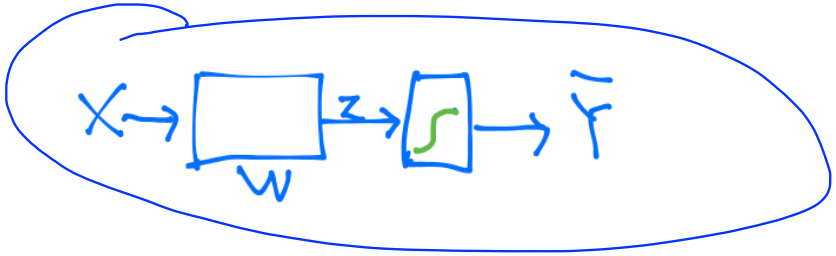
# Lecture 9-1
## Neural Nets(NN) for XOR

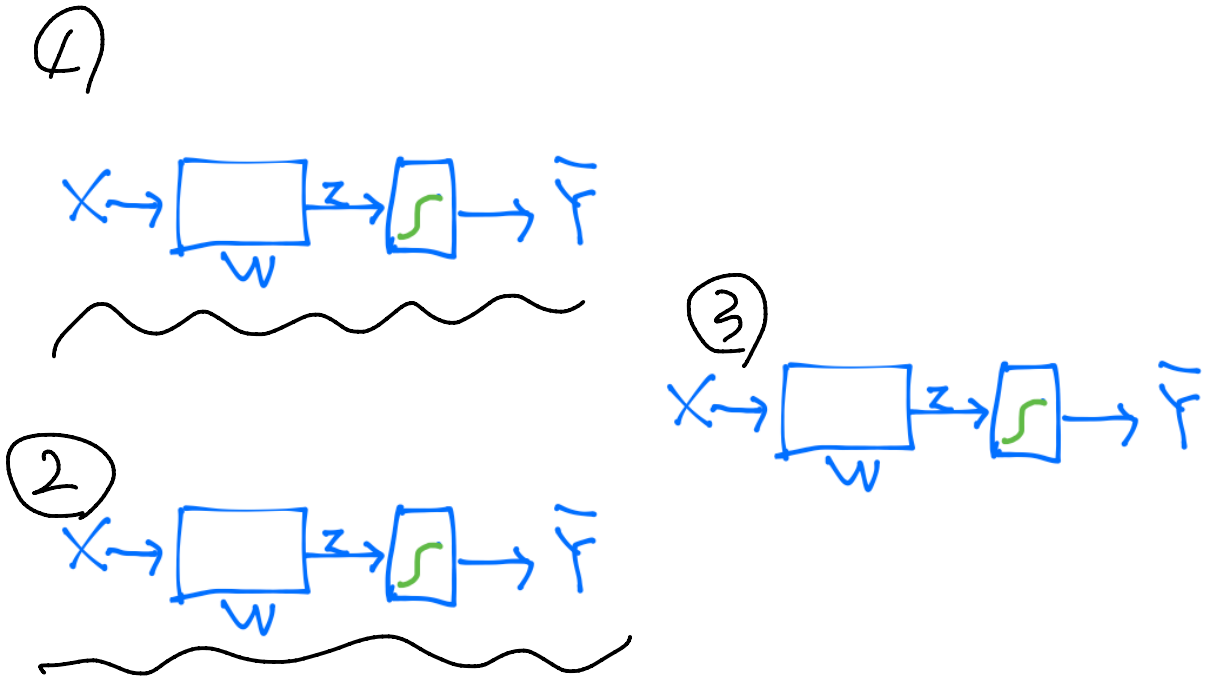
Sung Kim <hunkim+mr@gmail.com>

# One logistic regression unit cannot separate XOR

하나의 로지스틱 유닛은 XOR 는 여기로 할 수 없다

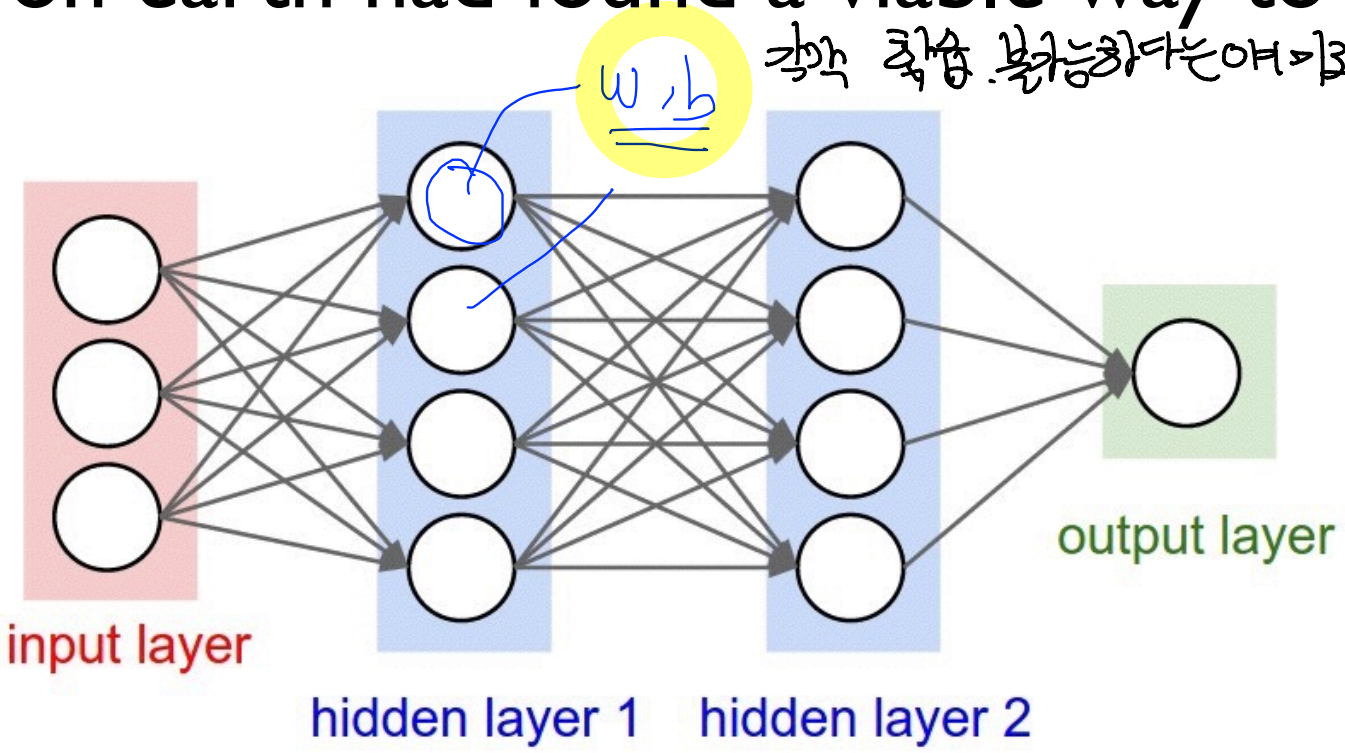


# Multiple logistic regression units



# Neural Network (NN)

"No one on earth had found a viable way to train*"

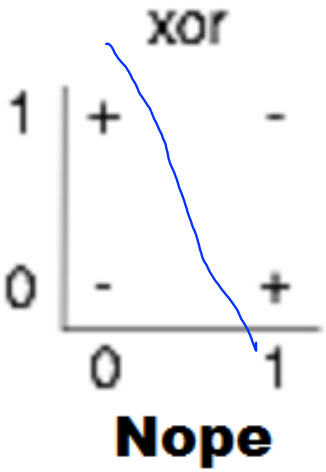



*Marvin Minsky

http://cs231n.github.io/convolutional-networks/

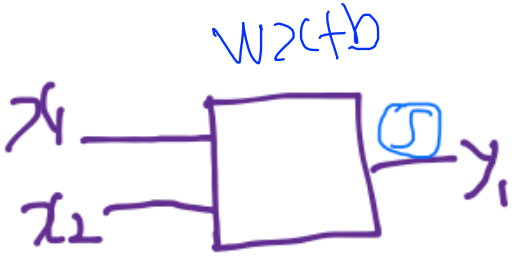
# XOR using NN

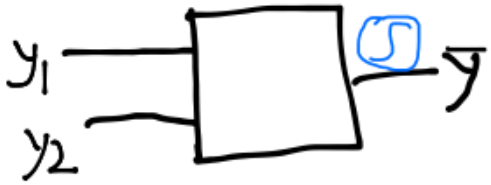
| $x_1$ | $x_2$ | XOR |
|---|---|---|
| 0 | 0 | 0 (−) |
| 0 | 1 | 1 (+) |
| 1 | 0 | 1 (+) |
| 1 | 1 | 0 (−) |

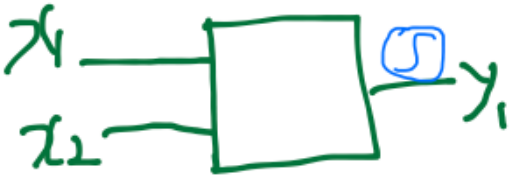


# Neural Net

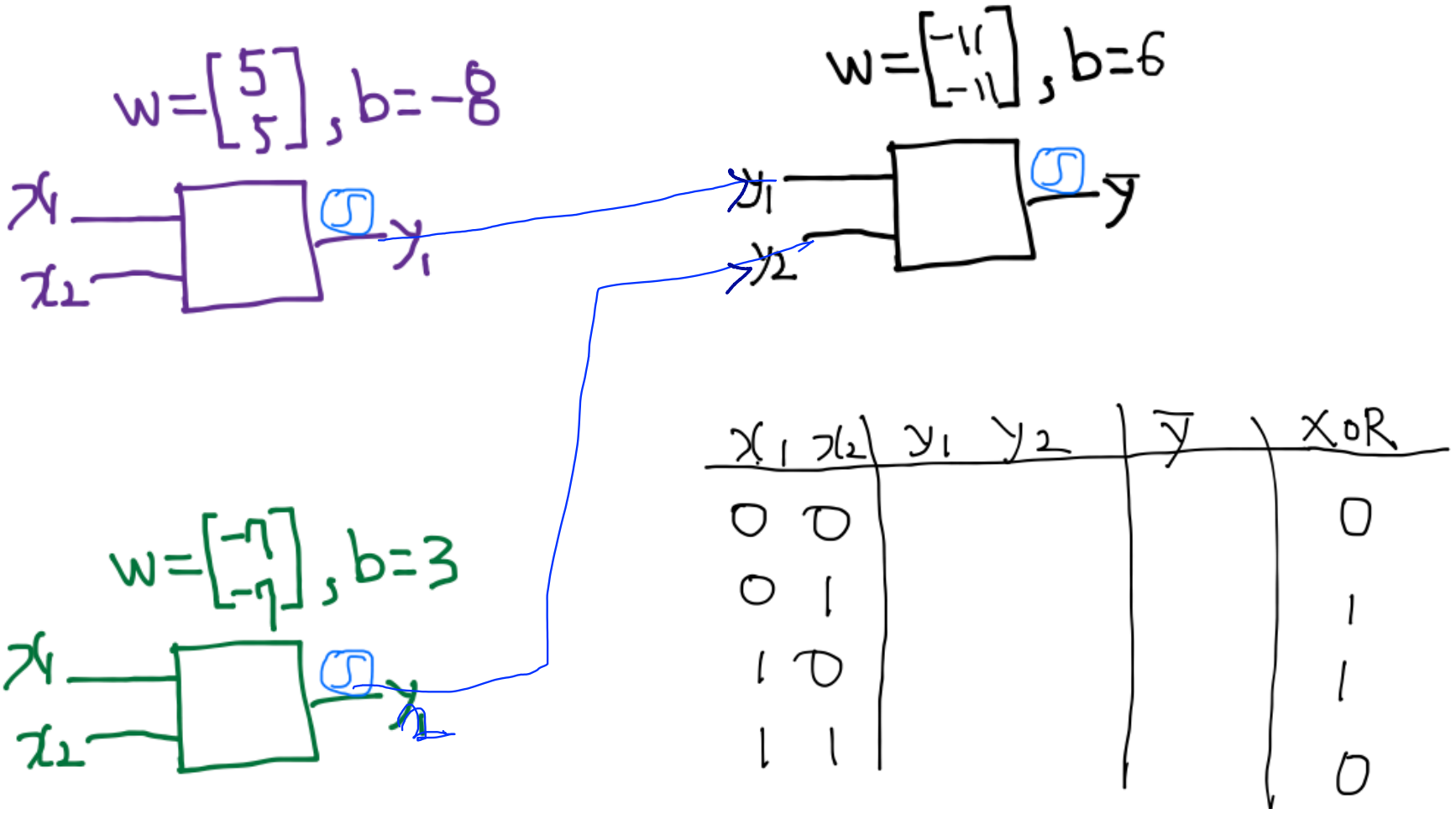
Logistic.



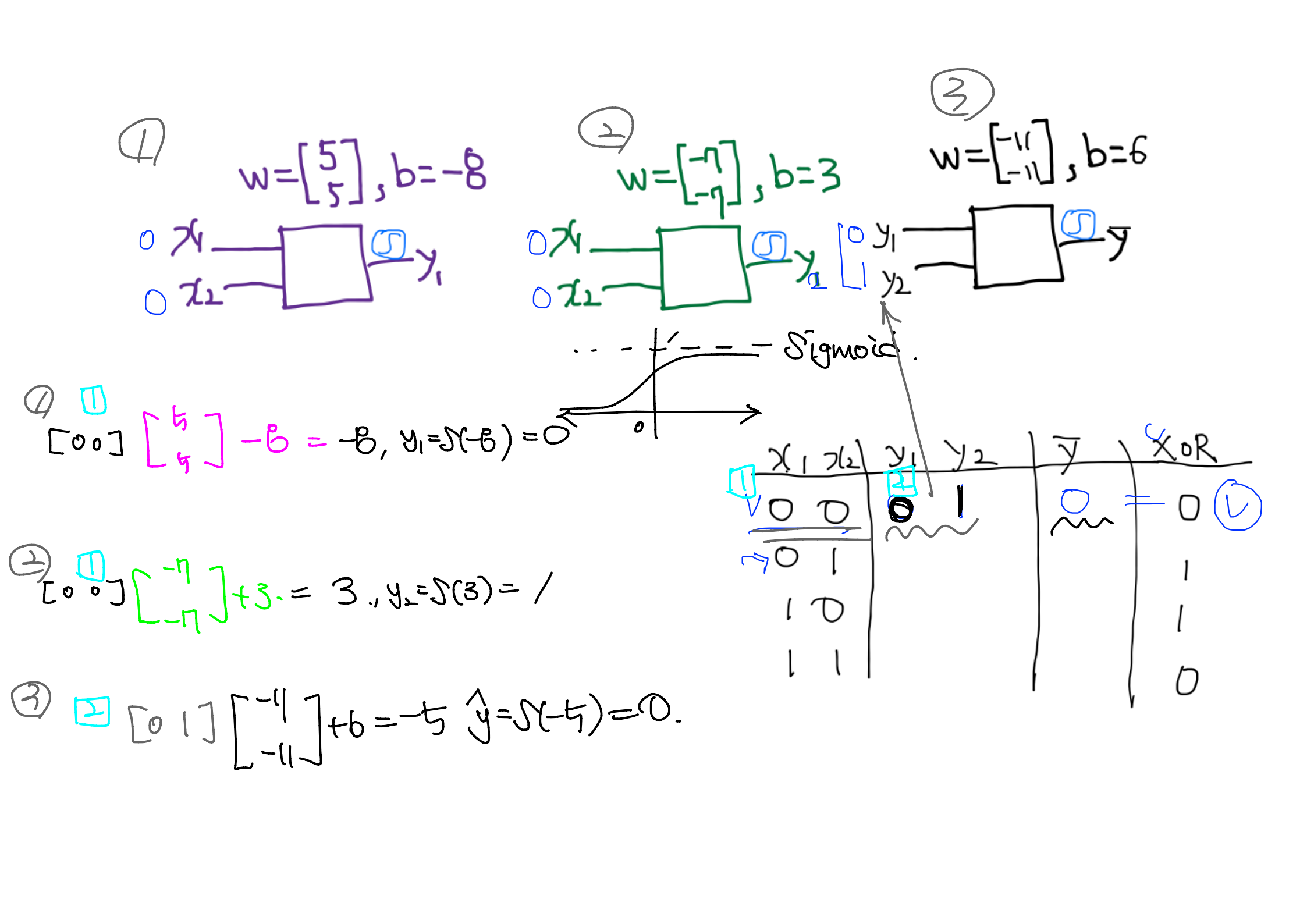



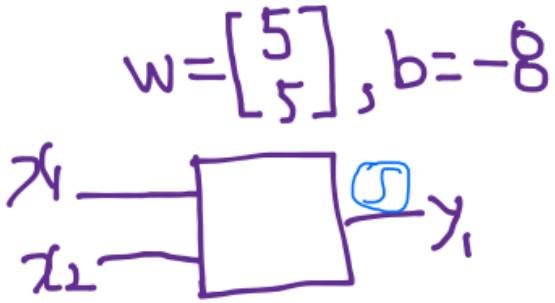



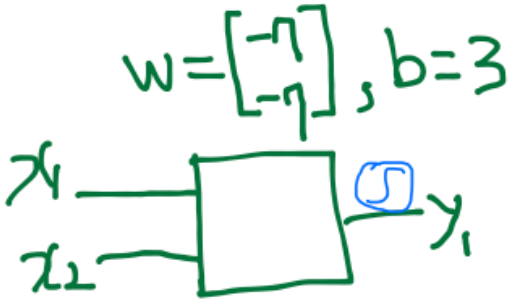
# Neural Net

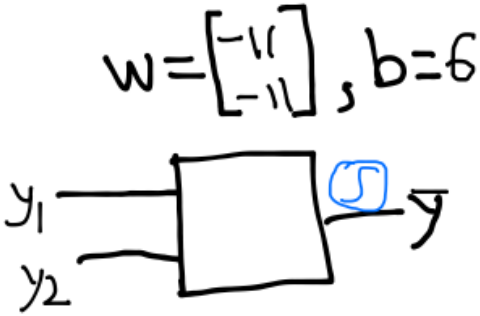



| $x_1$ | $x_2$ | $y_1$ | $y_2$ | $\bar{y}$ | XOR |
|---|---|---|---|---|---|
| 0 | 0 | | | | 0 |
| 0 | 1 | | | | 1 |
| 1 | 0 | | | | 1 |
| 1 | 1 | | | | 0 |


①
$w=\begin{bmatrix}5\\5\end{bmatrix}, b=-8$
$x_1$
$x_2$
$y_1$
②
$w=\begin{bmatrix}-7\\-7\end{bmatrix}, b=3$
$x_1$
$x_2$
$y_1$
③
$w=\begin{bmatrix}-11\\-11\end{bmatrix}, b=6$
$y_1$
$y_2$
$y$
Sigmoid.
0
① [0 0] $\begin{bmatrix}5\\5\end{bmatrix}$ $-8 = -8$, $y_1 = S(-8) = 0$
② [0 0] $\begin{bmatrix}-7\\-7\end{bmatrix}$ $+3 = 3$, $y_2 = S(3) = 1$
③ [0 1] $\begin{bmatrix}-11\\-11\end{bmatrix}$ $+6 = -5$ $\hat{y} = S(-5) = 0$.
$x_1$ $x_2$ $y_1$ $y_2$ $\hat{y}$ XOR
0 0 0 1 0 0
0 1 1 1 1
1 0 1 1
1 1 0 0



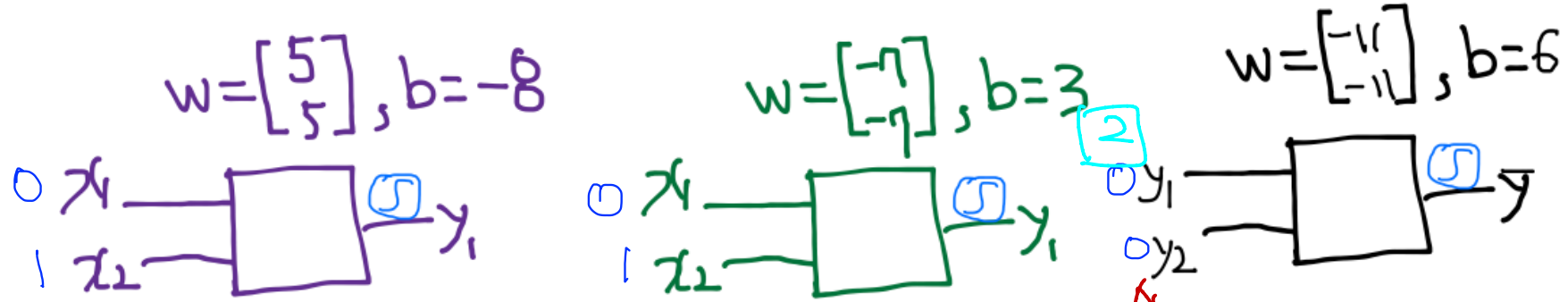

$[0\ 0]\begin{bmatrix}5\\5\end{bmatrix}-8=0+0-8=-8$, sigmoid$(-8)=0$

$[0\ 0]\begin{bmatrix}-7\\-7\end{bmatrix}+3=0+0+3=3$, sigmoid$(3)=1$

$[0\ 1]\begin{bmatrix}-11\\-11\end{bmatrix}+6=0+-11+6=-5$

sigmoid$(-5)=0$

| $x_1$ | $x_2$ | $y_1$ | $y_2$ | $y$ | XOR |
|---|---|---|---|---|---|
| 0 | 0 | 0 | 1 | 0 | 0 |
| 0 | 1 | | | | 1 |
| 1 | 0 | | | | 1 |
| 1 | 1 | | | | 0 |




1

$[0\ 1]\begin{bmatrix}5\\5\end{bmatrix}-8 = 0+5-8 = -3$, Sigmoid(-3) = 0

1

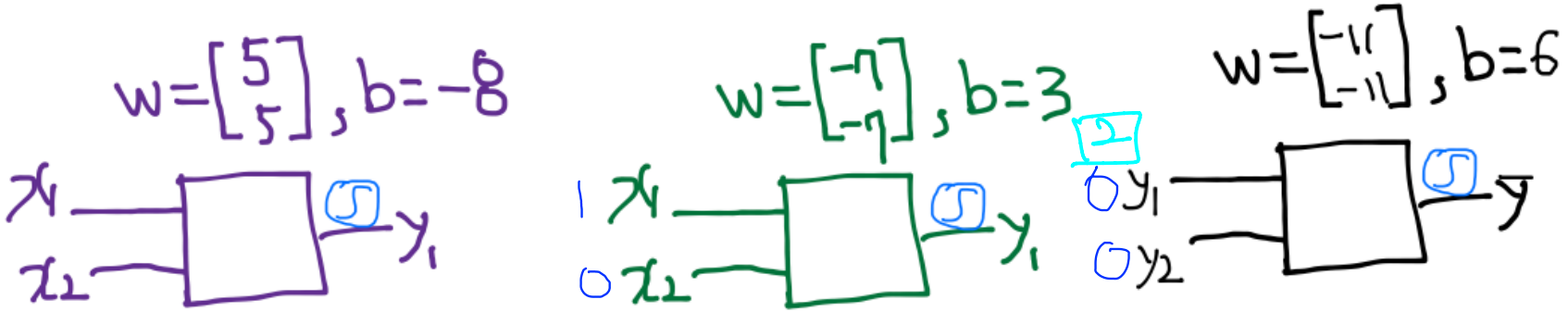
$[0\ 1]\begin{bmatrix}-7\\-7\end{bmatrix}+3 = 0+-7+3 = -4$, Sigmoid(-4) = 0

2

$[0\ 0]\begin{bmatrix}-11\\-11\end{bmatrix}+6 = 0+0+6 = 6$

Sigmoid(6) = 1

| x1 | x2 | y1 | y2 | y | XOR |
|---|---|---|---|---|---|
| 0 | 0 | 0 | 1 | 0 | 0 |
| 0 | 1 | 0 | 0 | 1 | 1 |
| 1 | 0 | | | | 1 |
| 1 | 1 | | | | 0 |




0

$[1\ 0]\begin{bmatrix}5\\5\end{bmatrix}-8 = 5+0-8 = -3$, sigmoid$(-3)=0$

1

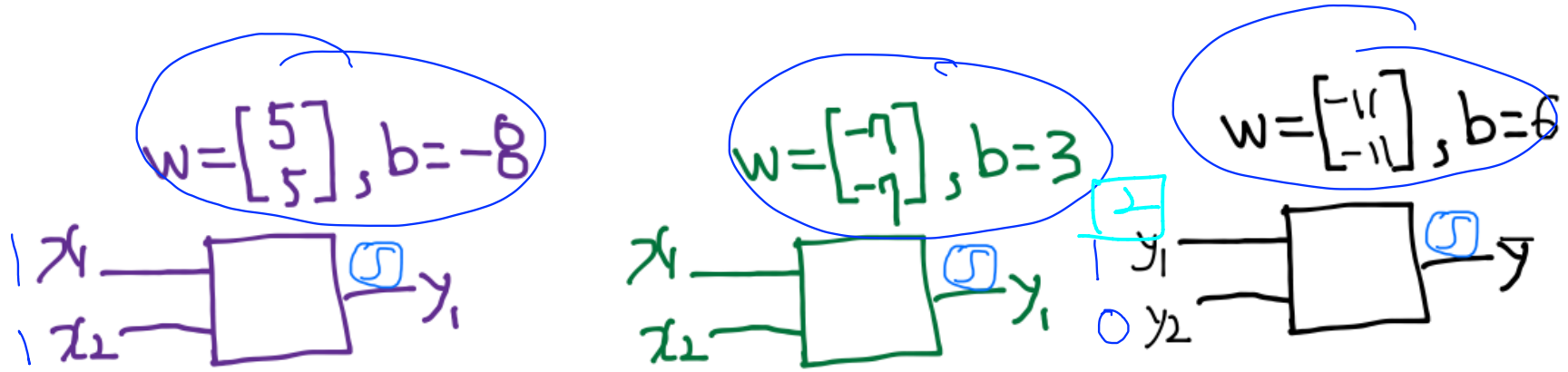
$[1\ 0]\begin{bmatrix}-7\\-7\end{bmatrix}+3 = -7+0+3 = -4$, sigmoid$(-4)=0$

2

$[0\ 0]\begin{bmatrix}-11\\-11\end{bmatrix}+6 = 0+0+6 = 6$

sigmoid$(6)=1$

| $x_1$ | $x_2$ | $y_1$ | $y_2$ | $y$ | XOR |
|---|---|---|---|---|---|
| 0 | 0 | 0 | 1 | 0 | 0 ✓ |
| 0 | 1 | 0 | 0 | 1 | 1 ✓ |
| 1 | 0 | 0 | 0 | 1 | 1 ✓ |
| 1 | 1 | | | | 0 ? |




1. $[1\ 1]\begin{bmatrix}5\\5\end{bmatrix}-8 = 5+5-8 = 2$, sigmoid(2) = 1

2. $[1\ 1]\begin{bmatrix}-7\\-7\end{bmatrix}+3 = -7-7+3 = -11$, sigmoid(-11) = 0

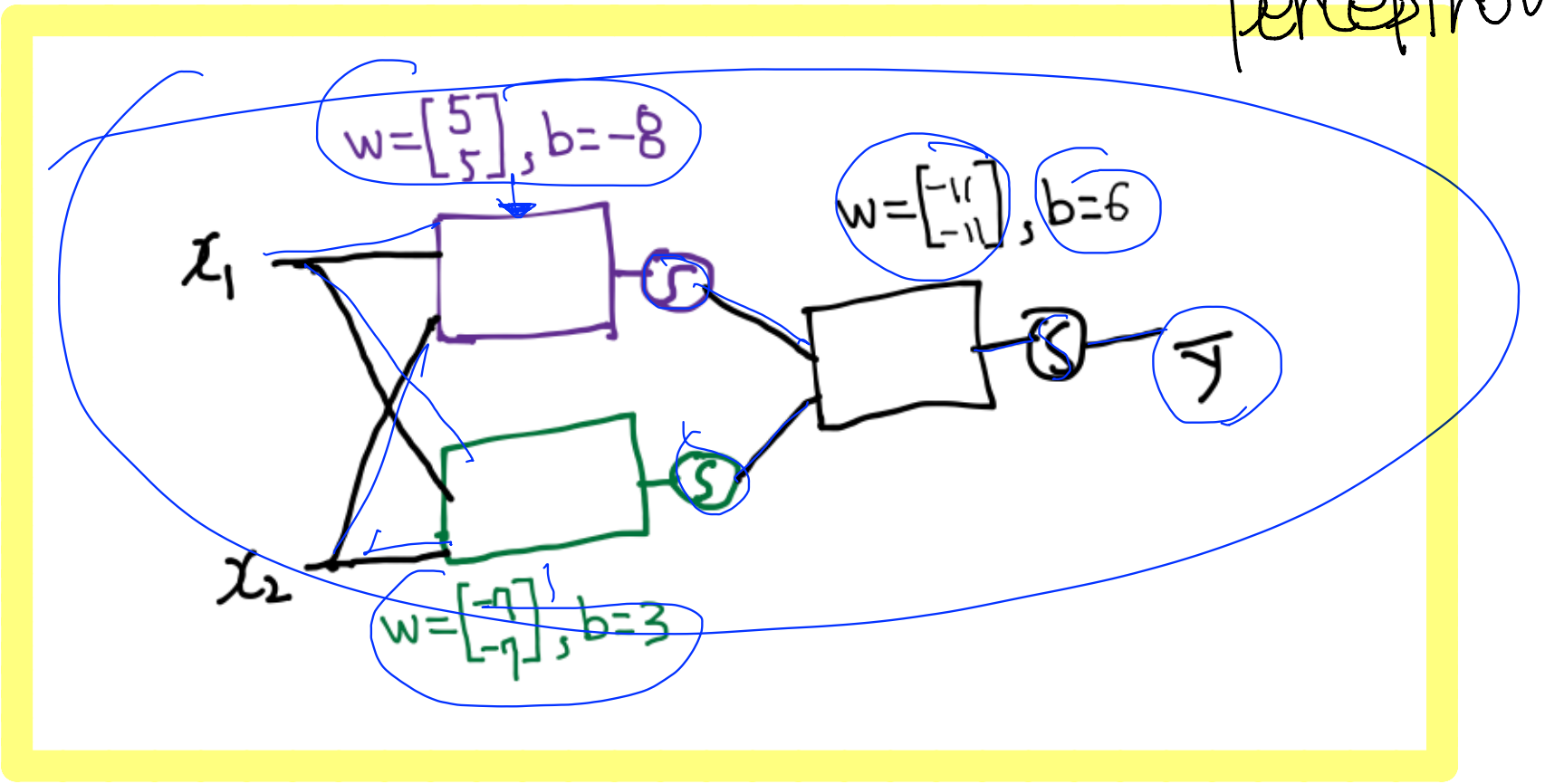
3. $[1\ 0]\begin{bmatrix}-11\\-11\end{bmatrix}+6 = -11+0+6 = -5$

   sigmoid(-5) = 0

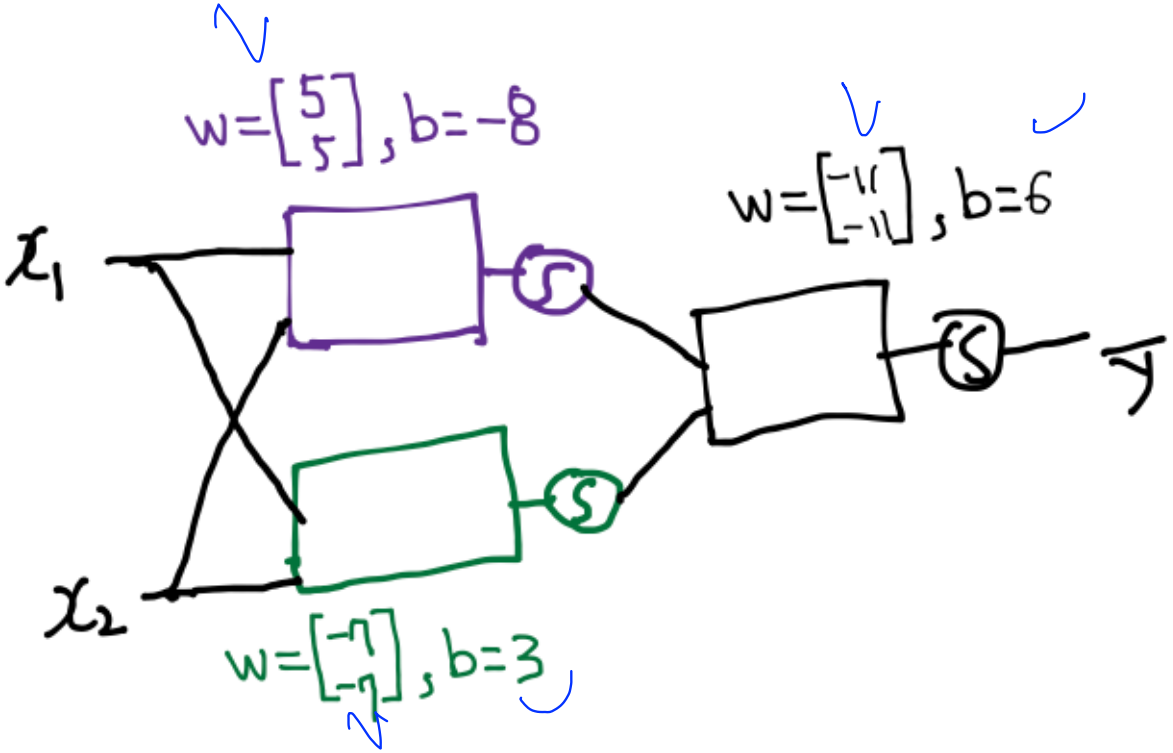
| $x_1$ | $x_2$ | $y_1$ | $y_2$ | $y$ | XOR |
|---|---|---|---|---|---|
| 0 | 0 | 0 | 1 | 0 | 0 |
| 0 | 1 | 0 | 0 | 1 | 1 |
| 1 | 0 | 0 | 0 | 1 | 1 |
| 1 | 1 | 1 | 0 | 0 | 0 |

# Forward propagation

perceptron.



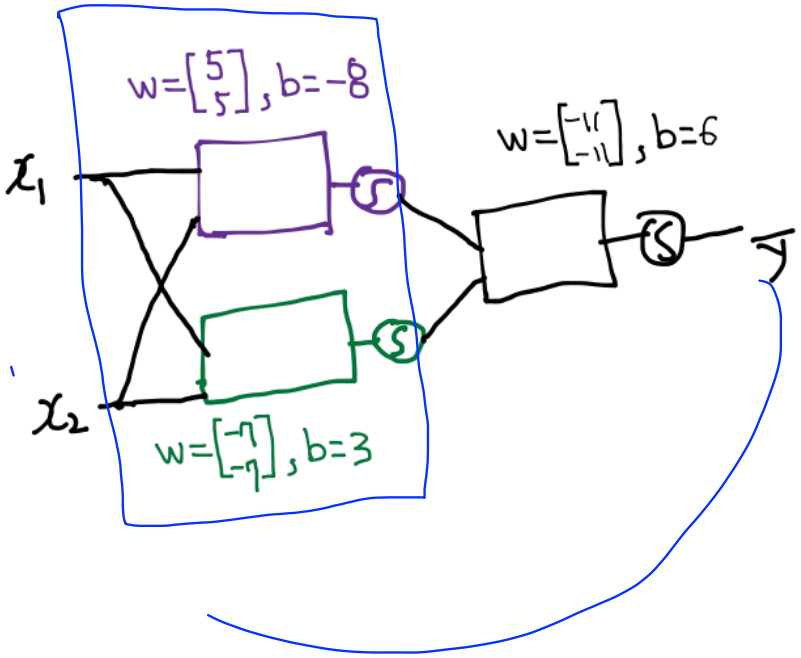
# Forward propagation

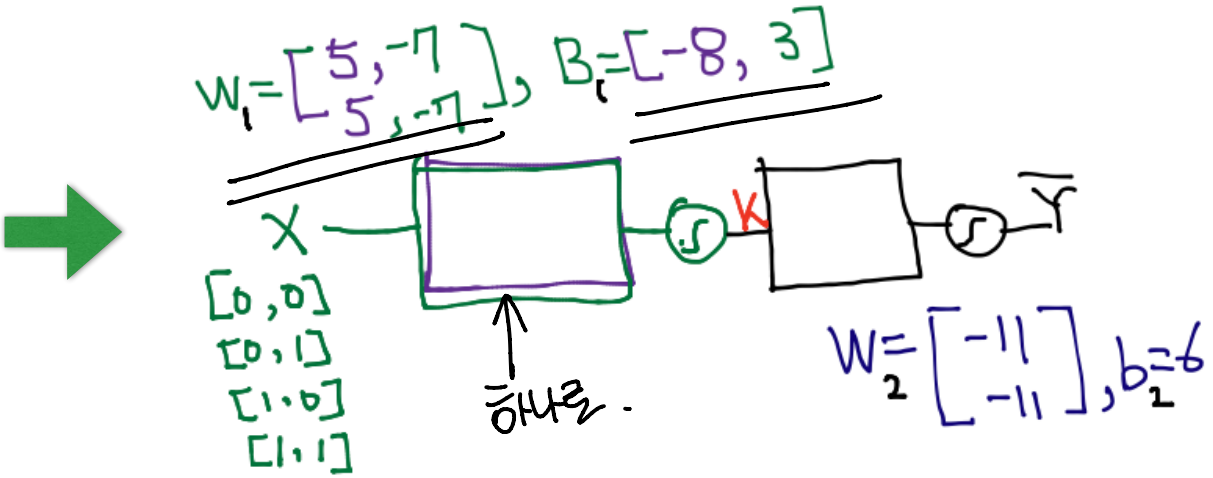



*Can you find another W and b for the XOR?*

Multivariables. (Next page)

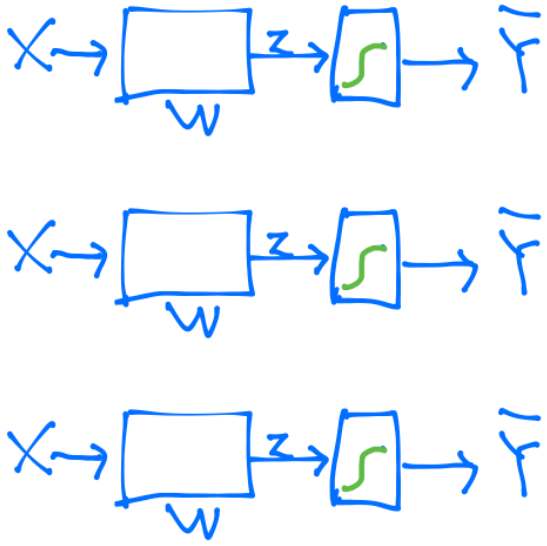
# NN



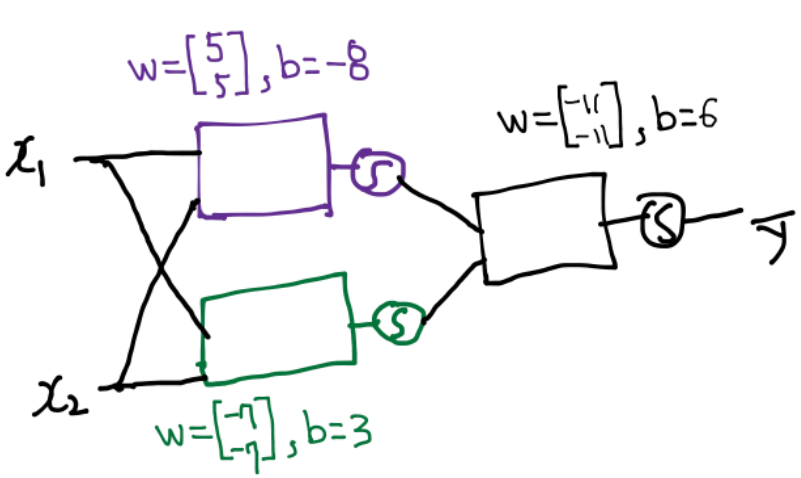



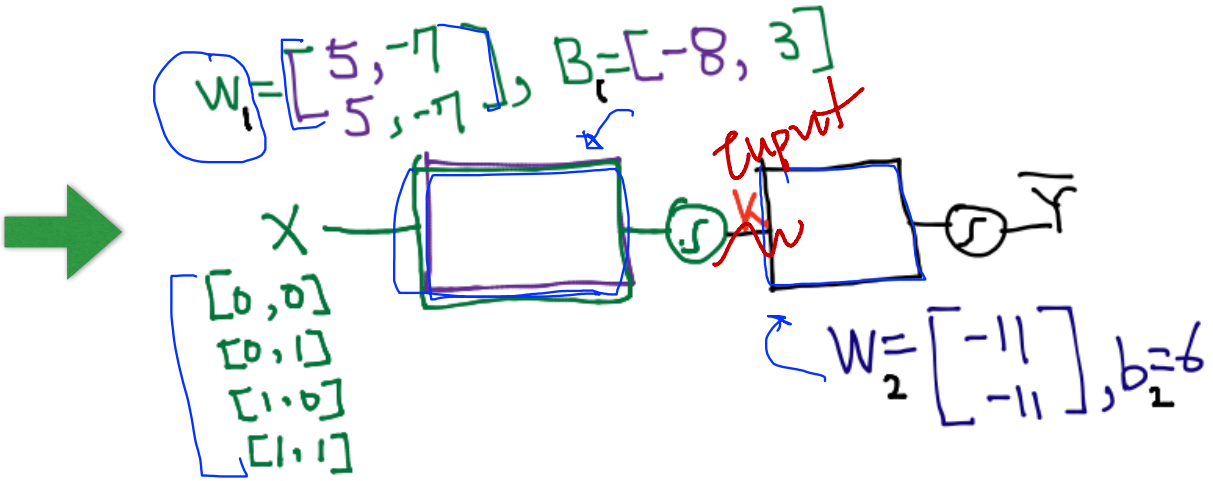
# Recap: Lec 6-1 Multinomial classification

$$\begin{bmatrix} w_{A1} & w_{A2} & w_{A3} \\ w_{B1} & w_{B2} & w_{B3} \\ w_{C1} & w_{C2} & w_{C3} \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ x_3 \end{bmatrix} = \begin{bmatrix} w_{A1}x_1 + w_{A2}x_2 + w_{A3}x_3 \\ w_{B1}x_1 + w_{B2}x_2 + w_{B3}x_3 \\ w_{C1}x_1 + w_{C2}x_2 + w_{C3}x_3 \end{bmatrix} = \begin{bmatrix} \overline{y}_A \\ \overline{y}_B \\ \overline{y}_C \end{bmatrix}$$



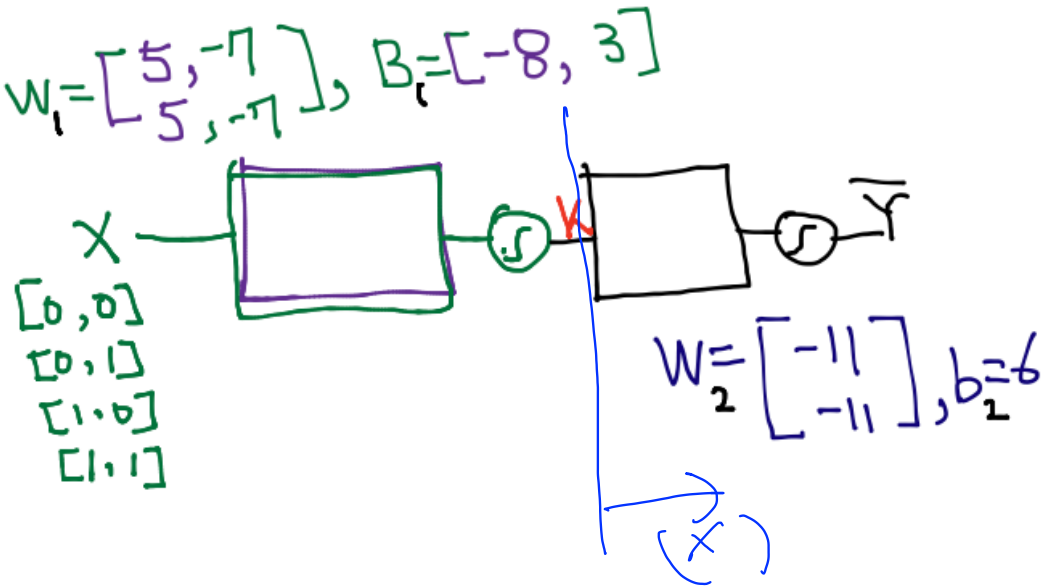
# NN



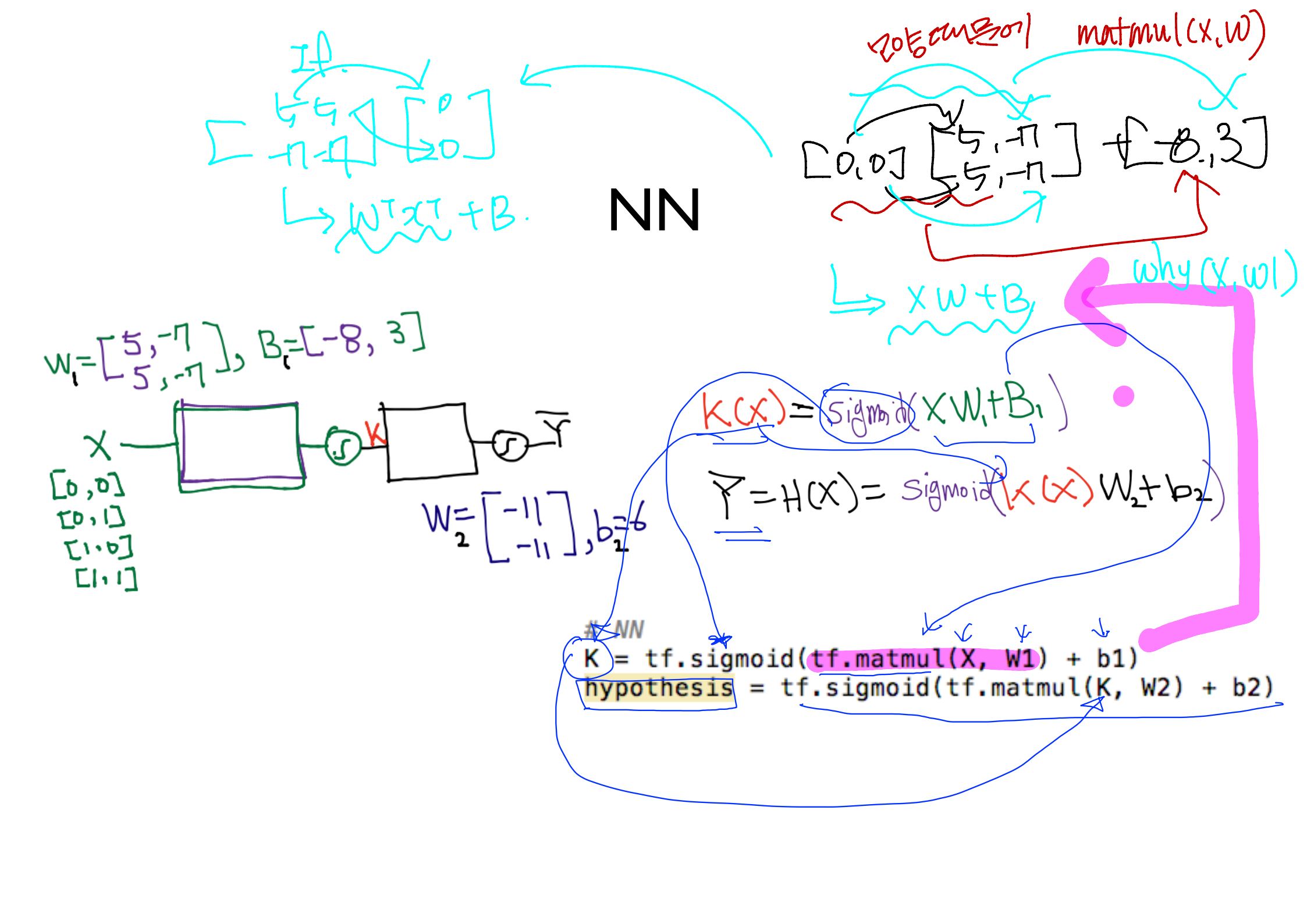

*How can we learn W, and b from trading data?*

# NN

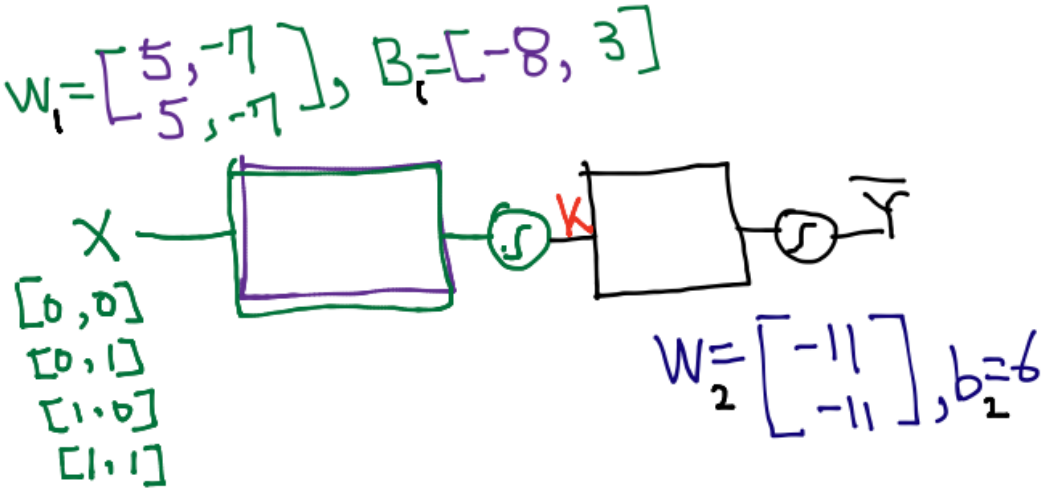



$$K(X) = \text{sigmoid}(XW_1 + B_1)$$

$$\hat{Y} = H(X) = \text{sigmoid}(K(X)W_2 + b_2)$$


NN
matmul(X,W)
XW+B
why(X,W)
$W_1=\begin{bmatrix}5,-7\\5,-7\end{bmatrix}$, $B_1=[-8, 3]$
X
[0,0]
[0,1]
[1,0]
[1,1]
K
$W_2=\begin{bmatrix}-11\\-11\end{bmatrix}$, $b_2=6$
$K(X)=\text{sigmoid}(XW_1+B_1)$
$\hat{Y}=H(X)=\text{sigmoid}(K(X)W_2+b_2)$
# NN
K = tf.sigmoid(tf.matmul(X, W1) + b1)
hypothesis = tf.sigmoid(tf.matmul(K, W2) + b2)

# NN




$$K(X) = \text{sigmoid}(XW_1 + B_1)$$

$$\hat{Y} = H(X) = \text{sigmoid}(K(X)W_2 + b_2)$$

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

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

인공지능의 아.

**Next**

Backpropagation

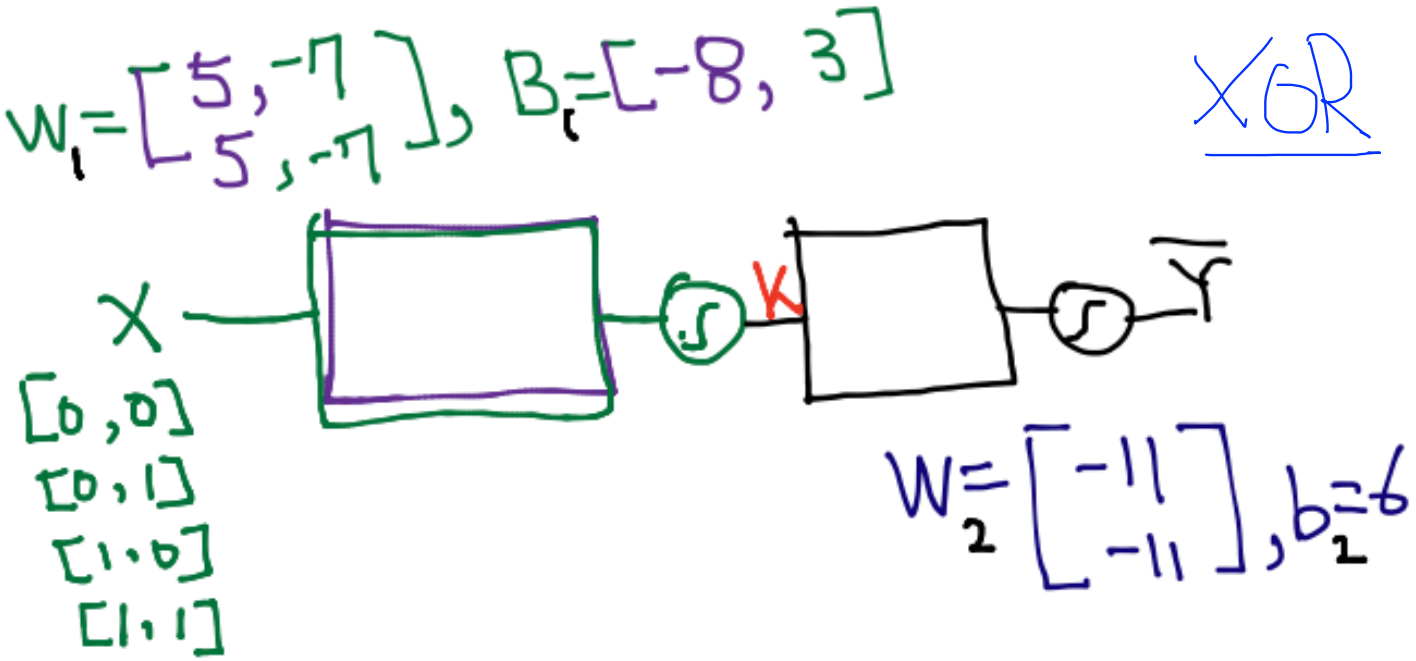
# Lecture 9-2

## Backpropagation

오차역전파알고리즘

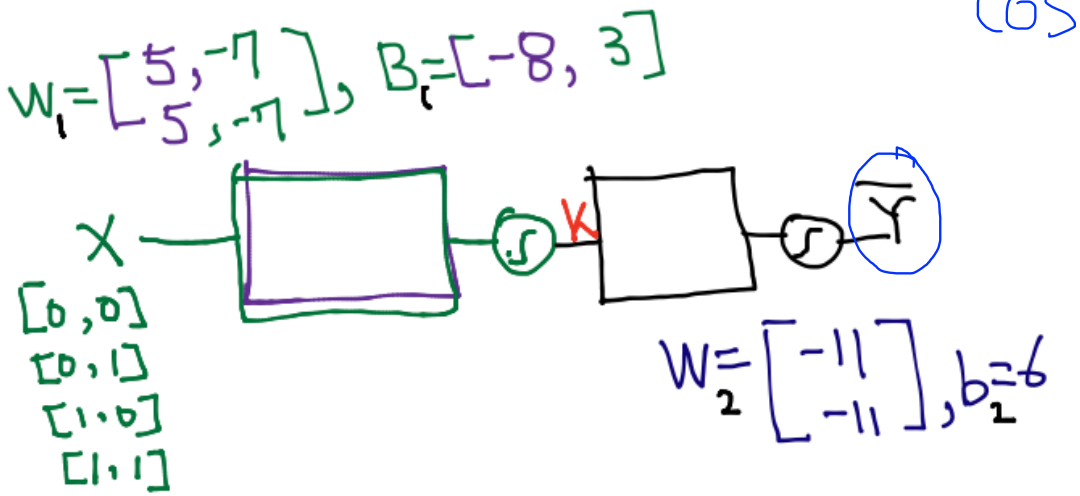
Sung Kim <hunkim+mr@gmail.com>

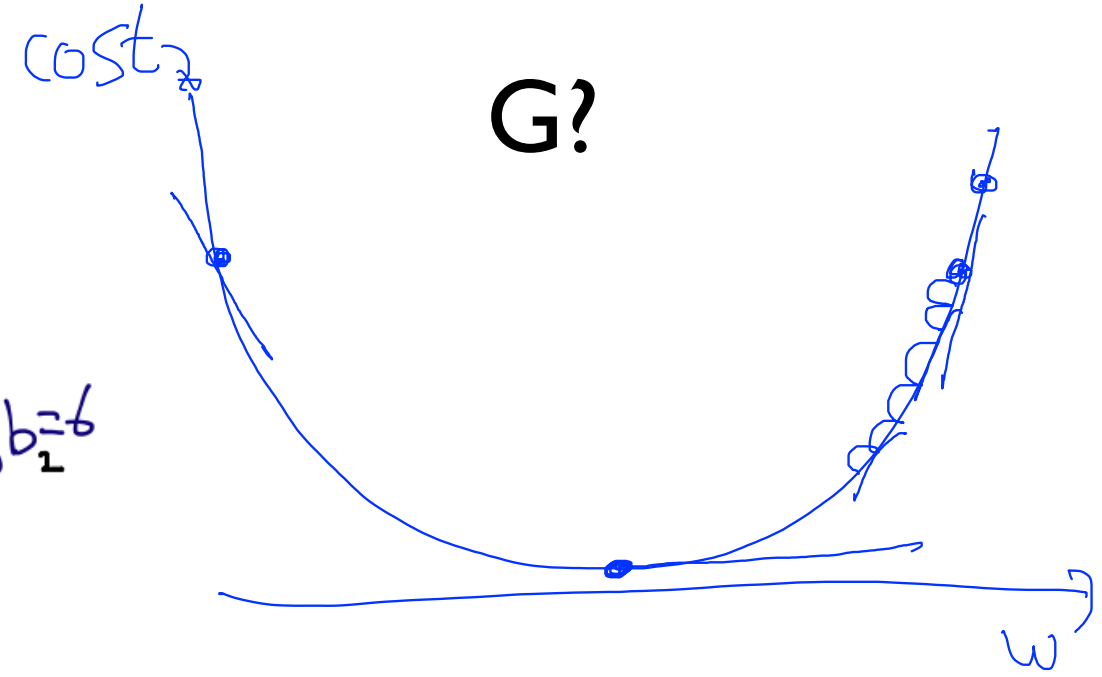
# Neural Network (NN)



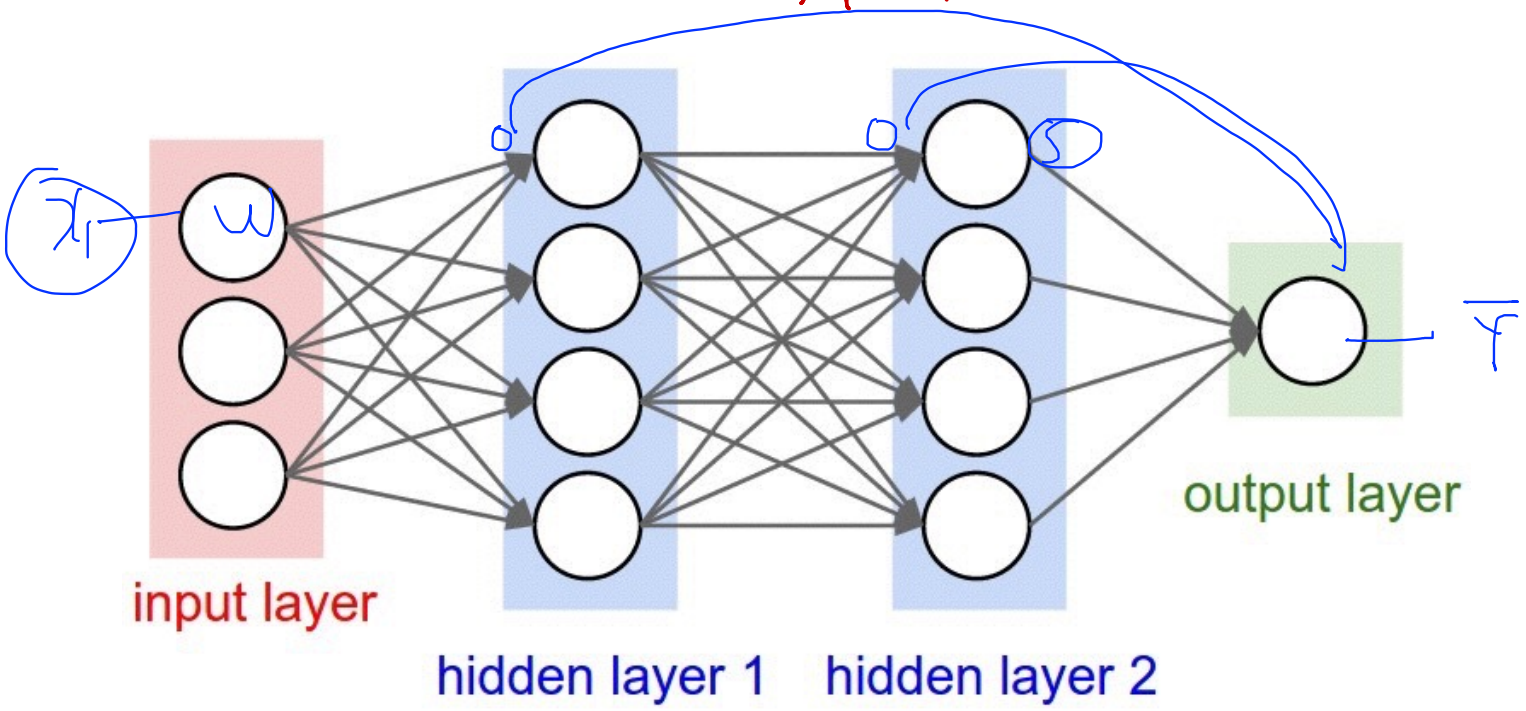

*How can we learn W1, W2, B1, b2 from training data?*

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



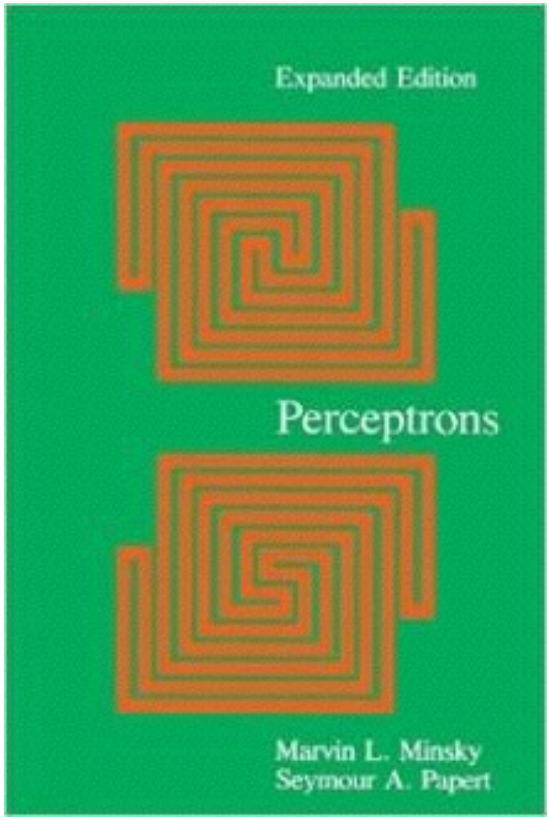



# Derivation



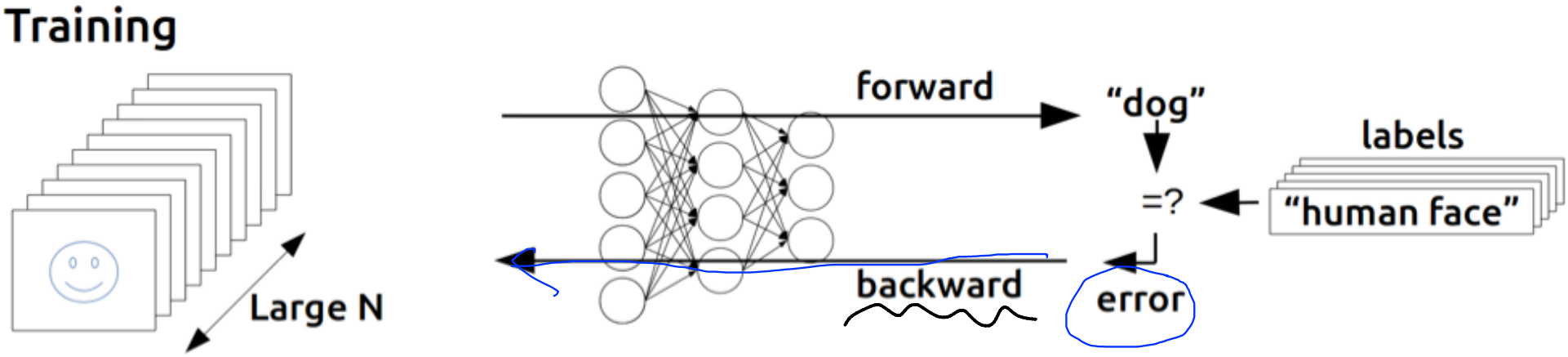
# Perceptrons (1969)

## by Marvin Minsky, founder of the MIT AI Lab



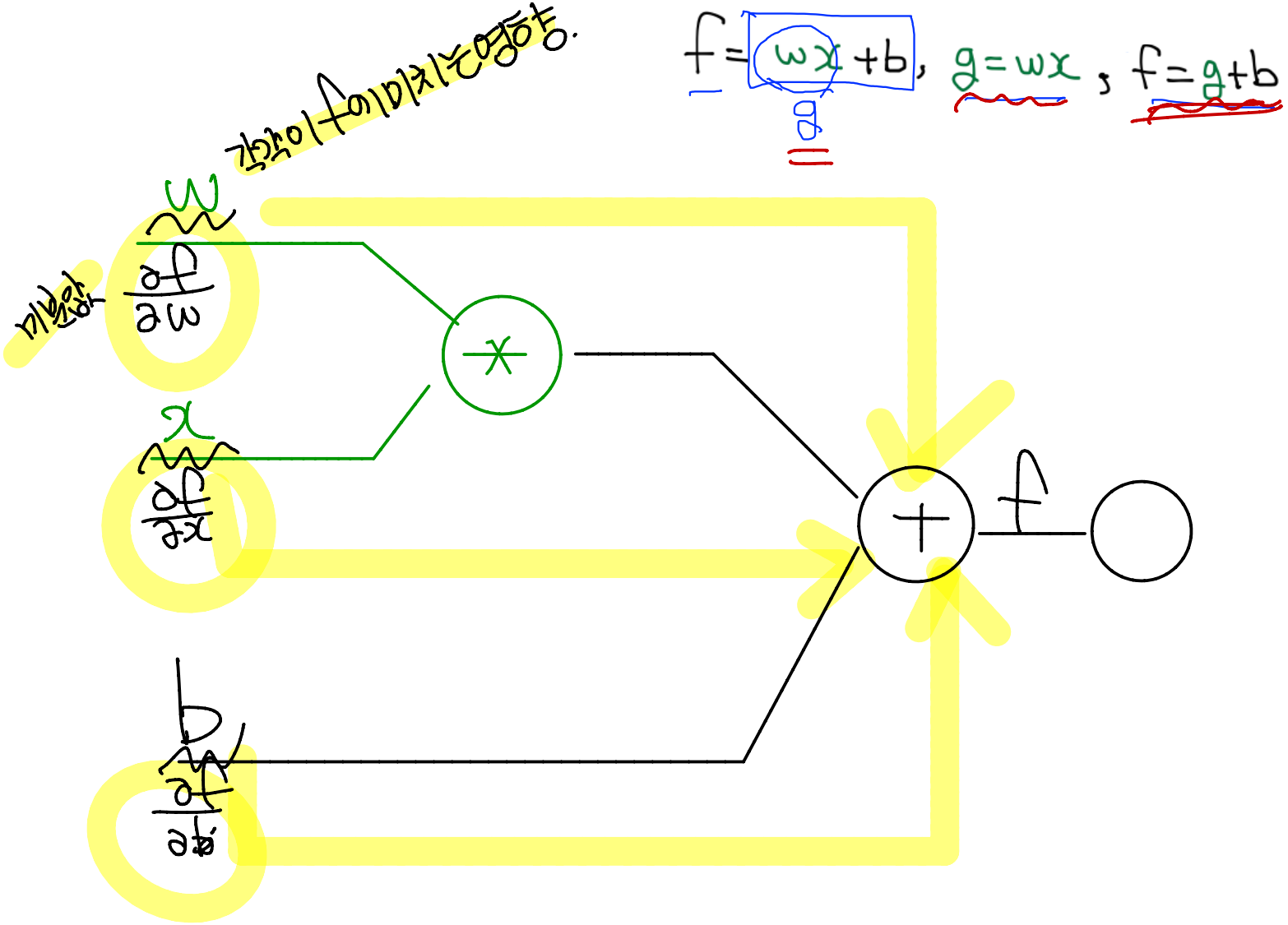

- We need to use MLP, multilayer perceptrons (multilayer neural nets)
- No one on earth had found a viable way to train MLPs good enough to learn such simple functions.

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

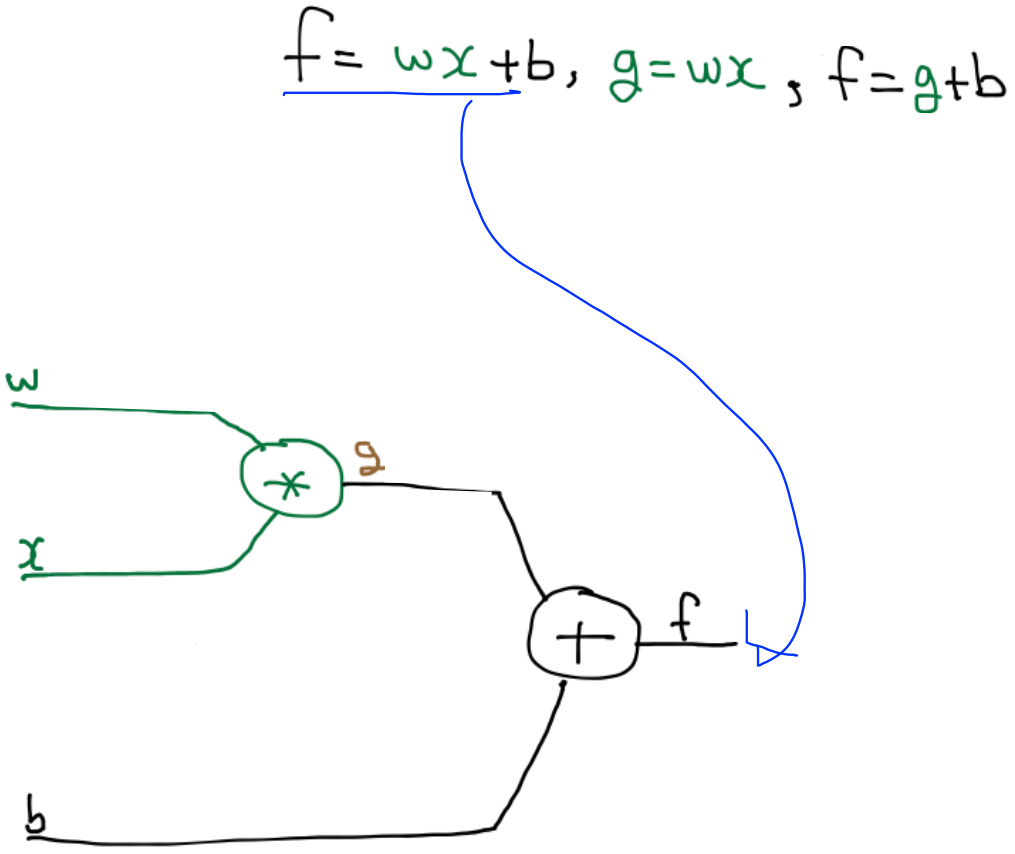



https://devblogs.nvidia.com/parallelforall/inference-next-step-gpu-accelerated-deep-learning/

# Back propagation (chain rule)

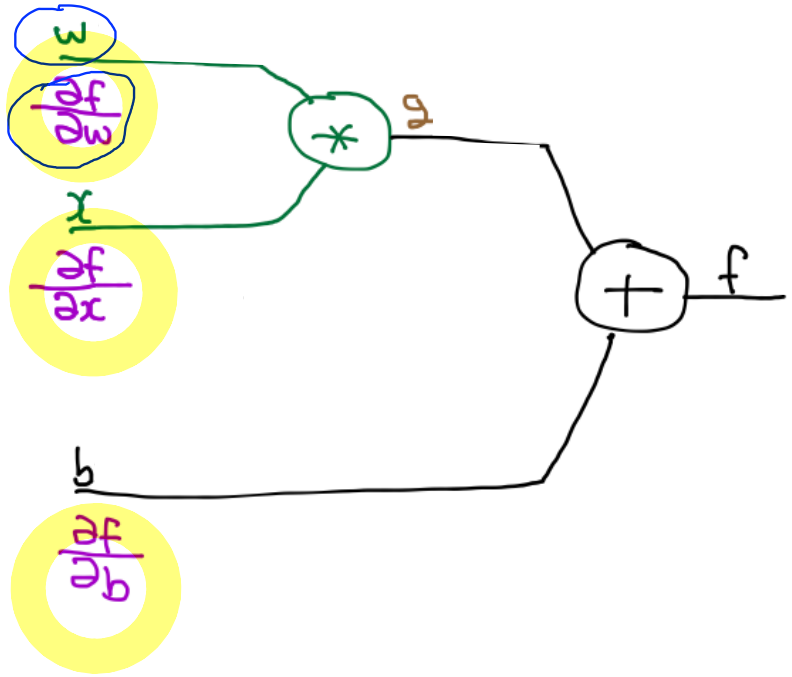


# Back propagation (chain rule)

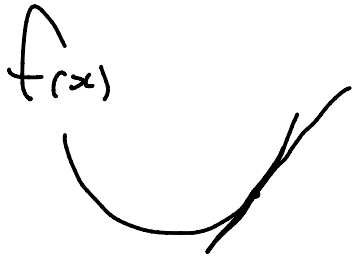


# Back propagation (chain rule)

$f = wx + b$, $g = wx$, $f = g + b$



# Basic derivative




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

$f(x) = 3$

$f(x) = x$

$f(x) = 2x$

https://ko.wikipedia.org/wiki/%EB%AF%B8%EB%B6%84

# Partial derivative: consider other variables as constants

$f(x) = 2x$

$f(x, y) = xy, \frac{\partial f}{\partial x}$

$f(x, y) = xy, \frac{\partial f}{\partial y}$

https://ko.wikipedia.org/wiki/%EB%AF%B8%EB%B6%84

# Partial derivative: consider other variables as constants

$f(x) = 3$ ⇒ 0

$f(x) = 2x$ $f(x) = x + x$

2 2

$f(x) = x + 3$

1

$f(x, y) = x + y, \frac{\partial f}{\partial x}$ 1

$f(x, y) = x + y, \frac{\partial f}{\partial y}$ 1

$f(g(x))$

⇒ x가 f에 어떻게 영향을 끼치나

$z = wx + b$

$k = Sig(z)$

$\frac{\partial z}{\partial x} \cdot \frac{\partial k}{\partial z}$

Chain rule

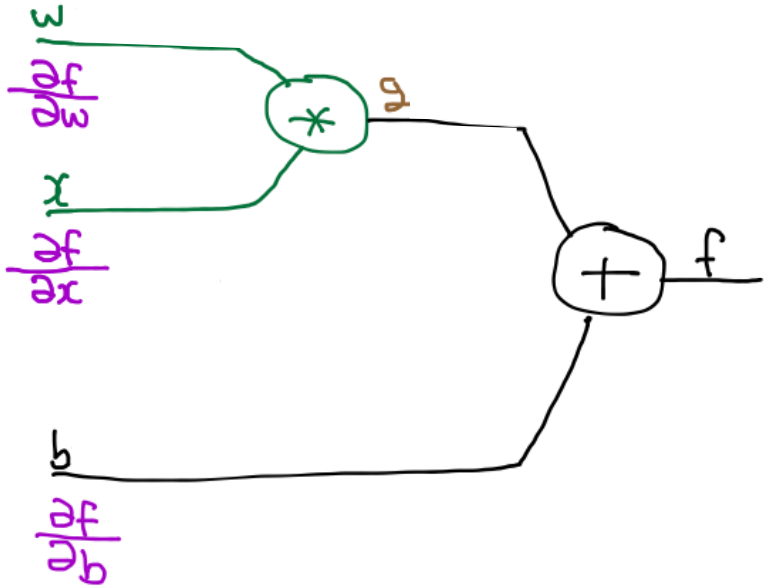
$f(g(x))$

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

https://ko.wikipedia.org/wiki/%EB%AF%B8%EB%B6%84

# Back propagation (chain rule)

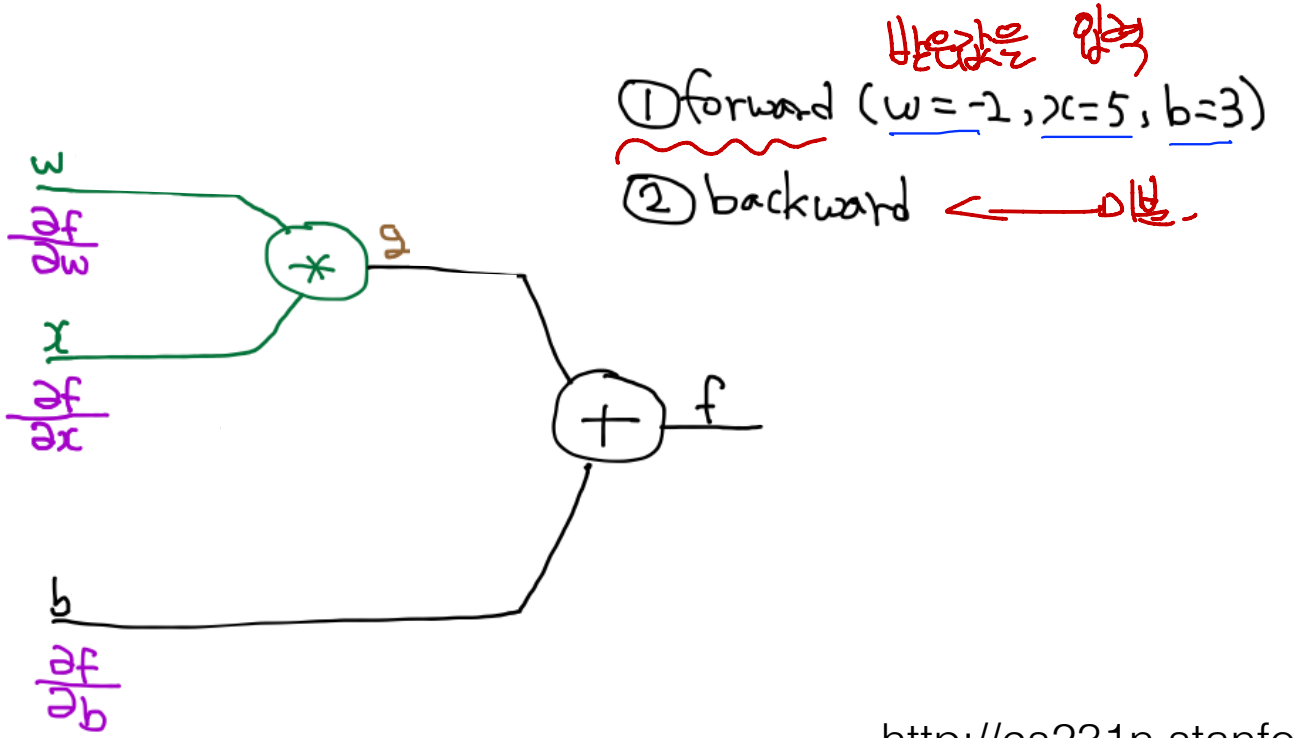
$f = wx + b$, $g = wx$, $f = g + b$




http://cs231n.stanford.edu/

# Back propagation (chain rule)

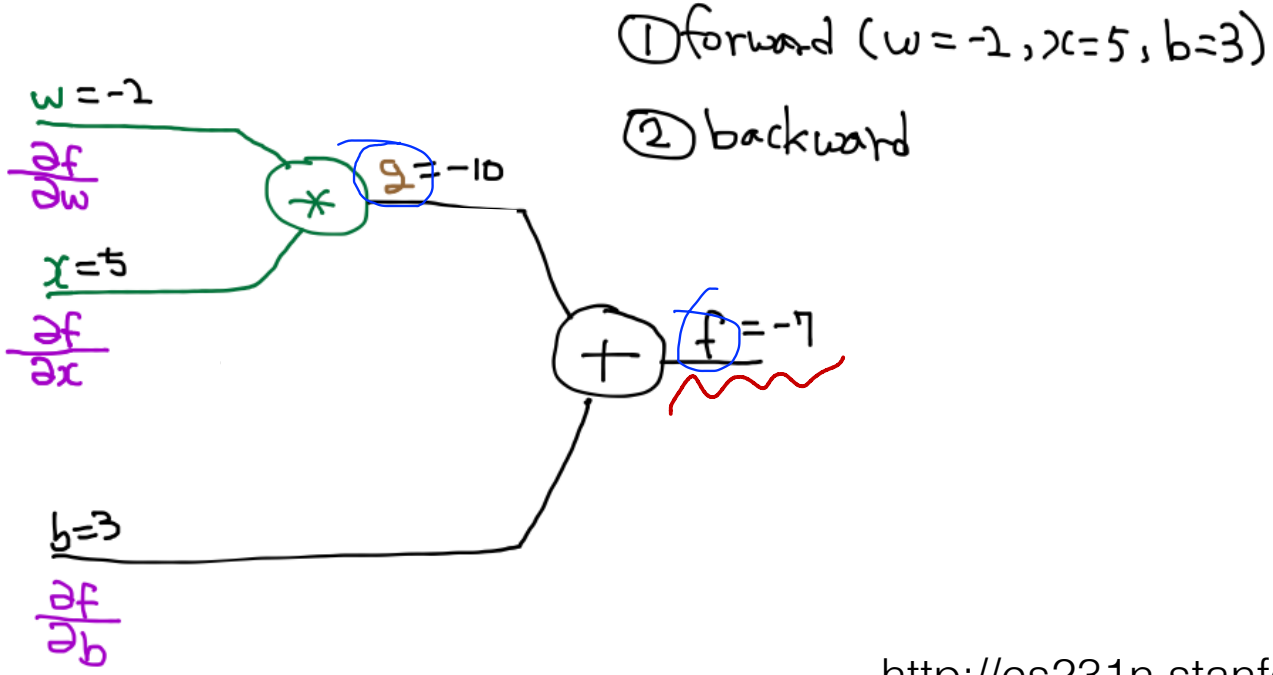
$f = wx + b,\ g = wx,\ f = g + b$




http://cs231n.stanford.edu/

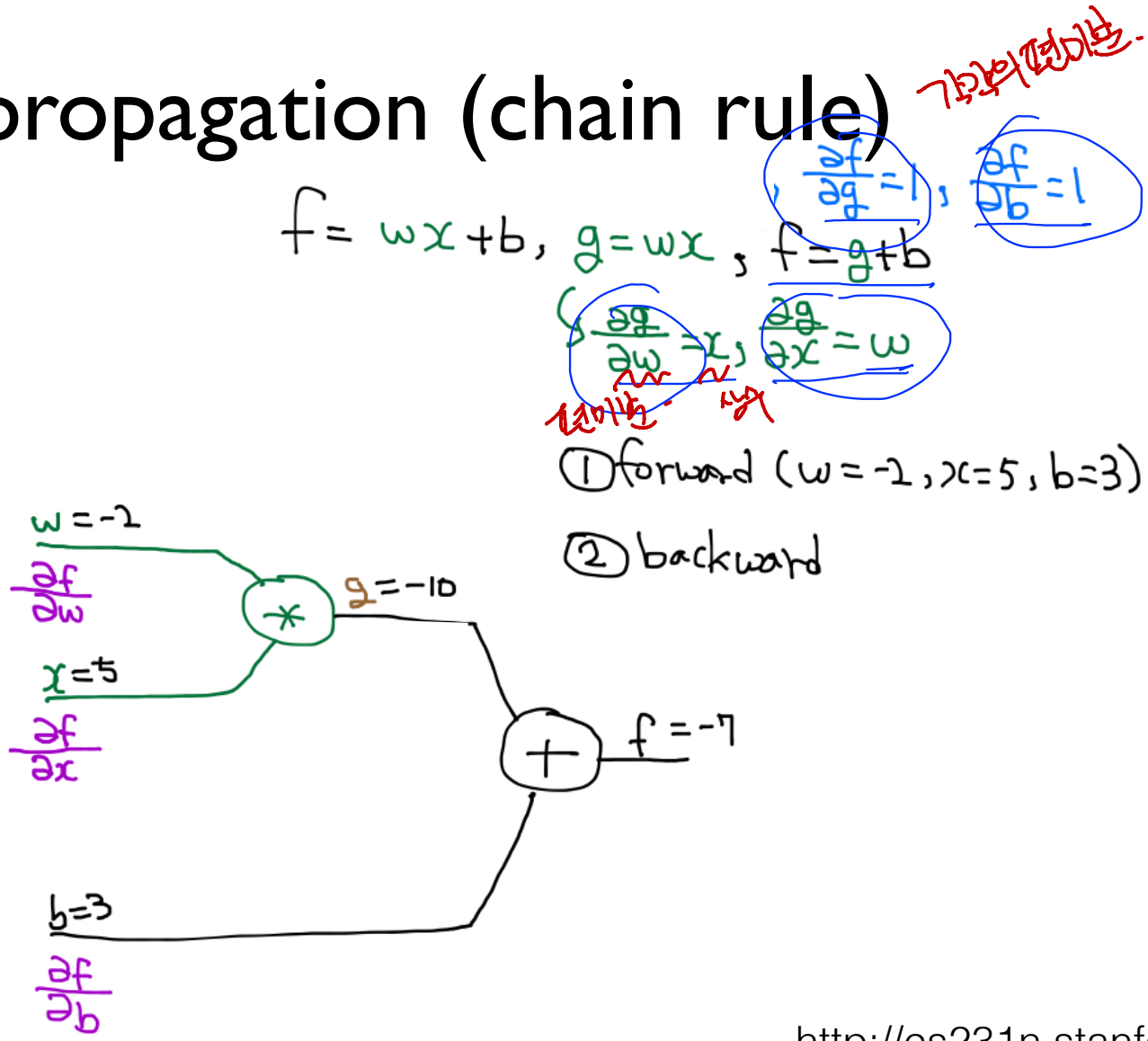
# Back propagation (chain rule)

$f = wx + b$, $g = wx$, $f = g + b$




http://cs231n.stanford.edu/

# Back propagation (chain rule)




http://cs231n.stanford.edu/

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

# Back propagation (chain rule)

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

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

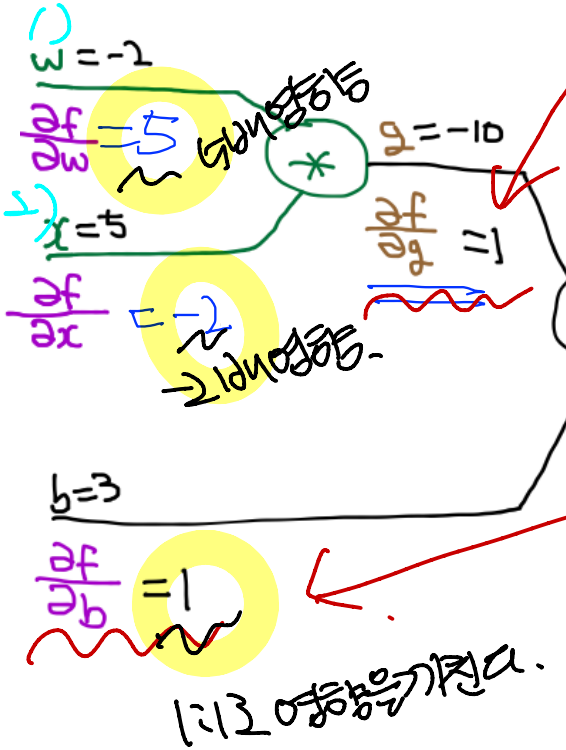
$f = wx + b$, $g = wx$, $f = g + b$

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

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

① forward ($w = -2, x = 5, b = 3$)

② backward




http://cs231n.stanford.edu/

# Back propagation (chain rule)

$f = wx + b$, $g = wx$, $f = g + b$

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

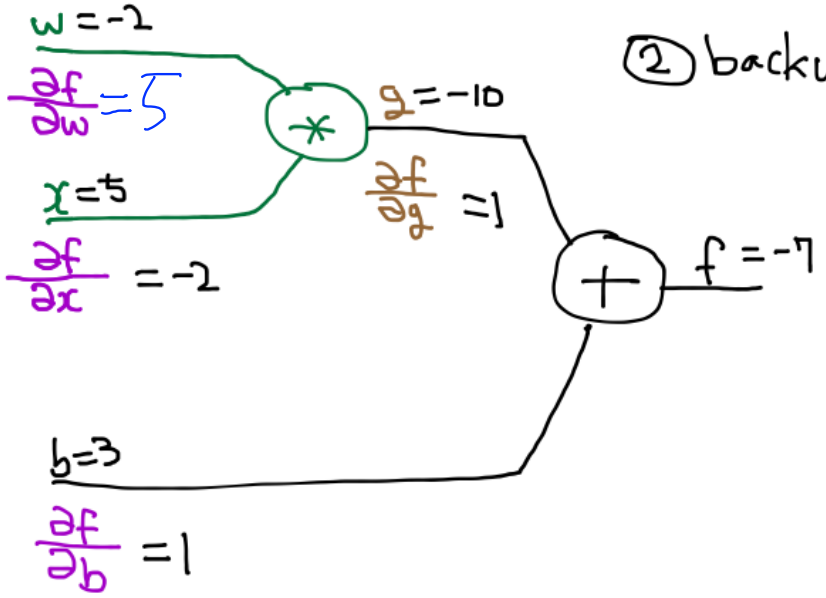
$\frac{\partial g}{\partial w} = x$, $\frac{\partial g}{\partial x} = w$

① forward ($w = -2, x = 5, b = 3$)

② backward

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

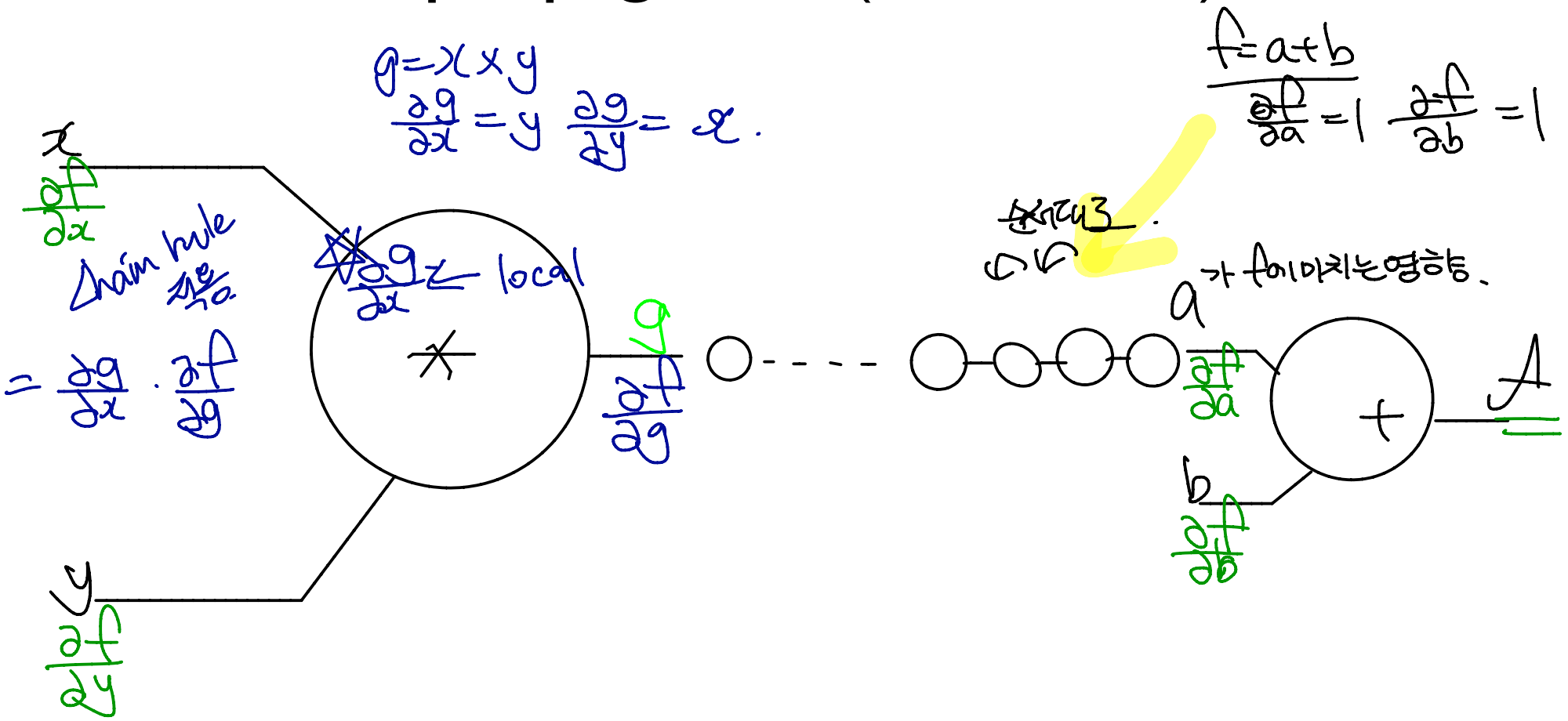
$\frac{\partial f}{\partial w} = \frac{\partial f}{\partial g}\frac{\partial g}{\partial x} = 1 * x = 5$




http://cs231n.stanford.edu/

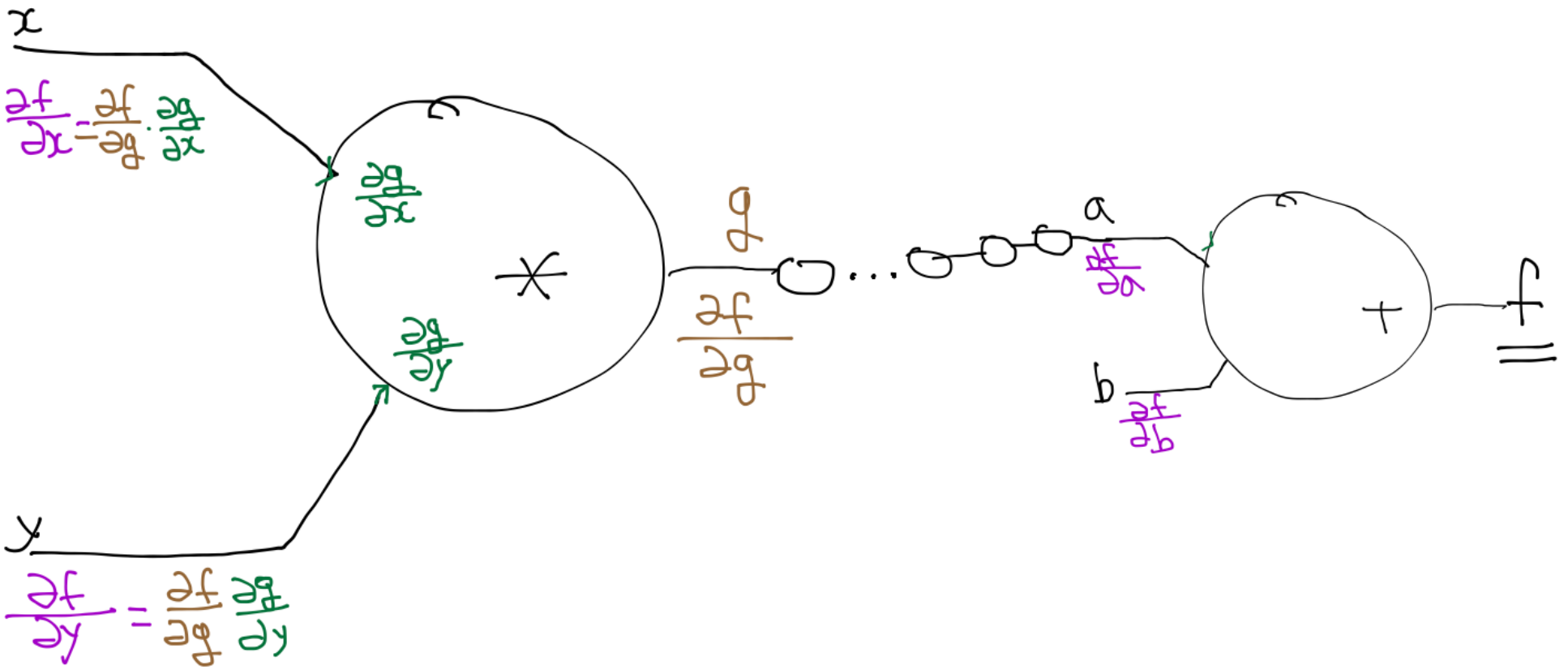
· 순서대로 차례로 수행하면된다

# Back propagation (chain rule)




http://cs231n.stanford.edu/

# Back propagation (chain rule)




http://cs231n.stanford.edu/

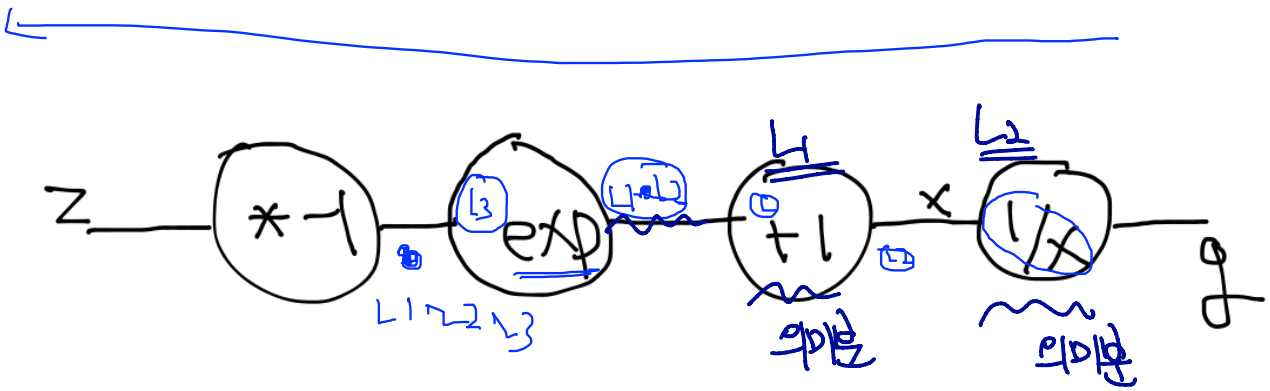
# Sigmoid

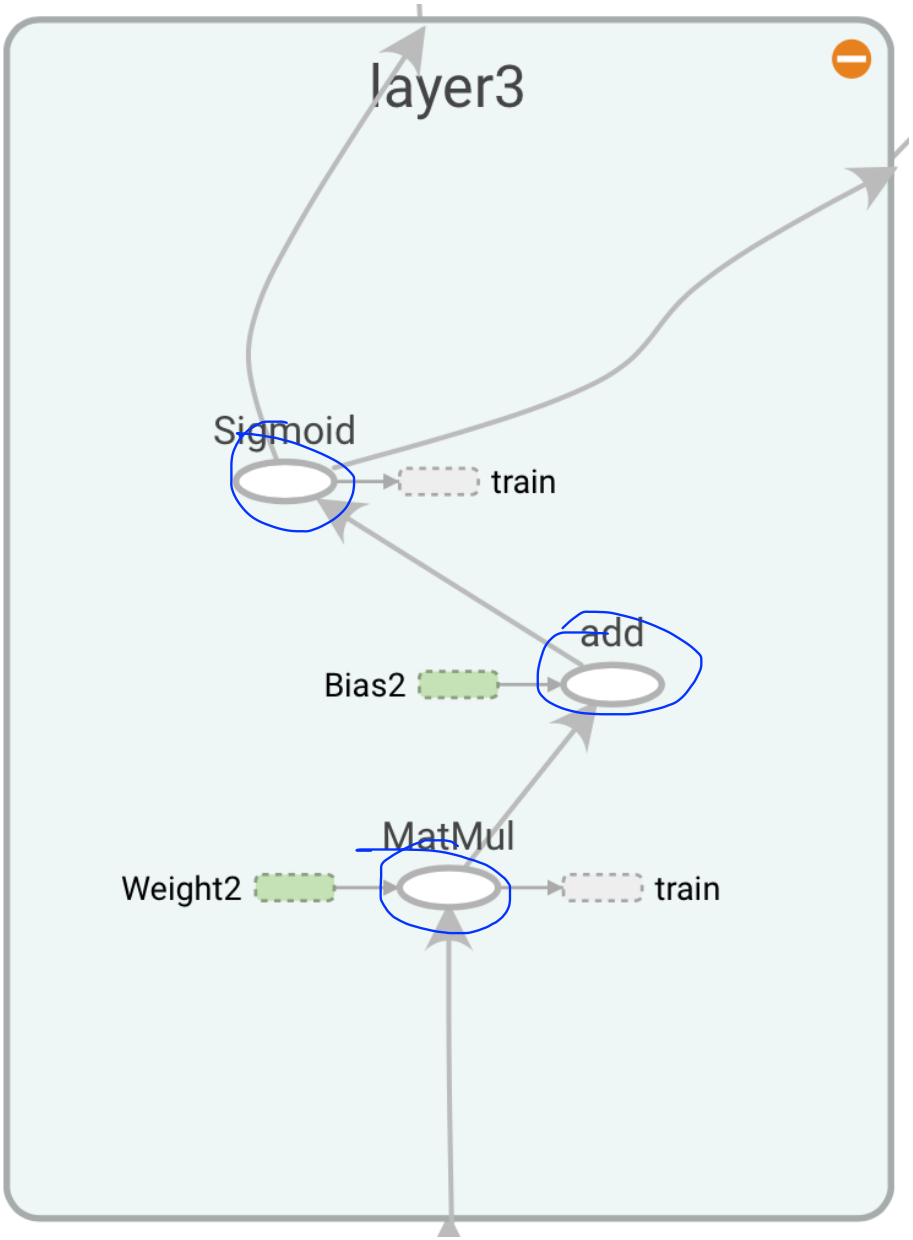
$$g(z) = \frac{1}{1+e^{-z}}$$

$$\frac{\partial g}{\partial z}$$

# Sigmoid

$$g(z) = \frac{1}{1+e^{-z}} \qquad \frac{\partial g}{\partial z}$$

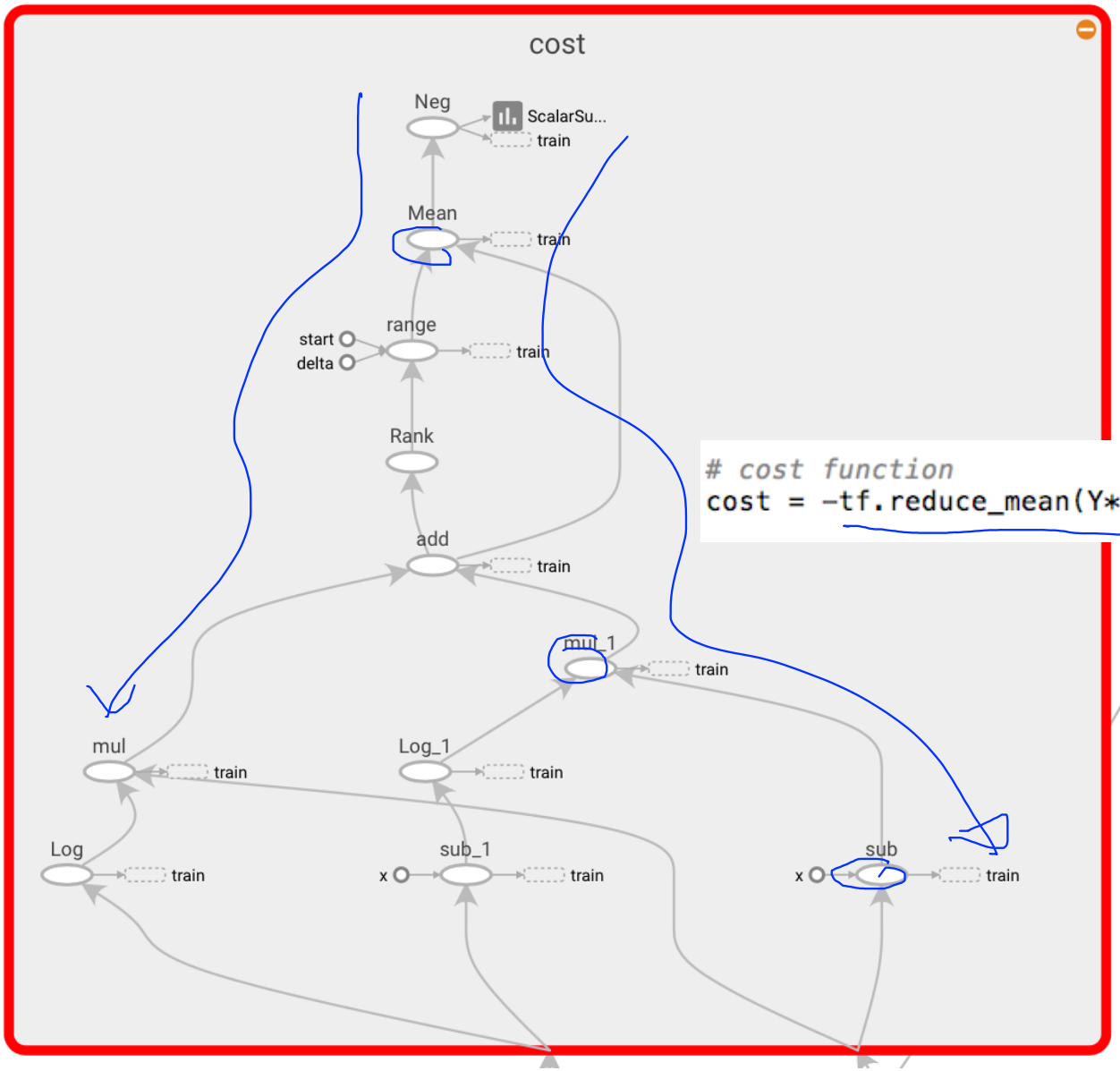





# Back propagation in TensorFlow

## TensorBoard

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

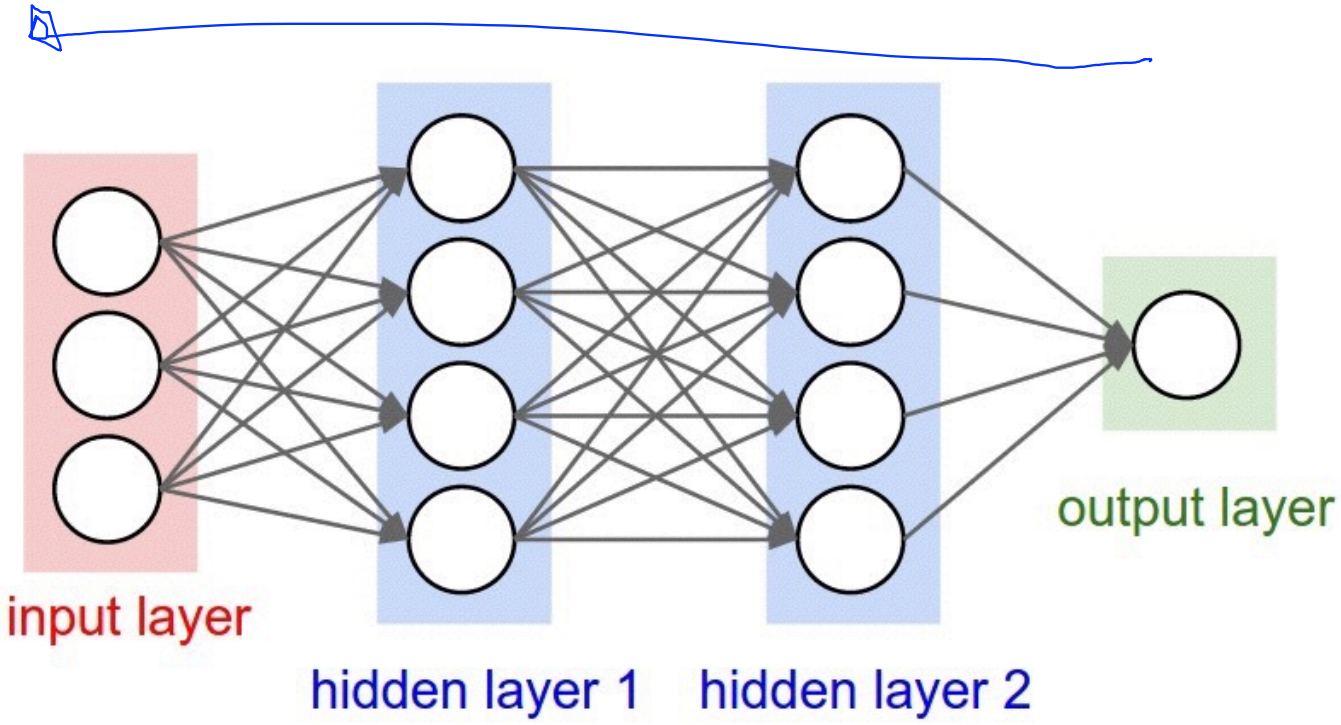



# Back propagation in TensorFlow

## TensorBoard

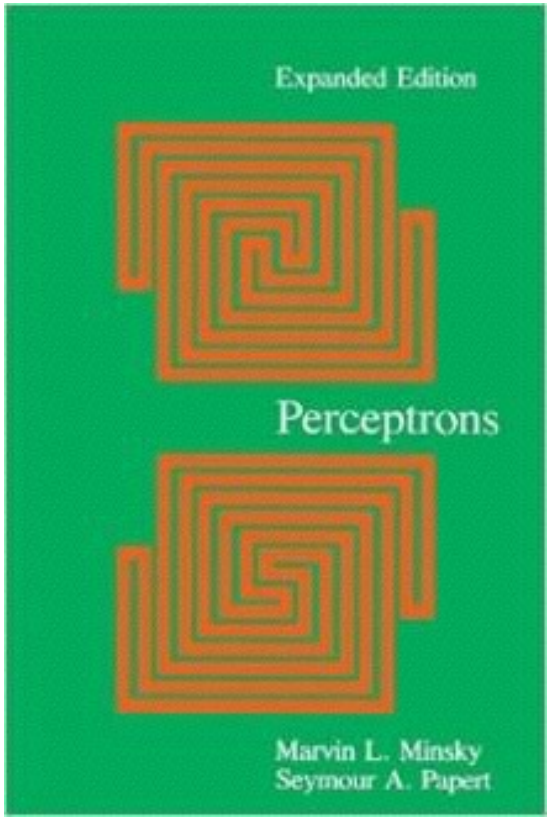
```
# cost function
cost = -tf.reduce_mean(Y*tf.log(hypothesis) + (1-Y)*tf.log(1-hypothesis))
```

# 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.

Next
ReLU