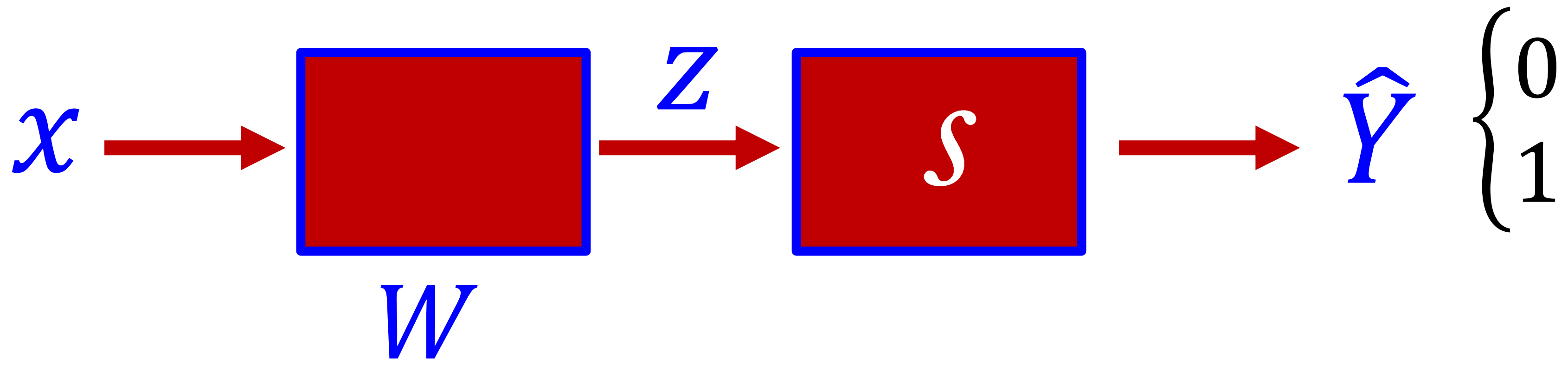
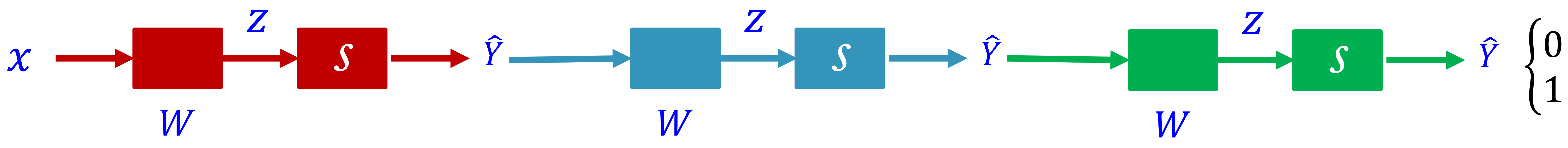
Lecture 09

Neural Nets(NN) for XOR, Back propagation

One logistic regression unit cannot separate XOR

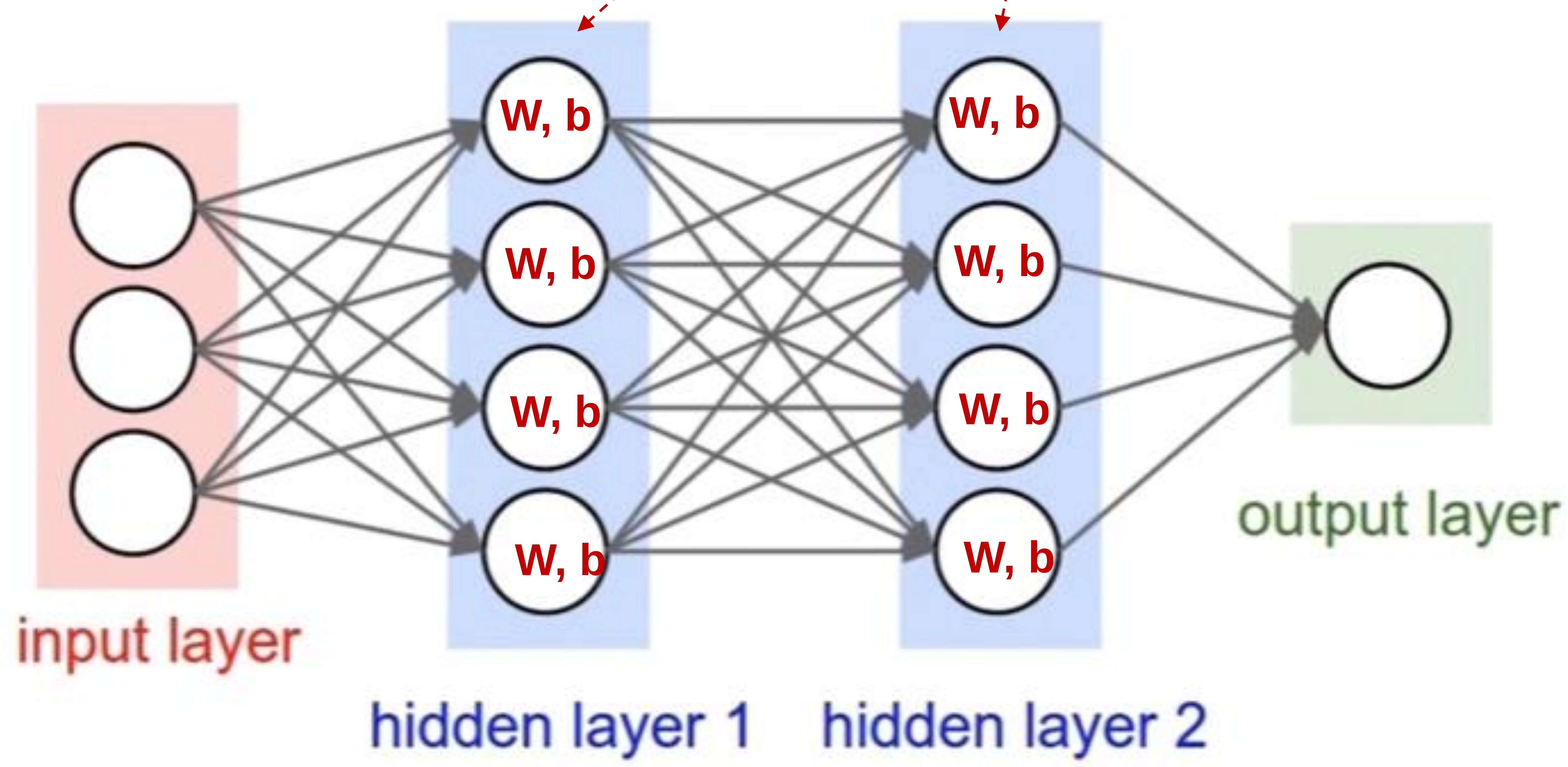


Multiple logistic regression units



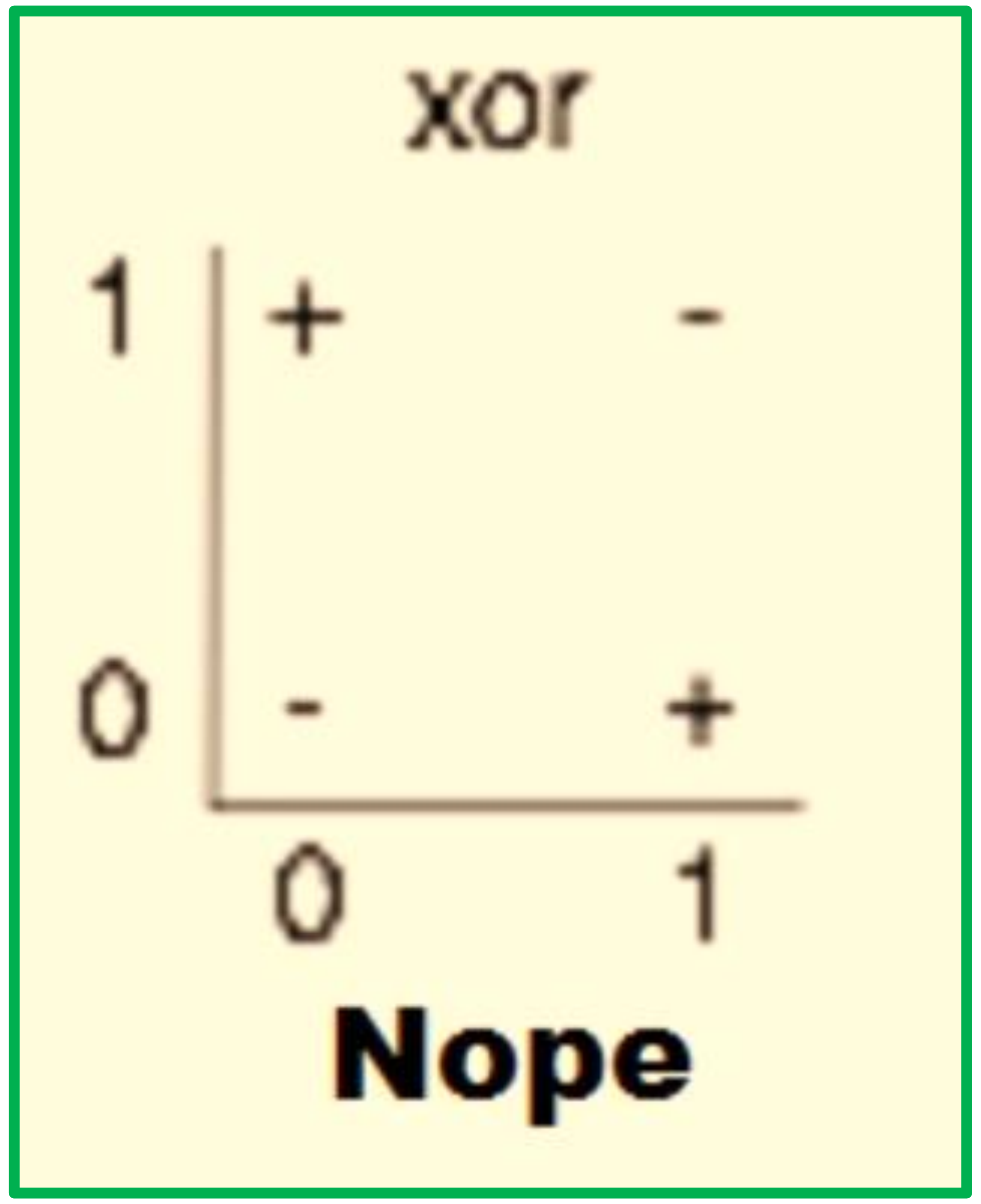
Neural Network (NN) : "No one on earth had found a viable way to train"

[Marvin Minsky]



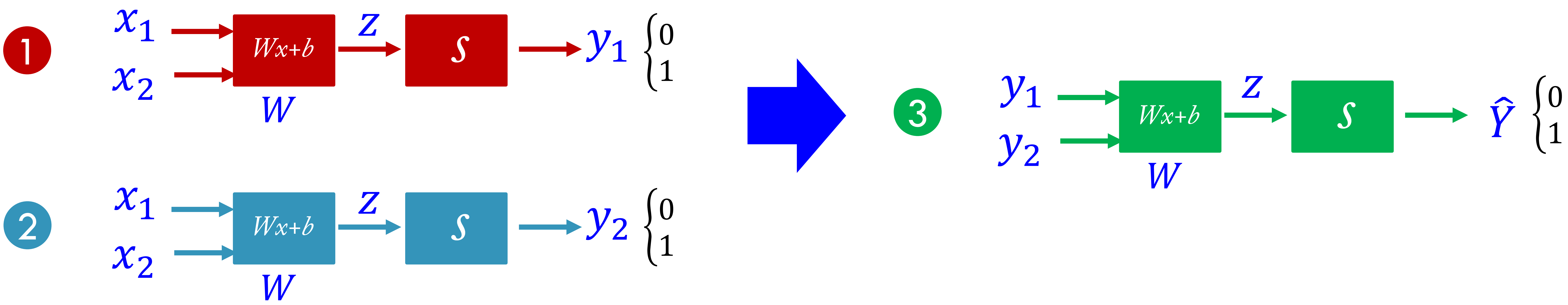
XOR using NN

X_1	X_2	XOR
0	0	0 (-)
0	1	1 (+)
1	0	1 (+)
1	1	0 (-)



Neural Net

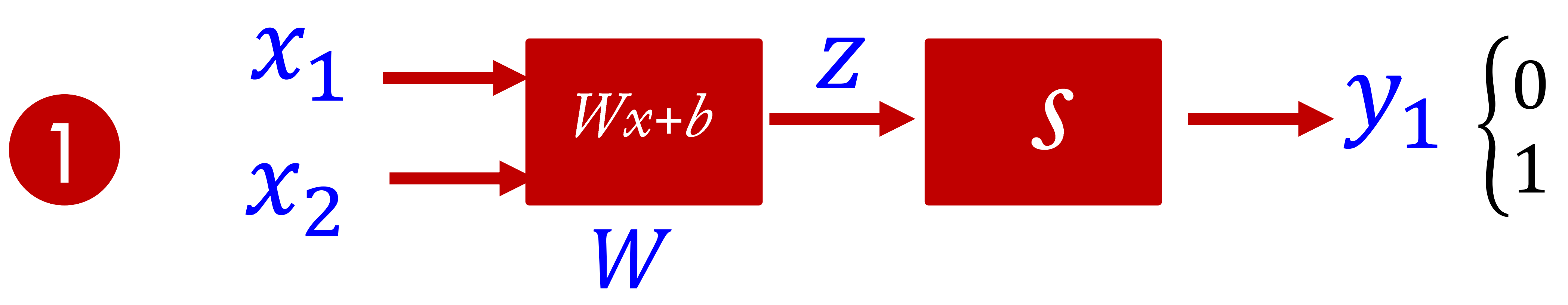
$$H(x) = Wx + b$$



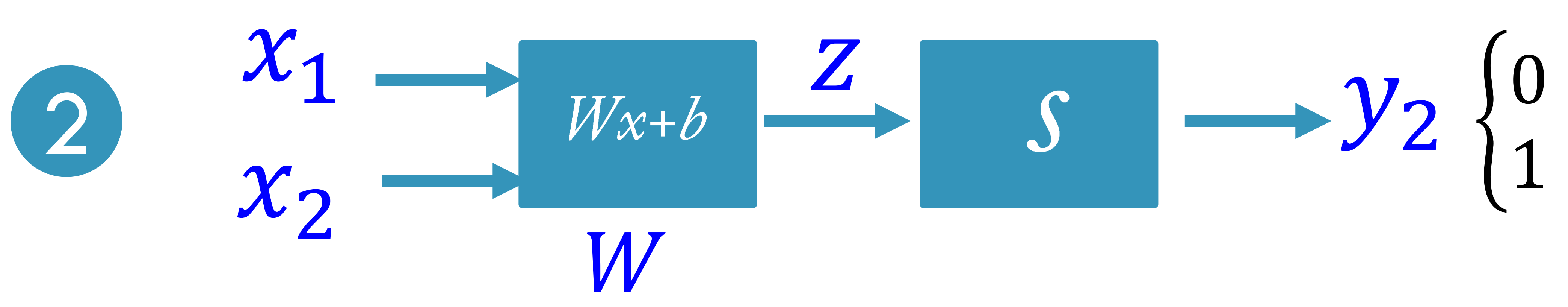
Neural Net

$$H(x) = Wx + b$$

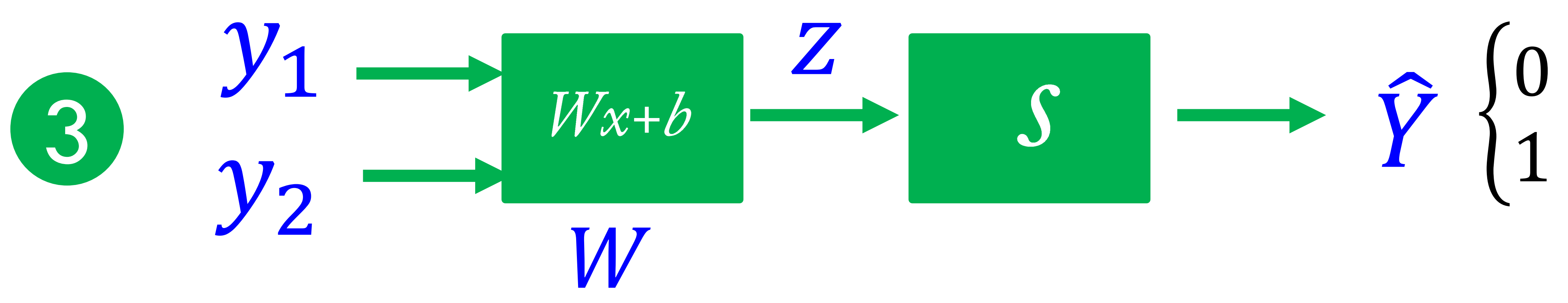
if) $w = \begin{bmatrix} 5 \\ 5 \end{bmatrix}, b = -8$



if) $w = \begin{bmatrix} -7 \\ -7 \end{bmatrix}, b = 3$



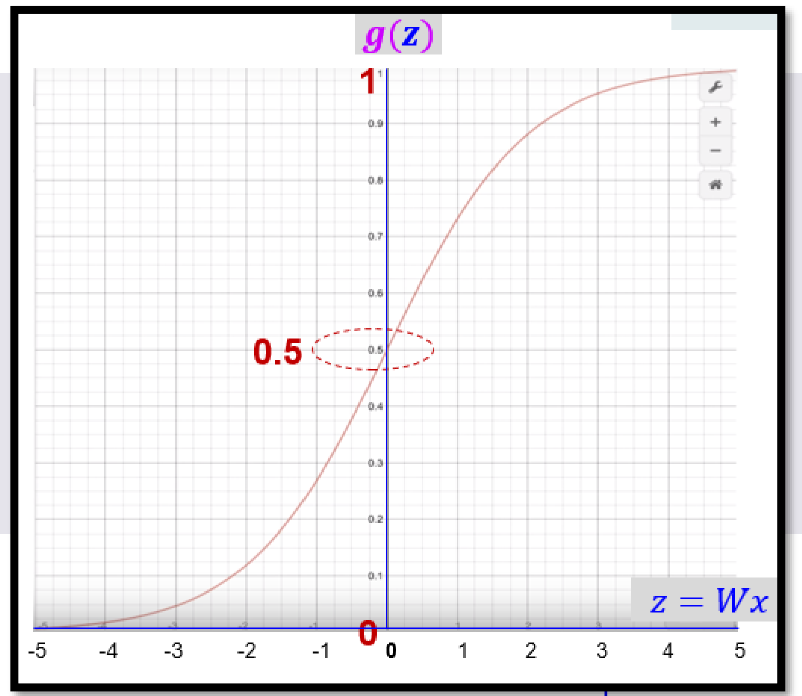
if) $w = \begin{bmatrix} -11 \\ -11 \end{bmatrix}, b = 6$



Neural Net

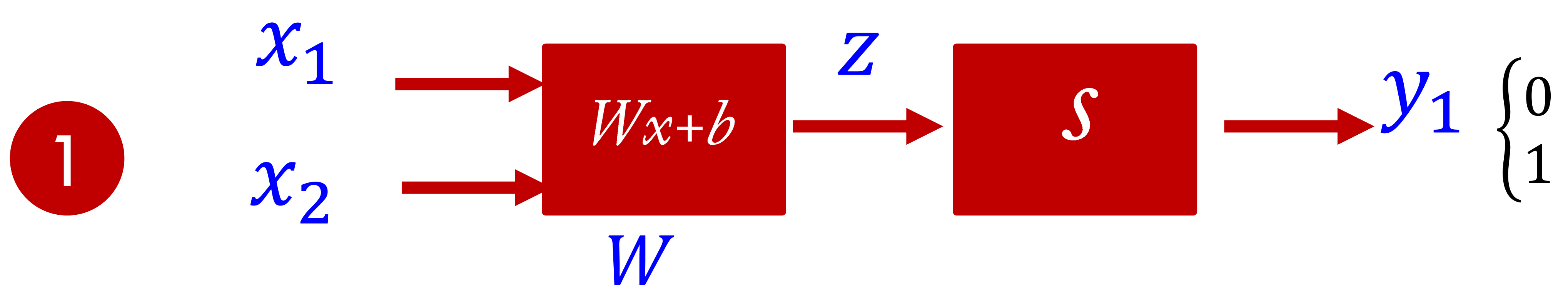
X_1	X_2	XOR
0	0	0 (-)
0	1	1 (+)
1	0	1 (+)
1	1	0 (-)

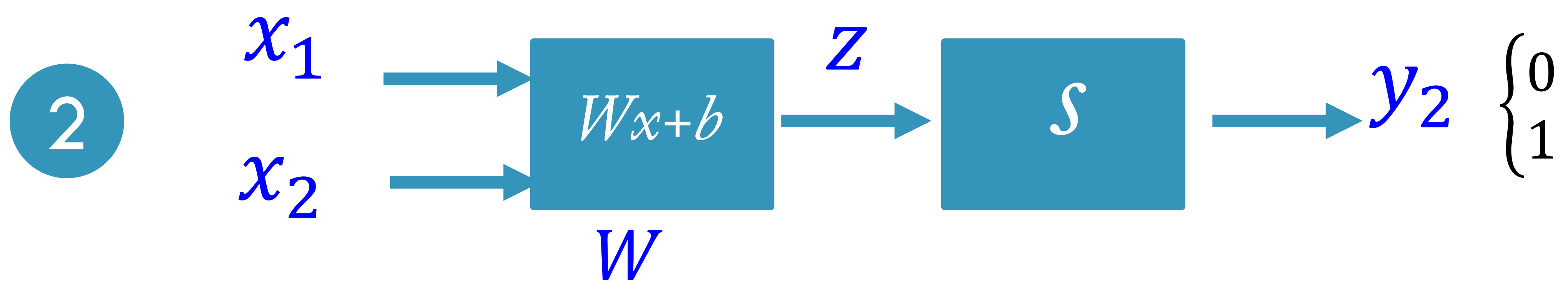
$H(x) = Wx + b$



W=(5, 5), b=-8 일때
(0*5 + 0*5) + (-8) = -8 ==> sigmoid(-8) = 0
(0*5 + 1*5) + (-8) = -3 ==> sigmoid(-3) = 0
(1*5 + 0*5) + (-8) = -3 ==> sigmoid(-3) = 0
(1*5 + 1*5) + (-8) = 2 ==> sigmoid(2) = 1

$$w = \begin{bmatrix} 5 \\ 5 \end{bmatrix}, b = -8$$



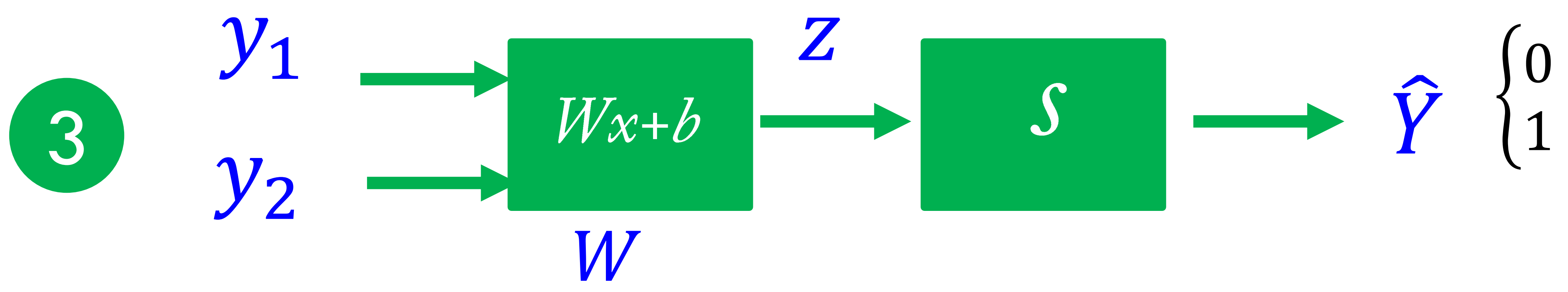


W=(-7, -7), b=3 일때,
(0*-7 + 0*-7) + 3 = 3   ==> sigmoid(3) = 1
(0*-7 + 1*-7) + 3 = -4   ==> sigmoid(-4) = 0
(1*-7 + 0*-7) + 3 = -4   ==> sigmoid(-4) = 0
(1*-7 + 1*-7) + 3 = -11 ==> sigmoid(-11) = 0

$$w = \begin{bmatrix} -7 \\ -7 \end{bmatrix}, b = 3$$

$$w = \begin{bmatrix} -11 \\ -11 \end{bmatrix}, b = 6$$

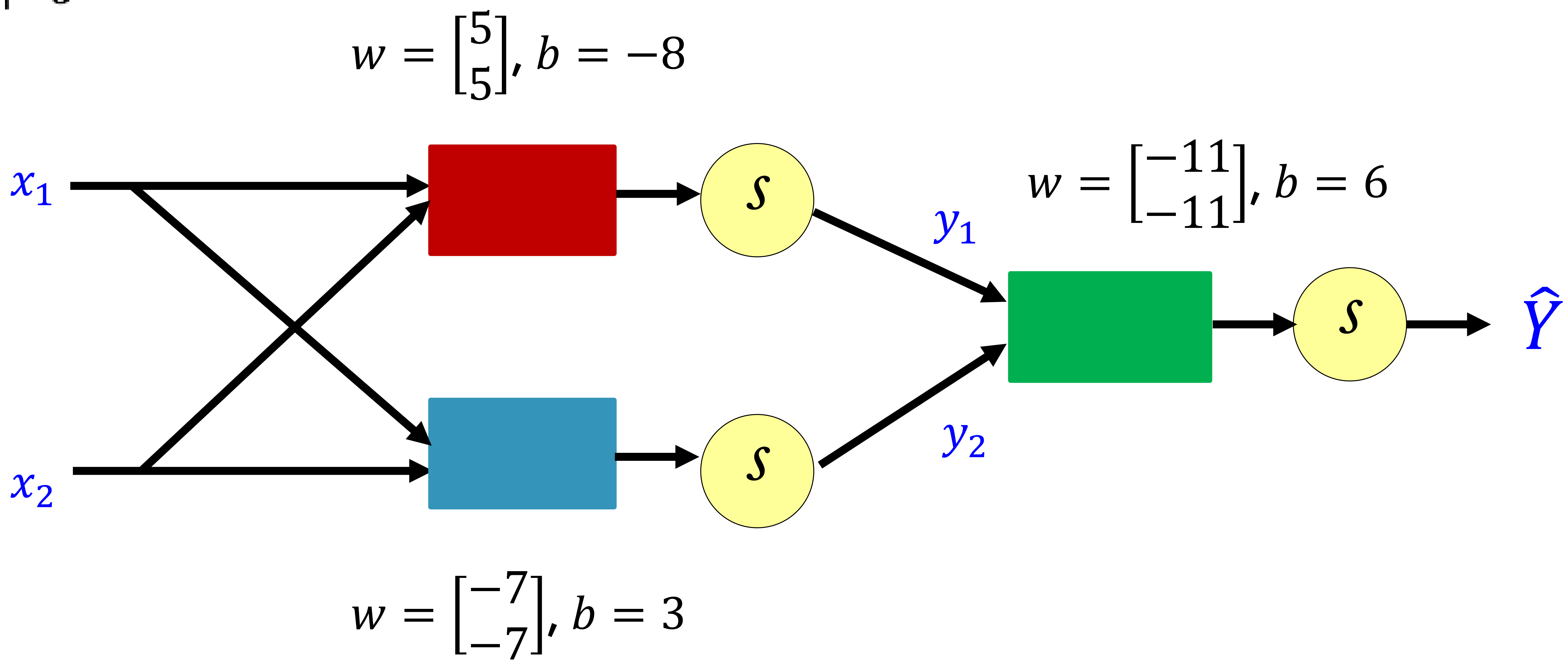
W=(-11, -11), b=6 일때
(0*-11 + 1*-11) + 6 = -11 ==> sigmoid(-5) = 0
(0*-11 + 0*-11) + 6 = 6 ==> sigmoid(6) = 1
(0*-11 + 0*-11) + 6 = 6 ==> sigmoid(6) = 1
(1*-11 + 0*-11) + 6 = -11 ==> sigmoid(-5) = 0



X_1	X_2	y1	y2	$\hat{Y}$	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 (-)

Forward propagation

$$H(x) = Wx + b$$

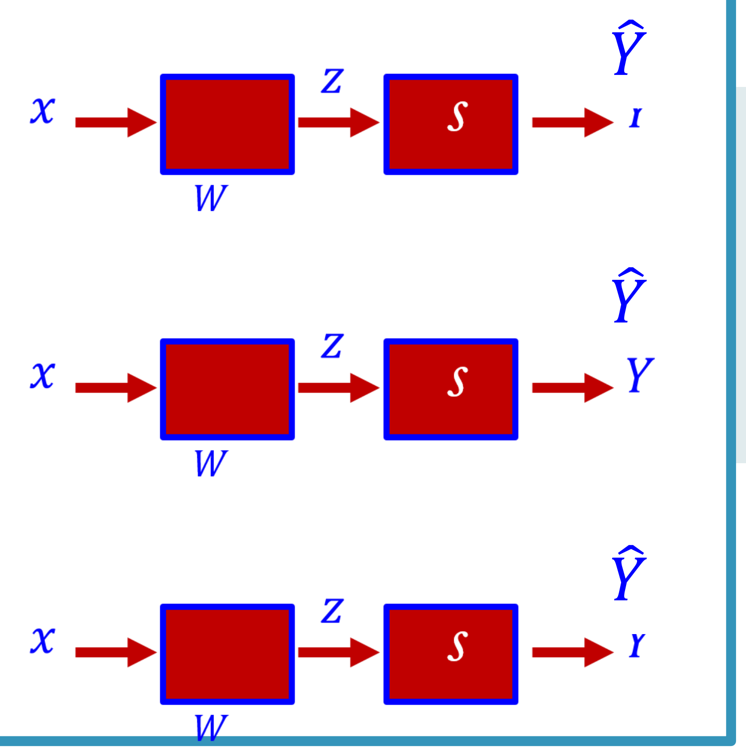


Can you find another W and b for the XOR?

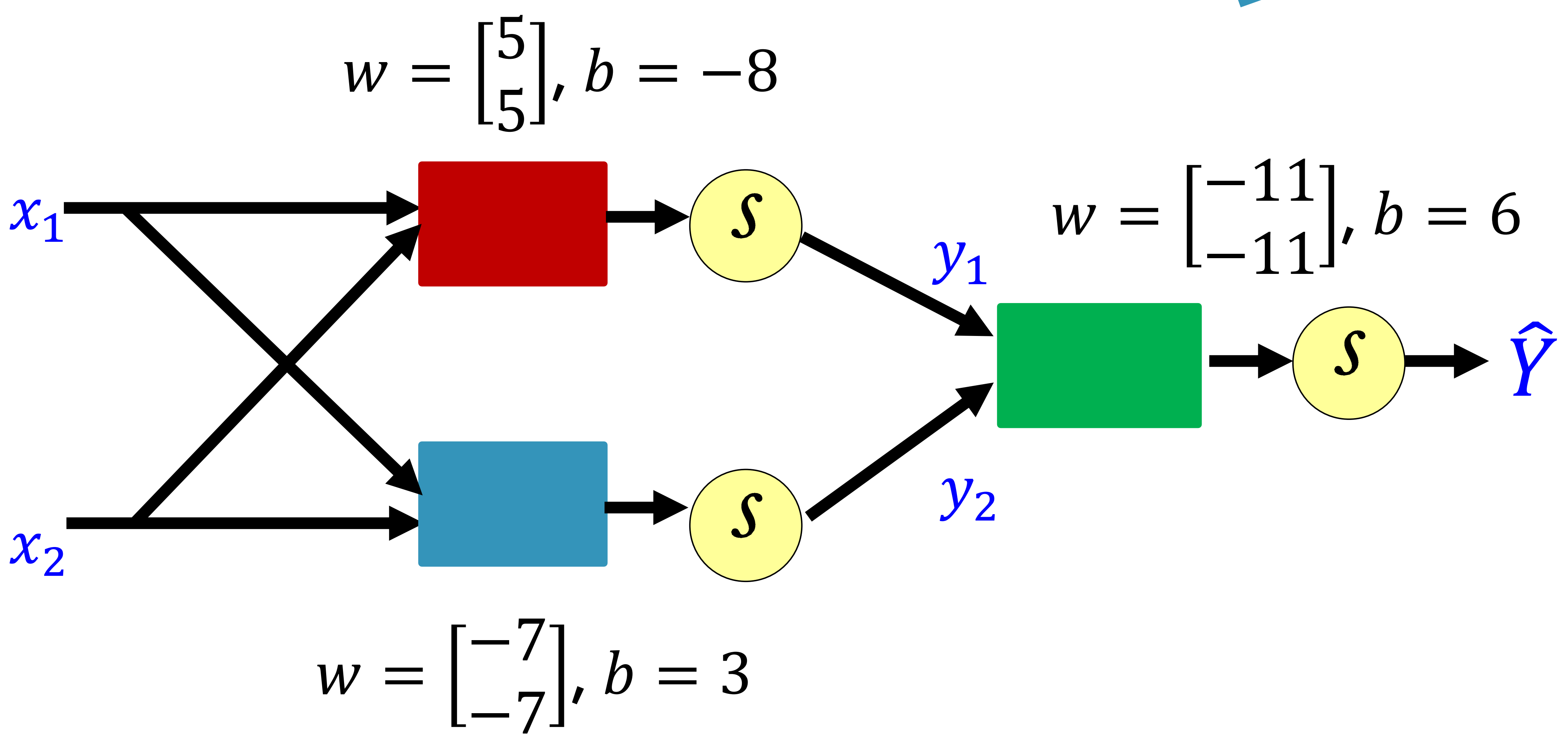
Neural Network (NN)

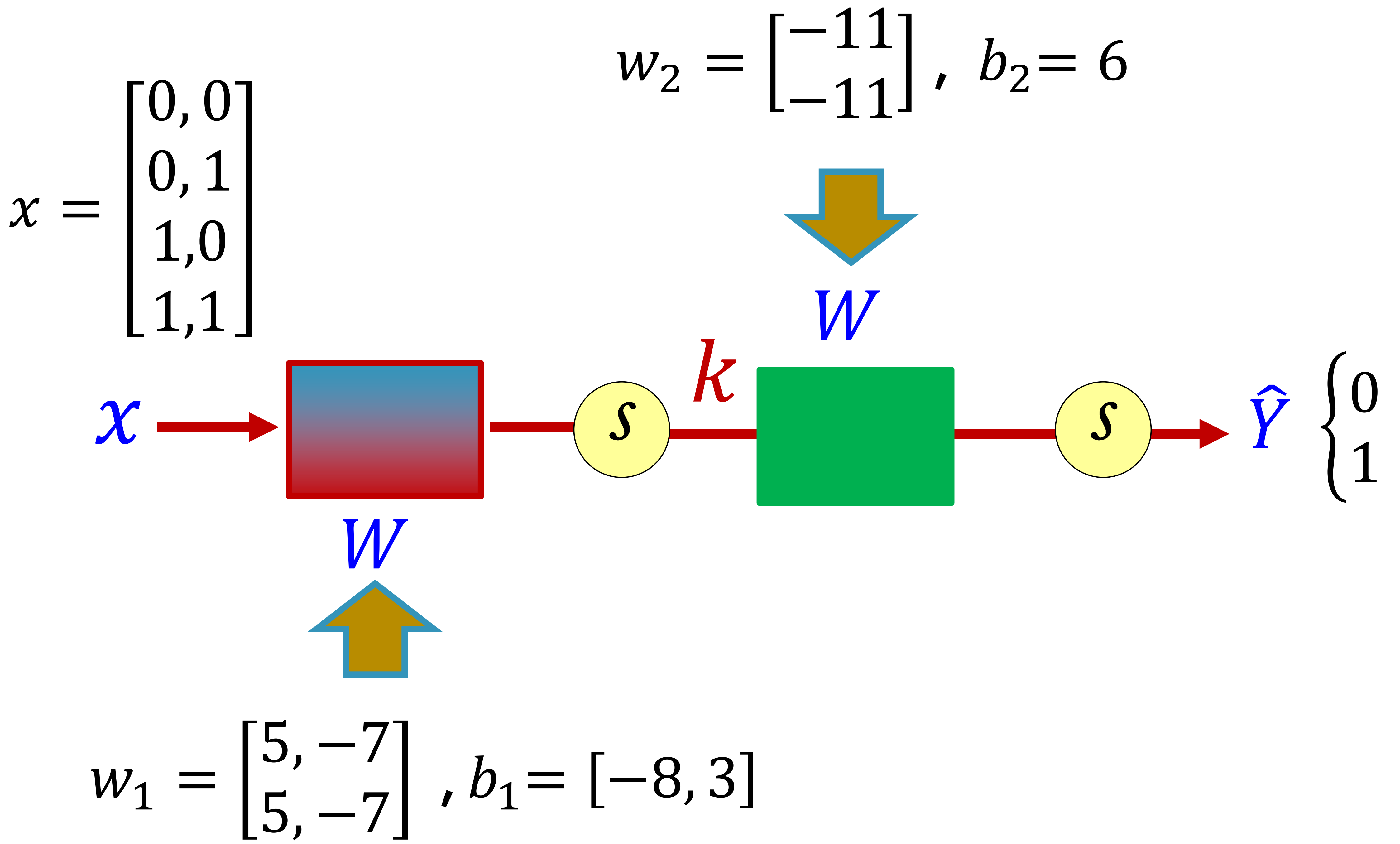
Multinomial classification

$$\begin{bmatrix} w_{A1} & w_{A2} & w_{A3} \\ w_{B1} & w_{B2} & w_{B3} \\ w_{C1} & w_{C2} & w_{C3} \end{bmatrix} \begin{bmatrix} x1 \\ x2 \\ x3 \end{bmatrix} = \begin{bmatrix} w_{A1}\,x1 + w_{A2}\,x2 + w_{A3}\,x3 \\ w_{B1}\,x1 + w_{B2}\,x2 + w_{B3}\,x3 \\ w_{C1}\,x1 + w_{C2}\,x2 + w_{C3}\,x3 \end{bmatrix} = \begin{bmatrix} \hat{Y}_A \\ \hat{Y}_B \\ \hat{Y}_C \end{bmatrix} \begin{matrix} H_A(x) \\ H_B(x) \\ H_C(x) \end{matrix}$$



$$H(x) = Wx + b$$

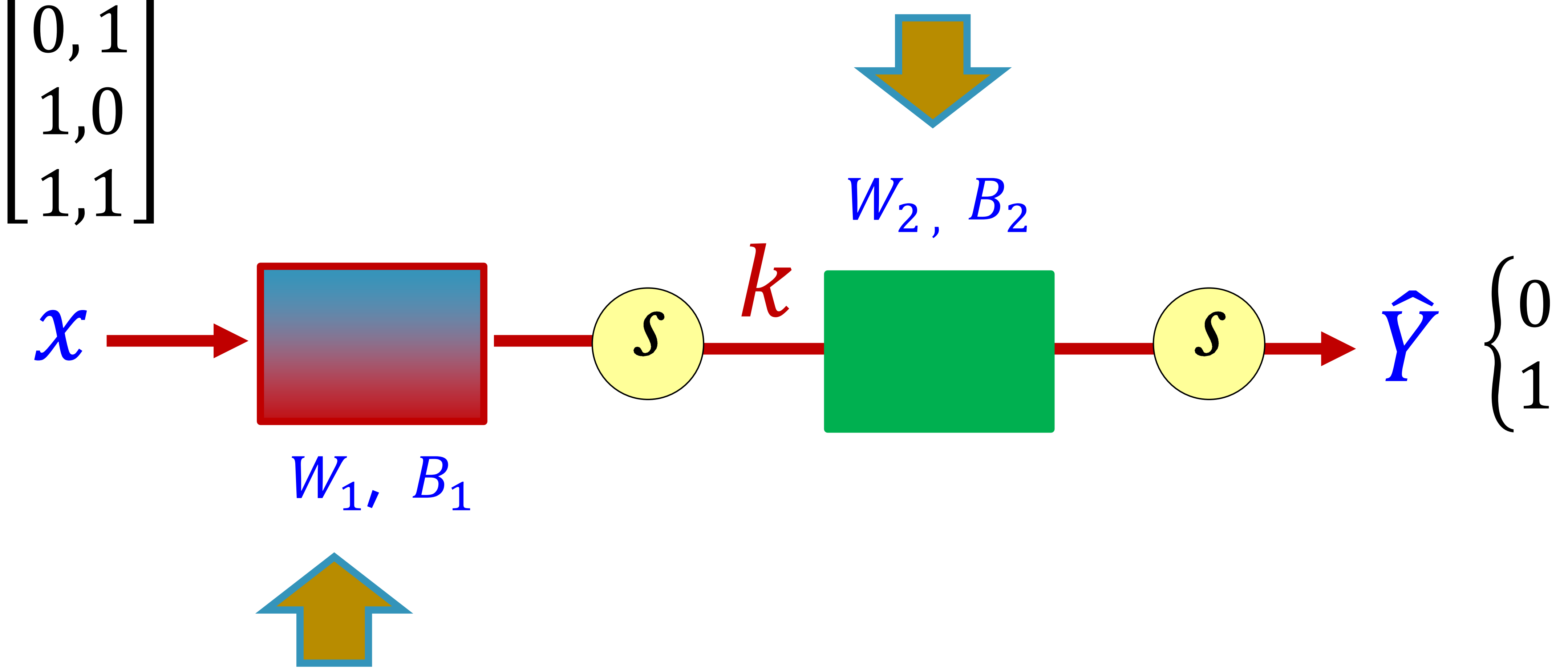




Neural Network (NN)

$$W_2 = \begin{bmatrix} -11 \\ -11 \end{bmatrix},\ B_2 = 6$$

$$x = \begin{bmatrix} 0,0 \\ 0,1 \\ 1,0 \\ 1,1 \end{bmatrix}$$



$$W_1 = \begin{bmatrix} 5,-7 \\ 5,-7 \end{bmatrix},\ B_1 = [-8, 3]$$

$$H(x) = Wx + b$$

$$k(x) = sigmoid(x \cdot W_1 + B_1)$$

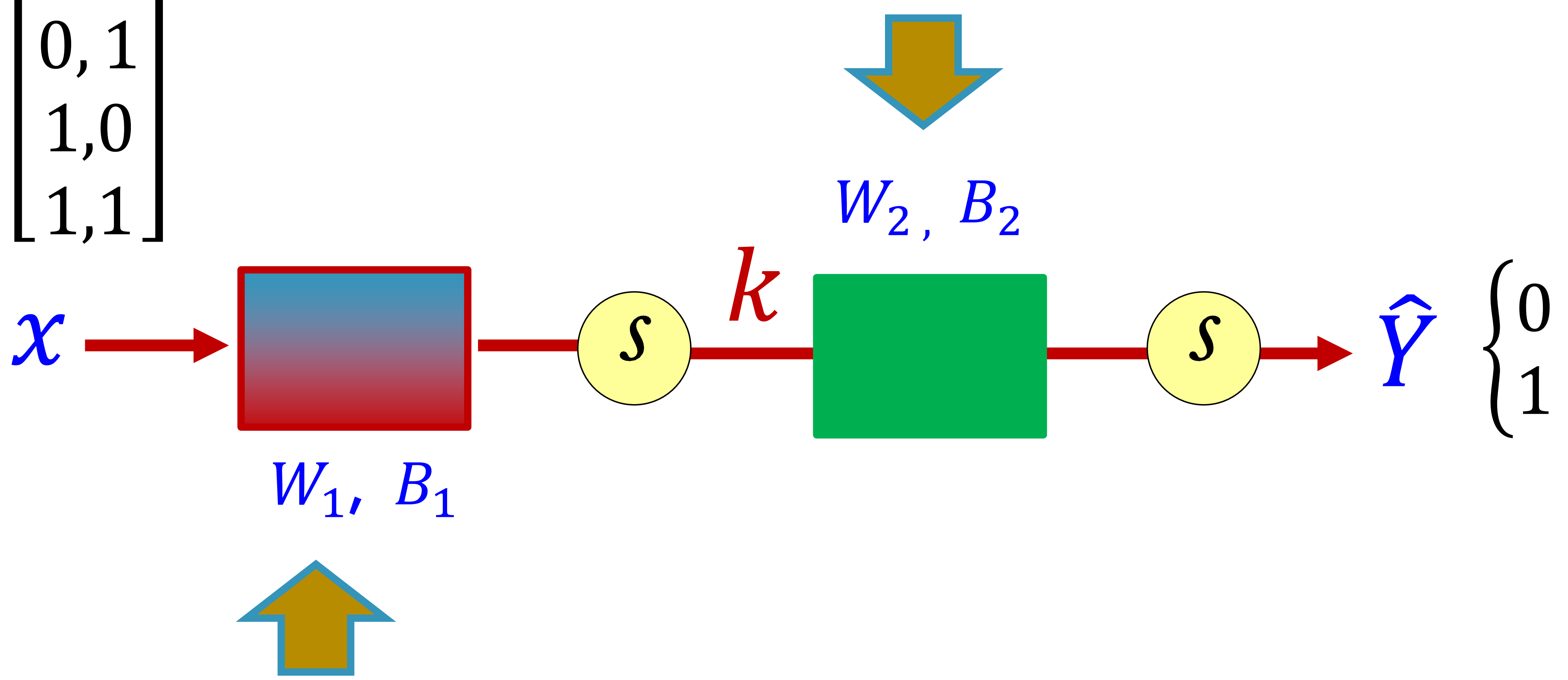
$$\hat{Y} = H(x) = Sigmoid(k(x) \cdot W_2 + B_2)$$

```
# NN
K = tf.sigmoid(tf.matmul(X, W1) + b1)
hypothesis = tf.sigmoid(tf.matmul(K, W2) + b2)
```

Neural Network (NN)

$W_2 = \begin{bmatrix} -11 \\ -11 \end{bmatrix},\ B_2 = 6$

$x = \begin{bmatrix} 0,0 \\ 0,1 \\ 1,0 \\ 1,1 \end{bmatrix}$



$W_1 = \begin{bmatrix} 5,-7 \\ 5,-7 \end{bmatrix},\ B_1 = [-8, 3]$

$$H(x) = Wx + b$$

$$k(x) = sigmoid(x \cdot W_1 + B_1)$$

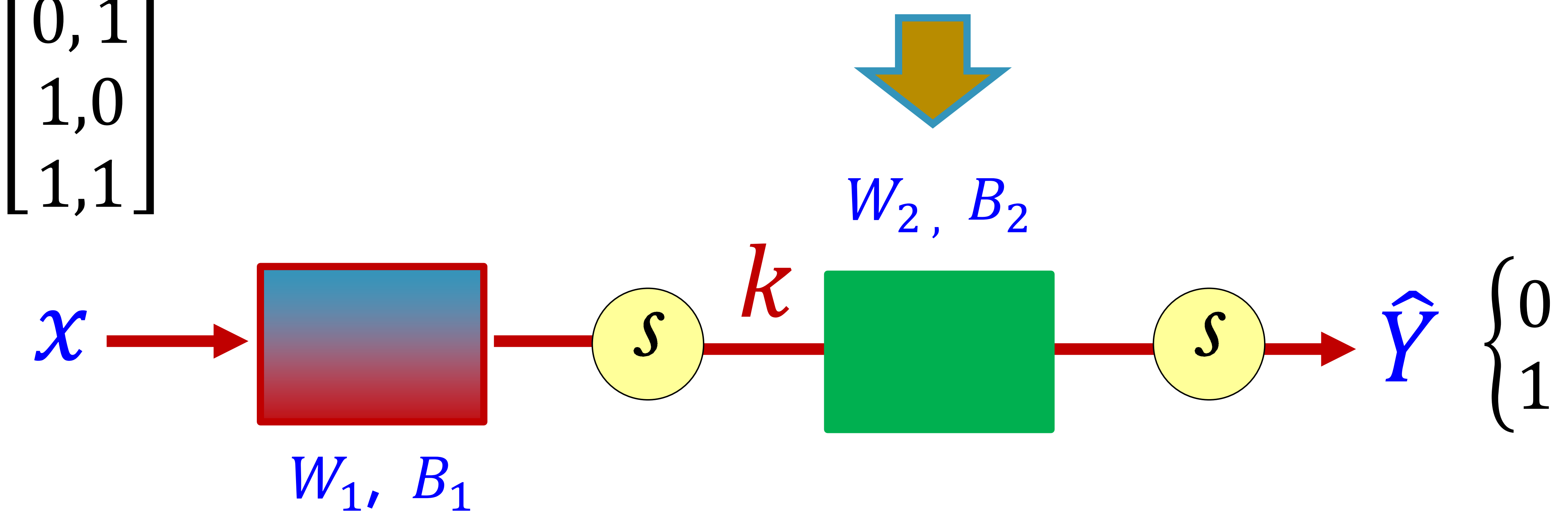
$$\hat{Y} = H(x) = Sigmoid(k(x) \cdot W_2 + B_2)$$

How can we learn W_1, W_2, B_1, b_2 from training data?

Neural Network (NN)

$$W_2 = \begin{bmatrix} -11 \\ -11 \end{bmatrix},\ B_2 = 6$$

$$x = \begin{bmatrix} 0, 0 \\ 0, 1 \\ 1, 0 \\ 1, 1 \end{bmatrix}$$



$$W_1 = \begin{bmatrix} 5, -7 \\ 5, -7 \end{bmatrix},\ B_1 = [-8, 3]$$

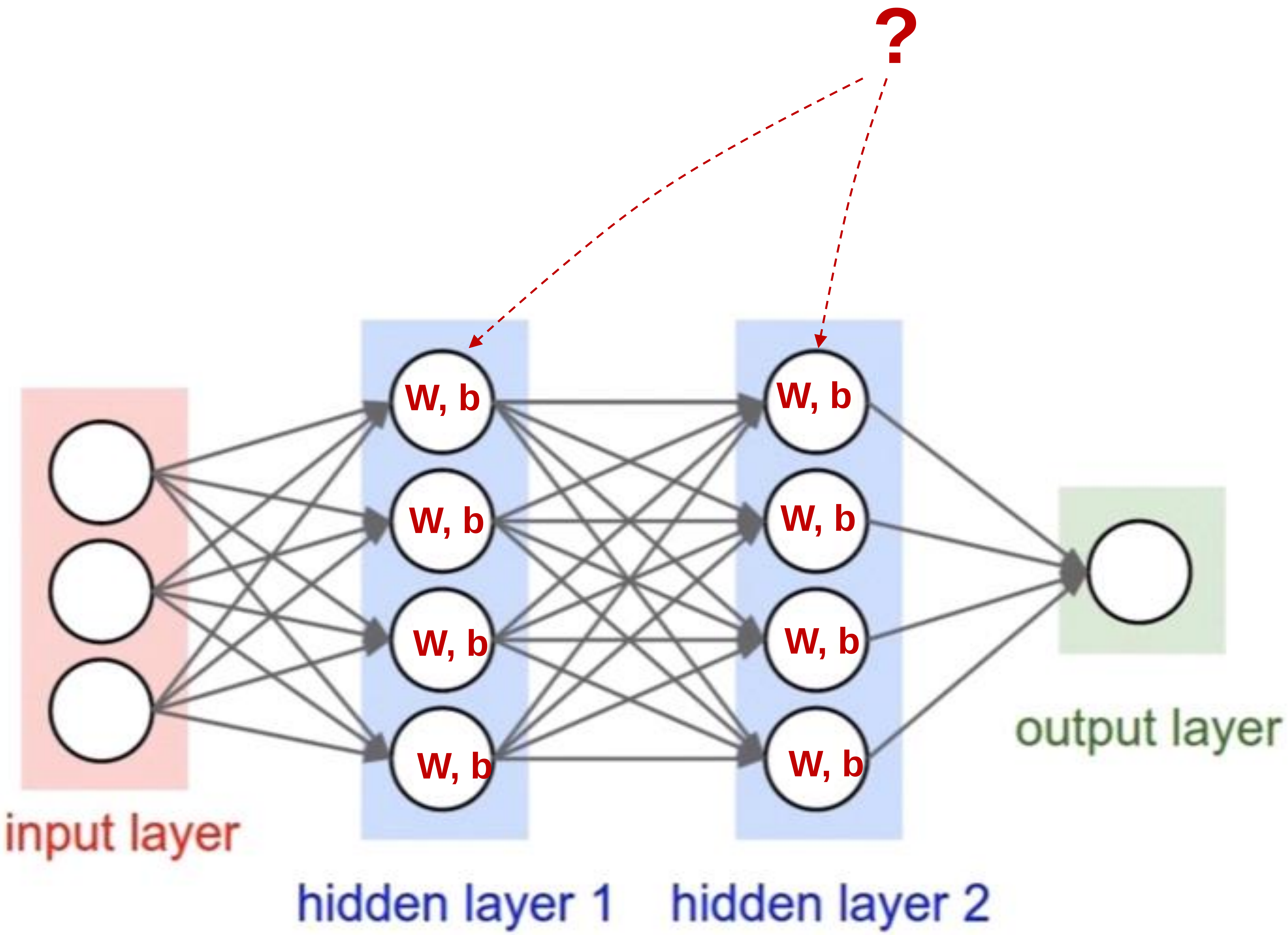
$$H(x) = Wx + b$$

$$k(x) = sigmoid(x \cdot W_1 + B_1)$$

$$\hat{Y} = H(x) = Sigmoid(k(x) \cdot W_2 + B_2)$$

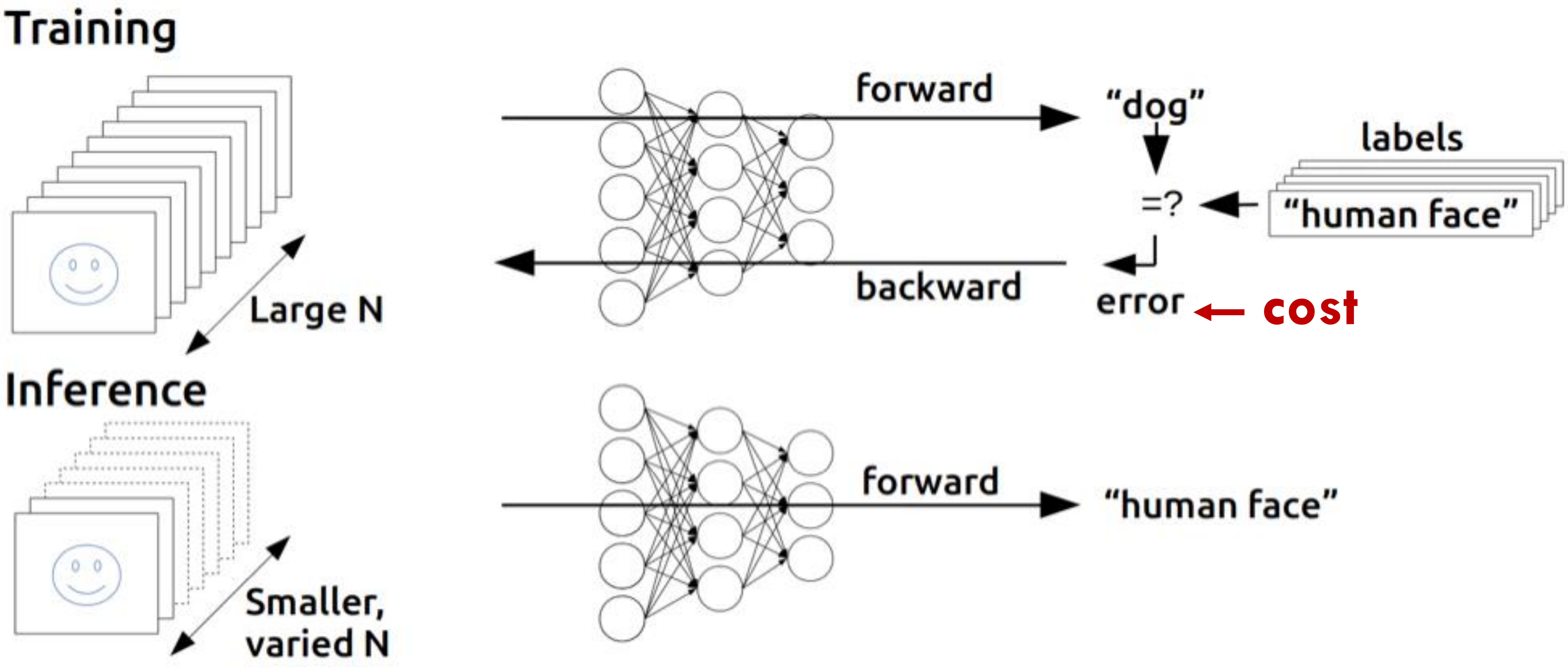
How can we learn W1, W2, B1, b2 from training data? ***Gradient Descent***

Derivation



?
W, b
W, b
W, b
W, b
W, b
W, b
W, b
W, b
input layer
hidden layer 1
hidden layer 2
output layer

Backpropagation (1974, 1982 by Paul Werbos, 1986 by Hinton)



Basic Derivative

$$\frac{d}{dx}f(x) = \lim_{\Delta x \to 0} \frac{f(x+\Delta x) - f(x)}{\Delta x}$$

$= \frac{df}{dx}$

if) $\Delta x = 0.01$

C $f(x) = 3 \rightarrow \frac{f(x+0.01)-f(x)}{0.01} = \frac{3-3}{0.01} = \frac{0}{0.01} = 0$

$x : 1 \rightarrow 3$
$x : 2 \rightarrow 3$

V $f(x) = x \rightarrow \frac{f(x+0.01)-f(x)}{0.01} = \frac{\cancel{x}+0.01-\cancel{x}}{0.01} = \frac{0.01}{0.01} = 1$

$x : 1 \rightarrow f(x) : 1$
$x : 2 \rightarrow f(x) : 2$

C V $f(x) = 2x \rightarrow \frac{f(x+0.01)-f(x)}{0.01} = \frac{2(x+0.01)-2x}{0.01} = \frac{\cancel{2x}+(2*0.01)-\cancel{2x}}{0.01} = \frac{2*\cancel{0.01}}{\cancel{0.01}} = 2$

$x : 1 \rightarrow f(x) : 2$

$f(x) = 3x \rightarrow f(x) : 3$
$f(x) = 4x \rightarrow f(x) : 4$

V + C $f(x) = x + 3 \rightarrow \frac{f(x+0.01)-f(x)}{0.01} = \frac{(\cancel{x}+0.01+\cancel{3})-(\cancel{x}+\cancel{3})}{0.01} = \frac{0.01}{0.01} = 1$

Partial Derivative : consider other variables as constants

C V $f(x) = 2x \rightarrow \frac{f(x+0.01)-f(x)}{0.01} = \frac{2(x+0.01)-2x}{0.01} = \frac{2x+(2*0.01)-2x}{0.01} = \frac{2*0.01}{0.01} = 2$

$x : 1 \rightarrow f(x) : 2$

$f(x) = 3x \rightarrow f(x) : 3$
$f(x) = 4x \rightarrow f(x) : 4$

상수

V C $f(x,y) = x \cdot y, \frac{\partial f}{\partial x} \rightarrow$ **Partial Derivative** $\rightarrow 1 \cdot y = y$

상수

C V $f(x,y) = x \cdot y, \frac{\partial f}{\partial y} \rightarrow$ **Partial Derivative** $\rightarrow x \cdot 1 = x$

Partial Derivative : consider other variables as constants

v + c $\quad f(x) = x + 3 \rightarrow \dfrac{f(x+0.01)-f(x)}{0.01} = \dfrac{(\cancel{x}+0.01+\cancel{3})-(\cancel{x}+\cancel{3})}{0.01} = \dfrac{0.01}{0.01} = 1$

상수

v + c $\quad f(x,y) = x + \boldsymbol{y}, \dfrac{\partial f}{\partial x}$ → **Partial Derivative** → $\mathbf{1 + 0 = 1}$

상수

c + v $\quad f(x,y) = \boldsymbol{x} + y, \dfrac{\partial f}{\partial y}$ → **Partial Derivative** → $\mathbf{0 + 1 = 1}$

Chain rule

$$z = W\mathbf{x} + b$$

$$H = sigmoid(z)$$

$$H(x) = Wx + b$$

$$\boldsymbol{k(x)} = \boldsymbol{sigmoid}(x \cdot W_1 + B_1)$$

$$\hat{Y} = H(x) = \boldsymbol{Sigmoid}(\boldsymbol{k(x)} \cdot W_2 + B_2)$$

```
# NN
K = tf.sigmoid(tf.matmul(X, W1) + b1)
hypothesis = tf.sigmoid(tf.matmul(K, W2) + b2)
```

Chain rule

$$f(\boldsymbol{g}(\boldsymbol{x}))$$

$$\frac{\partial f}{\partial \boldsymbol{x}} = \frac{\partial f}{\partial g} \cdot \frac{\partial g}{\partial x}$$

Complex function

Back propagation (chain rule)

$$\boxed{f = wx + b}$$

$$g = wx$$

상수

$$w \cdot x, \frac{\partial g}{\partial w} = 1 \cdot x = x$$

상수

$$w \cdot x, \frac{\partial g}{\partial x} = w \cdot 1 = w$$

$$f = g + b$$

상수

$$g + b, \frac{\partial f}{\partial b} \rightarrow 0 + 1 = 1$$

상수

$$g + b, \frac{\partial f}{\partial g} \rightarrow 1 + 0 = 1$$

Back propagation (chain rule)

$f = wx + b$ ➡ $g = wx$ ➡ $f = g + b$

$$\frac{\partial g}{\partial w} = x, \frac{\partial g}{\partial x} = w$$

$$\frac{\partial f}{\partial g} = 1, \frac{\partial f}{\partial b} = 1$$

if

1 Forward $(w = -2, x = 5, b = 3)$

2 Backward

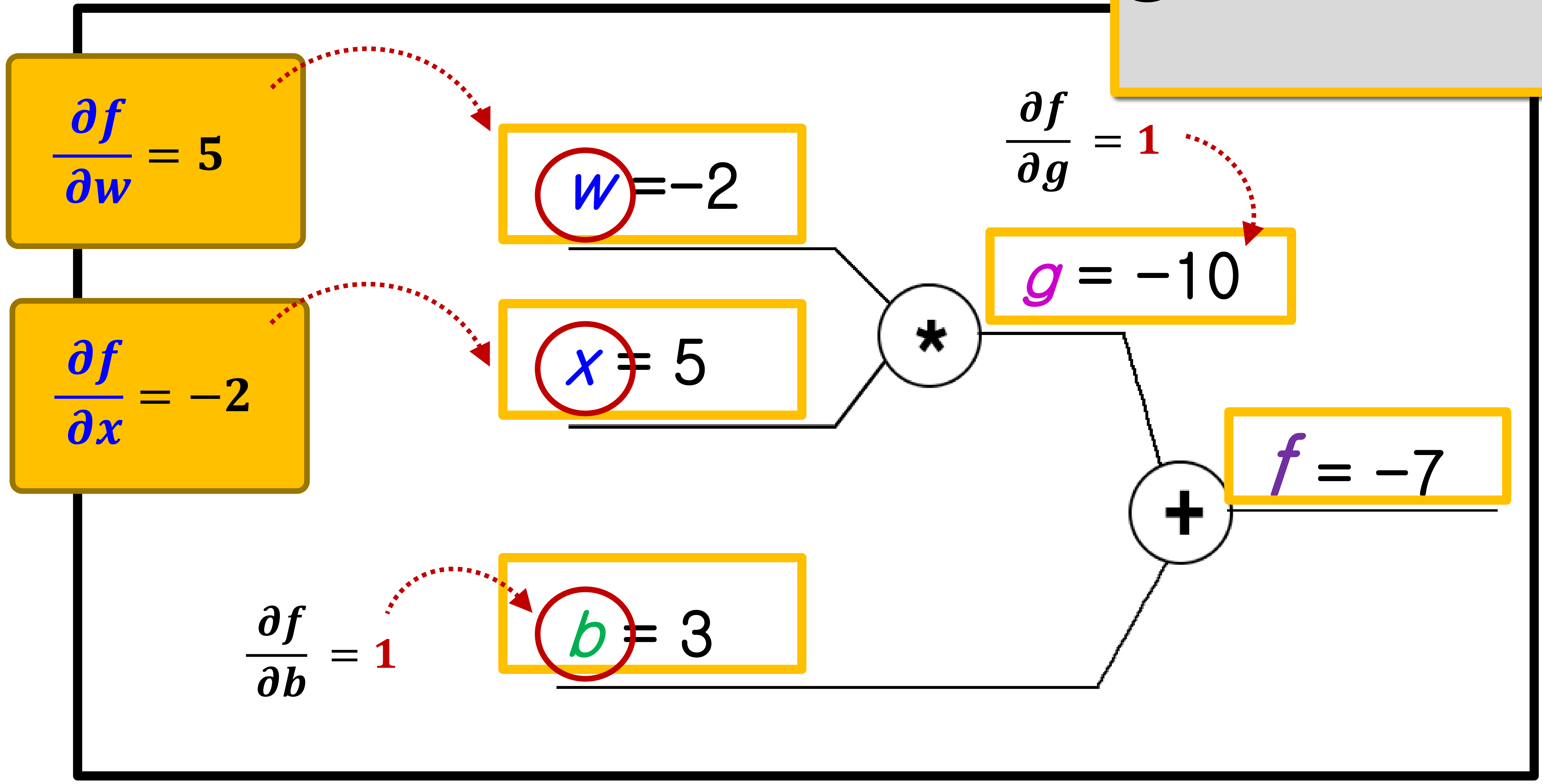
Chain rule

$$f(g(x))$$

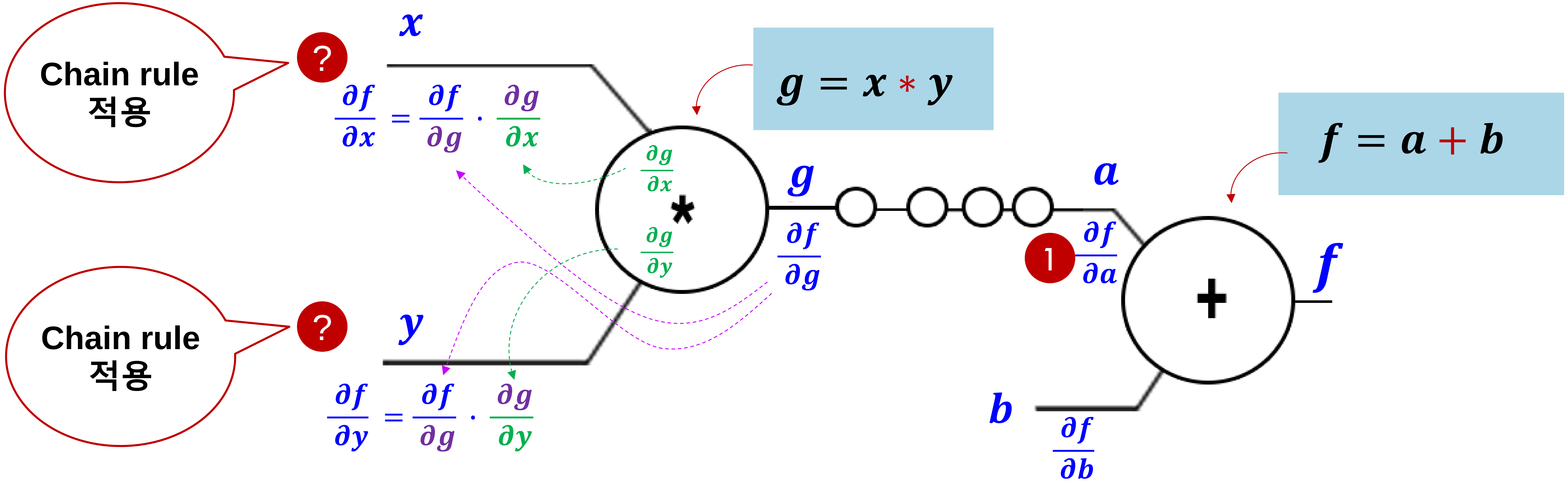
$$\frac{\partial f}{\partial x} = \frac{\partial f}{\partial g} \cdot \frac{\partial g}{\partial x}$$

$$\frac{\partial f}{\partial w} = \frac{\partial f}{\partial g} \cdot \frac{\partial g}{\partial w} = 1 \cdot x = 1 \cdot 5 = 5$$

$$\frac{\partial f}{\partial x} = \frac{\partial f}{\partial g} \cdot \frac{\partial g}{\partial x} = 1 \cdot w = 1 \cdot -2 = -2$$



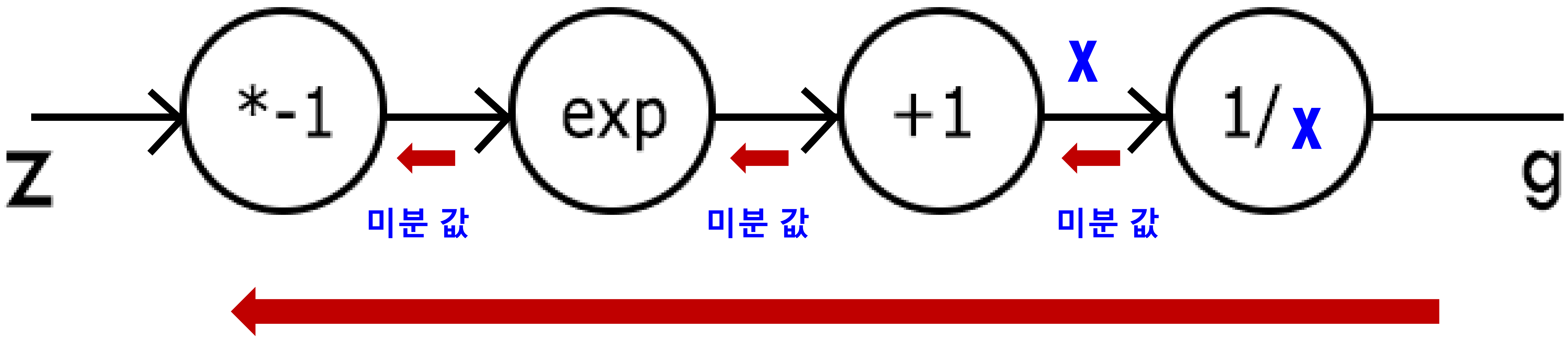
Back propagation (chain rule)



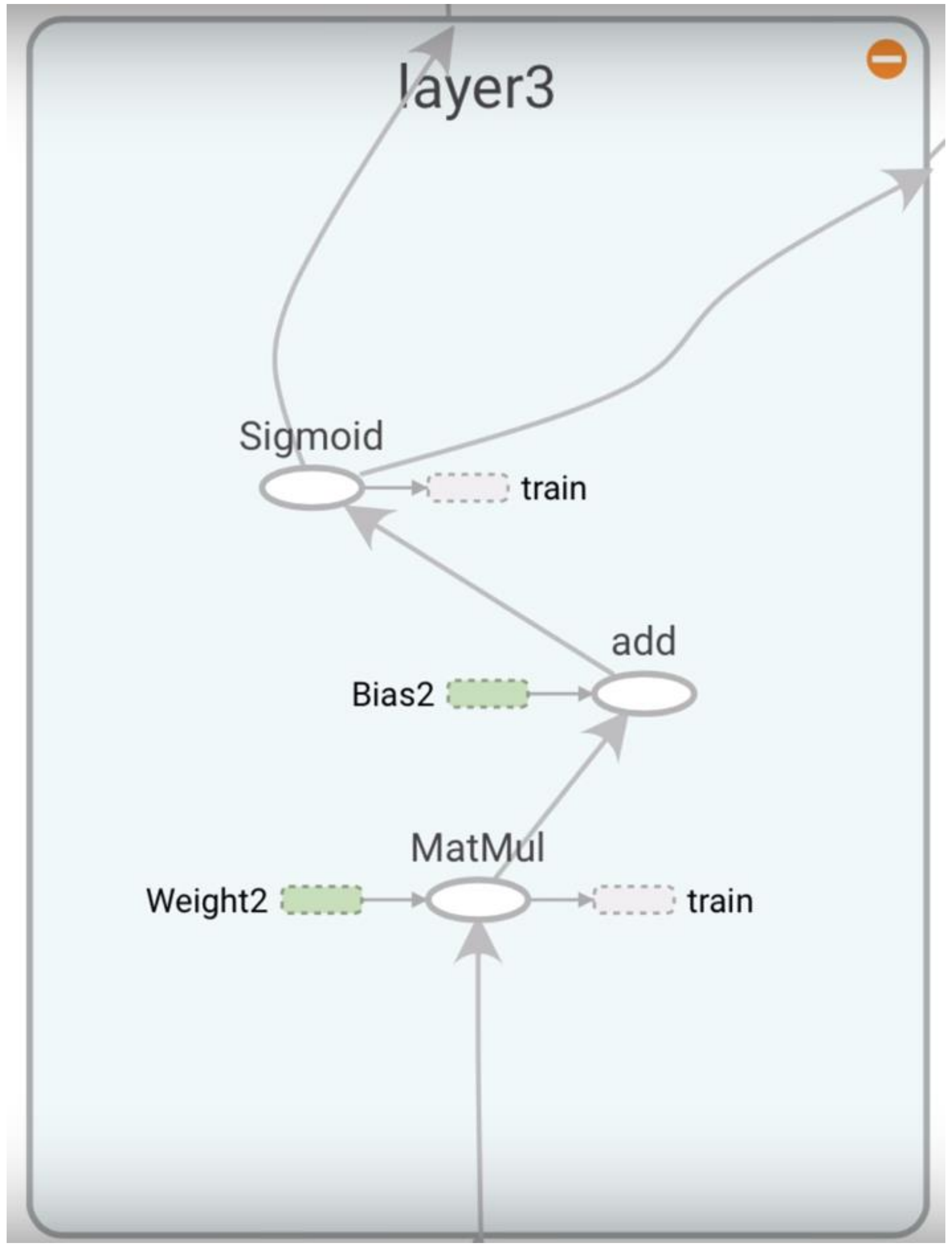
Back propagation (chain rule) : Sigmoid

$$g(z) = \frac{1}{\boxed{1 + e^{-z}}} \Rightarrow \text{?}\ \frac{\partial g}{\partial z}$$

x

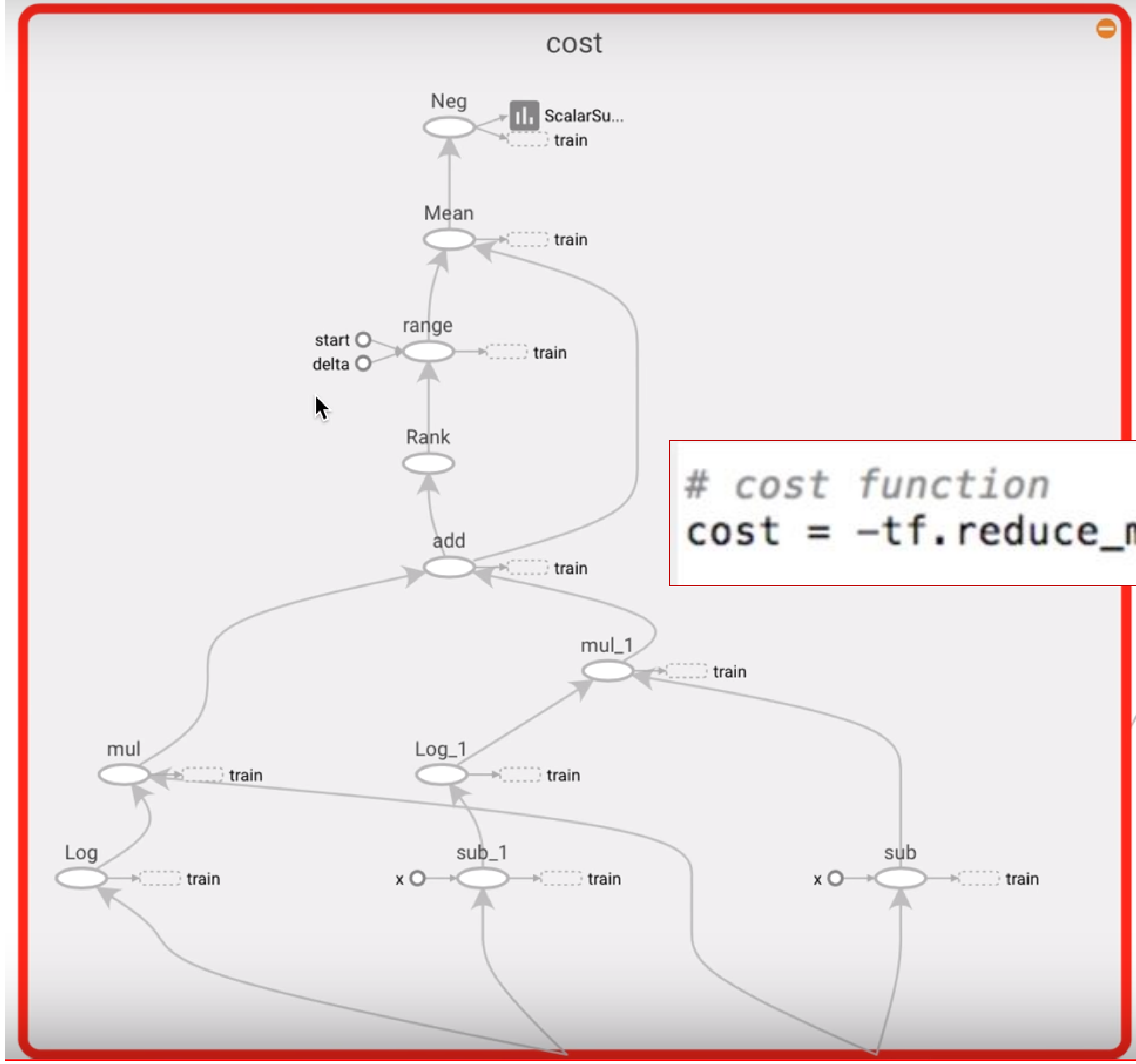


Back propagation in TensorFlow → TensorBoard




```
hypothesis = tf.sigmoid(tf.matmul(L2, W2) + b2)
```

Back propagation in TensorFlow → TensorBoard

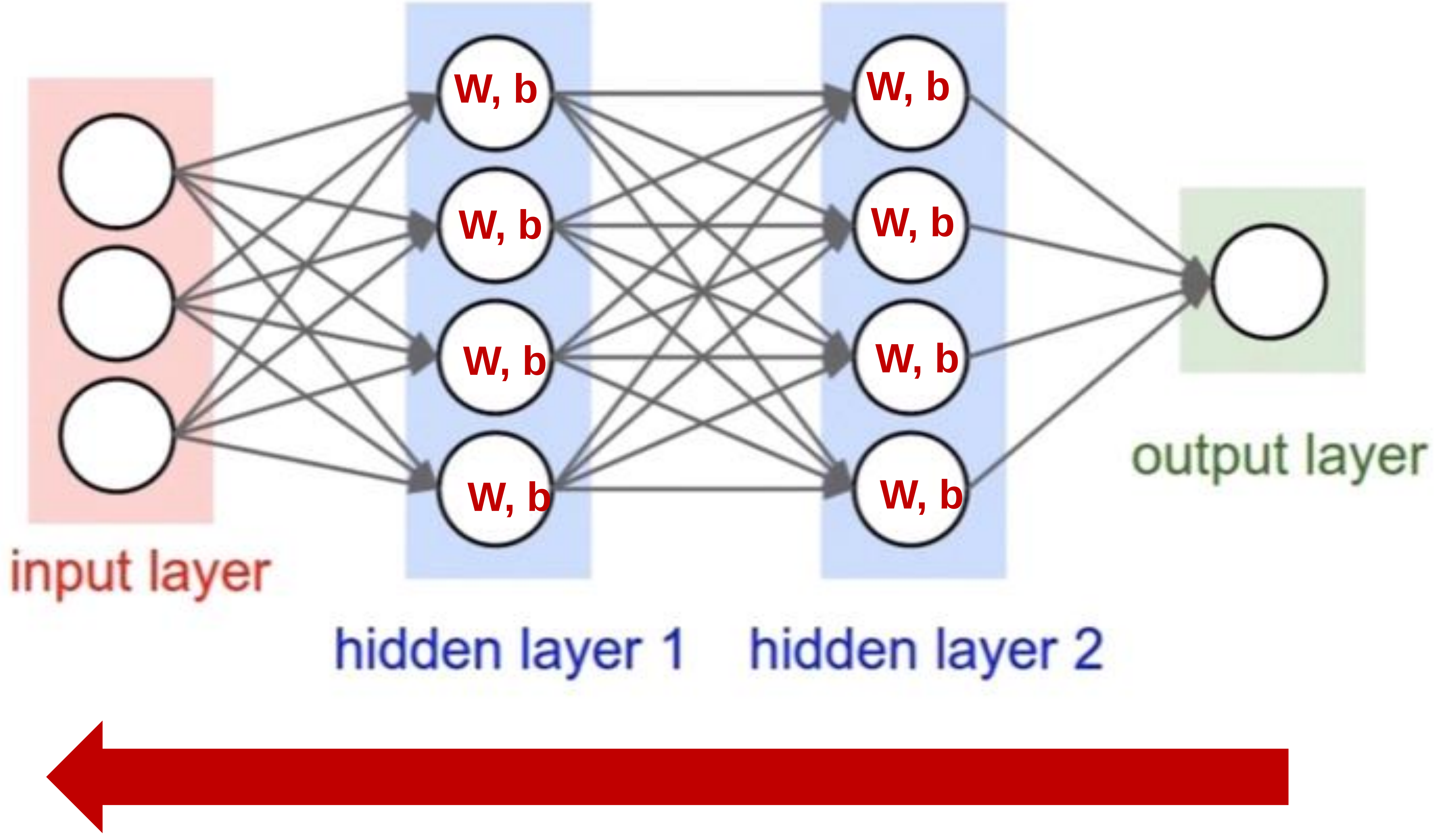


Back propagation in TensorFlow

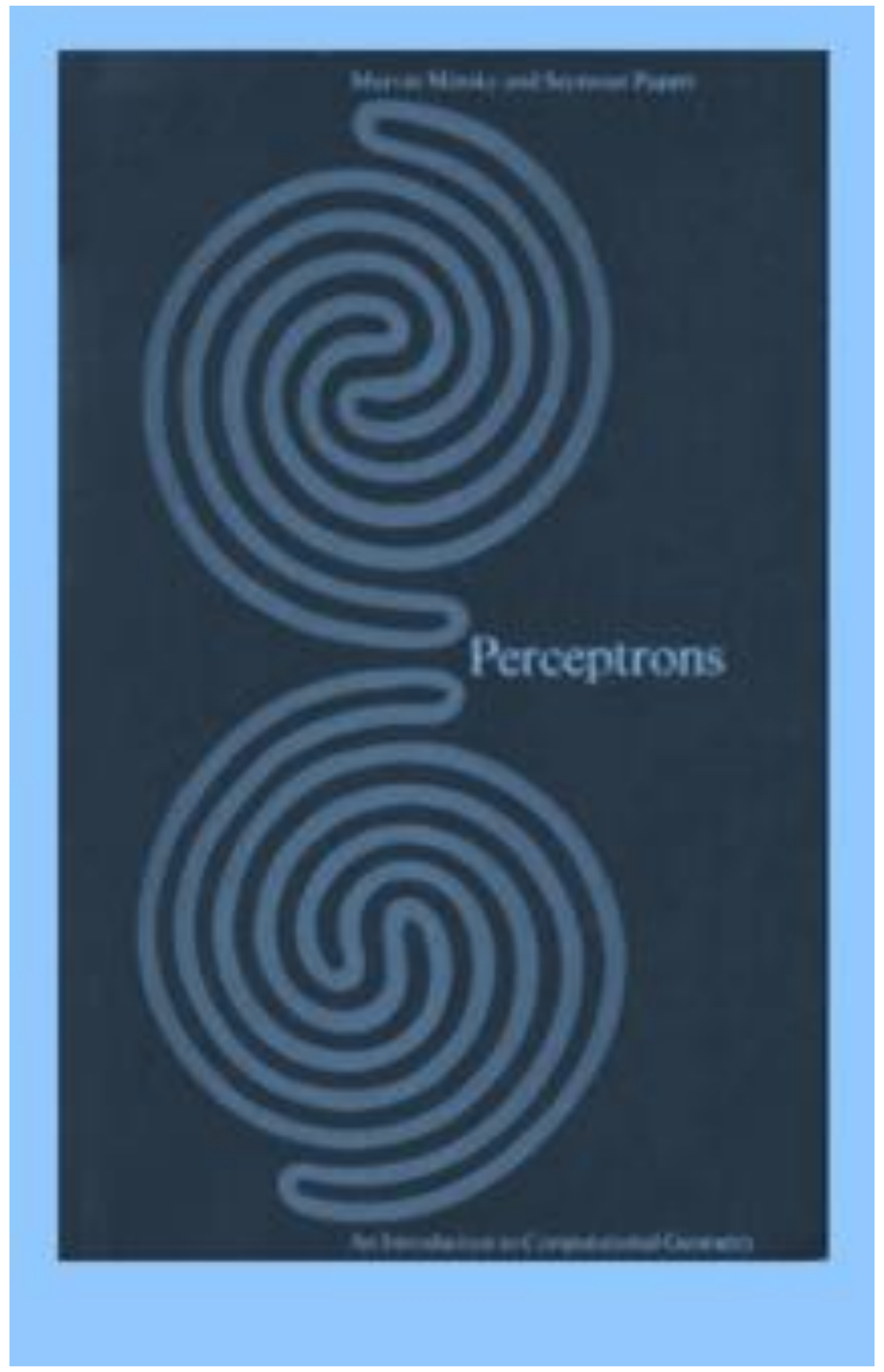
TensorBoard

```
# cost function
cost = -tf.reduce_mean(Y*tf.log(hypothesis) + (1-Y)*tf.log(1-hypothesis))
```

Back propagation



Perceptrons (1969) : by Marvin Minsky, founder of the MIT AI Lab



- We need to use MLP, multilayer perceptrons (multilayer neural nets)
- No one on earth had found a viable way to train MLPs good enough to learn such simple functions.