# CZ4041/CE4041: Machine Learning

## Lesson 7a: Perceptron

Kelly KE

School of Computer Science and Engineering, NTU, Singapore

Acknowledgements: Some figures are adopted from the lecture notes of the book "Introduction to Data Mining" (Chap. 5). Slides are modified from the version prepared by Dr. Sinno Pan.

# Instructor's Information

- **Name:** Ke Yiping, Kelly
- **Email:** ypke@ntu.edu.sg
- **Phone:** 6790-5046
- **Office:** N4-02a-12

# Lecture and Tutorial Information

- Lecture time/venue (2nd half)
  - Weeks 7 -12, Tuesdays 11:30am – 1:30pm
  - Online via MS Teams
    - CZ/CE4041 in NTULearn → Information → Teams link (The SAME link will be reused for all lectures)
- Tutorial time/venue (2nd half)
  - **Weeks 8, 9, 10,12**, Thursdays 1:30 – 2:30pm
  - Online via MS Teams
    - CZ/CE4041 in NTULearn → Information → Teams link (The SAME link will be reused for all tutorials)

# Lecture and Tutorial Information

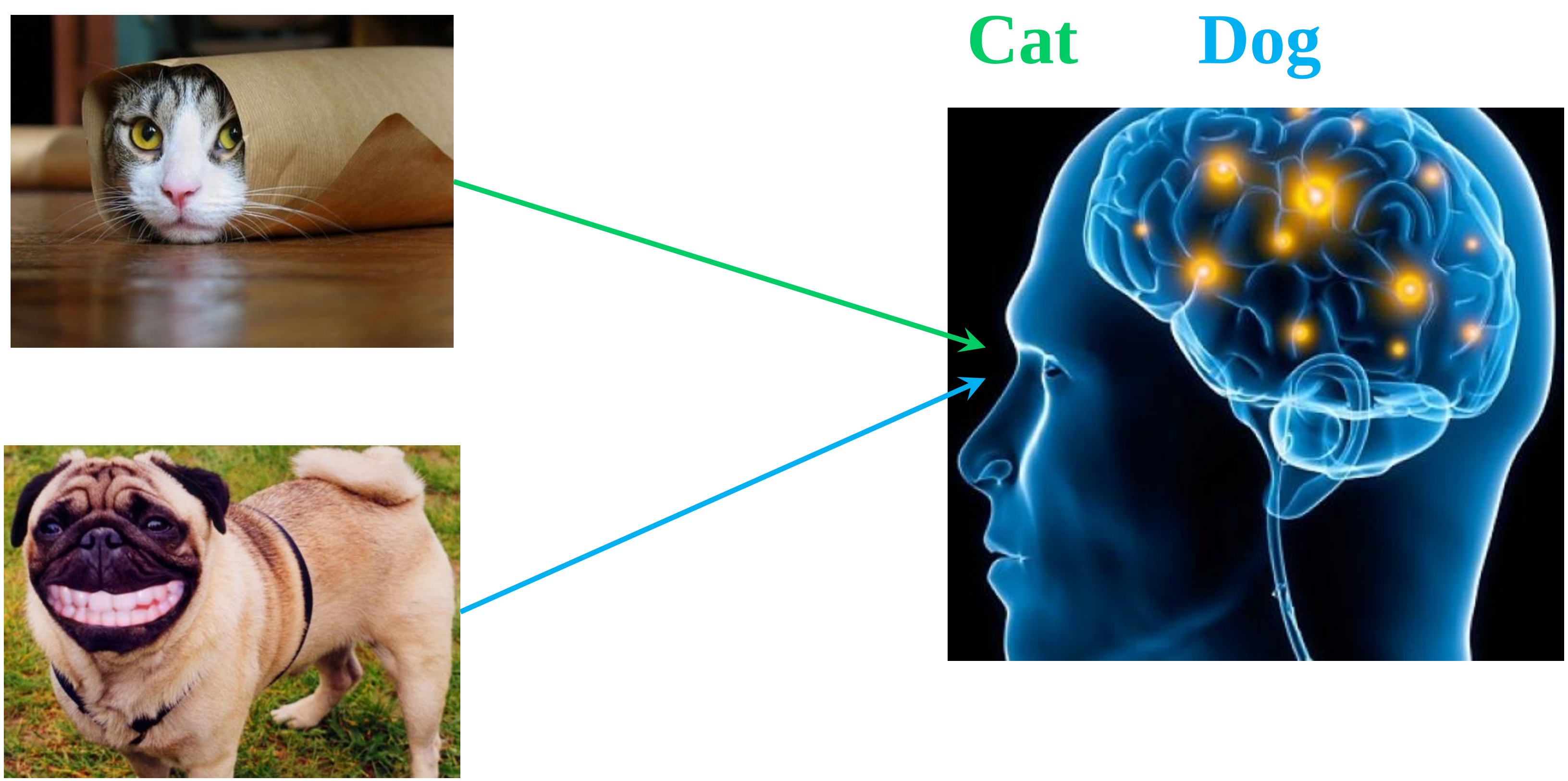
- Q&A (2nd half)
  - Send questions via email ypke@ntu.edu.sg
  - Send questions via Teams
  - Make an appointment
- My personal site
  - https://keyiping.wixsite.com/index
  - Check information when NTULearn is down

# Outline

- Artificial Neural Networks
  - Perceptrons
  - Multi-layer Neural Networks

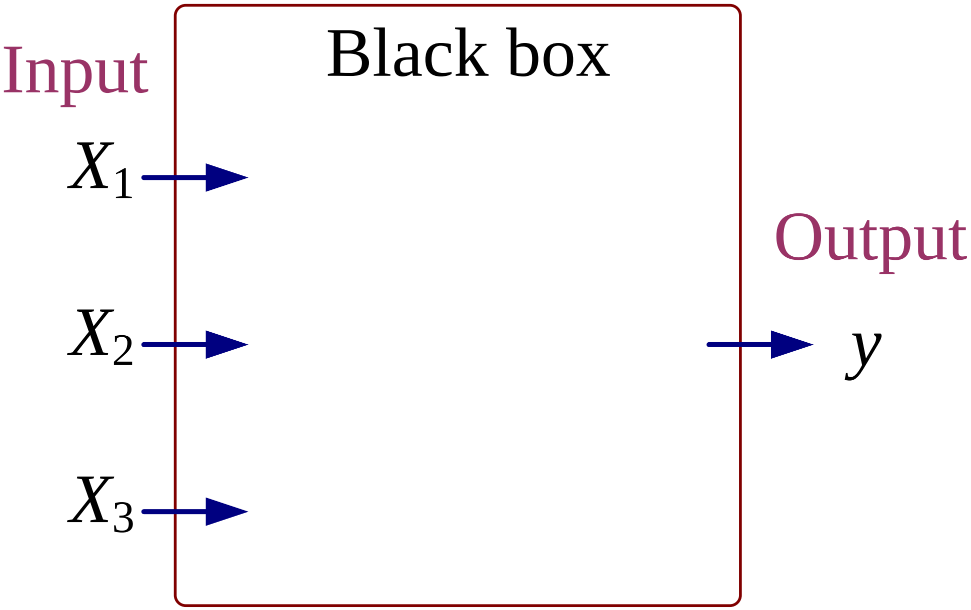
# Artificial Neural Networks (ANN)

- The study of ANN was inspired by attempts to simulate biological neural systems



# Artificial Neural Networks (cont.)

| $X_1$ | $X_2$ | $X_3$ | $y$ |
|---|---|---|---|
| 1 | 0 | 0 | -1 |
| 1 | 0 | 1 | 1 |
| 1 | 1 | 0 | 1 |
| 1 | 1 | 1 | 1 |
| 0 | 0 | 1 | -1 |
| 0 | 1 | 0 | -1 |
| 0 | 1 | 1 | 1 |
| 0 | 0 | 0 | -1 |




Output $y$ is 1 if at least two of the three inputs are equal to 1

# Artificial Neural Networks (cont.)

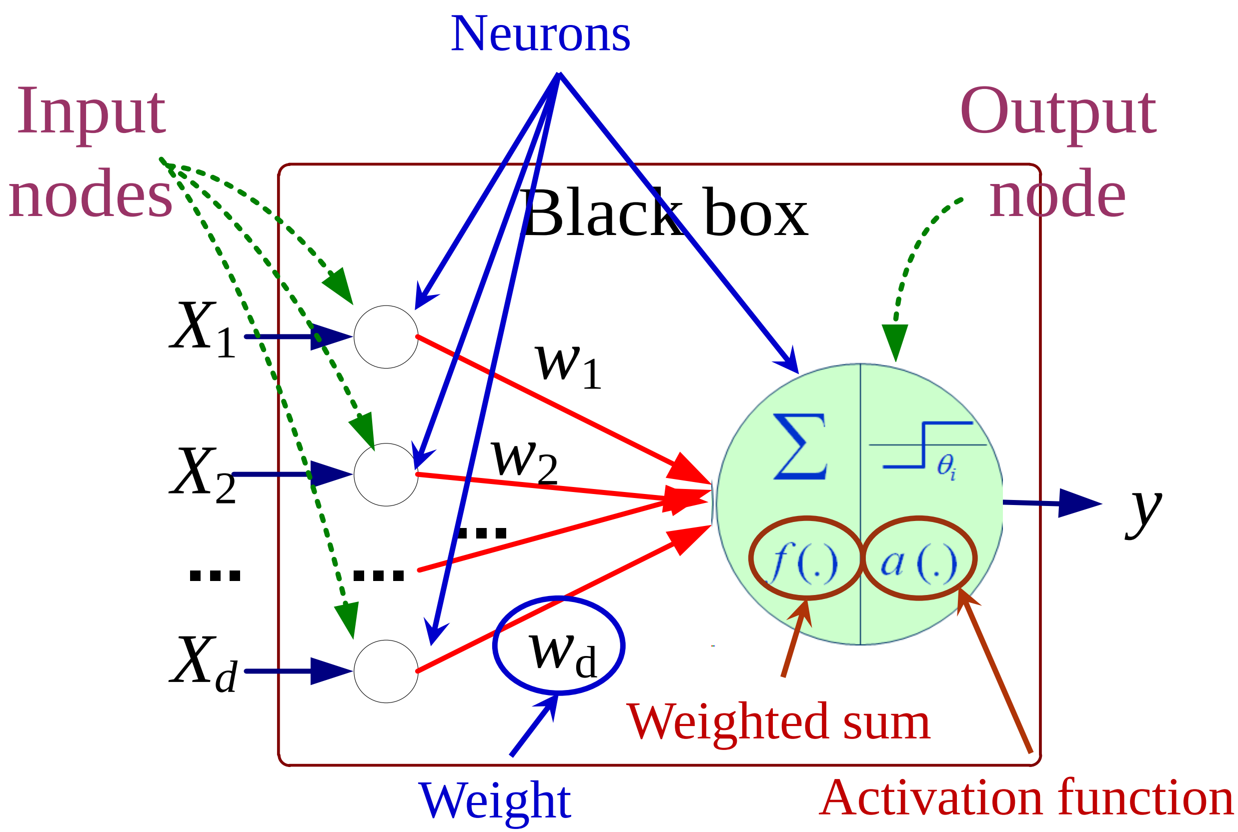
- Human brain is a densely interconnected network of neurons, connected to others via axons.
- Axons are used to transmit nerve impulses from one neuron to another
- The human brain learns by changing the strength of the synaptic connection between neurons
- An ANN is composed of an interconnected assembly of nodes and directed links.

# Outline

- Artificial Neural Networks
  - Perceptrons
  - Multi-layer Neural Networks

# ANN: Perceptron

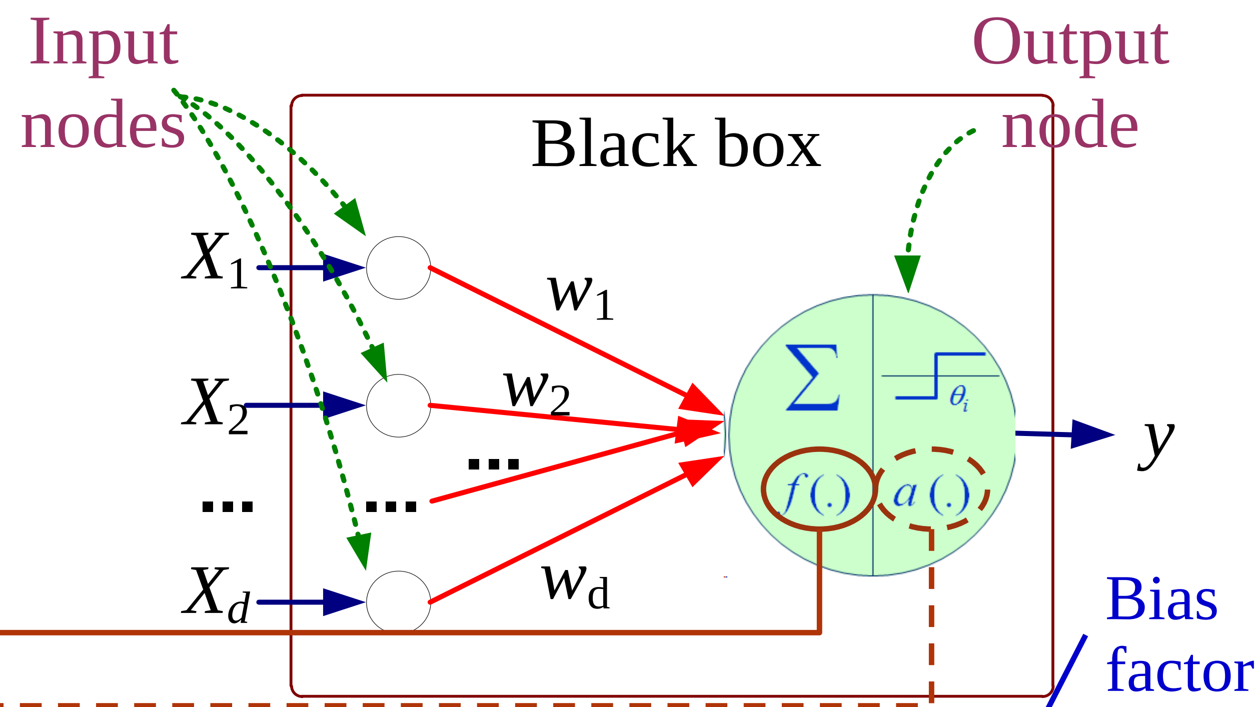
| $X_1$ | $X_2$ | … | $X_d$ |
|---|---|---|---|
| 1 | 0 | … | 0 |
| 1 | 0 | … | 1 |
| 1 | 1 | … | 0 |
| 1 | 1 | … | 1 |
| 0 | 0 | … | 1 |
| 0 | 1 | … | 0 |
| 0 | 1 | … | 1 |
| 0 | 0 | … | 0 |




Each input node is connected via a weighted link to the output node. Weights can be positive, negative or zero (no connection)

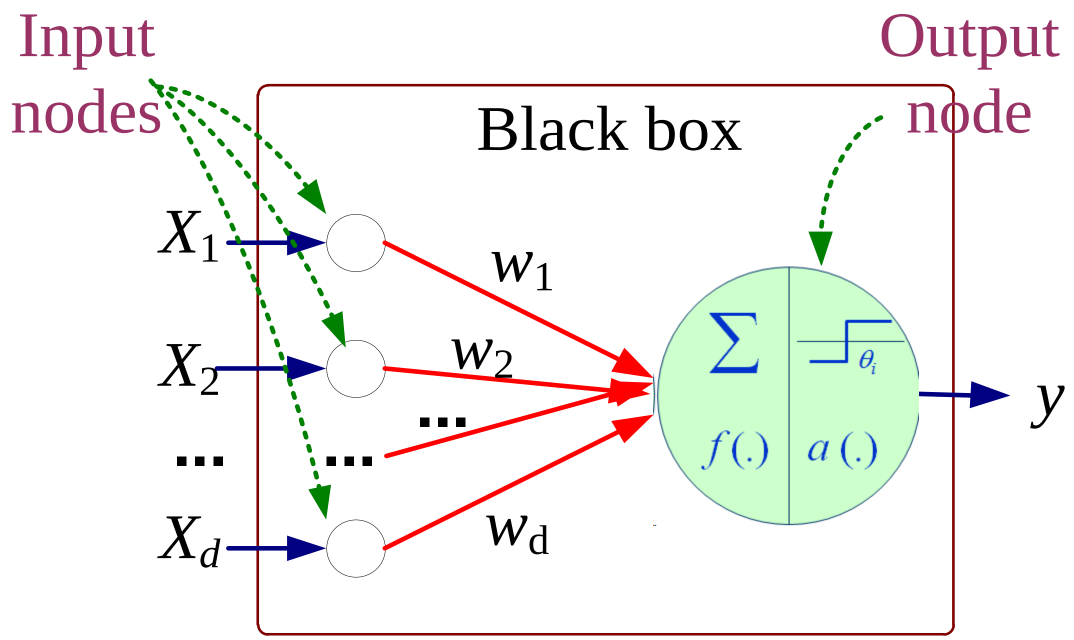
# ANN: Perceptron (cont.)

| $X_1$ | $X_2$ | … | $X_d$ |
|---|---|---|---|
| 1 | 0 | … | 0 |
| 1 | 0 | … | 1 |
| 1 | 1 | … | 0 |
| 1 | 1 | … | 1 |
| 0 | 0 | … | 1 |
| 0 | 1 | … | 0 |
| 0 | 1 | … | 1 |
| 0 | 0 | … | 0 |



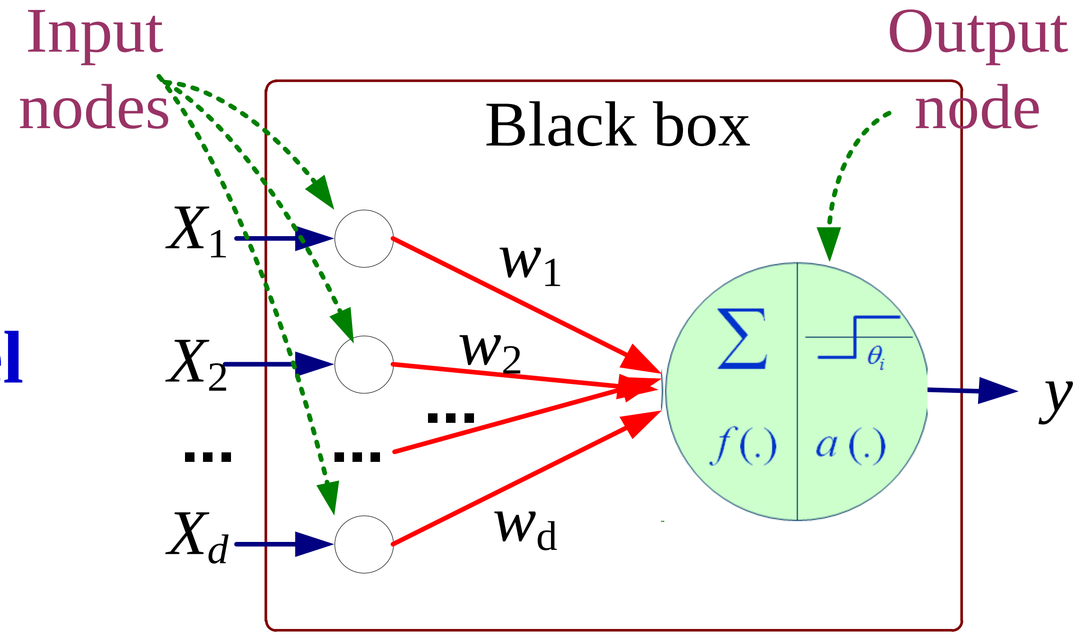

$$z = f(\mathbf{x}, \mathbf{w}) = w_1x_1 + w_2x_2 + \cdots + w_dx_d - \theta$$

$$y = a(z), \text{ where } a(z) = \text{sign}(z) = \begin{cases} 1, & z \geq 0 \\ -1, & \text{otherwise} \end{cases}$$




- Model is an assembly of inter-connected nodes and weighted links
- Output node first sums up each of its input value according to the weights of its links
- Compare the weighted sum against some threshold $\theta$
- Produce an output based on the sign of the result

# Perceptron Model




$$z = \sum_{i=1}^{d} w_i x_i - \theta \longrightarrow y = a(z) = \text{sign}(z)$$

$$y = \text{sign}\left(\sum_{i=1}^{d} w_i x_i - \theta\right)$$

# ANN: Perceptron (cont.)

- Mathematically, the output of a perceptron model can be expressed in a more compact form

$$y = \mathrm{sign}\left(\sum_{i=1}^{d} w_i x_i - \theta\right)$$

$$\Downarrow$$

Inner product

$$y = \mathrm{sign}(\mathbf{w} \cdot \boldsymbol{x})$$

$$\text{where } \boldsymbol{x} = (x_0, x_1, x_2, \dots, x_d)$$

$$\mathbf{w} = (w_0, w_1, w_2, \dots, w_d)$$

$$w_0 = -\theta, \text{and } x_0 = 1$$

# Inner Product: Review

- Given two vectors $\boldsymbol{x}$ and $\boldsymbol{z}$, which are both of *d* dimensions, the inner product between $\boldsymbol{x}$ and $\boldsymbol{z}$ is defined as

$$\boldsymbol{x} \cdot \boldsymbol{z} = \sum_{i=1}^{d} (x_i \times z_i)$$

$$\boldsymbol{x} = (x_1, x_2, \ldots, x_d) \qquad \boldsymbol{z} = (z_1, z_2, \ldots, z_d)$$

# ANN: Perceptron (cont.)

$$y = \text{sign}(\mathbf{w} \cdot \boldsymbol{x})$$

$$\boldsymbol{x} = (x_0, x_1, x_2, \dots, x_d)$$

$$\mathbf{w} = (w_0, w_1, w_2, \dots, w_d)$$

$$\mathbf{w} \cdot \boldsymbol{x} = \sum_{i=0}^{d} (w_i \times x_i) = \sum_{i=1}^{d} (w_i \times x_i) + w_0 \times x_0$$

$$w_0 = -\theta, \text{and } x_0 = 1$$

$$-\theta$$

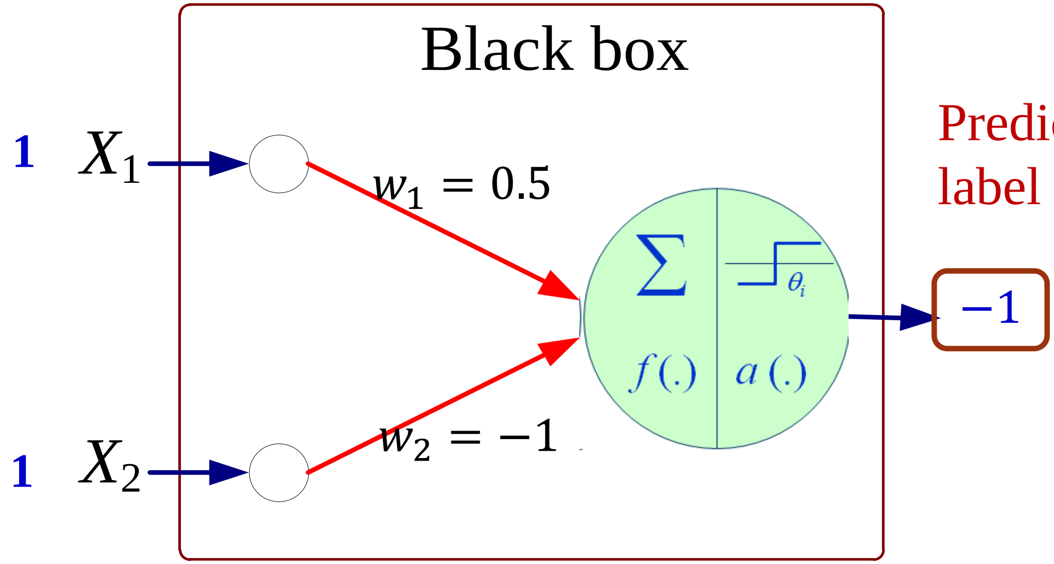
$$y = \text{sign}\left(\sum_{i=1}^{d} w_i x_i - \theta\right) \Longleftrightarrow y = \text{sign}(\mathbf{w} \cdot \boldsymbol{x})$$

# Perceptron: Making Prediction

- Given a learned perceptron with $w_1 = 0.5$, $w_2 = -1$, and $\theta = 0$

Test data:

| | $X_1$ | $X_2$ |
|---|---|---|
| $\boldsymbol{x}$ | 1 | 1 |

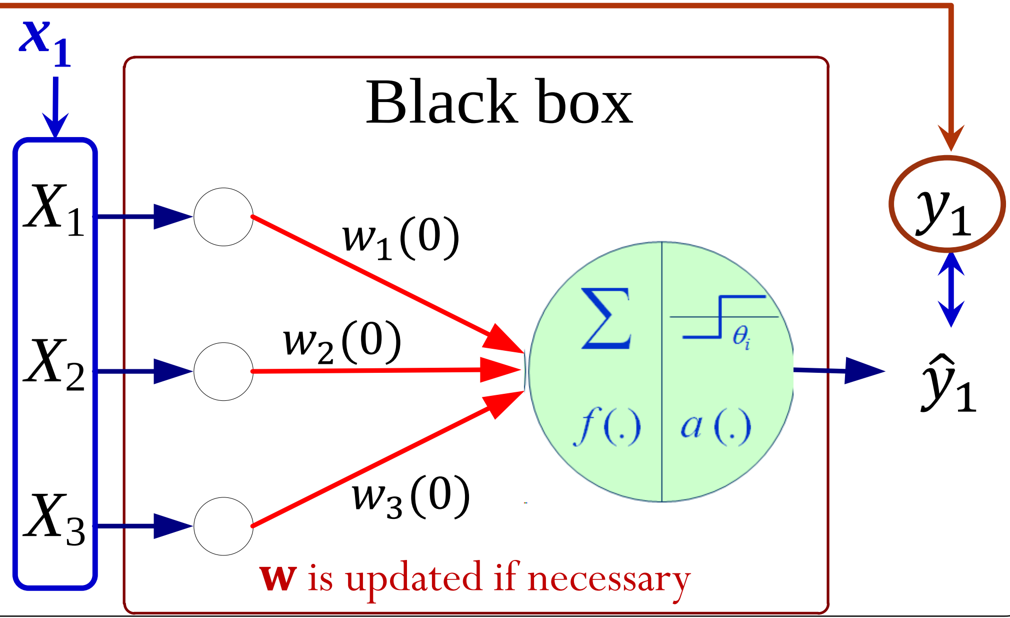



$$y = \text{sign}\big(1 \times 0.5 + 1 \times (-1)\big)$$
$$= \text{sign}(-0.5) \ = -1$$

# Perceptron: Learning

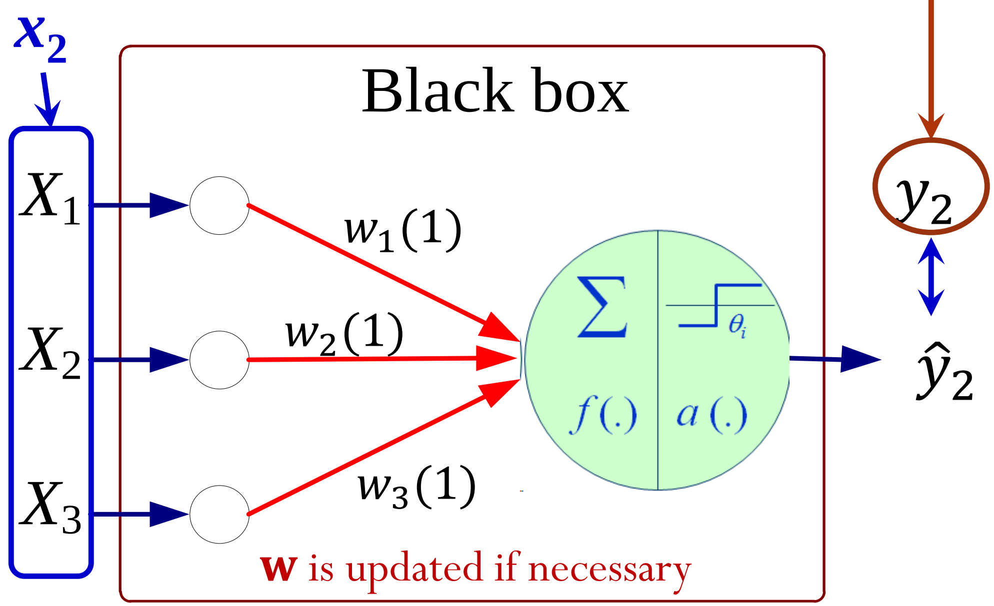
- During training, the weight parameters **w** are adjusted until the outputs of the perceptron become consistent with the true outputs of training data
- The weight parameters **w** are updated iteratively or in an online learning manner

| | $X_1$ | $X_2$ | $X_3$ | $y$ |
|---|---|---|---|---|
| $\mathbf{x}_1$ | 1 | 0 | 0 | -1 |
| $\mathbf{x}_2$ | 1 | 0 | 1 | 1 |
| | 1 | 1 | 0 | 1 |
| $\mathbf{x}_i$ | 1 | 1 | 1 | 1 |
| | 0 | 0 | 1 | -1 |
| | 0 | 1 | 0 | -1 |
| | 0 | 1 | 1 | 1 |
| $\mathbf{x}_8$ | 0 | 0 | 0 | -1 |



# Perceptron: Learning (cont.)

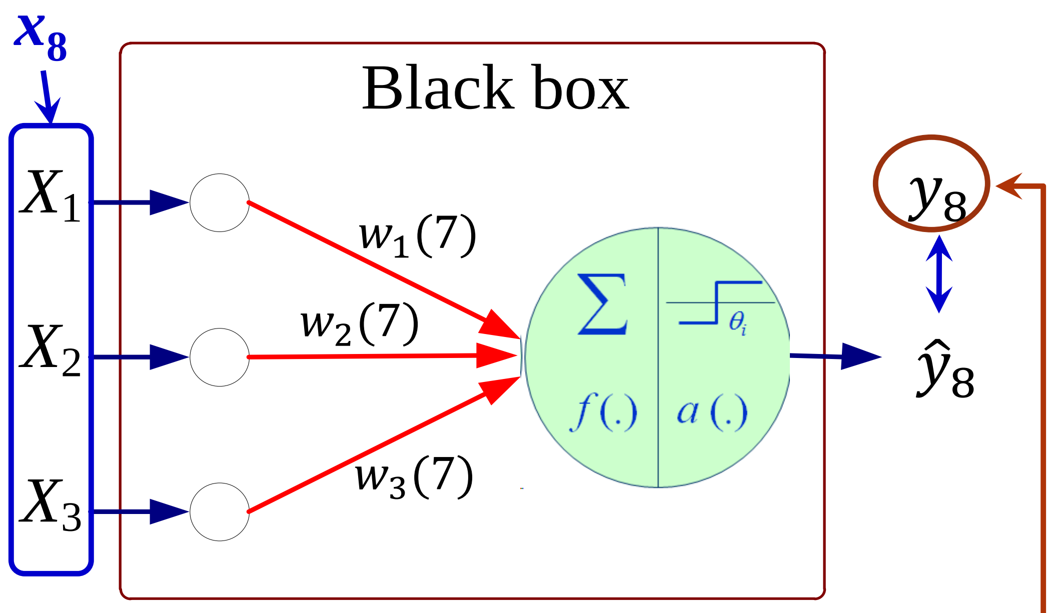
| | $X_1$ | $X_2$ | $X_3$ | $Y$ |
|---|---|---|---|---|
| $\mathbf{x}_1$ | 1 | 0 | 0 | -1 |
| $\mathbf{x}_2$ | 1 | 0 | 1 | 1 |
| | 1 | 1 | 0 | 1 |
| $\mathbf{x}_i$ | 1 | 1 | 1 | 1 |
| | 0 | 0 | 1 | -1 |
| | 0 | 1 | 0 | -1 |
| | 0 | 1 | 1 | 1 |
| $\mathbf{x}_8$ | 0 | 0 | 0 | -1 |




• • •

# Perceptron: Learning (cont.)

| | $X_1$ | $X_2$ | $X_3$ | $y$ |
|---|---|---|---|---|
| $\boldsymbol{x}_1$ | 1 | 0 | 0 | -1 |
| $\boldsymbol{x}_2$ | 1 | 0 | 1 | 1 |
| | 1 | 1 | 0 | 1 |
| $\boldsymbol{x}_i$ | 1 | 1 | 1 | 1 |
| | 0 | 0 | 1 | -1 |
| | 0 | 1 | 0 | -1 |
| | 0 | 1 | 1 | 1 |
| $\boldsymbol{x}_8$ | 0 | 0 | 0 | -1 |

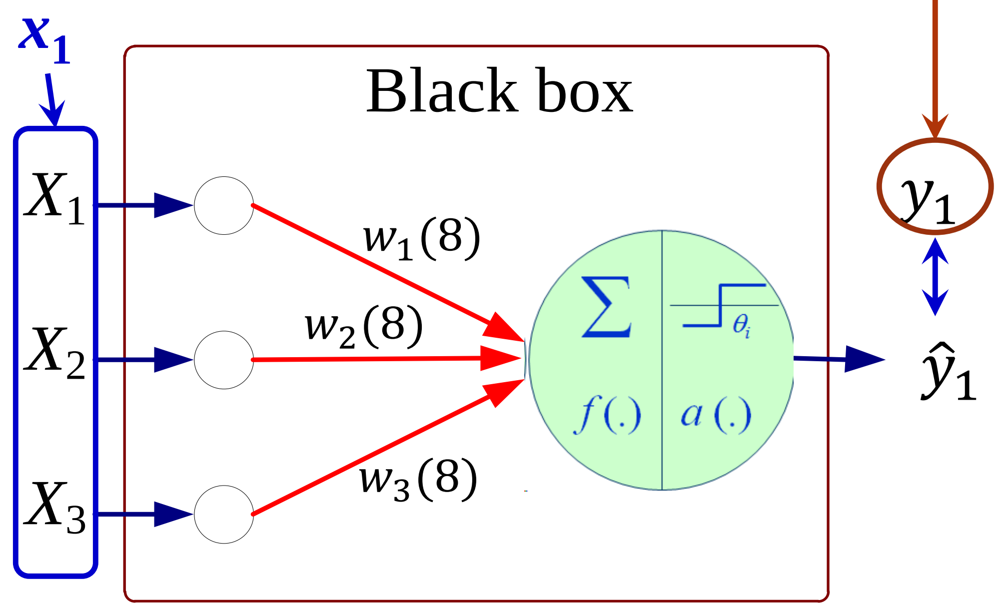



End of the 1st Epoch

The $2^{nd}$ Epoch starts

# Perceptron: Learning (cont.)

| | $X_1$ | $X_2$ | $X_3$ | $y$ |
|---|---|---|---|---|
| $\mathbf{x}_1$ | 1 | 0 | 0 | -1 |
| $\mathbf{x}_2$ | 1 | 0 | 1 | 1 |
| | 1 | 1 | 0 | 1 |
| $\mathbf{x}_i$ | 1 | 1 | 1 | 1 |
| | 0 | 0 | 1 | -1 |
| | 0 | 1 | 0 | -1 |
| | 0 | 1 | 1 | 1 |
| $\mathbf{x}_8$ | 0 | 0 | 0 | -1 |




• • •

# Perceptron: Learning (cont.)

- Algorithm:

$d$ dimensions

1. Let $D = \{(\boxed{\boldsymbol{x}_i}, y_i) \mid i = 1,2, \ldots, N\}$ be the set of training examples, $t = 0$
2. Initialize $\mathbf{w}$ with random values $\mathbf{w}_0$
3. **Repeat**
4. **for** each training example $(\boldsymbol{x}_i, y_i)$ **do**
5. Compute the predicted output $\hat{y}_i$
6. Update $\mathbf{w}_t$ by $\boxed{\mathbf{w}_{t+1} = \mathbf{w}_t + \lambda(y_i - \hat{y}_i)\boldsymbol{x}_i}$
7. $t = t + 1$
8. **end for**
9. **Until** stopping condition is met

# Perceptron: Learning (cont.)

- Why using the following weight update rule?

$$\mathbf{w}_{t+1} = \mathbf{w}_t + \lambda(y_i - \hat{y}_i)\boldsymbol{x}_i$$

- Induced based on a gradient descent method

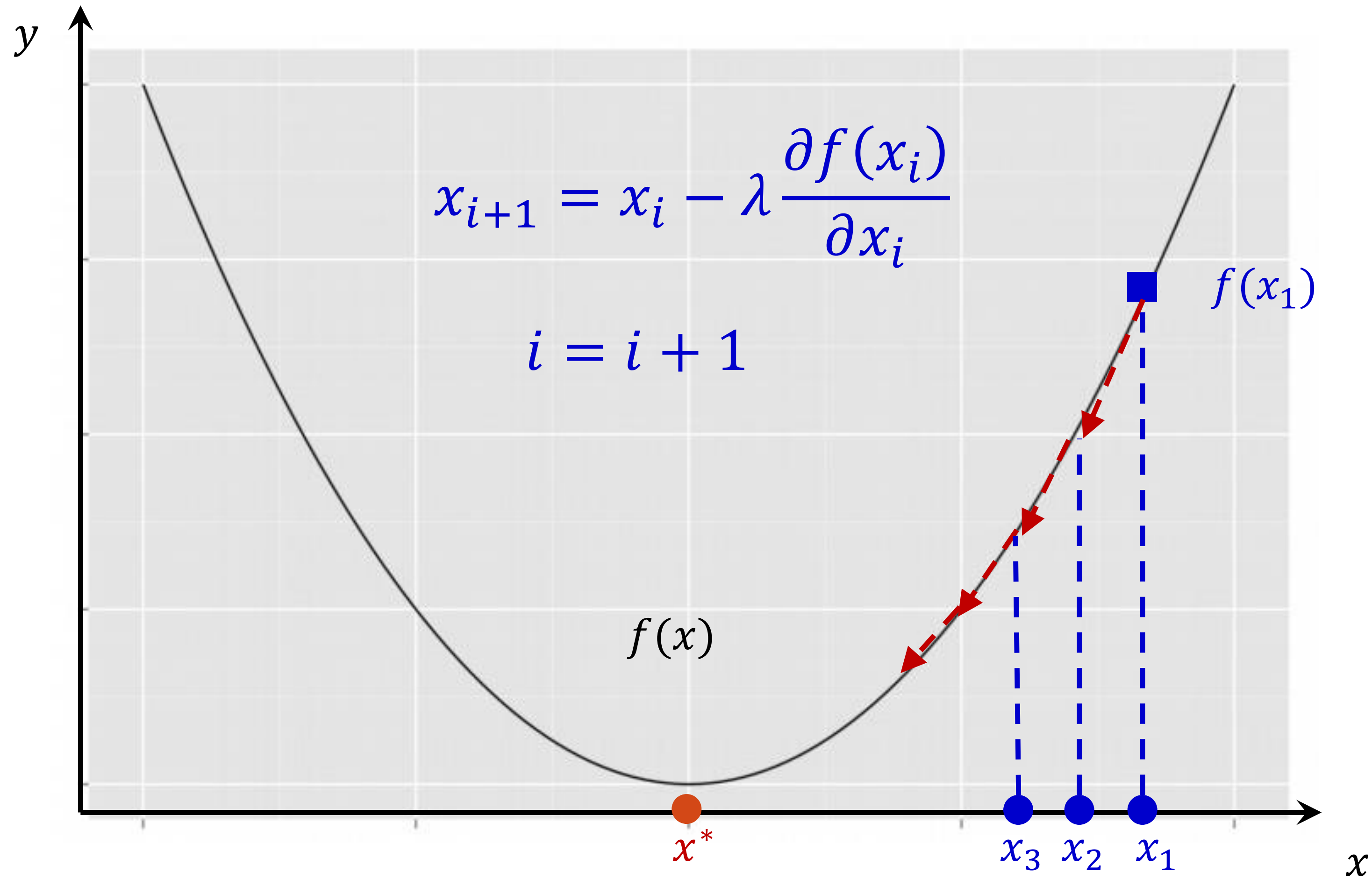
$$\mathbf{w}_{t+1} = \mathbf{w}_t - \boxed{\lambda}\frac{\partial \boxed{E(\mathbf{w})}}{\partial \mathbf{w}}$$

Function to be minimized, e.g., the error function

Learning rate $\lambda \in (0,1]$

# Gradient Descent

$$x^* = \arg\min_x f(x)$$



# Perceptron: Learning (cont.)

- Weight update rule

$$\mathbf{w}_{t+1} = \mathbf{w}_t + \lambda(y_i - \hat{y}_i)\boldsymbol{x}_i \qquad \Leftarrow \qquad \mathbf{w}_{t+1} = \mathbf{w}_t - \lambda\frac{\partial E(\mathbf{w})}{\partial \mathbf{w}}$$

- Consider the loss function for each training example as $E_i \triangleq y_i - \hat{y}_i$ $\qquad \hat{y}_i = \mathrm{sign}(\mathbf{w}_t \cdot \boldsymbol{x}_i)$

$$E = \frac{1}{2}E_i^2 = \frac{1}{2}(y_i - \hat{y}_i)^2 = \frac{1}{2}\left(y_i - \mathrm{sign}(\mathbf{w}_t \cdot \boldsymbol{x}_i)\right)^2$$

  - Update the weight using a gradient descent method

$$\mathbf{w}_{t+1} = \mathbf{w}_t - \lambda\frac{\partial E(\mathbf{w})}{\partial \mathbf{w}} = \mathbf{w}_t - \lambda\boxed{\frac{\partial E(\hat{y})}{\partial \hat{y}}\frac{\partial \hat{y}(\mathrm{z})}{\partial \mathrm{z}}\frac{\partial z(\mathbf{w})}{\partial \mathbf{w}}}$$

Chain rule

$$E = \frac{1}{2}(y - \hat{y})^2 \qquad \hat{y} = \mathrm{sign}(z) \qquad z = \mathbf{w} \cdot \boldsymbol{x}$$

# Chain Rule of Calculus (Review)

- Suppose that $y = g(x)$ and $z = f(y) = f\big(g(x)\big)$
- Chain rule of calculus:

$$\frac{\partial z}{\partial x} = \frac{\partial z}{\partial y}\frac{\partial y}{\partial x}$$

- Generalized to the vector case: suppose $\boldsymbol{x} \in R^m$, $\boldsymbol{y} \in R^n$, $\boldsymbol{y} = g(\boldsymbol{x})$ and $z = f(\boldsymbol{y})$

$$\frac{\partial z}{\partial x_i} = \sum_j \frac{\partial z}{\partial y_j}\frac{\partial y_j}{\partial x_i}$$

# Perceptron: Learning (cont.)

- The weight update formula for perceptron:

$$\mathbf{w}_{t+1} = \mathbf{w}_t - \lambda \frac{\partial E(\mathbf{w})}{\partial \mathbf{w}} = \mathbf{w}_t - \lambda \boxed{\frac{\partial E(\hat{y})}{\partial \hat{y}}} \boxed{\frac{\partial \hat{y}(\mathrm{z})}{\partial \mathrm{z}}} \boxed{\frac{\partial z(\mathbf{w})}{\partial \mathbf{w}}}$$

$$E = \frac{1}{2}(y_i - \hat{y}_i)^2$$

$$\cancel{\hat{y}_i = \mathrm{sign}(z_i)}$$

$$\hat{y}_i = z_i$$

$$(y_i - \hat{y}_i) \times -1$$

$$1$$

$$z_i = \mathbf{w}_t \cdot \boldsymbol{x}_i$$

*Note: the sign() function is not differentiable everywhere. Therefore, to compute the derivative, the sign() function is removed*

$$\boldsymbol{x}_i$$

$$\mathbf{w}_{t+1} = \mathbf{w}_t + \lambda(y_i - \hat{y}_i)\boldsymbol{x}_i$$

# Perceptron Weights Update

$$\mathbf{w}_{t+1} = \mathbf{w}_t + \lambda(y_i - \hat{y}_i)\boldsymbol{x}_i$$

- If the prediction is correct, $(y - \hat{y}) = 0$, then weight remains unchanged $\mathbf{w}_{t+1} = \mathbf{w}_t$
- If $y = +1$ and $\hat{y} = -1$, then $(y - \hat{y}) = 2$
- The weights of all links with positive inputs need to be updated by increasing their values
- The weights of all links with negative inputs need to be updated by decreasing their weights

| | | | | | |
|---|---|---|---|---|---|
| $\boldsymbol{x}_i$ | $x_{i1}$ | … | $x_{ik}$ | … | $x_{id}$ |
| | $> \mathbf{0}$ | | $= \mathbf{0}$ | | $< \mathbf{0}$ |
| $\mathbf{w}_{t+1}$ | $w_1 \uparrow$ | … | $w_k$ | … | $w_d \downarrow$ |

# Perceptron Weights Update (cont.)

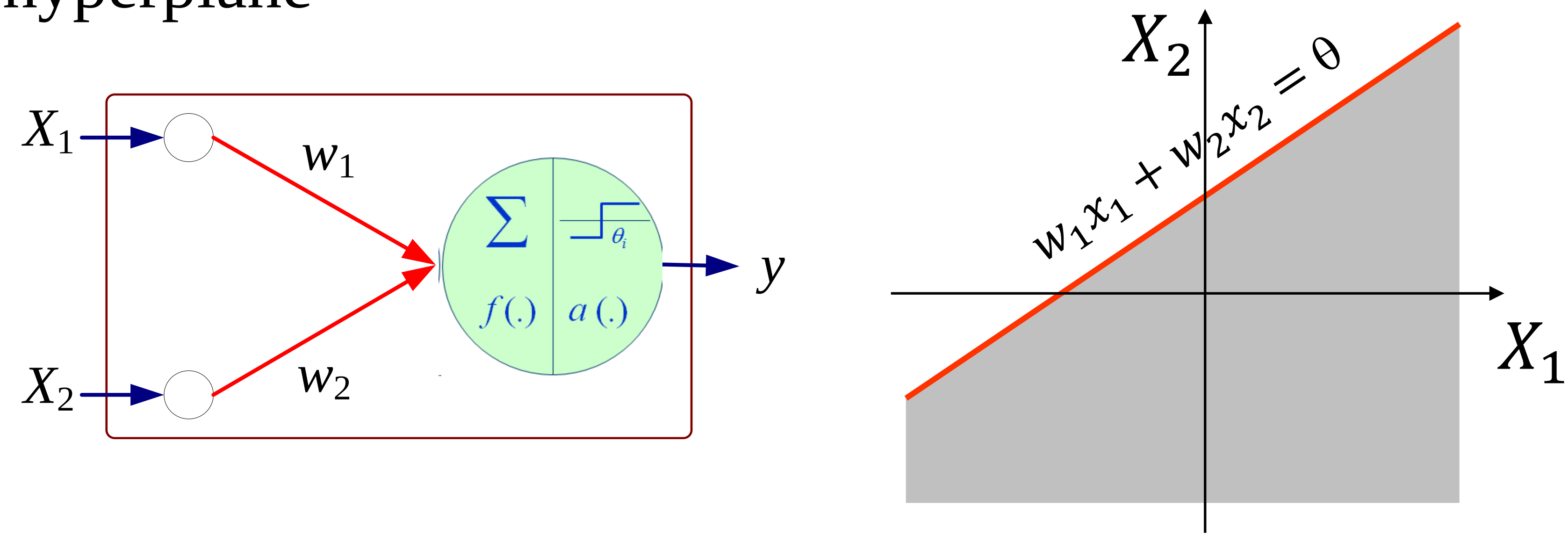
$$\mathbf{w}_{t+1} = \mathbf{w}_t + \lambda(y_i - \hat{y}_i)\boldsymbol{x}_i$$

- If $y = -1$ and $\hat{y} = +1$, then $(y - \hat{y}) = -2$
- The weights of all links with positive inputs need to be updated by decreasing their values
- The weights of all links with negative inputs need to be updated by increasing their weights

| | | | | | |
|---|---|---|---|---|---|
| $\boldsymbol{x}_i$ | $X_{i1}$ | … | $X_{ik}$ | … | $X_{id}$ |
| | $> \mathbf{0}$ | | $= \mathbf{0}$ | | $< \mathbf{0}$ |
| $\mathbf{w}_{t+1}$ | $w_1 \downarrow$ | … | $w_k$ | … | $w_d \uparrow$ |

# Convergence

- The decision boundary of a perceptron is a linear hyperplane




- The perceptron learning algorithm is guaranteed to converge to an optimal solution for linear classification problems

# Perceptron Limitation

- If the problem is not linearly separable, the algorithm fails to converge

Nonlinearly separable data given by the XOR function

| $X_1$ | $X_2$ | y |
|---|---|---|
| 0 | 0 | −1 |
| 1 | 0 | 1 |
| 0 | 1 | 1 |
| 1 | 1 | −1 |