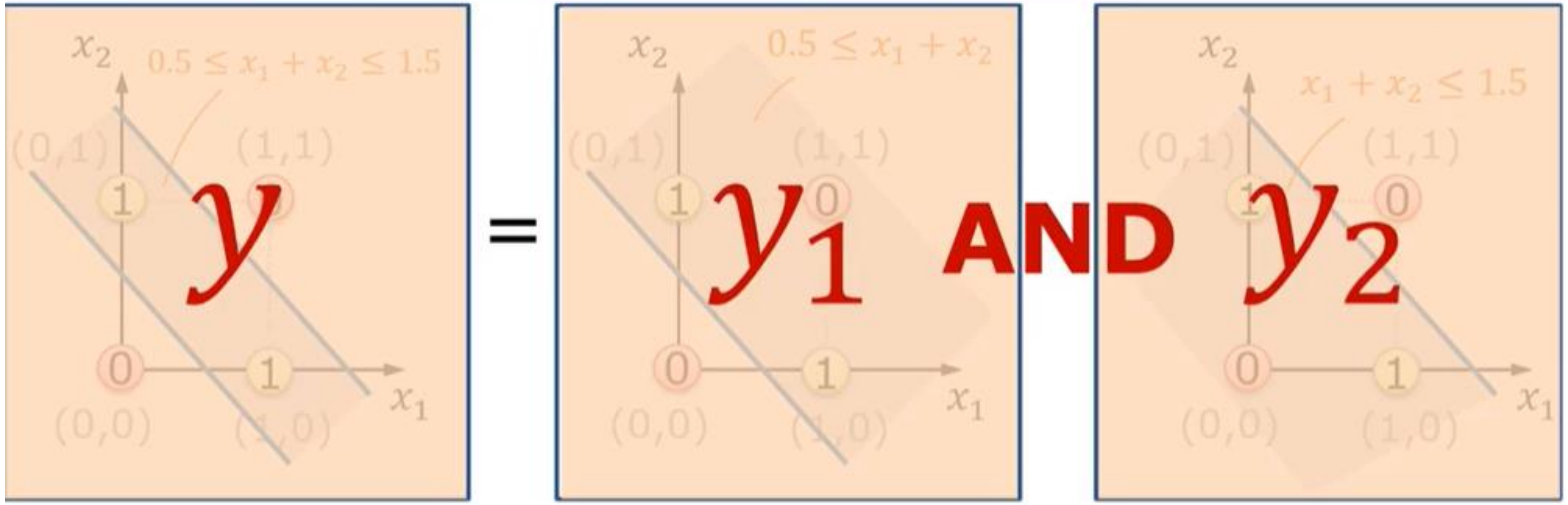
$\therefore (w_1, w_2, b) = (-1, -1, 1.5)$

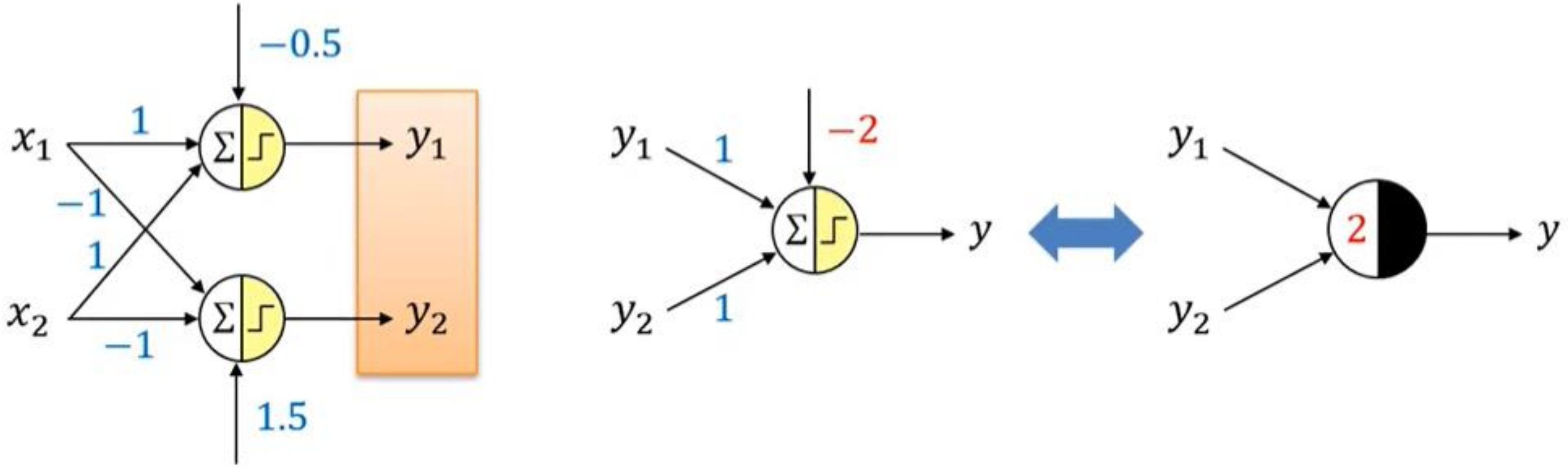




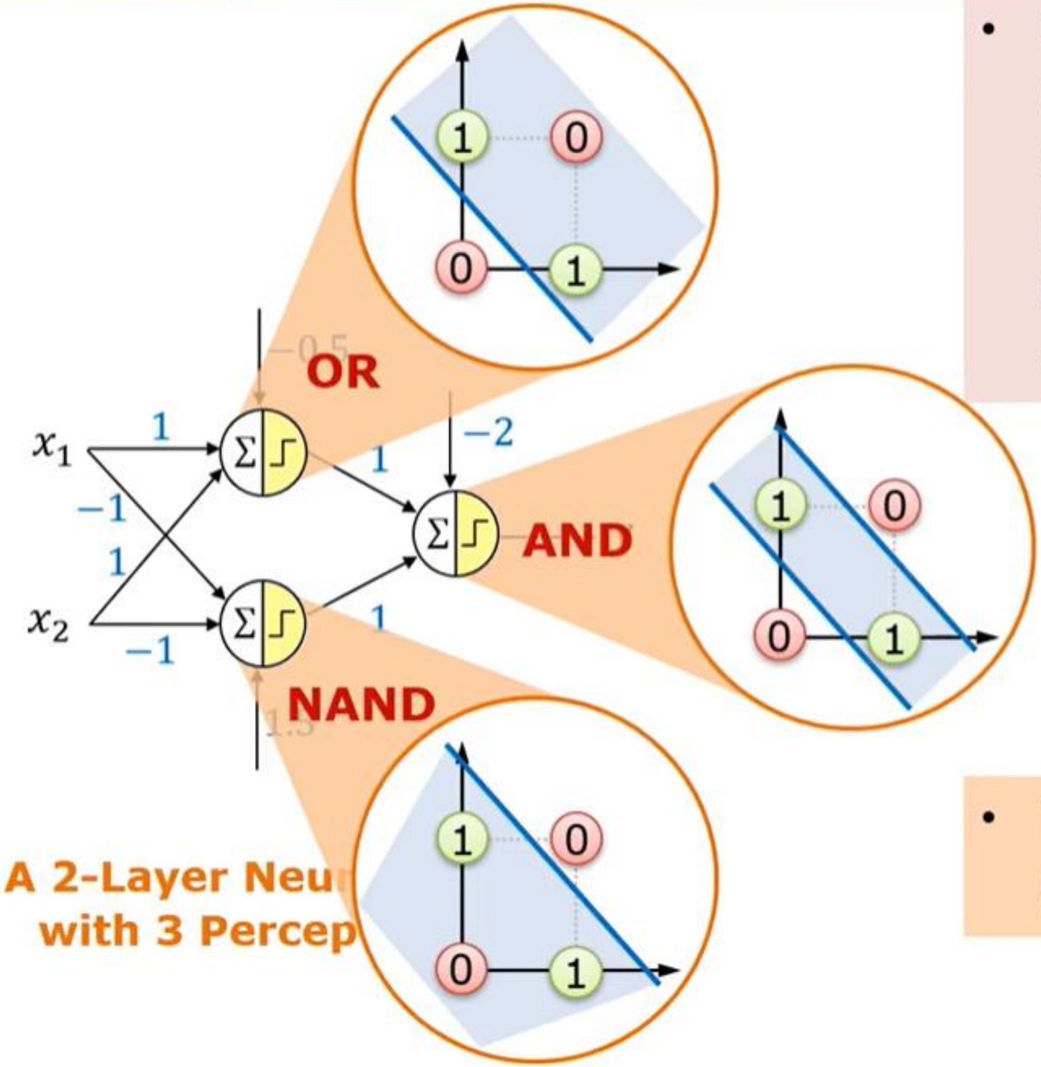


Method 2 – Increase Layers of Perceptron





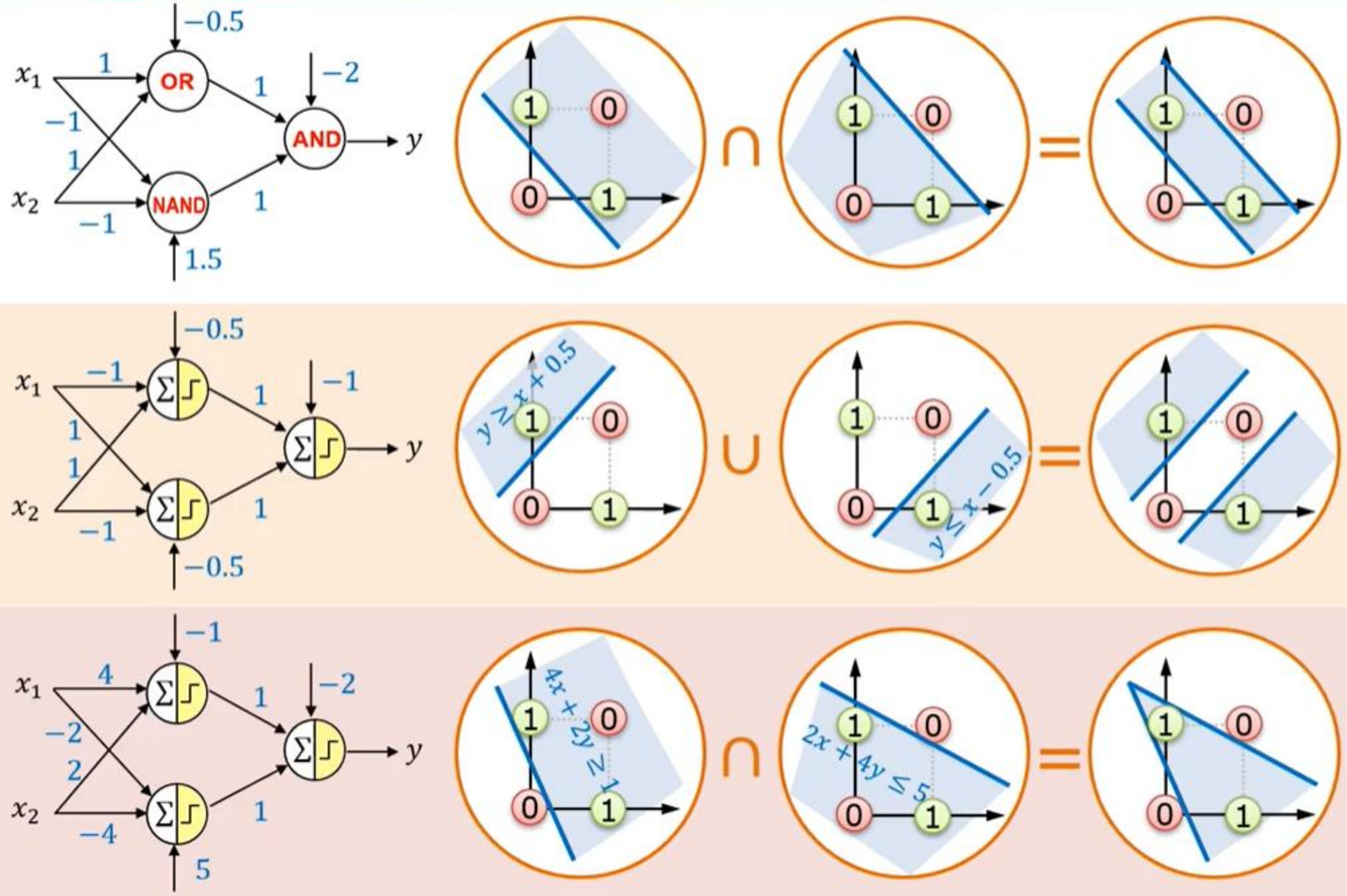
Multi-Layer Perceptron (MLP)



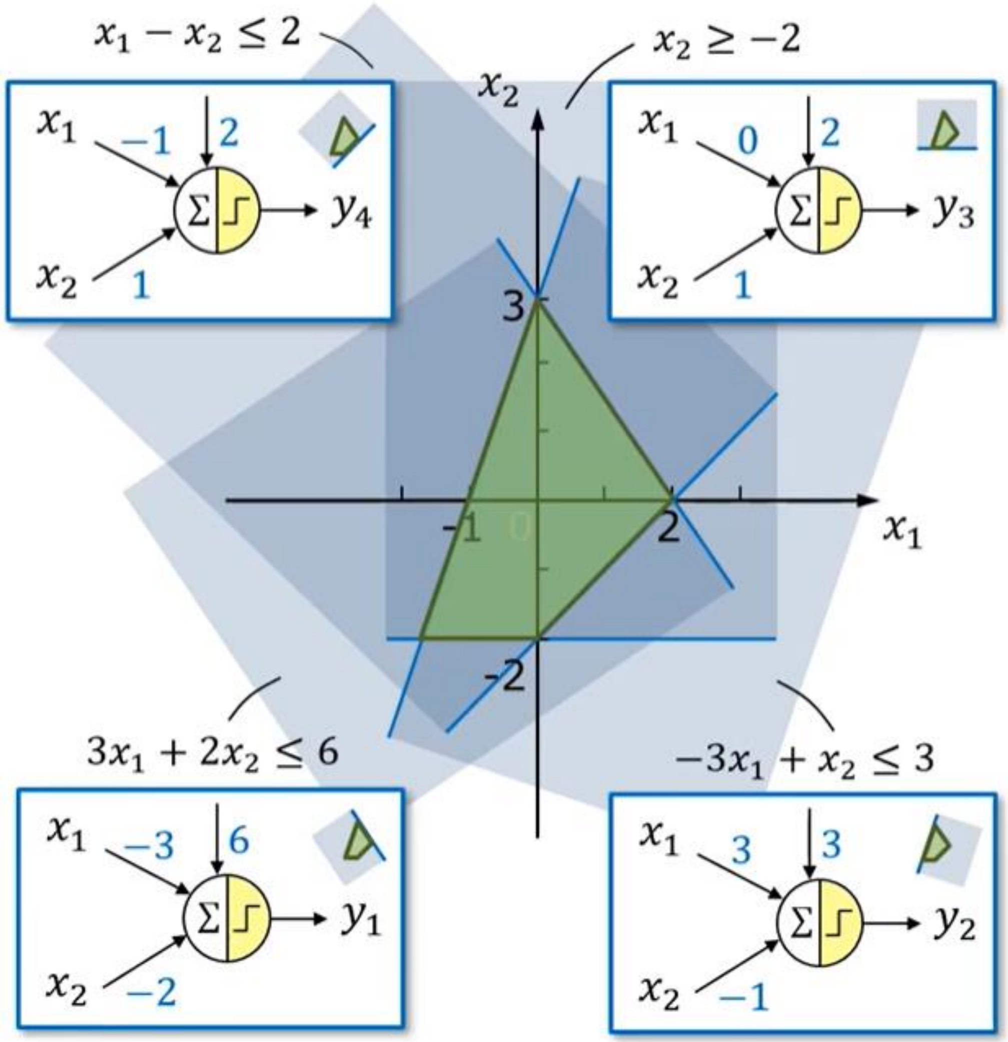
- Minsky and Papert (1969) provided a solution to the XOR problem by combining three perceptron units using a hidden layer.

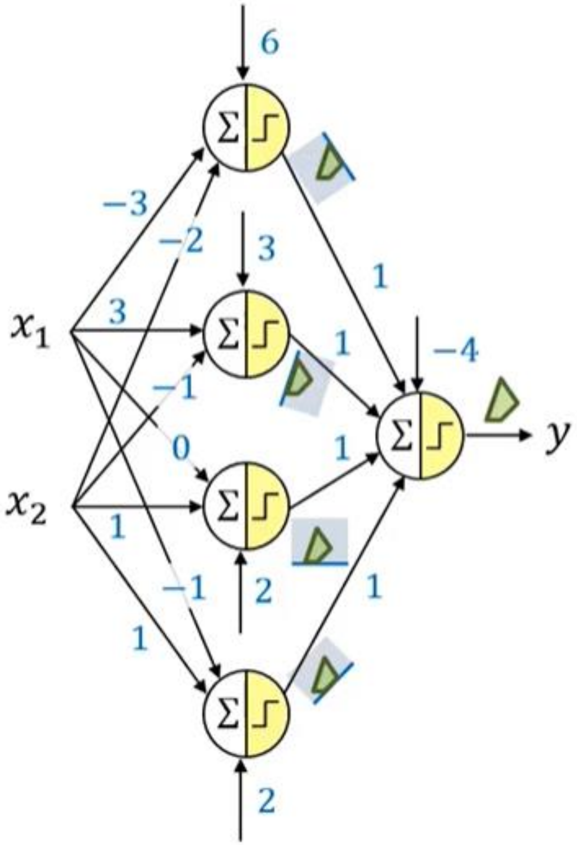
- The solution to XOR gate is not unique.

MLP for Emulating XOR is NOT Unique

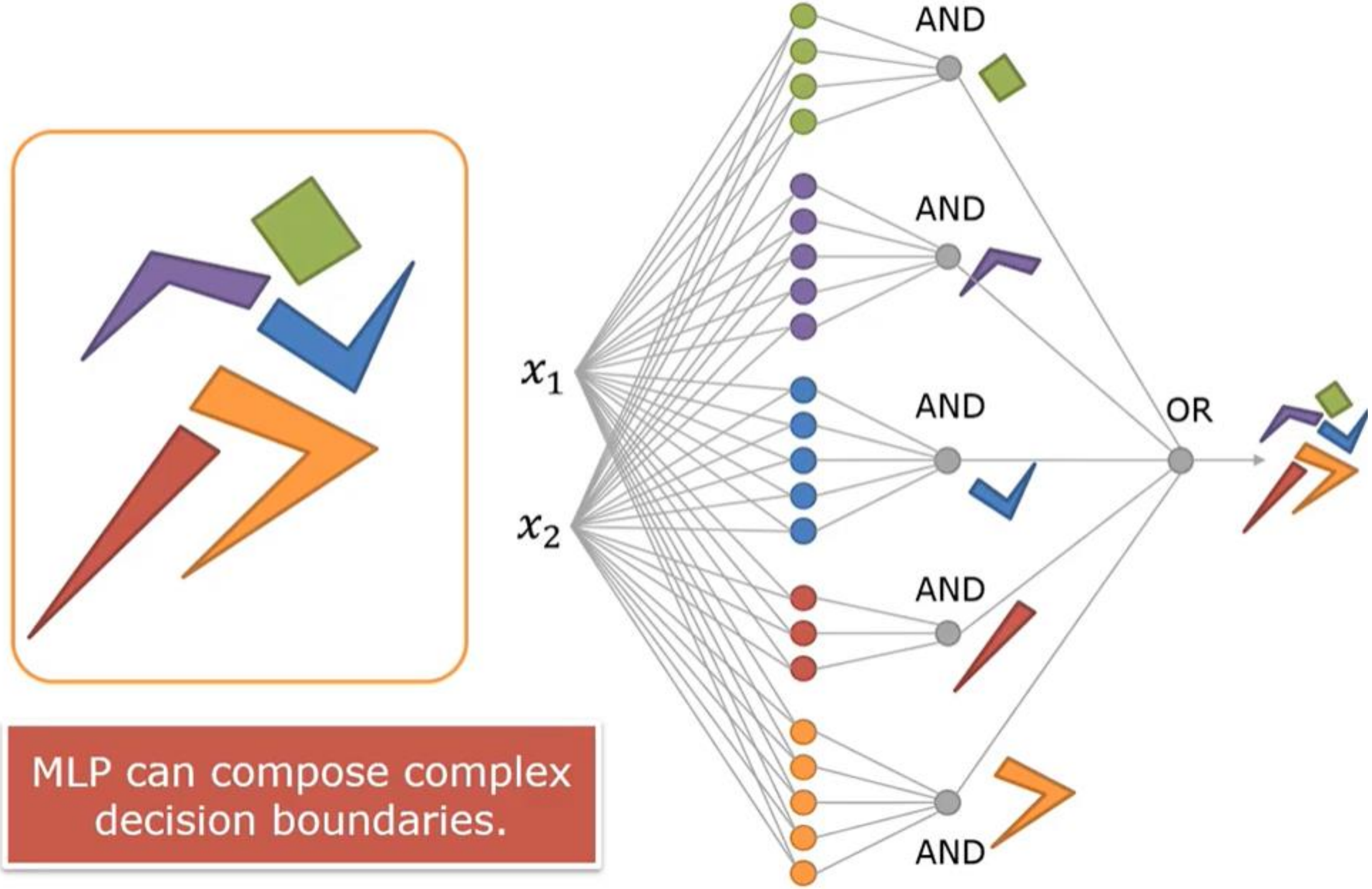


Decision Boundary of MLP



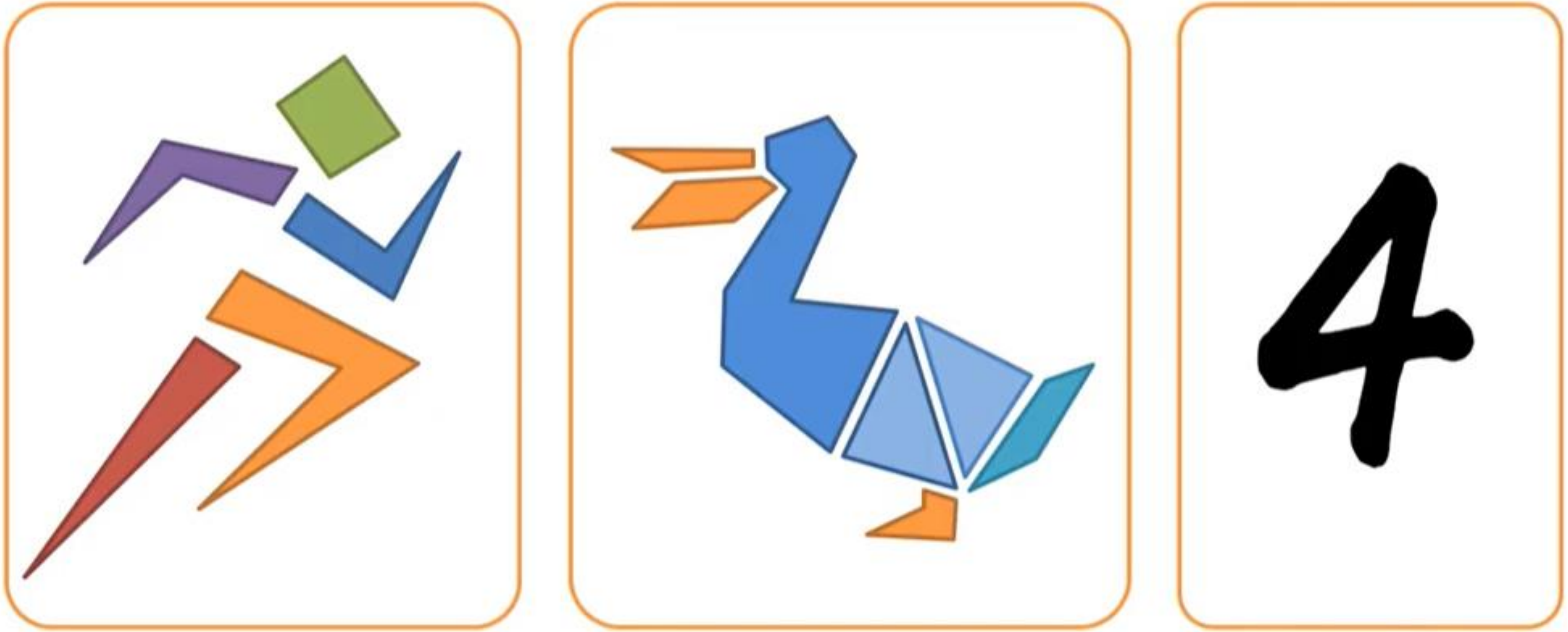


Complex Decision Boundaries



MLP can compose complex decision boundaries.

Complex Decision Boundaries



MLP can compose complex decision boundaries, which lays the foundations for image recognition.