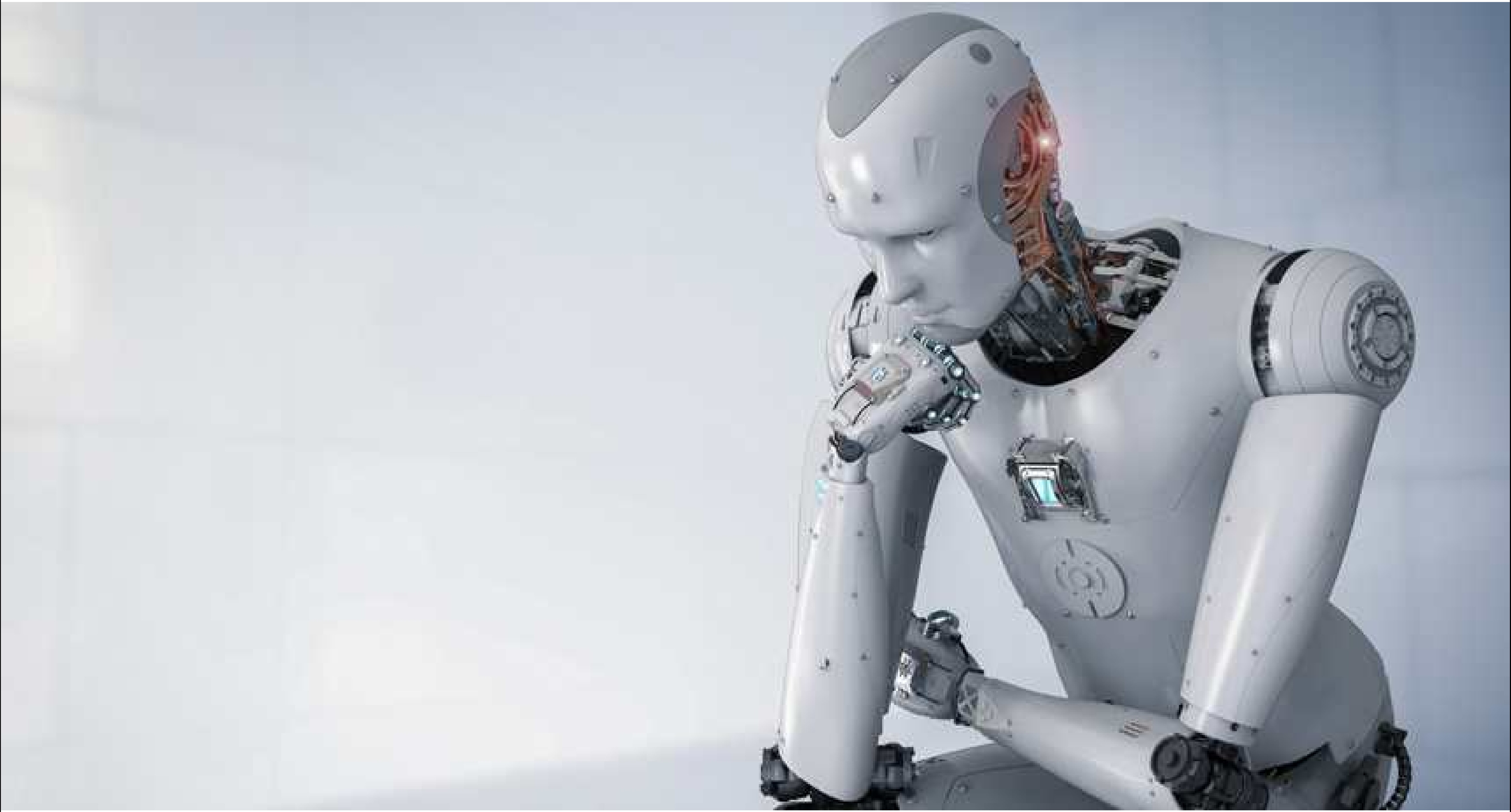
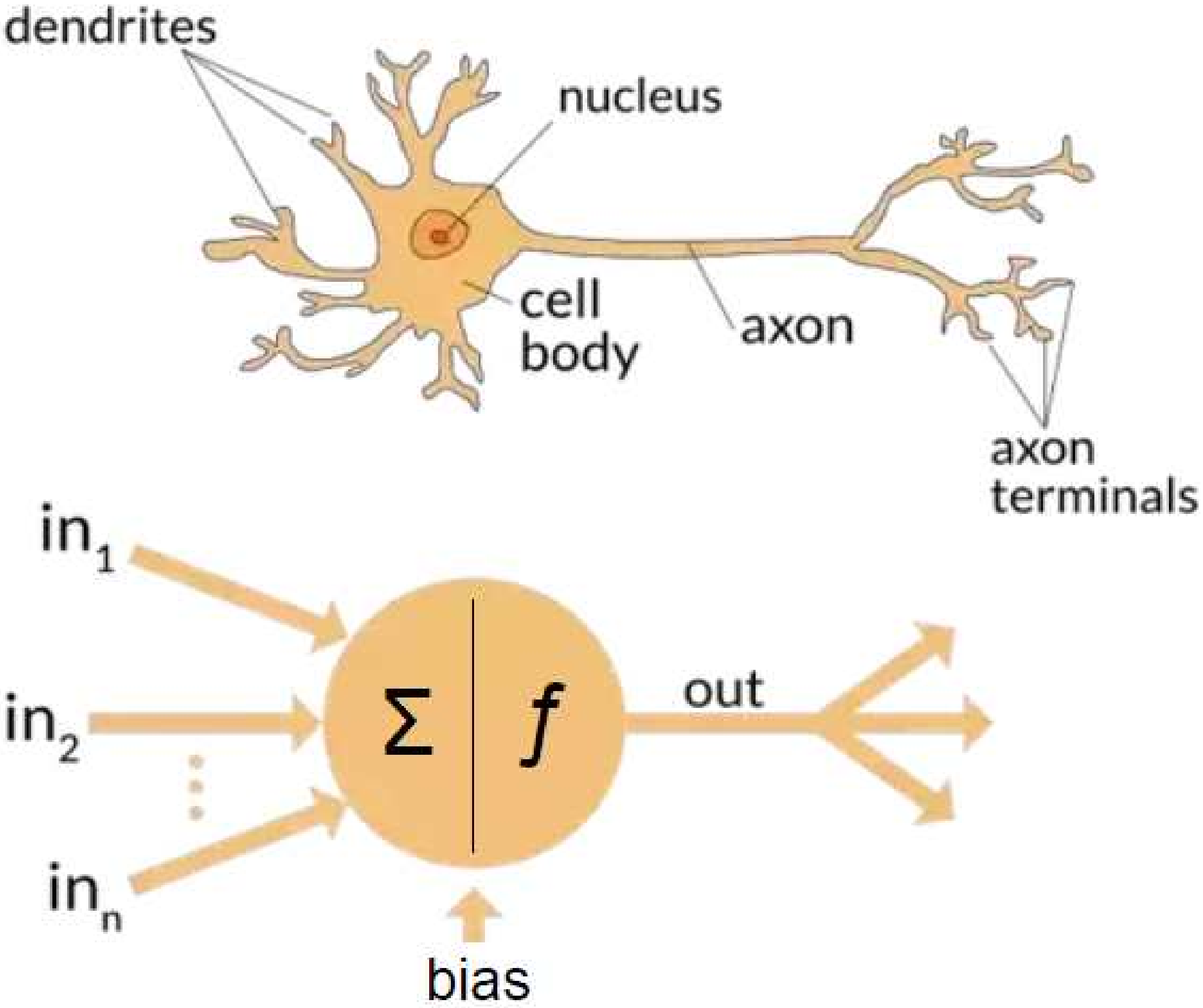
Ⅰ. Deep Learning

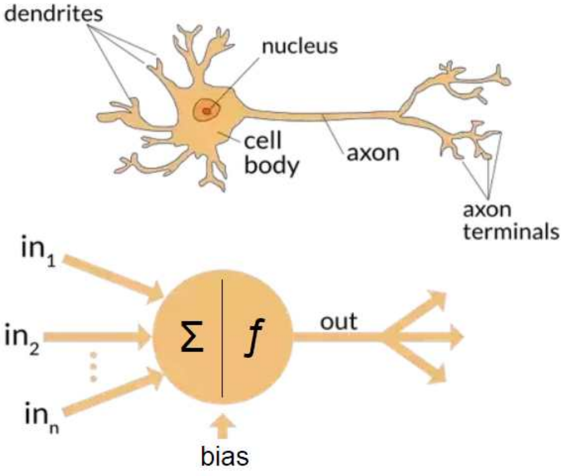
01 Thinking Machines

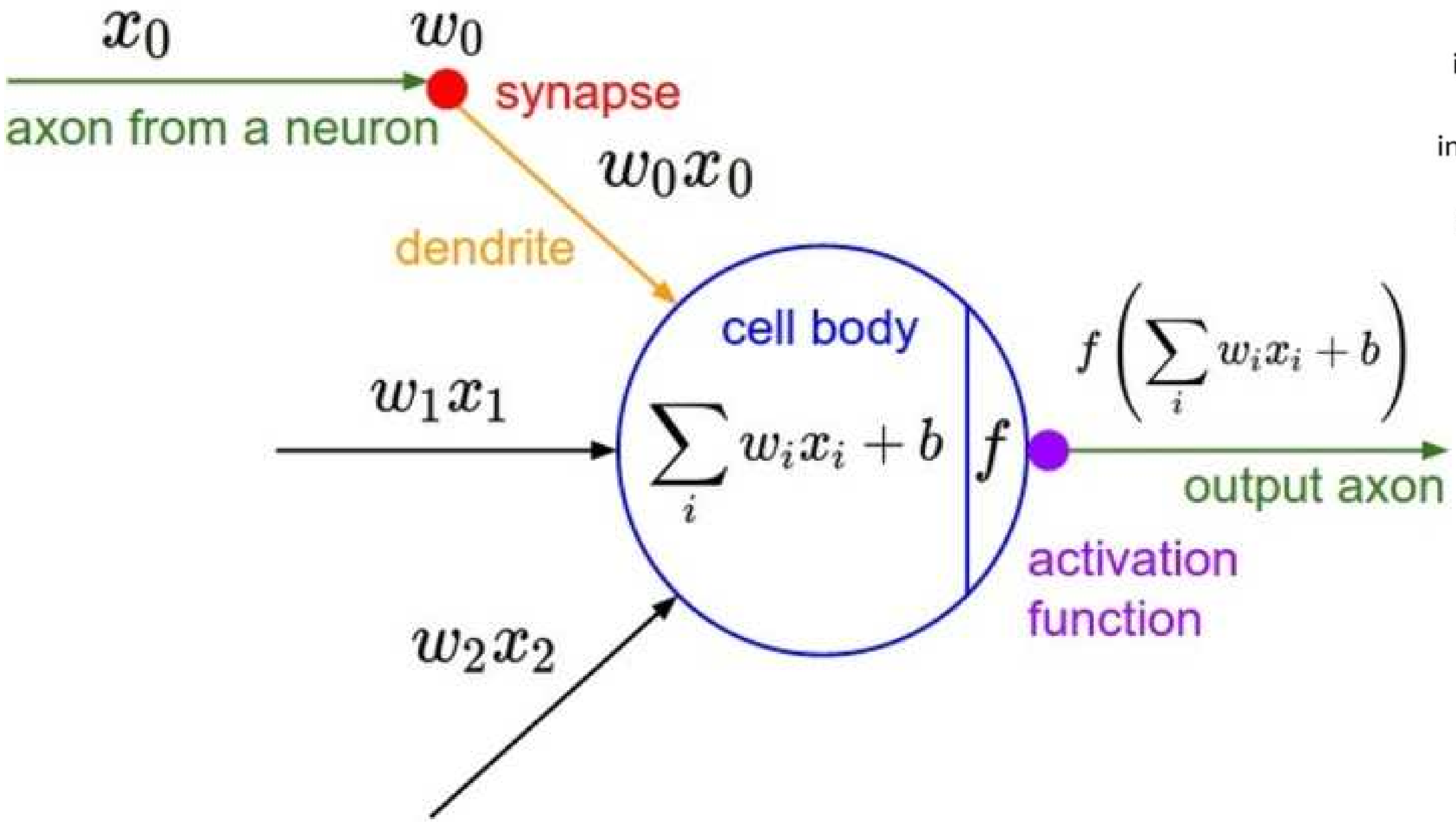


01 Thinking Machines

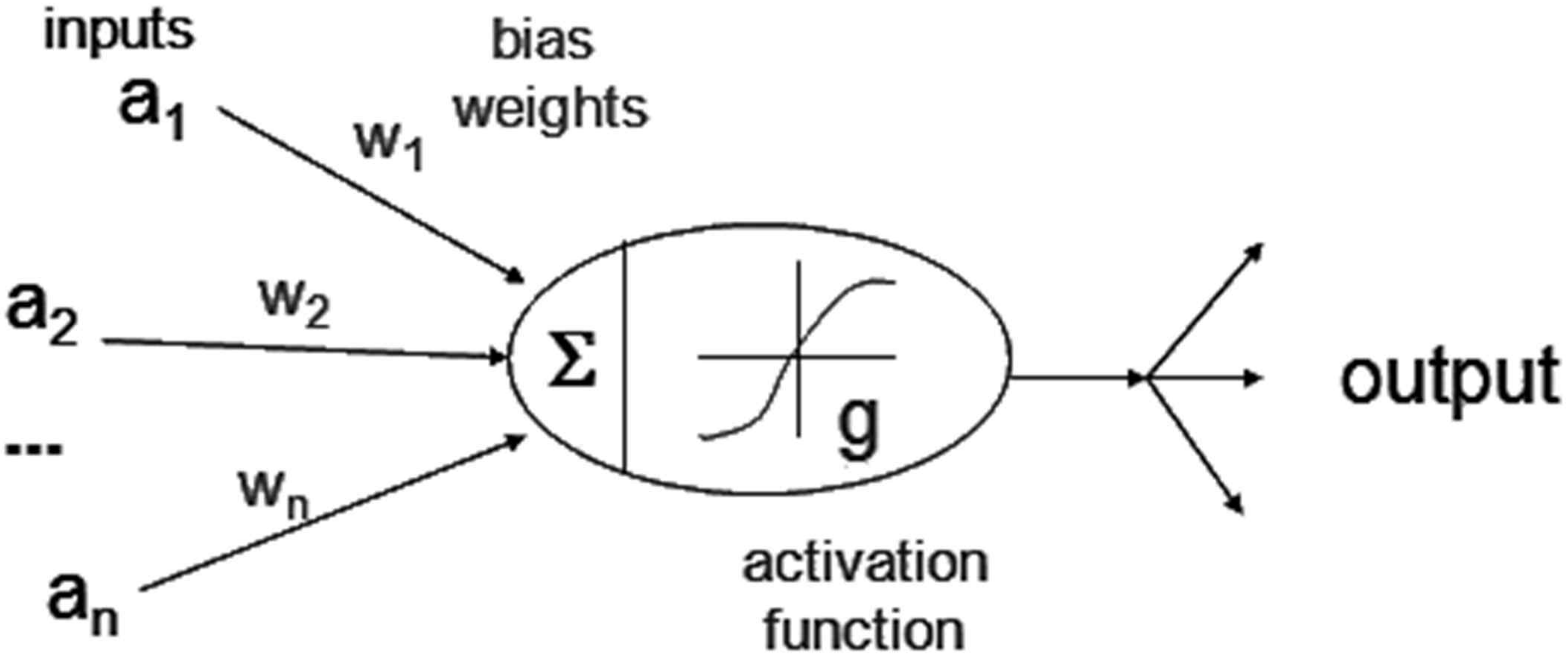


02 Activation Functions

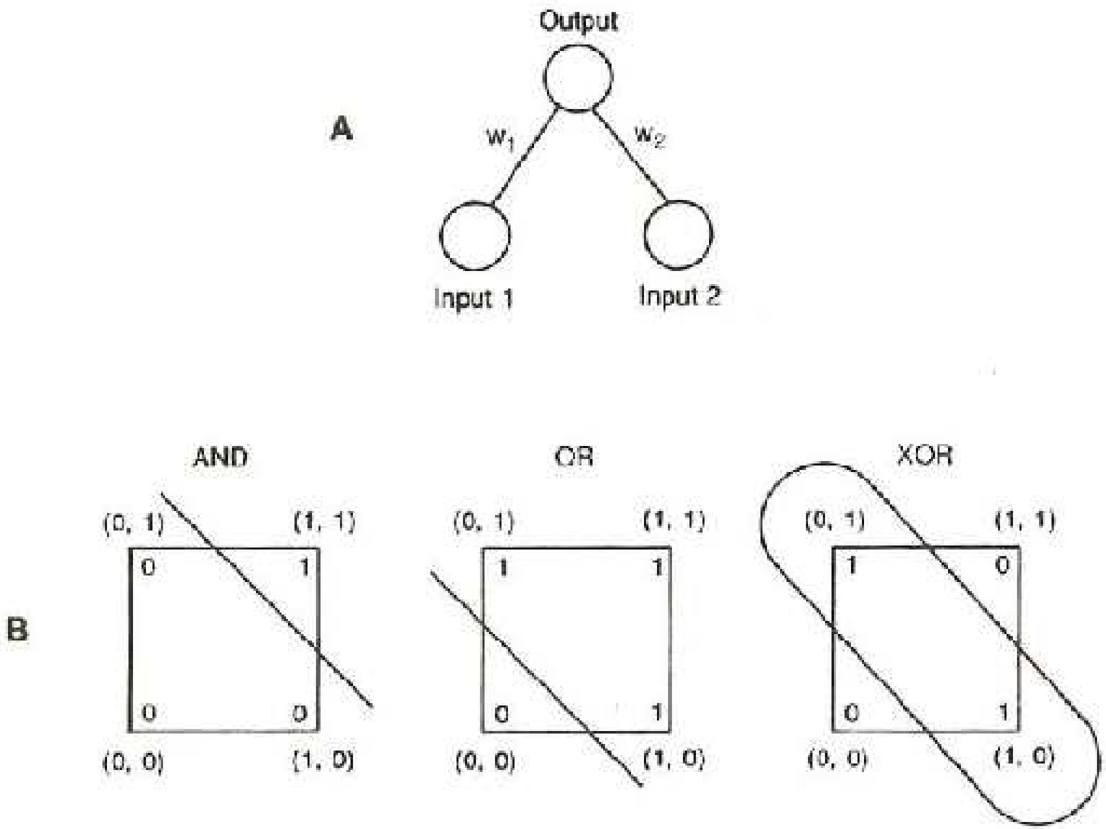




03 Logistic Regression Units

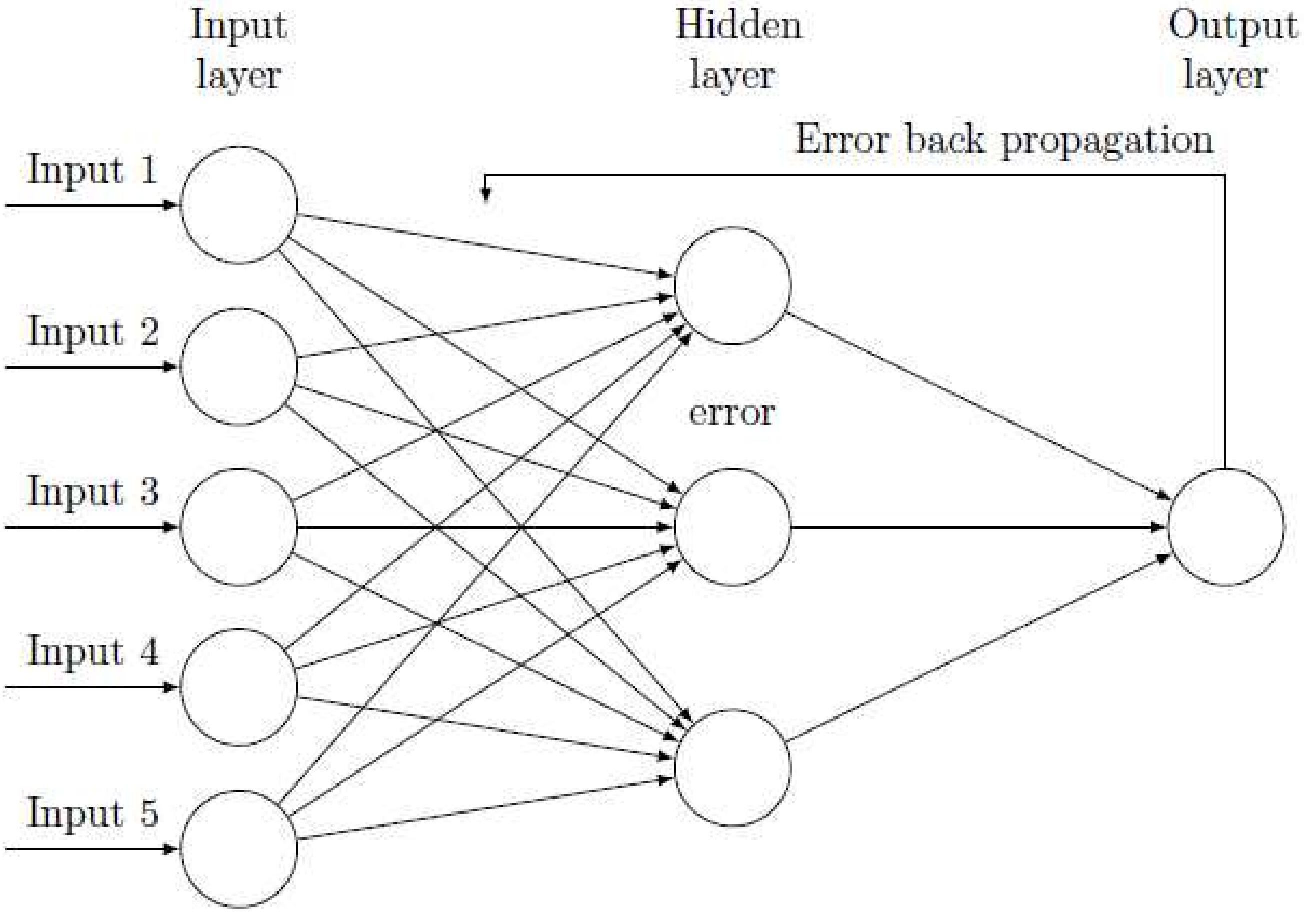


04 XOR Problem

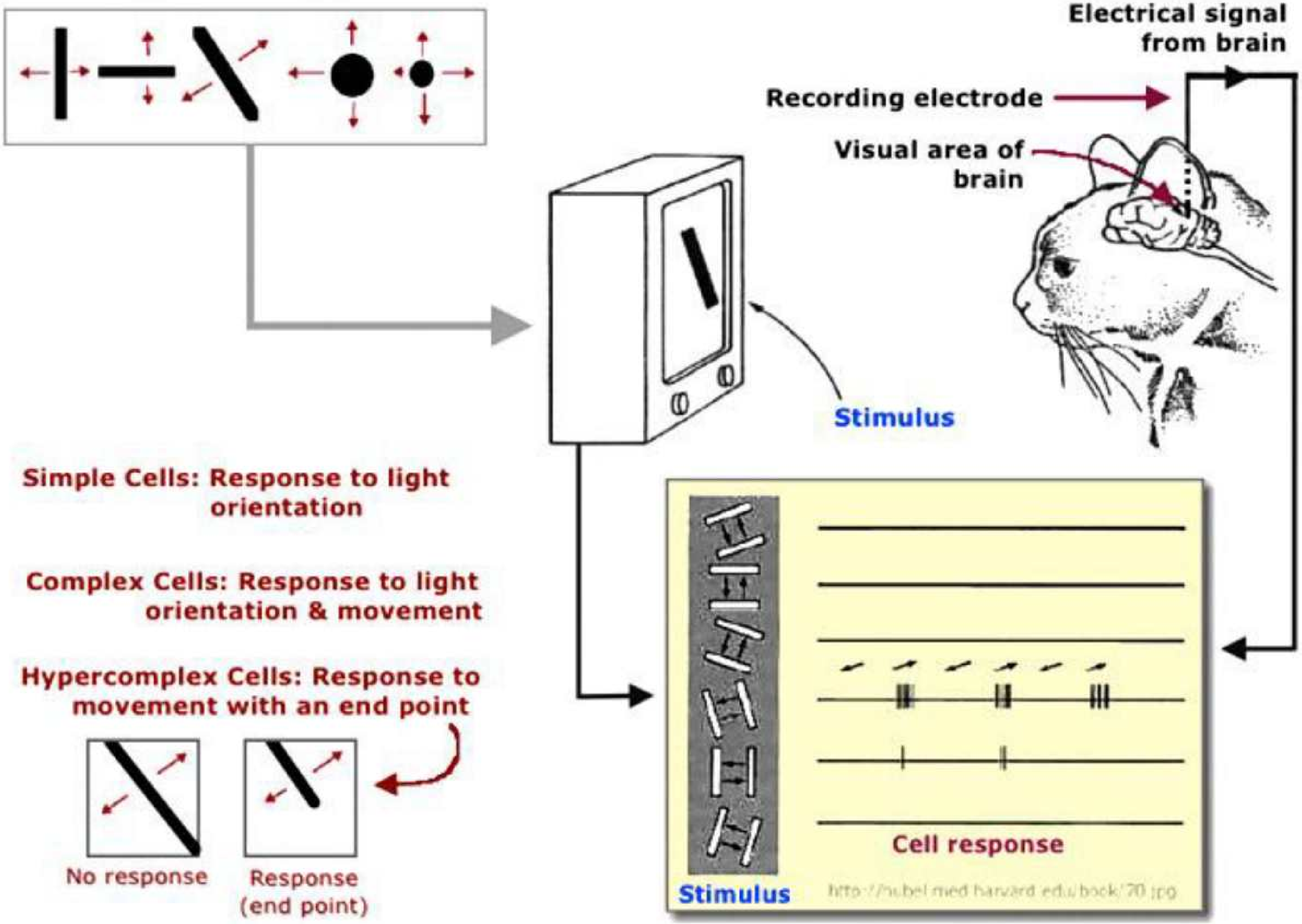


Input Patterns		Output Patterns
00	→	0
01	→	1
10	→	1
11	→	0

05 Backpropagation

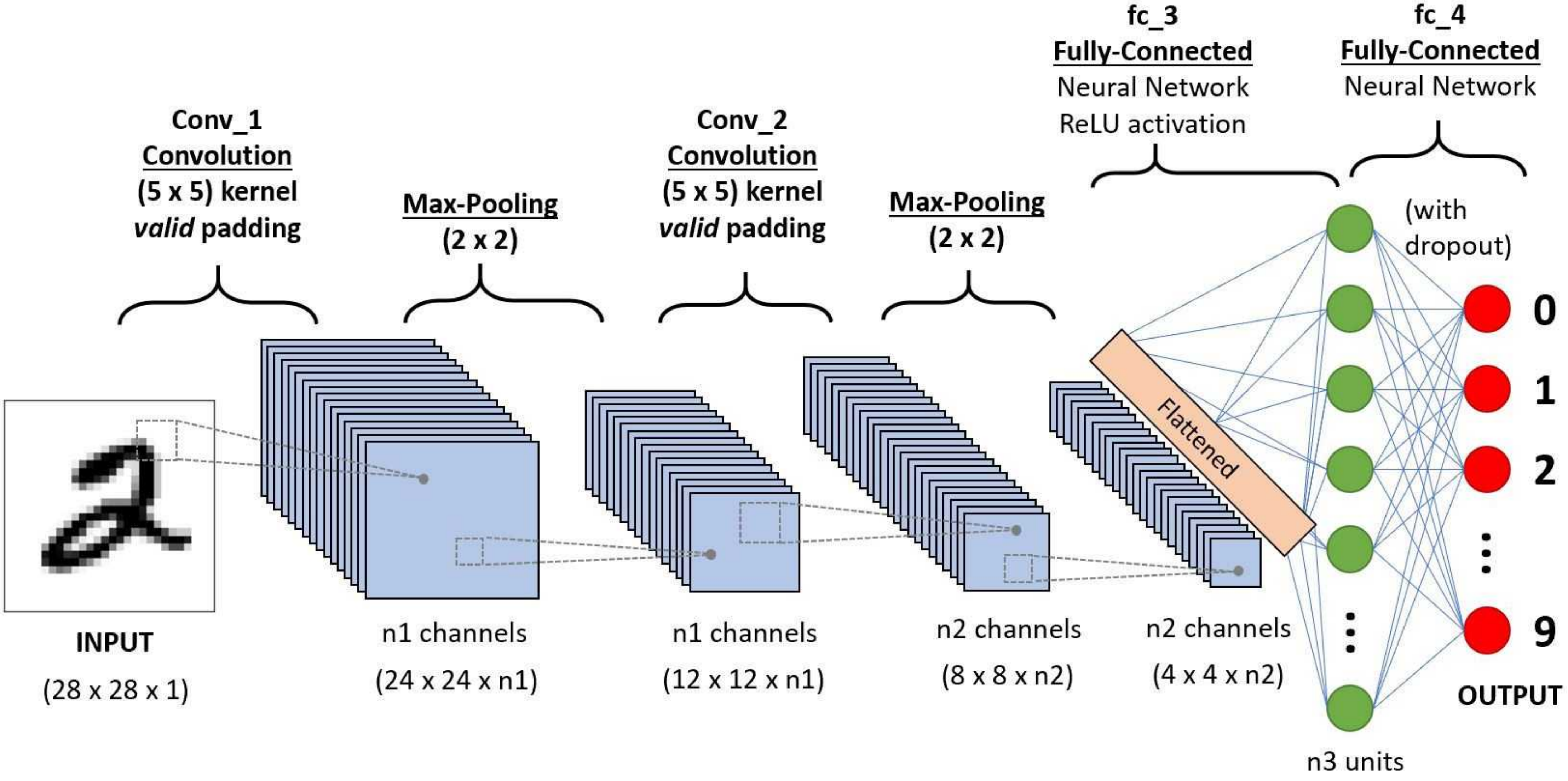


06 Convolutional Neural Networks



http://hubel.med.harvard.edu/book/70.jpg

06 Convolutional Neural Networks



07 Big Problem

- Backpropagation just did not work well for normal neural nets with many layers

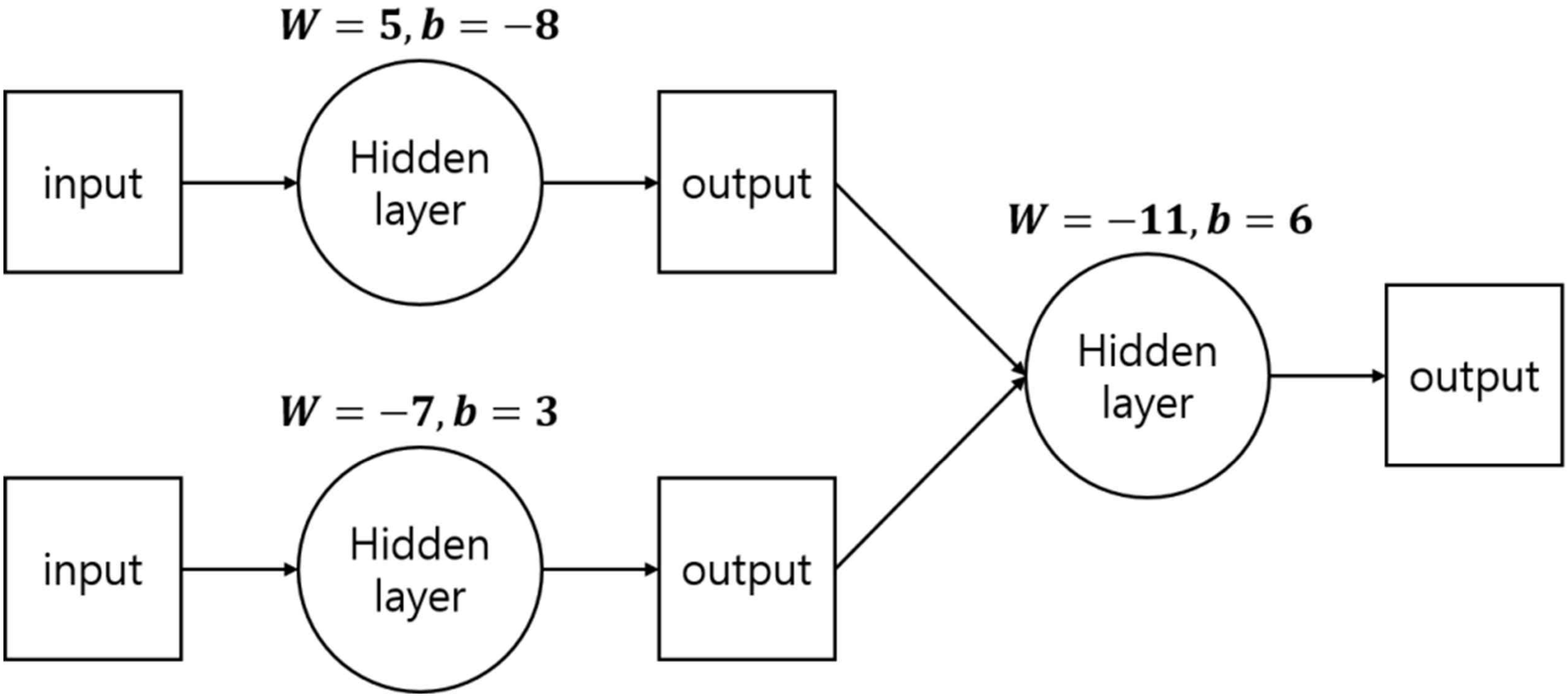
- Other rising machine learning algorithms: SVM, RandomForest, etc.

08 Breakthrough – Deep Learning

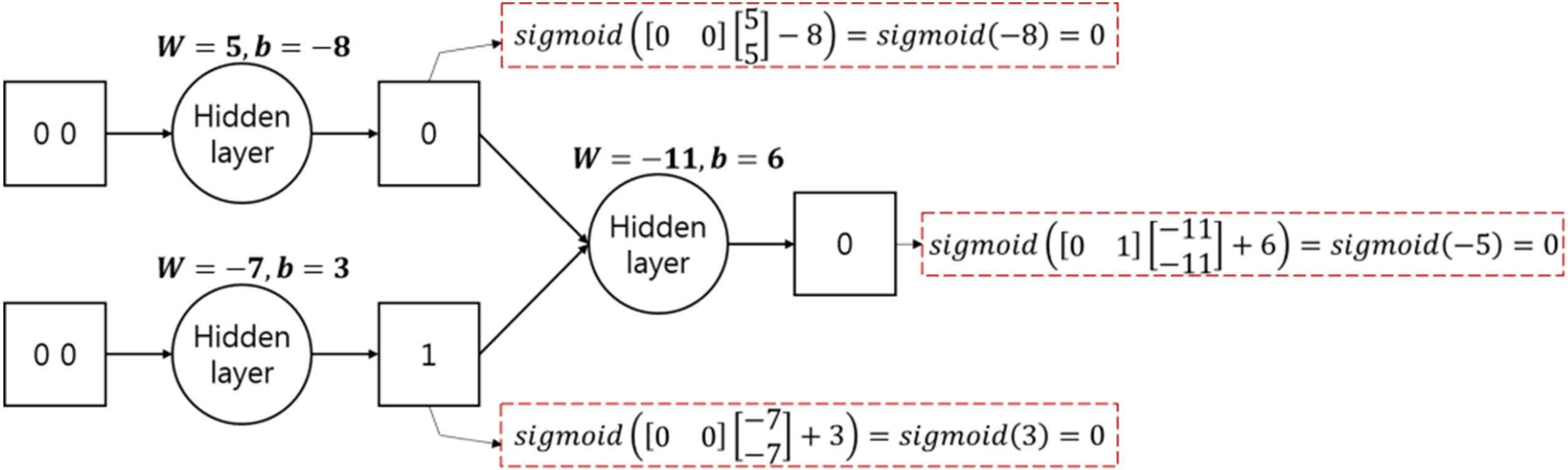
- Neural networks with many layers really could be trained well, if the weights are initialized in a clever way rather than randomly
- Deep machine learning methods are more efficient for difficult problems than shallow methods
- Rebranding to Deep Nets, Deep Learning

Ⅱ. Neural Nets(NN) for XOR & Backpropagation

01 Solving XOR with a Neural Net



01 Solving XOR with a Neural Net

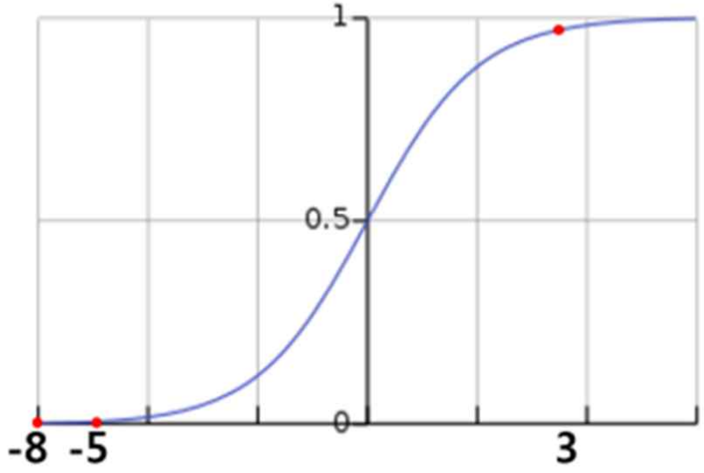


$$sigmoid\left([0 \quad 0]\begin{bmatrix}5\\5\end{bmatrix} - 8\right) = sigmoid(-8) = 0$$

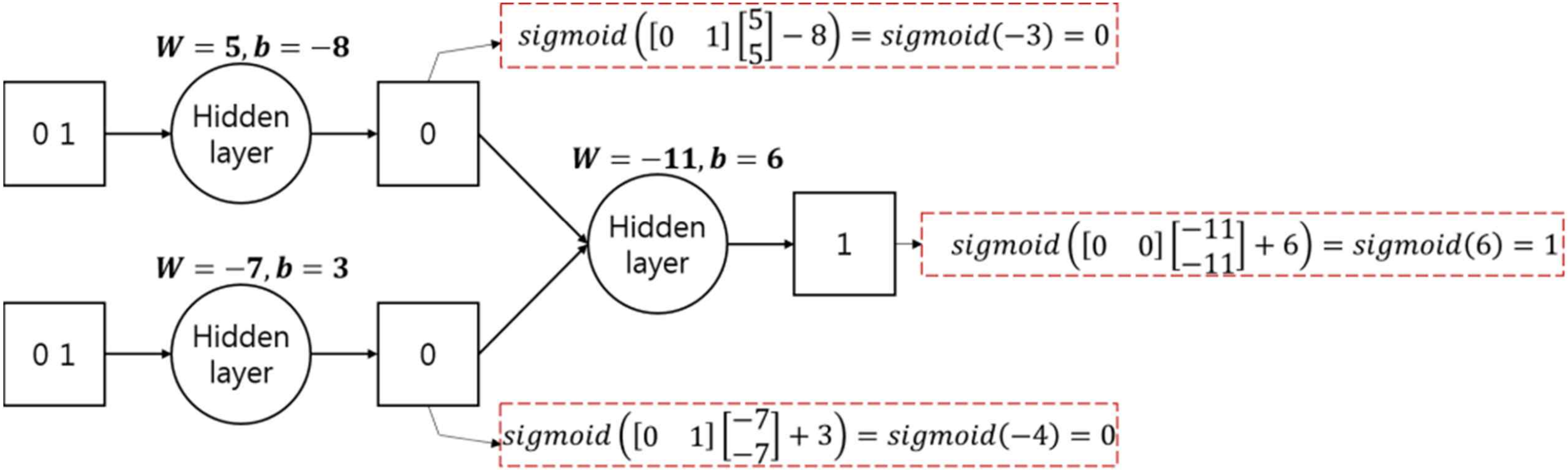
$$sigmoid\left([0 \quad 1]\begin{bmatrix}-11\\-11\end{bmatrix} + 6\right) = sigmoid(-5) = 0$$

$$sigmoid\left([0 \quad 0]\begin{bmatrix}-7\\-7\end{bmatrix} + 3\right) = sigmoid(3) = 0$$

x_1	x_2	y_1	y_2	$\hat{y}$	XOR
0	0	0	1	0	0
0	1				1
1	0				1
1	1				0



01 Solving XOR with a Neural Net

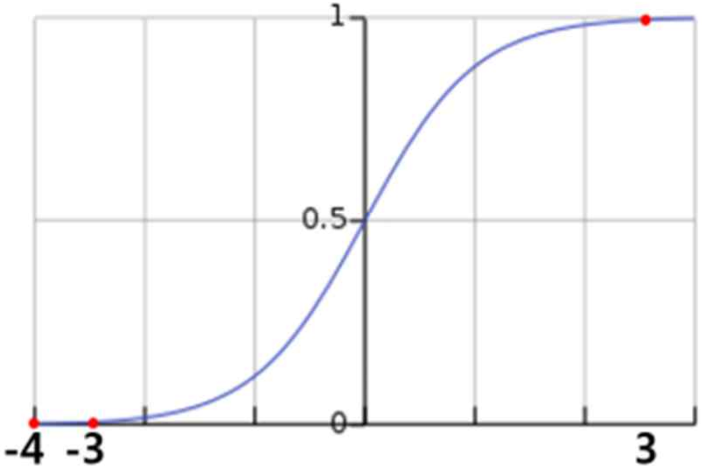


$$sigmoid\left([0 \quad 1]\begin{bmatrix}5\\5\end{bmatrix} - 8\right) = sigmoid(-3) = 0$$

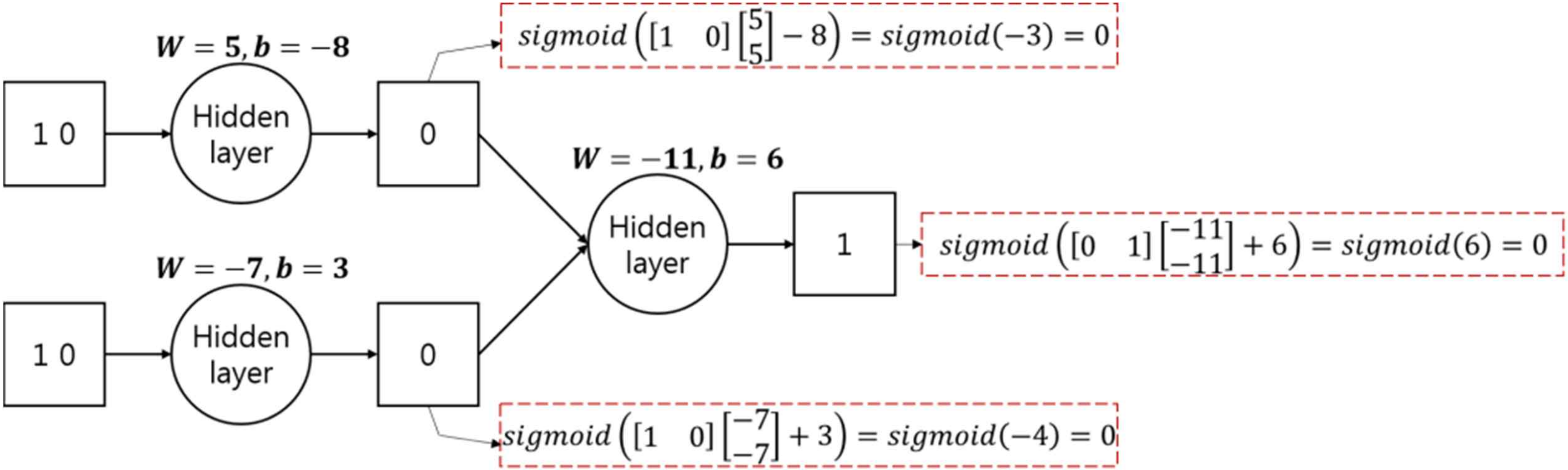
$$sigmoid\left([0 \quad 0]\begin{bmatrix}-11\\-11\end{bmatrix} + 6\right) = sigmoid(6) = 1$$

$$sigmoid\left([0 \quad 1]\begin{bmatrix}-7\\-7\end{bmatrix} + 3\right) = sigmoid(-4) = 0$$

x_1	x_2	y_1	y_2	$\hat{y}$	XOR
0	0	0	1	0	0
0	1	0	0	1	1
1	0				1
1	1				0



01 Solving XOR with a Neural Net

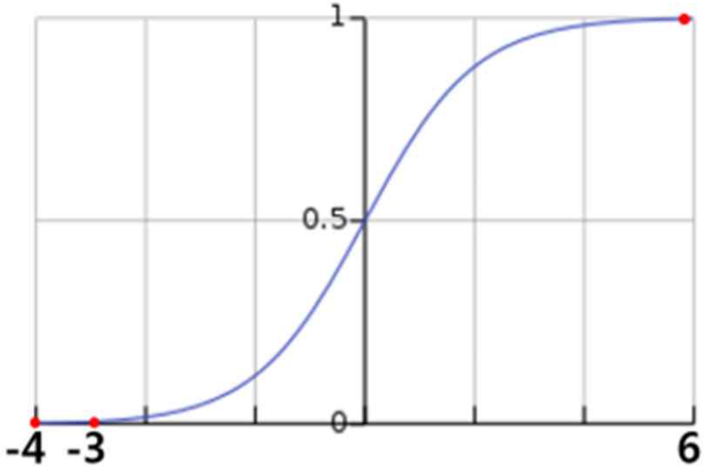


$$sigmoid\left(\begin{bmatrix}1 & 0\end{bmatrix}\begin{bmatrix}5\\5\end{bmatrix} - 8\right) = sigmoid(-3) = 0$$

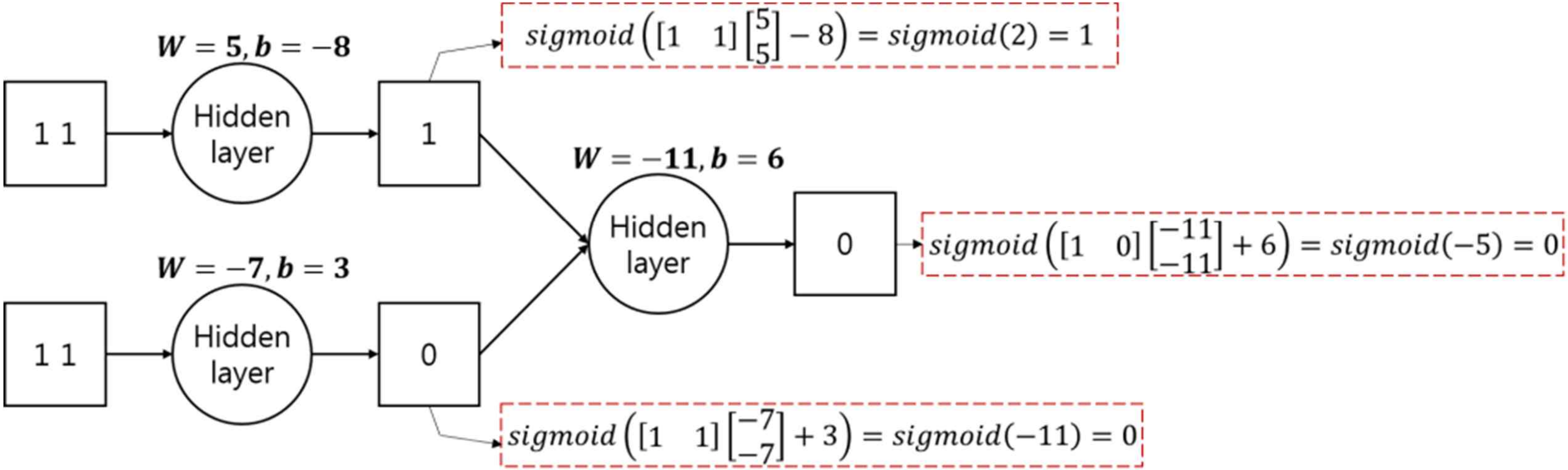
$$sigmoid\left(\begin{bmatrix}0 & 1\end{bmatrix}\begin{bmatrix}-11\\-11\end{bmatrix} + 6\right) = sigmoid(6) = 0$$

$$sigmoid\left(\begin{bmatrix}1 & 0\end{bmatrix}\begin{bmatrix}-7\\-7\end{bmatrix} + 3\right) = sigmoid(-4) = 0$$

x_1	x_2	y_1	y_2	$\hat{y}$	XOR
0	0	0	1	0	0
0	1	0	0	1	1
1	0	0	0	1	1
1	1				0



01 Solving XOR with a Neural Net

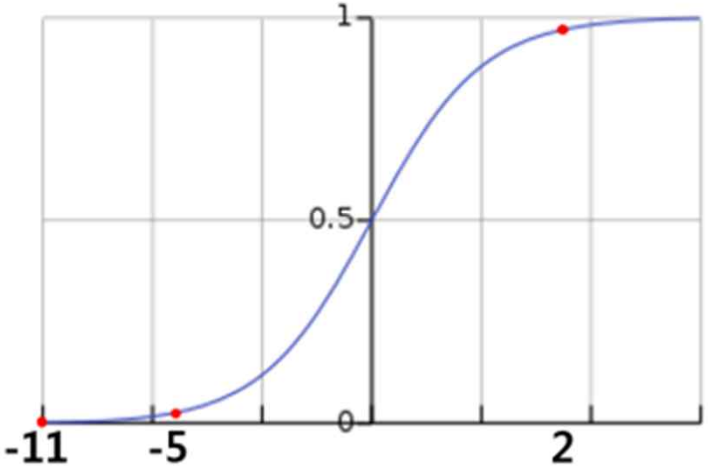


$$sigmoid\left([1 \quad 1]\begin{bmatrix}5\\5\end{bmatrix} - 8\right) = sigmoid(2) = 1$$

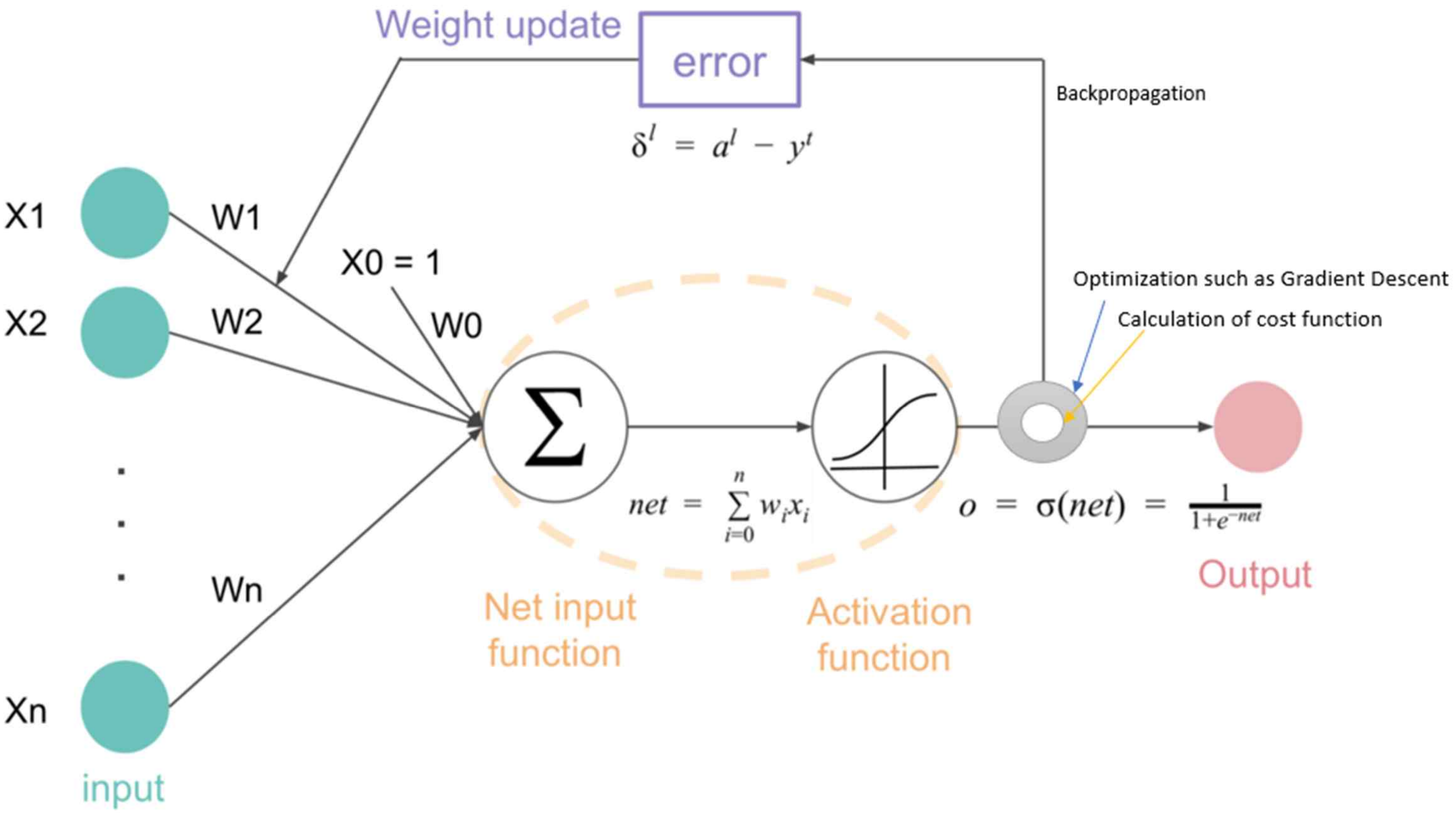
$$sigmoid\left([1 \quad 0]\begin{bmatrix}-11\\-11\end{bmatrix} + 6\right) = sigmoid(-5) = 0$$

$$sigmoid\left([1 \quad 1]\begin{bmatrix}-7\\-7\end{bmatrix} + 3\right) = sigmoid(-11) = 0$$

x_1	x_2	y_1	y_2	$\hat{y}$	XOR
0	0	0	1	0	0
0	1	0	0	1	1
1	0	0	0	1	1
1	1	1	0	0	0



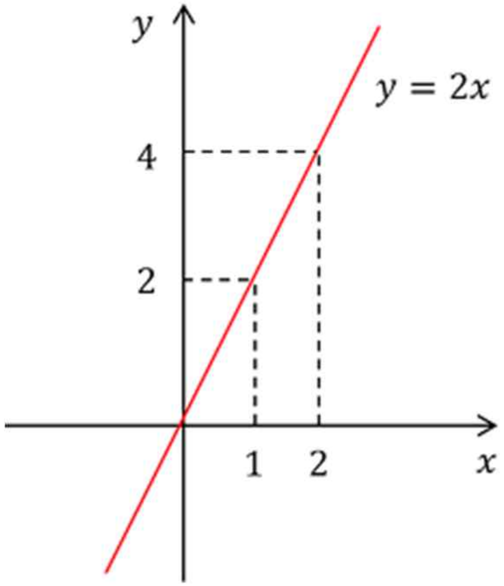
02 Backpropagation

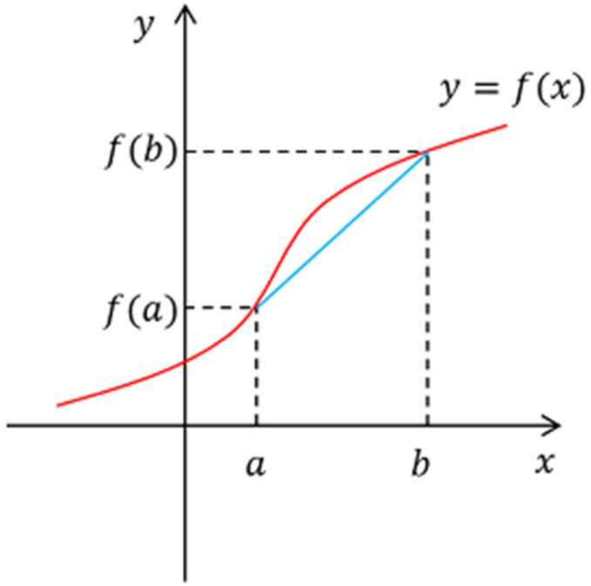

Weight update
error
$\delta^l = a^l - y^t$
Backpropagation
X1
W1
X2
W2
X0 = 1
W0
Wn
Xn
input
Optimization such as Gradient Descent
Calculation of cost function
$net = \sum_{i=0}^{n} w_i x_i$
$o = \sigma(net) = \frac{1}{1+e^{-net}}$
Net input function
Activation function
Output

02 Backpropagation

미분의 이해

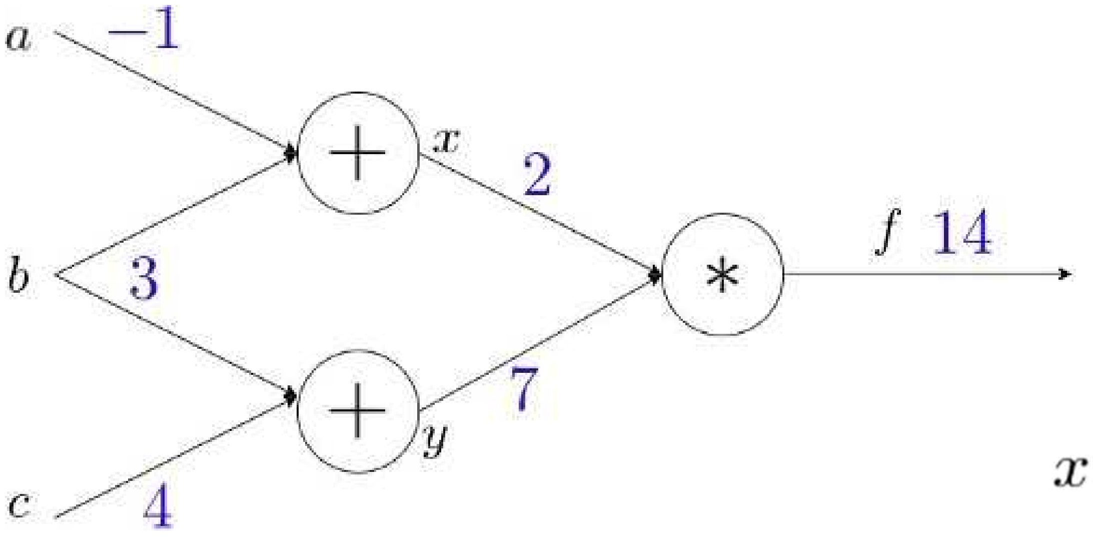
$$A\text{의}B\text{에 대한 변화율} = \frac{A\text{의 변화량}}{B\text{의 변화량}} = \frac{dA}{dB}$$





02 Backpropagation: example

$$f = (a+b)(b+c) \qquad with \quad a=-1, \quad b=3, \quad c=4$$

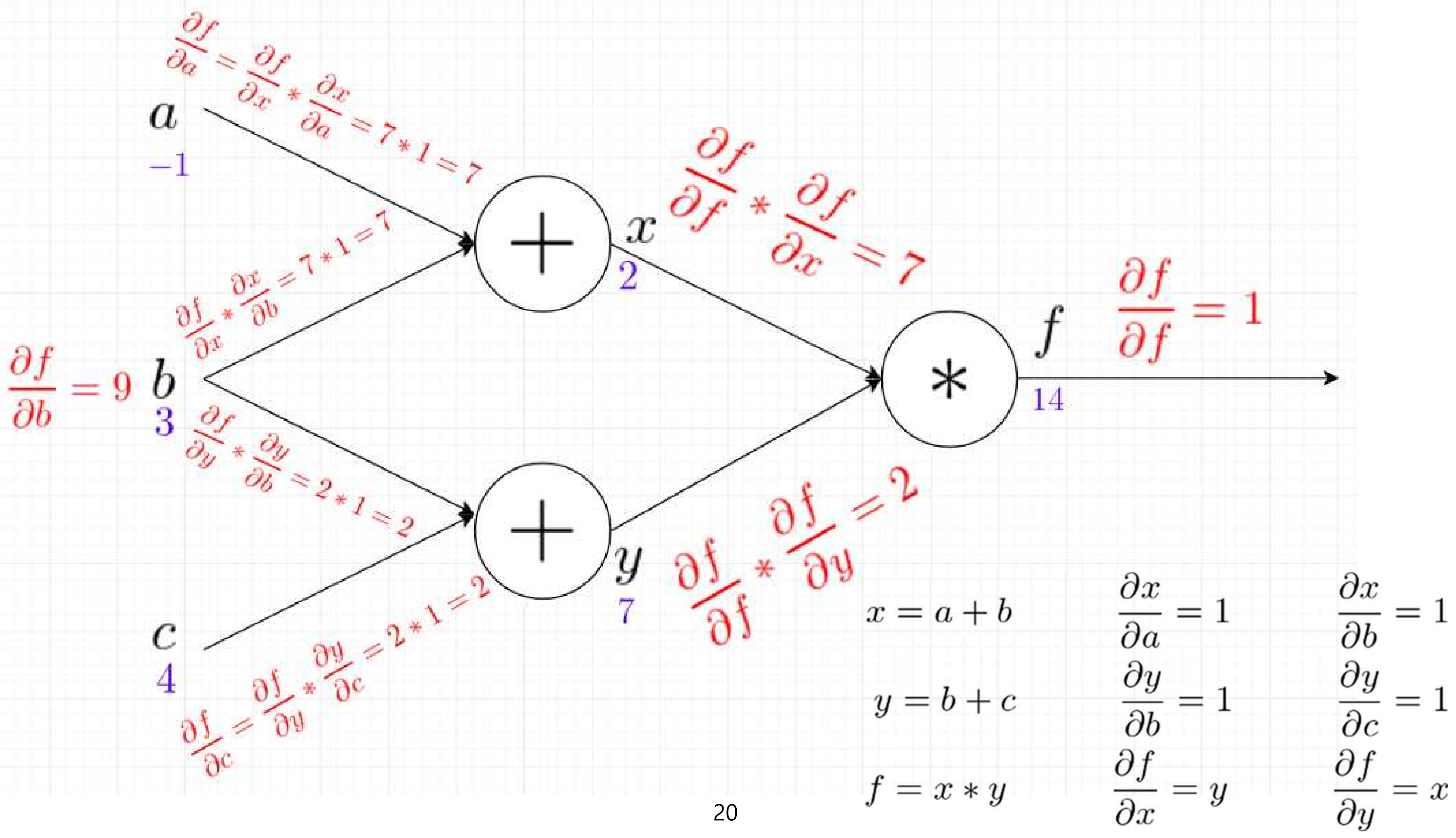


$$x = a + b \qquad \frac{\partial x}{\partial a} = 1 \qquad \frac{\partial x}{\partial b} = 1$$

$$y = b + c \qquad \frac{\partial y}{\partial b} = 1 \qquad \frac{\partial y}{\partial c} = 1$$

$$f = x * y \qquad \frac{\partial f}{\partial x} = y \qquad \frac{\partial f}{\partial y} = x$$

02 Backpropagation: example

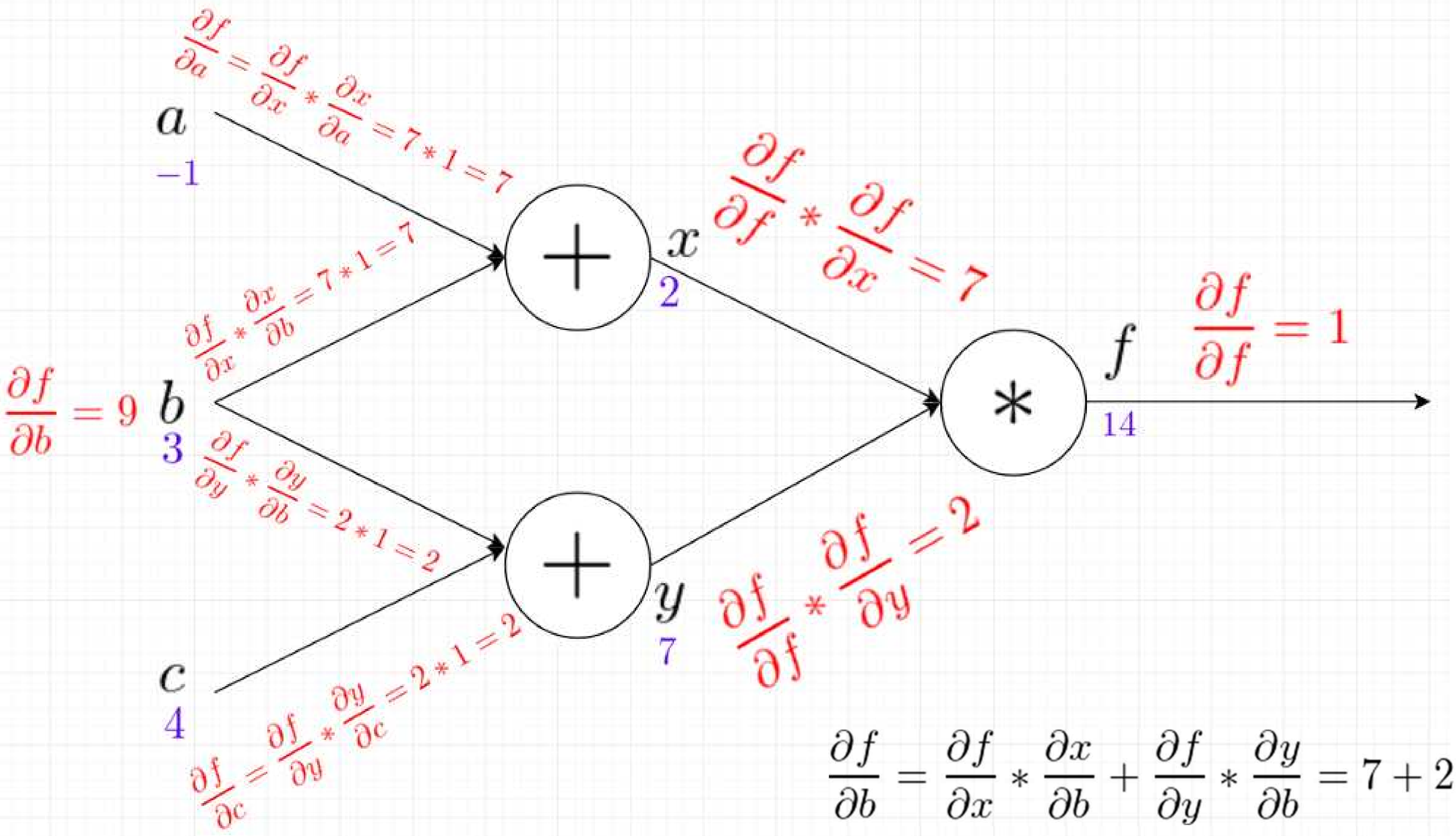

$\frac{\partial f}{\partial a} = \frac{\partial f}{\partial x} * \frac{\partial x}{\partial a} = 7 * 1 = 7$
a
-1
$\frac{\partial f}{\partial x} * \frac{\partial x}{\partial b} = 7 * 1 = 7$
$\frac{\partial f}{\partial b} = 9$
b
3
$\frac{\partial f}{\partial y} * \frac{\partial y}{\partial b} = 2 * 1 = 2$
c
4
$\frac{\partial f}{\partial c} = \frac{\partial f}{\partial y} * \frac{\partial y}{\partial c} = 2 * 1 = 2$
$+$
$+$
x
2
y
7
$\frac{\partial f}{\partial f} * \frac{\partial f}{\partial x} = 7$
$\frac{\partial f}{\partial f} * \frac{\partial f}{\partial y} = 2$
$*$
f
14
$\frac{\partial f}{\partial f} = 1$

$$x = a + b \qquad \frac{\partial x}{\partial a} = 1 \qquad \frac{\partial x}{\partial b} = 1$$

$$y = b + c \qquad \frac{\partial y}{\partial b} = 1 \qquad \frac{\partial y}{\partial c} = 1$$

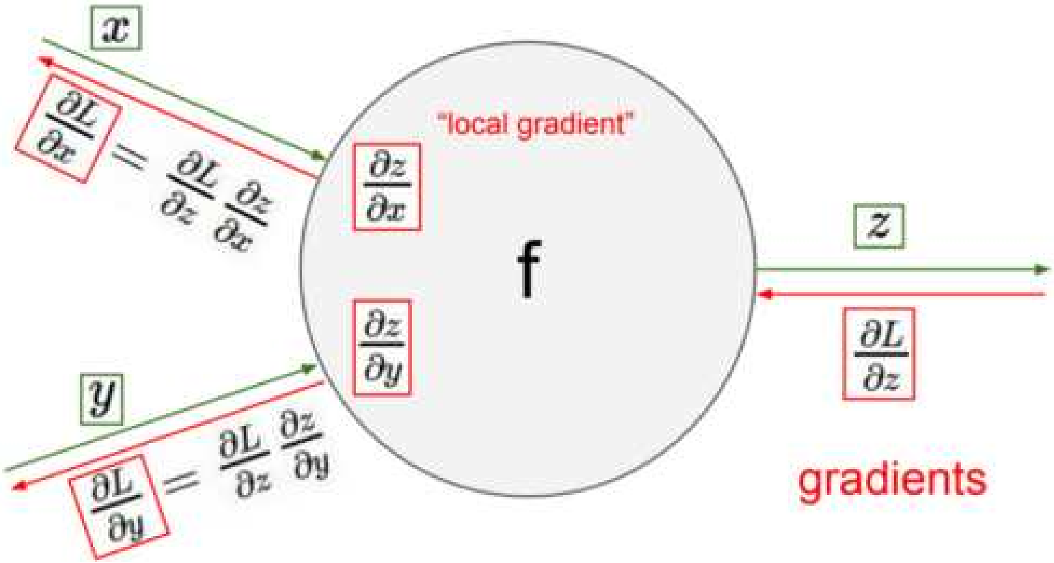
$$f = x * y \qquad \frac{\partial f}{\partial x} = y \qquad \frac{\partial f}{\partial y} = x$$

02 Backpropagation: example

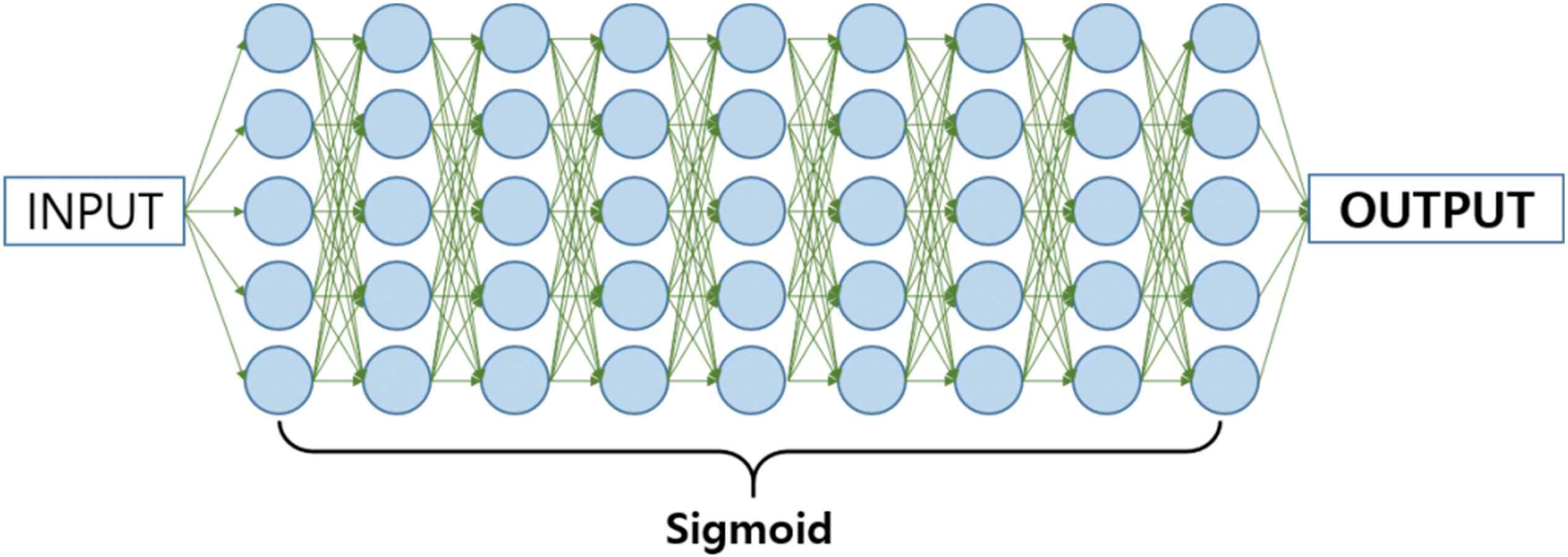

$\frac{\partial f}{\partial a} = \frac{\partial f}{\partial x} * \frac{\partial x}{\partial a} = 7 * 1 = 7$
a
-1
$\frac{\partial f}{\partial x} * \frac{\partial x}{\partial b} = 7 * 1 = 7$
$\frac{\partial f}{\partial b} = 9$
b
3
$\frac{\partial f}{\partial y} * \frac{\partial y}{\partial b} = 2 * 1 = 2$
c
4
$\frac{\partial f}{\partial c} = \frac{\partial f}{\partial y} * \frac{\partial y}{\partial c} = 2 * 1 = 2$
$+$
x
2
$\frac{\partial f}{\partial f} * \frac{\partial f}{\partial x} = 7$
$+$
y
7
$\frac{\partial f}{\partial f} * \frac{\partial f}{\partial y} = 2$
$*$
f
14
$\frac{\partial f}{\partial f} = 1$

$$\frac{\partial f}{\partial b} = \frac{\partial f}{\partial x} * \frac{\partial x}{\partial b} + \frac{\partial f}{\partial y} * \frac{\partial y}{\partial b} = 7 + 2 = 9$$

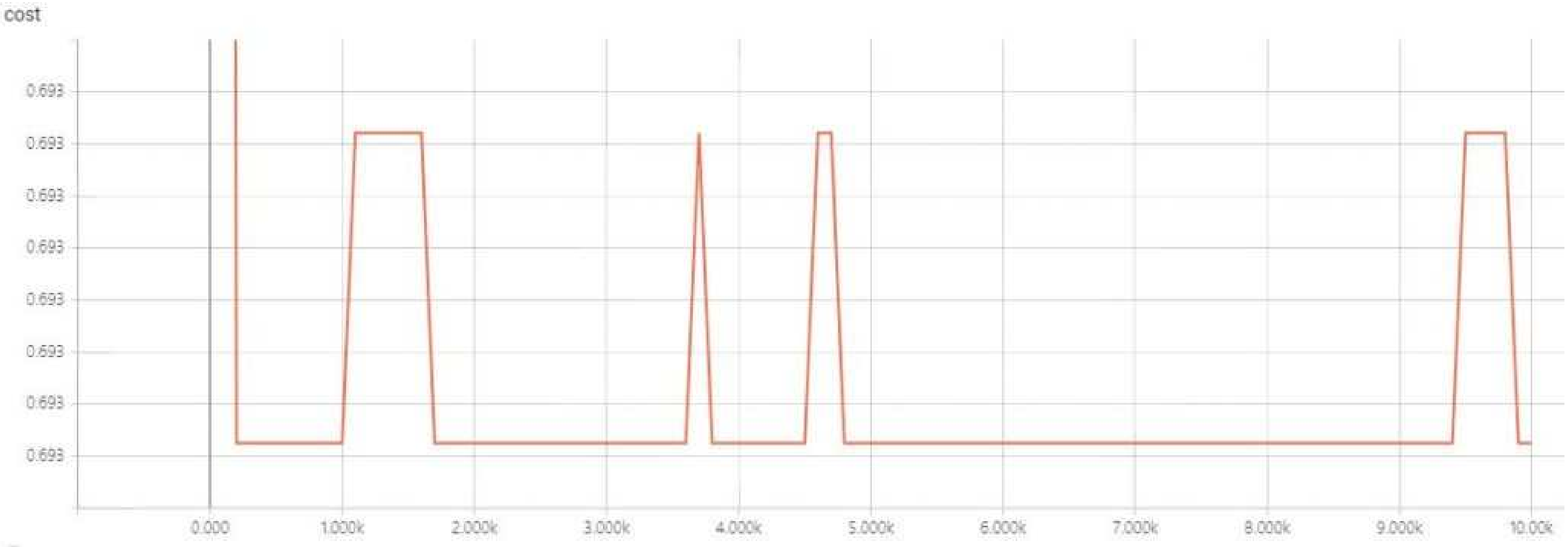
02 Backpropagation


x
$\frac{\partial L}{\partial x} = \frac{\partial L}{\partial z}\frac{\partial z}{\partial x}$
"local gradient"
$\frac{\partial z}{\partial x}$
f
$\frac{\partial z}{\partial y}$
z
$\frac{\partial L}{\partial z}$
y
$\frac{\partial L}{\partial y} = \frac{\partial L}{\partial z}\frac{\partial z}{\partial y}$
gradients

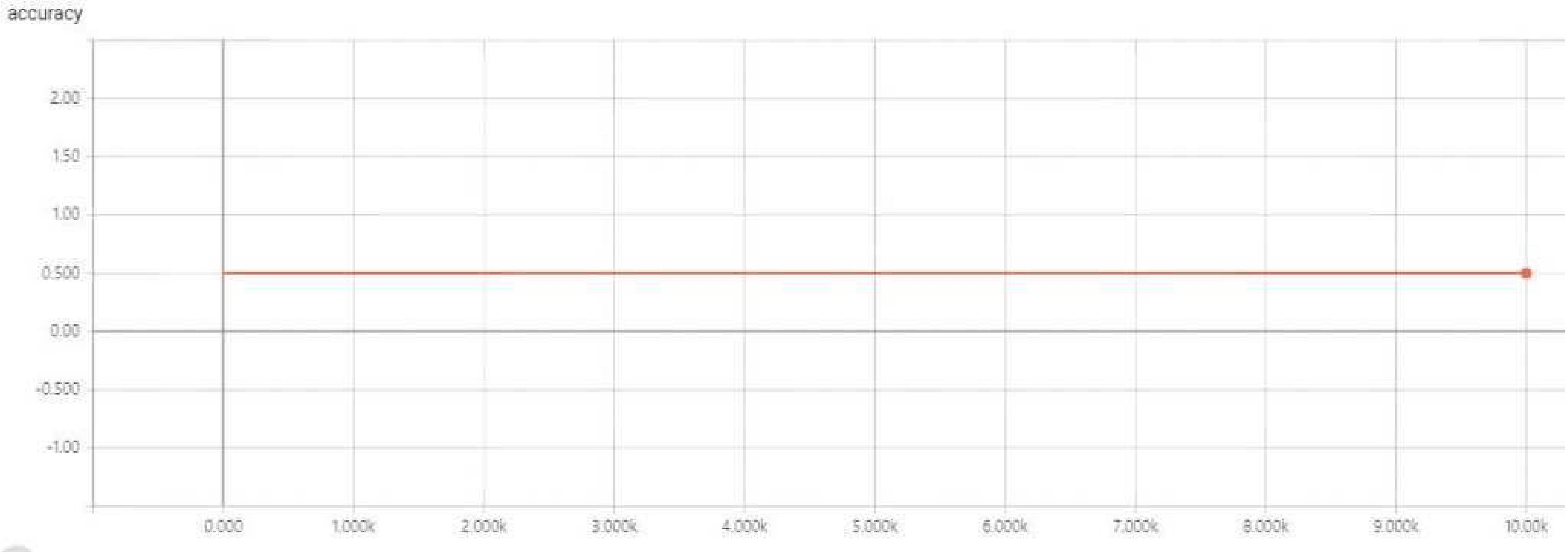
04 Deep & Wide NN



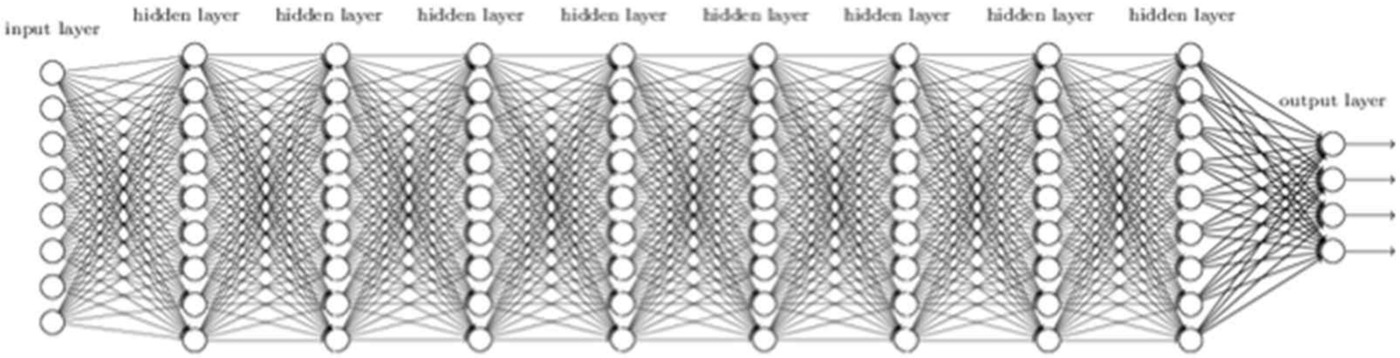
04 Deep & Wide NN



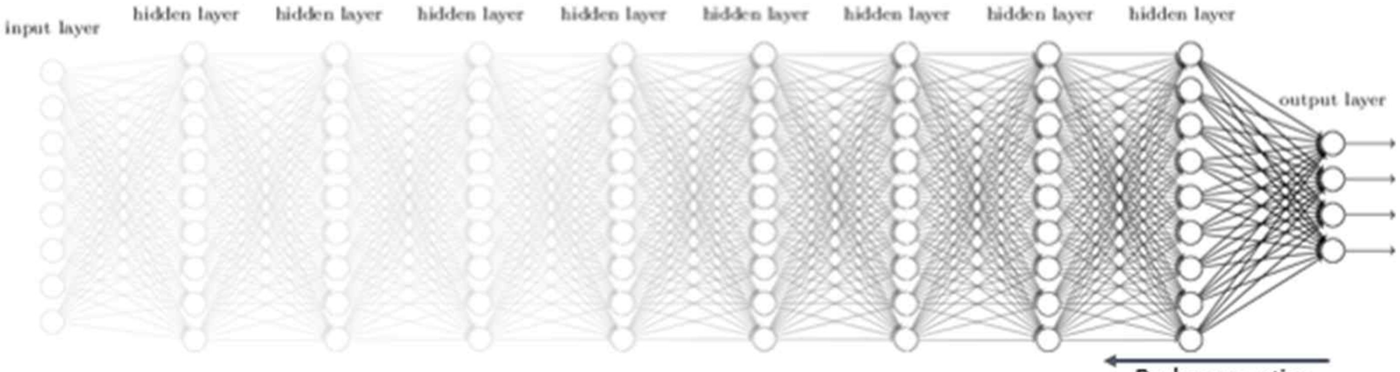
04 Deep & Wide NN



04 Vanishing Gradient Problem

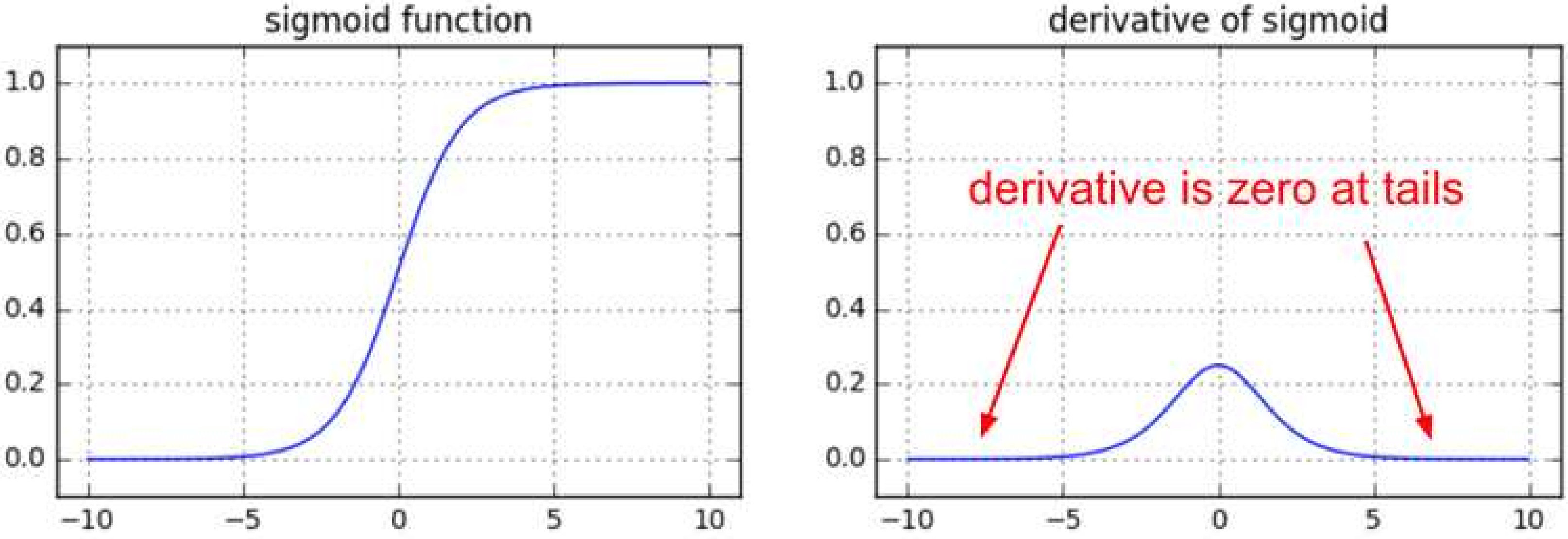


Deep Neural Network

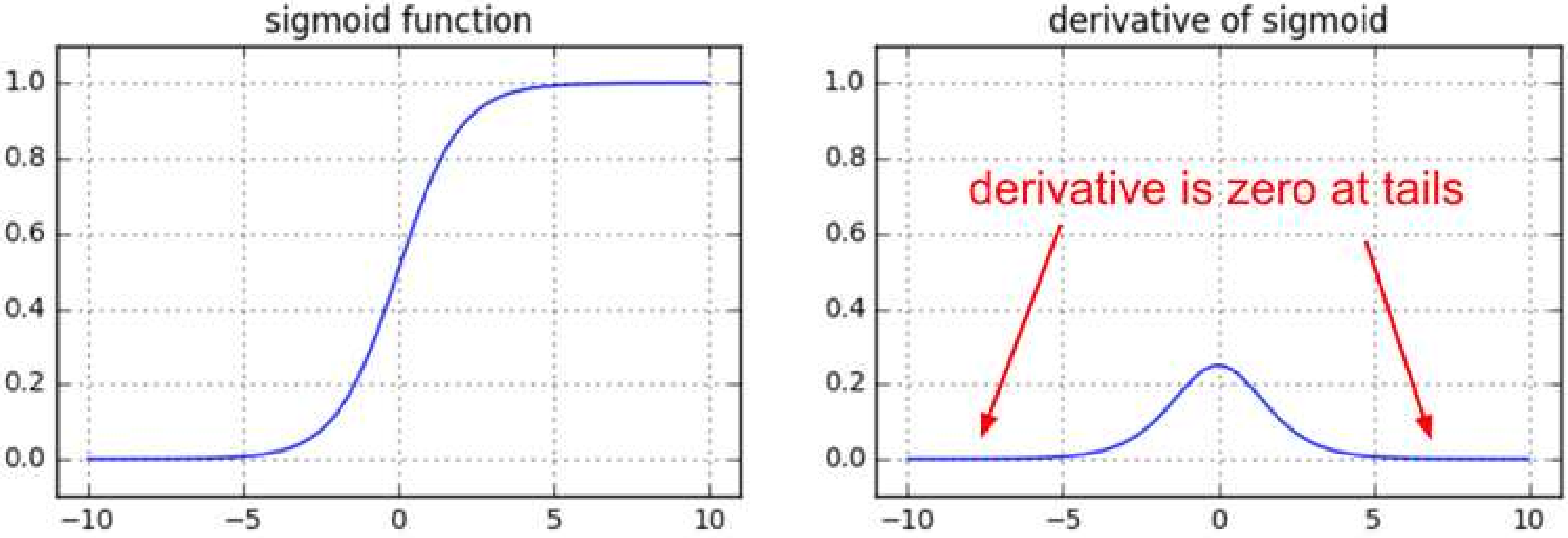


Vanishing Gradient

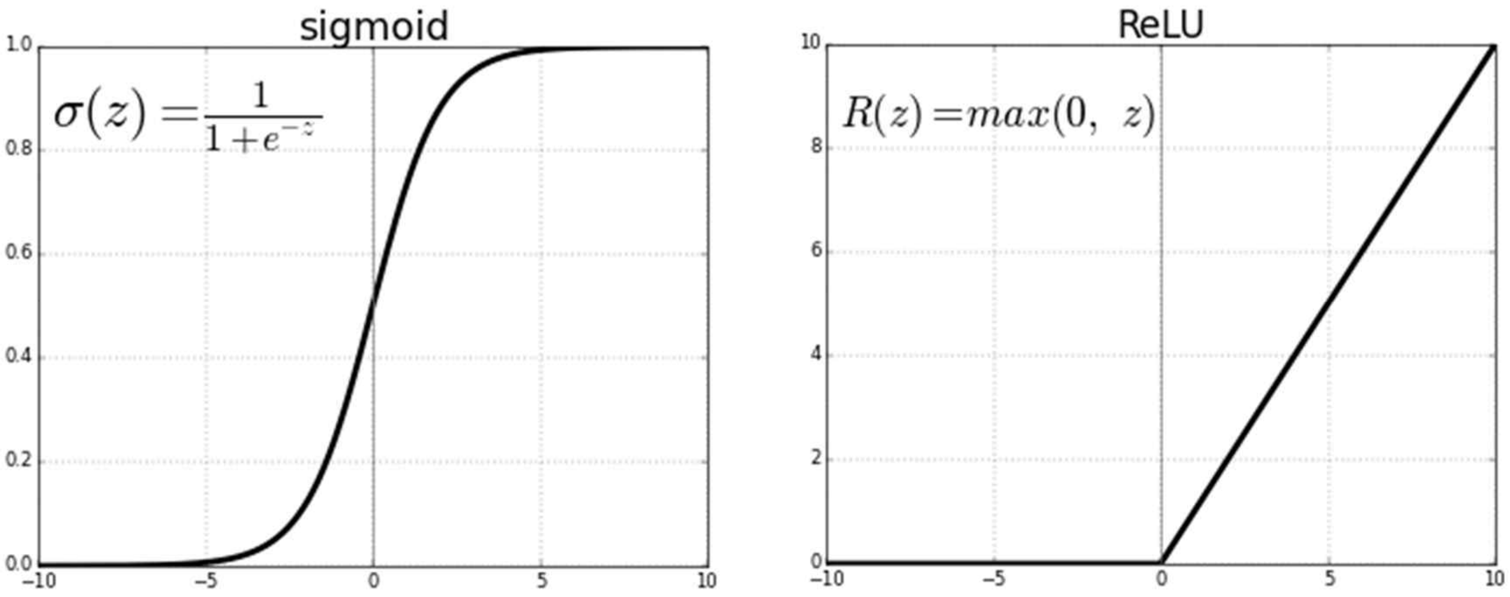
04 Vanishing Gradient Problem



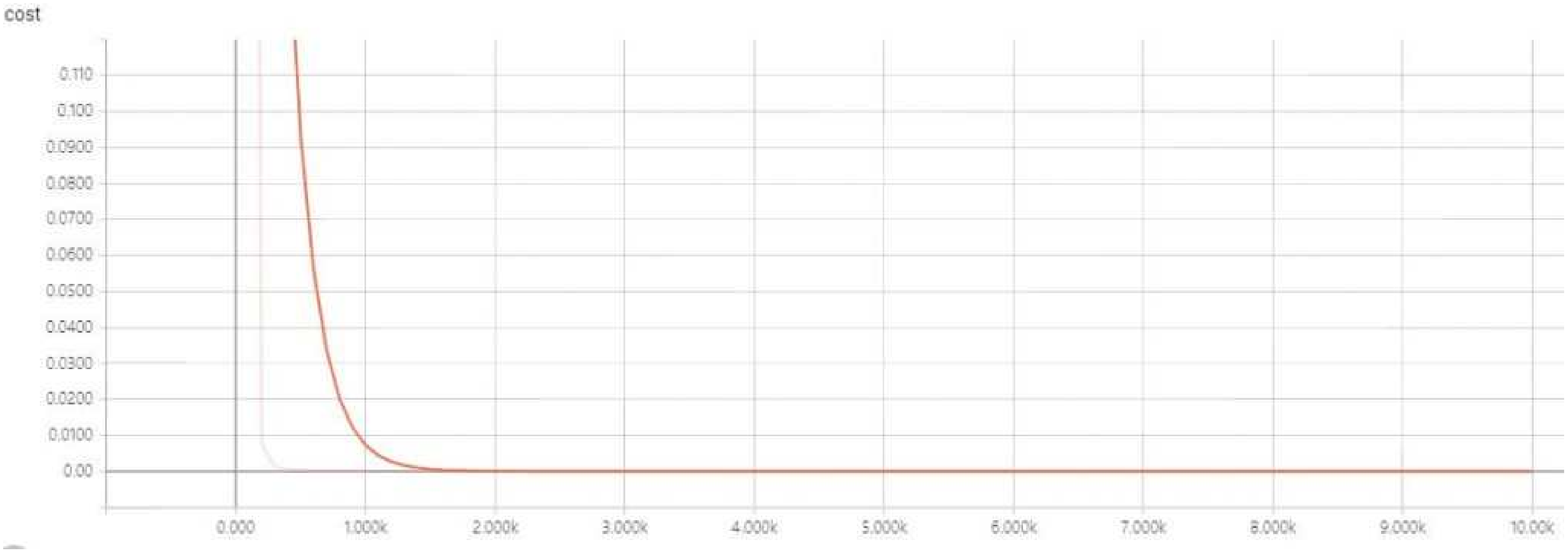
05 ReLU: Rectified Linear Unit



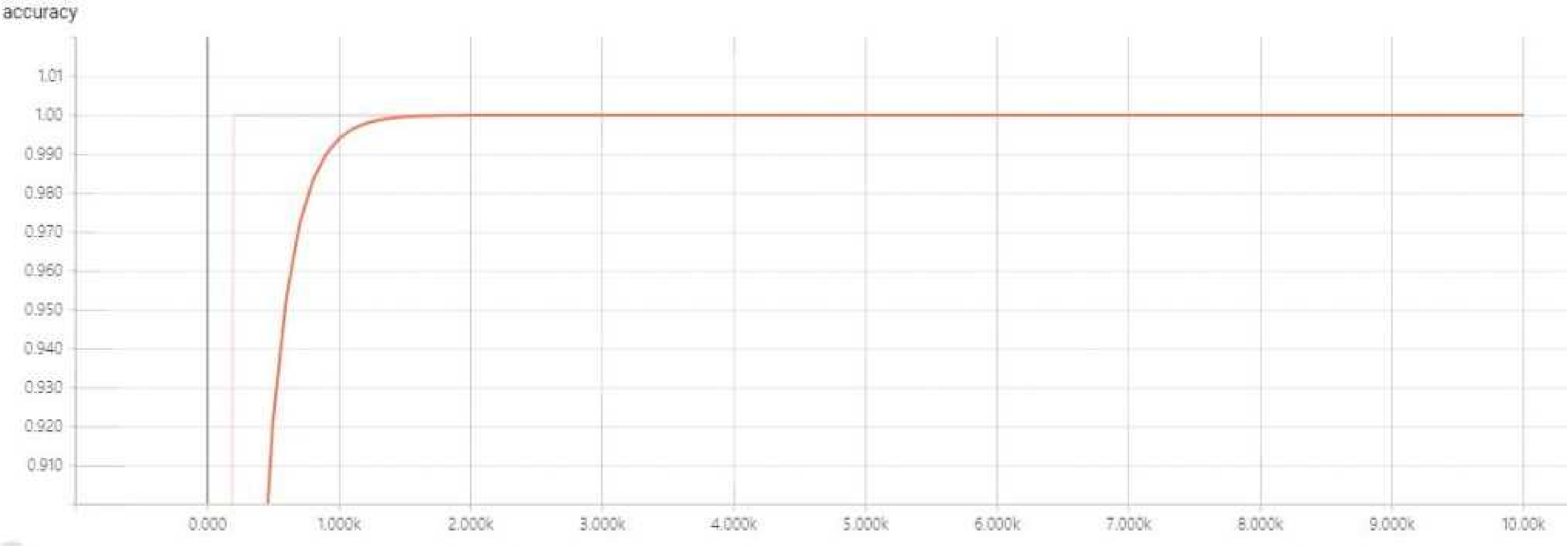
05 ReLU: Rectified Linear Unit



05 ReLU: Rectified Linear Unit



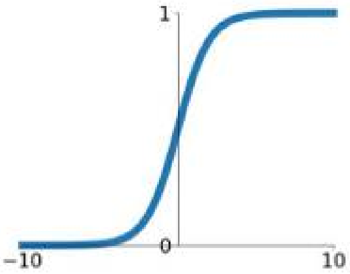
05 ReLU: Rectified Linear Unit



06 Activation Functions

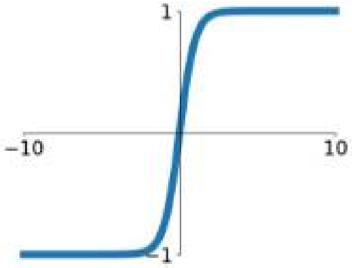
Sigmoid

$\sigma(x) = \frac{1}{1+e^{-x}}$



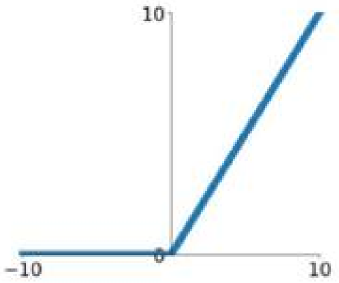
tanh

$\tanh(x)$



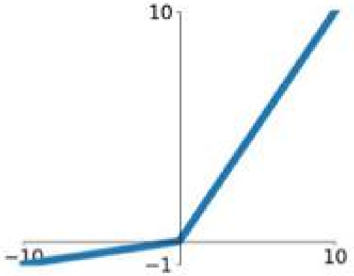
ReLU

$\max(0, x)$



Leaky ReLU

$\max(0.1x, x)$

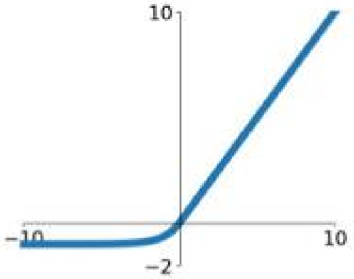


Maxout

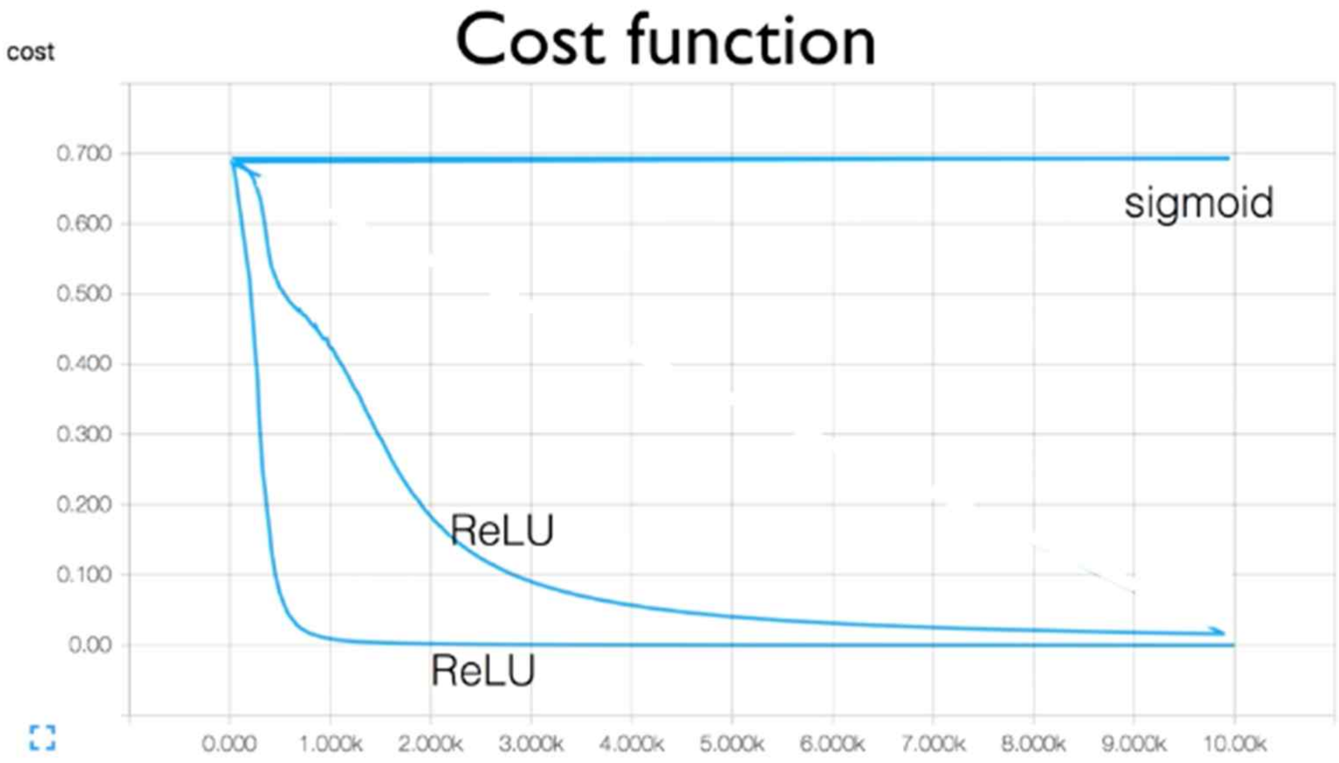
$\max(w_1^T x + b_1, w_2^T x + b_2)$

ELU

$$\begin{cases} x & x \geq 0 \\ \alpha(e^x - 1) & x < 0 \end{cases}$$



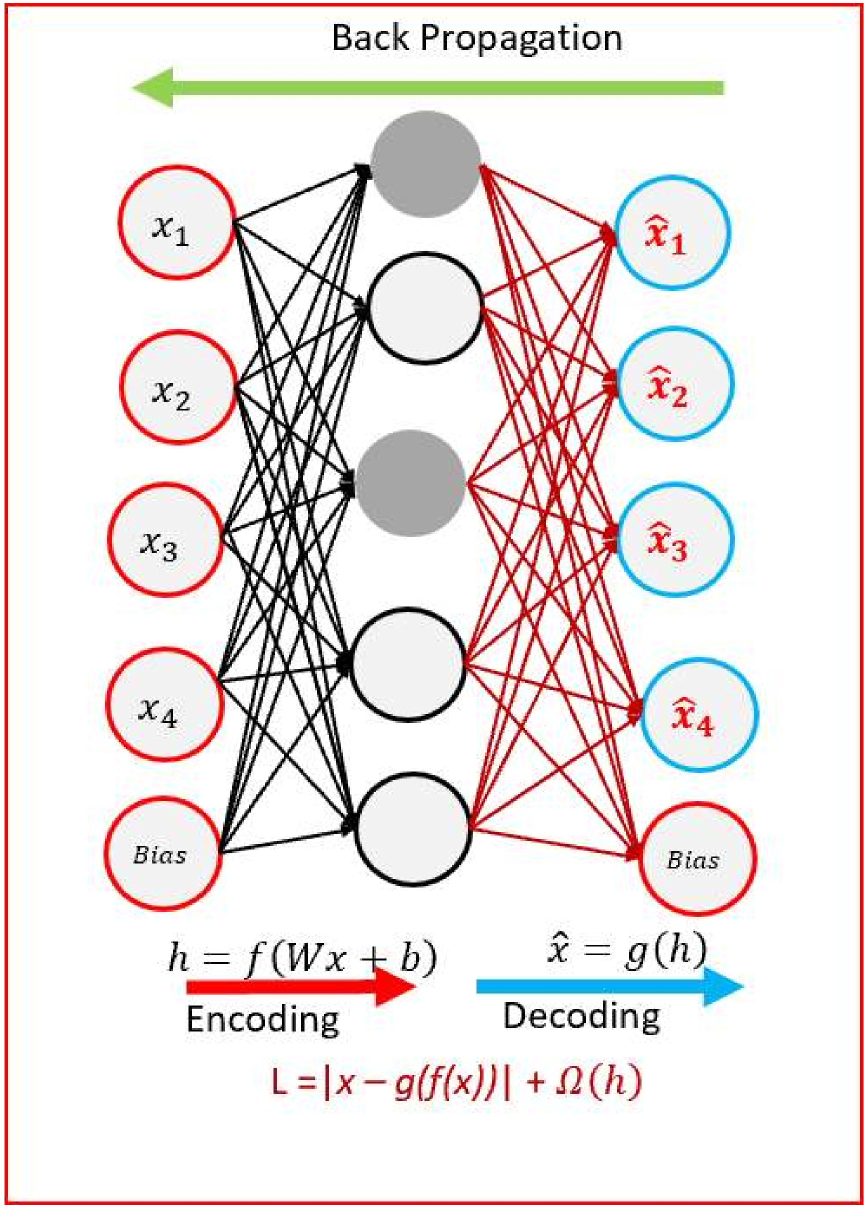
07 Initialize Weights in a Smart Way



07 Initialize Weights in a Smart Way

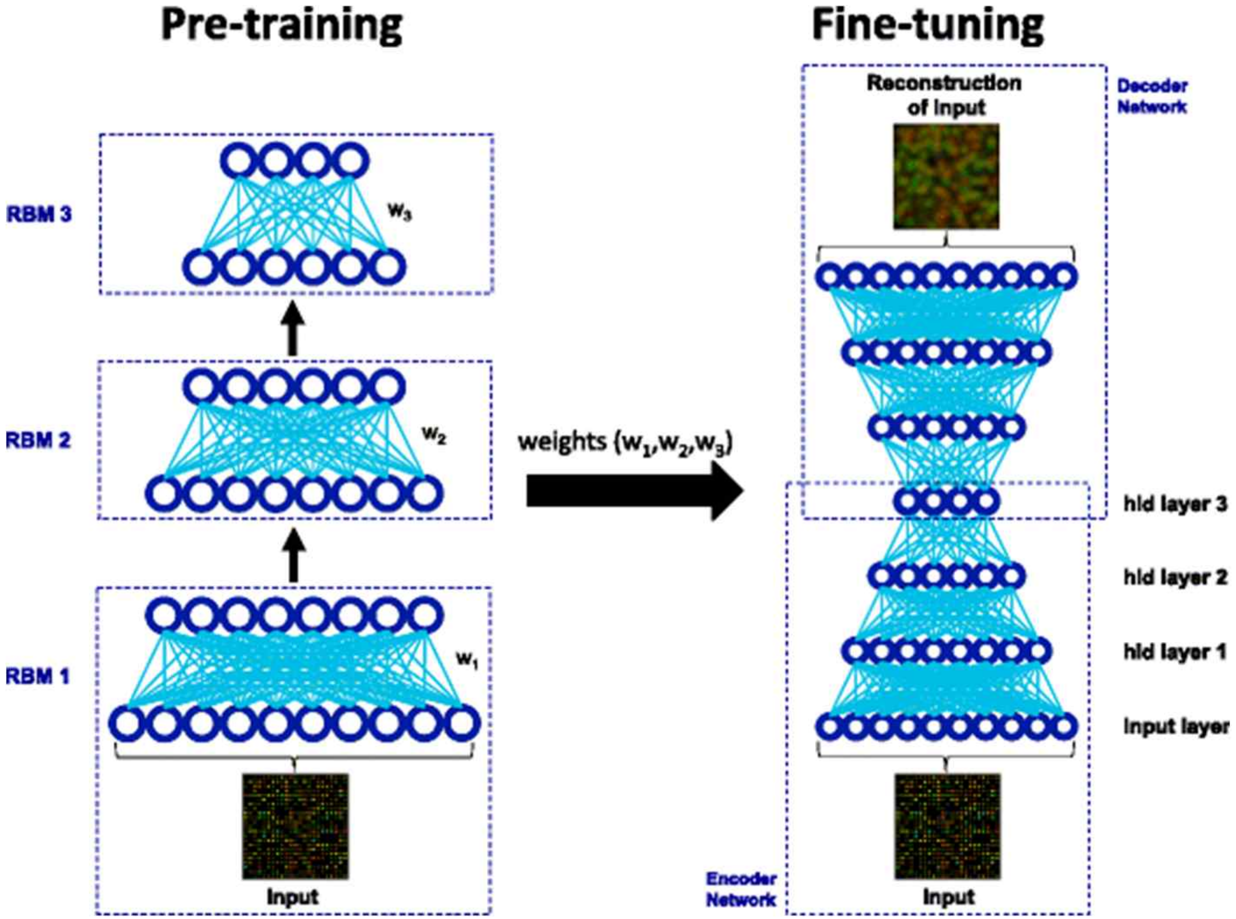
Deep Belief Network

- Weight initialized by RBM



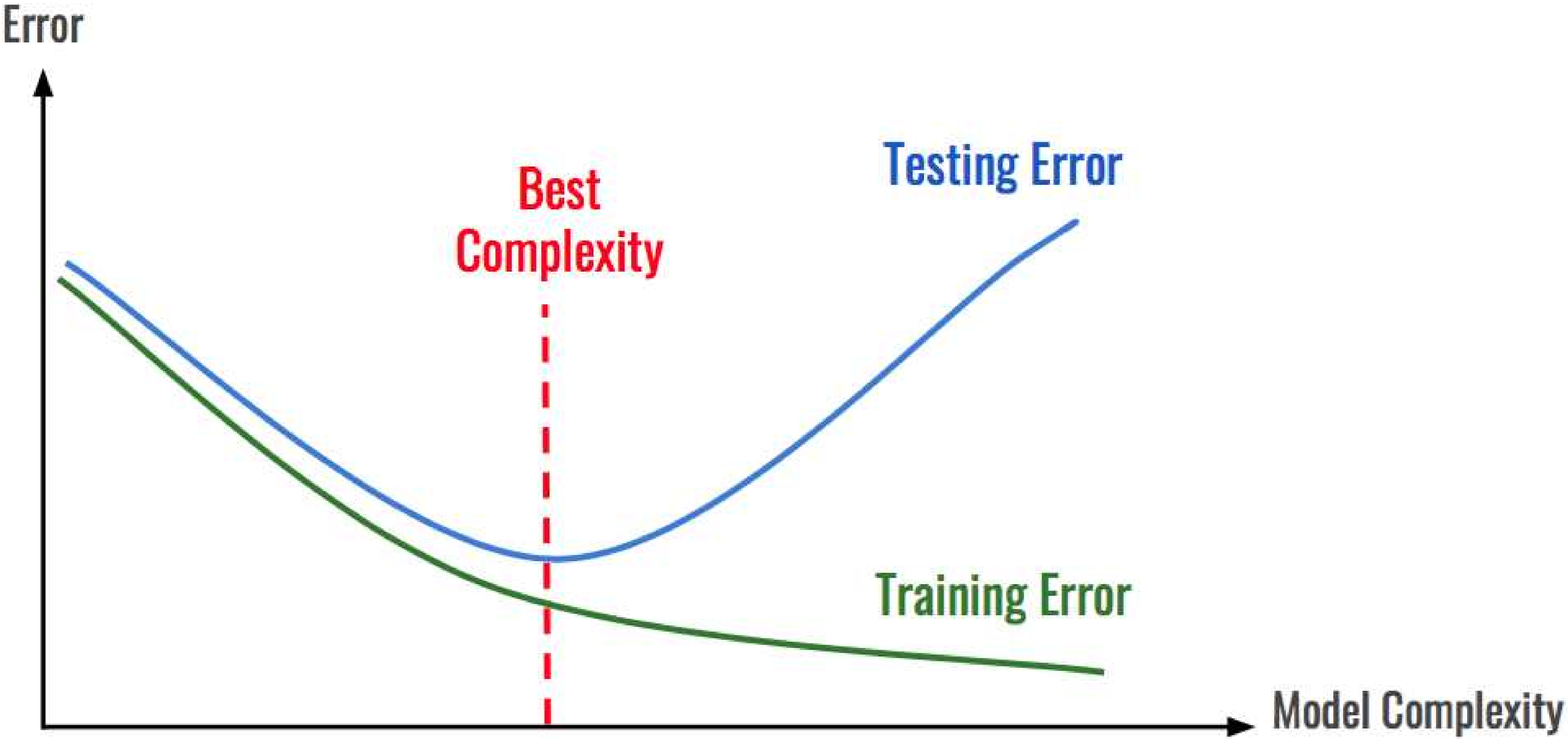
07 Initialize Weights in a Smart Way

Pre-training and Fine tuning

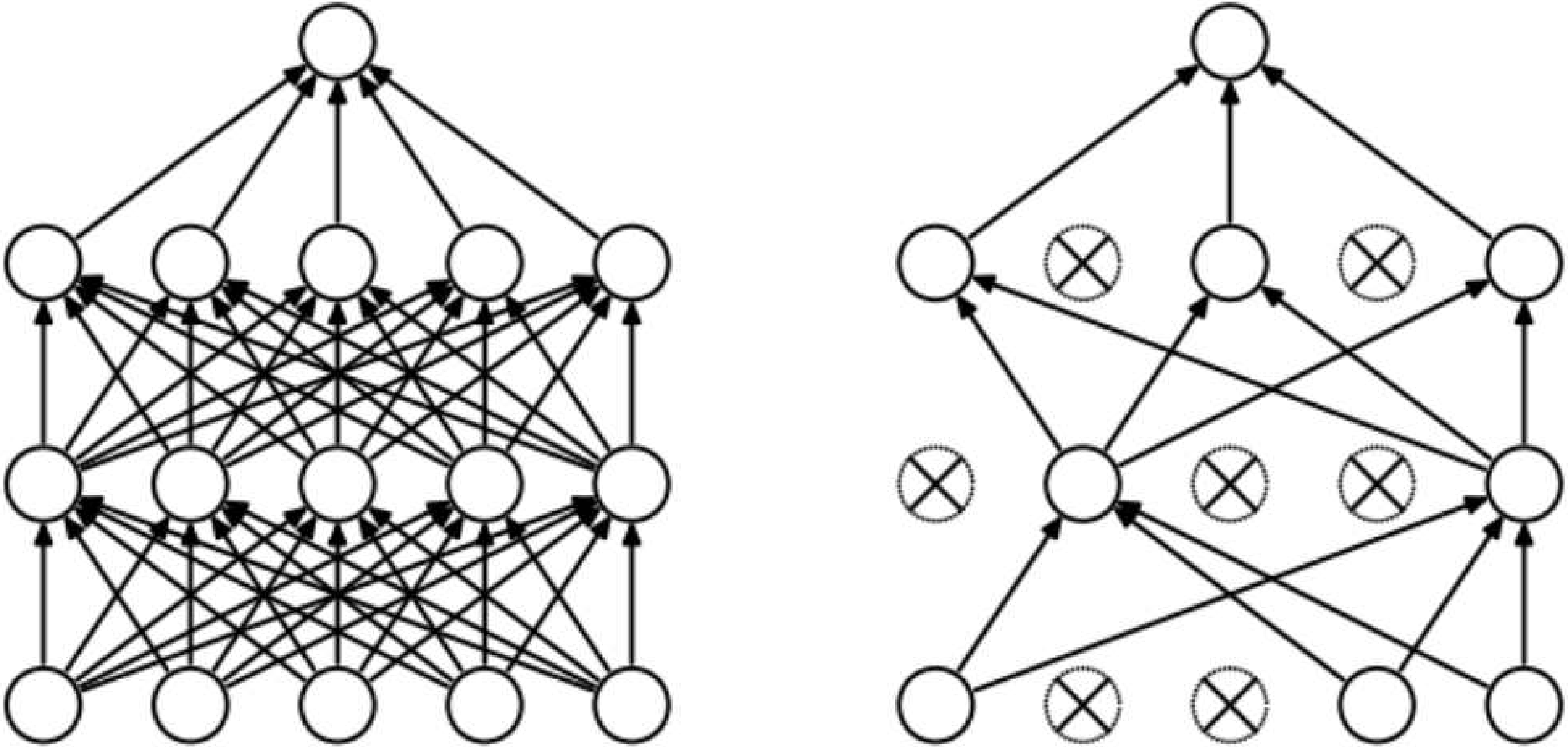


08 Dropout

Overfitting



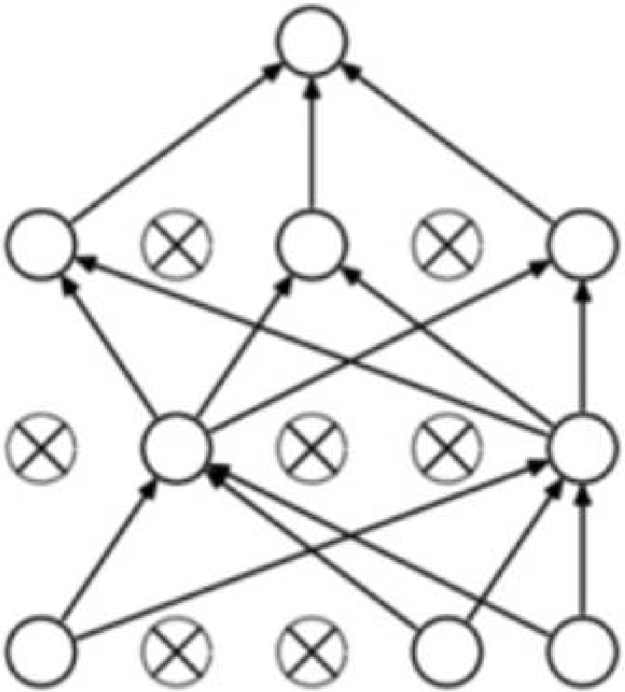
08 Dropout



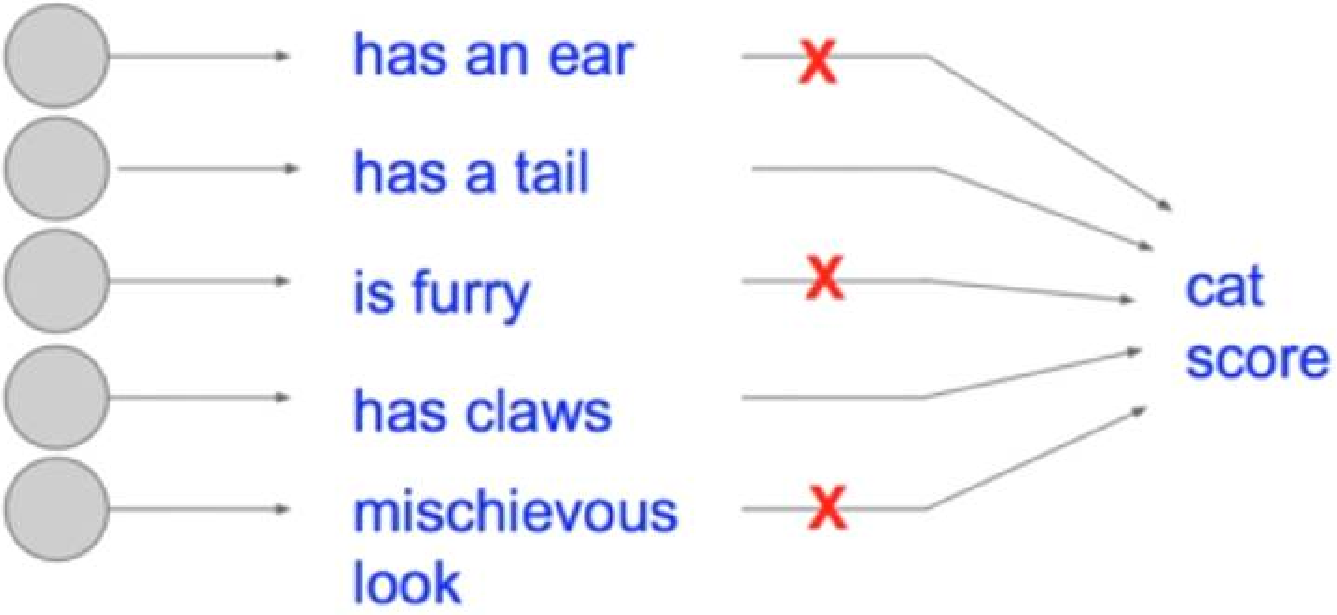
(a) Standard Neural Net

(b) After applying dropout.

08 Dropout



Forces the network to have a redundant representation.



has an ear

has a tail

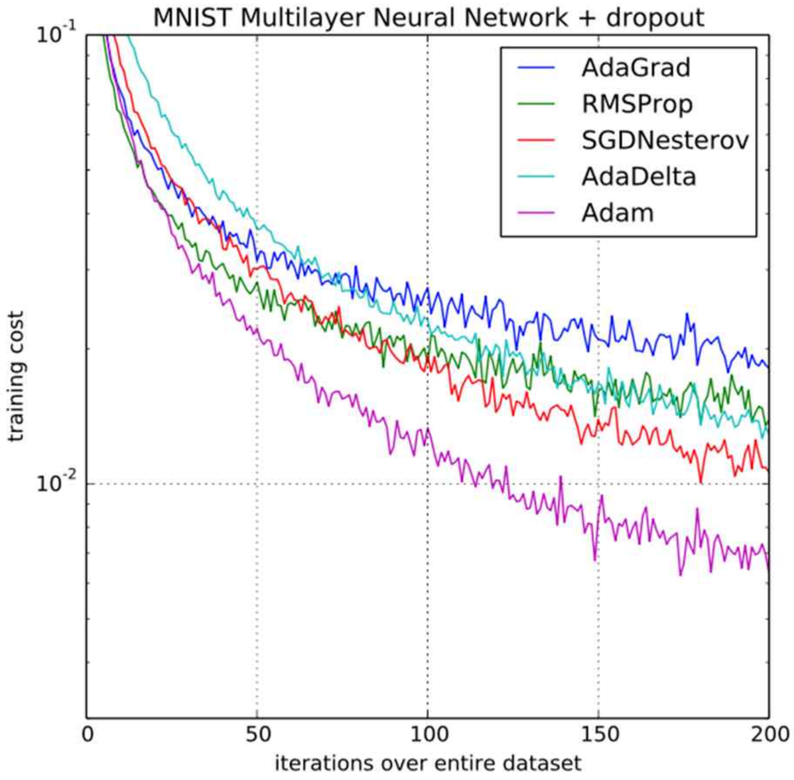
is furry

has claws

mischievous look

cat score

09 Optimizers



02 Exercise

Wide and Deep NN for MNIST