

AI and Deep Learning

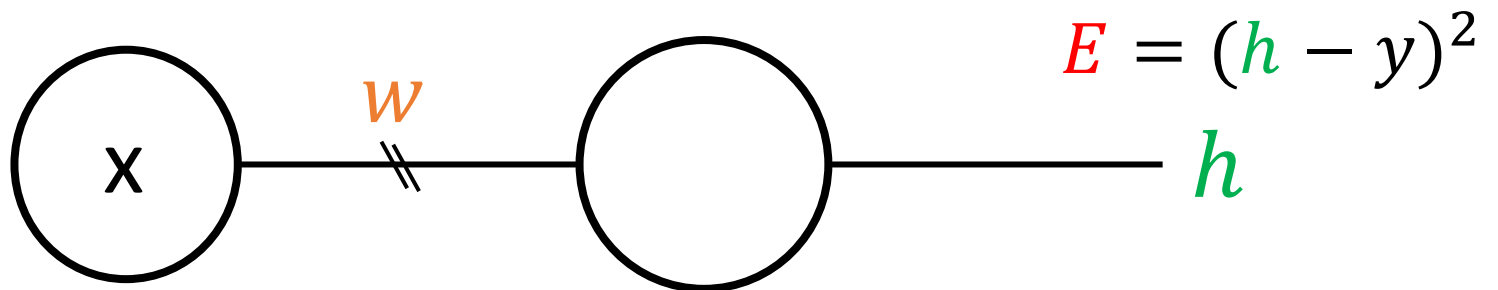
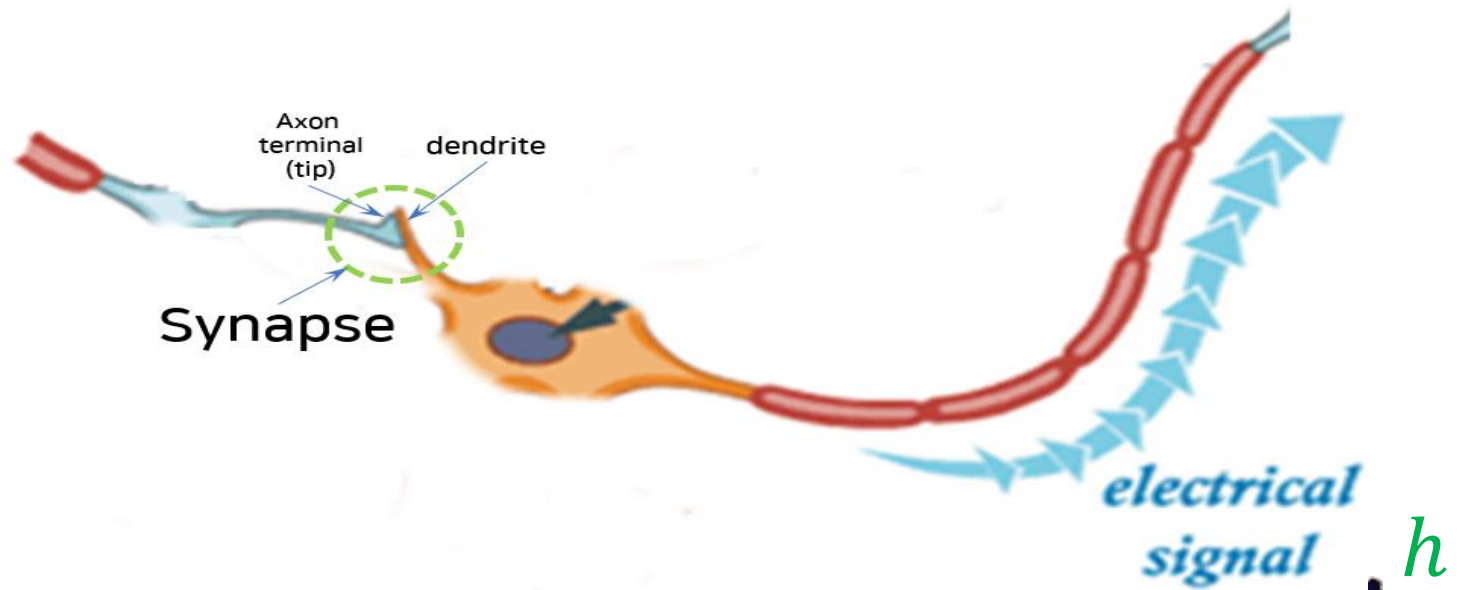
# 로지스틱 회귀와 분류(1)

- 결정 경계 -

제주대학교

변영철

<http://github.com/yungbyun/mllecture>



# Logistic Regression

The shape of regression is **not linear but logistic**.

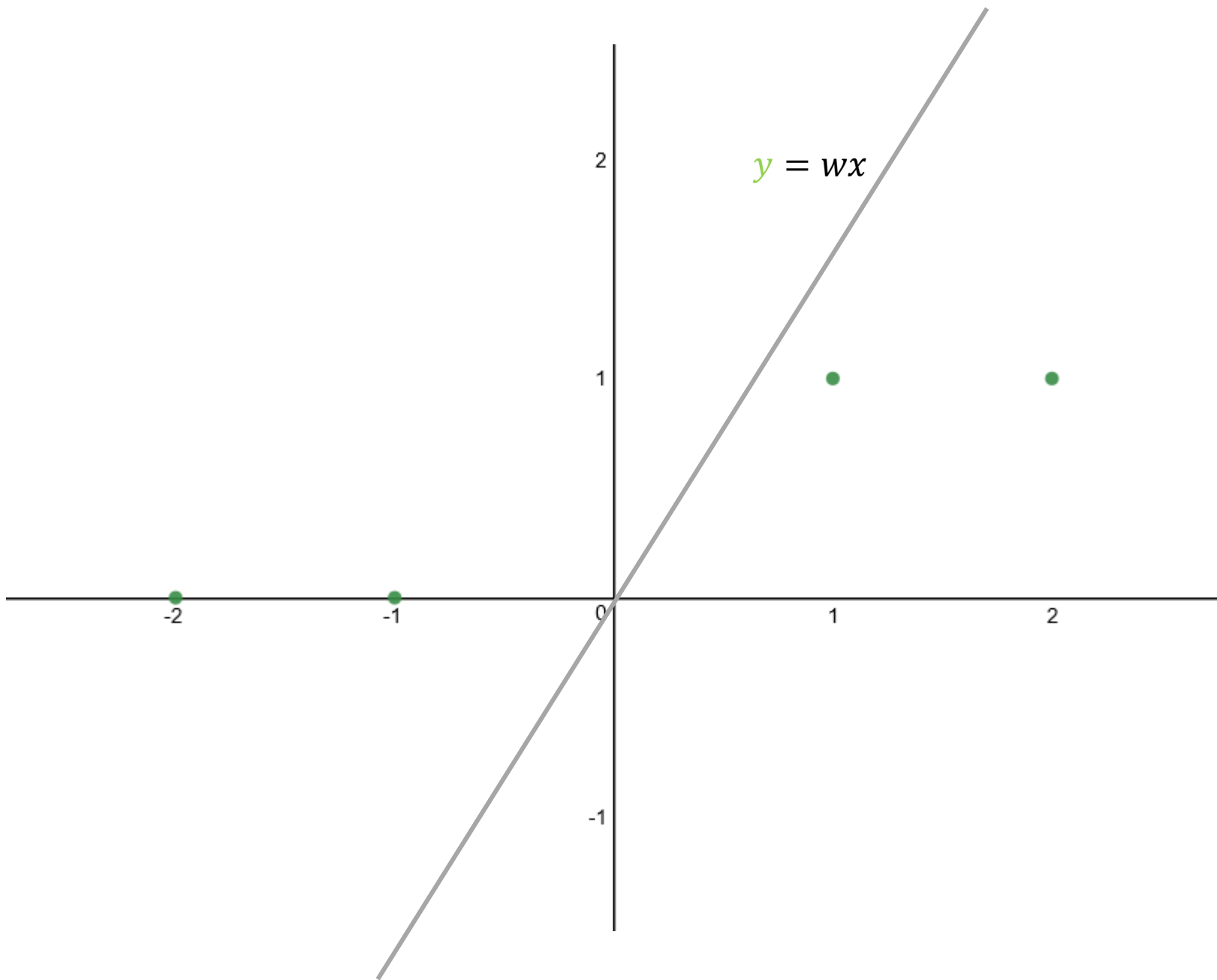
What does that mean?

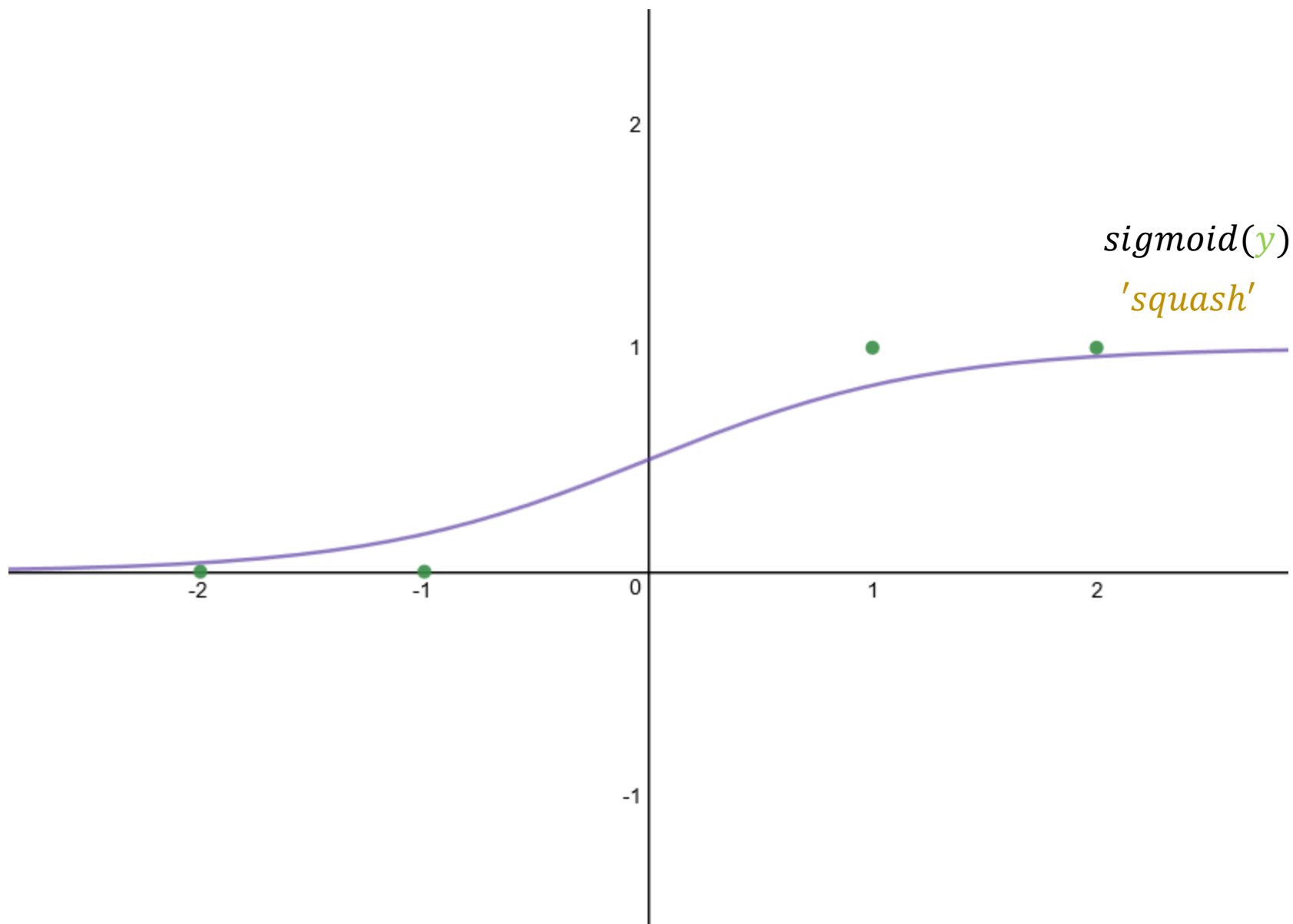


점  $(-2, 0)$ ,  $(-1, 0)$ ,  $(1, 1)$ ,  $(2, 1)$  표시

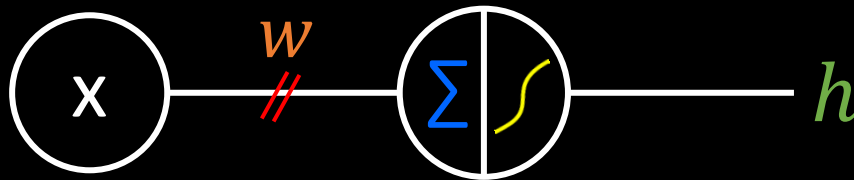
$$y = wx$$

$$y = \frac{1}{1 + e^{-wx}}$$







# 신경 세포 기능



- 신경세포 1개가 할 수 있는 것은?
- 입력  $x$ 에 따라 0, 혹은 1(fire)을 출력함.
- 0 혹은 1을 결정하는 기준은?

$x$	 $y$
-2	0
-1	0
1	1
2	1

 $y$	0	0	1	1
$x$	-2	-1	1	2

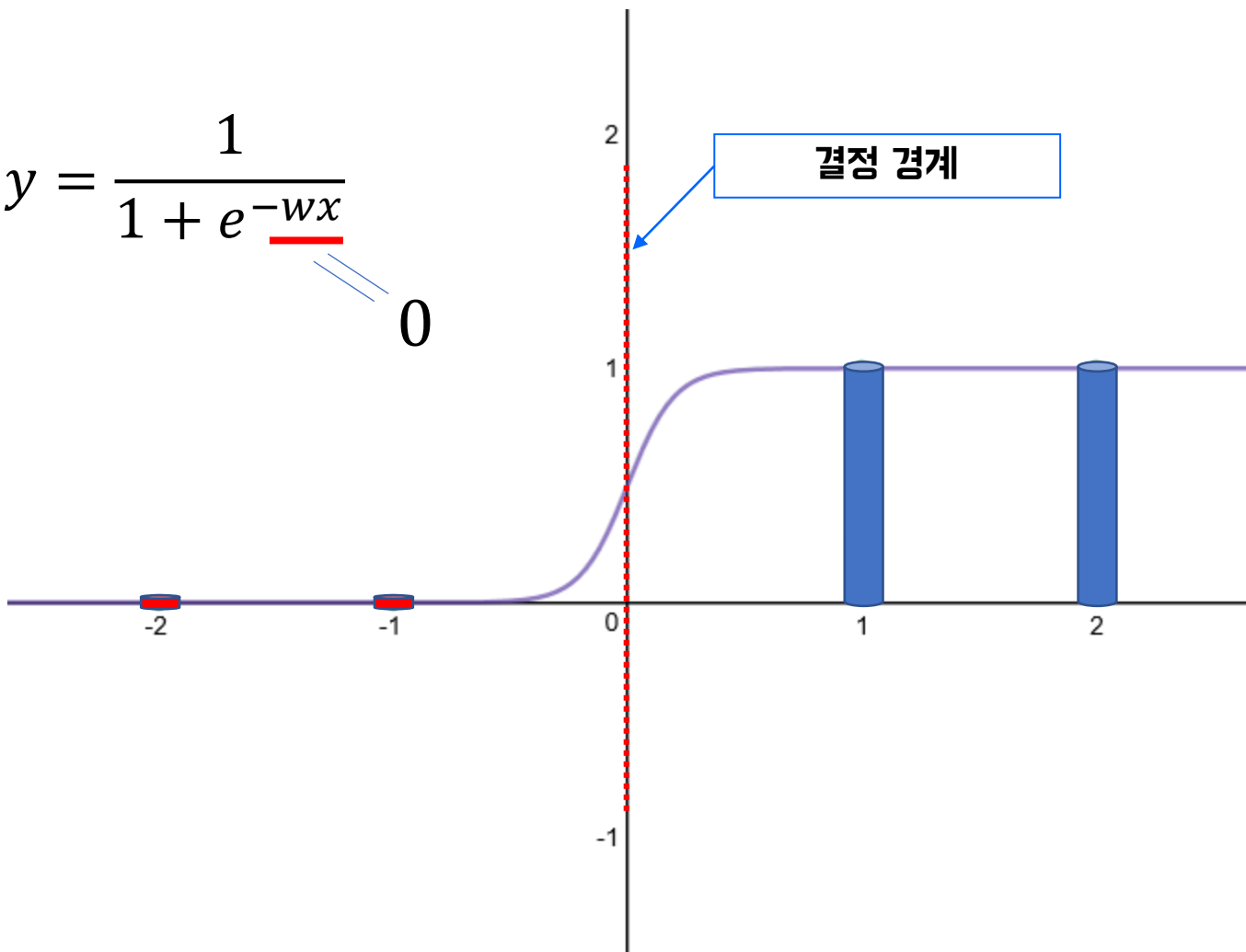




0, 1을 결정(decision)하는  
경계(boundary)

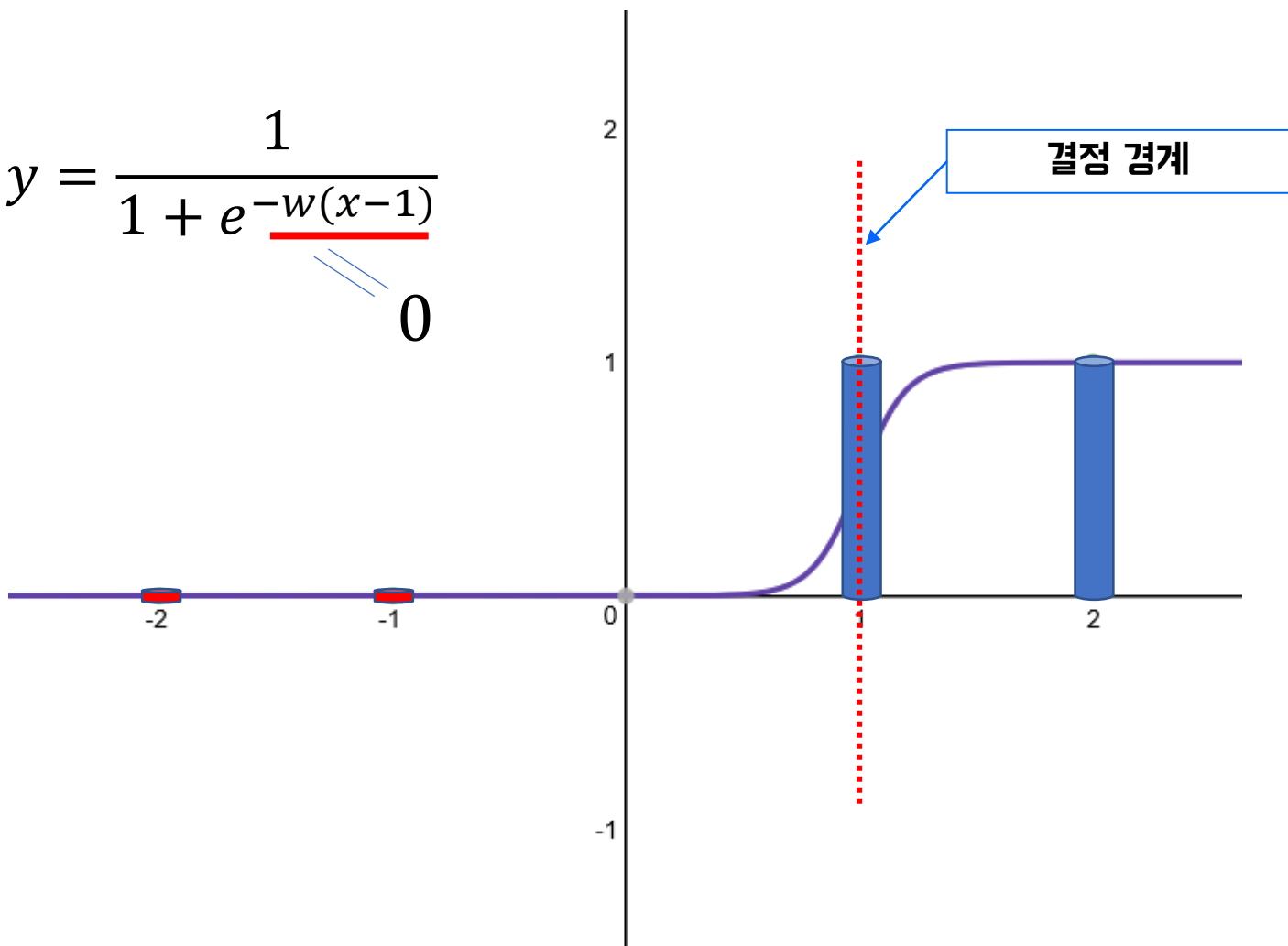
$$y = \frac{1}{1 + e^{-wx}}$$

0



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

$\underbrace{-w(x-1)}_{=0}$



# 결정 경계

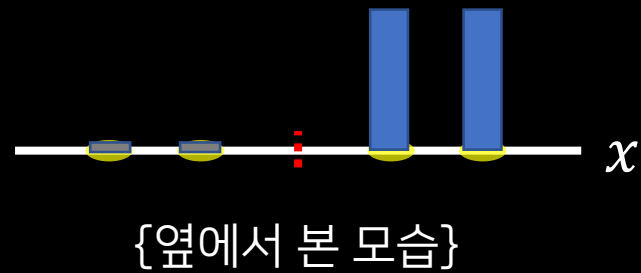
$$wx = 0$$

$$x = 0$$

$$wx + b = 0$$

$$x + 1 = 0$$

$$2x + 3 = 0$$

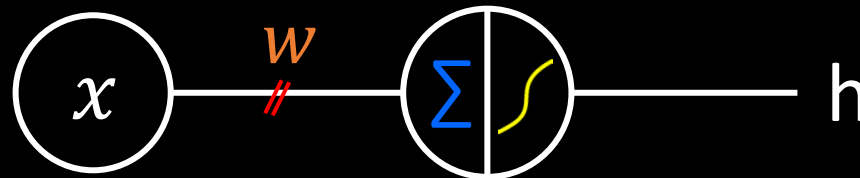


# Classification

- Pass(1) or Fail(0)
- Spam(1) or Ham(0)
- Scam(fraud, 1) or not(0)
- Safe(1) or Dangerous(0)
- Intrusion/virus(1) or not(0)
- Cancer(1) or not(0)
- Binary classification ->  
Multiple classification

뉴런 그림만 보고 결정 경계를 알 수 있을까?

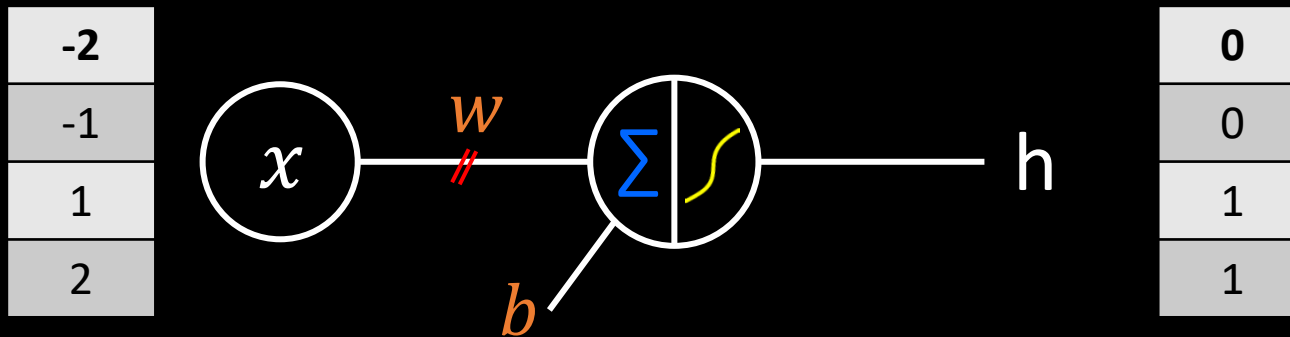
-2
-1
1
2



$y$
0
0
1
1

$$h = \begin{cases} 1 & \text{if } wx \geq 0 \\ 0 & \text{otherwise} \end{cases}$$

뉴런 그림만 보고 결정 경계를 알 수 있을까?



$$h = \begin{cases} 1 & \text{if } wx + b \geq 0 \\ 0 & \text{otherwise} \end{cases}$$

# 가설

$$hypo = \frac{1}{1 + e^{-wx}}$$

가설은 뭐다?

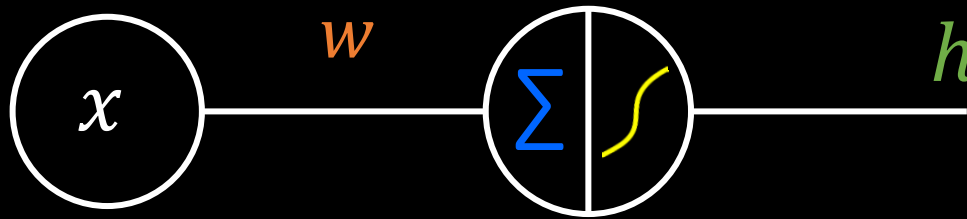
가설 안에 결정 경계



# 선형 회귀 오류 함수



$$E = (h - y)^2$$



# 선형 회귀 오류 함수

$$E = \frac{1}{m} \sum_{i=1}^m (h_i - y_i)^2$$

Prediction by a neuron

Correct answer

“로지스틱 리그레션에도 동작할까?”



점  $(-2, 0)$ ,  $(-1, 0)$ ,  $(1, 1)$ ,  $(2, 1)$  표시

$$y = wx$$

$$y = \frac{1}{1 + e^{-wx}}$$

점  $(1, 1)$ 만 표시

$$E = \left( \frac{1}{1 + e^{-w \cdot 1}} - 1 \right)^2$$

$(w, E)$



점  $(-1, 0)$ ,  $(1, 1)$  표시

$$h = \frac{1}{1 + e^{-wx}}$$

$$E = \left( \frac{1}{1 + e^{-w \cdot -1}} - 0 \right)^2 + \left( \frac{1}{1 + e^{-w \cdot 1}} - 1 \right)^2$$

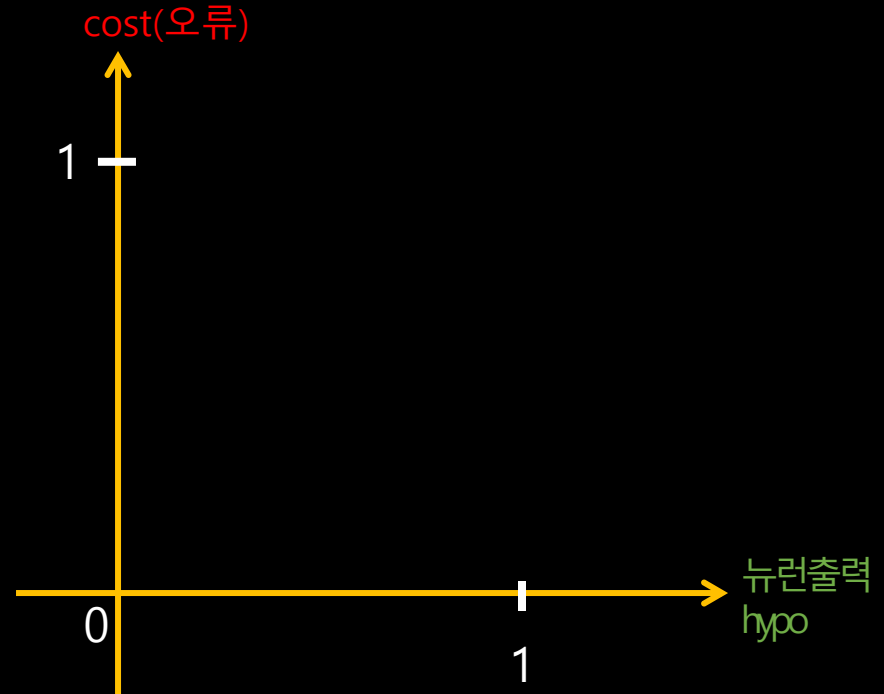
$(w, E)$

에러 그래프 모양 vs. 바람직한 모양  
은???

동작하지 않는다.  
경사하강이 불가능

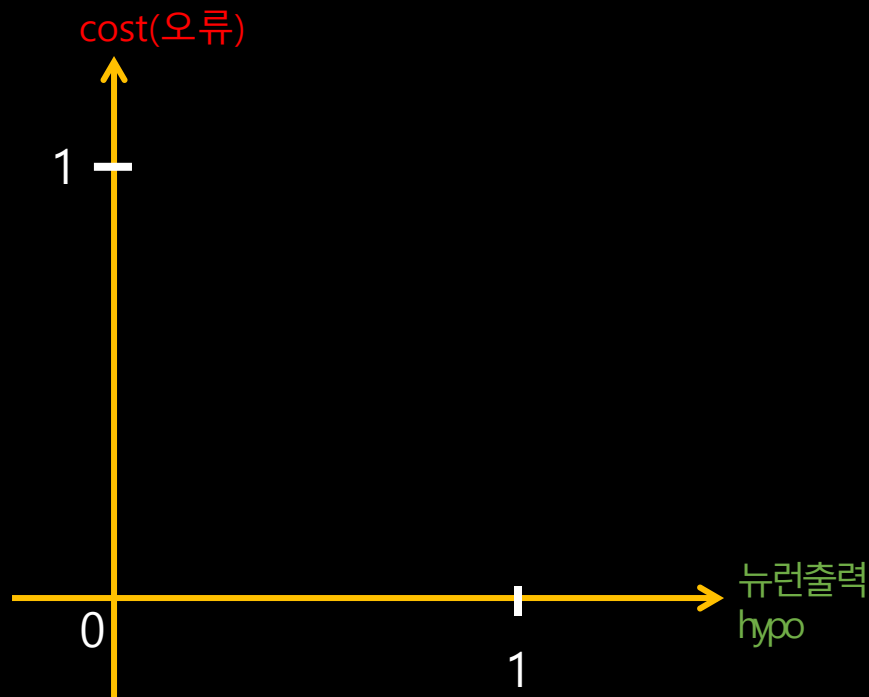
# 새로운 오류 함수

- 정답이 1일 때
  - 뉴런이(hypo) 정답을 맞추면(1이면) 오류는 없다고(0) 알려주고,
  - 뉴런이(hypo) 정답과는 정반대로 대답하면(0이면) 오류는 크다고( $\infty$ ) 알려주자.



# 새로운 오류 함수

- 정답이 0일 때
  - 뉴런이(hypo) 정답을 맞추면(0이면) 오류는 0
  - 뉴런이(hypo) 정답과는 정반대로 대답하면(1이면) 오류는  $\infty$ 가 되게



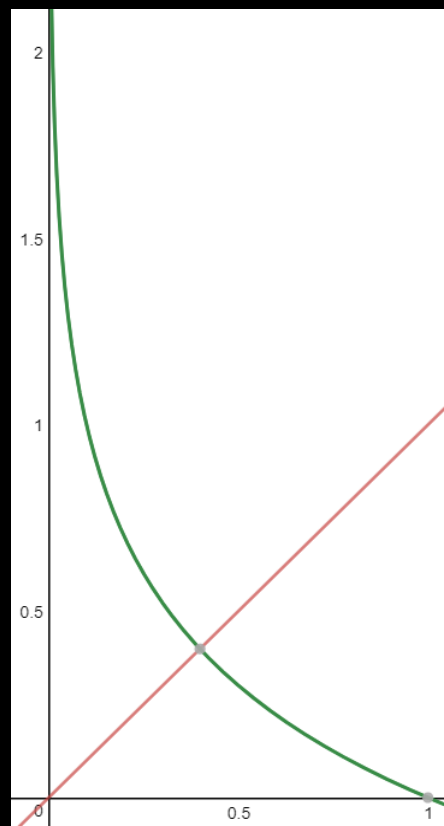




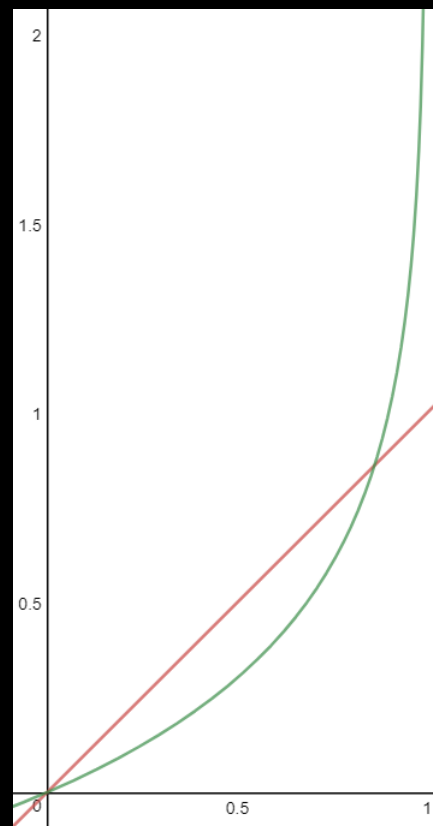
desmos

$h$  : Prediction by a neuron :  $w x$

$$y = -\log(h)$$



$$y = -\log(1 - h)$$

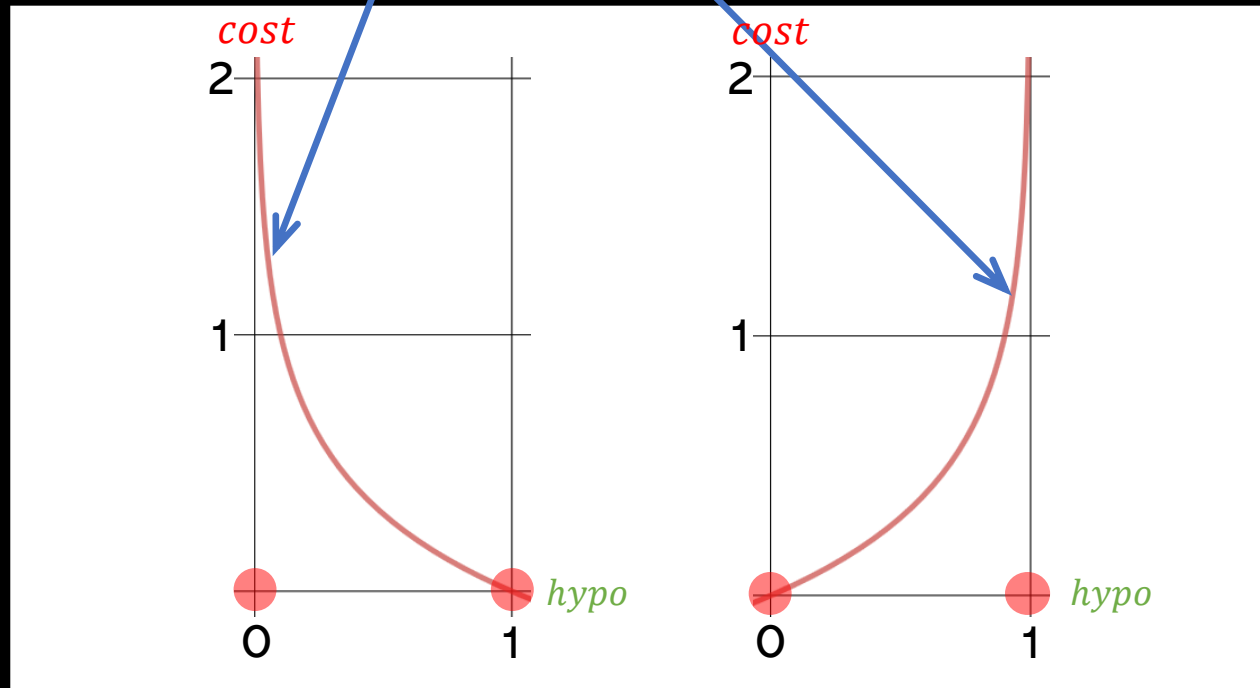


# Error Function

Prediction by a neuron

Correct answer

$$\text{cost} = \begin{cases} -\log(h) & : y = 1 \\ -\log(1 - h) & : y = 0 \end{cases}$$



# Error Function

$$cost = \begin{cases} -\log(H(X)) & : y = 1 \\ -\log(1 - H(X)) & : y = 0 \end{cases}$$

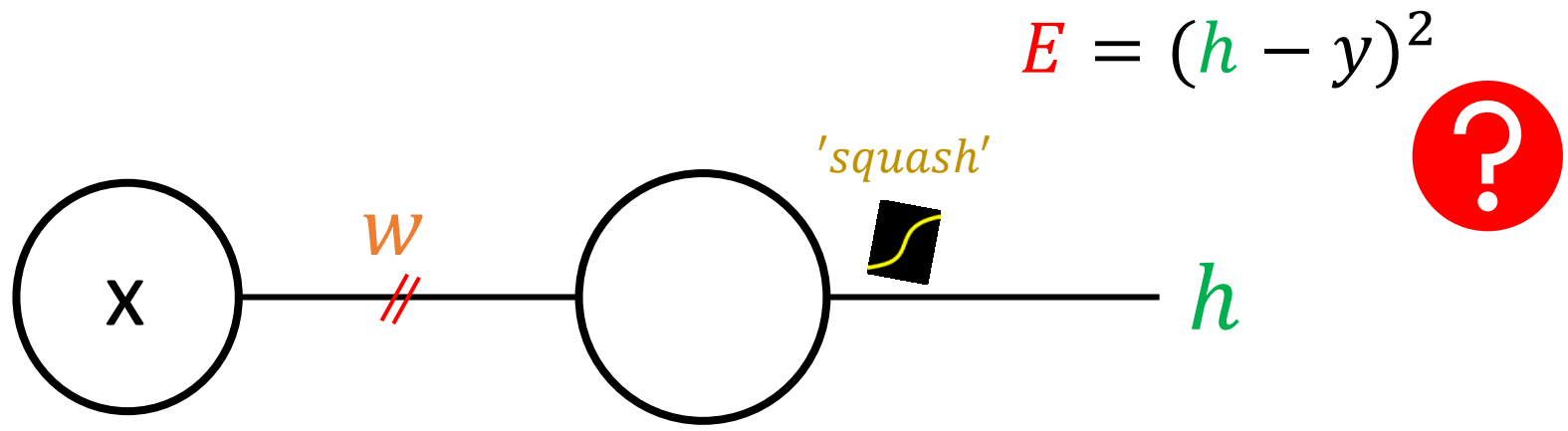


$$E = -y \log(H(X)) - (1 - y) \log(1 - H(X))$$

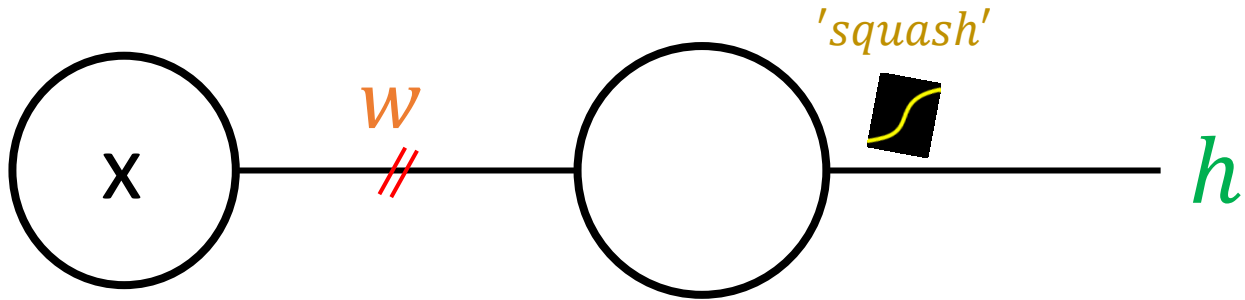
$$E = -(y \log(H(X)) + (1 - y) \log(1 - H(X)))$$

$$w = w - \alpha \cdot Gradient$$

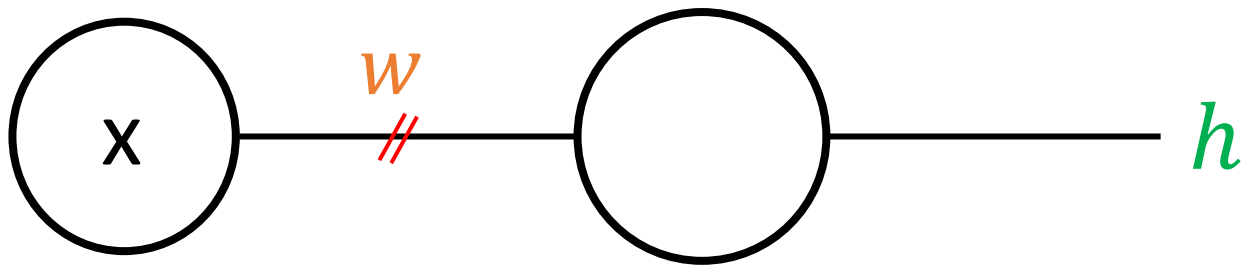
$$\frac{\partial E}{\partial w}$$



$$E = -(y \log(h) + (1 - y) \log(1 - h))$$



$$E = (h - y)^2$$

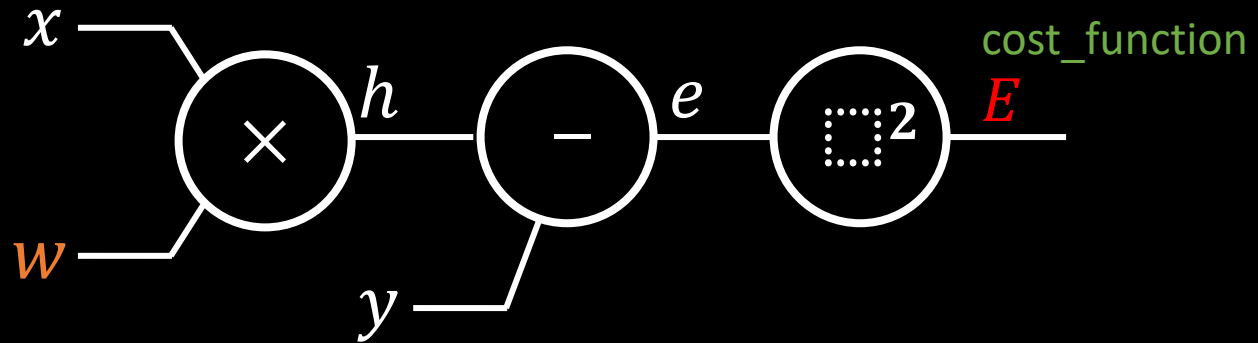


# Computational Graph

$$E = (wx - y)^2$$

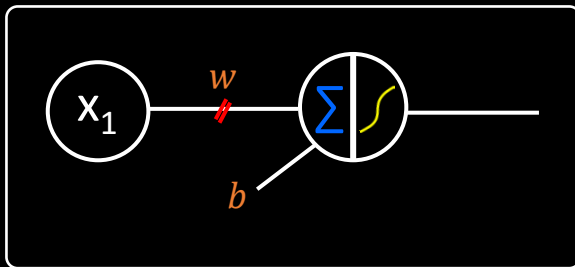
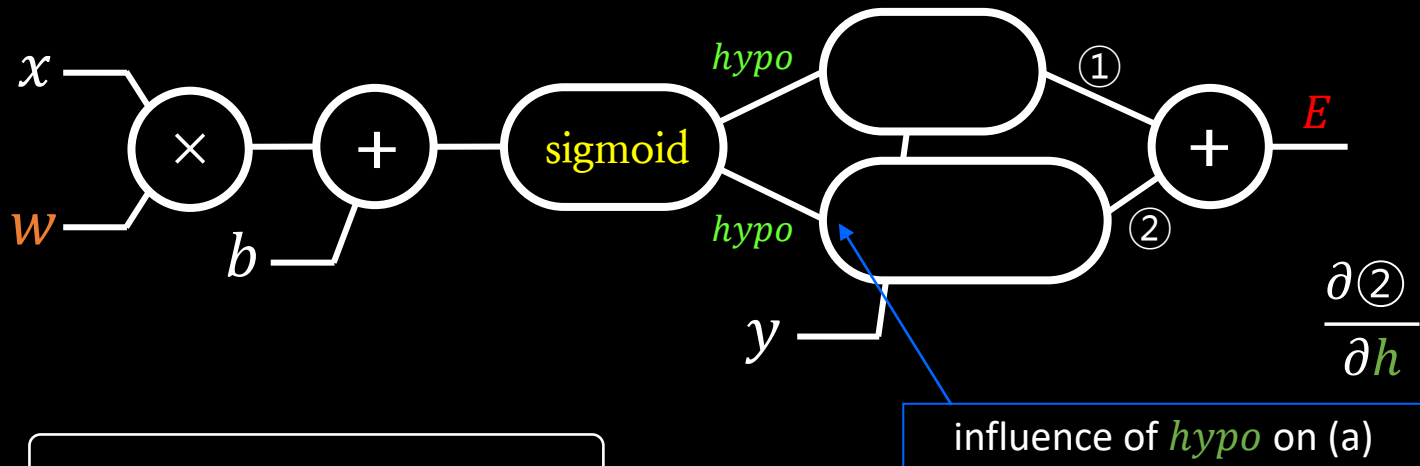
hypo = w \* x

cost\_function( $E$ ) = (hypo - y) \*\* 2



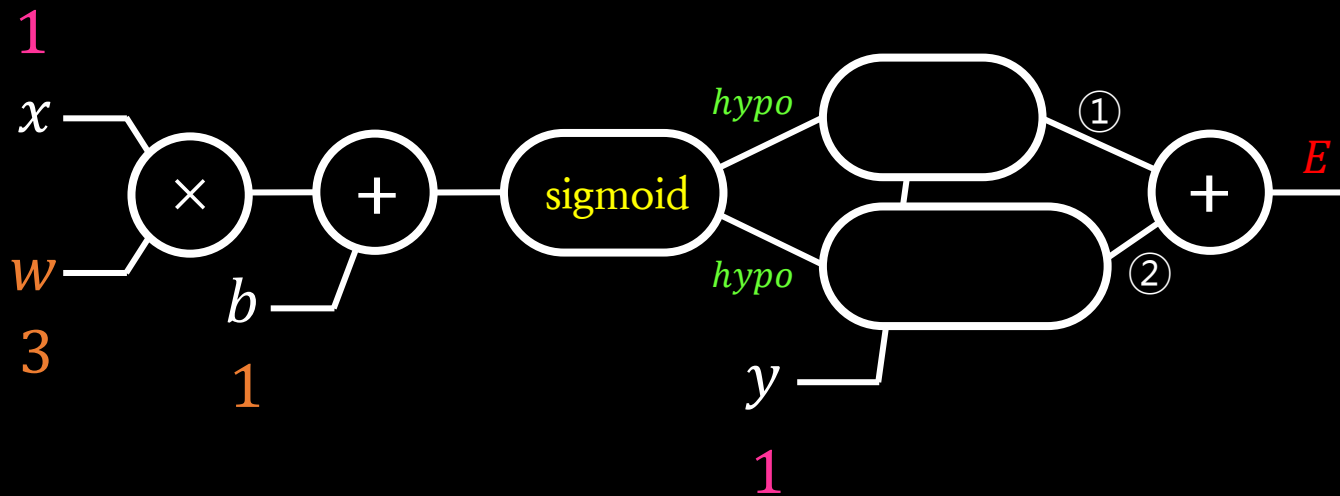
# Computational Graph

$$E = \overset{\textcircled{1}}{-(y \cdot \log(\textit{hypo}))} + \overset{\textcircled{2}}{(1 - y) \cdot \log(1 - \textit{hypo}))}$$

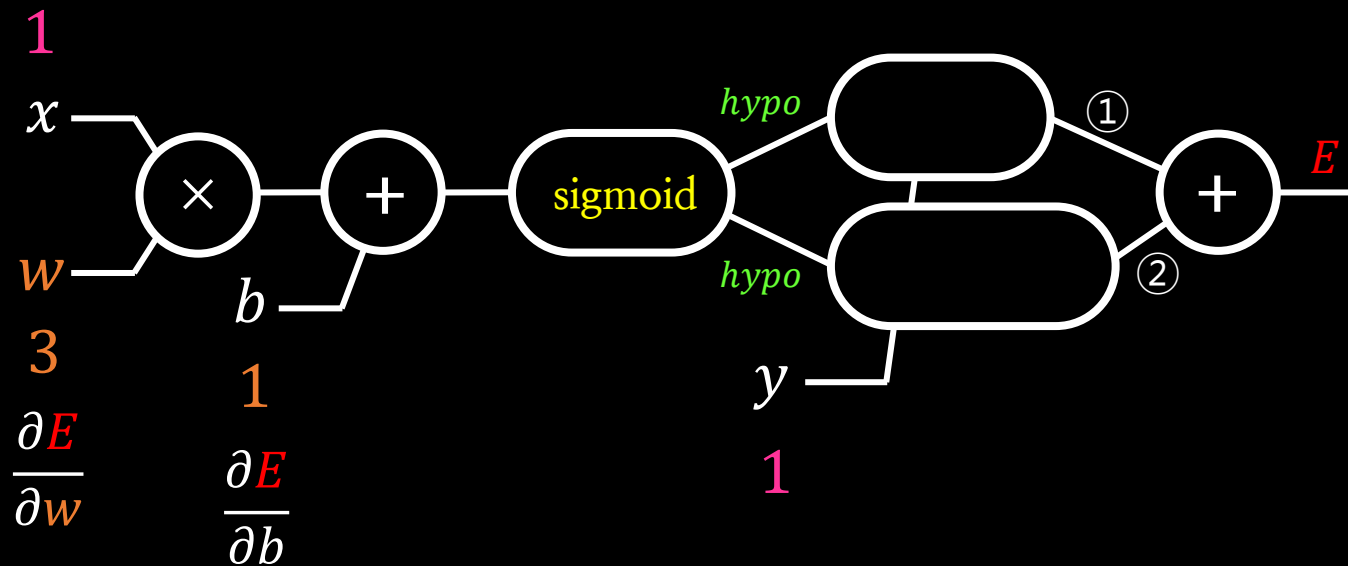




# Forward propagation



# Back-propagation



$$w = w - \alpha \cdot \frac{\partial E}{\partial w}$$

$$b = b - \alpha \cdot \frac{\partial E}{\partial b}$$

# (실습) 11.py

입력이 음수면 0을,

입력이 양수면 1을 출력

$$cost = -(y \log(H(X)) + (1 - y) \log(1 - H(X)))$$

```
x_data = [-2., -1, 1, 2]
y_data = [0., 0, 1, 1]
```

```
#----- a neuron
w = tf.Variable(tf.random_normal([1]))
hypo = tf.sigmoid(x_data * w)
```

```
#----- learning
cost = -tf.reduce_mean(y_data * tf.log(hypo) +
                        tf.subtract(1., y_data) * tf.log(tf.subtract(1., hypo)))

train = tf.train.GradientDescentOptimizer(learning_rate=0.01).minimize(cost)

sess = tf.Session()
sess.run(tf.global_variables_initializer())

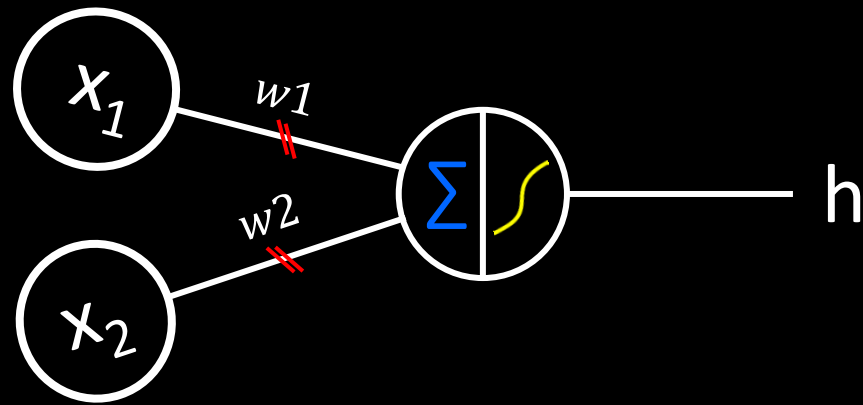
for step in range(5001):
    sess.run(train)
```

```
#----- testing(classification)
predicted = tf.cast(hypo > 0.5, dtype=tf.float32)
p = sess.run(predicted)
print("Predicted: ", p)
```

(실습) 12.py

바이어스를 갖는 뉴런

# 신경 세포 (2 입력)



$$h = \frac{1}{1 + e^{-(w_1 x_1 + w_2 x_2)}}$$

# 신경 세포 (2 입력)

- 결정 경계는?

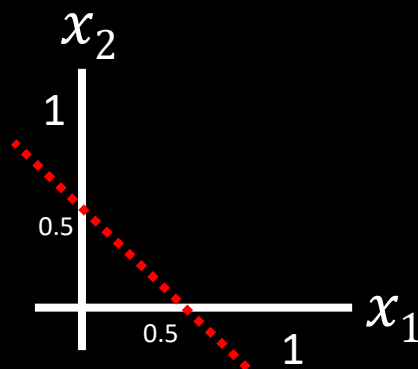
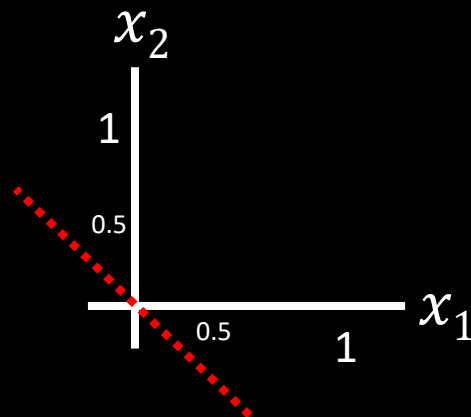
$$w_1x_1 + w_2x_2 = 0$$

$$x_1 + x_2 = 0$$

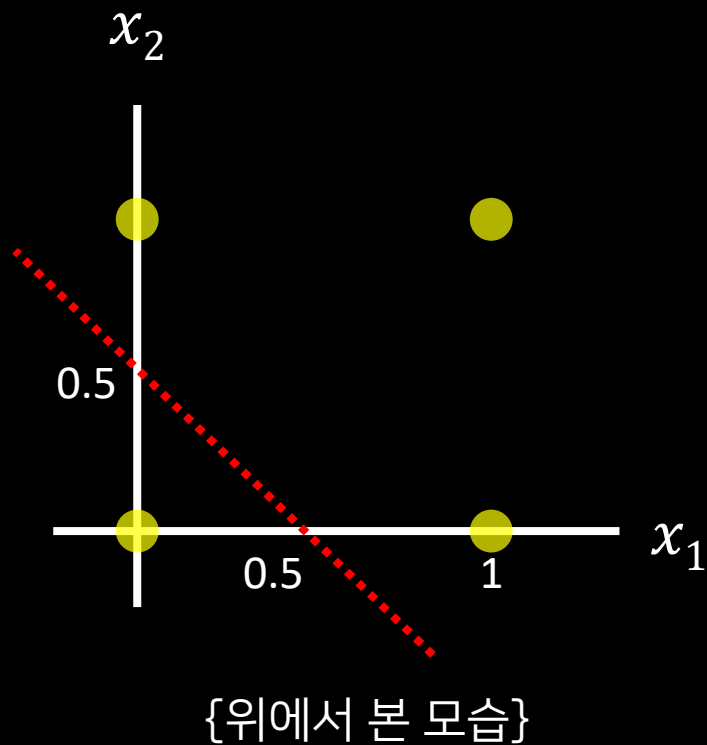
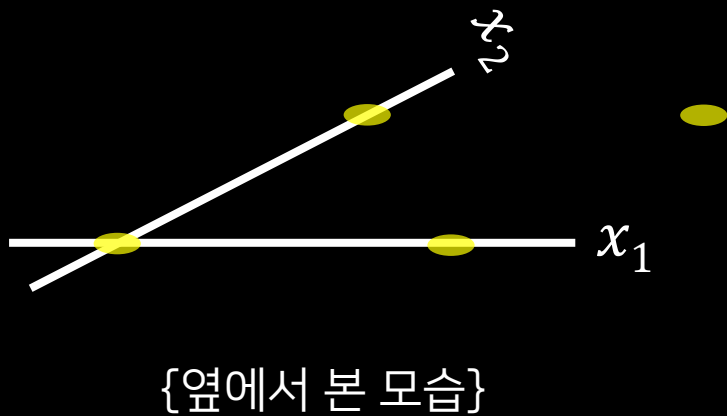
$$w_1x_1 + w_2x_2 = b$$

$$x_1 + x_2 = 0.5$$

{위에서 본 모습}

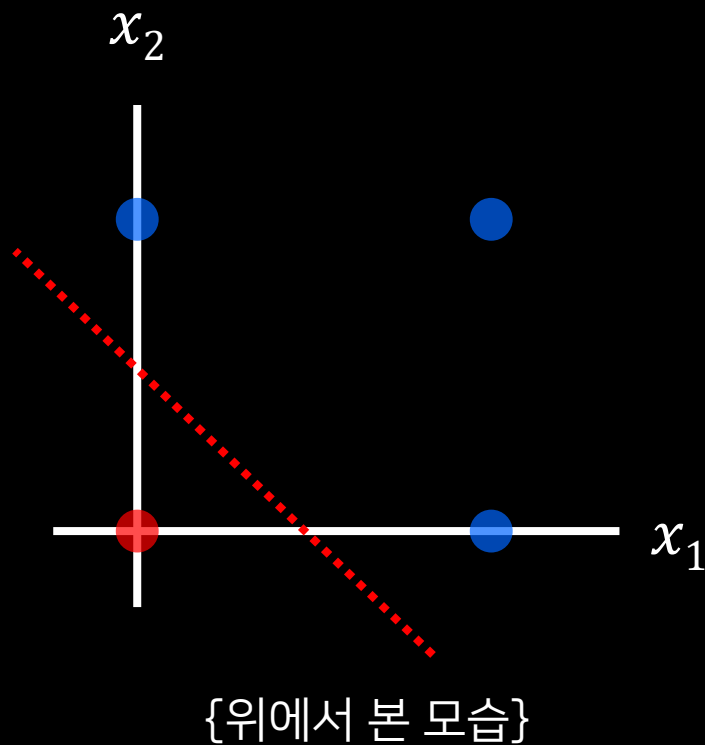
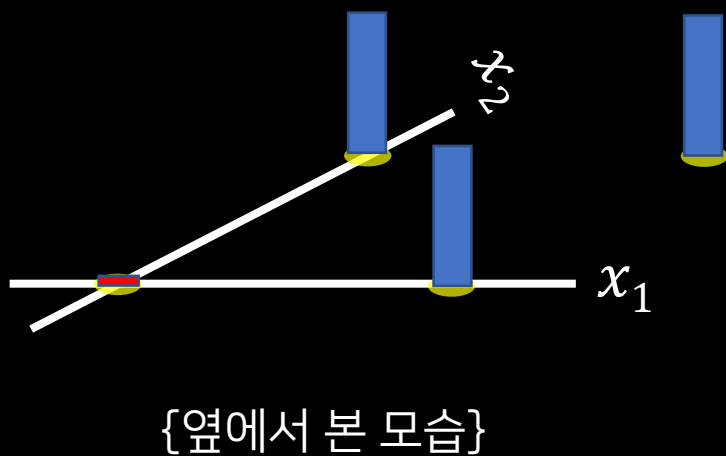


# 신경 세포 (2 입력)





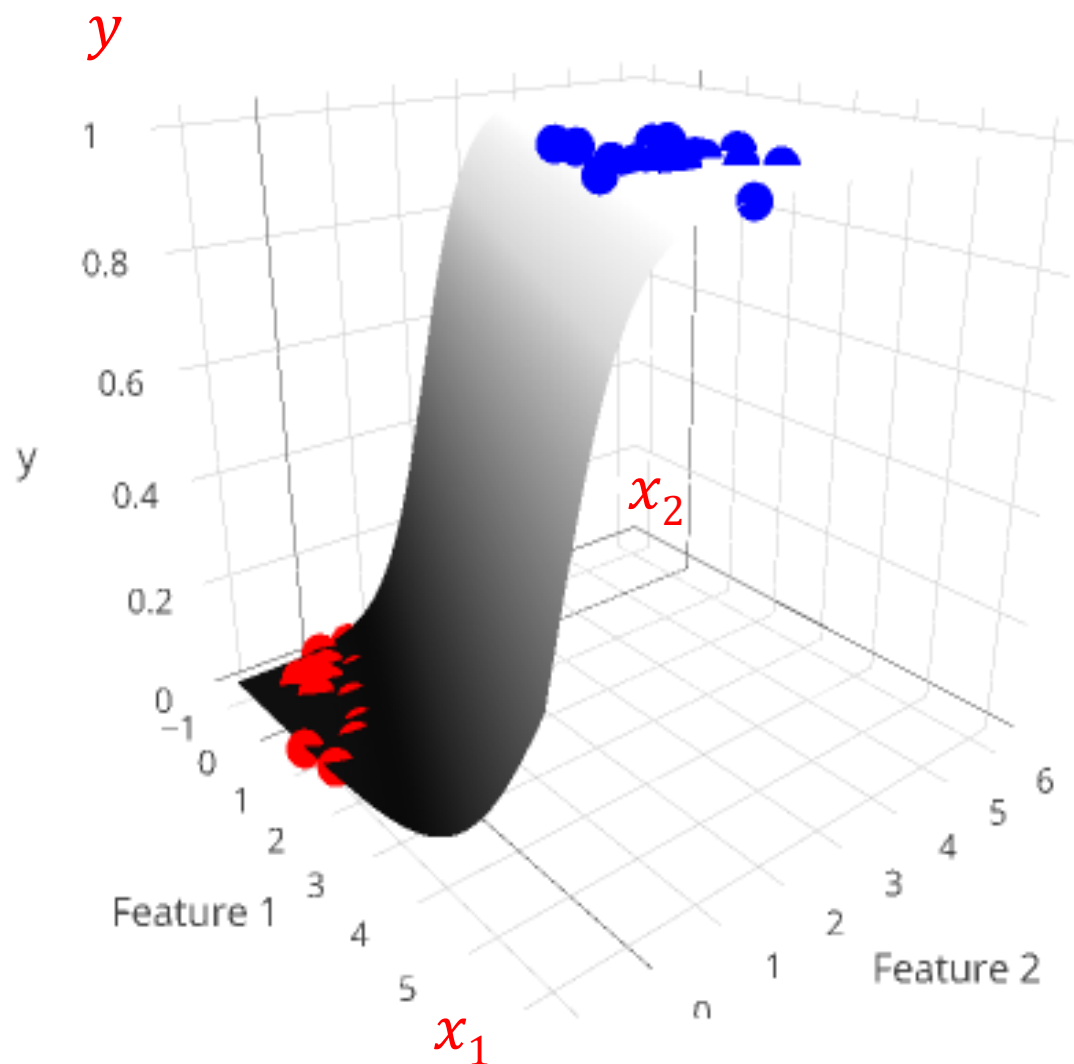
# 신경 세포 (2 입력)



{옆에서 본 모습}

## Logistic Regression: 2 Features

미끄럼틀



{위에서 본 모습}

$x_2$

3

2

1

-4

-3

-2

-1

0

1

2

3

4

$x_1$

입력이 2개인 신경 세포 1개는  
이런 **결정 경계**를 만든다!

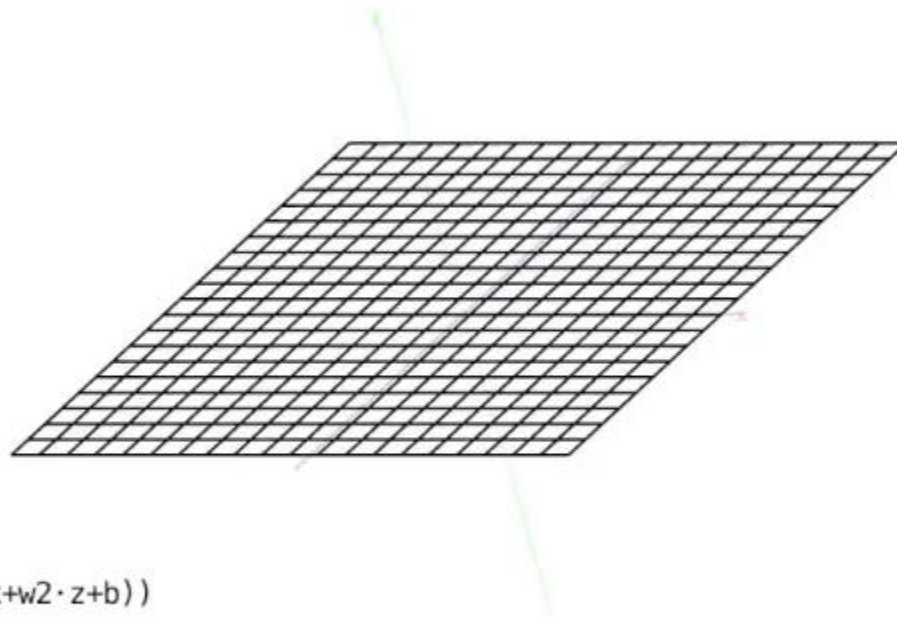
-1

-2

-3

# Decision Boundary in 3D

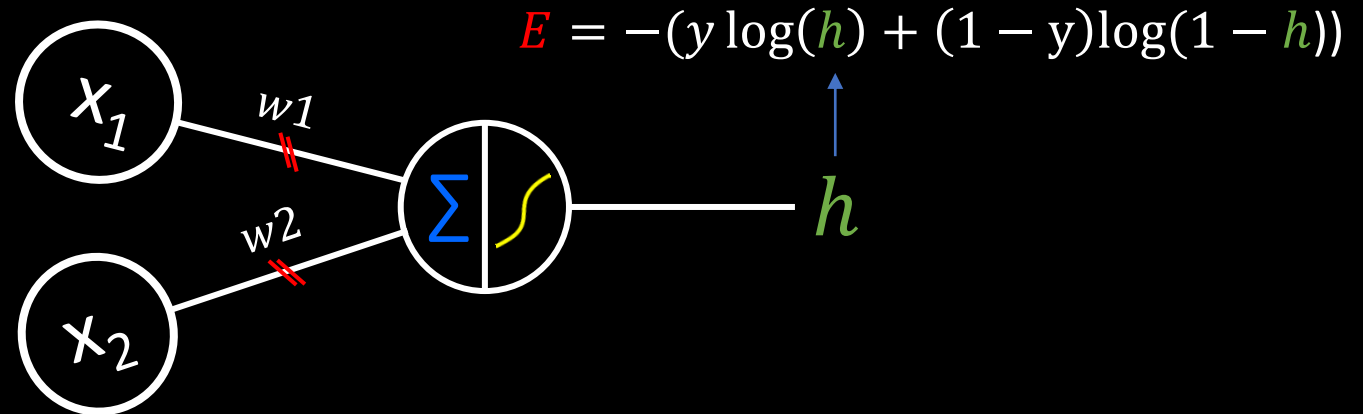
```
sigmoid(w1·length + w2·width + b)
```



```
surface(f(x,z)=sig(w1·x+w2·z+b))  
w1 = 0.00  
w2 = 0.00  
b = 0.00
```

## (실습) 13.py

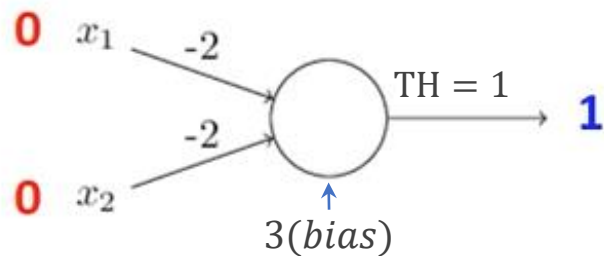
- 입력 두개( $x_1, x_2$ )를 갖는 뉴런을 이용하여 OR 맞추기
- 한 개의 결정 경계



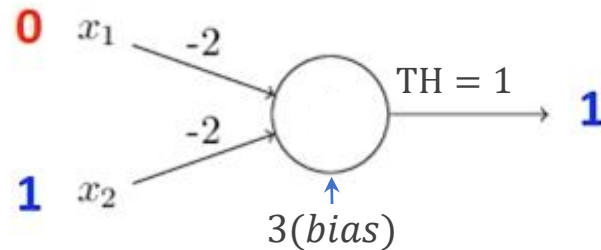
$x_1$	$x_2$	$AND(h)$
0	0	0
0	1	0
1	0	0
1	1	1

# NAND

- NAND gates are functionally complete.
- We can build any logical functions out of them.



$$(-2)*0 + (-2)*0 + 3 = 3$$

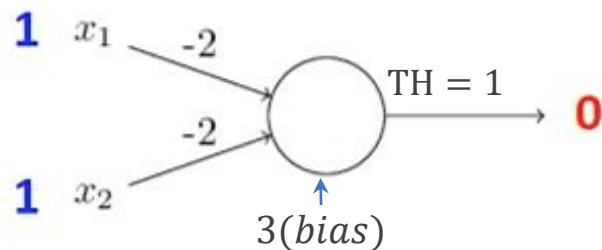


$$(-2)*0 + (-2)*1 + 3 = 1$$

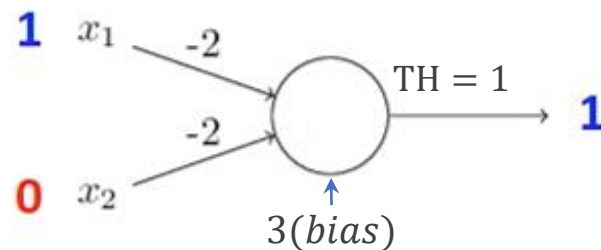
NAND

Truth Table

Input A	Input B	Output Q
0	0	1
0	1	1
1	0	1
1	1	0



$$(-2)*1 + (-2)*1 + 3 = -1$$



$$(-2)*1 + (-2)*0 + 3 = 1$$



# 이번 학습에서는

- 로지스틱 리그레션을 위한 오류함수를 어떻게 디자인 하는지 이해할 수 있다.
- 한 개의 뉴런이 만들어 내는 결정 경계를 이해할 수 있다.
- 결정경계를 옆에서 본 모습, 위에서 본 모습을 이해할 수 있다.
- 신경세포의 입력의 수에 따른 결정경계를 이해할 수 있다.