

AI and Deep Learning

# 다층신경망과 비선형 결정경계

제주대학교

변영철

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

# 학습할 내용

- 학습과 결정경계
- 플레이스 홀더(Place holder)
- 선형 결정경계와 XOR 문제
- 다층 신경망과 비선형 결정경계
- 복잡한 비선형 결정경계 만들기

# 학습과 결정경계

대충 만들어진 결정경계



1. 파라미터 난수 초기화 ( $w, b$ )
2. TensorFlow에 의한 오류 함수  $E$ 의 계산 그래프 생성
3. 앞으로 전파 및 오류 계산
4. 체인 룰을 이용한 역전파 및 기울기 계산, 그리고  $w, b$  업데이트
5. 반복적으로  $w, b$  업데이트(goto 3)
6. 학습 완료 후 올바른 결정경계  $w, b$

# 학습하는 과정

```
import tensorflow as tf
```

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

```
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)
```

# 학습하는 과정

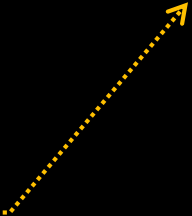
```
sess = tf.Session()  
sess.run(tf.global_variables_initializer())
```

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

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

```
#----- test (classification)  
x_data = [-2, 4]  
print(sess.run(hypo))
```

w를 1001번 업데이트  
하여 최적화



# 새로운 데이터로 테스트

- 학습 완료 후 제대로된 결정 경계에 의하여 정답을 잘 맞출 수 있음. 하지만..

```
#----- test (classification)
x_data = [-2, 4]
print(sess.run(hypo))
```

- **Failure!**
- 맨 처음 사용했던 x\_data가 사용되고 있음.
- 새로운 데이터가 hypo로 들어가지 않음.

# Place Holder

- 계산 그래프의 어떤 부분을 표시한 후
- 나중에 sess.run 함수 호출 시 그 부분에 데이터를 보냄.

`sess.run ( )`

# Place Holder

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

```
cost = -tf.reduce_mean(Y * tf.log(hypo) +  
    tf.subtract(1., Y) * 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 i in range(1001):  
    sess.run(train, feed_dict={X:x_data, Y:y_data})
```

```
if i % 100 == 0:  
    print(sess.run(w), sess.run(cost, feed_dict={X:x_data, Y:y_data}))
```



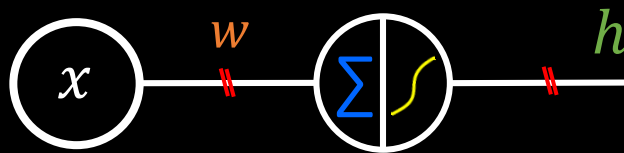
# Place Holder

```
#----- testing(classification)
x_data = [-2, 4]
result = sess.run(hypo, feed_dict={X: x_data})
print(result)
```

# (실습) 15.py

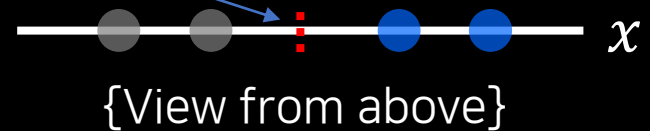
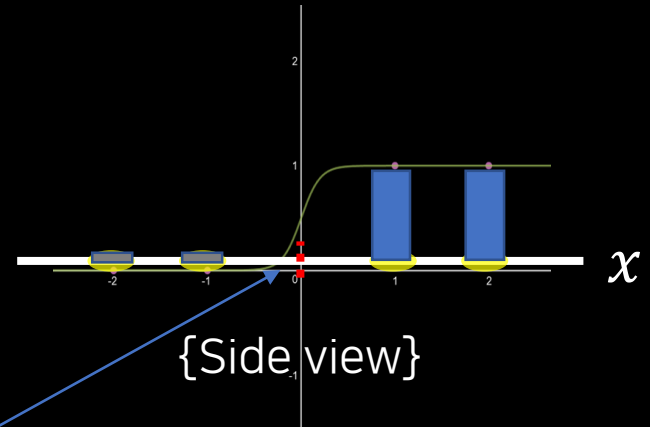
- 4가지 중 하나로 인식하기 →  
플레이스 홀더 이용

# 1-Input Neuron

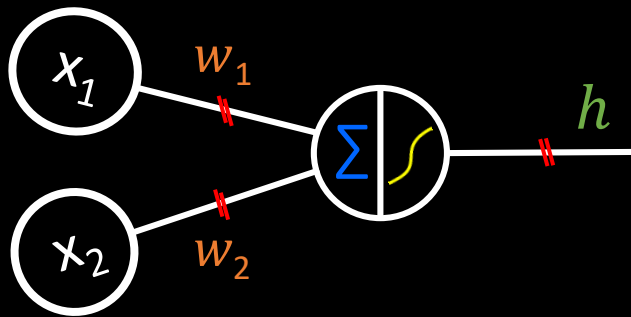


Decision boundary : Value

$$\begin{aligned}x \cdot w &= 0 \\x &= 0\end{aligned}$$

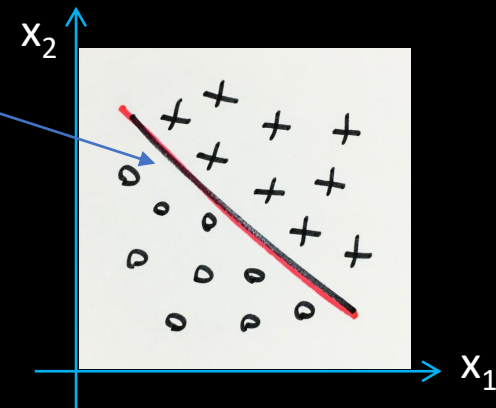
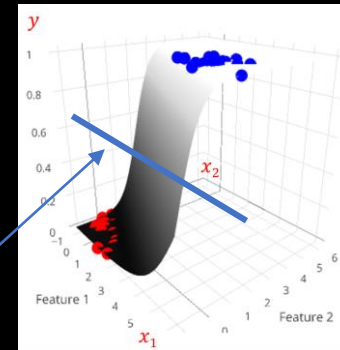


# 2-Input Neuron

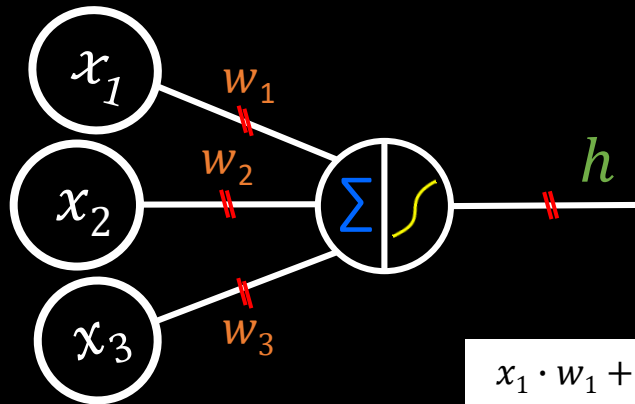


Decision boundary : **Line**

$$x_1 \cdot w_1 + x_2 \cdot w_2 = 0$$

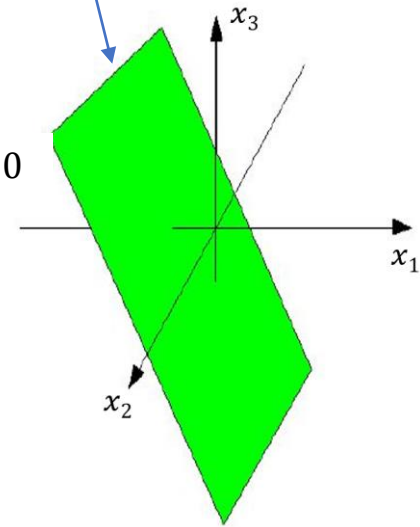


# 3-Input Neuron



Decision boundary : **Plane**

$$x_1 \cdot w_1 + x_2 \cdot w_2 + x_3 \cdot w_3 + 1 = 0$$

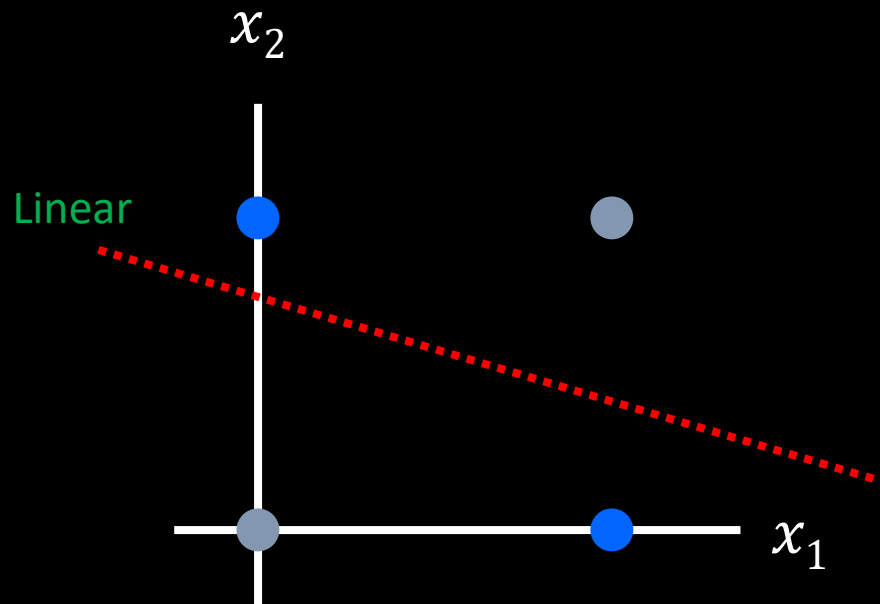
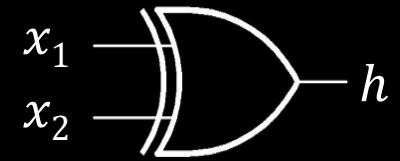


$$x_1 \cdot w_1 + x_2 \cdot w_2 + x_3 \cdot w_3 + x_4 \cdot w_4 = 0$$

4입력 이상이면  
→ 초평면(hyperplane)

이제까지는 모두  
선형 결정경계로 분류하는 문제

# XOR 문제



$x_1$	$x_2$	$h$
0	0	0
0	1	1
1	0	0
1	1	1

{View from above}

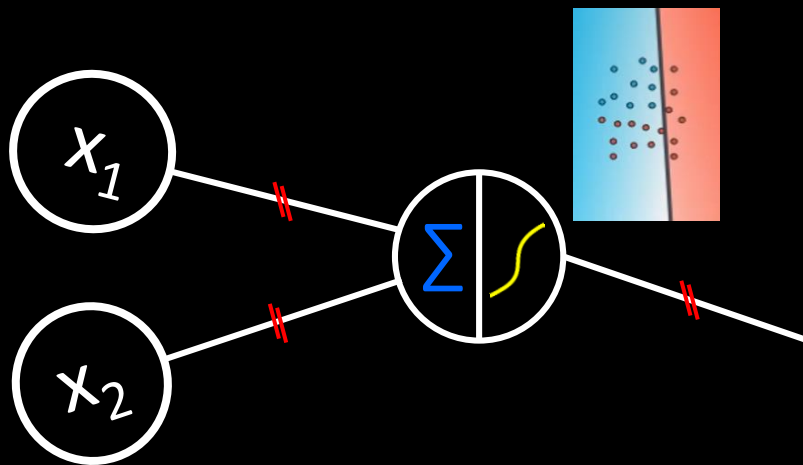


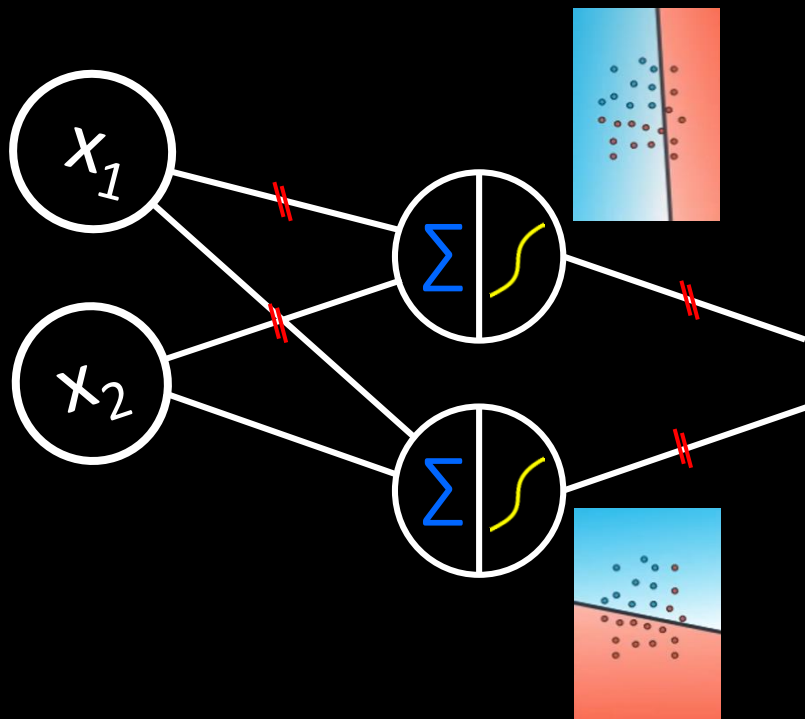
# XOR 문제

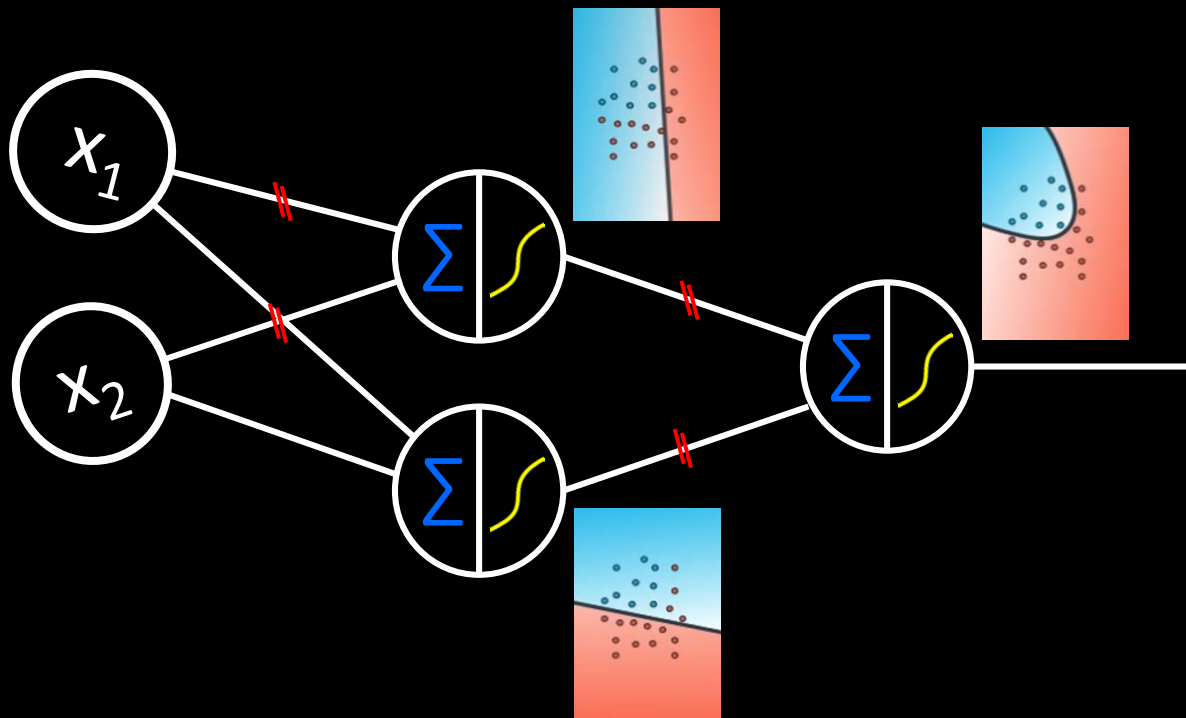
- 신경세포의 출력은 0 혹은 1
- 따라서 필요한 결정경계의 수는?
- 선형 결정경계 1개로는 불가능
- 비선형 결정경계가 필요

# (실습) 16.py

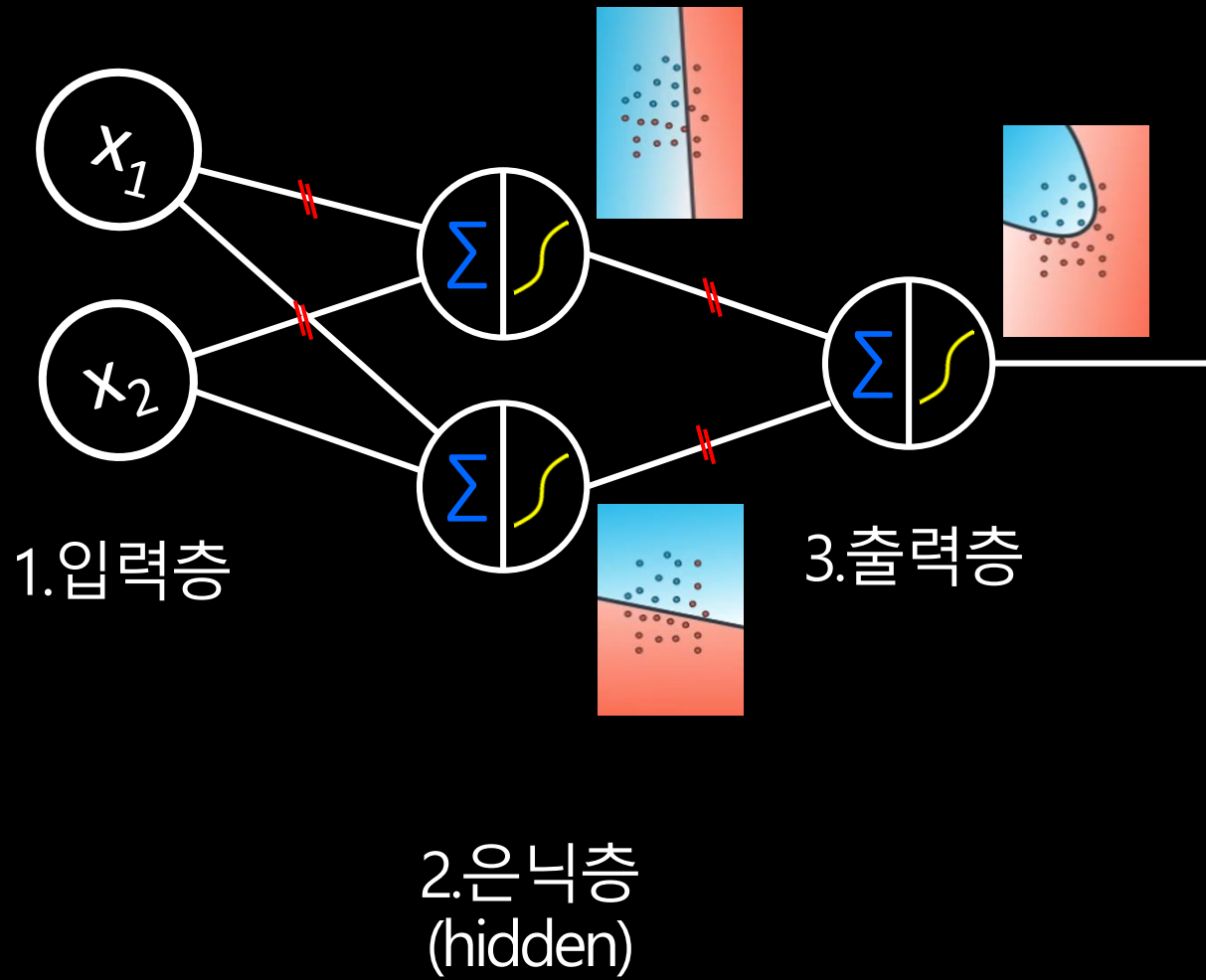
- 신경세포 하나
- 선형 결정 경계 1개
- 해결 불가능

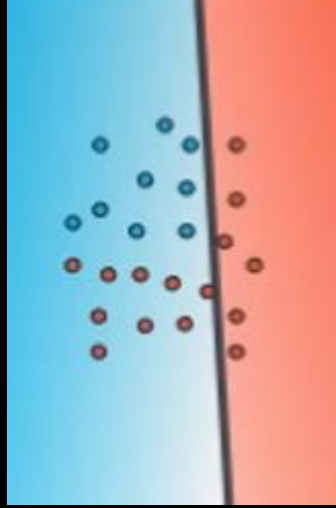




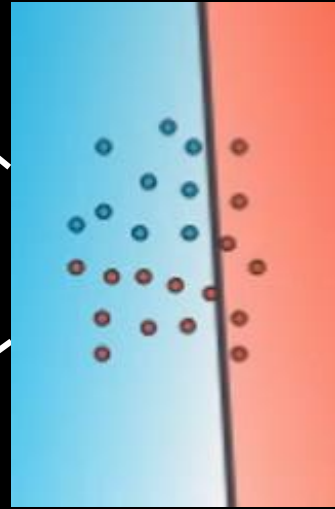


# 3층 신경망

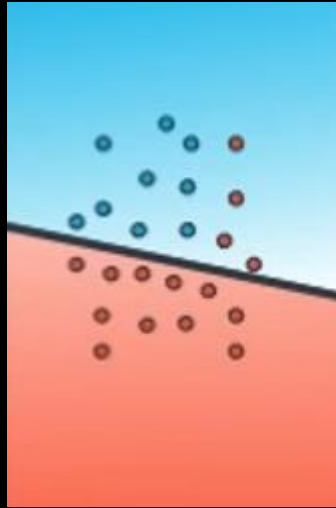


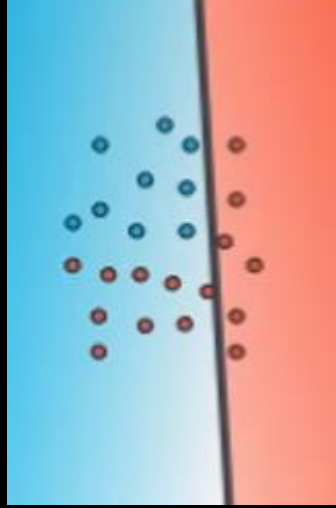


1

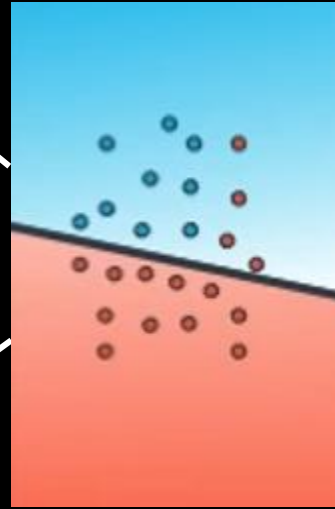


0

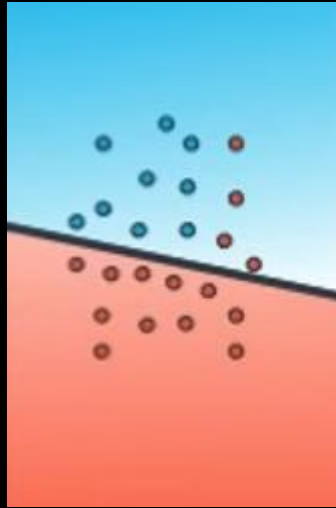




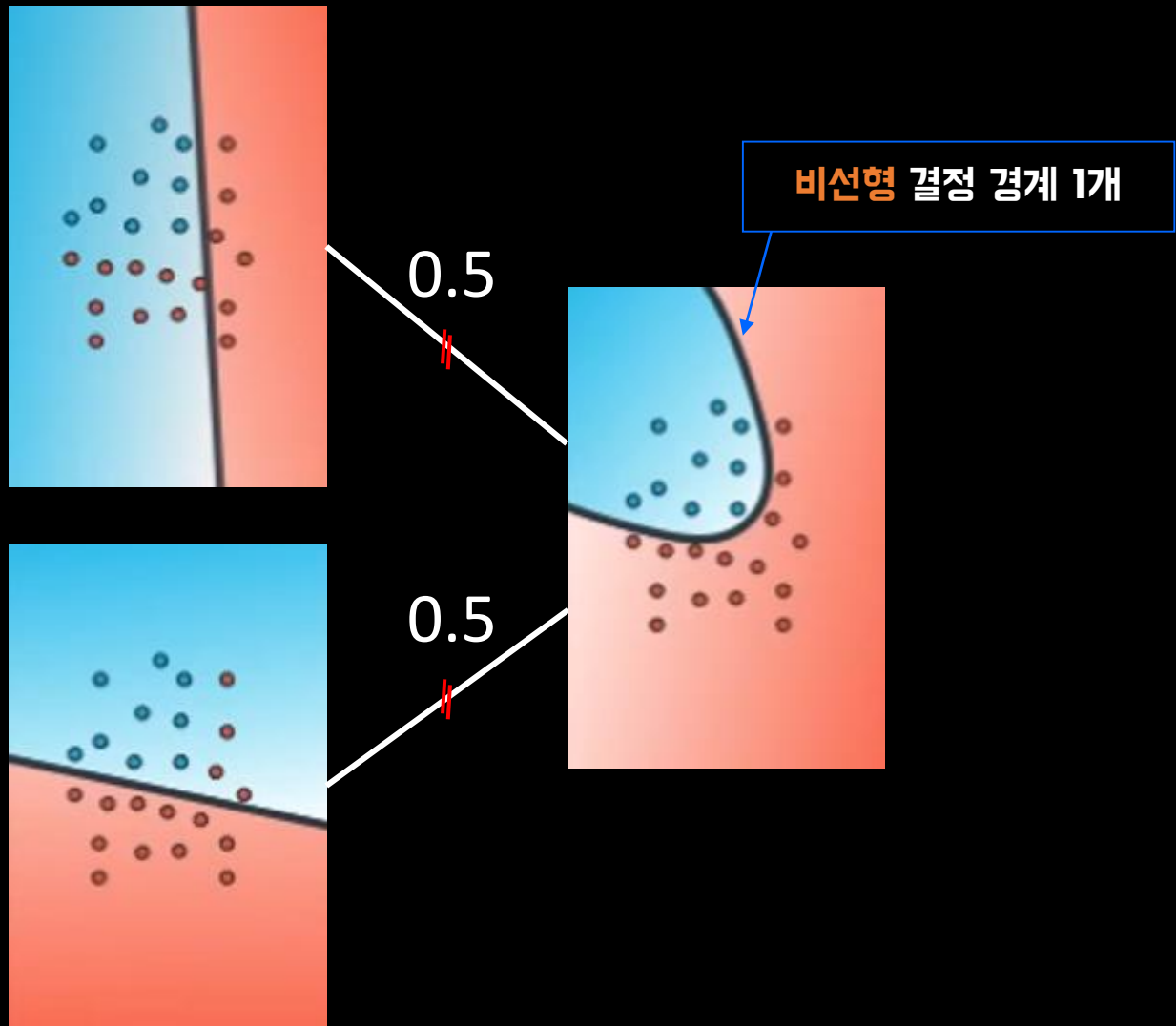
0



1





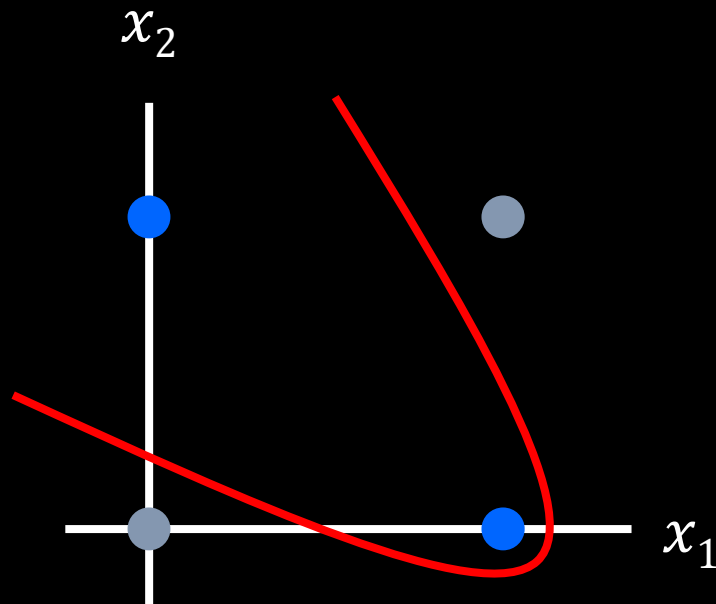


# 선형 결정경계 2개를 합쳐서 1개의 비선형 결정경계

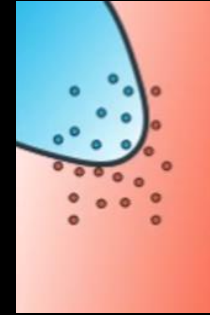
넓어질 수록 합칠 수 있는 결정경계가 많아지고,  
깊어질 수록 더욱 복잡한 결정경계를 만들어낸다.

# 비선형 결정 경계를 위한 3-층 신경망

# XOR



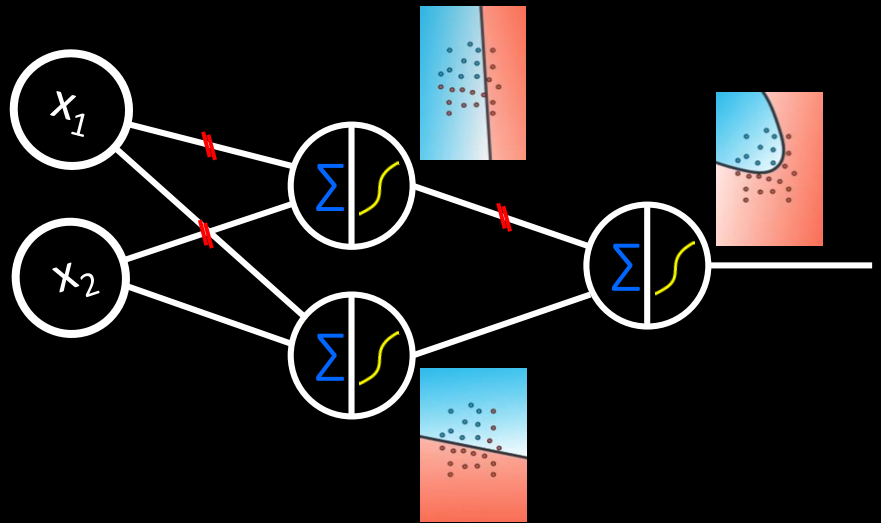
{위에서 본 모습}



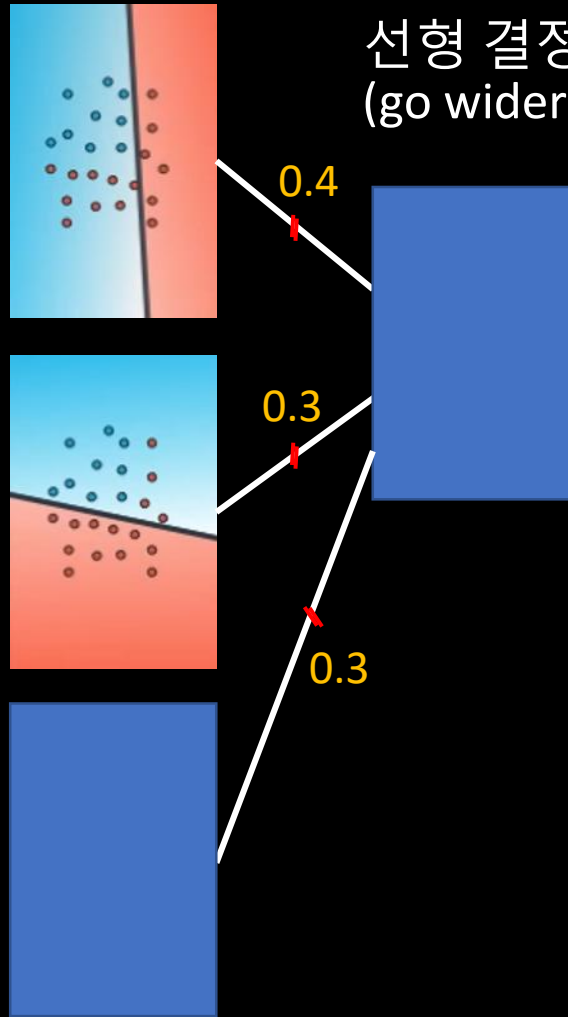
{옆에서 본 모습}

# (Lab) 17.py

- 3-층 신경망
- 입력층-은닉층-출력층
- 비선형 결정경계

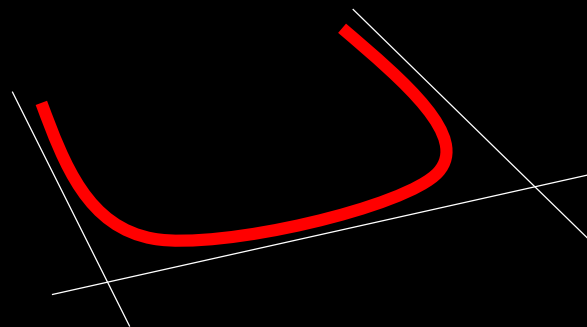


선형 결정경계 하나 더 추가  
(go wider), 그리고 조합



# 비선형 결정경계

- 3개의 선형 결정 경계를 조합하면

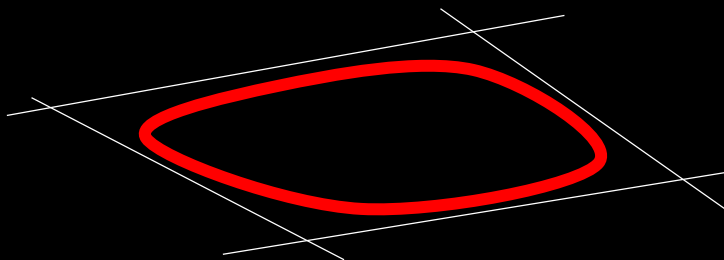






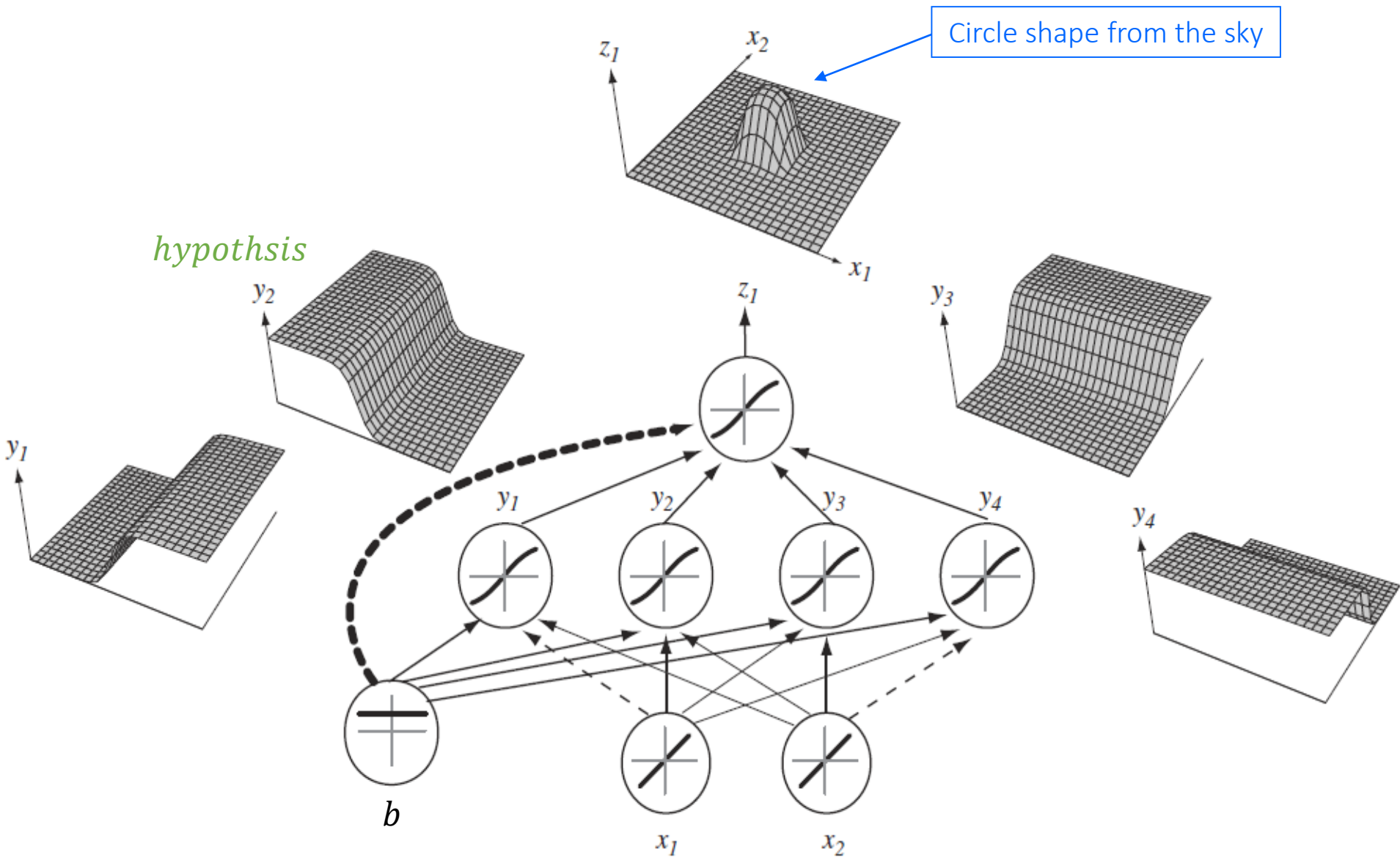
# 비선형 결정경계

- 4개의 선형 결정경계를 조합하면



View from above

# Nonlinear Decision Boundary



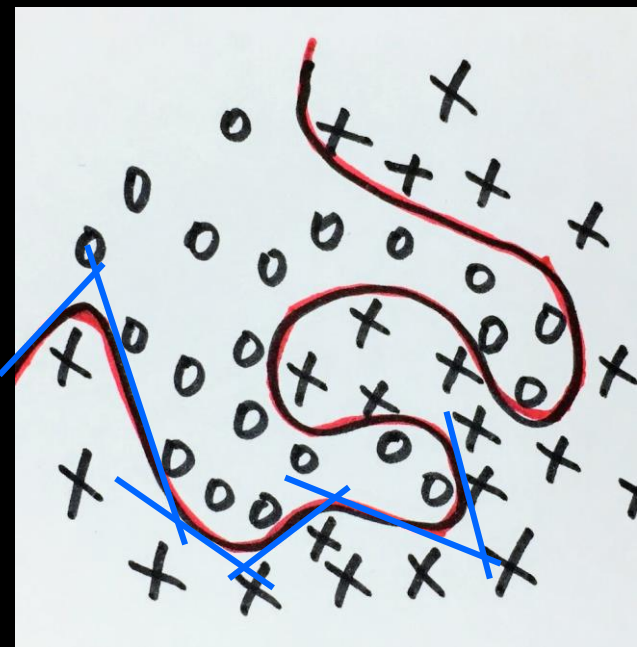
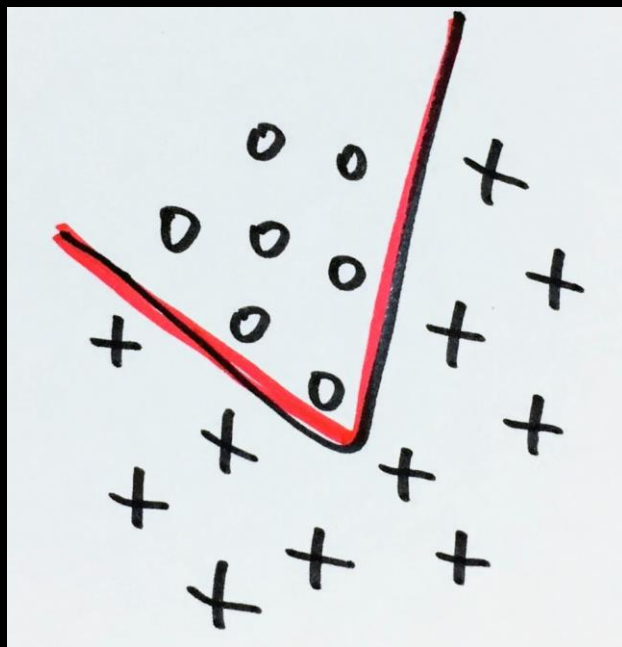
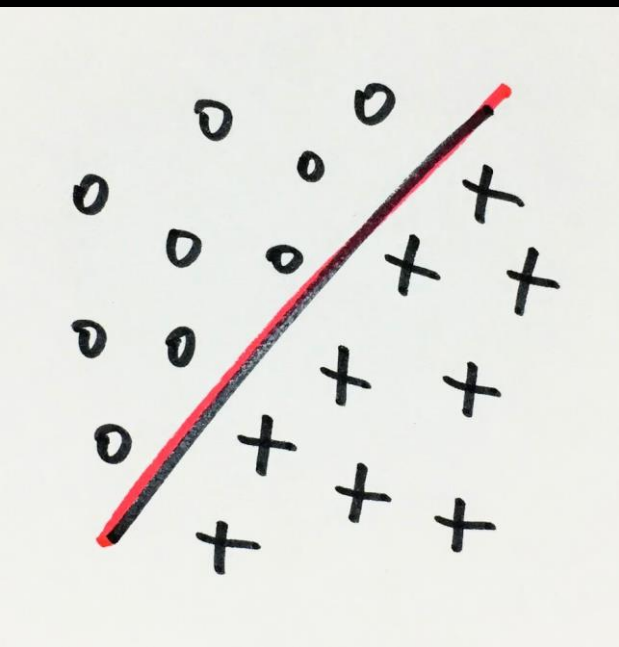


위에서 본 모습



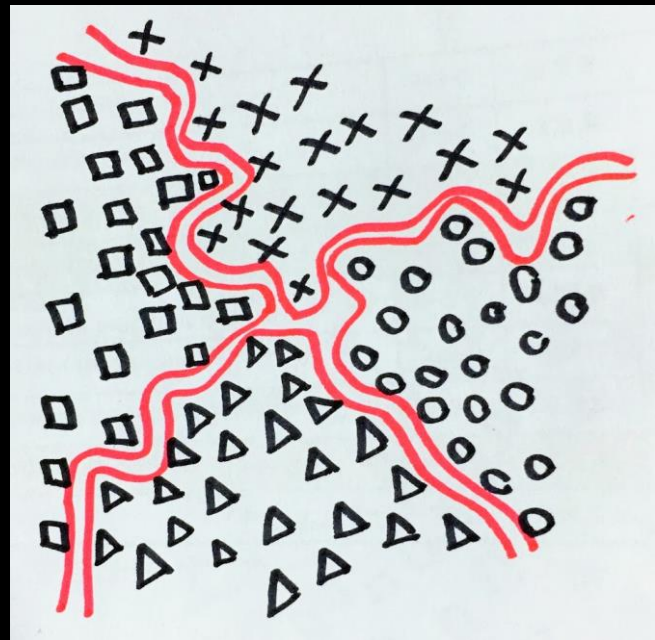
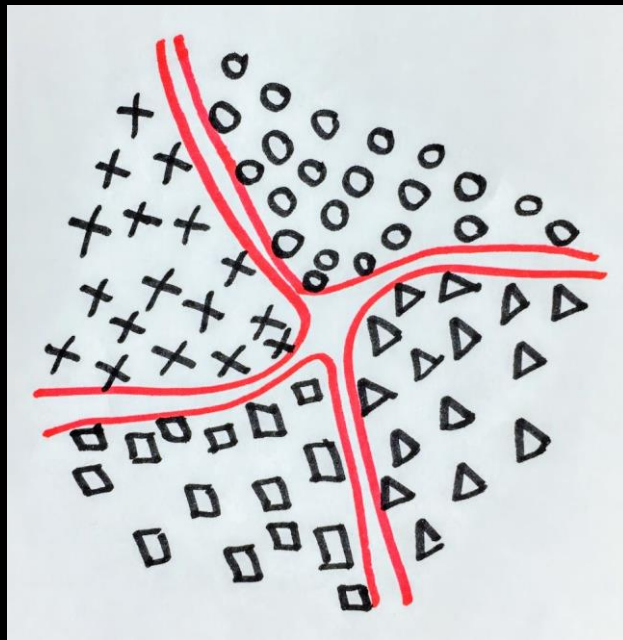
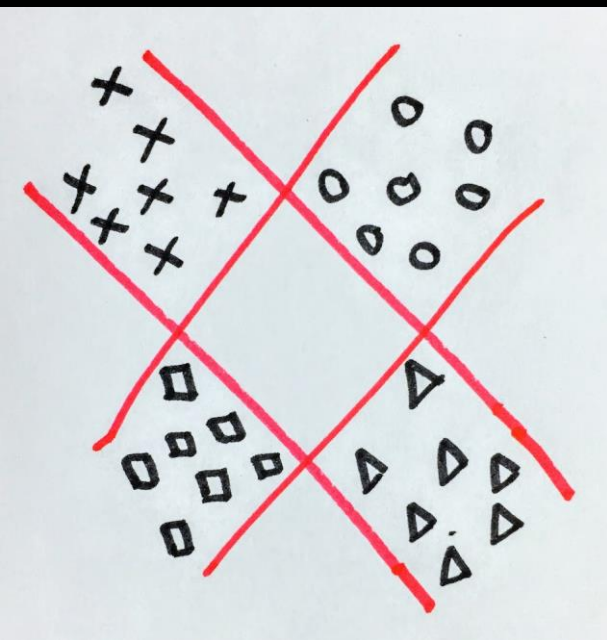
옆 모습

# 결정 경계 내 맘대로 (2 클래스)



하늘에서 본 모습

# 결정 경계 내 맘대로 (4 클래스)



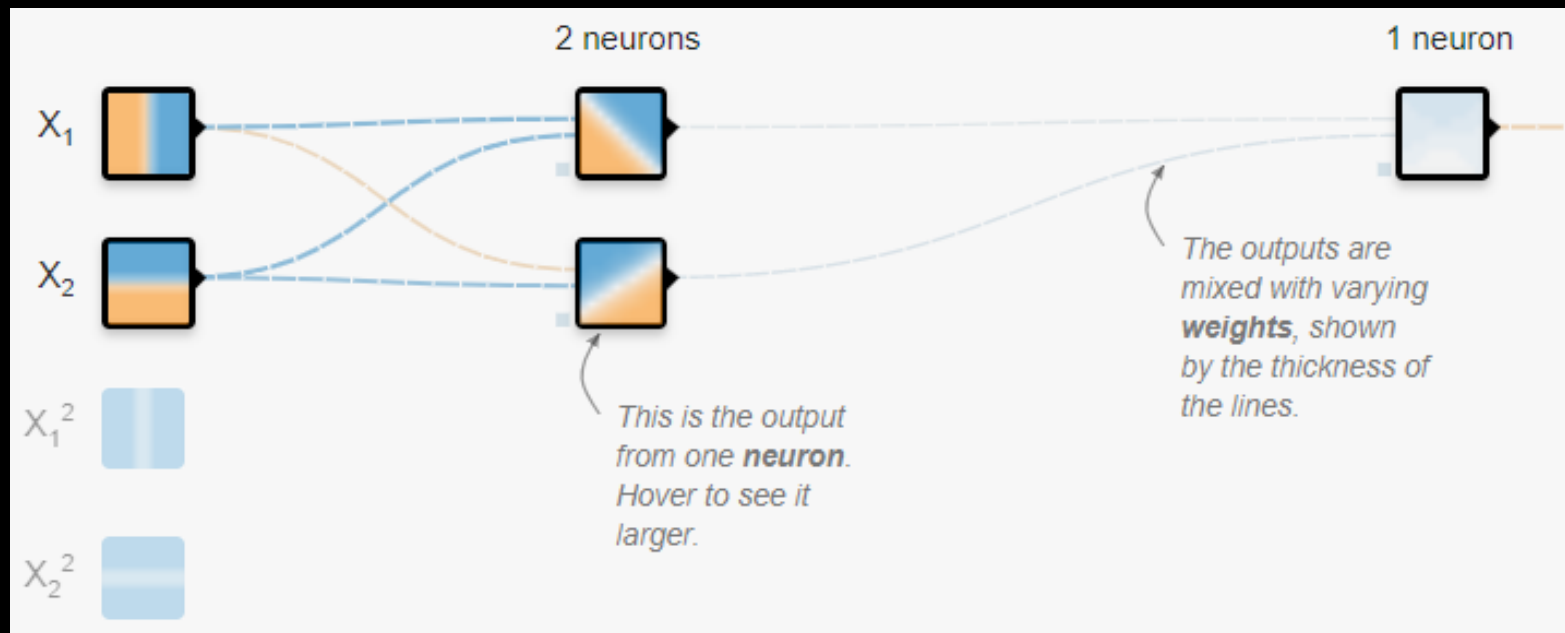


# 우리가 원하는 대로

## 더 깊고 더 넓게

- 더 복잡한 비선형 결정 경계를 만들 수 있음.
- 우리가 원하는 모든 것을 분류할 수 있음.

<http://playground.tensorflow.org>



# 머신러닝은

- 사람이 배우는 것같이 배우고 또 배우고
- 실수하면 '니가 틀렸어' 라고 얘기하면 됨.
- 그러면 다음에 더 잘하도록 가중치를 스스로 수정함.
- 아이처럼 계속해서...



# 학습이냐 프로그래밍이냐

“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



# 패러다임의 변화

이제는 프로그래밍이 아닌,  
데이터를 이용한 학습 (parameter  
tuning)