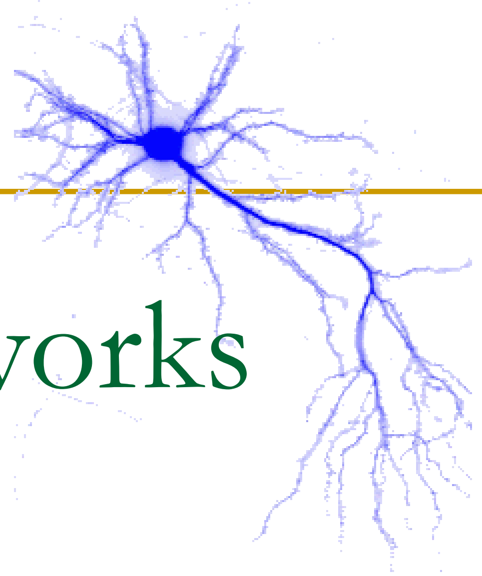
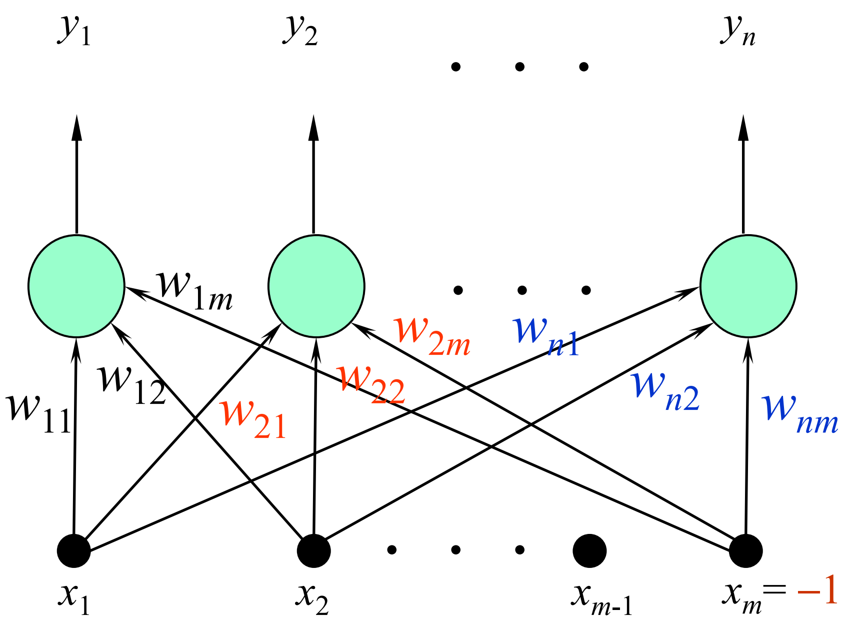
# Feed-Forward Neural Networks



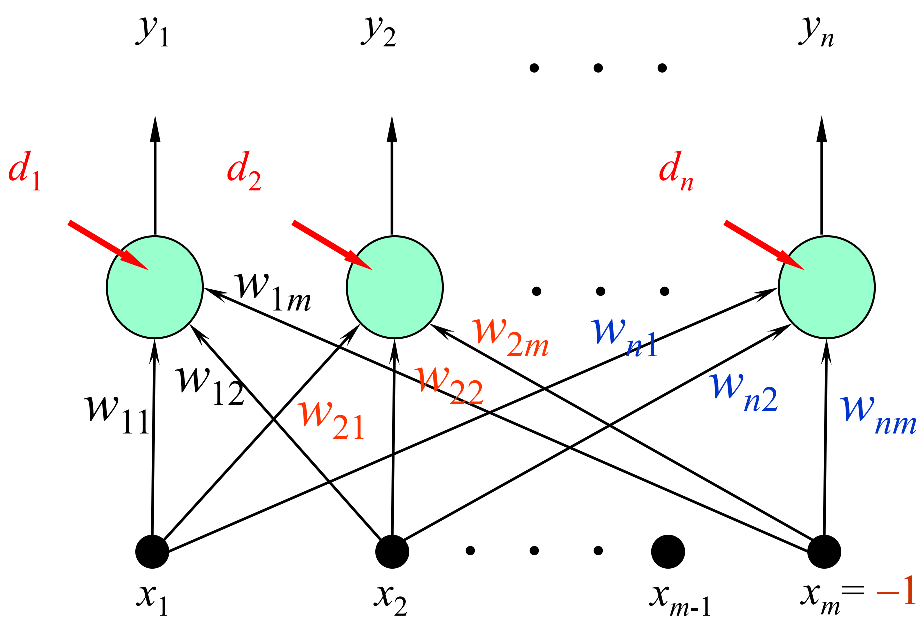
## Perceptron

# The Single-Layered Perceptron



# Training Perceptron

Training Set $\mathbf{T} = \left\{ (\mathbf{x}^{(1)}, \mathbf{d}^{(1)}), (\mathbf{x}^{(2)}, \mathbf{d}^{(2)}), \ldots, (\mathbf{x}^{(p)}, \mathbf{d}^{(p)}) \right\}$




Goal:

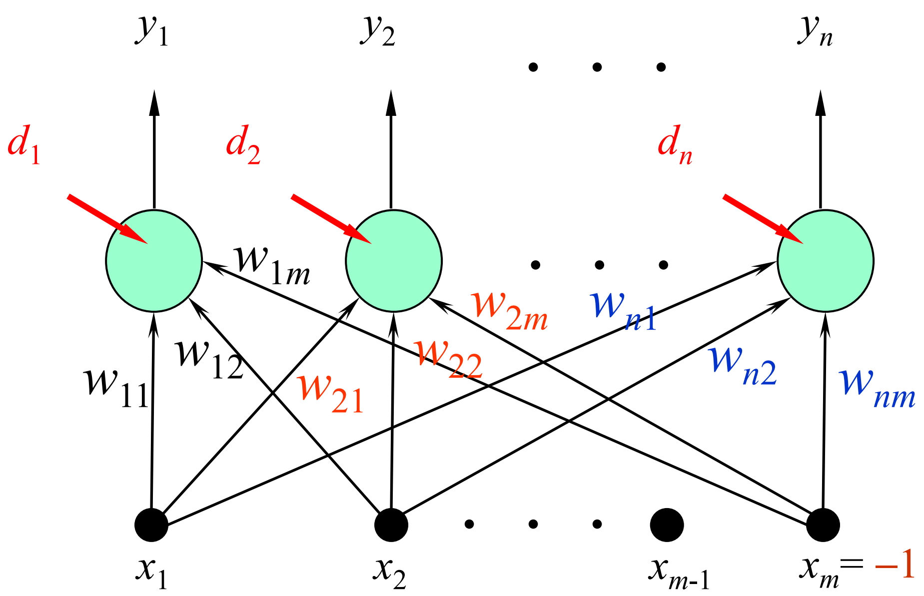
$$y_i^{(k)} = a(\mathbf{w}_i^T \mathbf{x}^{(k)})$$

$$= a\left( \sum_{l=1}^{m} w_{il} x_l^{(k)} \right) = d_i^{(k)}$$

$$\forall \begin{array}{l} i = 1, 2, \ldots, n \\ k = 1, 2, \ldots, p \end{array}$$

# Learning Rules

- Linear Threshold Units (LTUs) : Perceptron Learning Rule
- Linearly Graded Units (LGUs) : Widrow-Hoff learning Rule

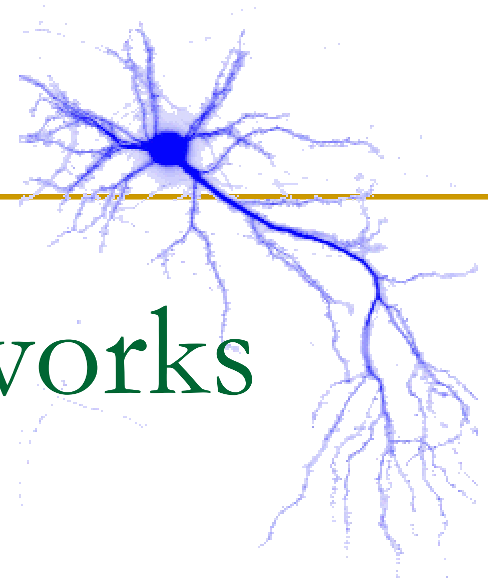



Goal:

$$y_i^{(k)} = a(\mathbf{w}_i^T \mathbf{x}^{(k)}) = a\left(\sum_{l=1}^{m} w_{il} x_l^{(k)}\right) = d_i^{(k)}$$

$$\forall \begin{array}{l} i = 1, 2, \ldots, n \\ k = 1, 2, \ldots, p \end{array}$$

# Feed-Forward Neural Networks

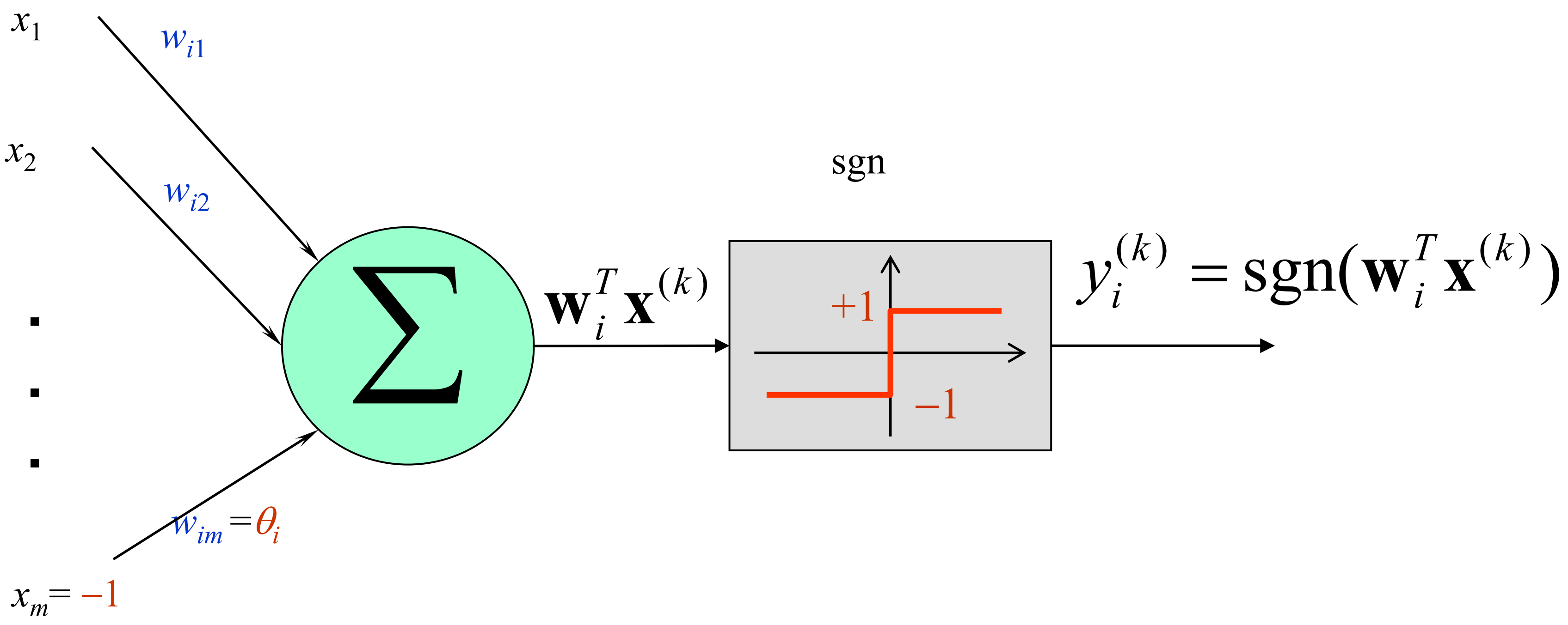


## Learning Rules for Perceptron

- Adaline Leaning Rule
- $\delta$-Learning Rule

# Perceptron

## Linear Threshold Unit

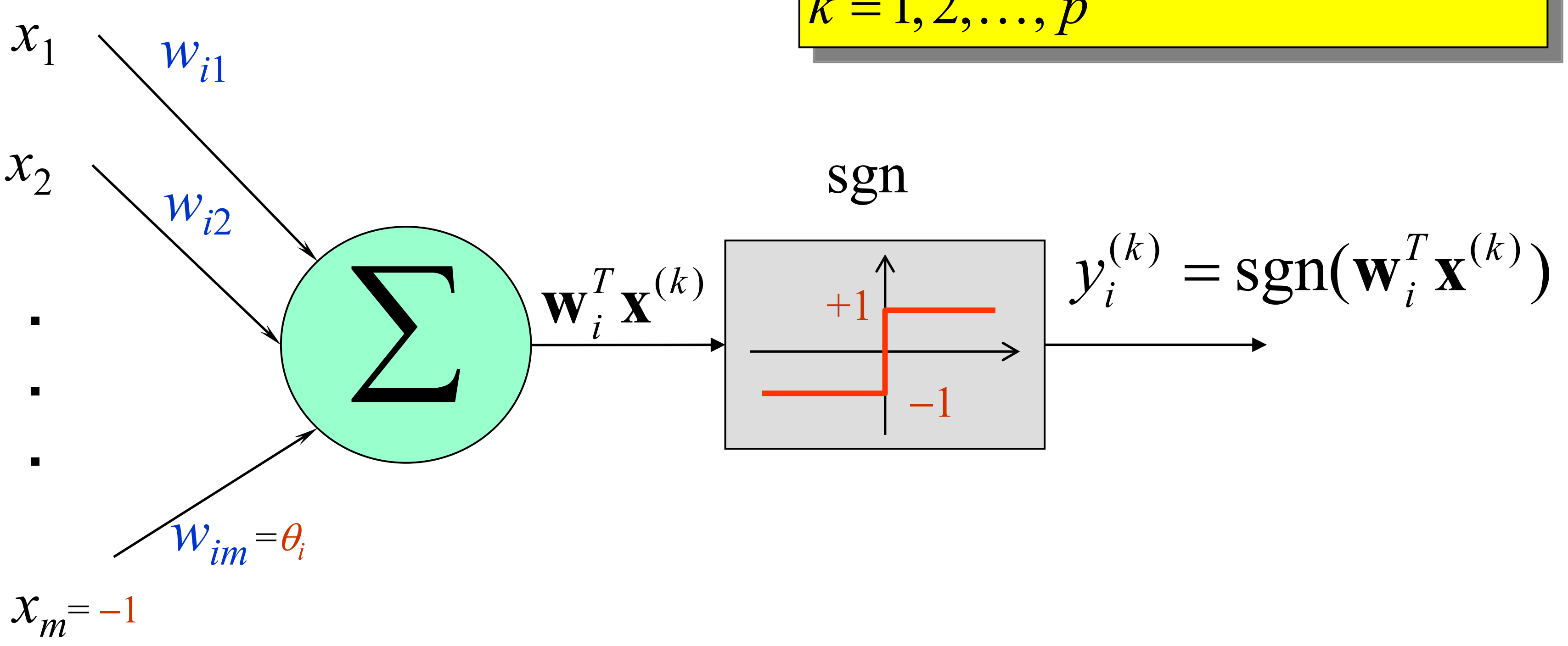


# Perceptron

Goal:

$$y_i^{(k)} = \mathrm{sgn}(\mathbf{w}_i^T \mathbf{x}^{(k)}) = d_i^{(k)} \in \{1, -1\}$$
$$i = 1, 2, \ldots, n$$
$$k = 1, 2, \ldots, p$$

Linear Threshold Unit



# Example

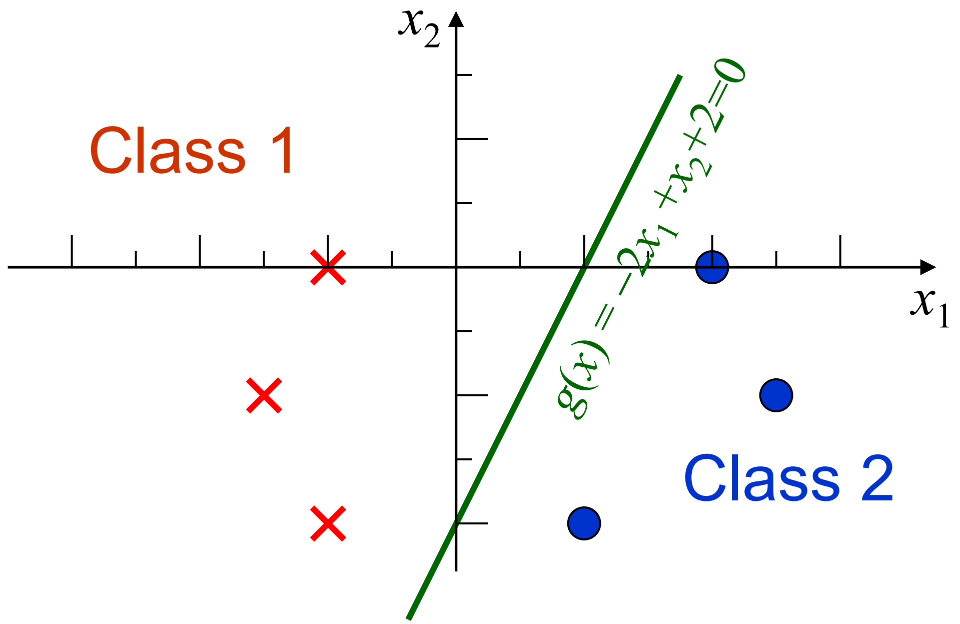
Goal: $y_i^{(k)} = \operatorname{sgn}(\mathbf{w}_i^T \mathbf{x}^{(k)}) = d_i^{(k)} \in \{1, -1\}$

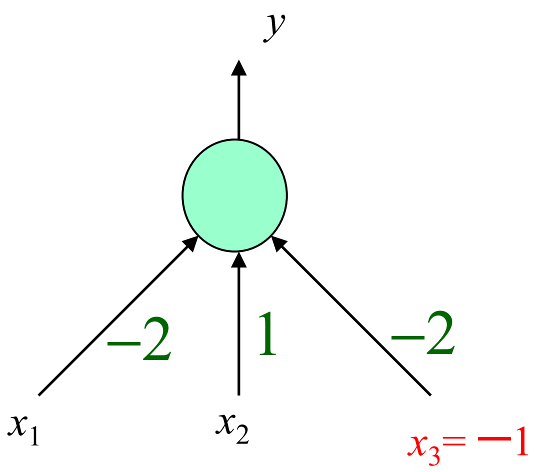
$i = 1, 2, \ldots, n$

$k = 1, 2, \ldots, p$

Class 1 (+1) — $\{[-1,0]^T, [-1.5,-1]^T, [-1,-2]^T\}$

Class 2 (–1) — $\{[2,0]^T, [2.5,-1]^T, [1,-2]^T\}$



# Augmented input vector

Goal: $y^{(k)} = \mathrm{sgn}(\mathbf{w}^T \mathbf{x}^{(k)}) = d^{(k)}$

$\mathbf{w} = (w_1, w_2, w_3)^T$

Class 1 (+1) — $\left\{[-1,0]^T, [-1.5,-1]^T, [-1,-2]^T\right\}$

Class 2 (–1) — $\left\{[2,0]^T, [2.5,-1]^T, [1,-2]^T\right\}$

Class 1 (+1)

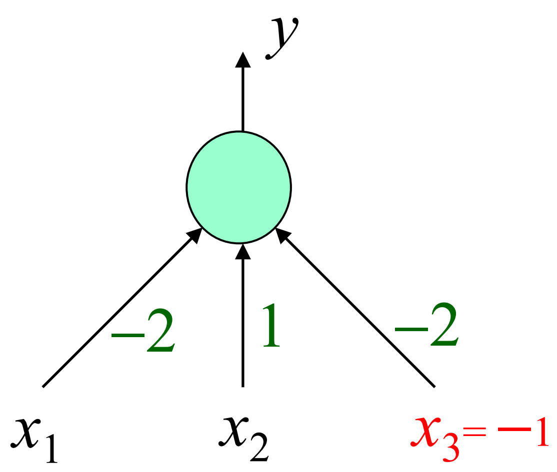
$$x^{(1)} = \begin{bmatrix} -1 \\ 0 \\ -1 \end{bmatrix},\quad x^{(2)} = \begin{bmatrix} -1.5 \\ -1 \\ -1 \end{bmatrix},\quad x^{(3)} = \begin{bmatrix} -1 \\ -2 \\ -1 \end{bmatrix}$$

$$d^{(1)} = 1,\quad d^{(2)} = 1,\quad d^{(3)} = 1$$

Class 2 (–1)

$$x^{(4)} = \begin{bmatrix} 2 \\ 0 \\ -1 \end{bmatrix},\quad x^{(5)} = \begin{bmatrix} 2.5 \\ -1 \\ -1 \end{bmatrix},\quad x^{(6)} = \begin{bmatrix} 1 \\ -2 \\ -1 \end{bmatrix}$$

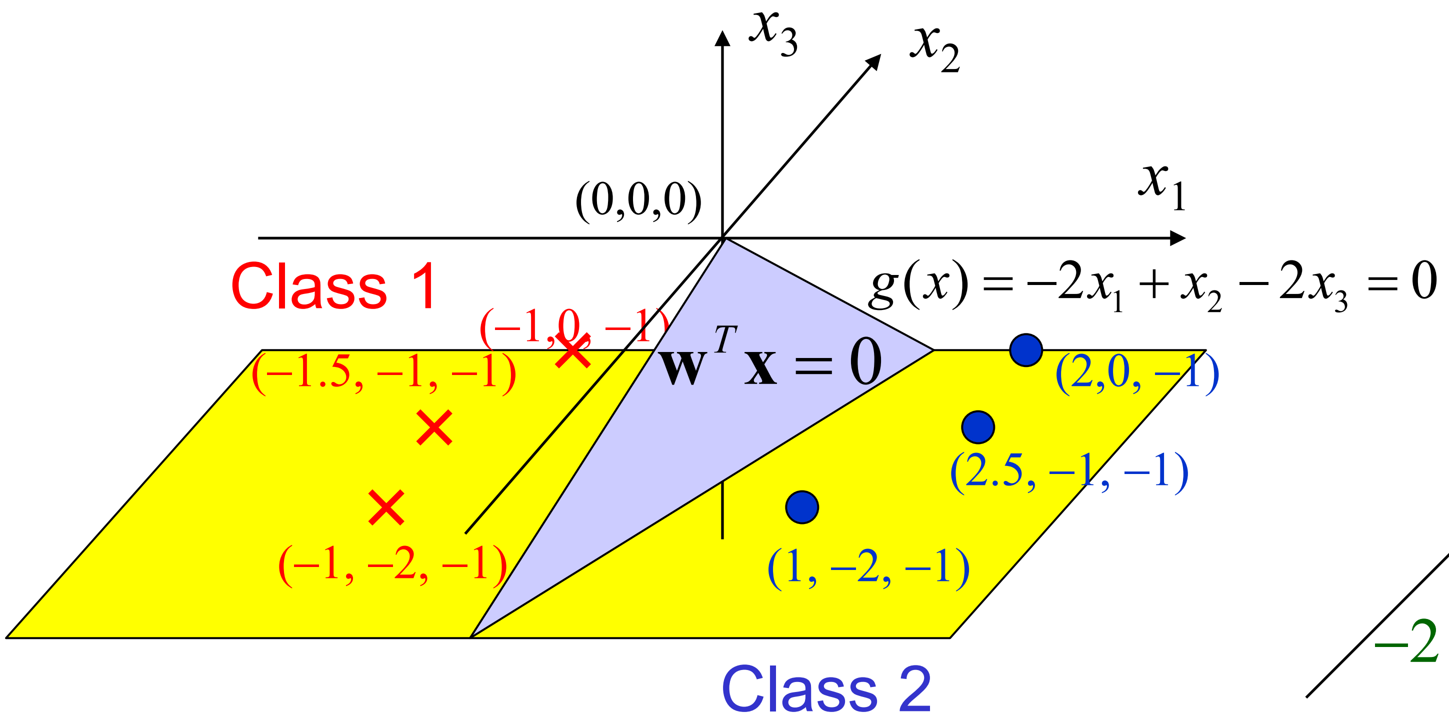
$$d^{(4)} = -1,\quad d^{(5)} = -1,\quad d^{(6)} = -1$$

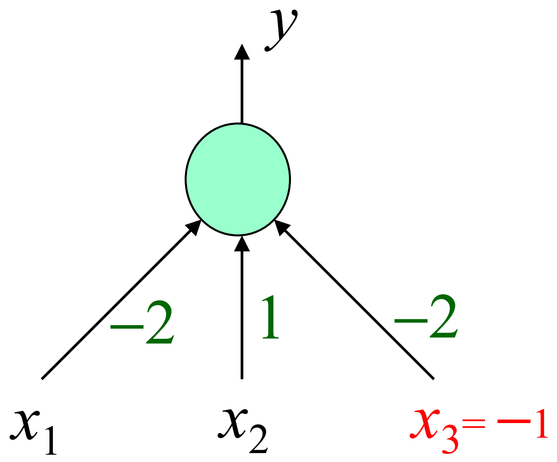


# Augmented input vector

Goal: $y^{(k)} = \mathrm{sgn}(\mathbf{w}^T \mathbf{x}^{(k)}) = d^{(k)}$

$\mathbf{w} = (w_1, w_2, w_3)^T$

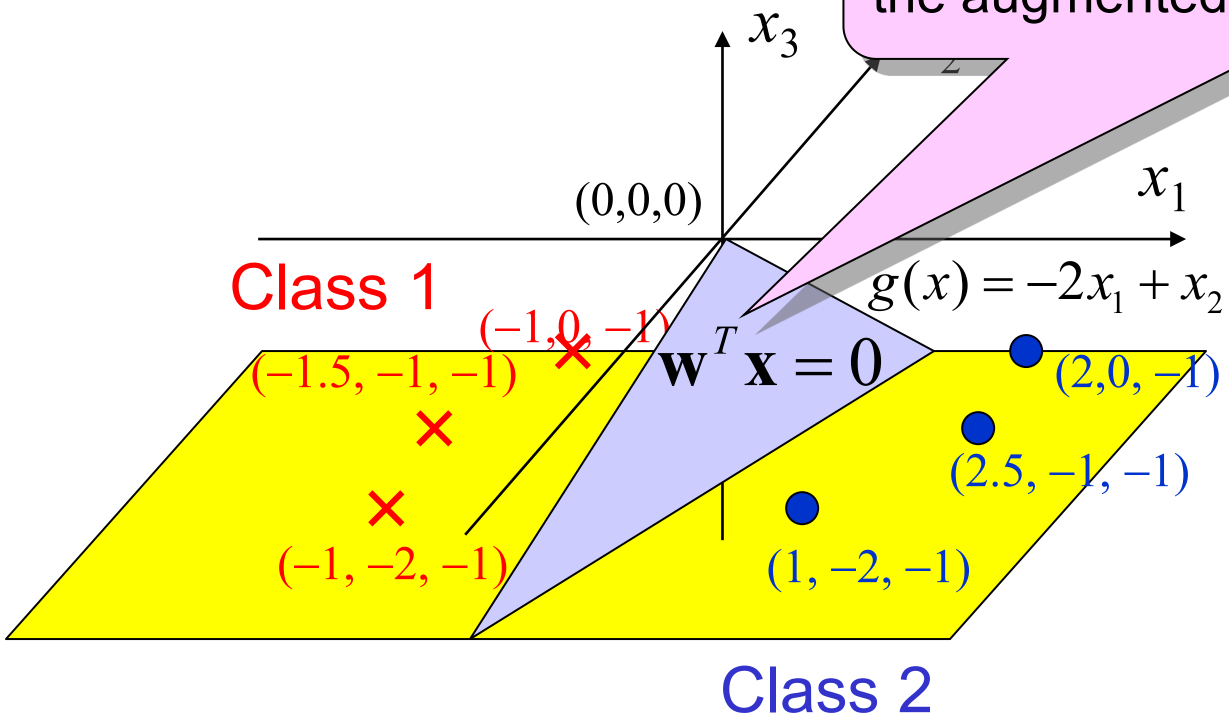


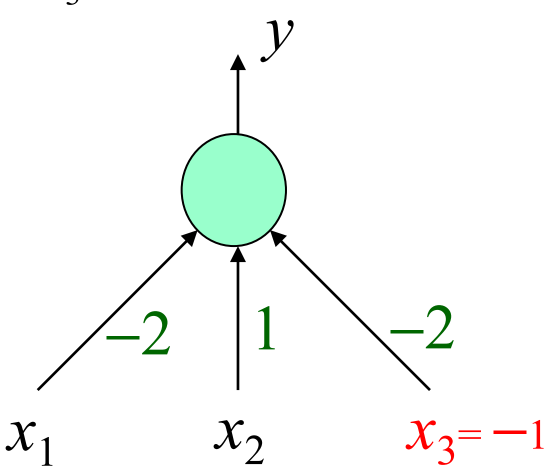



# Augmented input vector

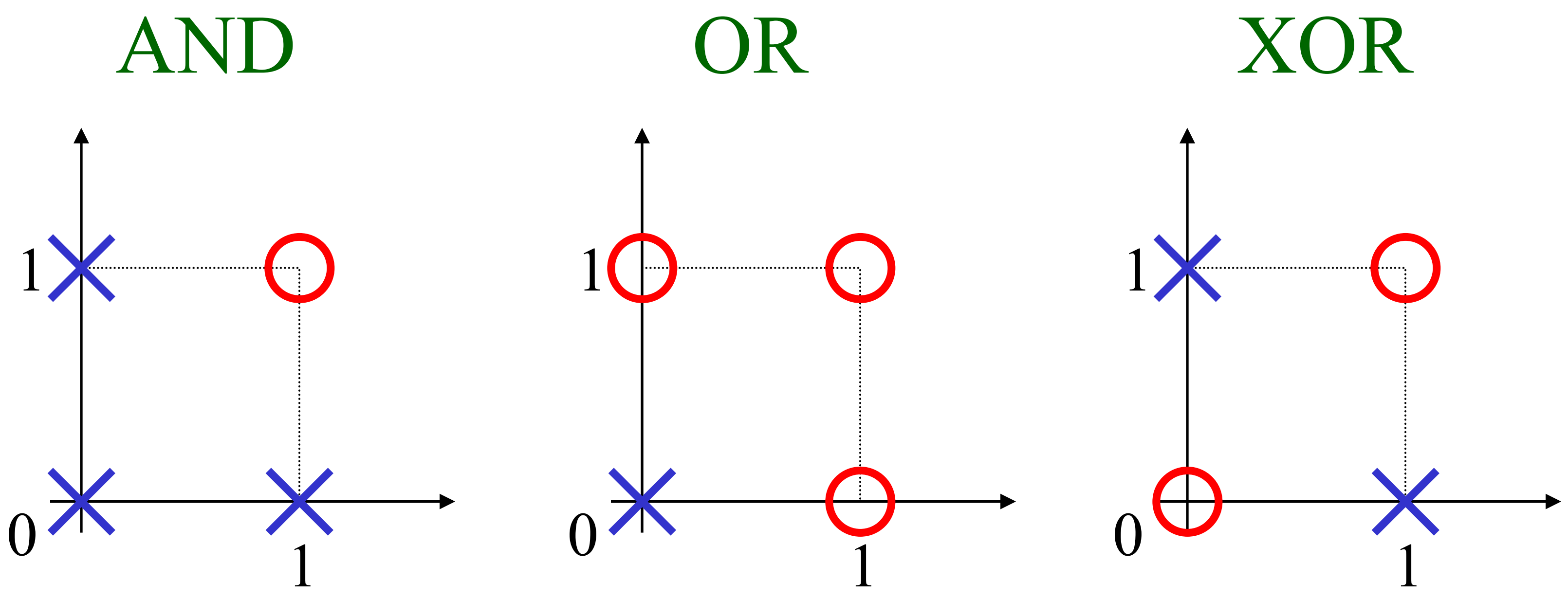
Goal: $y^{(k)} = \mathrm{sgn}(\mathbf{w}^T\mathbf{x}^{(k)}) = d^{(k)}$

A plane passes through the origin in the augmented input space.



# Linearly Separable vs. Linearly Non-Separable




Linearly Separable　　Linearly Separable　　Linearly Non-Separable

# Goal

- Given training sets $T_1 \in C_1$ and $T_2 \in C_2$ with elements in form of $\mathbf{x}=(x_1, x_2, \ldots, x_{m-1}, x_m)^T$, where $x_1, x_2, \ldots, x_{m-1} \in R$ and $x_m = -1$.
- Assume $T_1$ and $T_2$ are linearly separable.
- Find $\mathbf{w}=(w_1, w_2, \ldots, w_m)^T$ such that

$$\boxed{\operatorname{sgn}(\mathbf{w}^T\mathbf{x}) = \begin{cases} 1 & \mathbf{x} \in T_1 \\ -1 & \mathbf{x} \in T_2 \end{cases}}$$

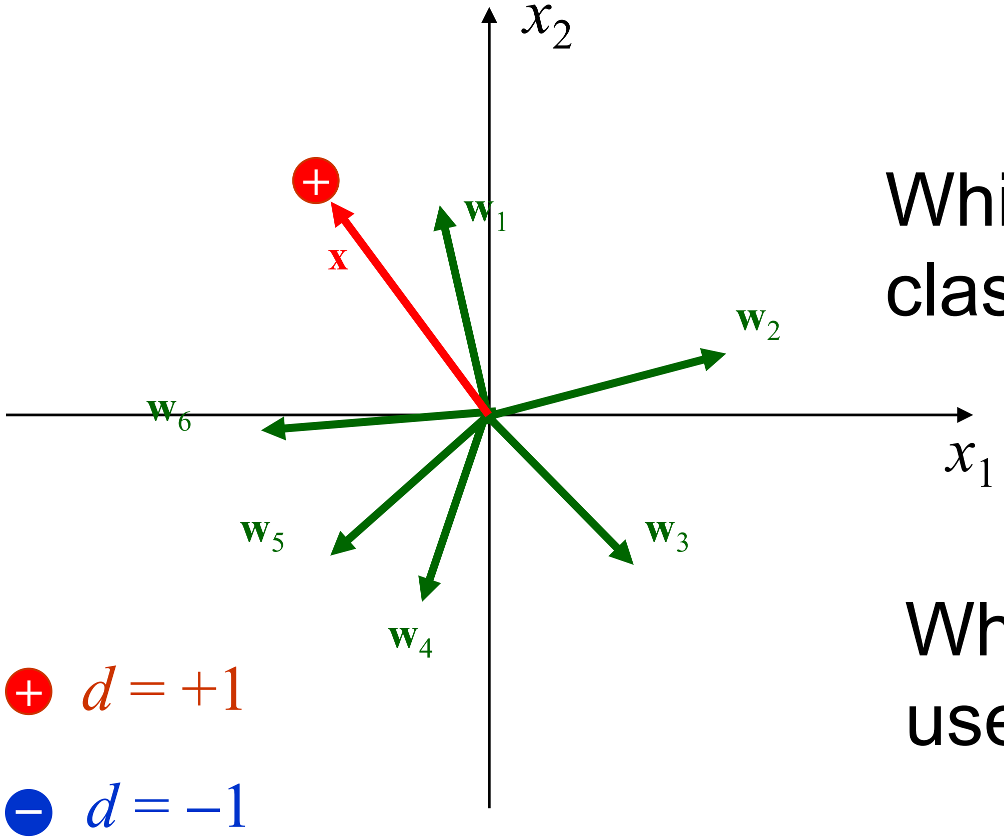
# Goal

$\mathbf{w}^T\mathbf{x} = 0$ is a hyperplane passes through the origin of *augmented input space*.

- Given training sets $T_1 \in C_1$ and $T_2 \in C_2$ with elements in form of $\mathbf{x}=(x_1, x_2, \ldots, x_{m-1}, x_m)^T$, where $x_1, x_2, \ldots, x_{m-1} \in R$ and $x_m = -1$.
- Assume $T_1$ and $T_2$ are linearly separable.
- Find $\mathbf{w}=(w_1, w_2, \ldots, w_m)^T$ such that

$$\operatorname{sgn}(\mathbf{w}^T\mathbf{x}) = \begin{cases} 1 & \mathbf{x} \in T_1 \\ -1 & \mathbf{x} \in T_2 \end{cases}$$

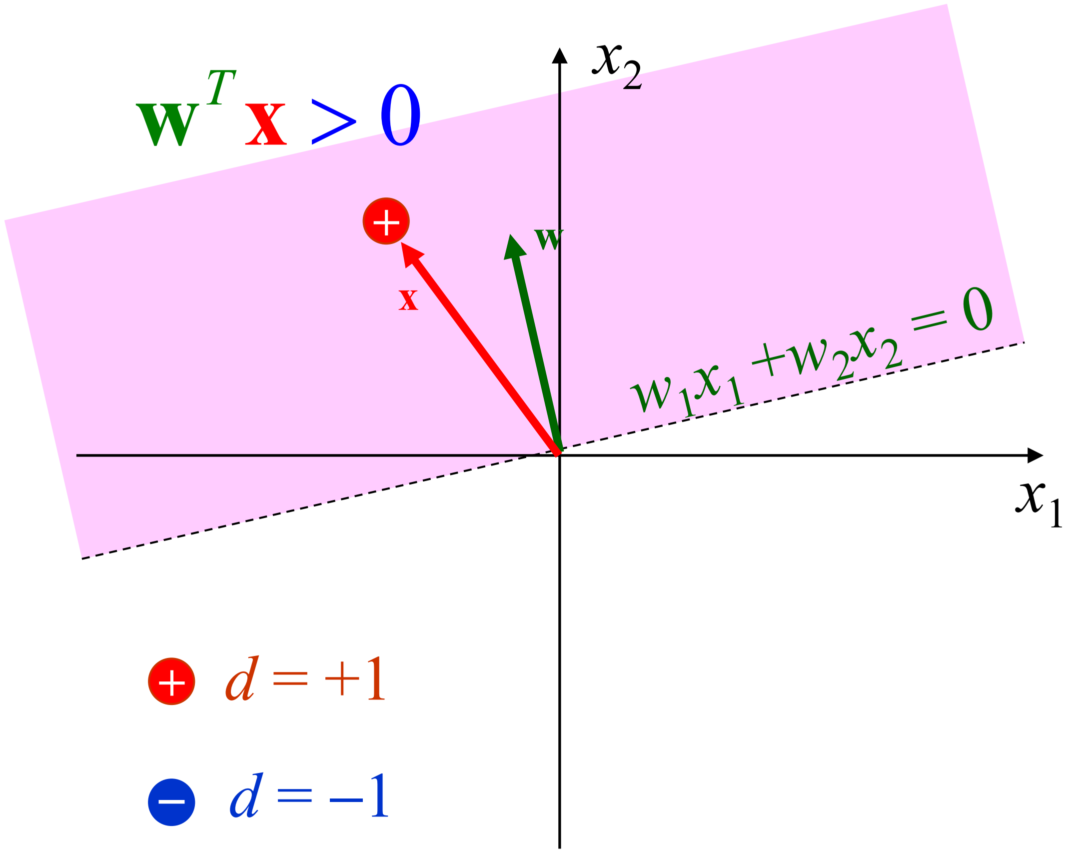
# Observation




Which **w**'s correctly classify **x**?

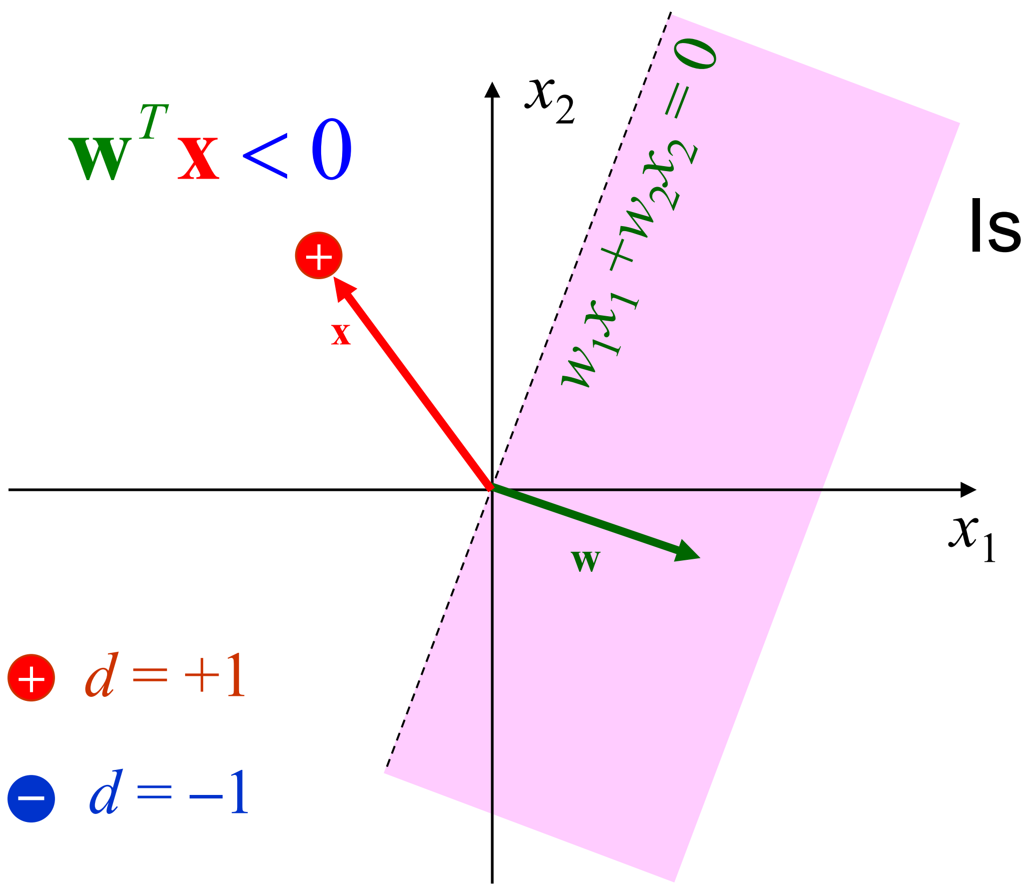
What trick can be used?

# Observation



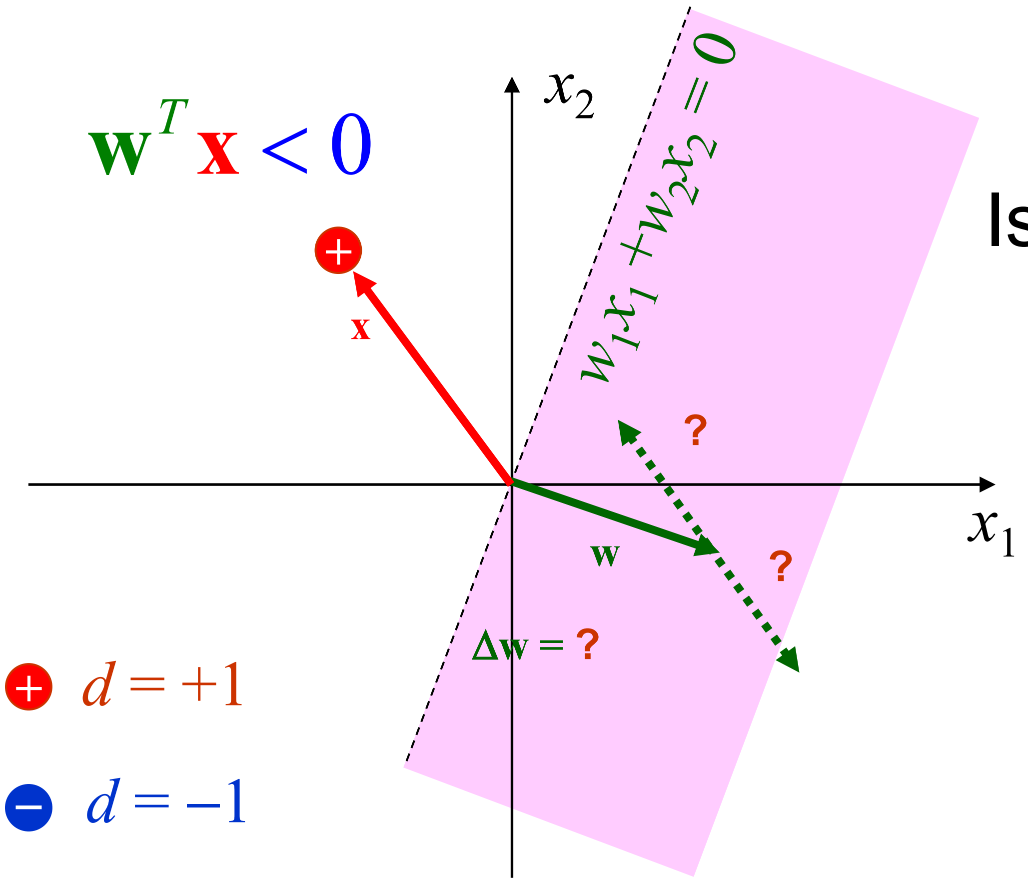

Is this $\mathbf{w}$ ok?

# Observation




Is this w ok?

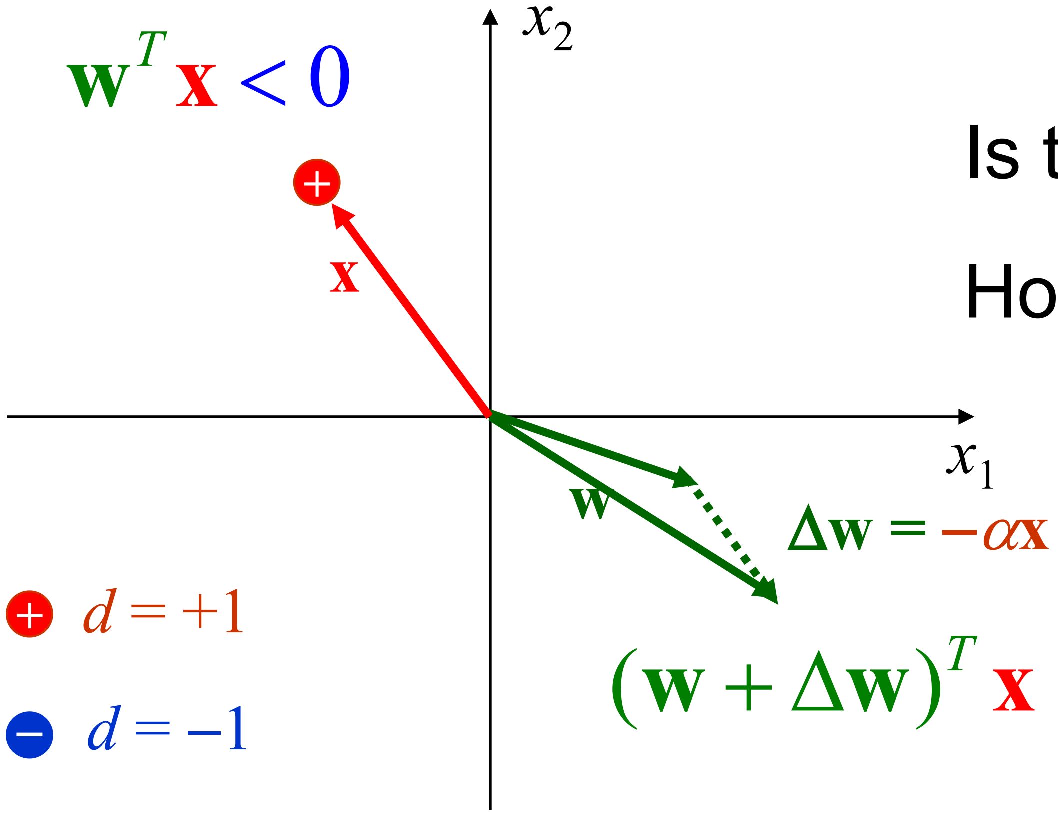
# Observation




Is this w ok?

How to adjust w?

# Observation




Is this w ok?

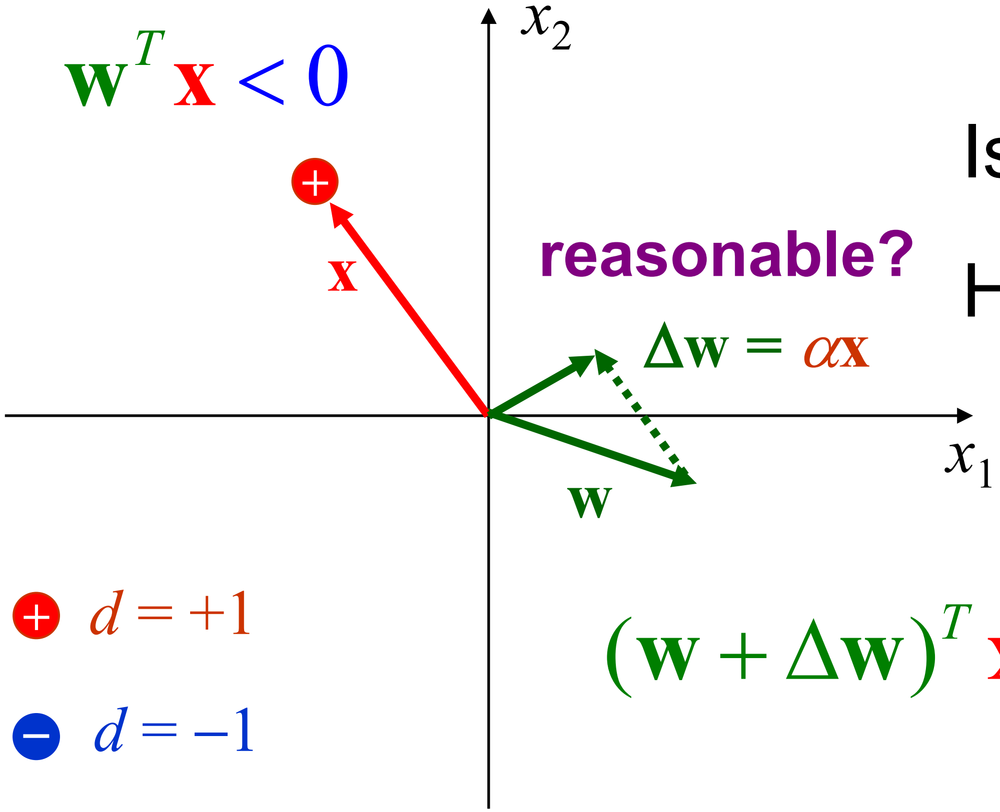
How to adjust w?

**reasonable?**

$$(\mathbf{w}+\Delta\mathbf{w})^T\mathbf{x} = \underbrace{\mathbf{w}^T\mathbf{x}}_{<0} - \underbrace{\alpha\mathbf{x}^T\mathbf{x}}_{>0}$$

# Observation

$\mathbf{w}^T\mathbf{x} < 0$




$\oplus$ $d = +1$

$\ominus$ $d = -1$

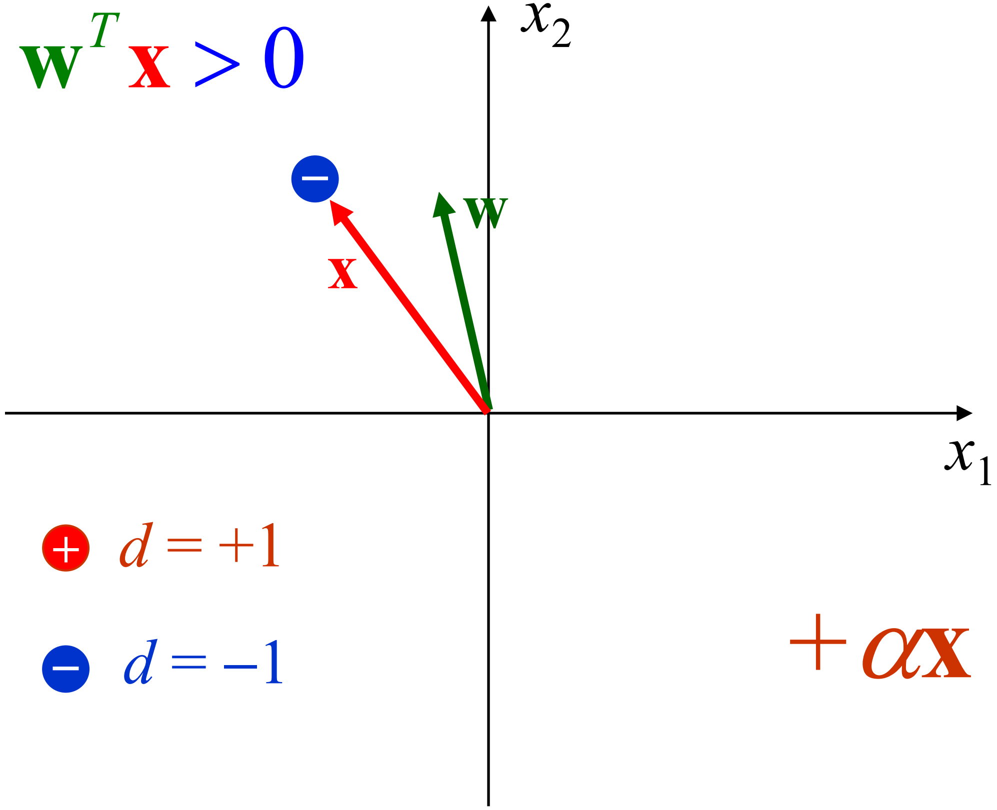
Is this w ok?

How to adjust w?

$$(\mathbf{w} + \Delta\mathbf{w})^T\mathbf{x} = \underbrace{\mathbf{w}^T\mathbf{x}}_{<0} + \underbrace{\alpha\mathbf{x}^T\mathbf{x}}_{>0}$$

# Observation

$$\mathbf{w}^T\mathbf{x} > 0$$




Is this $\mathbf{w}$ ok?

$$\Delta\mathbf{w} = ?$$

$d = +1$

$d = -1$

$+\alpha\mathbf{x}$ or $-\alpha\mathbf{x}$

# Perceptron Learning Rule

Upon misclassification on

$\oplus \quad d = +1 \qquad \Delta\mathbf{w} = \alpha\mathbf{x}$

$\ominus \quad d = -1 \qquad \Delta\mathbf{w} = -\alpha\mathbf{x}$

$$\alpha > 0$$

**Define error**

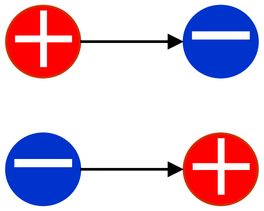
$$r = d - y = \begin{cases} +2 & \oplus \to \ominus \\ -2 & \ominus \to \oplus \\ 0 & \textbf{No error} \end{cases}$$

# Perceptron Learning Rule

$$\Delta \mathbf{w} = \eta r \mathbf{x}$$

**Define error**

$$r = d - y = \begin{cases} +2 \\ -2 \\ 0 \end{cases}$$



**No error**

# Perceptron Learning Rule

$$\Delta \mathbf{w} = \eta r \mathbf{x}$$

Learning Rate

Error $(d - y)$

Input

# Summary – Perceptron Learning Rule

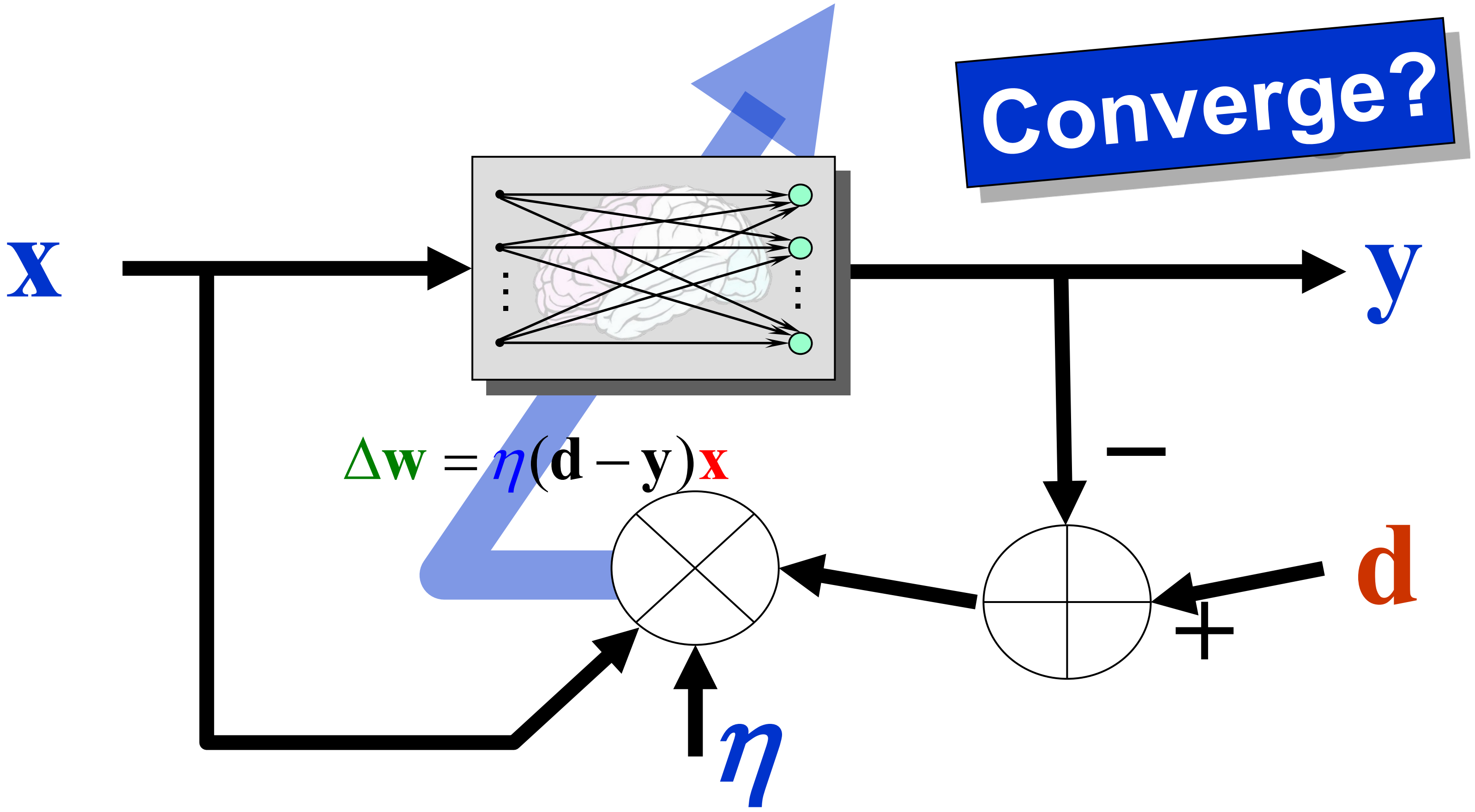
Based on the general weight learning rule.

$$\Delta w_i(t) = \eta r_i x_i(t)$$

$$r_i = d_i - y_i \quad = \begin{cases} 0 & d_i = y_i \quad \text{correct} \\ +2 & d_i = 1, y_i = -1 \\ -2 & d_i = -1, y_i = 1 \end{cases} \Big\} \text{incorrect}$$

$$\Delta w_i(t) = \eta (d_i - y_i) x_i(t)$$

# Summary – Perceptron Learning Rule



# Perceptron Convergence Theorem

**If the given training set is linearly separable, the learning process will converge in a finite number of steps.**