

AI and Deep Learning

Logistic Regression (1)

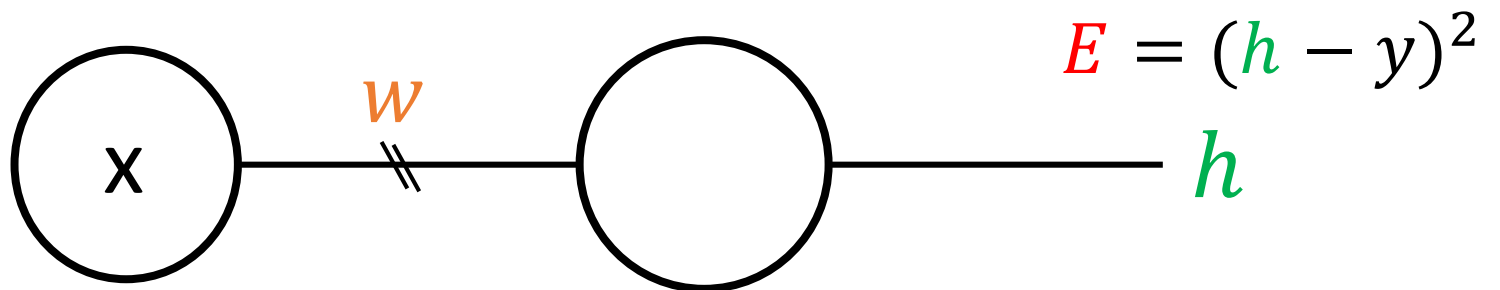
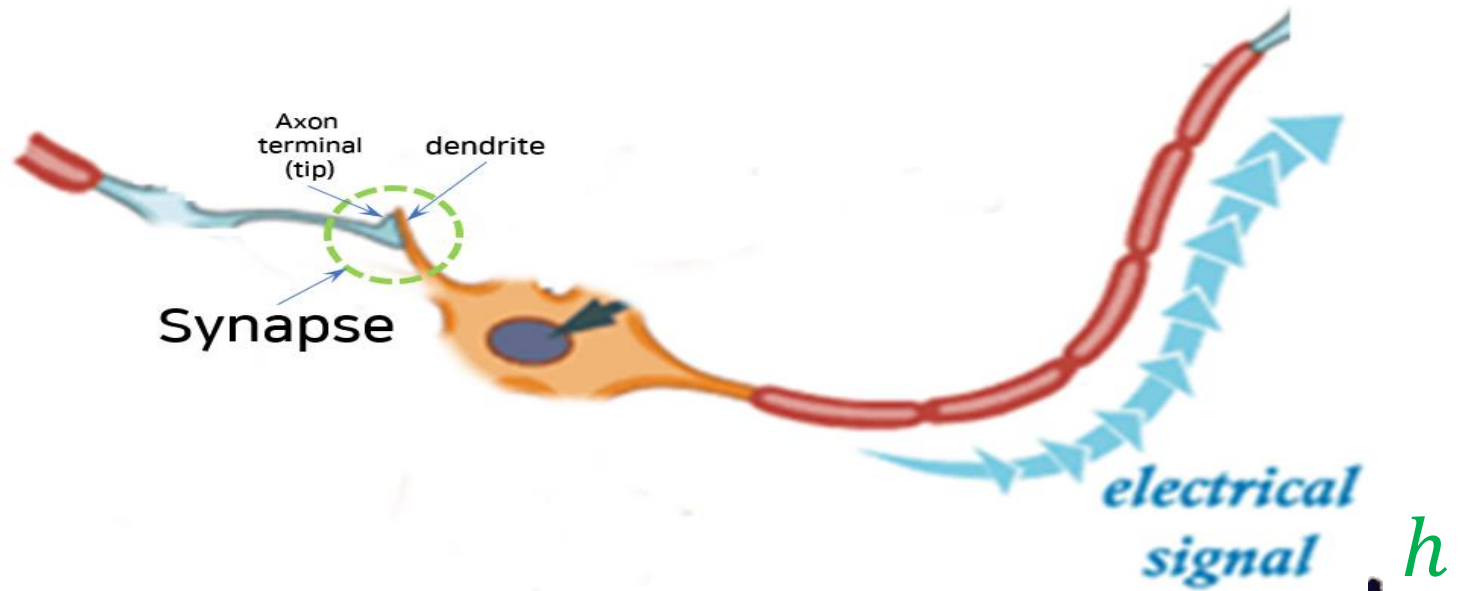
Jeju National University

Yung-Cheol Byun

Agenda

- Artificial Intelligence
- Brain, Neurons
- Learning
- Regression
- Deep Neural Networks
- CNN

Supervised
Learning



Logistic Regression

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

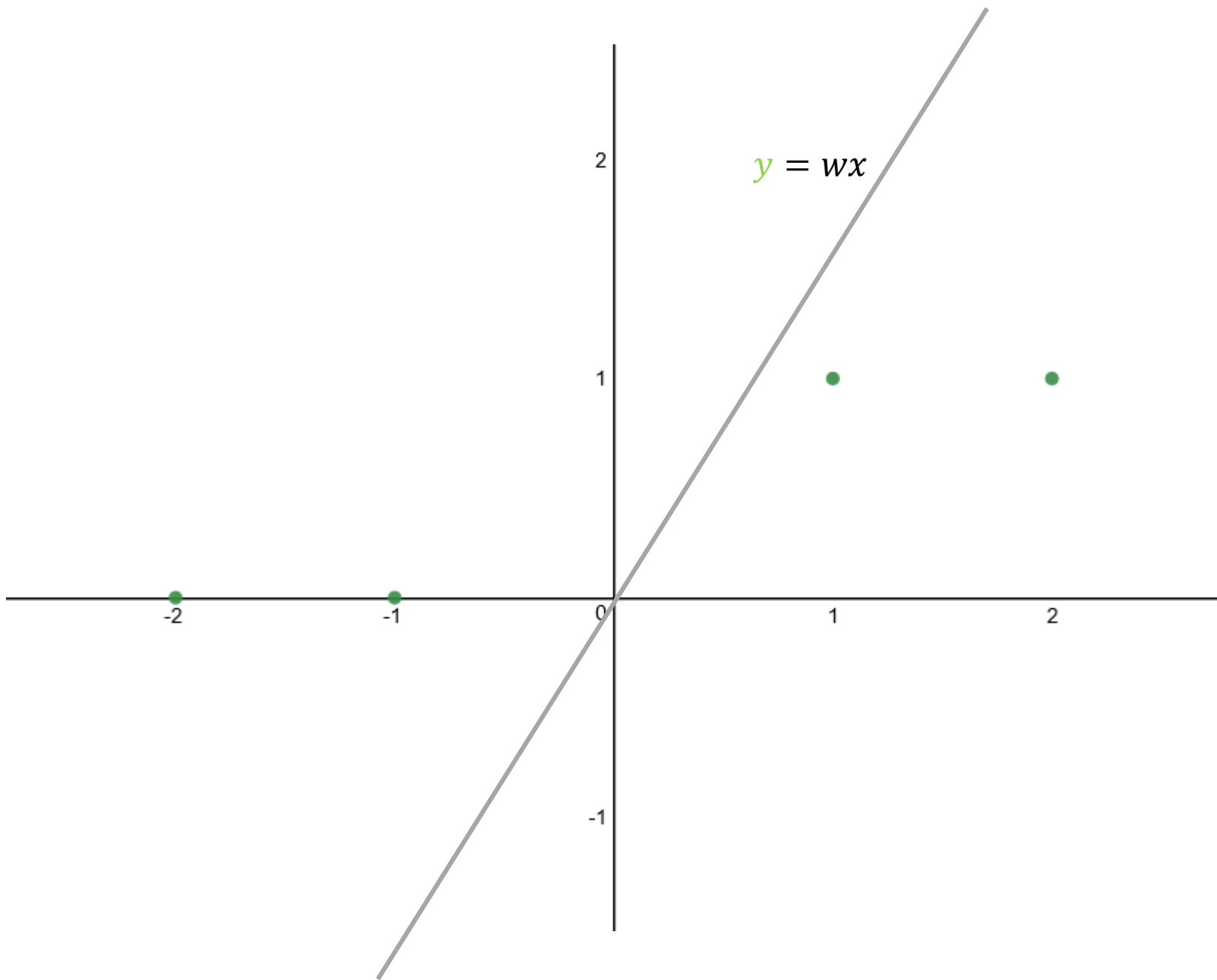
What does that mean?

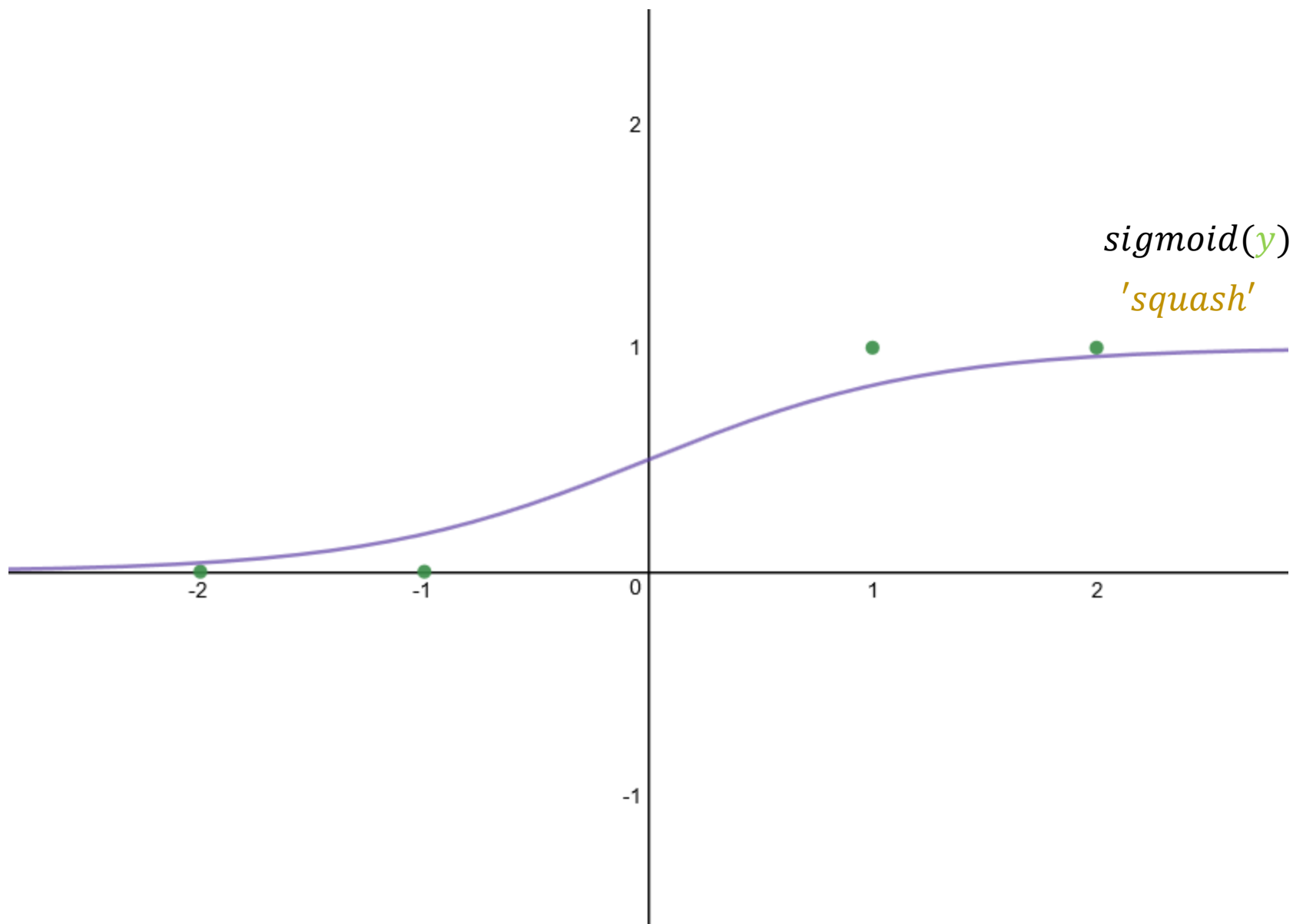


점 $(-2, 0)$, $(-1, 0)$, $(1, 1)$, $(2, 1)$ 표시

$$y = wx$$

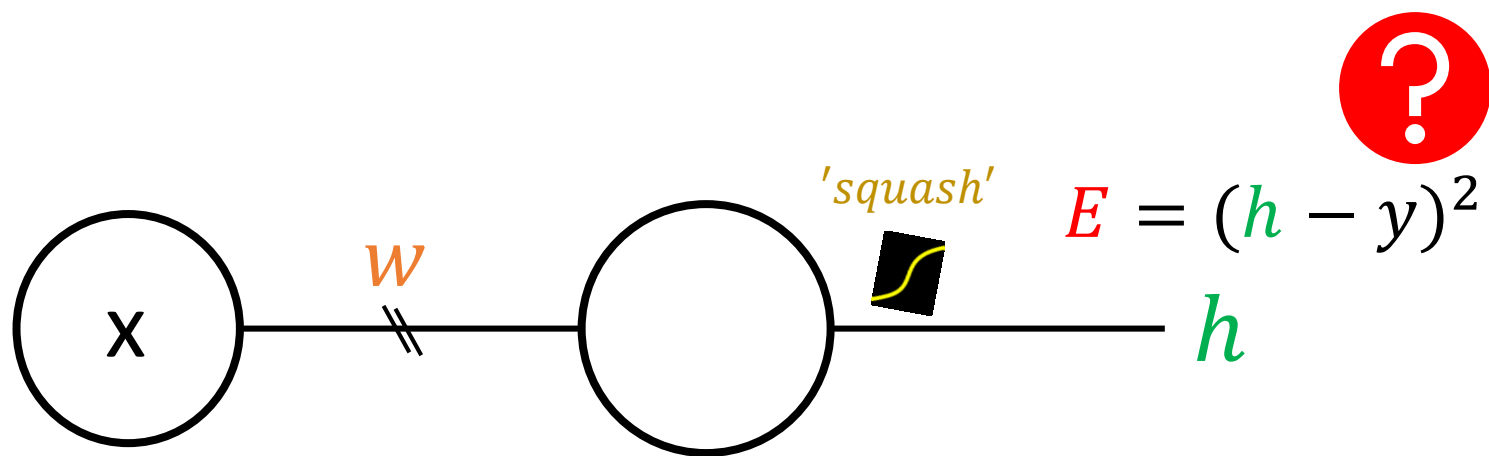
$$y = \frac{1}{1 + e^{-wx}}$$





가설

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



선형 회귀 오류 함수

Prediction by a neuron

$$\text{cost} = \frac{1}{m} \sum_{i=1}^m (\text{hypo}(x_i) - y_i)^2$$

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)

Cost/Loss/Stress Function

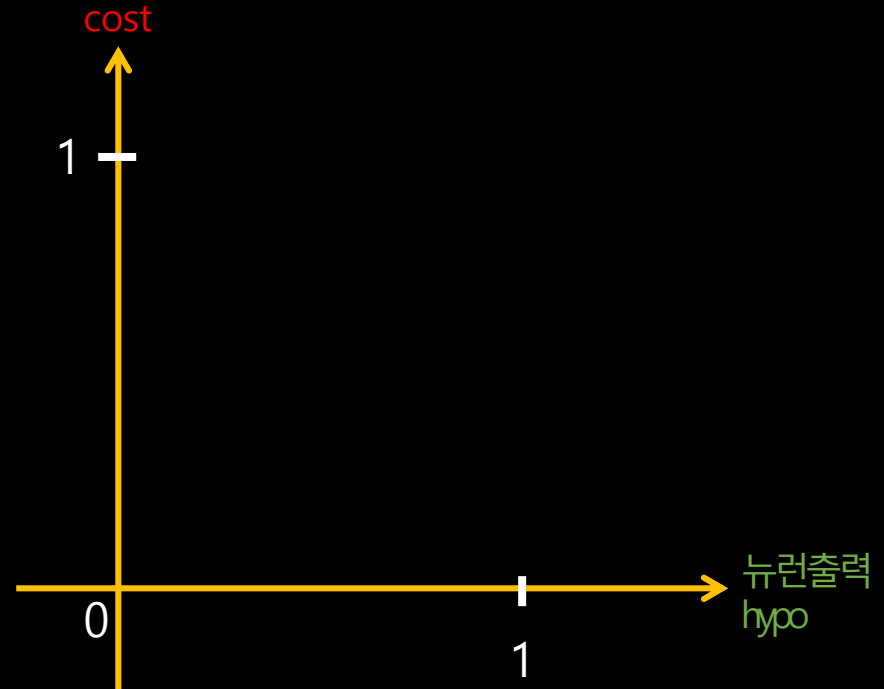
문제

오류 함수

- 뉴런의 출력 (hypo, 가설)이 정답과 일치하면 에러=0
- 뉴런의 출력 (hypo, 가설)이 정답과는 정반대 값이면 에러= ∞

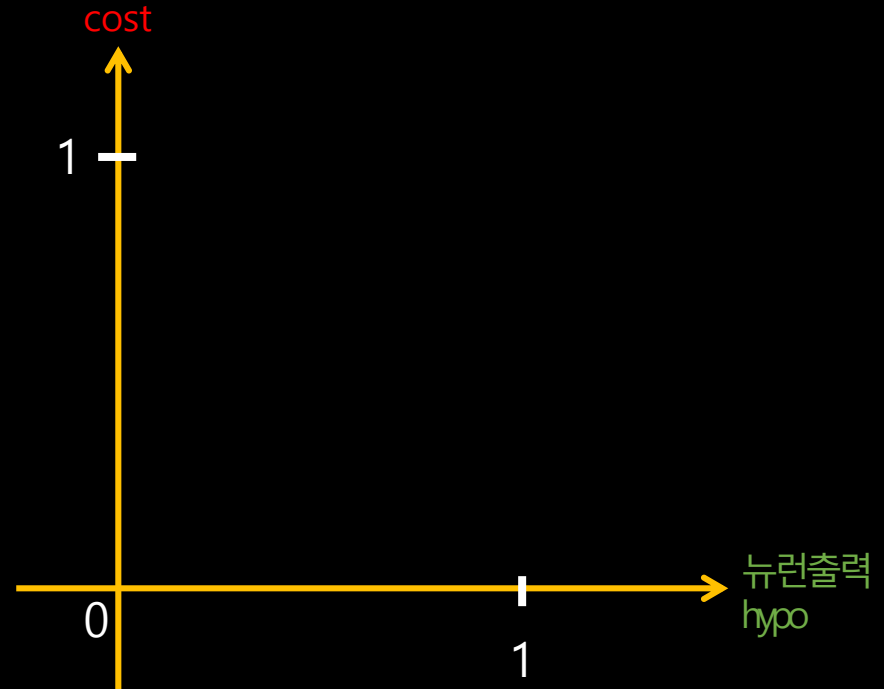
오류 함수

- 정답이 1일 때
 - 뉴런 출력(*hypo*)이 1이면 오류는 0이 되게
 - 뉴런 출력(*hypo*)이 0이면 오류는 ∞ 가 되게



오류 함수

- 정답이 0일 때
 - 뉴런 출력(*hypo*)이 0이면 오류는 0이 되게
 - 뉴런 출력(*hypo*)이 1이면 오류는 ∞ 가 되게



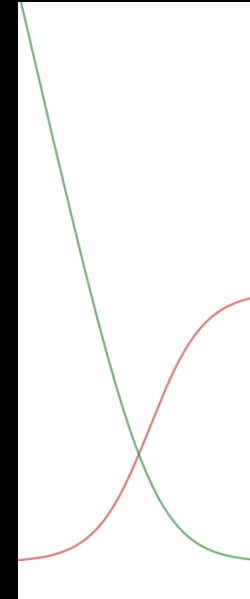
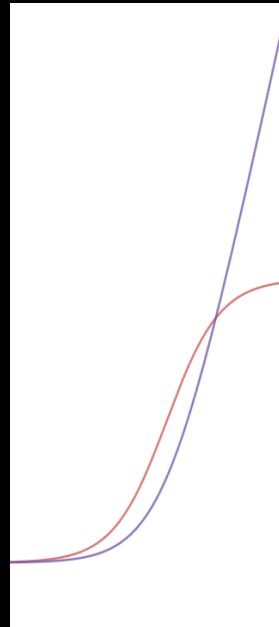


desmos

x: Prediction by a neuron

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

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

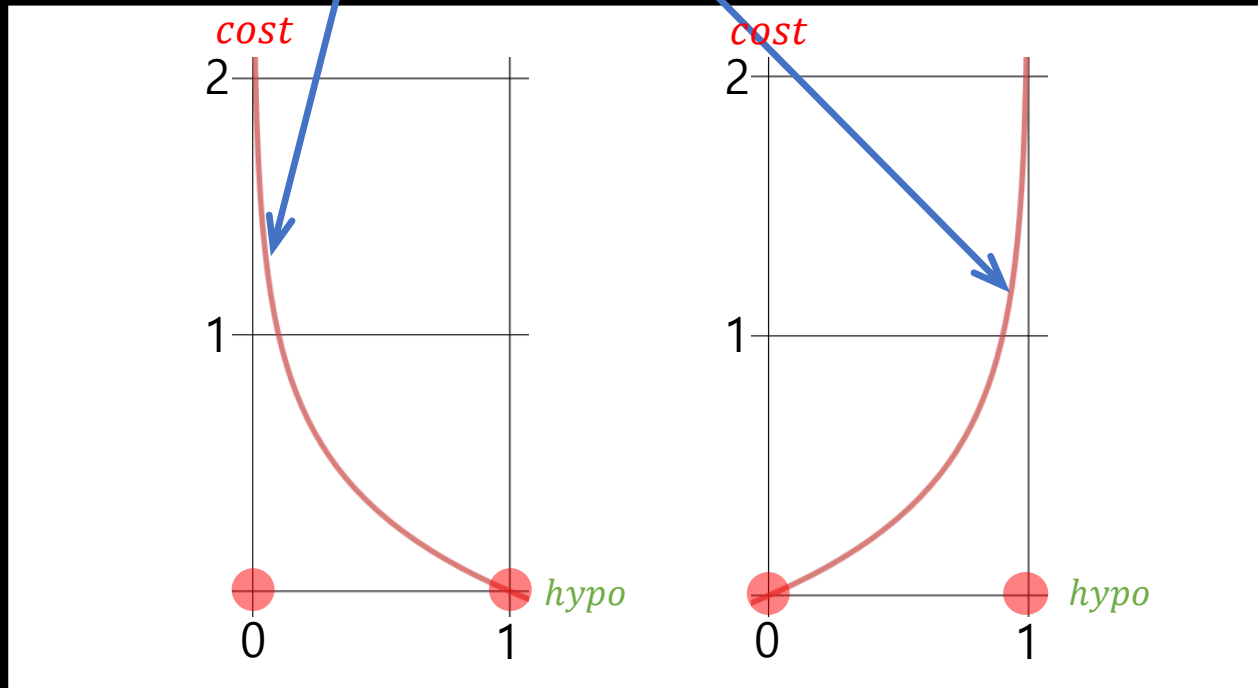


오류 함수

Prediction by a neuron

Correct answer

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



오류 함수

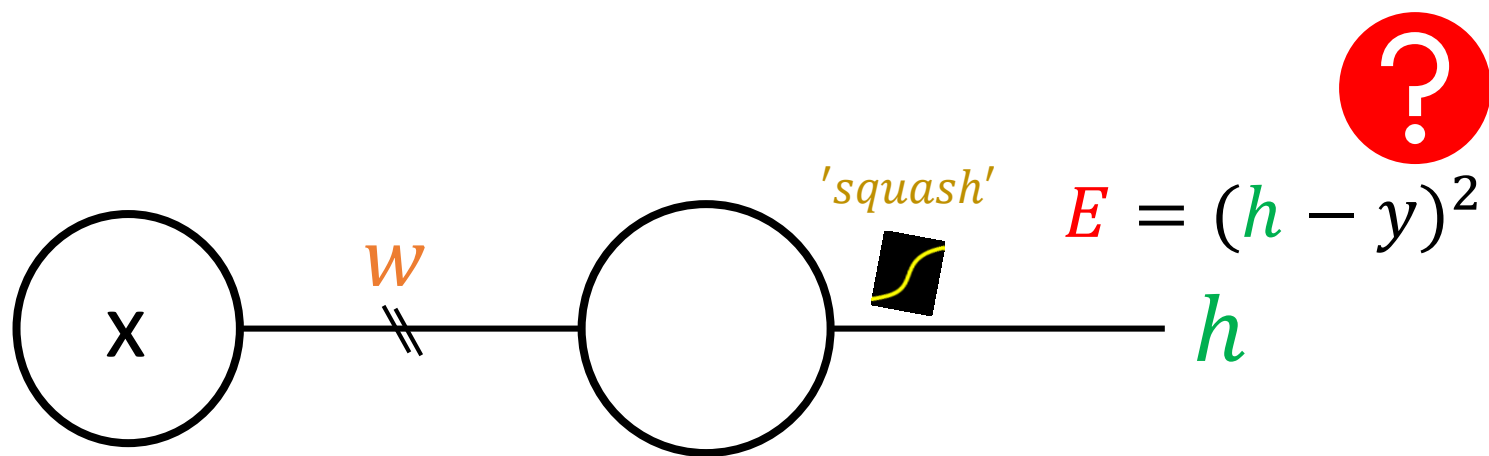
$$cost = \begin{cases} -\log(H(X)) & : y = 1 \\ -\log(1 - H(X)) & : y = 0 \end{cases}$$



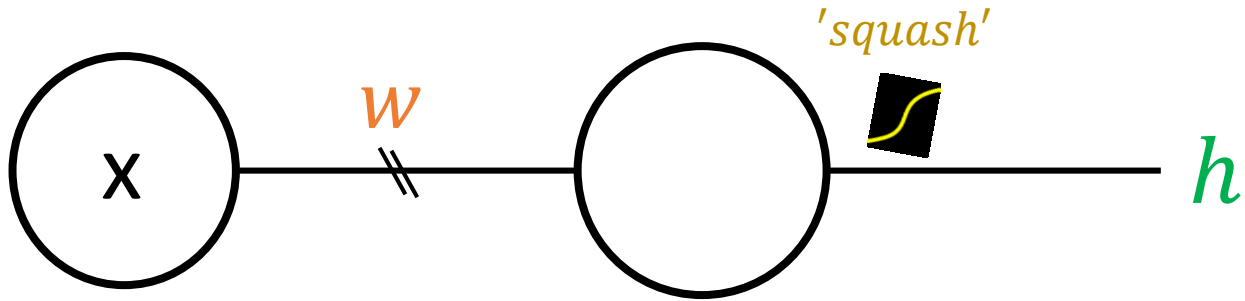
$$cost = -y \log(H(X)) - (1 - y) \log(1 - H(X))$$

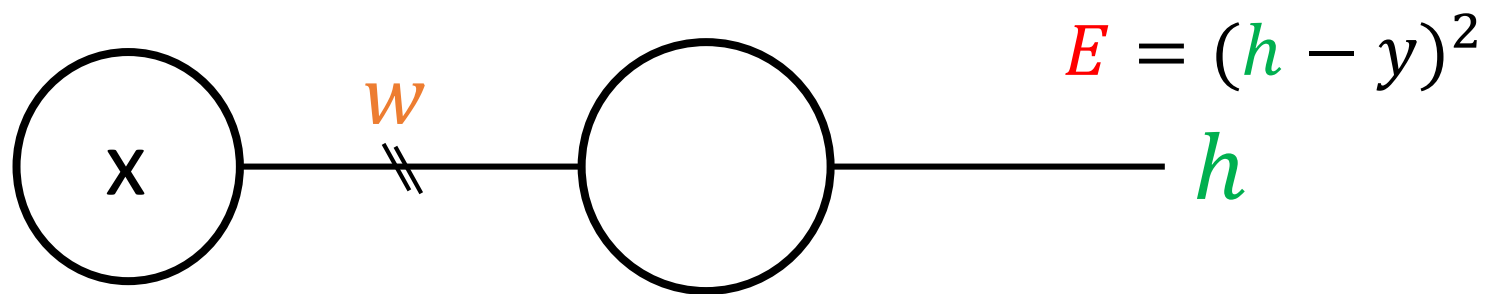
$$cost = -(y \log(H(X)) + (1 - y) \log(1 - H(X)))$$


$$W = W - \alpha \frac{\partial}{\partial W} cost(W)$$




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





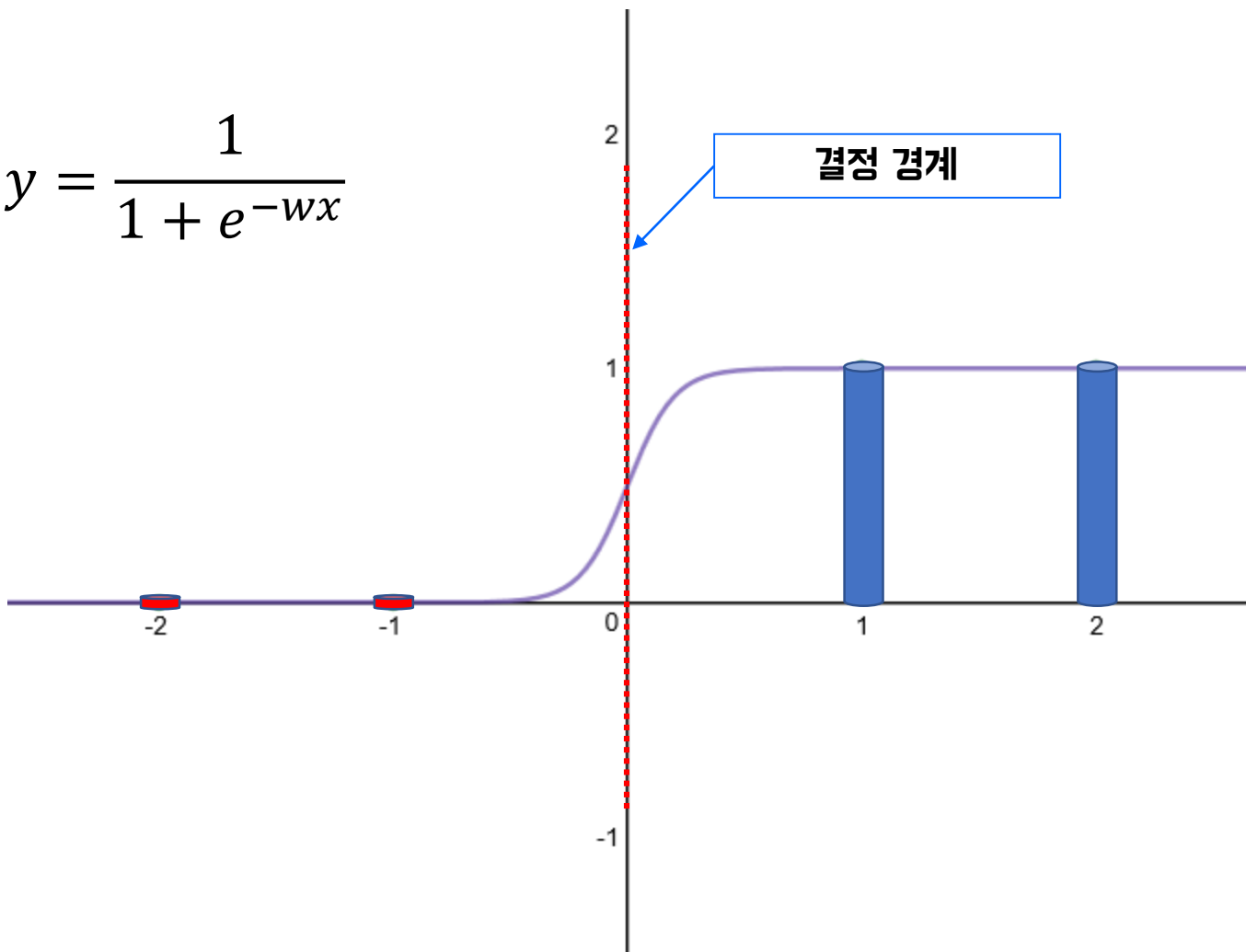
x	 y
-2	0
-1	0
1	1
2	1

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

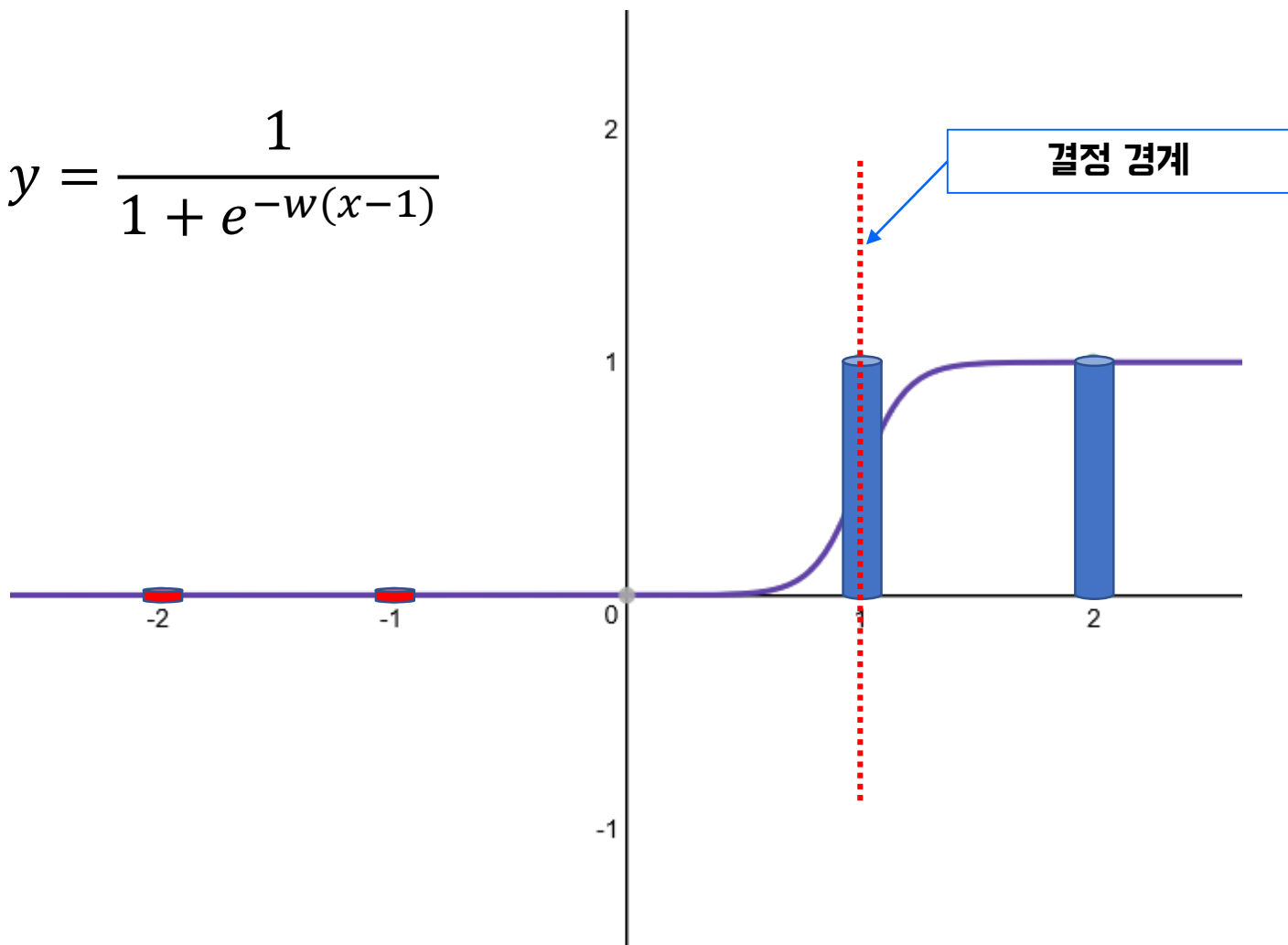


0, 1을 결정(decision)하는
경계(boundary)

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



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



결정 경계

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

0

결정 경계

or,

$$h = \text{sigmoid}(wx)$$

결정 경계

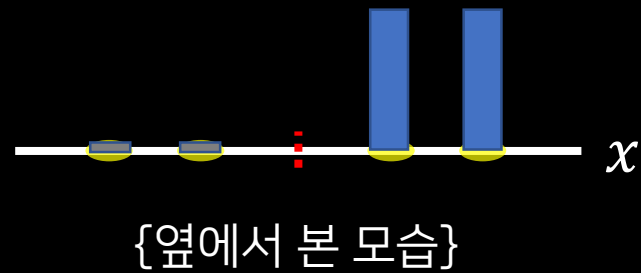
$$wx = 0$$

$$x = 0$$

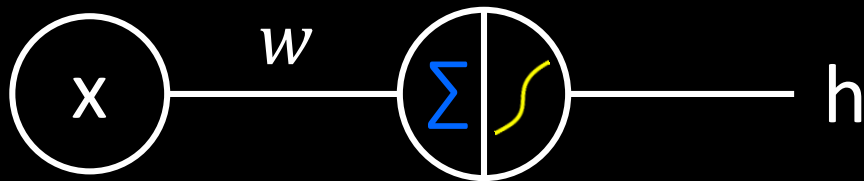
$$wx + b = 0$$

$$x + 1 = 0$$

$$2x + 3 = 0$$



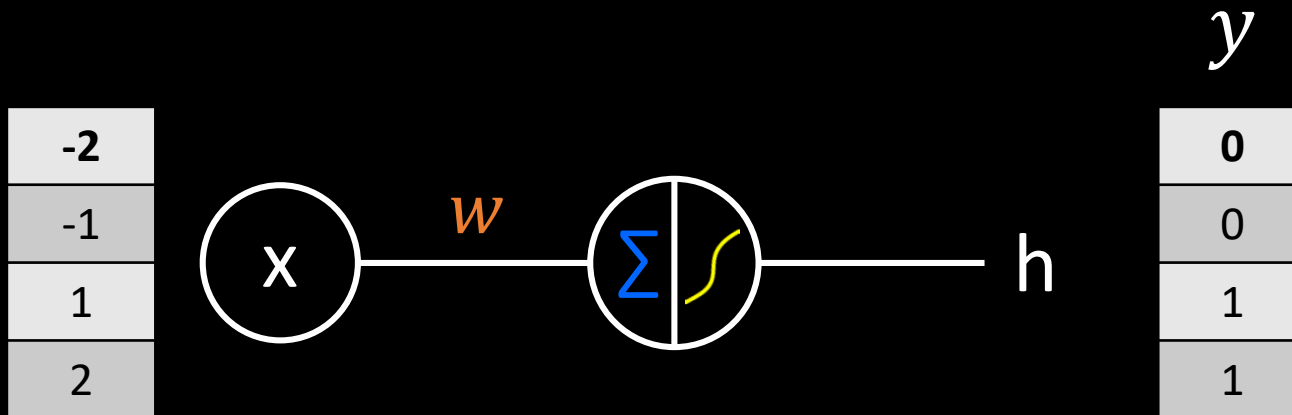
신경 세포 기능



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

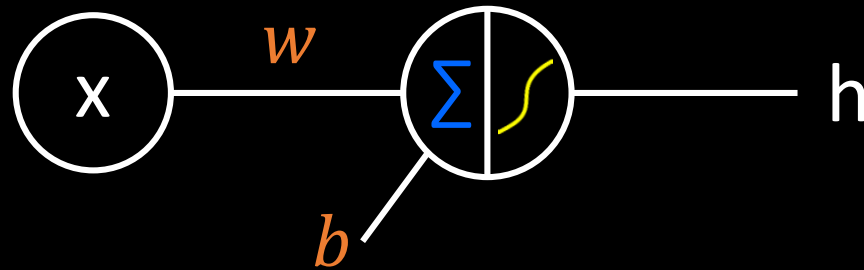
Classification

- 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



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

-2
-1
1
2



<i>y</i>
0
0
1
1

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

(실습) 11.py

음수는 0으로 양수는 1로 분류

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

```
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())

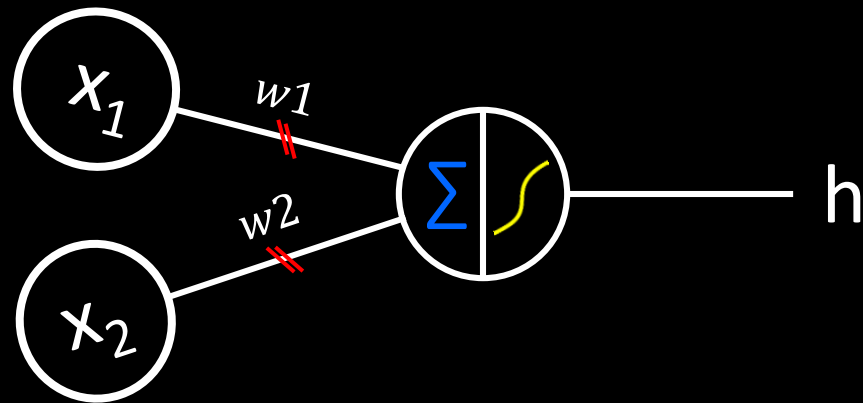
for step in range(5001):
    sess.run(train)
```

```
#----- 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?

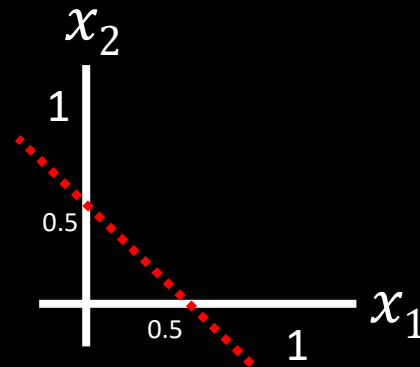
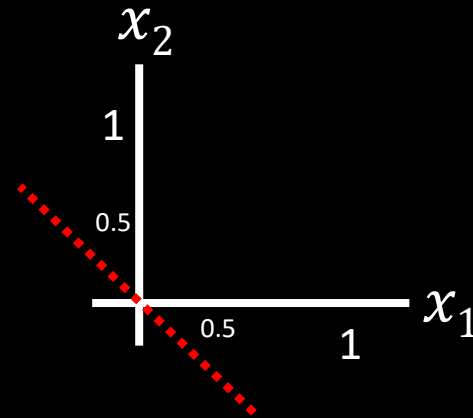
$$w_1x_1 + w_2x_2 = 0$$

$$x_1 + x_2 = 0$$

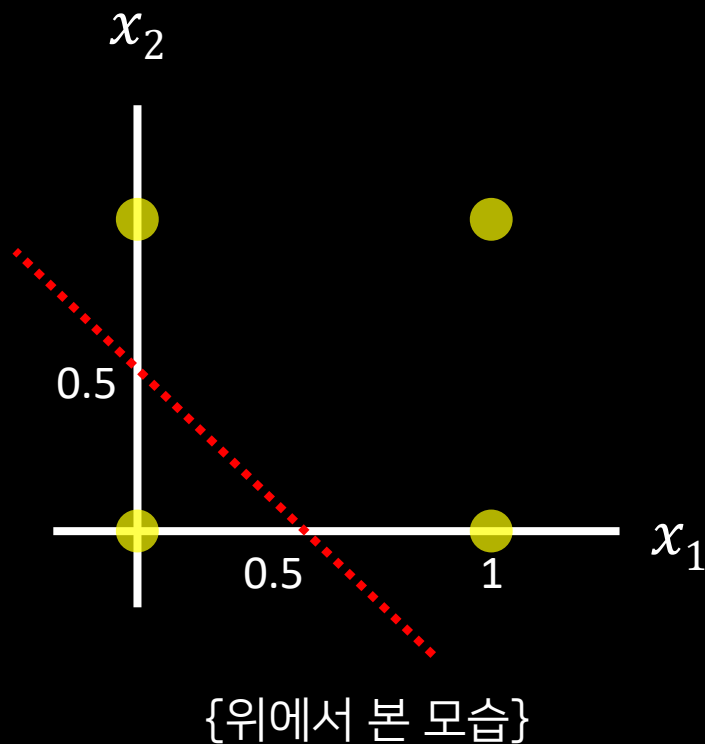
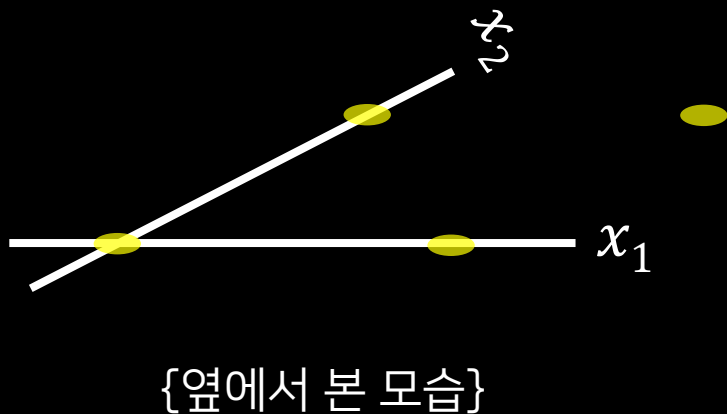
$$w_1x_1 + w_2x_2 = b$$

$$x_1 + x_2 = 0.5$$

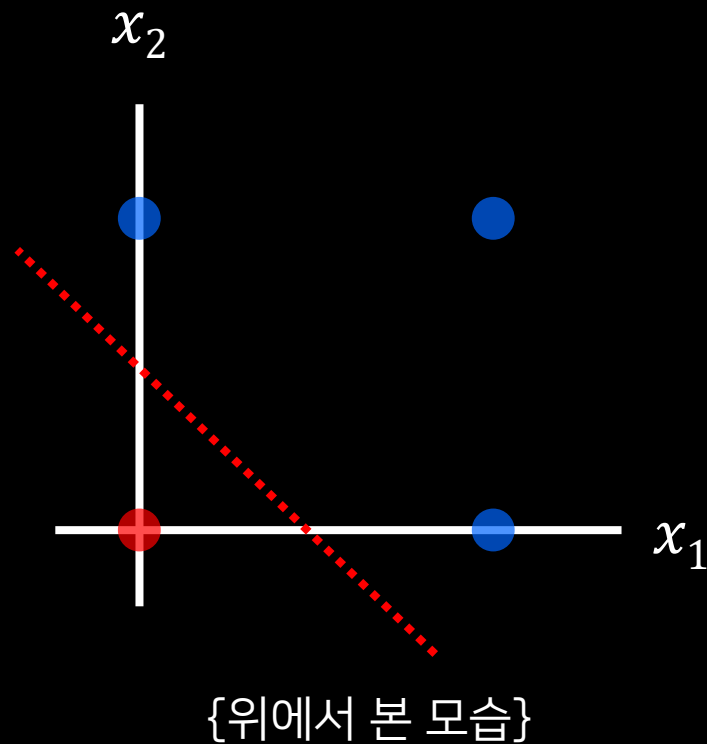
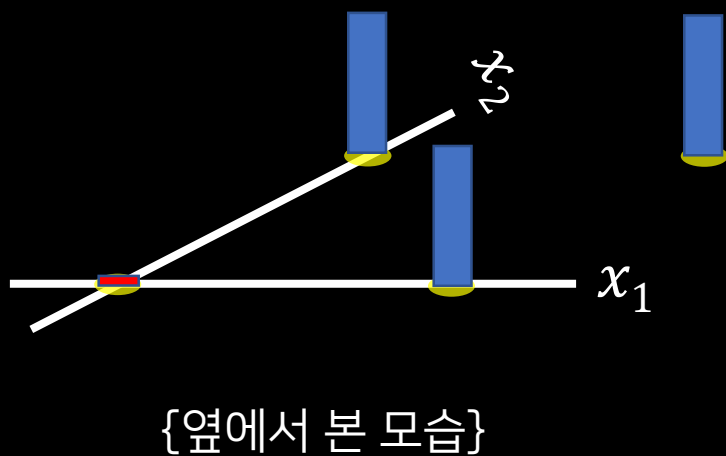
{위에서 본 모습}



신경 세포 (2 입력)



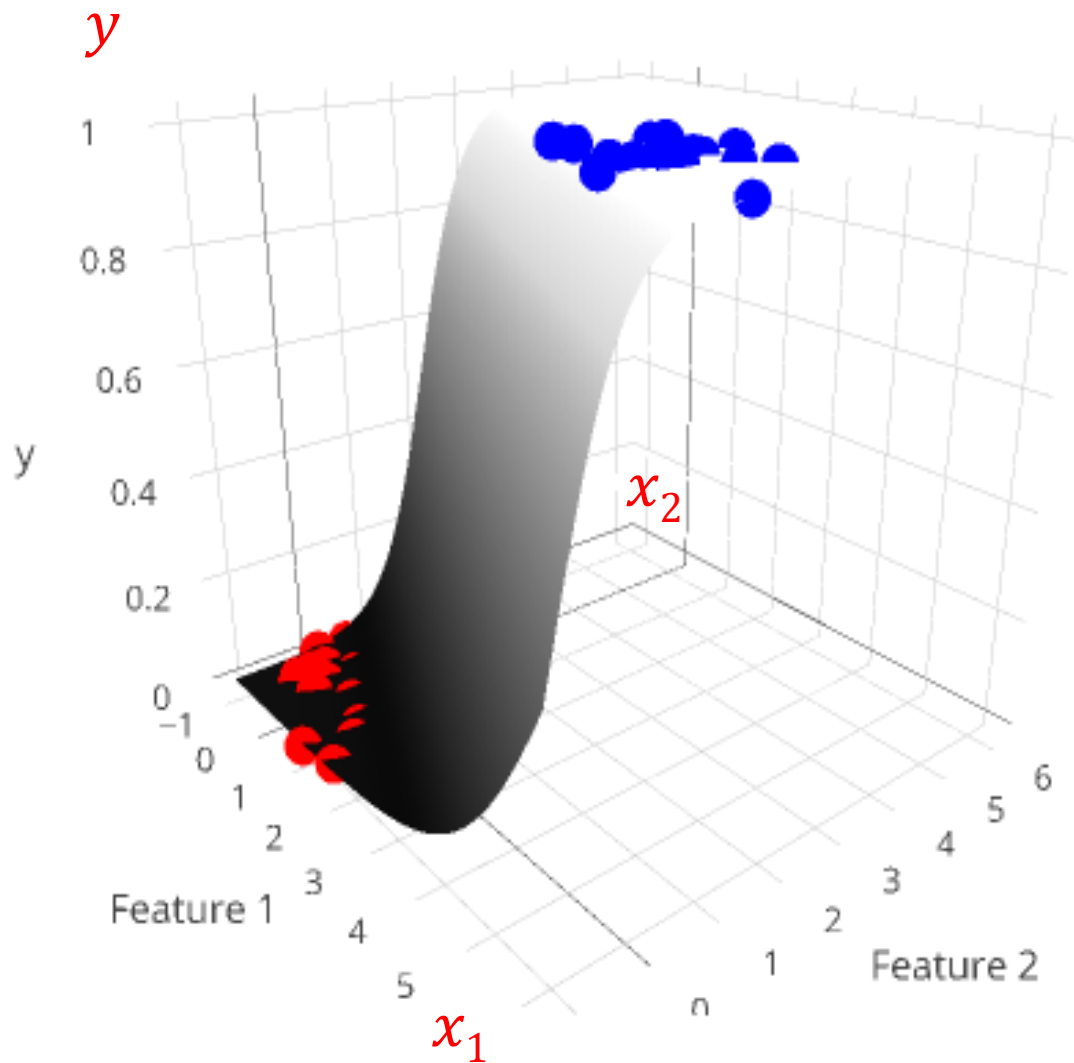
신경 세포 (2 입력)



{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!

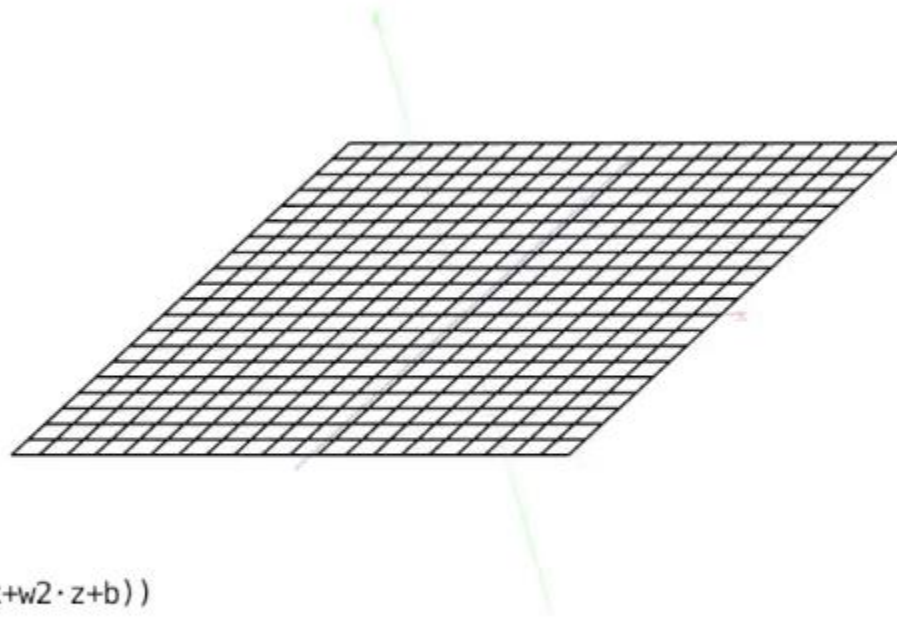
-1

-2

-3

Decision Boundary in 3D

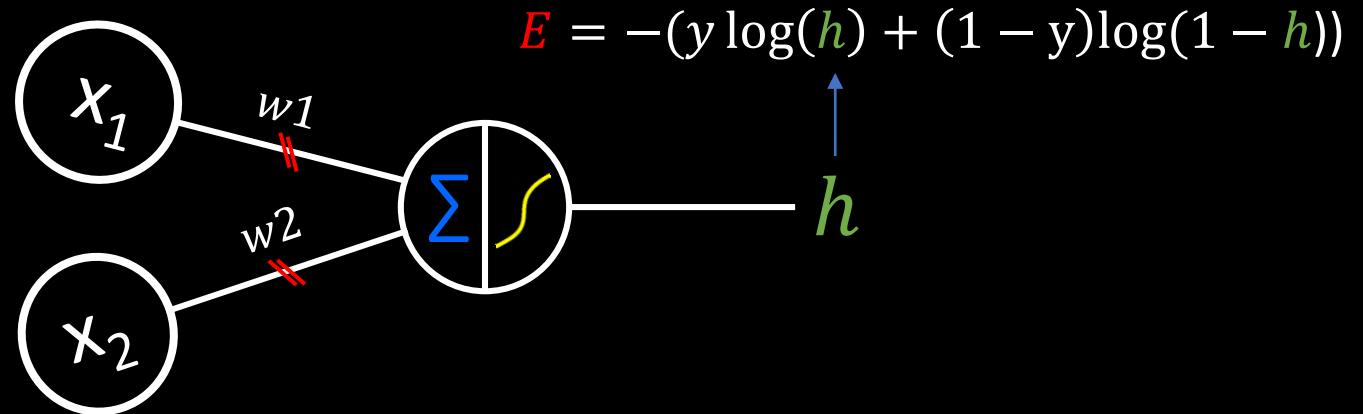
```
sigmoid(w1·length + w2·width + b)
```



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

(실습) 13.py

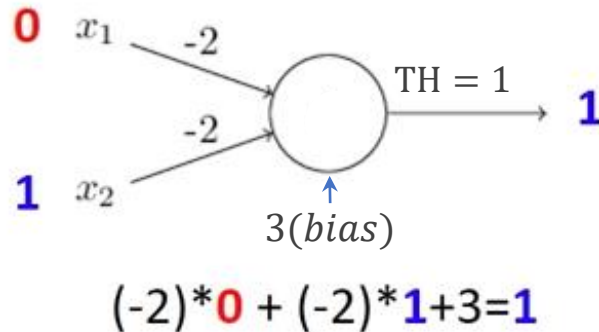
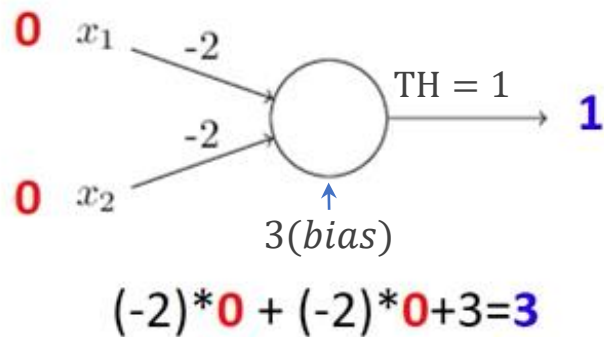
- 입력 두개(x_1, x_2)를 갖는 뉴런을 이용하여 OR 맞추기
- 한 개의 결정 경계



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

NAND

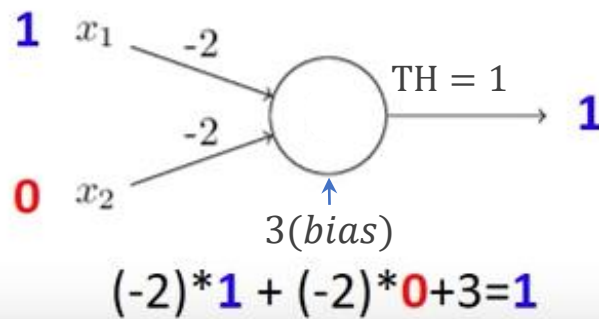
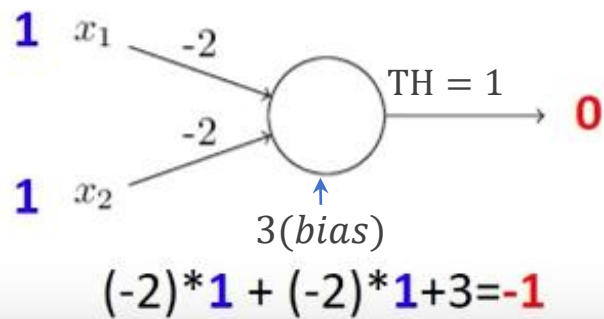
- NAND gates are functionally complete.
- We can build any logical functions out of them.



NAND

Truth Table

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



요약

- 로지스틱 리그레션과 분류(classification)
- 로지스틱 리그레션을 위한 cost 함수
- 한 개의 뉴런이 만들어 내는 결정 경계
- 텐서 플로우를 이용한 ML 프로그래밍