CZ4041/SC4000: Machine Learning

Lesson 7: Artificial Neural Networks

LI Boyang, Albert

School of Computer Science and Engineering,
NTU, Singapore

Instructor's Information

- LI Boyang, Albert
- Nanyang Associate Professor, SCSE
- Previously: Disney Research Pittsburgh, Baidu Research USA.
- **Email:** boyang.li@ntu.edu.sg

Full-time Arrangement

- Lectures
  - 3.30-5.30pm on Thursdays, LT2A
  - Weeks 7-12
- Tutorials
  - 3.30-4.30pm on Mondays, LT2A
  - Weeks 9, 11 and 13
  - Week 12 (3.30-4.30pm, 4 Apr), guest lecture from GovTech: Analysing Feedback on Municipal Issues with NLP and Deep Learning.

Part-time Arrangement

- Lectures & QA sessions
  - 6.30-8.30pm on Thursdays, TR+3.
  - Weeks 8-13
  - 1-hour review, followed by one hour of QA
  - It is highly recommended that you watch the lecture & tutorial videos uploaded in the previous week.

I have questions …

- Full-time students: Find me after the tutorials (but not the lectures)
- Part-time students: Dedicated QA time on Thursdays
- Send questions via email boyang.li@ntu.edu.sg or via Microsoft Teams
- Make an appointment

Purpose of Machine Learning

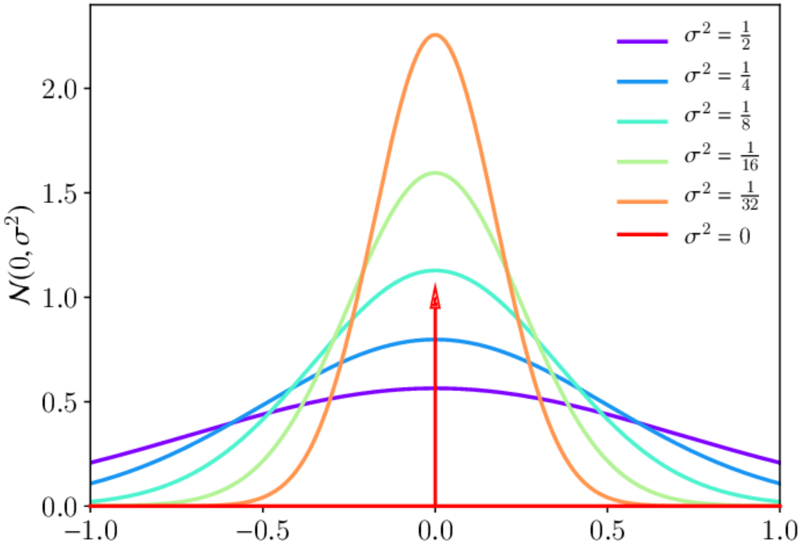
- Make Accurate Predictions
  - Which team will win a soccer match?
  - Which stock will see its price skyrocket?
  - Which patient is at higher risk?
- Usually it is difficult to write down rules manually
- Rather, we learn to make the predictions from paired data $(\boldsymbol{x}_i, y_i)$

Purpose of Machine Learning

- Make Accurate Predictions
  - Artificial Neural Networks (Week 7)
  - Support Vector Machines (Week 8)
  - Regression (Week 8)
  - Ensemble Learning (Week 9)

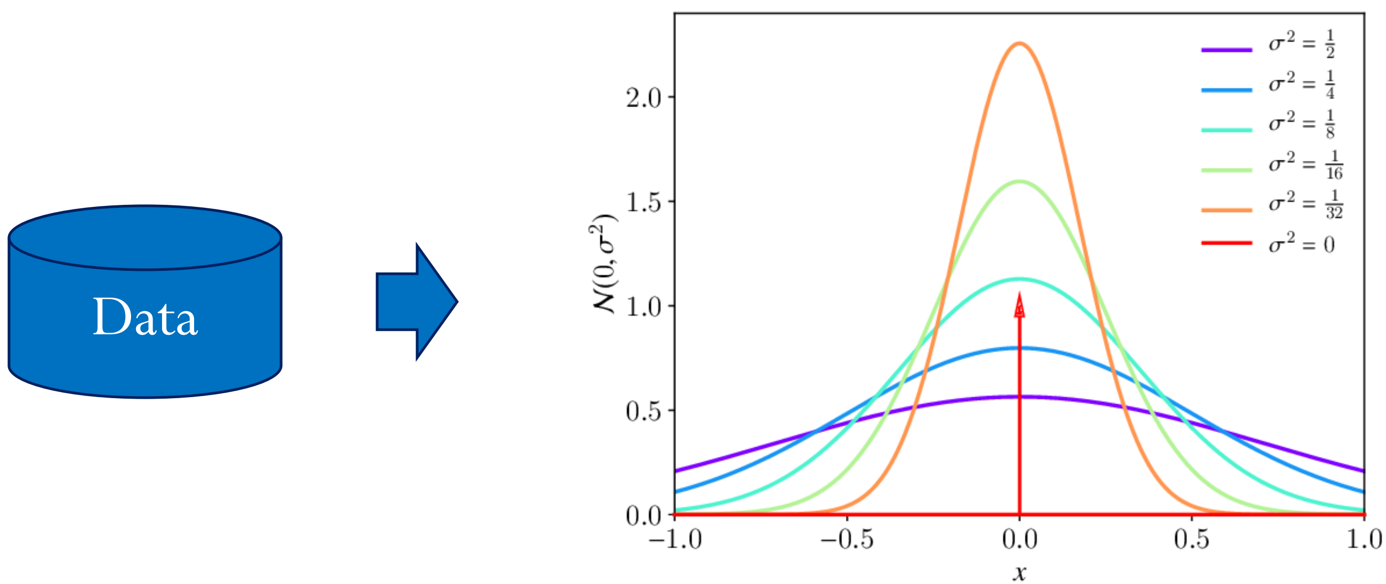
Purpose of Machine Learning

- Model Uncertainty
  - Germany will probably beat Japan, but what are the odds? 60/40, 70/30, or 80/20?
  - I’m willing to bet more money if the odds are in my favor.
  - Often translates to: what is the shape of the probability distribution?



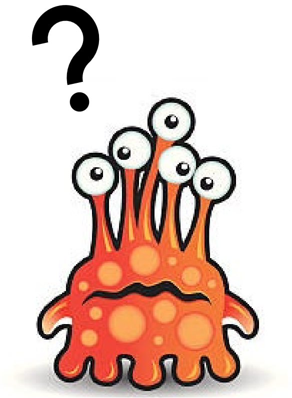
Purpose of Machine Learning

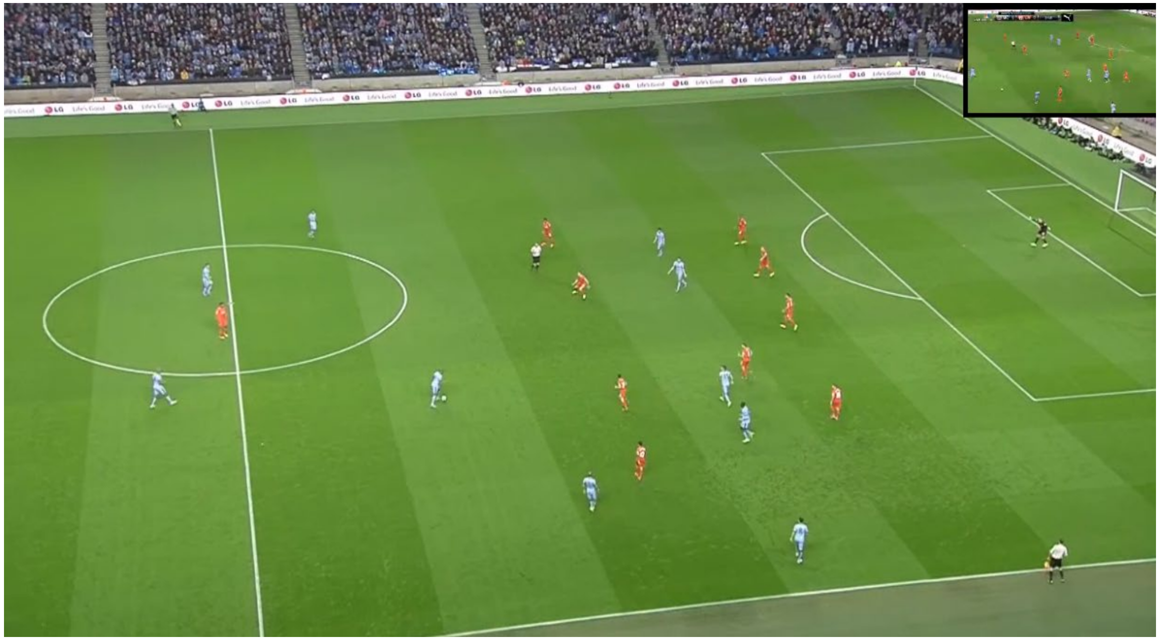
- Model Uncertainty
  - Density Estimation (Week 11)



Purpose of Machine Learning

- Pattern Discovery
  - Imagine you are an alien from another planet. You watch a soccer match. What do you observe?
  - Two groups of humans. One ball.
  - Behavior change when the ball goes into the net.





Purpose of Machine Learning

- Pattern Discovery
  - We often have little prior experience, knowledge, or insight into the causal mechanisms that generated the data.
  - Still, with only statistical tools, we can identify many important data characteristics
  - Obviously, domain knowledge can enrich and complement statistical tools.

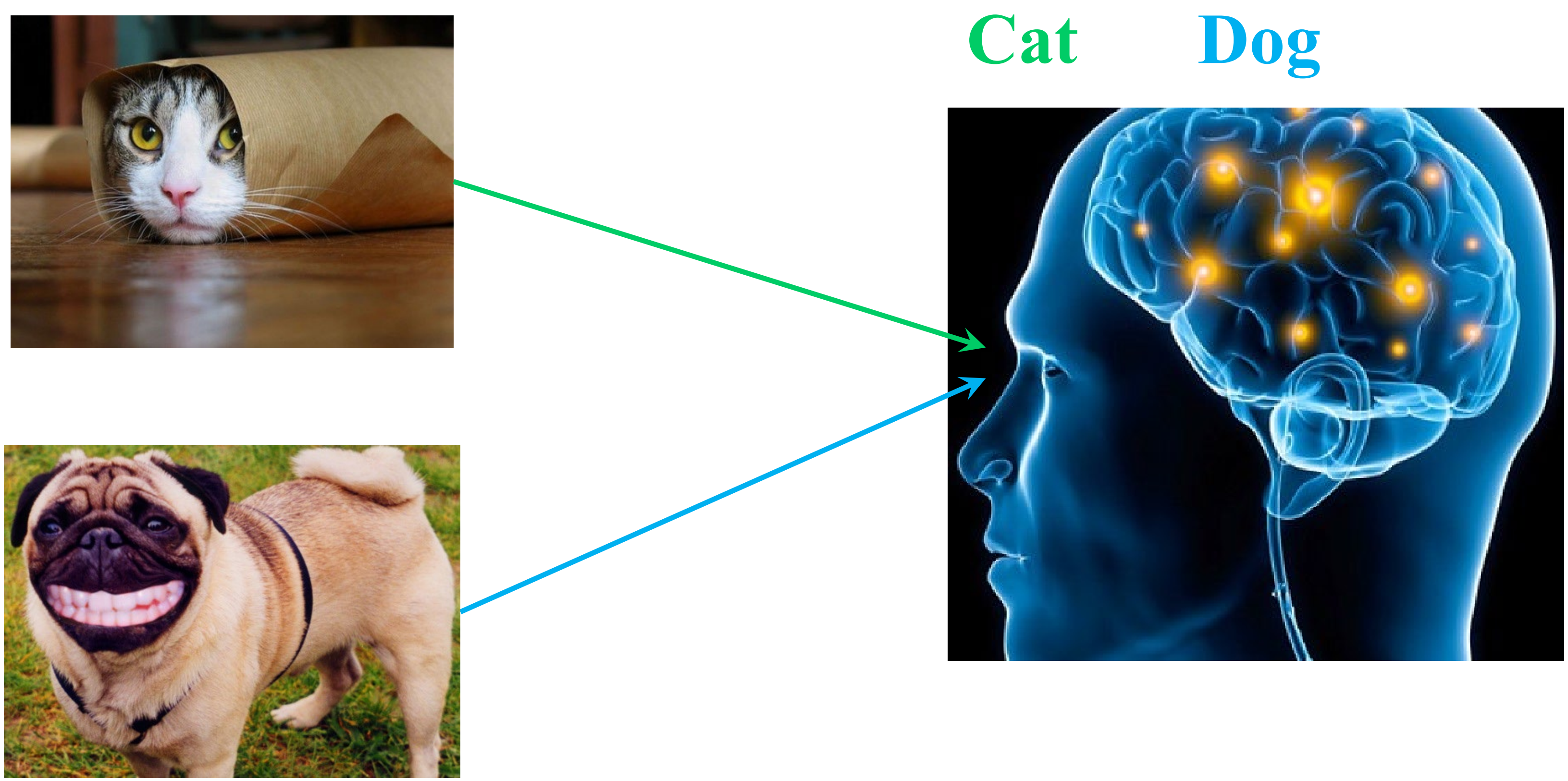
Purpose of Machine Learning

- Pattern Discovery
  - Clustering (Week 10)
  - Dimensionality Reduction (Week 12)

Artificial Neural Networks: Perceptron

Artificial Neural Networks (ANN)

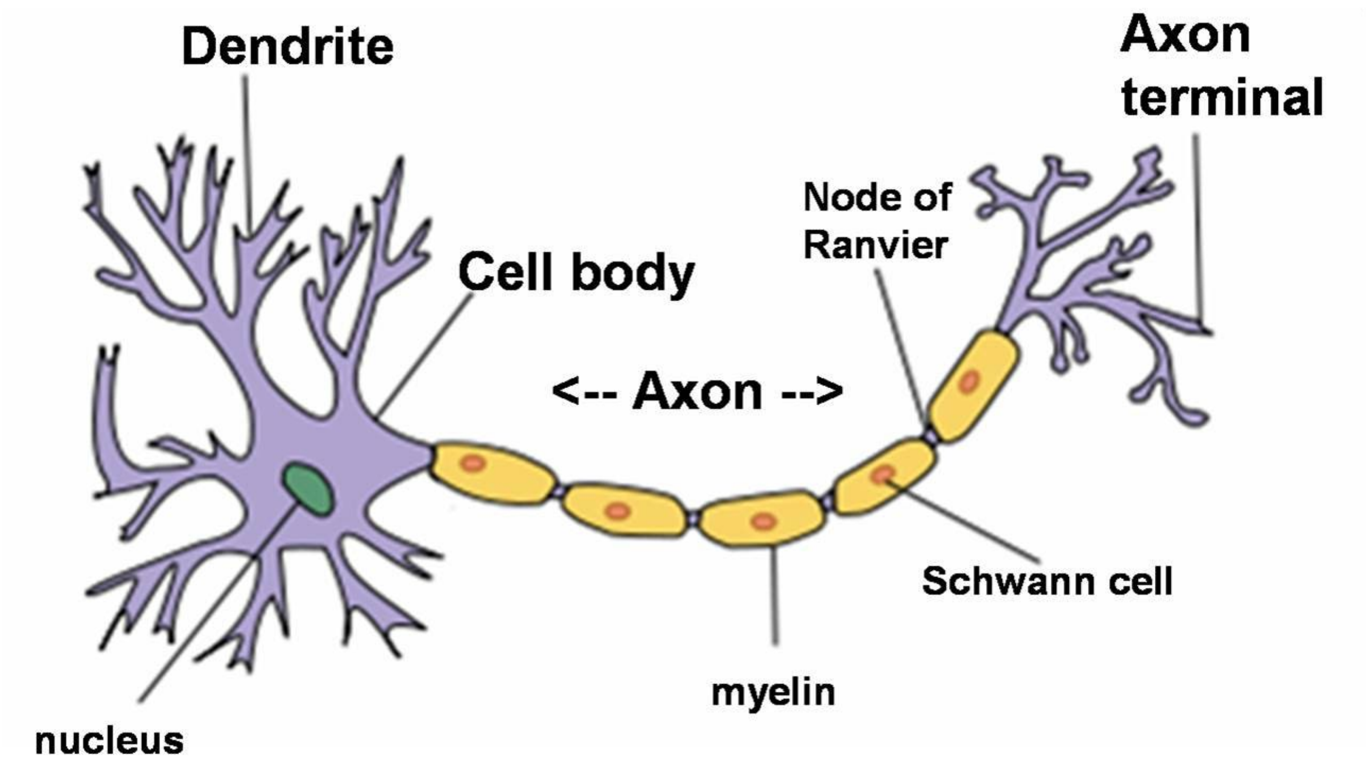
- The study of ANN was inspired by biological neural systems



"Biology"

- Human brain is a densely interconnected network of neurons, connected to others via dendrites and axons.
- Dendrites and axons transmit electrical signals 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.

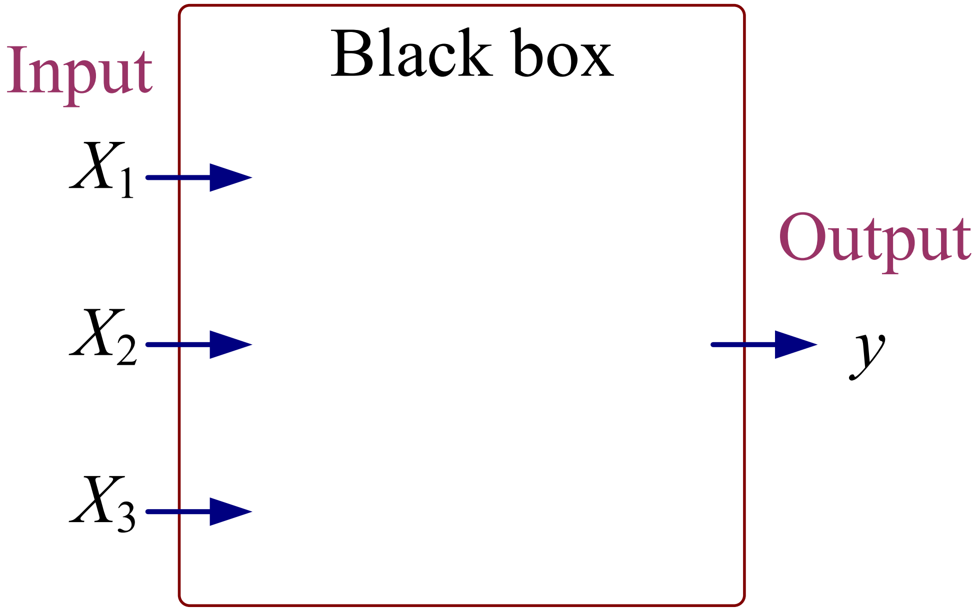
“Biology”



- A neuron sends out a spike from the axon after receiving enough input from the dendrites.

Artificial Neural Networks (cont.)

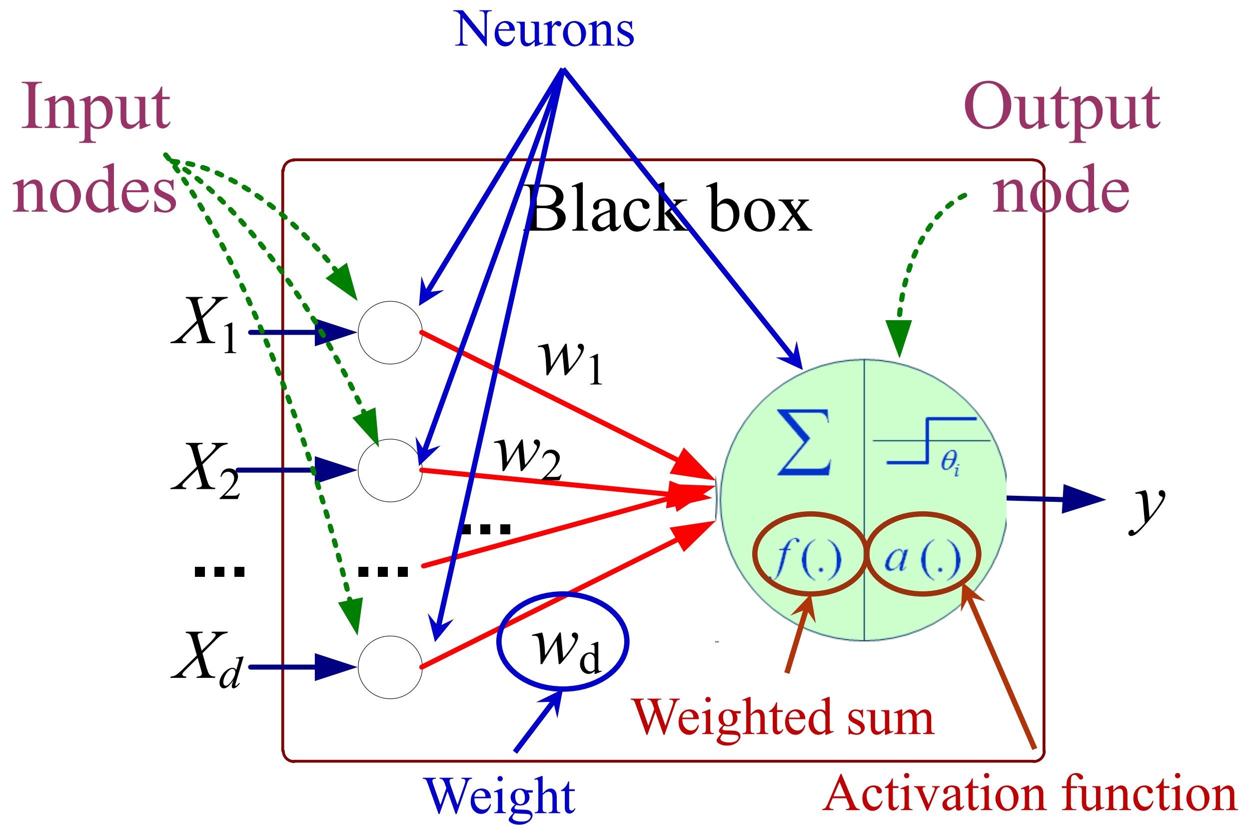
X_1	X_2	X_3	y
1	0	0	-1
1	0	1	1
1	1	0	1
1	1	1	1
0	0	1	-1
0	1	0	-1
0	1	1	1
0	0	0	-1



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

ANN: Perceptron

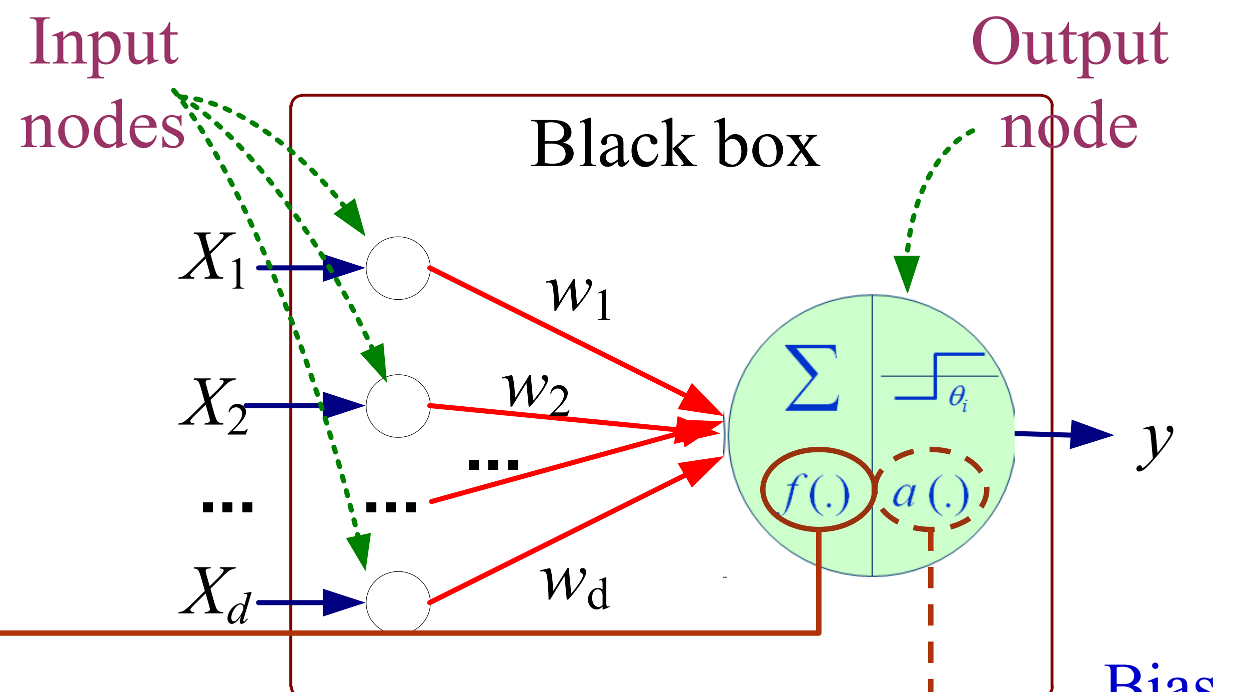
X_1	X_2	…	X_d
1	0	…	0
1	0	…	1
1	1	…	0
1	1	…	1
0	0	…	1
0	1	…	0
0	1	…	1
0	0	…	0



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.)

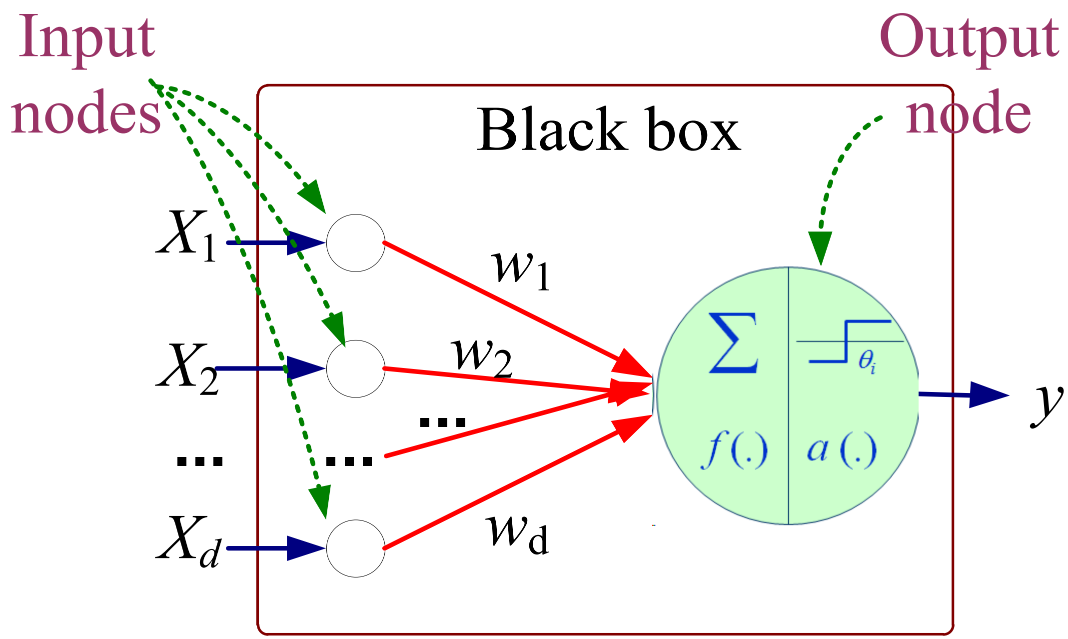
X_1	X_2	…	X_d
1	0	…	0
1	0	…	1
1	1	…	0
1	1	…	1
0	0	…	1
0	1	…	0
0	1	…	1
0	0	…	0



Bias factor

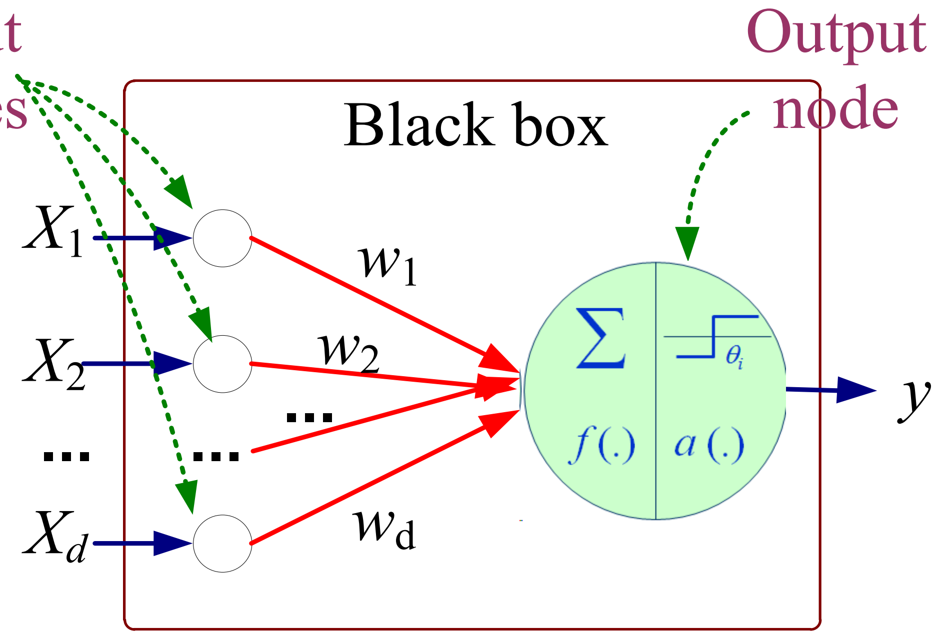
$$z = f(\mathbf{x}, \mathbf{w}) = w_1x_1 + w_2x_2 + \cdots + w_dx_d - \theta$$

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



- 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 θ
- Produce an output based on the sign of the result

Perceptron Model



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

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

ANN: Perceptron (cont.)

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

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

$$\Downarrow$$

Inner product

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

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

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

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

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

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

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

ANN: Perceptron (cont.)

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

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

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

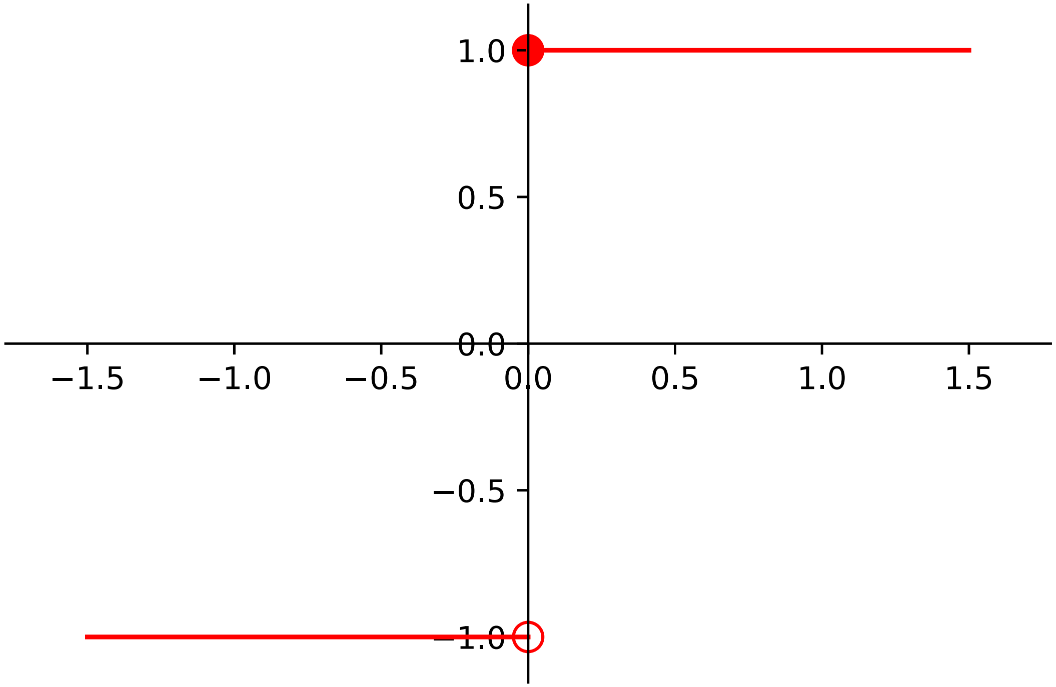
$$\mathbf{w} \cdot \boldsymbol{x} = \sum_{i=0}^{d} (w_i x_i) = \sum_{i=1}^{d} (w_i x_i) + w_0 x_0$$

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

$$-\theta$$

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

ANN: Sign Function



$$\text{sign}(z) = \begin{cases} 1, z \geq 0 \\ -1, z < 0 \end{cases}$$

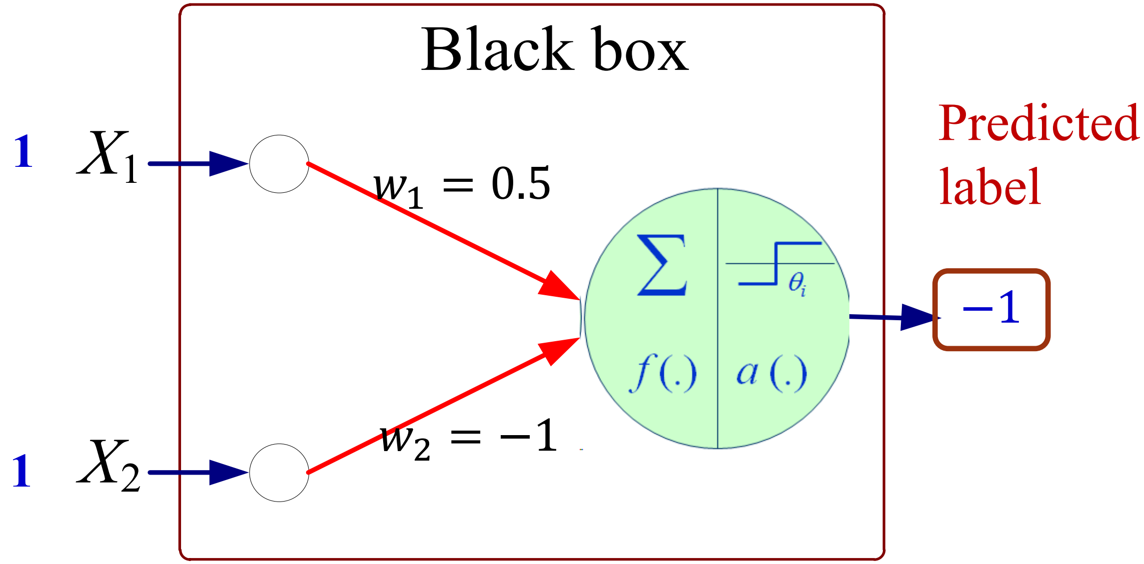
Note: the sign(z) function has derivative = 0 everywhere, except at z = 0.

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

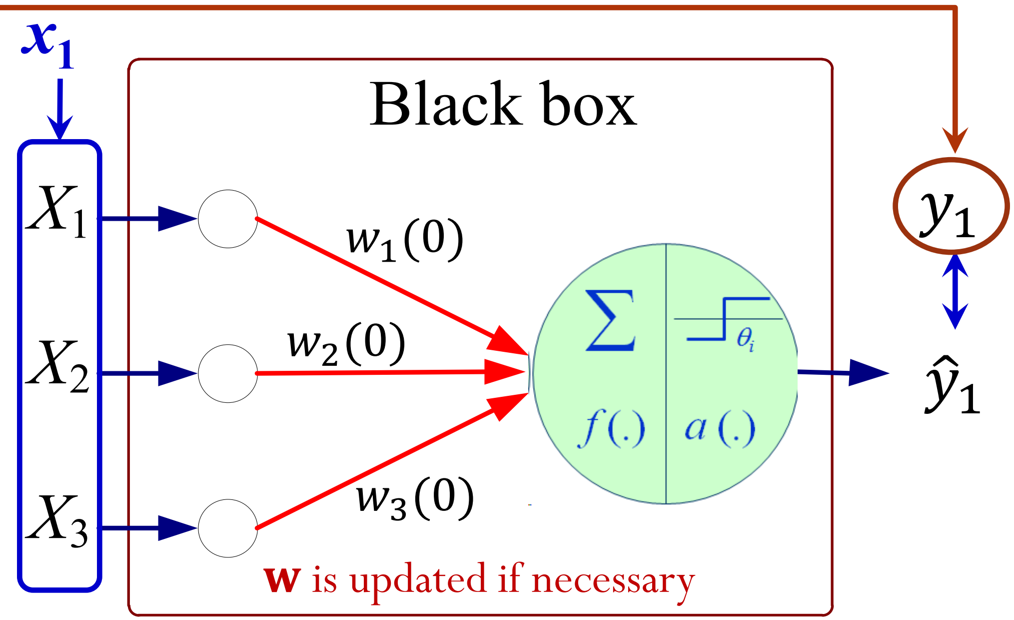


$$y = \text{sign}\big(1 \times 0.5 + 1 \times (-1)\big)$$
$$= \text{sign}(-0.5) \ = -1$$

Perceptron: Learning

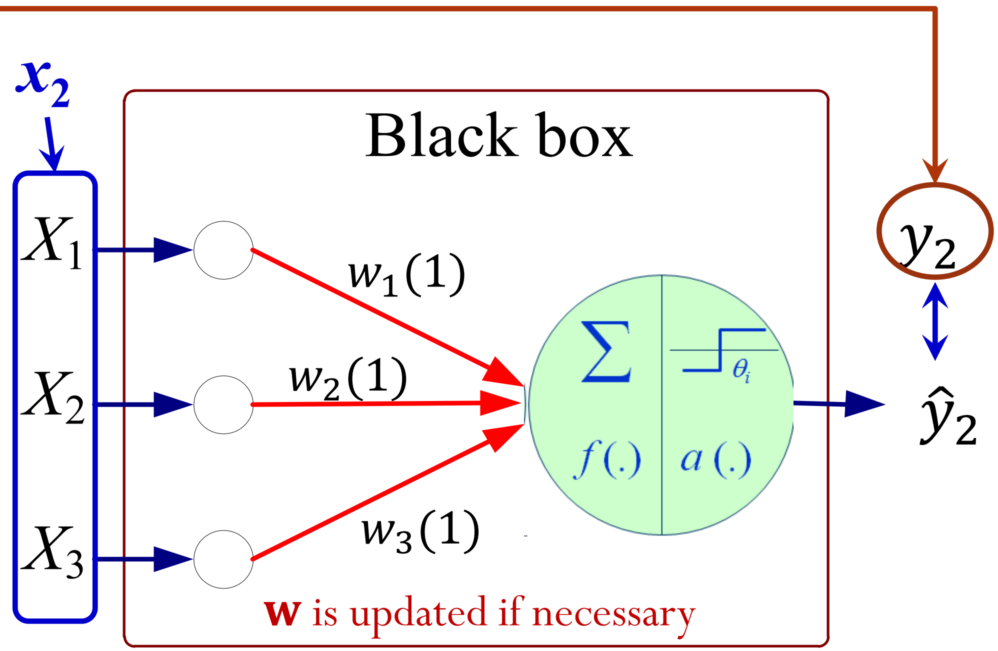
- During training, the weight parameters **w** are adjusted until the outputs of the perceptron become consistent with the true outputs of training data
- The weight parameters **w** are updated iteratively or in an online learning manner

	X_1	X_2	X_3	y
$\boldsymbol{x}_1$	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



Perceptron: Learning (cont.)

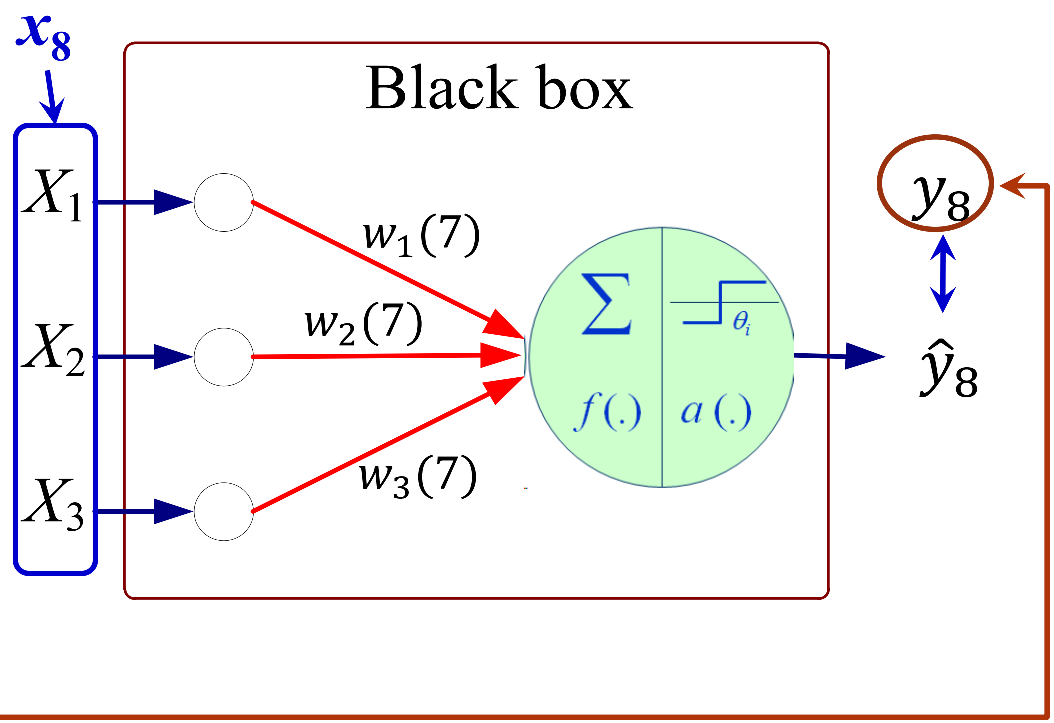
	X_1	X_2	X_3	Y
$\boldsymbol{x}_1$	1	0	0	-1
$\boldsymbol{x}_2$	1	0	1	1
	1	1	0	1
$\boldsymbol{x}_i$	1	1	1	1
	0	0	1	-1
	0	1	0	-1
	0	1	1	1
$\boldsymbol{x}_8$	0	0	0	-1



• • •

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

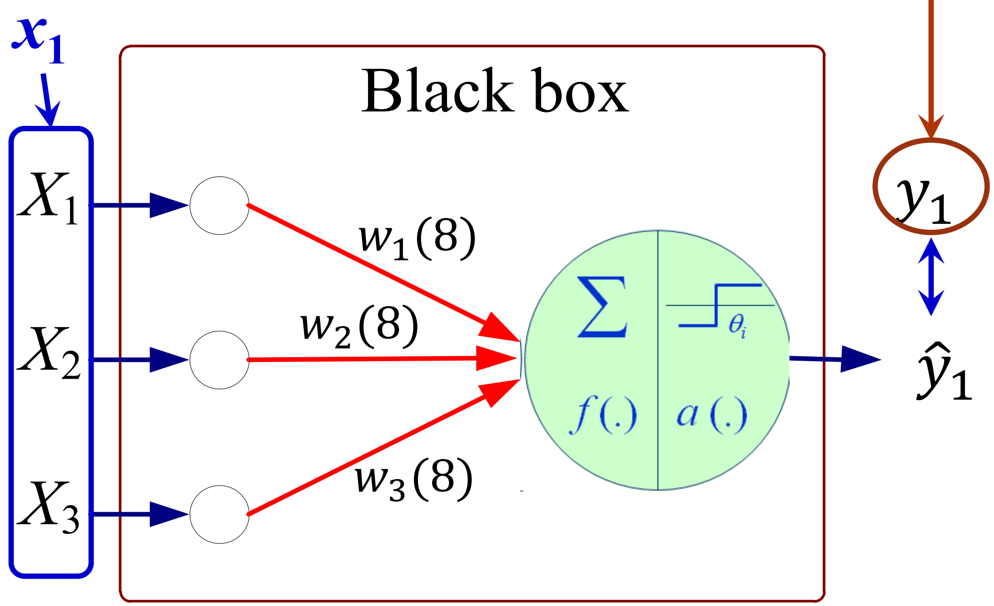


End of the 1st Epoch

The 2^{nd} Epoch starts

Perceptron: Learning (cont.)

	X_1	X_2	X_3	y
$\boldsymbol{x}_1$	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



• • •

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 use 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

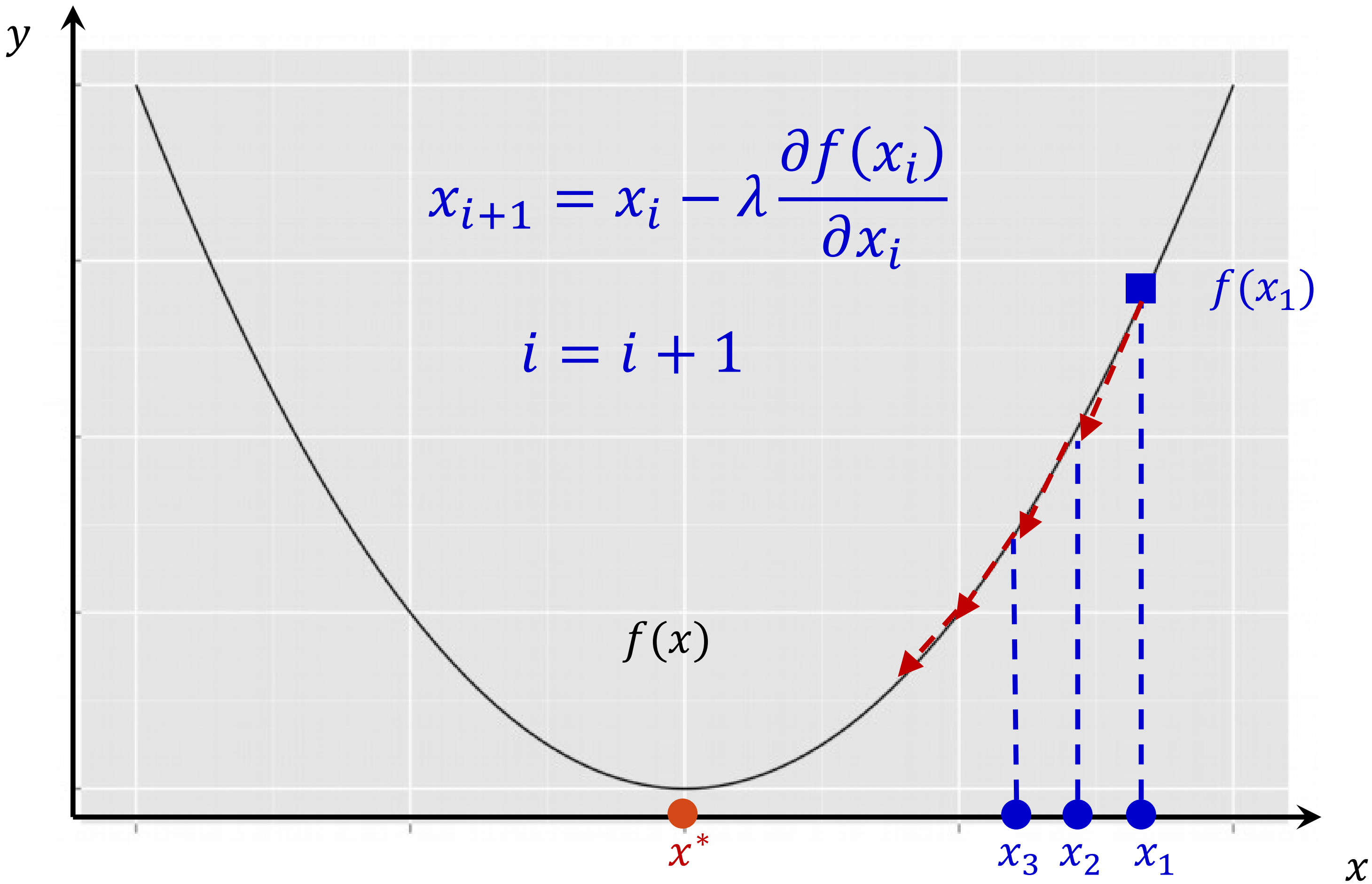
$$\mathbf{w}_{t+1} = \mathbf{w}_t - \boxed{\lambda}\frac{\partial \boxed{\mathcal{L}(\mathbf{w})}}{\partial \mathbf{w}}$$

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

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

Gradient Descent

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



Perceptron: Learning (cont.)

- Weight update rule

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

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

$$\mathcal{L} = \frac{1}{2}e_i^2 = \frac{1}{2}(y_i - \hat{y}_i)^2 = \frac{1}{2}\big(y_i - \mathrm{sign}(\mathbf{w}_t \cdot \boldsymbol{x}_i)\big)^2$$

  - Update the weight using a gradient descent method

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

Chain rule

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

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:

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

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

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

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

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

Note: the sign() function has derivative = 0 except at z = 0. Therefore, to compute meaningful derivative, we remove sign() from consideration

$$1$$

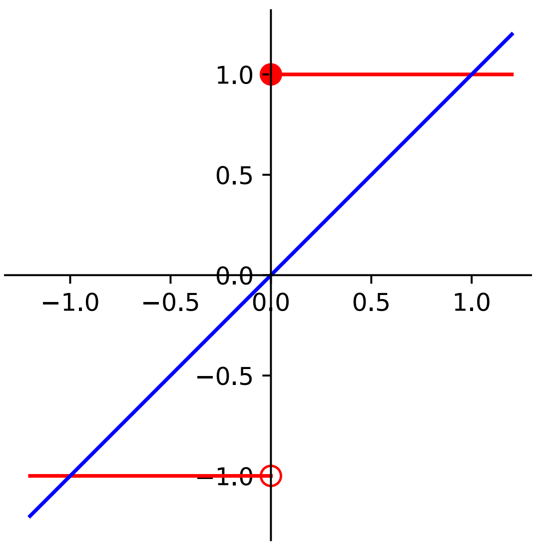
$$z_i = \mathbf{w}_t \cdot \boldsymbol{x}_i$$

$$\boldsymbol{x}_i$$

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

Approximating the derivative?

- The equation used to compute $\hat{y}_i$ from z_i
$$\hat{y}_i = \text{sign}(z_i)$$
- The equation used to compute $\frac{\partial \hat{y}(z)}{\partial z}$
$$\hat{y}_i = z_i$$
- Why? This is approximating the step function with a linear function.



While the derivative itself is incorrect, its direction is correct.
That is, if you want to increase $\hat{y}_i$, you should increase z_i, and vice versa.

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}
	> 0		$= 0$		< 0
$\mathbf{w}_{t+1}$	$w_1 \uparrow$	…	w_k	…	$w_d \downarrow$

Perceptron Weights Update (cont.)

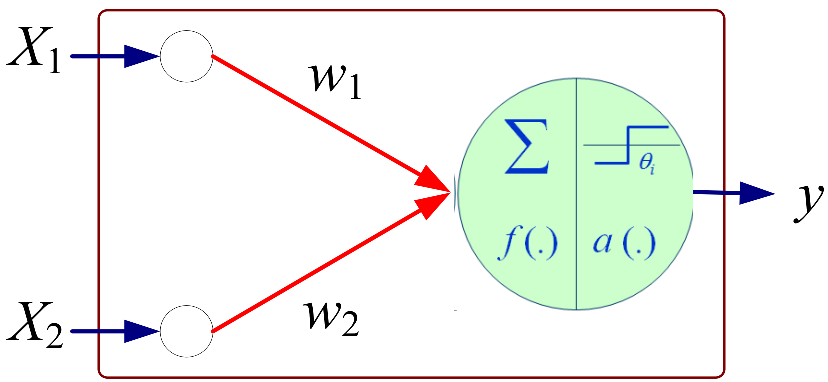
$$\mathbf{w}_{t+1} = \mathbf{w}_t + \lambda(y_i - \hat{y}_i)\boldsymbol{x}_i$$

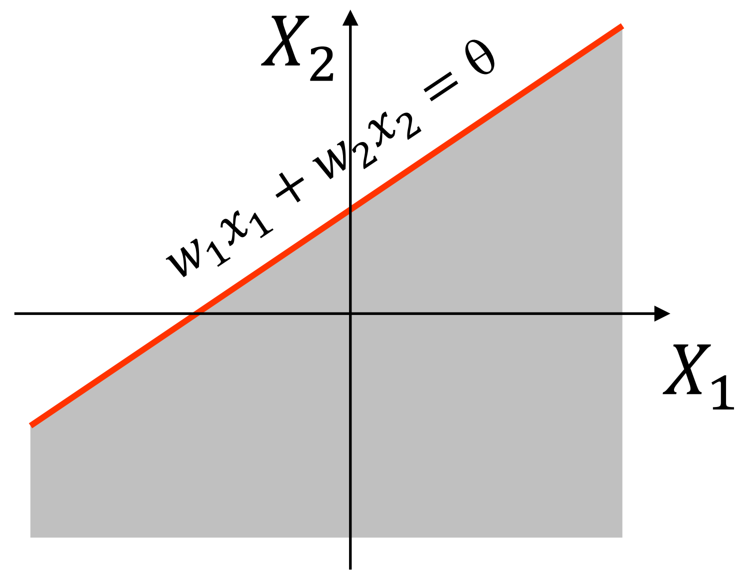
- If $y = -1$ and $\hat{y} = +1$, then $(y - \hat{y}) = -2$
- The weights of all links with positive inputs need to be updated by decreasing their values
- The weights of all links with negative inputs need to be updated by increasing their weights

$\boldsymbol{x}_i$	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





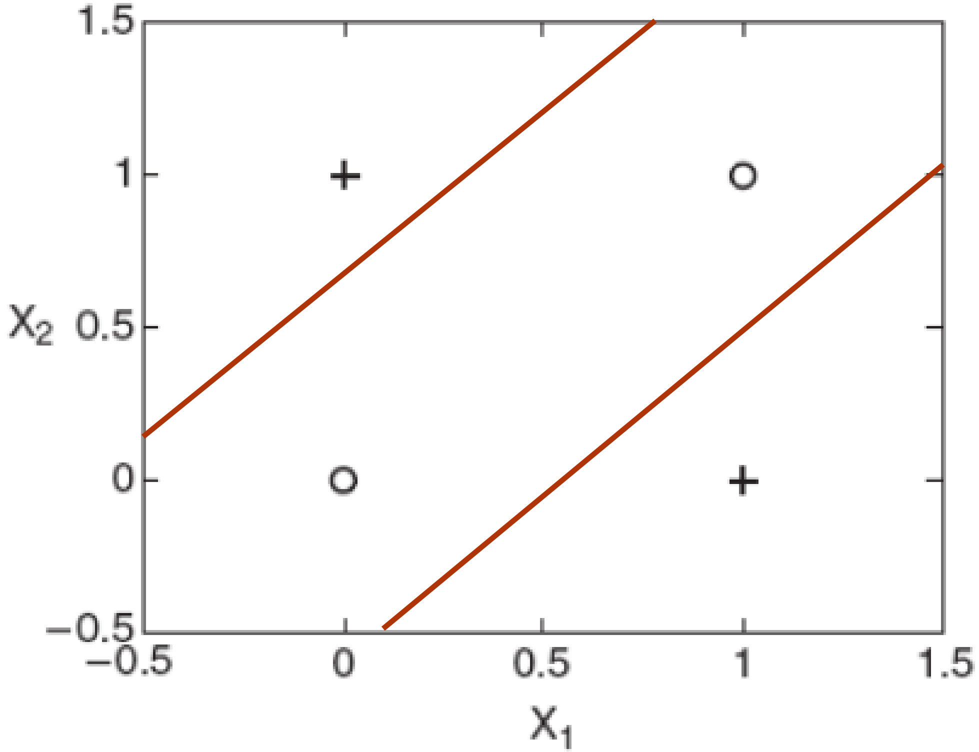
- The perceptron learning algorithm is guaranteed to converge to an optimal solution for linear classification problems

Perceptron Limitation

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

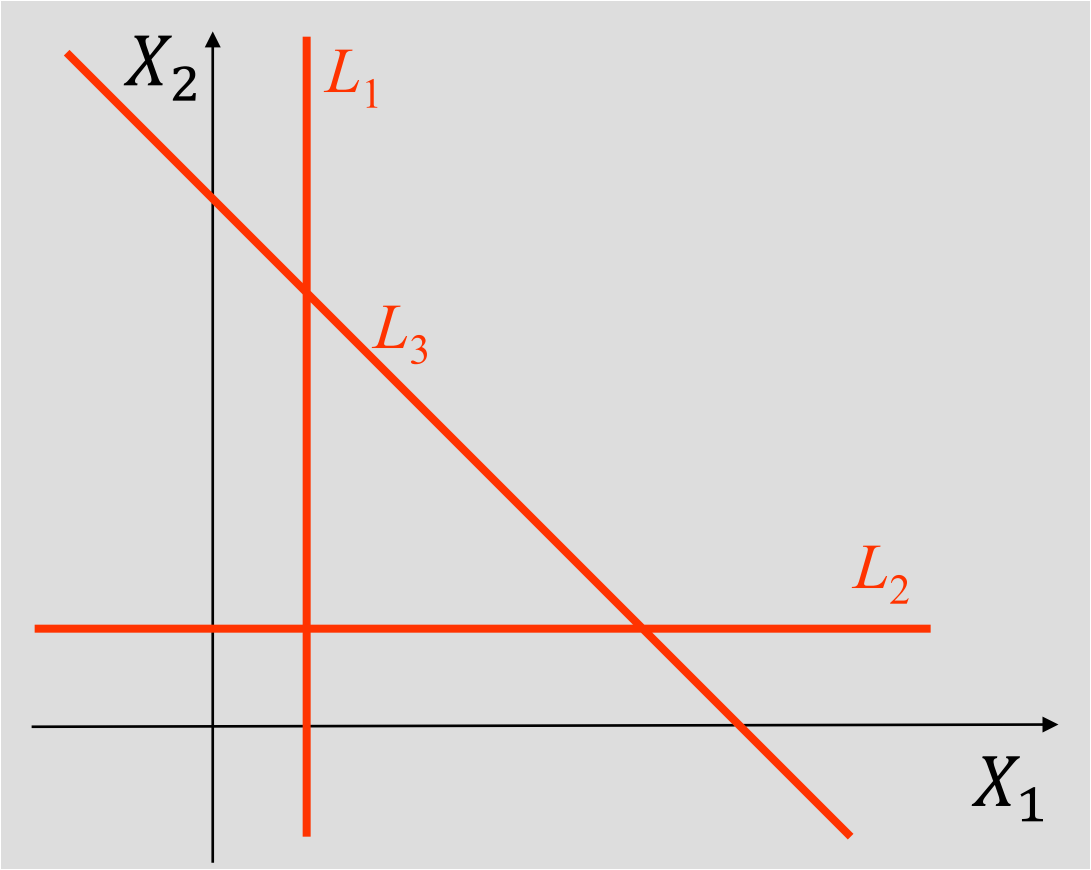
Nonlinearly separable data given by the XOR function

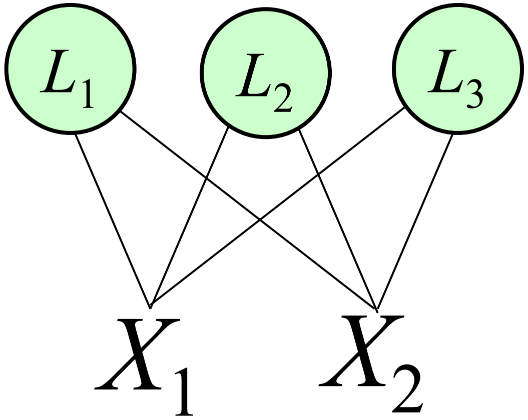
X_1	X_2	y
0	0	−1
1	0	1
0	1	1
1	1	−1



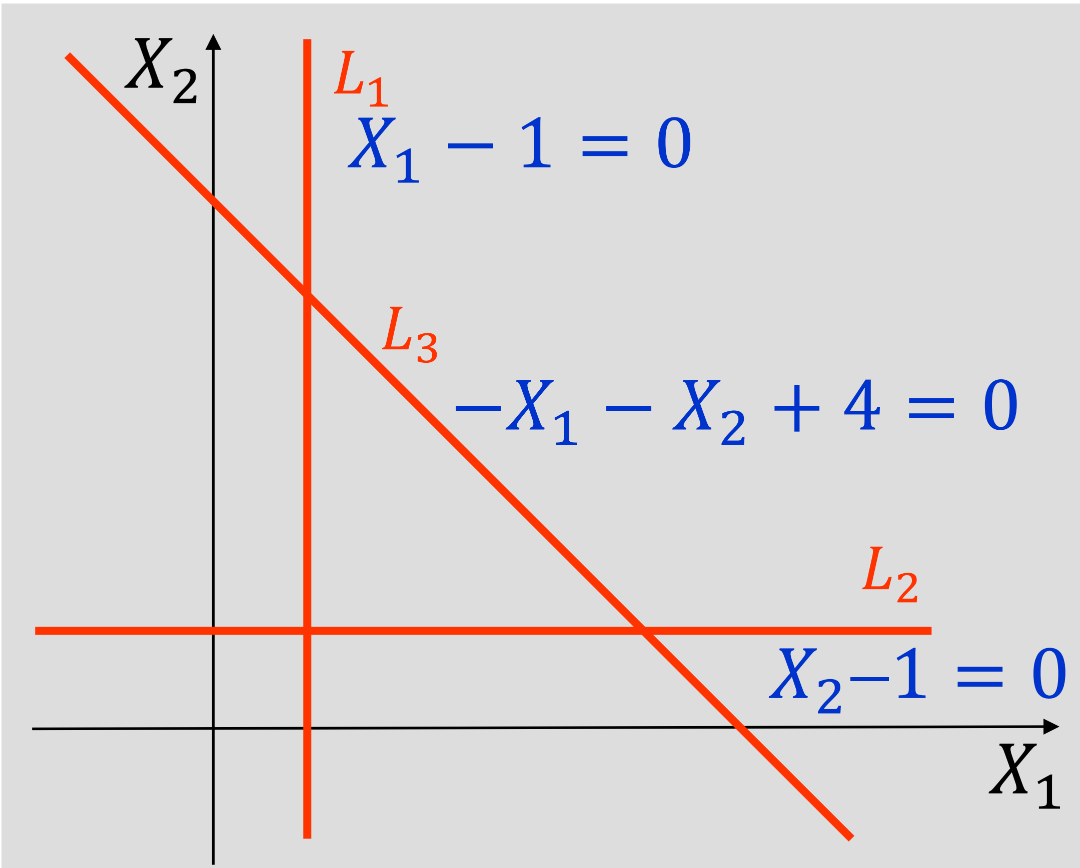
Multi-layer Perceptron

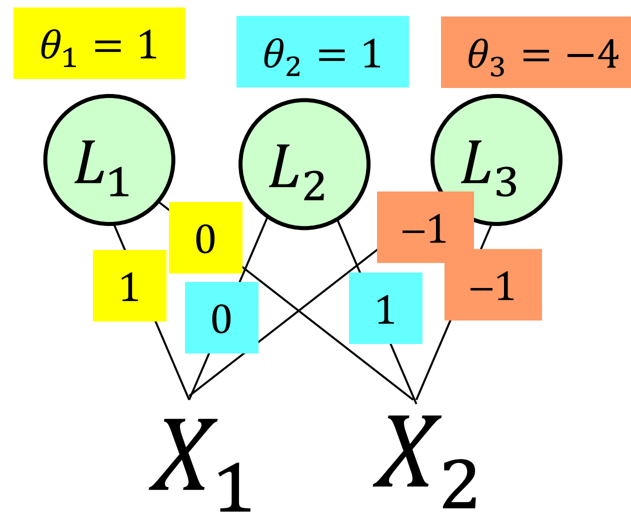
Example: Not Linearly Separable



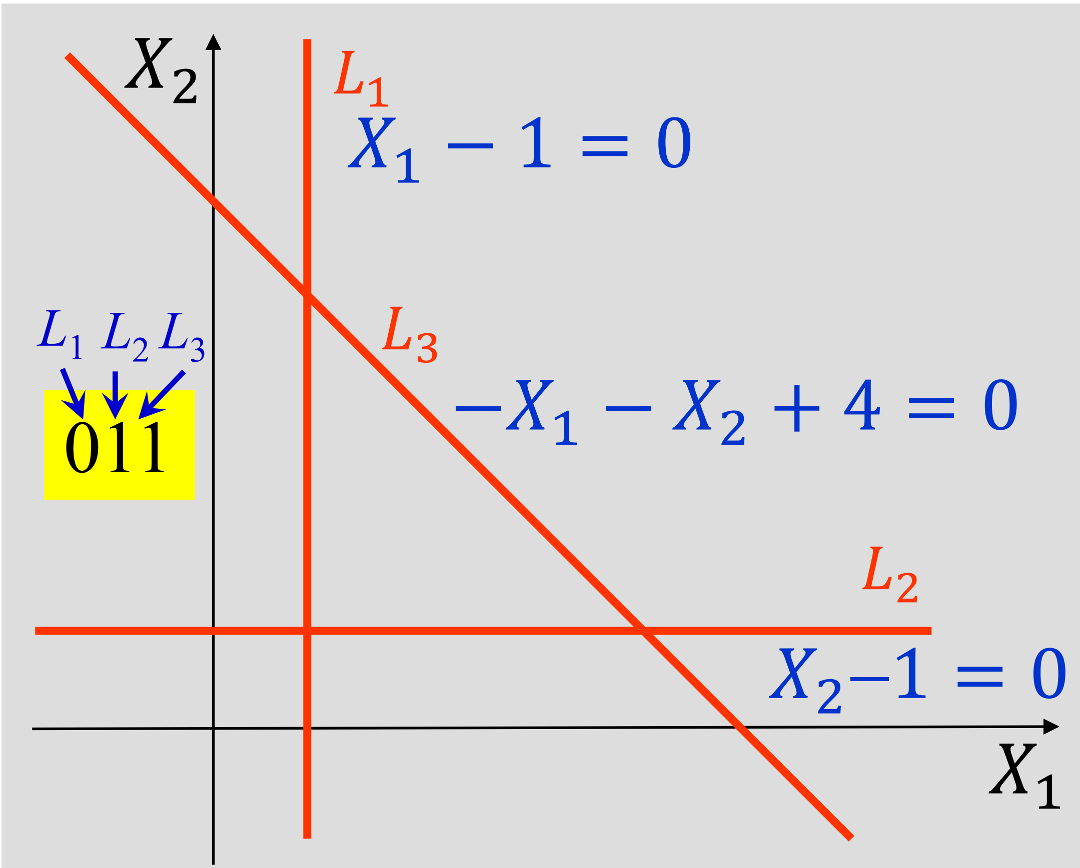


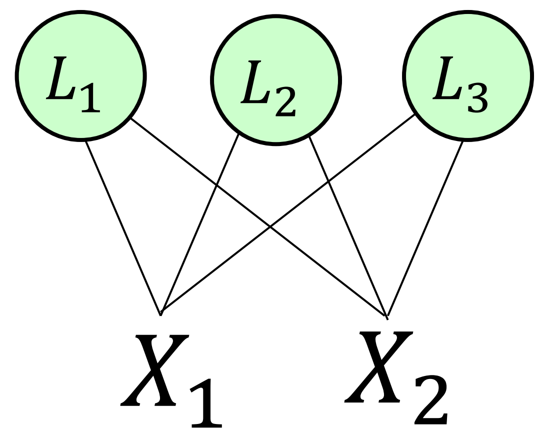
Example: Not Linearly Separable



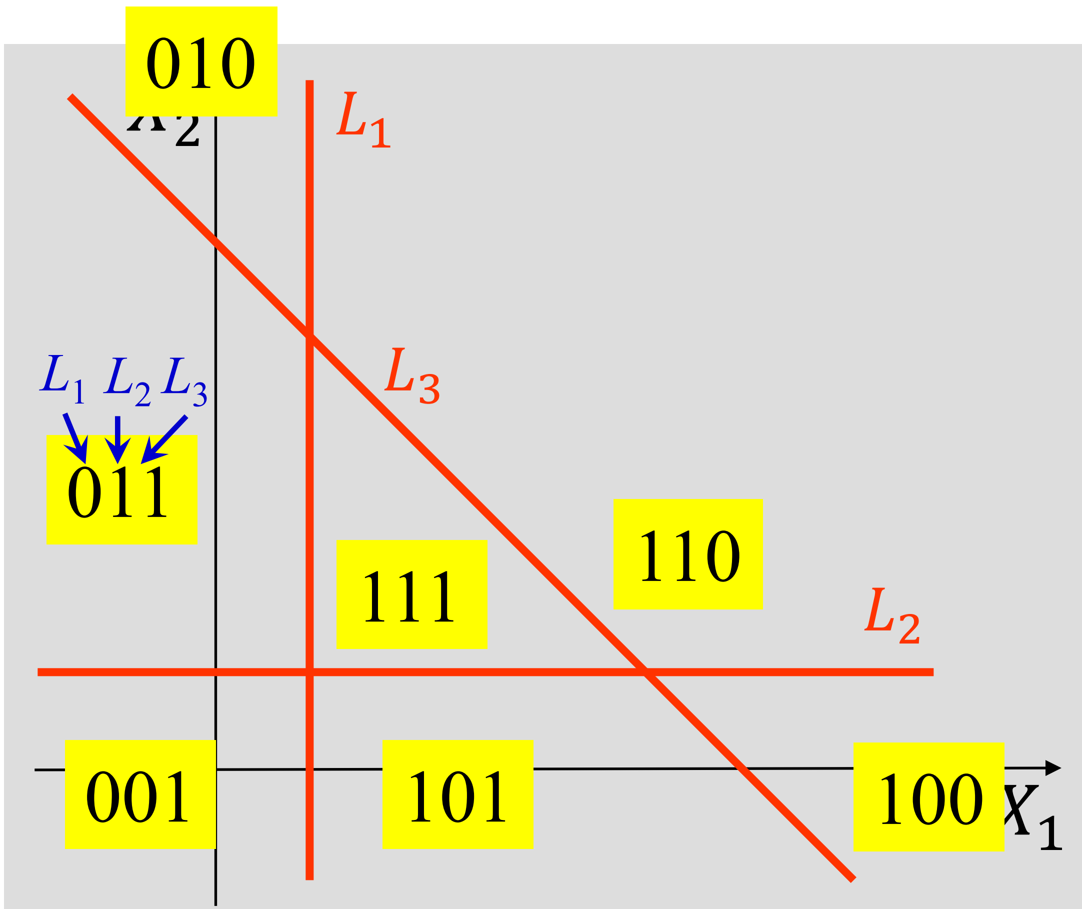


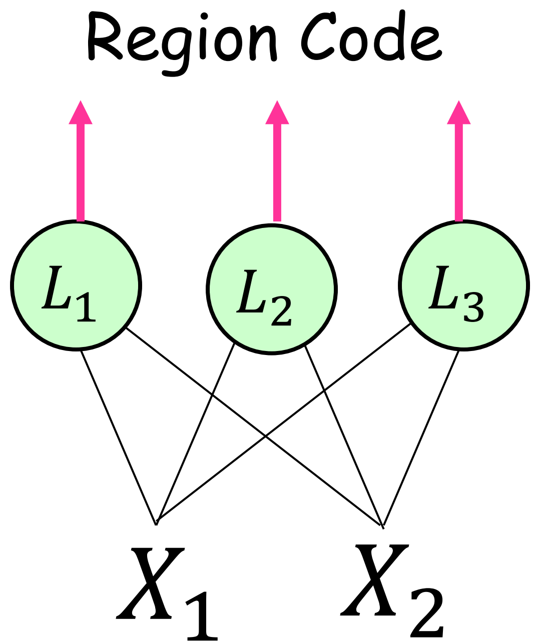
Example: Not Linearly Separable



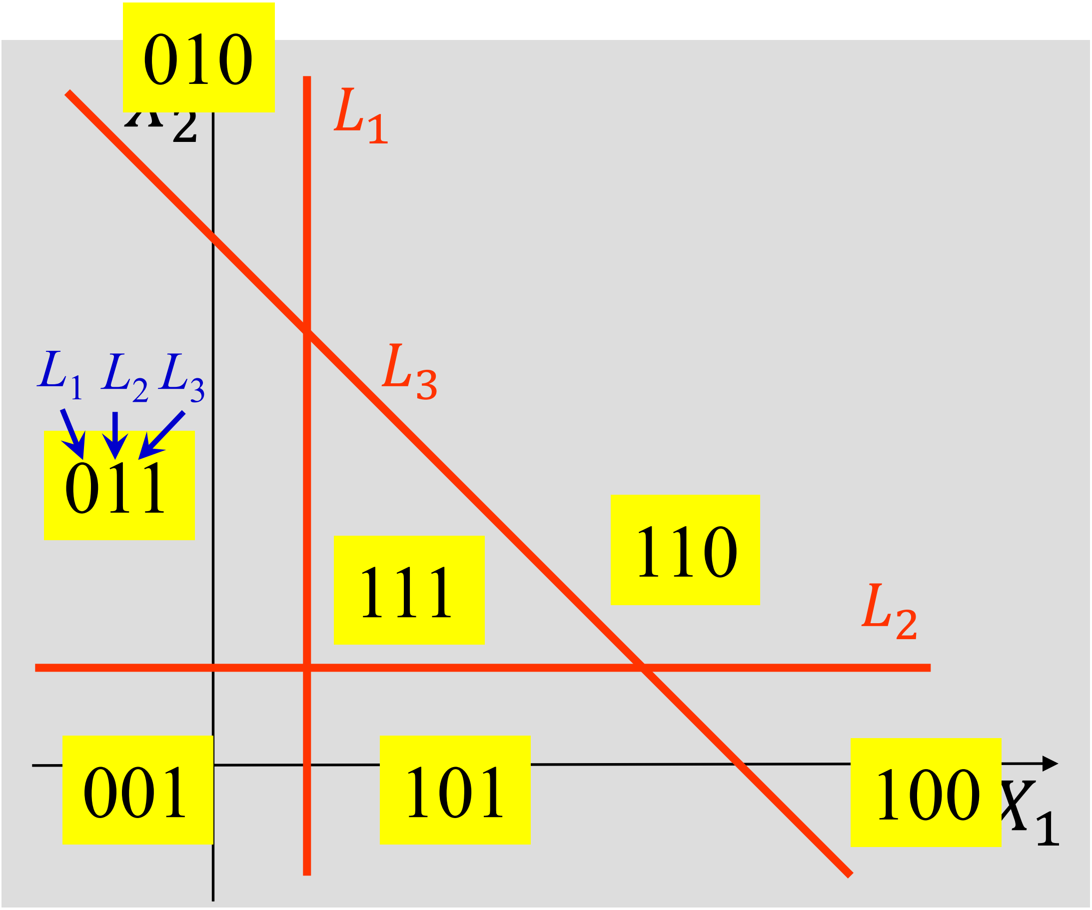


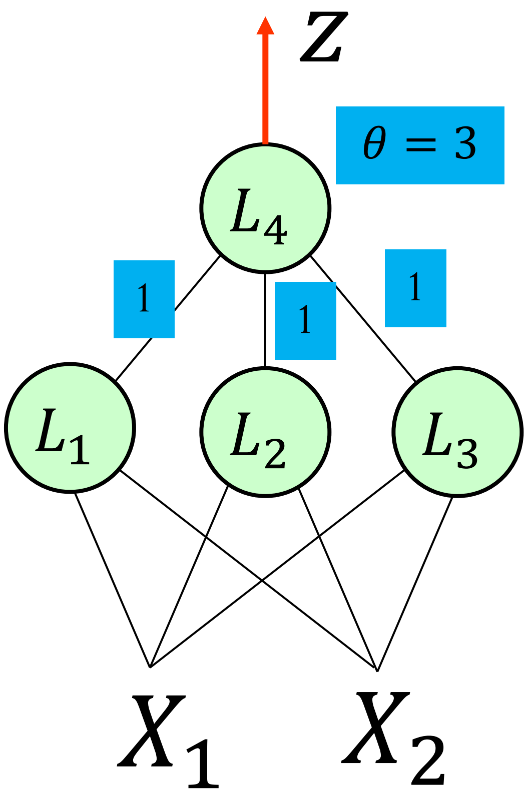
Example: Not Linearly Separable



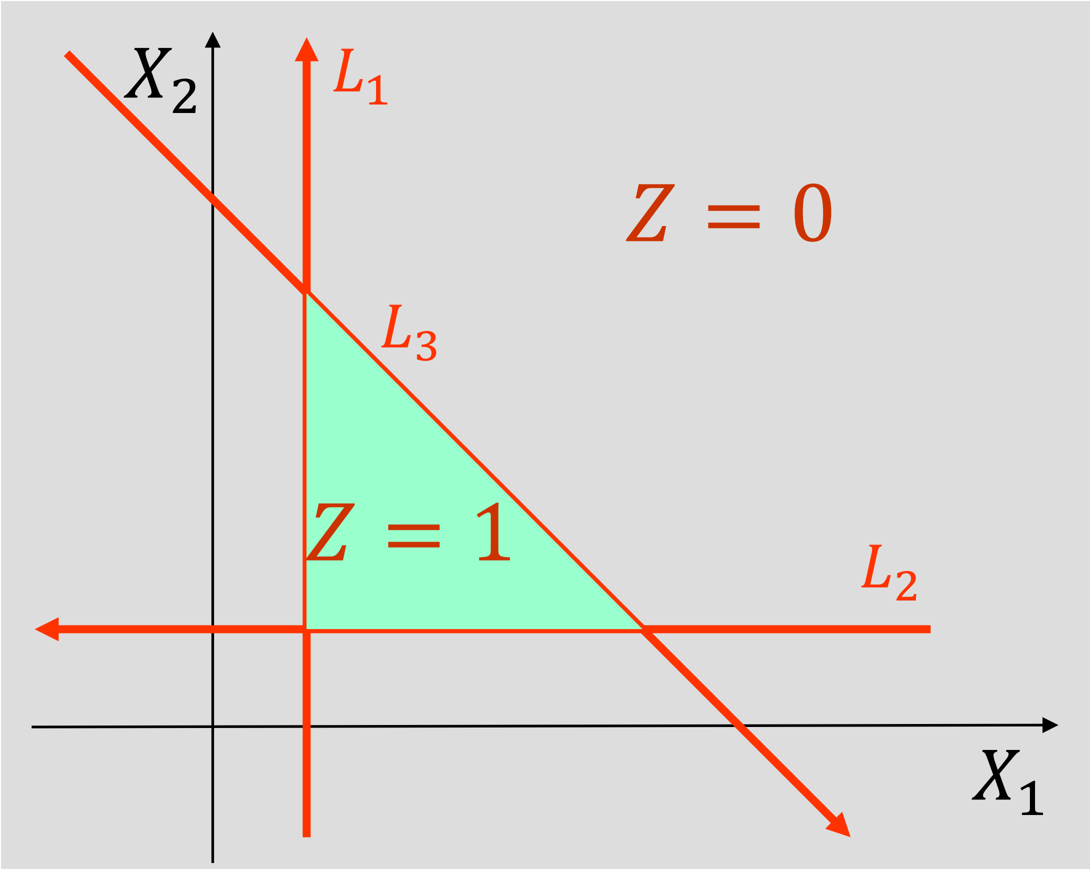


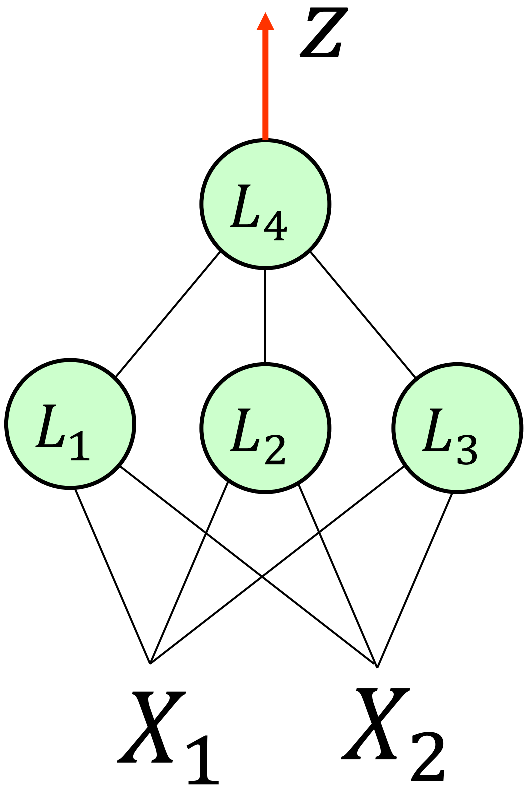
Example: Not Linearly Separable



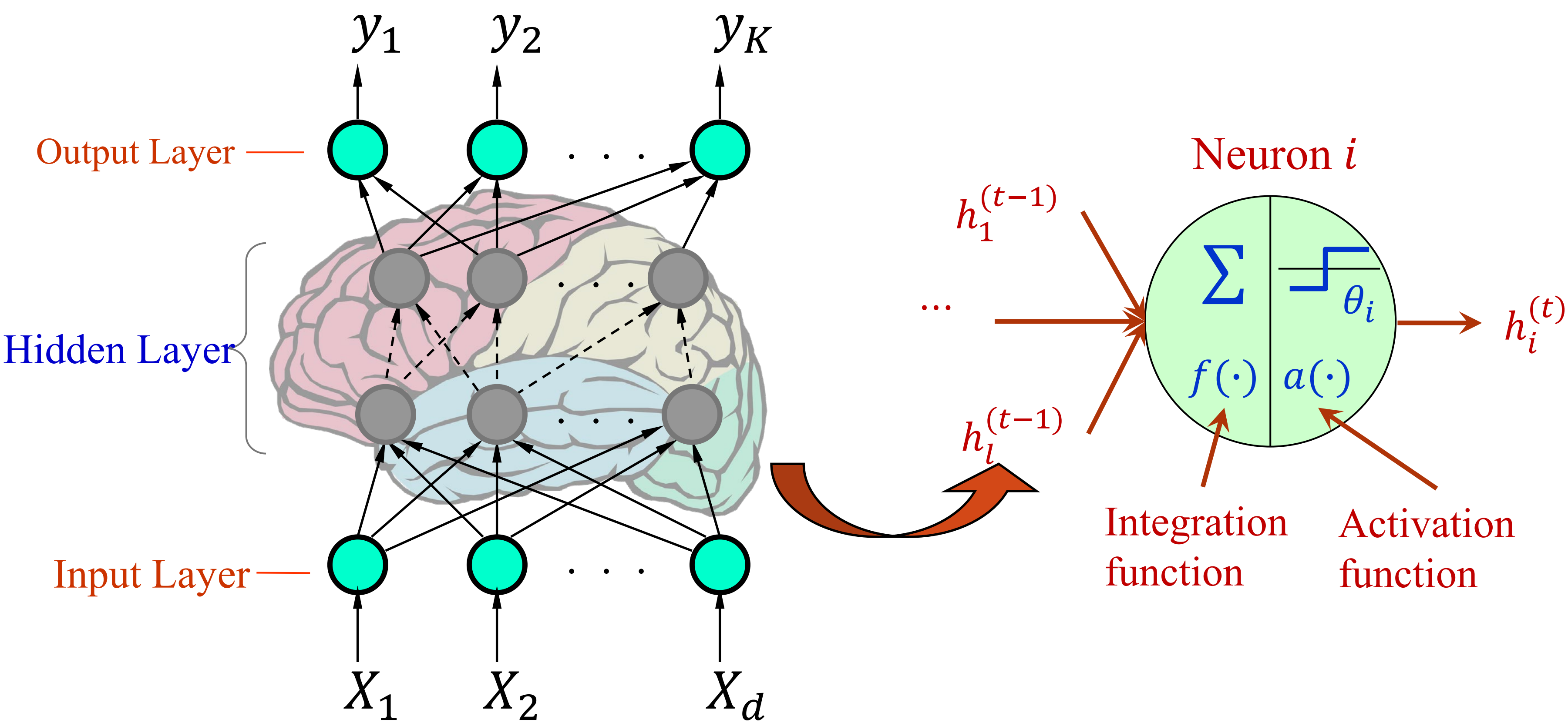


Example: Not Linearly Separable …





General Structure: Multilayer ANN



Integration Functions

- Weighted sum:

$$\sum_{i=1}^{d} w_i X_i - \theta$$

- Quadratic function

$$\sum_{i=1}^{d} w_i X_i^2 - \theta$$

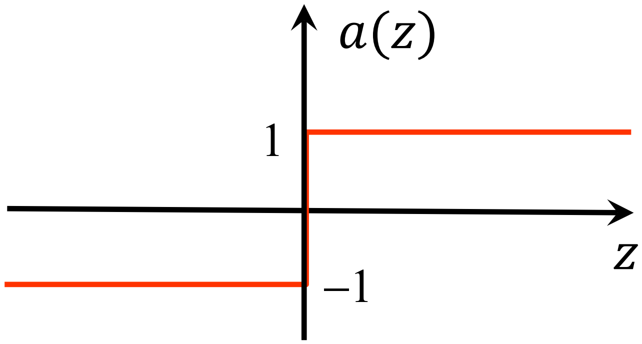
- Spherical function

$$\sum_{i=1}^{d} (X_i - w_i)^2 - \theta$$

Activation Functions

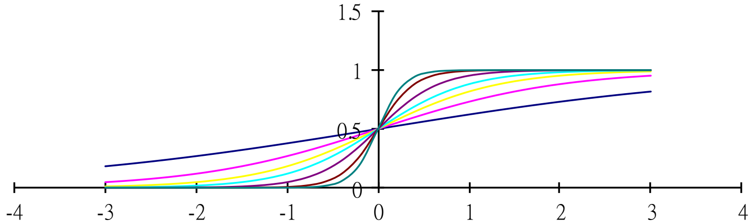
- Sign function (Threshold function)

$$a(z) = \mathrm{sign}(z) = \begin{cases} 1 & z \geq 0 \\ -1 & z < 0 \end{cases}$$



- Unipolar sigmoid function:

$$a(z) = \frac{1}{1 + e^{-\lambda z}}$$



When $\lambda = 1$, it is called sigmoid function

Update Weights for Multi-layer NNs

- Initialize the weights in each layer $\left(\mathbf{w}^{(1)}, \ldots, \mathbf{w}^{(k)}, \ldots, \mathbf{w}^{(m)}\right)$
- Adjust the weights such that the output of ANN is consistent with class labels of training examples
  - Loss function for each training instance:

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

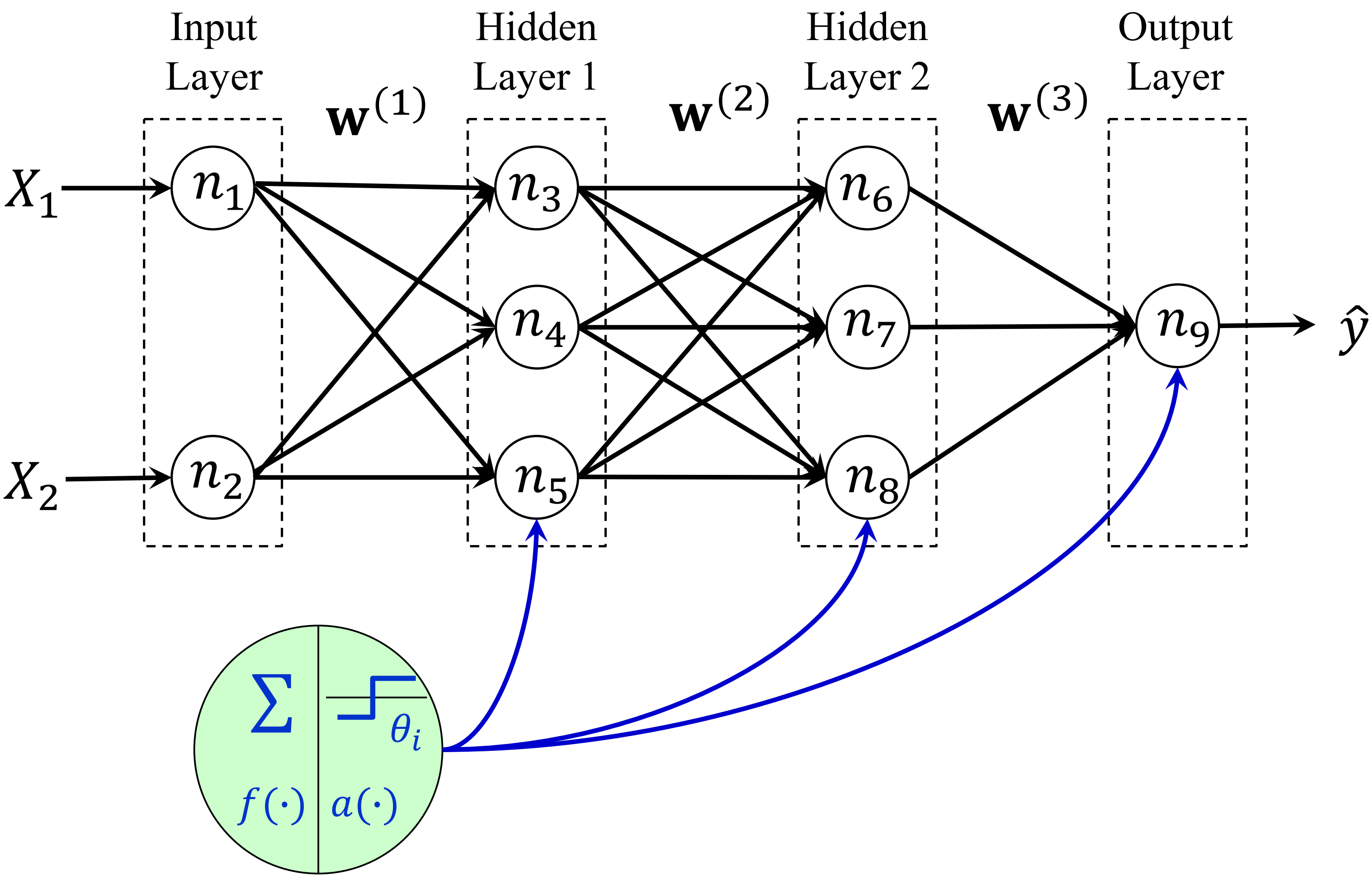
  - For each layer k, update the weights, $\mathbf{w}^{(k)}$, by gradient descent at each iteration t:

$$\mathbf{w}_{t+1}^{(k)} = \mathbf{w}_t^{(k)} - \lambda \frac{\partial \mathcal{L}}{\partial \mathbf{w}^{(k)}}$$

  - Computing the gradient w.r.t. weights in each layer is computationally expensive!

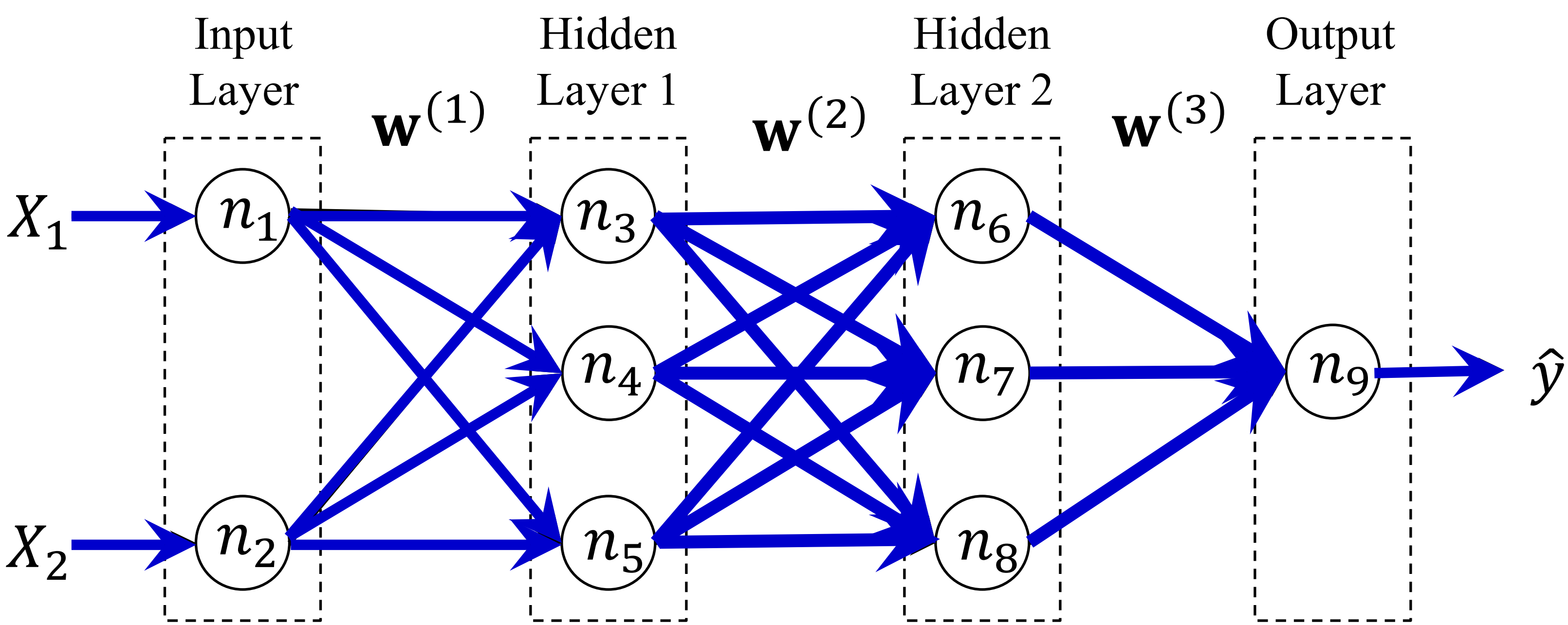
The Backpropagation Algorithm

A Multi-layer Feed-forward NN



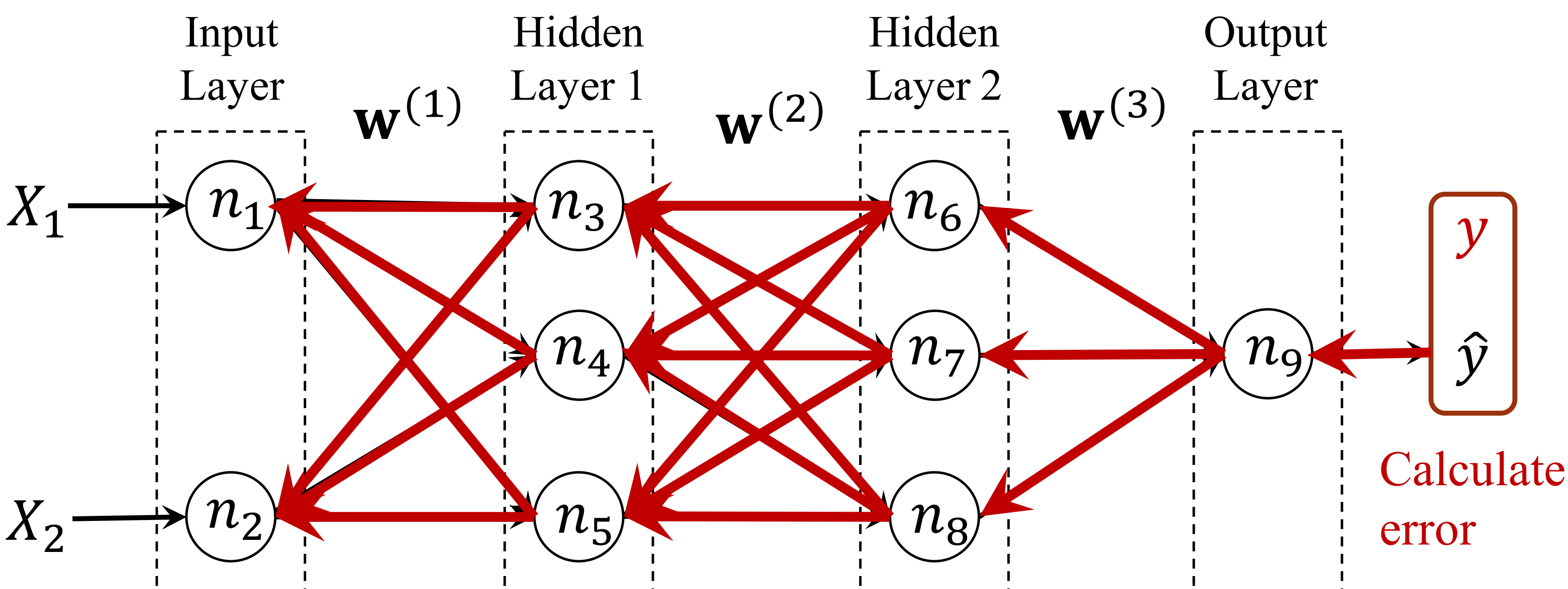
Backpropagation: Basic Idea

- Initialize the weights ($\mathbf{w}^{(1)}, \ldots, \mathbf{w}^{(3)}$)
- **Forward pass:** each training examples$(\boldsymbol{x}_i, y_i)$ is used to compute outputs of each hidden layer and generate the final output $\hat{y}_i$ based on the ANN

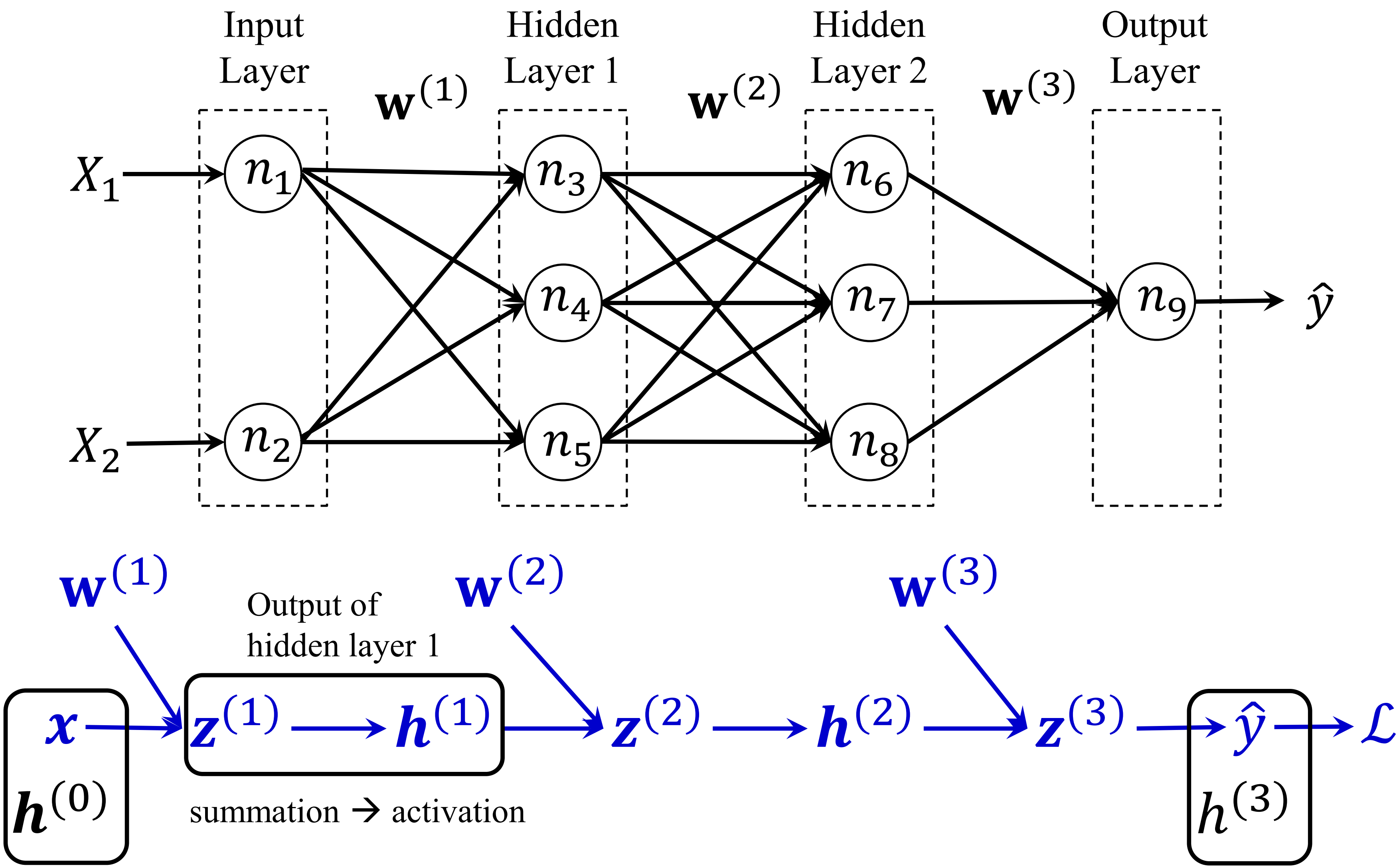


Backpropagation: Basic Idea (cont.)

- **Backpropagation:** Starting with the output layer, to propagate error back to the previous layer in order to update the weights between the two layers, until the earliest hidden layer is reached



The Computational Graph



Backpropagation (BP)

- Gradient of $\mathcal{L}$ w.r.t. $w^{(3)}$: $\dfrac{\partial \mathcal{L}}{\partial w^{(3)}} = \boxed{\dfrac{\partial \mathcal{L}}{\partial \hat{y}} \dfrac{\partial \hat{y}}{\partial z^{(3)}}} \dfrac{\partial z^{(3)}}{\partial w^{(3)}}$
- Gradient of $\mathcal{L}$ w.r.t. $w^{(2)}$:

$$\frac{\partial \mathcal{L}}{\partial w^{(2)}} = \boxed{\boxed{\frac{\partial \mathcal{L}}{\partial \hat{y}} \frac{\partial \hat{y}}{\partial z^{(3)}}} \frac{\partial z^{(3)}}{\partial h^{(2)}} \frac{\partial h^{(2)}}{\partial z^{(2)}}} \frac{\partial z^{(2)}}{\partial w^{(2)}}$$

- Gradient of $\mathcal{L}$ w.r.t. $w^{(1)}$:

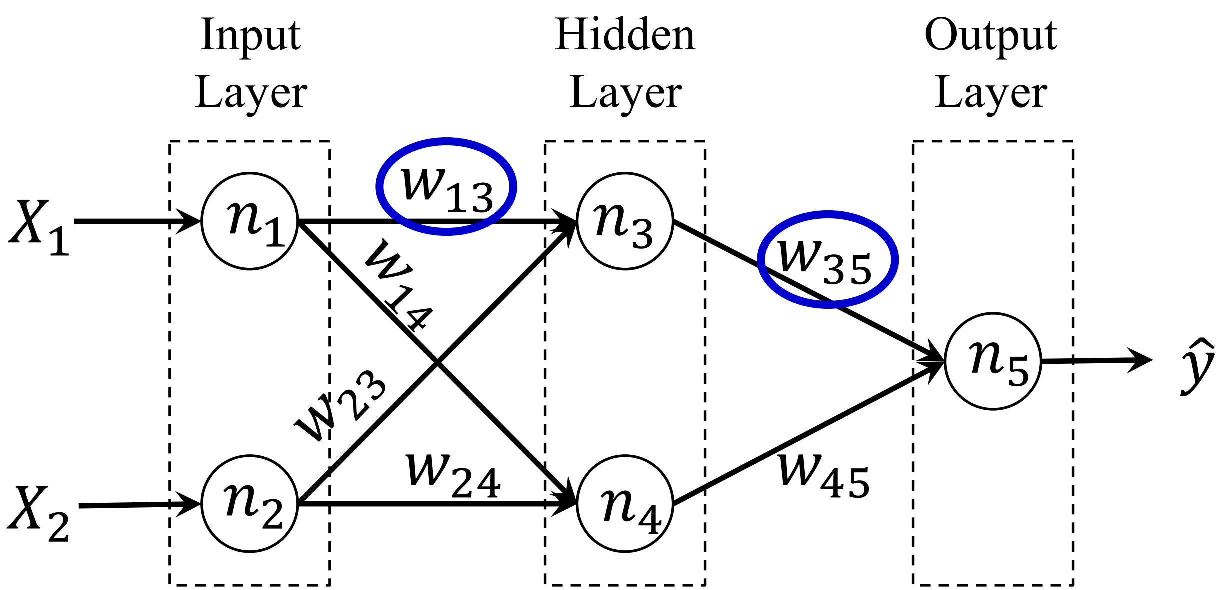
$$\frac{\partial \mathcal{L}}{\partial w^{(1)}} = \boxed{\frac{\partial \mathcal{L}}{\partial \hat{y}} \frac{\partial \hat{y}}{\partial z^{(3)}} \frac{\partial z^{(3)}}{\partial h^{(2)}} \frac{\partial h^{(2)}}{\partial z^{(2)}}} \frac{\partial z^{(2)}}{\partial h^{(1)}} \frac{\partial h^{(1)}}{\partial z^{(1)}} \frac{\partial z^{(1)}}{\partial w^{(1)}}$$

Consider each layer contains a single unit

$$\begin{tikzcd}
& w^{(1)} \arrow[d] & & w^{(2)} \arrow[d] & & w^{(3)} \arrow[d] & & \\
x \arrow[r] & z^{(1)} \arrow[r] & h^{(1)} \arrow[r] & z^{(2)} \arrow[r] & h^{(2)} \arrow[r] & z^{(3)} \arrow[r] & \hat{y} \arrow[r] & \mathcal{L}
\end{tikzcd}$$

An Example

- Consider an ANN of 1 hidden layer as follows. Suppose the sign function and the weighted sum function are used for both hidden and output nodes



$$\begin{tikzcd} & w_{13} \arrow[dr] & & & w_{35} \arrow[dr] & & & \\ X_1 \arrow[rr] & & z_3 \arrow[r] & h_3 \arrow[rr] & & z_5 \arrow[r] & \hat{y} \arrow[r] & \mathcal{L} \end{tikzcd}$$

$$w'_{35} = w_{35} + \lambda e_i h_3$$

$$w'_{35} = w_{35} - \lambda \frac{\partial \mathcal{L}}{\partial w_{35}} = w_{35} - \lambda \boxed{\frac{\partial \mathcal{L}}{\partial \hat{y}}}\frac{\partial \hat{y}}{\partial z_5}\boxed{\frac{\partial z_5}{\partial w_{35}}}$$

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

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

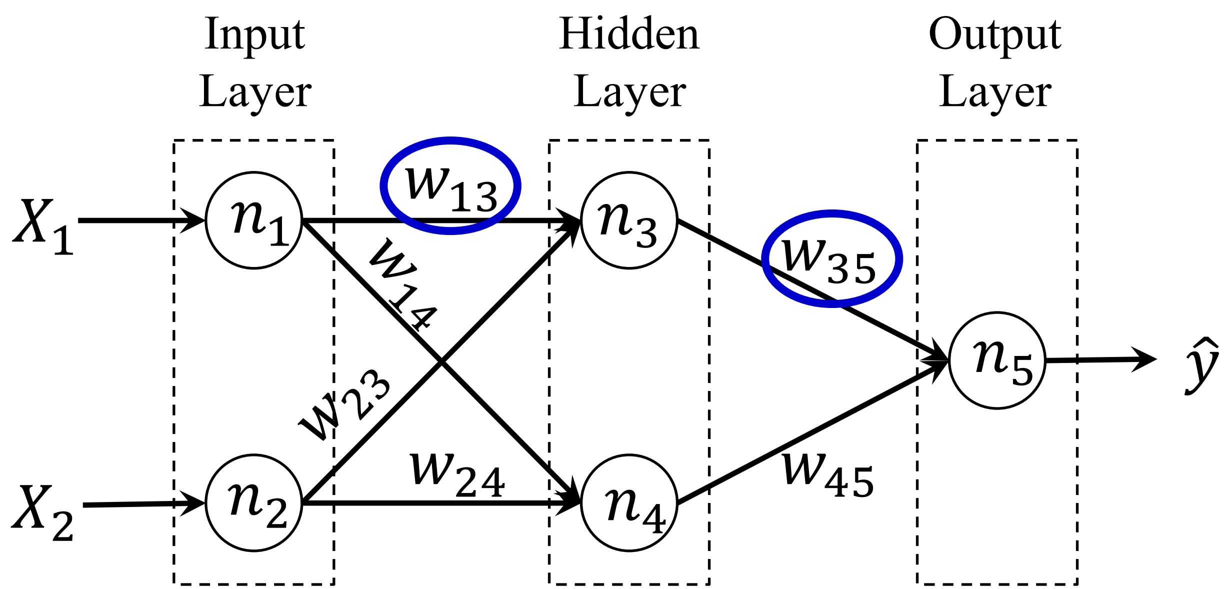
$\cancel{\hat{y}=\mathrm{sgn}(z_5)}$

$\hat{y}=z_5$

$$\frac{\partial \hat{y}}{\partial z_5} = 1$$

$$z_5 = w_{35}h_3 + w_{45}h_4$$

h_3



$$X_1 \xrightarrow{} z_3 \xrightarrow{} h_3 \xrightarrow{} z_5 \xrightarrow{} \hat{y} \xrightarrow{} \mathcal{L}$$

$w_{13} \to z_3$, $w_{35} \to z_5$

$$w'_{13} = w_{13} + \lambda e_i w_{35} X_1$$

$$w'_{13} = w_{13} - \lambda \frac{\partial \mathcal{L}}{\partial w_{13}} = w_{13} - \lambda \boxed{\frac{\partial \mathcal{L}}{\partial \hat{y}} \frac{\partial \hat{y}}{\partial z_5}} \boxed{\frac{\partial z_5}{\partial h_3}} \frac{\partial h_3}{\partial z_3} \boxed{\frac{\partial z_3}{\partial w_{13}}}$$

Obtained when updating w_{35}

$-e_i$

$z_5 = w_{35} h_3 + w_{45} h_4$

w_{35}

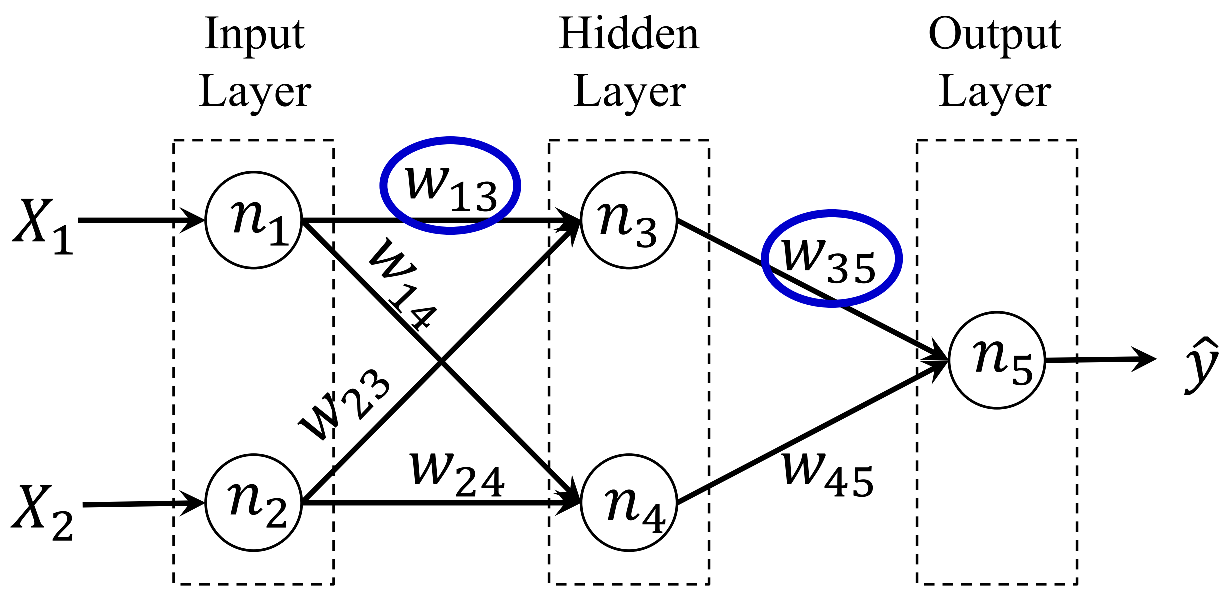
$\cancel{h_3 = \mathrm{sgn}(z_3)}$

$h_3 = z_3$

$$\frac{\partial h_3}{\partial z_3} = 1$$

$z_3 = w_{13} X_1 + w_{23} X_2$

X_1



$$\begin{tikzcd} & w_{13} \arrow[d] & & w_{35} \arrow[d] & & \\ X_1 \arrow[r] & z_3 \arrow[r] & h_3 \arrow[r] & z_5 \arrow[r] & \hat{y} \arrow[r] & \mathcal{L} \end{tikzcd}$$

$$w'_{23} = w_{23} + \lambda e_i w_{35} X_2$$

$$w'_{23} = w_{23} - \lambda \frac{\partial \mathcal{L}}{\partial w_{23}} = w_{23} - \lambda \frac{\partial \mathcal{L}}{\partial \hat{y}} \frac{\partial \hat{y}}{\partial z_5} \frac{\partial z_5}{\partial h_3} \frac{\partial h_3}{\partial z_3} \frac{\partial z_3}{\partial w_{23}}$$

Obtained when updating w_{35} → $\boxed{-e_i}$

$$z_5 = w_{35} h_3 + w_{45} h_4$$

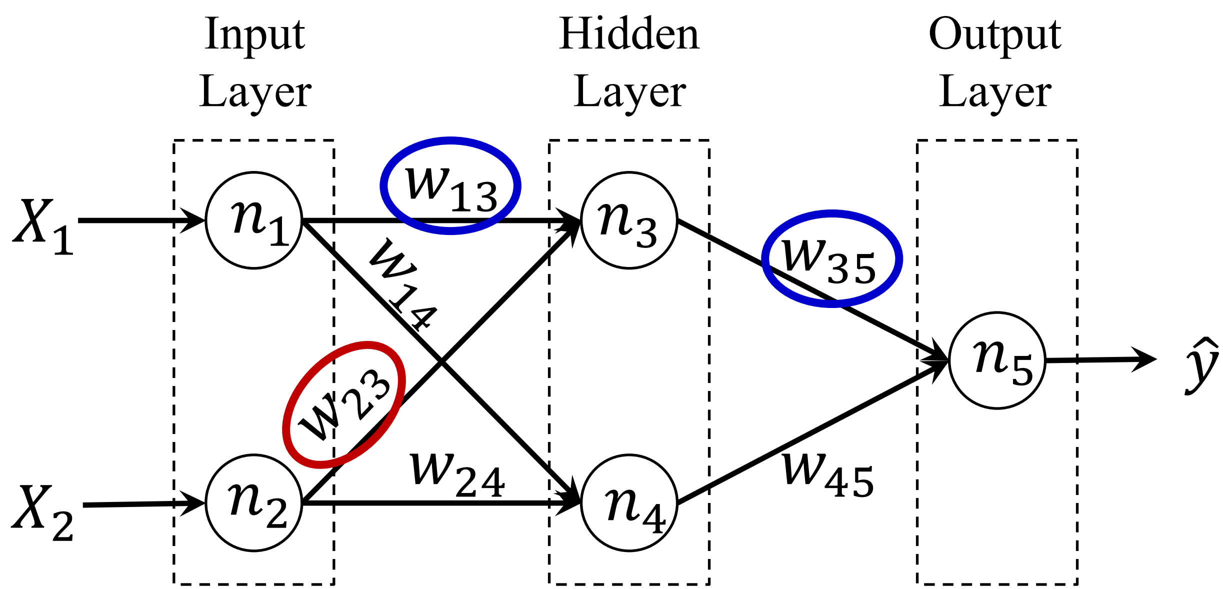
w_{35}

$$\cancel{h_3 = \mathrm{sgn}(z_3)}$$

$$h_3 = z_3 \qquad \frac{\partial h_3}{\partial z_3} = 1$$

$$z_3 = w_{13} X_1 + w_{23} X_2$$

X_2



$$\begin{tikzcd} & w_{23} \arrow[d] & & w_{35} \arrow[d] & & \\ X_2 \arrow[r] & z_3 \arrow[r] & h_3 \arrow[r] & z_5 \arrow[r] & \hat{y} \arrow[r] & \mathcal{L} \end{tikzcd}$$

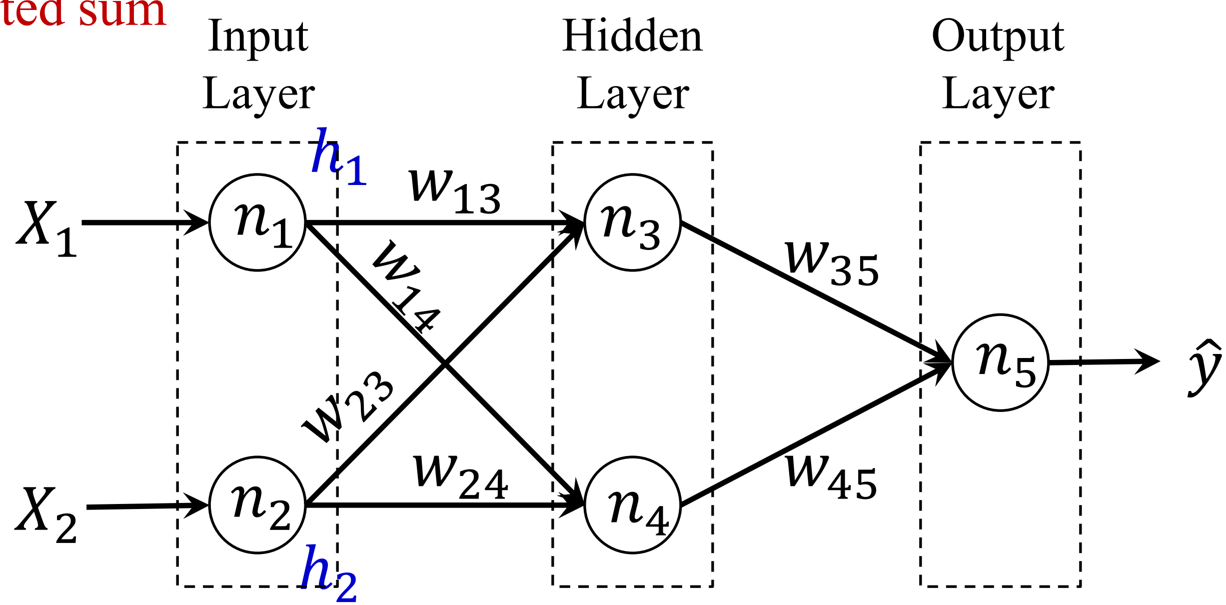
BP Algorithm: Example

Activation function: sign()

Integration function: weighted sum

$\lambda = 0.4,\ \theta = 0$

X_1	X_2	y
0	0	-1
1	0	1
0	1	1
1	1	1



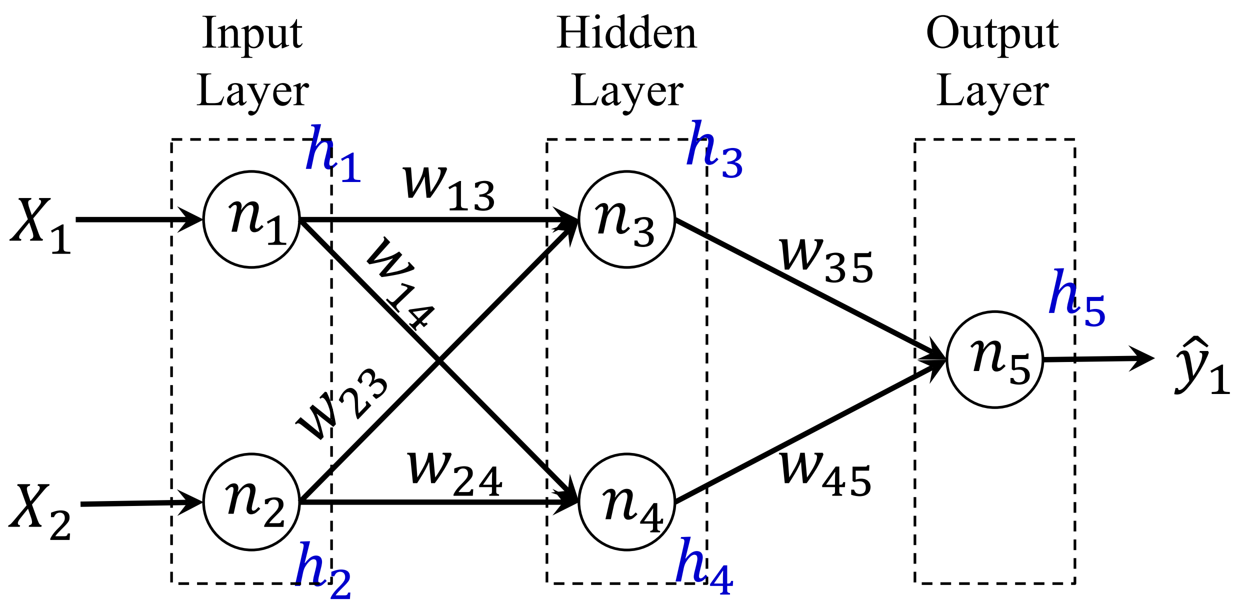
- **Initialization:**

  $(w_{13} = 1, w_{14} = 1, w_{23} = 1, w_{24} = 1, w_{35} = 1, w_{45} = 1)$

  For the 1st example: $h_1 = 0$ and $h_2 = 0$

BP Algorithm: Example (cont.)

X_1	X_2	y
0	0	-1
1	0	1
0	1	1
1	1	1



Forward pass:

$h_3 = \text{sign}(0 \times 1 + 0 \times 1) = 1$ and $h_4 = \text{sign}(0 \times 1 + 0 \times 1) = 1$

Then $\hat{y}_1 = h_5 = \text{sign}(1 \times 1 + 1 \times 1) = 1$

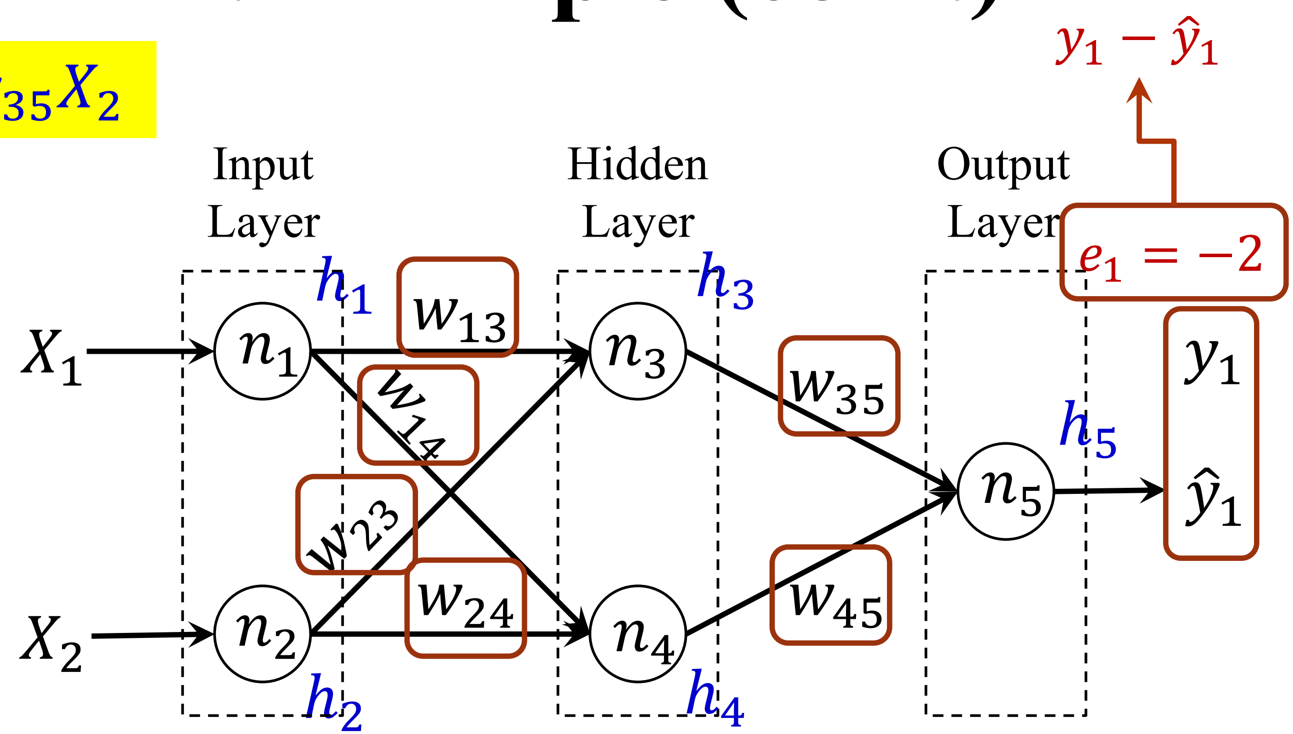
$w'_{13} = w_{13} + \lambda e_1 w_{35} X_1$

$w'_{35} = w_{35} + \lambda e_1 h_3$

BP Algorithm: Example (cont.)

$w'_{23} = w_{23} + \lambda e_i w_{35} X_2$

X_1	X_2	y
0	0	-1
1	0	1
0	1	1
1	1	1



Backpropagation:

$w_{35} = 1 + 0.4 \times (-2) \times 1 = 0.2$

$w_{13} = 1 + 0.4 \times (-2) \times 1 \times 0 = 1$

$w_{23} = 1 + 0.4 \times (-2) \times 1 \times 0 = 1$

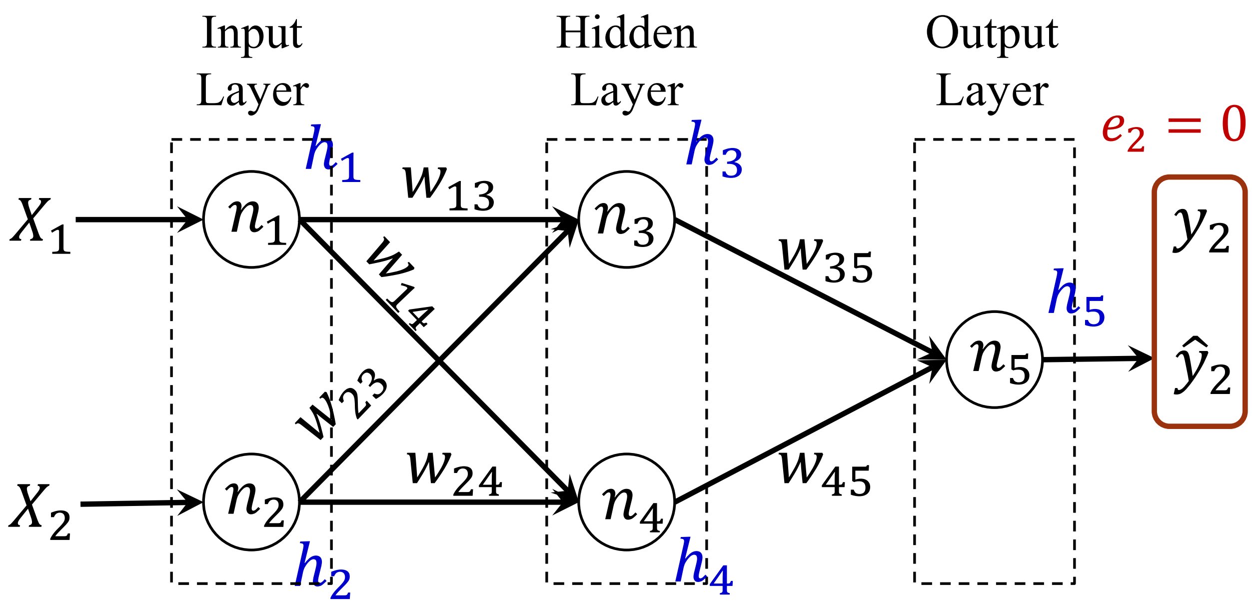
$w_{45} = 1 + 0.4 \times (-2) \times 1 = 0.2$

$w_{14} = 1 + 0.4 \times (-2) \times 1 \times 0 = 1$

$w_{24} = 1 + 0.4 \times (-2) \times 1 \times 0 = 1$

BP Algorithm: Example (cont.)

X_1	X_2	y
0	0	-1
1	0	1
0	1	1
1	1	1



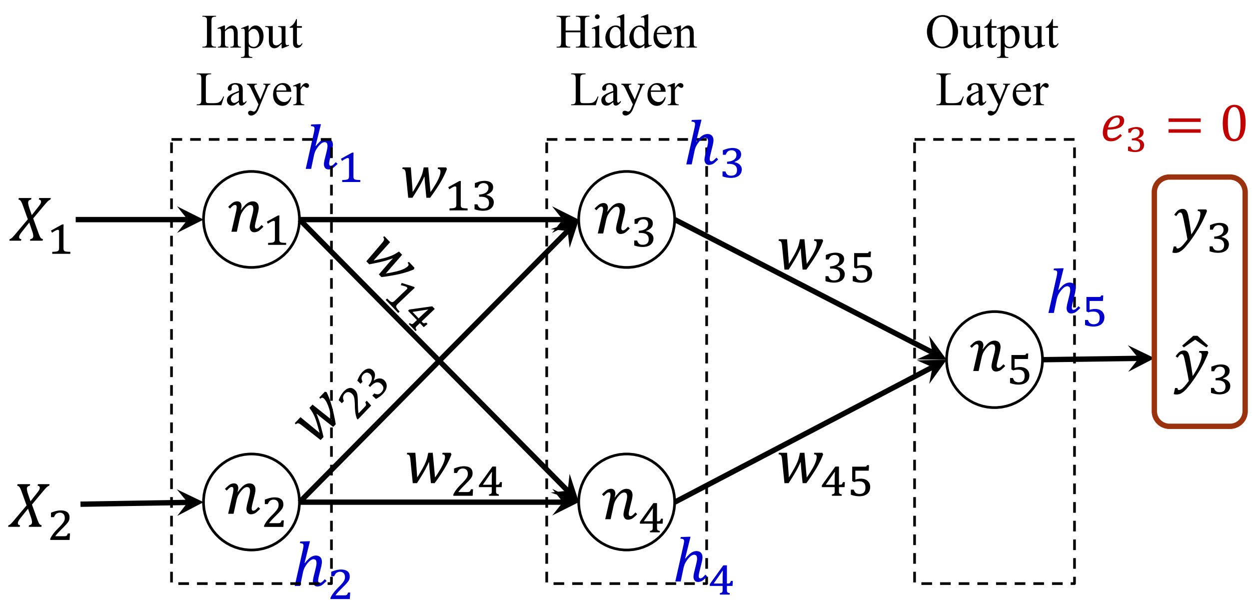
For the 2[nd] example: $h_1 = 1$ and $h_2 = 0$

$h_3 = \text{sign}(1 \times 1 + 0 \times 1) = 1$ and $h_4 = \text{sign}(1 \times 1 + 0 \times 1) = 1$

Then $\hat{y}_2 = h_5 = \text{sign}(1 \times 0.2 + 1 \times 0.2) = 1$

BP Algorithm: Example (cont.)

X_1	X_2	y
0	0	-1
1	0	1
0	1	1
1	1	1



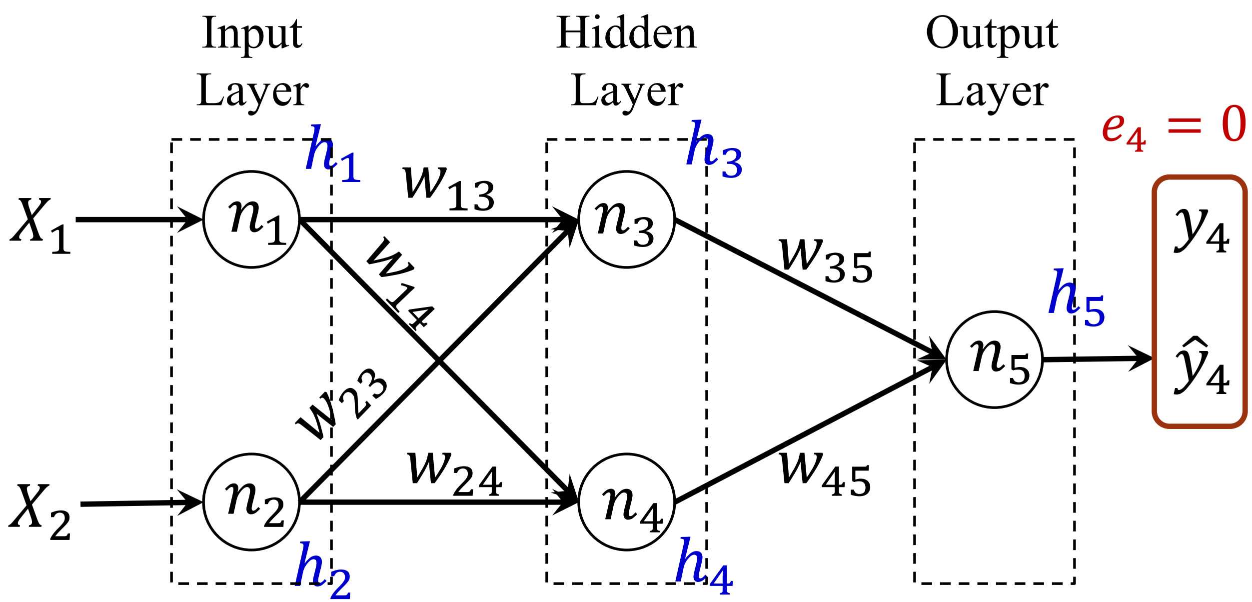
For the 3rd example: $h_1 = 0$ and $h_2 = 1$

$h_3 = \text{sign}(0 \times 1 + 1 \times 1) = 1$ and $h_4 = \text{sign}(0 \times 1 + 1 \times 1) = 1$

Then $\hat{y}_3 = h_5 = \text{sign}(1 \times 0.2 + 1 \times 0.2) = 1$

BP Algorithm: Example (cont.)

X_1	X_2	y
0	0	-1
1	0	1
0	1	1
1	1	1



For the 4th example: $h_1 = 1$ and $h_2 = 1$

$h_3 = \text{sign}(1 \times 1 + 1 \times 1) = 1$ and $h_4 = \text{sign}(1 \times 1 + 1 \times 1) = 1$

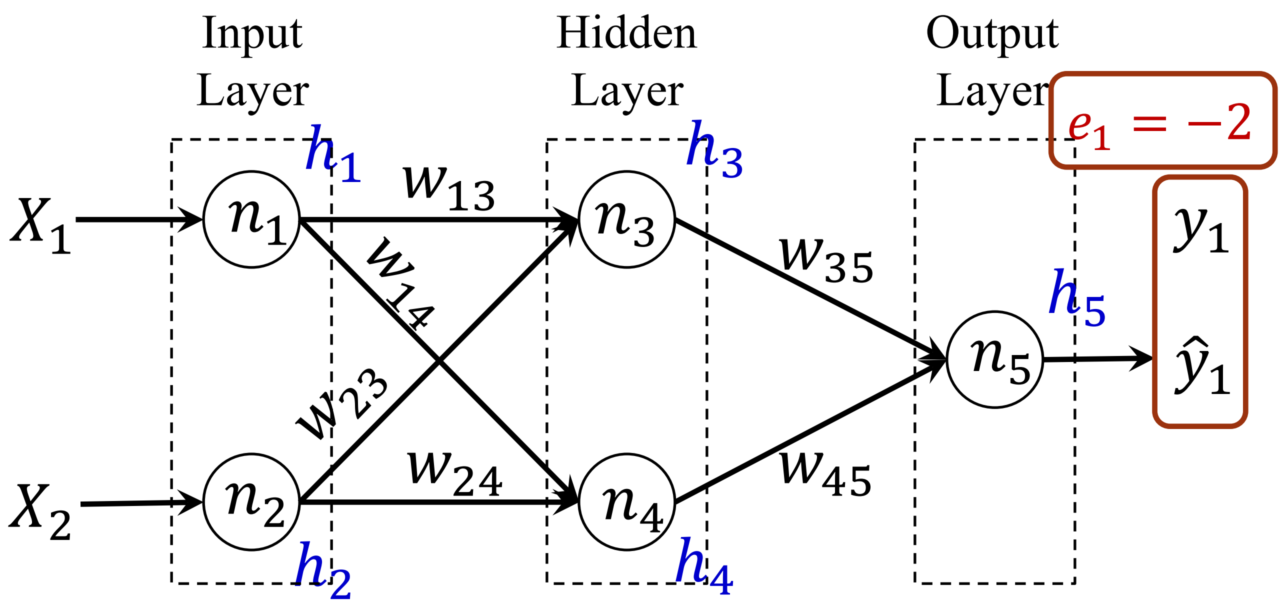
Then $\hat{y}_4 = h_5 = \text{sign}(1 \times 0.2 + 1 \times 0.2) = 1$

End of the 1st Epoch

The 2nd Epoch starts

BP Algorithm: Example (cont.)

X_1	X_2	y
0	0	-1
1	0	1
0	1	1
1	1	1



For the 1st example again: $h_1 = 0$ and $h_2 = 0$

$h_3 = \text{sign}(0 \times 1 + 0 \times 1) = 1$ and $h_4 = \text{sign}(0 \times 1 + 0 \times 1) = 1$

Then $\hat{y}_1 = h_5 = \text{sign}(0.2 \times 1 + 0.2 \times 1) = 1$

Weights need to be further updated via backpropagation

Design Issues for ANN

- The number of nodes in the input layer
  - Assign an input node to each numerical or binary input variable
- The number of nodes in the output layer
  - Binary class problem → single node
  - C-class problem → C output nodes
- How many nodes in the hidden layer(s)?
  - Too many parameters result in networks that are too complex and overfit the data

Design Issues for ANN

- How many nodes in the hidden layer(s)?
  - Too many parameters result in networks that are too complex and overfit the data
- If the network underfits
  - Try to increase the number of hidden units
- If the network overfits
  - Try to decrease the number of hidden units