

AI and Deep Learning

# Multi-Layer Neural Networks

Jeju National University

Yung-Cheol Byun

# 학습과 테스트

- 학습(파라미터 튜닝)이 끝났다는 의미는?
  - 처음에 난수로 할당되었던 파라미터  $w$ (와  $b$ )가 잘 튜닝 되었다는 뜻

```
import tensorflow as tf
```

```
#----- training data
```

```
x_data = [1]
```

```
y_data = [1]
```

```
#----- a neuron
```

```
w = tf.Variable(tf.random_normal([1]))
```

```
hypo = w * x_data
```

```
#----- learning
```

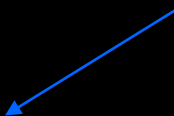
```
cost = (hypo - y_data) ** 2
```

```
train =
```

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

```
cost = (w * x_data - y_data) ** 2
```

**w를 살짝 바꾸면  
cost 값은 어떻게 변할까?  
(w가 cost에 미치는 영향)**



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

```
for i in range(1001):  
    sess.run(train)
```

```
    if i % 100 == 0:  
        print( ' w: ' , sess.run(w), ' cost: ' , sess.run(cost))
```

```
#----- testing (prediction)  
print(sess.run(hypo))
```

**이 곳까지  
실행되면 학습이  
끝남.**



# 학습과 테스트

- 새로운 데이터를 x\_data에 넣어 테스트해보자.

```
#----- testing(prediction)  
x_data = [2]  
print(sess.run(hypo))
```

처음 할당한 값이 계산 그래프  
안으로 복사되어 사용되었고, 이후  
새로운 값을 x\_data에  
할당하여도 반영되지 않음. **실패!**

# 학습과 테스트

- 학습 데이터가 매우 클 경우 한꺼번에 메모리로 넣을 수 없음.
  - 데이터를 여러 개로 잘라서 하나씩 `x_data`에 넣어 학습시킬 수 있어야 함.
  - 앞에서 살펴 본 이유로 **실패!**

# 플레이스 홀더

- 바뀔 수 있는 데이터 부분을 표시해 놓고  
나중에 원하는 데이터를 할당 → 플레이스  
홀더(place holder, 자리 표시자)

```
import tensorflow as tf
```

```
#----- training data
```

```
x_data = [1]
```

```
y_data = [1]
```

```
#----- a neuron
```

```
w = tf.Variable(tf.random_normal([1]))
```

```
hypo = w * x_data
```

```
#----- learning
```

```
cost = (hypo - y_data) ** 2
```

```
train =
```

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

1. 플레이스 홀더 X, Y를 선언한다.
2. 데이터가 들어가는 자리를 플레이스 홀더로 표시. 즉, x\_data를 X, y\_data를 Y로 표시한다.
3. 이후 hypo, cost, train 실행 시 원하는 데이터를 준다.



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

```
for i in range(1001):
    sess.run(train)
```

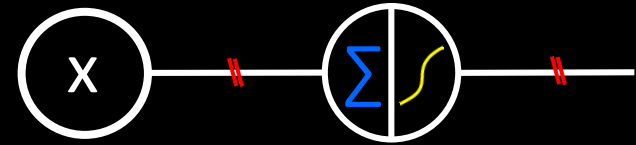
```
    if i % 100 == 0:
        print( ' w: ' , sess.run(w), ' cost: ' , sess.run(cost))
```

```
#----- testing (prediction)
print(sess.run(hypo))
```

(실습) 15.py

# 신경 세포 (1 입력)

- 결정 경계는? 값



$$wx = 0$$

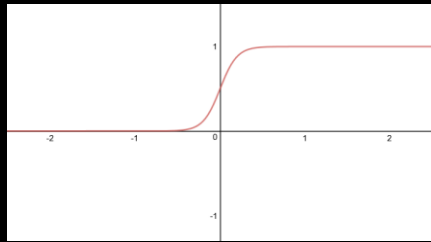
$$x = 0$$

“바이어스가 있을 경우 역할은?”  
만일  $w$ 와 바이어스가 각각 1이면?

$$x + 1 = 0$$

$$x = -1 \quad \text{결정경계}$$

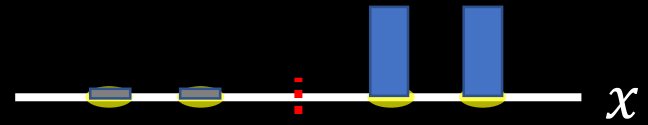
# 신경 세포 (1 입력)



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

결정 경계

0



{옆에서 본 모습}



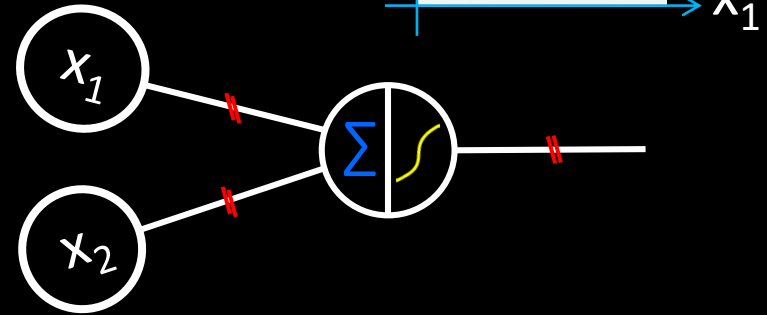
{위에서 본 모습}

# 신경 세포 (2 입력)

- 결정 경계는? 선

$$w_1x_1 + w_2x_2 = 0$$

$$x_1 + x_2 = 0$$



“바이어스가 있을 경우 역할은?”

# 신경 세포 (3 입력)

- 결정 경계는? 면

$$w_1x_1 + w_2x_2 + w_3x_3 = 0$$

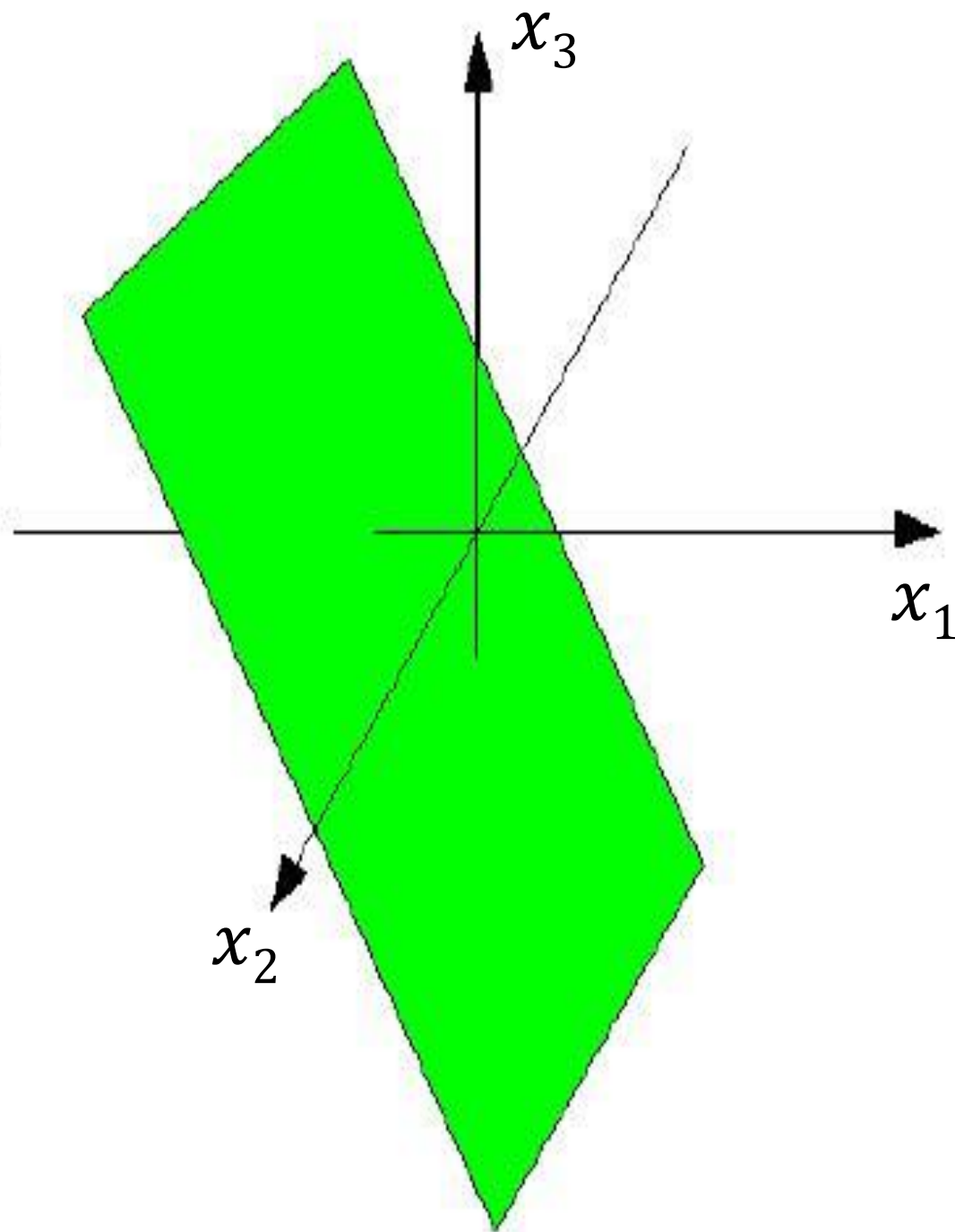
$$x_1 + x_2 + x_3 = 0$$

“바이어스가 있을 경우 역할은?”

입력이 3개인 신경 세포 1개는  
이런 **결정 경계**를 만든다.

$$x_1 + x_2 + x_3 + 1 = 0$$

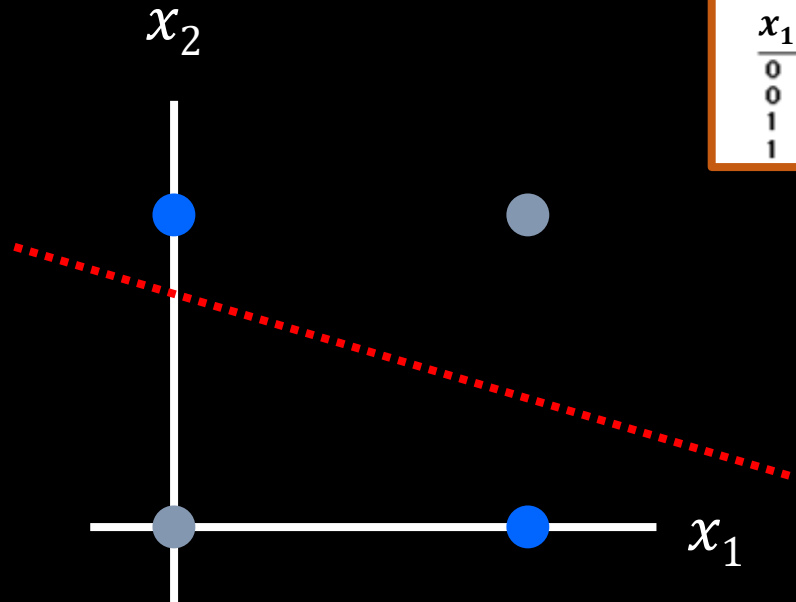
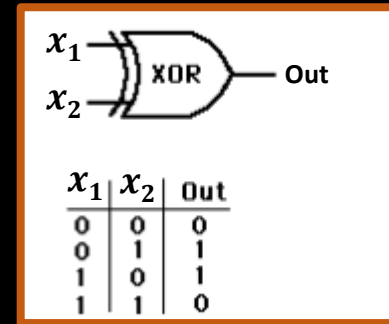
“바이어스 = +1”



이제까지는 모두  
선형 결정경계 로 분류하는 문제  
하지만...



# XOR 문제



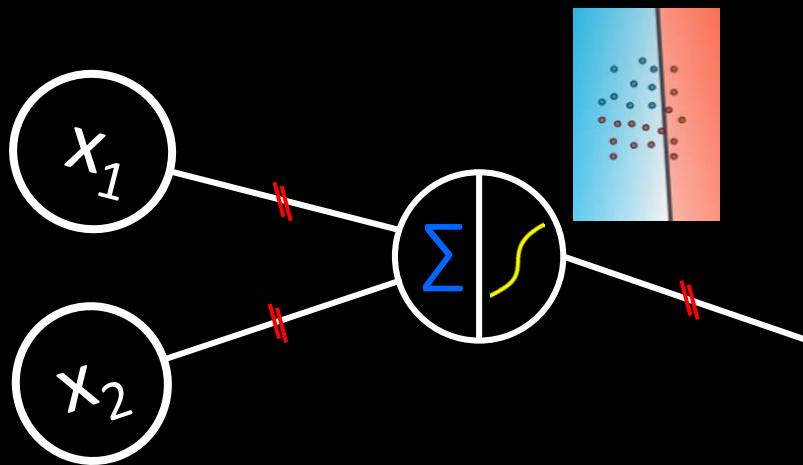
{위에서 본 모습}

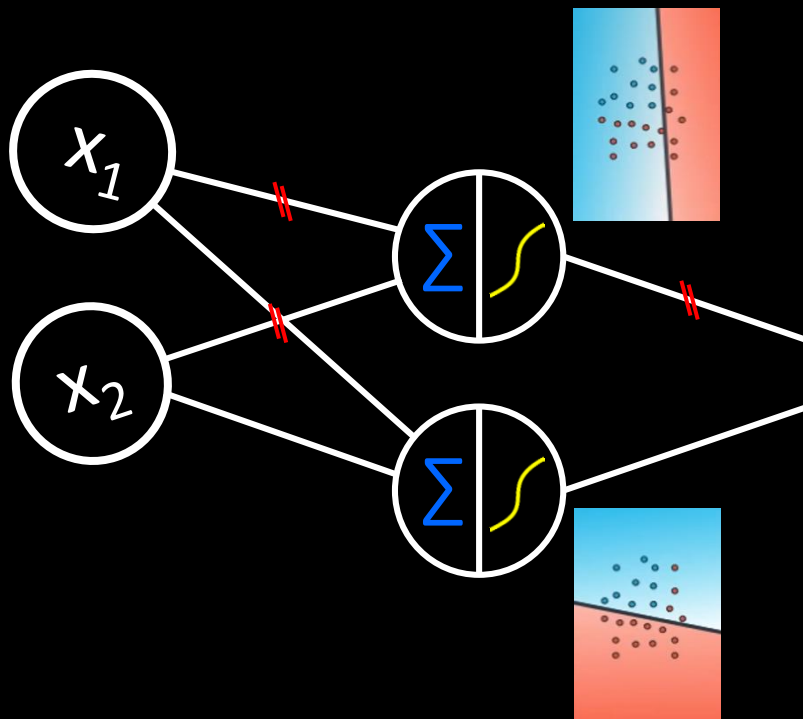
# XOR 문제

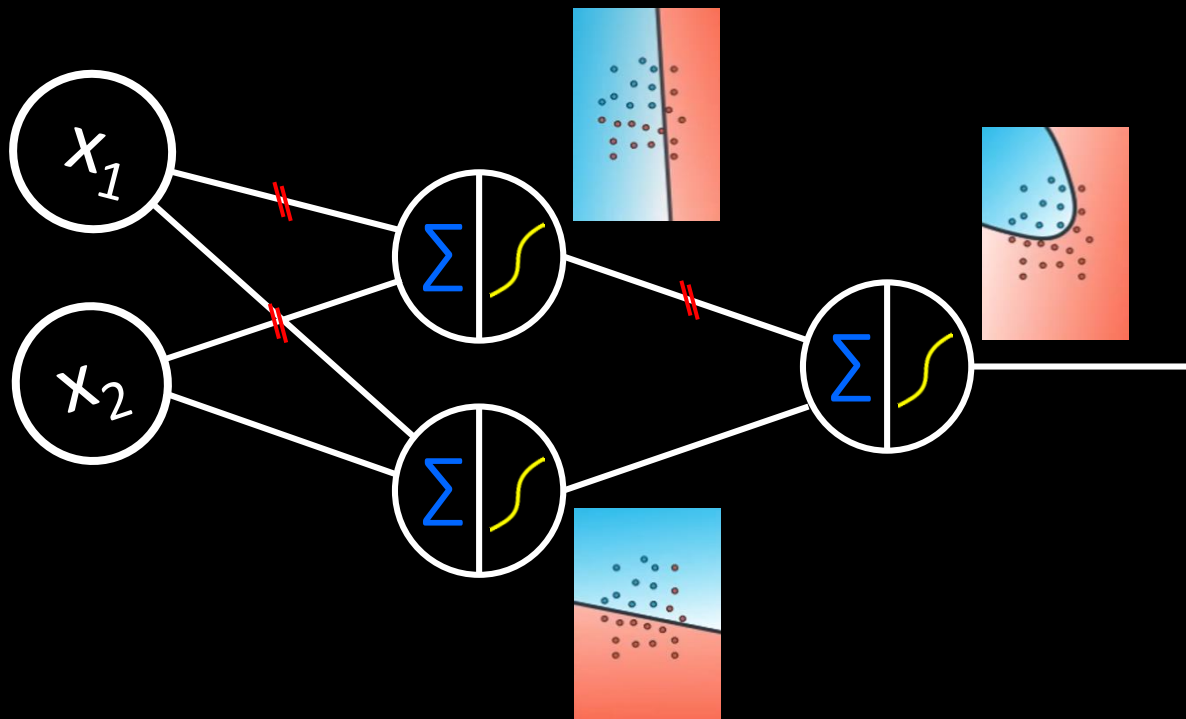
- 클래스 수는?
- 따라서 필요한 결정경계의 수는?
- 선형 결정경계 1개로는 불가능
- 선형 결정경계 2개로도 불가능 (왜?)
- 비선형 결정경계 1개가 필요

# (실습) 16.py

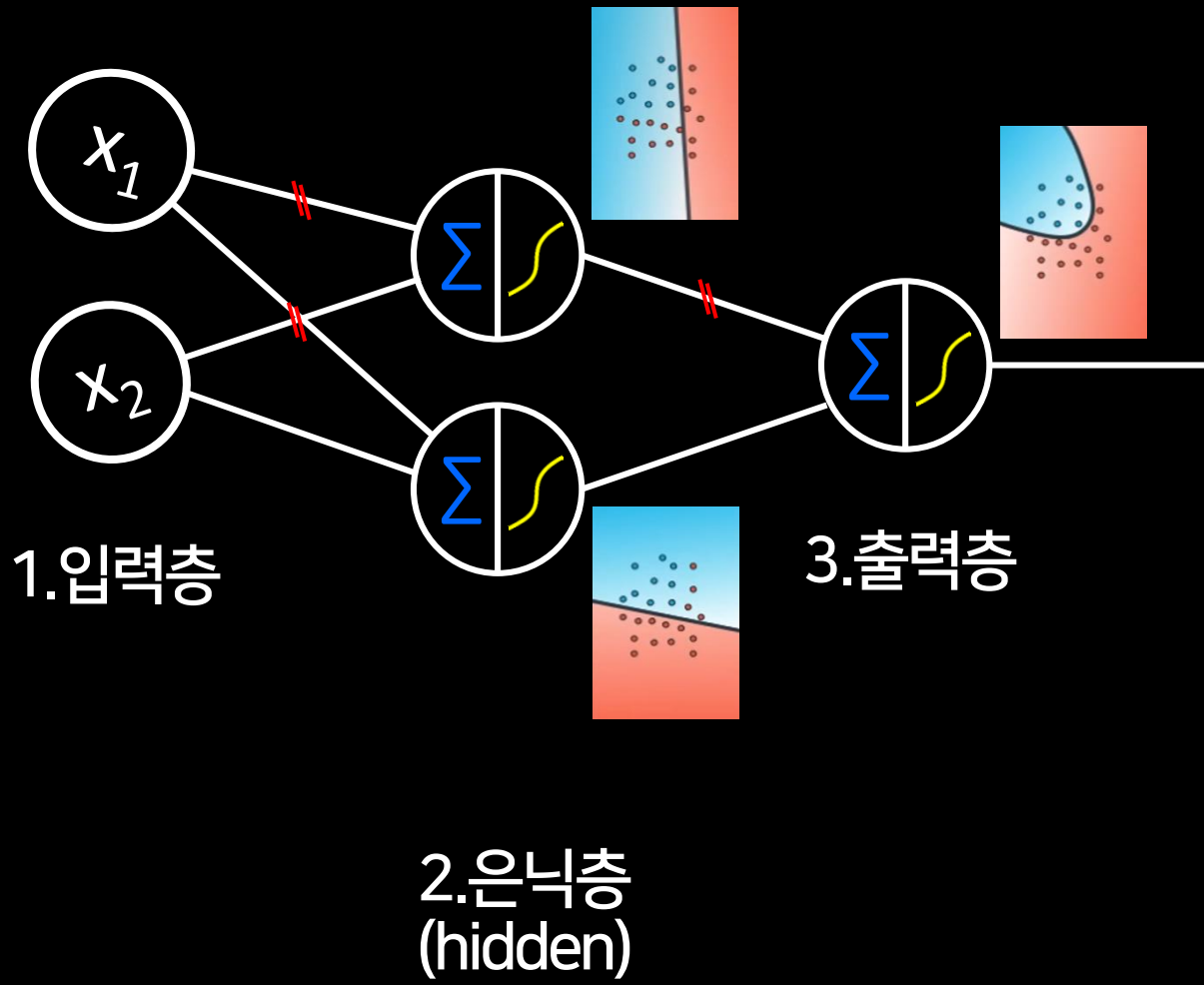
- 신경세포 하나
- 선형 결정 경계 1개
- 해결 불가능

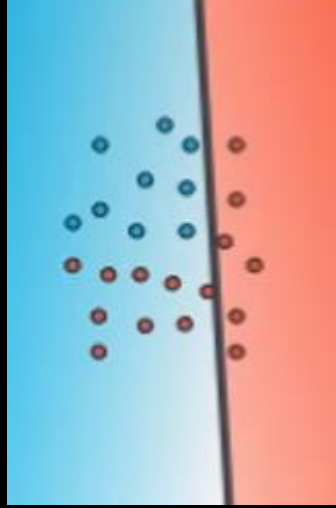




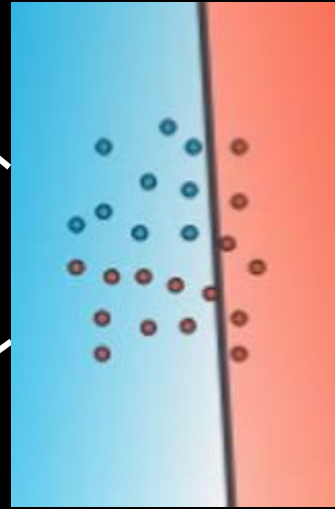


# "3층"

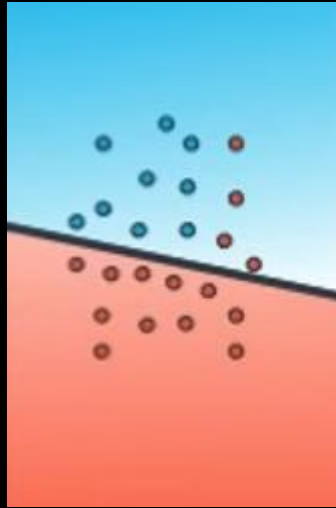




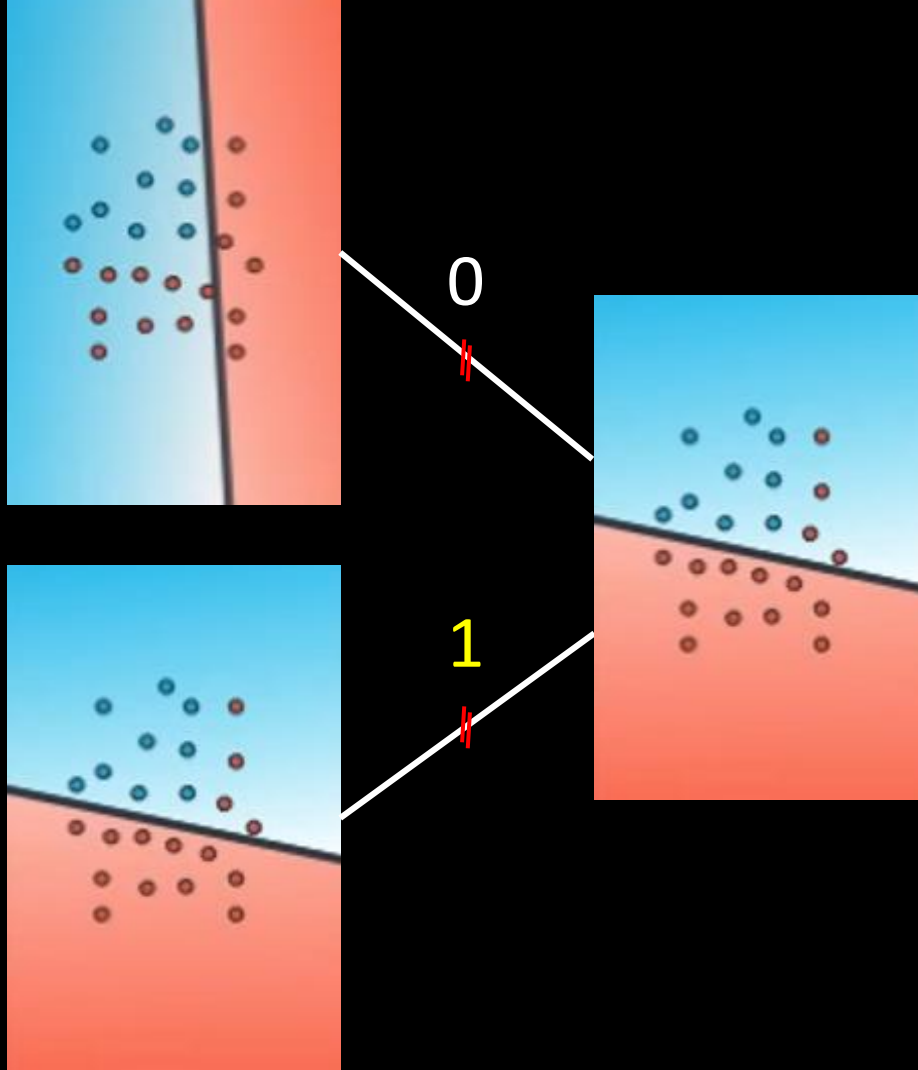
1

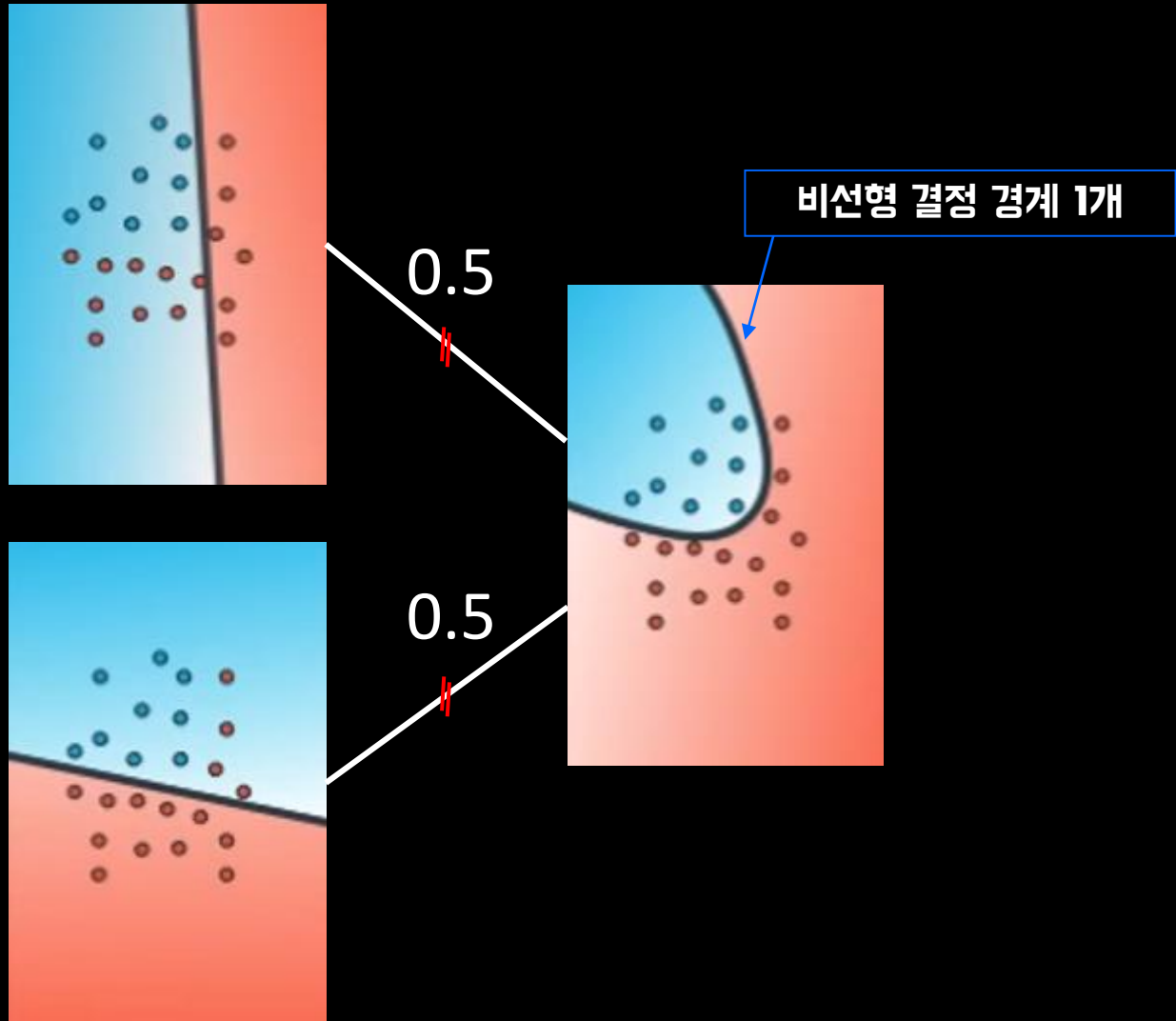


0

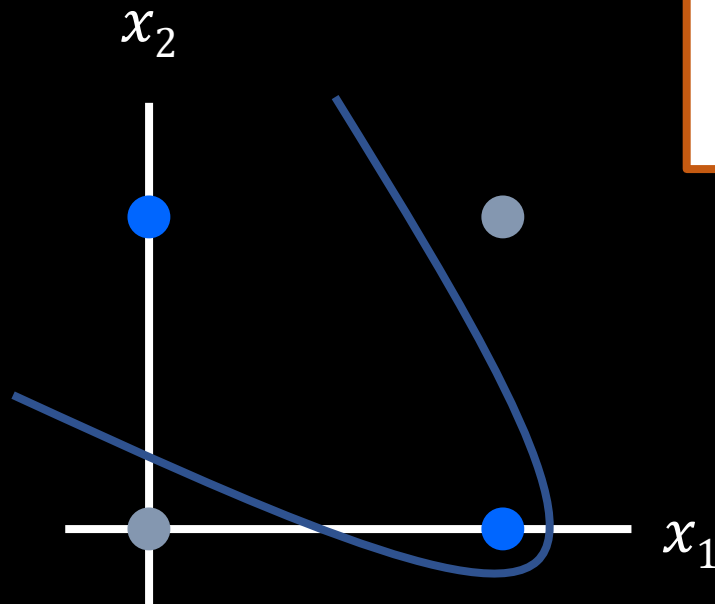
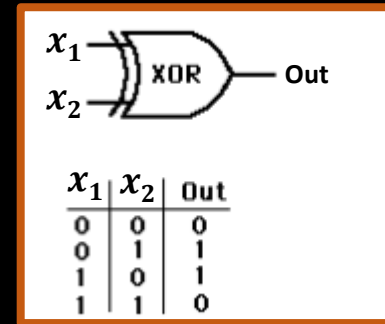








# XOR 문제

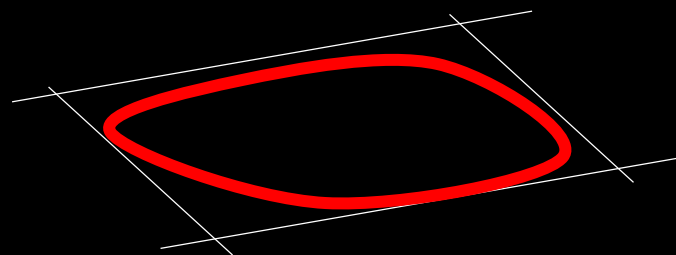
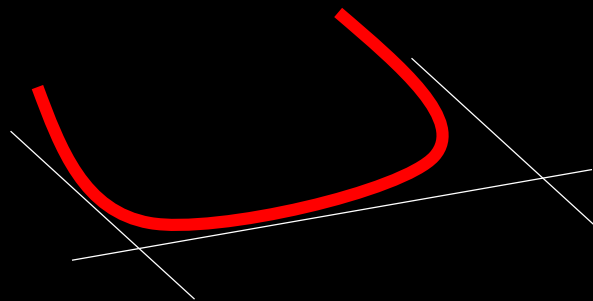


{위에서 본 모습}

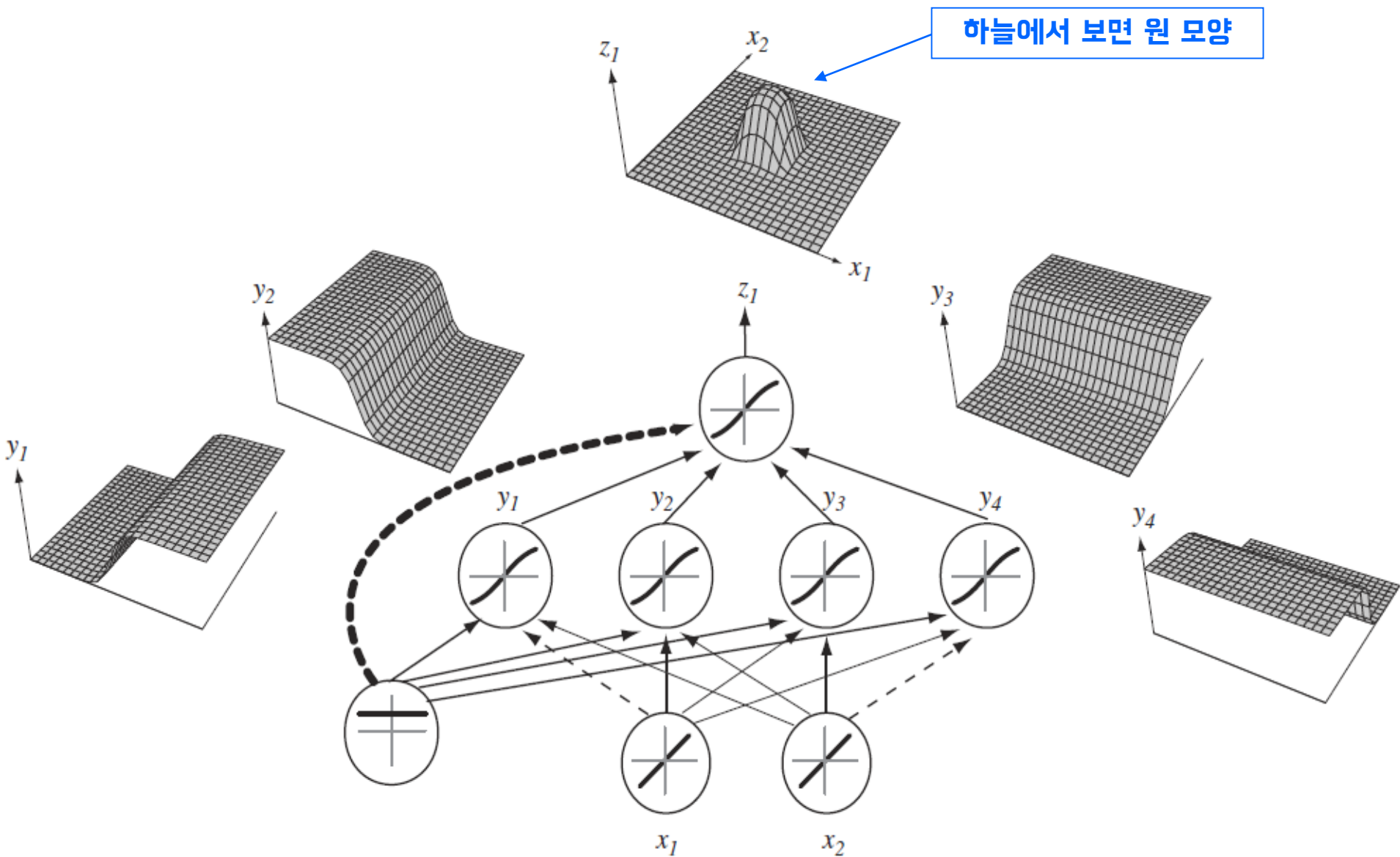
옆에서 본 모습은?

# 비선형 결정 경계

- 3개의 선형 결정 경계를 합치면 어떤 모양을 예상할 수 있을까?
- 4개의 선형 결정 경계를 합치면?



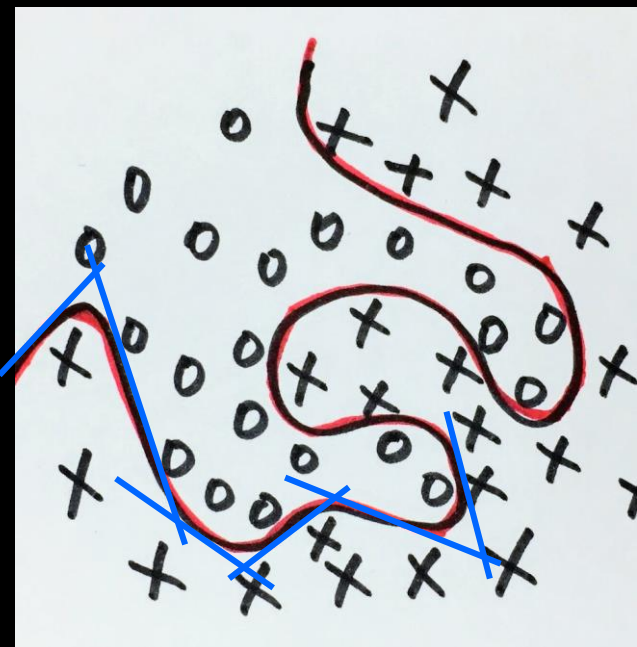
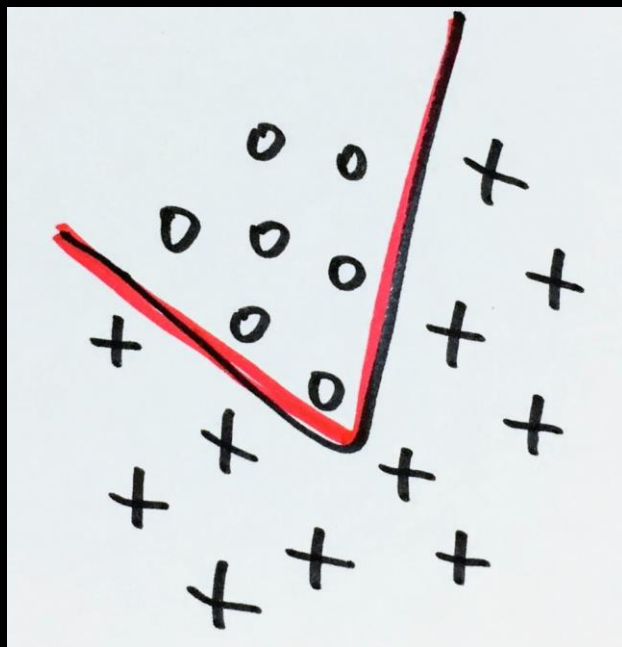
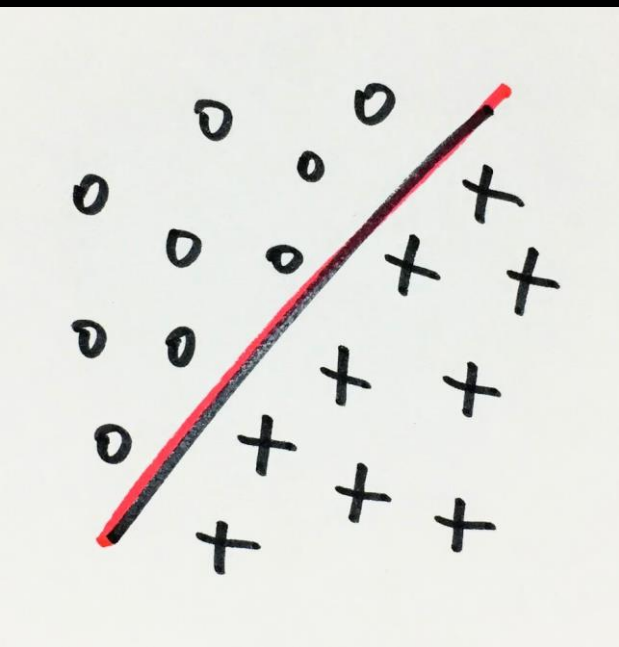
# 원 모양의 결정경계



# (실습) 17.py

- XOR 문제
- 3층: 입력층 - 은닉층 - 출력층

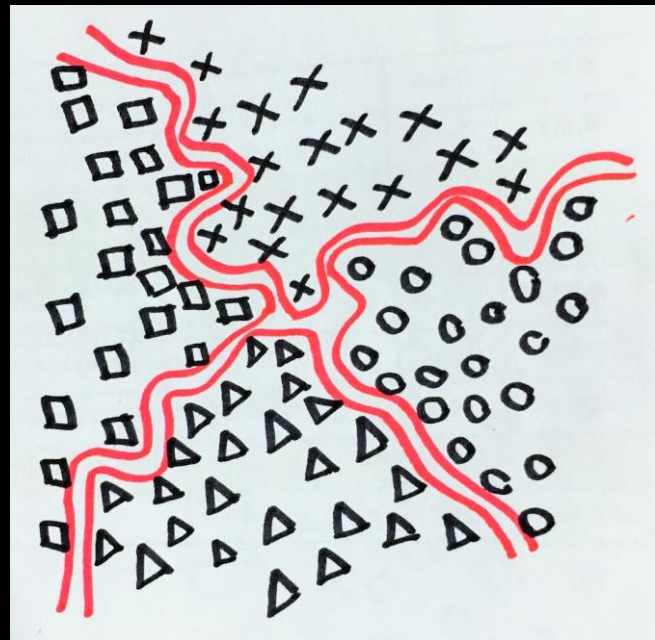
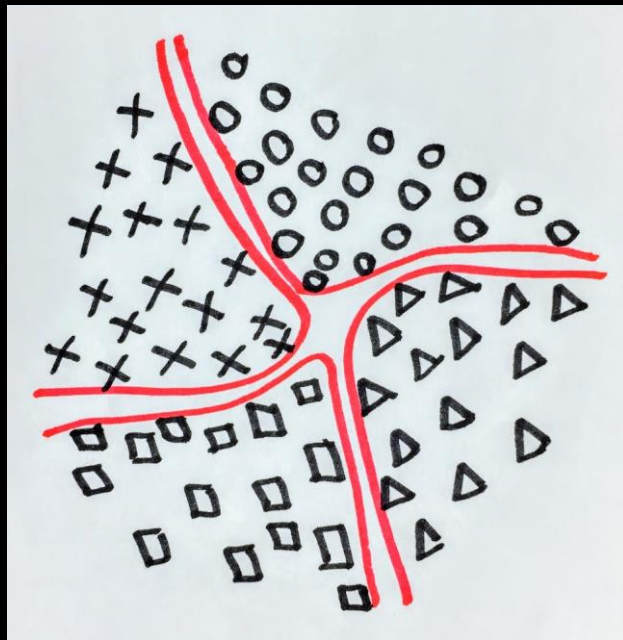
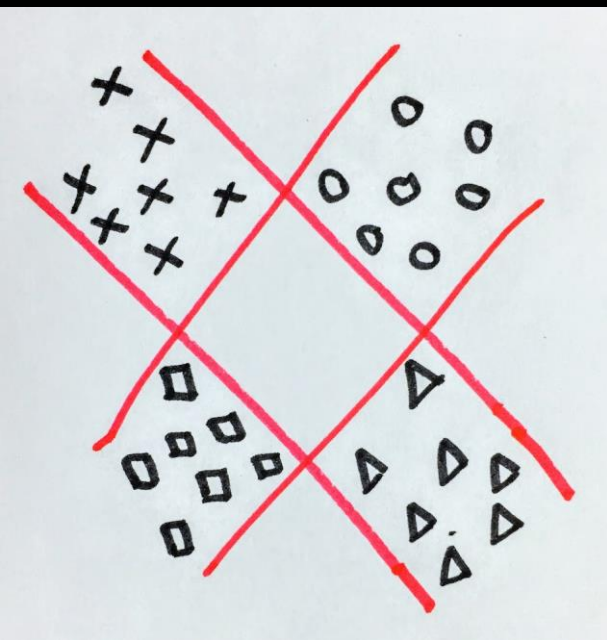
# 결정 경계 내 맘대로 (2 클래스)



하늘에서 본 모습



# 결정 경계 내 맘대로 (4 클래스)

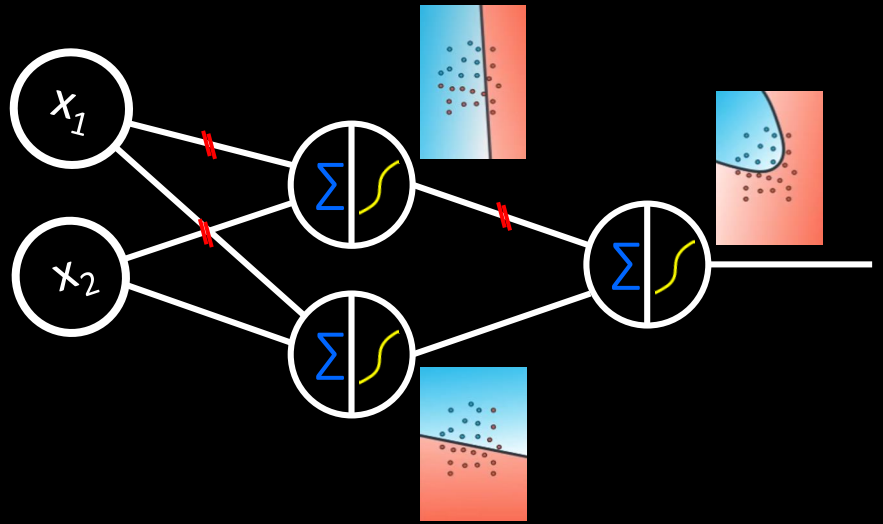


# 결정 경계 내 맘대로

- 더 복잡하고 섬세한 결정 경계를 만들려면
- 신경망을 더 Wide하고 Deep하게

# (실습) 18.py

- 3층 신경망
- 입력 - 은닉 - 출력
- 비선형 결정 경계



# The way of machine learning

- is sort of learns over and over again **just like human being**
- If it misrecognizes, we need tell it 'Nope, you were wrong!' which makes it **update its weights to do better next time.**
- Try it over and over again just like a child.

# Learning or Programming

“This (machine learning) is the next transformation...the programming paradigm is changing. Instead of programming a computer, you teach a computer to learn something and it does what you want”

— Eric Schmidt, Google



# Change of Paradigm

Not programming,  
but data-driven learning  
(parameter tuning)