AI and Deep Learning

Multi-Layer Neural Networks and Non-linear decision boundary

Jeju National University
Yung-Cheol Byun

Learning

- What's going on in learning
  1. Randomly Initialized parameters(w, b)
  2. Computation graph of E by TensorFlow
  3. Foreword propagation to put values into the graph and to calculate E
  4. Back-propagation to get the influence of w, b on the error (chain rules)
  5. Update w, b (goto 3)
  6. Decision boundary adjusted properly with w, b

Learning

```
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)
```

Learning

```
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))
```

Learning finished after 1001 times updates

Testing new data

- Now, the neuron can classify new input data correctly.

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

- Failure!
- Old data(1) was used.
- No feeding the new data into the computational graph

Place Holder

- Marking certain places in computational graph using place holders
- and then replace it with real data when it runs (is evaluated).

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}))
```

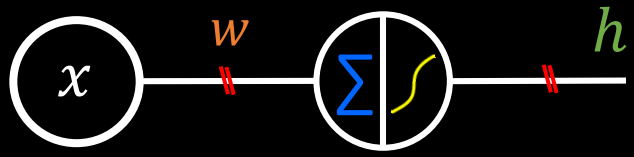
Place Holder

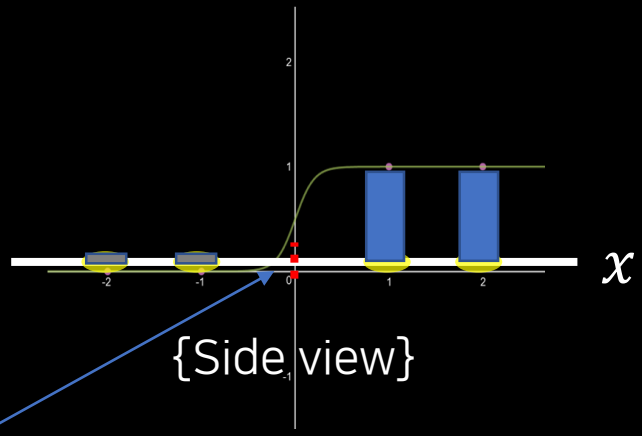
```
#----- testing(classification)
x_data = [-2, 4]
result = sess.run(hypo, feed_dict={X: x_data})
print(result)
```

(실습) 15.py

- 4가지 중 하나로 인식하기 → 플레이스 홀더 이용

1-Input Neuron





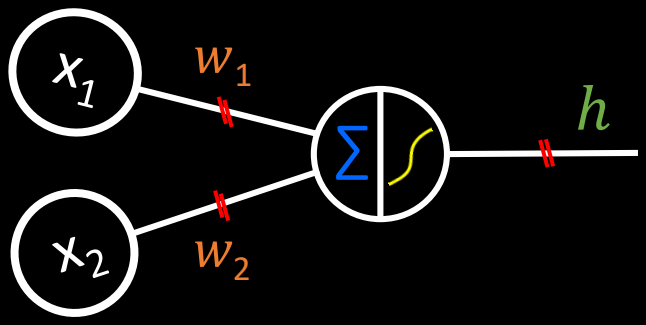
{Side view}

Decision boundary : Value

$$x \cdot w = 0$$
$$x = 0$$

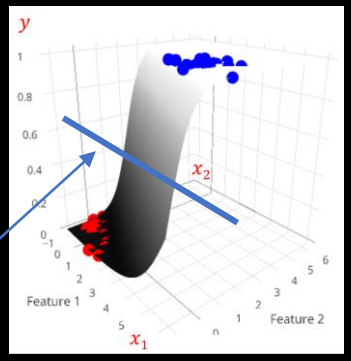
{View from above}

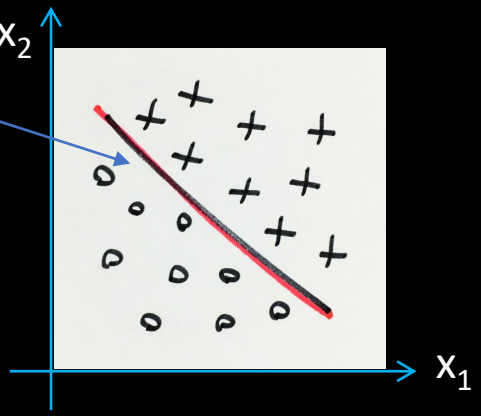
2-Input Neuron



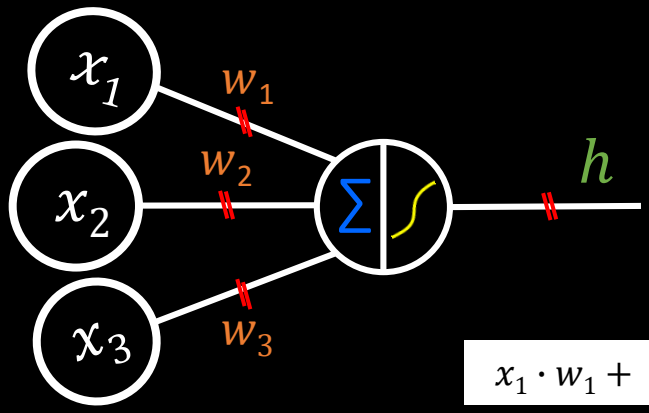
Decision boundary : Line

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



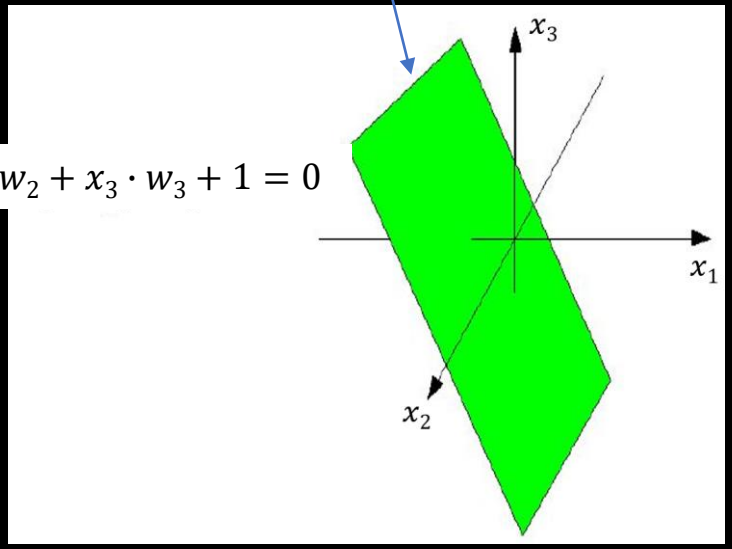


3-Input Neuron



Decision boundary : Plane

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

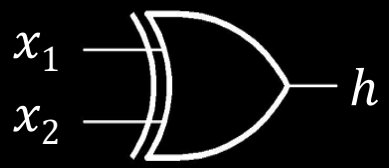


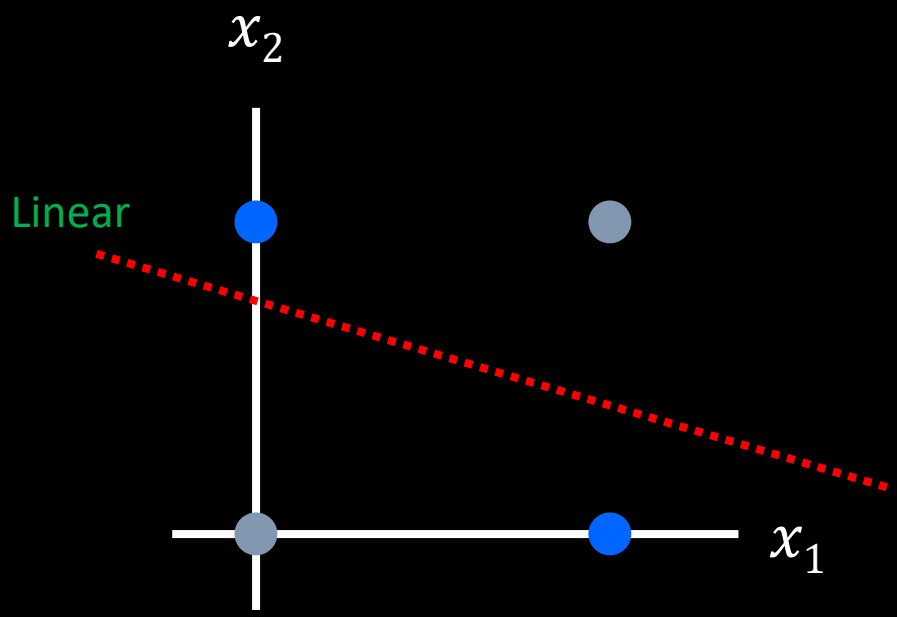
$$x_1 \cdot w_1 + x_2 \cdot w_2 + x_3 \cdot w_3 + x_4 \cdot w_4 = 0$$

More than 4 inputs
→ hyperplane

이제까지는 모두
선형 결정경계로 분류하는 문제

XOR





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

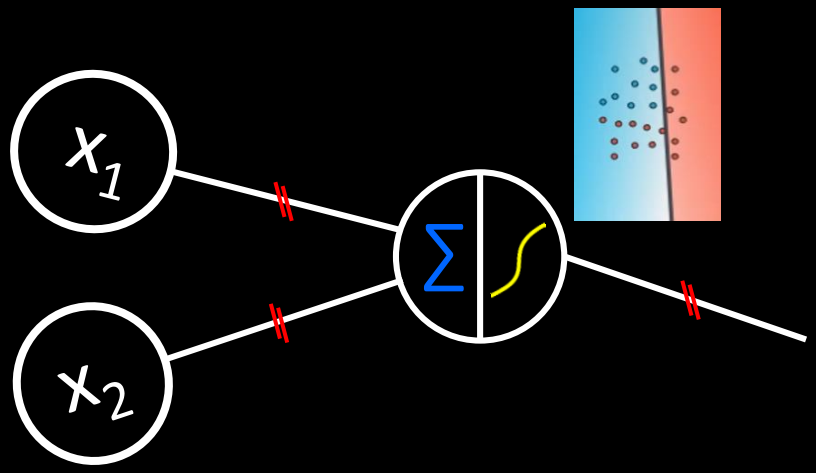
{View from above}

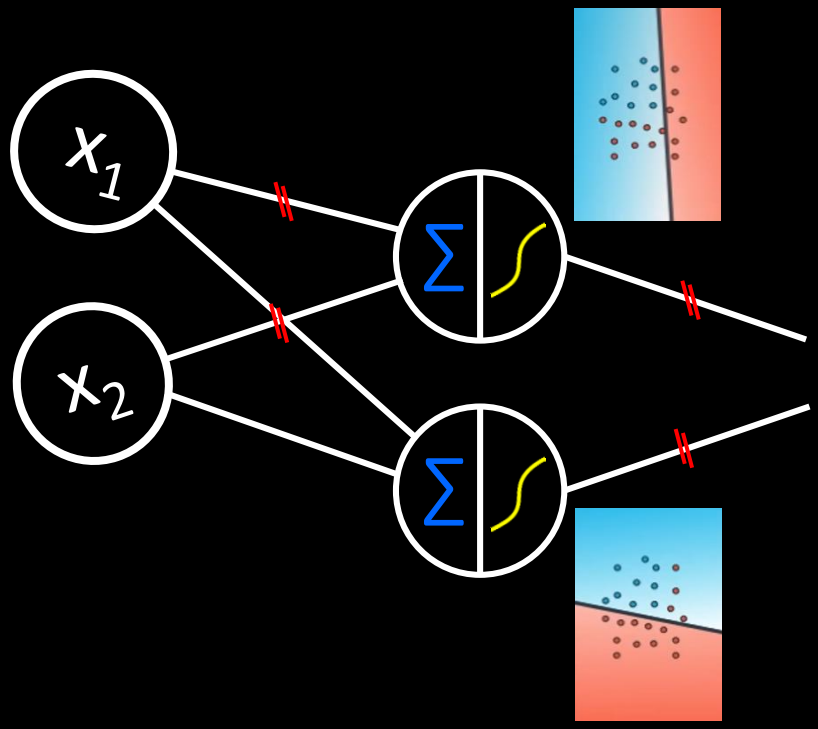
XOR 문제

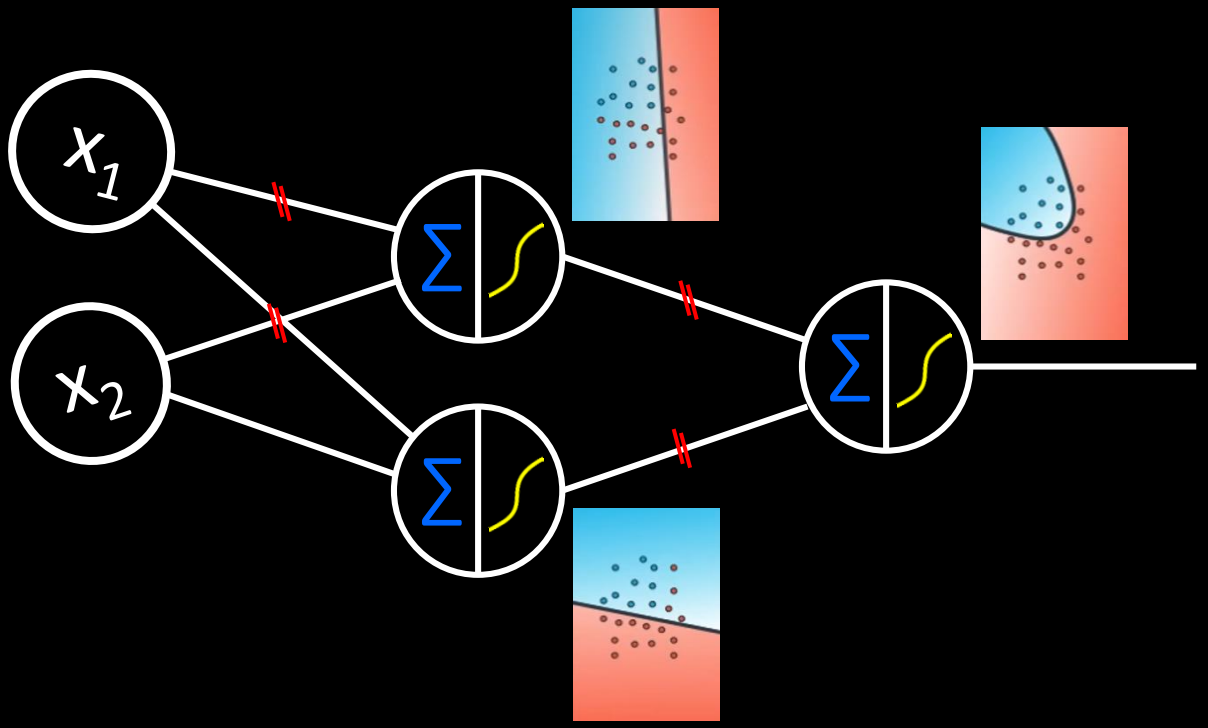
- 클래스 수는?
- 따라서 필요한 결정경계의 수는?
- 선형 결정경계 1개로는 불가능
- 선형 결정경계 2개로도 불가능 (왜?)
- 비선형 결정경계 1개가 필요

(실습) 16.py

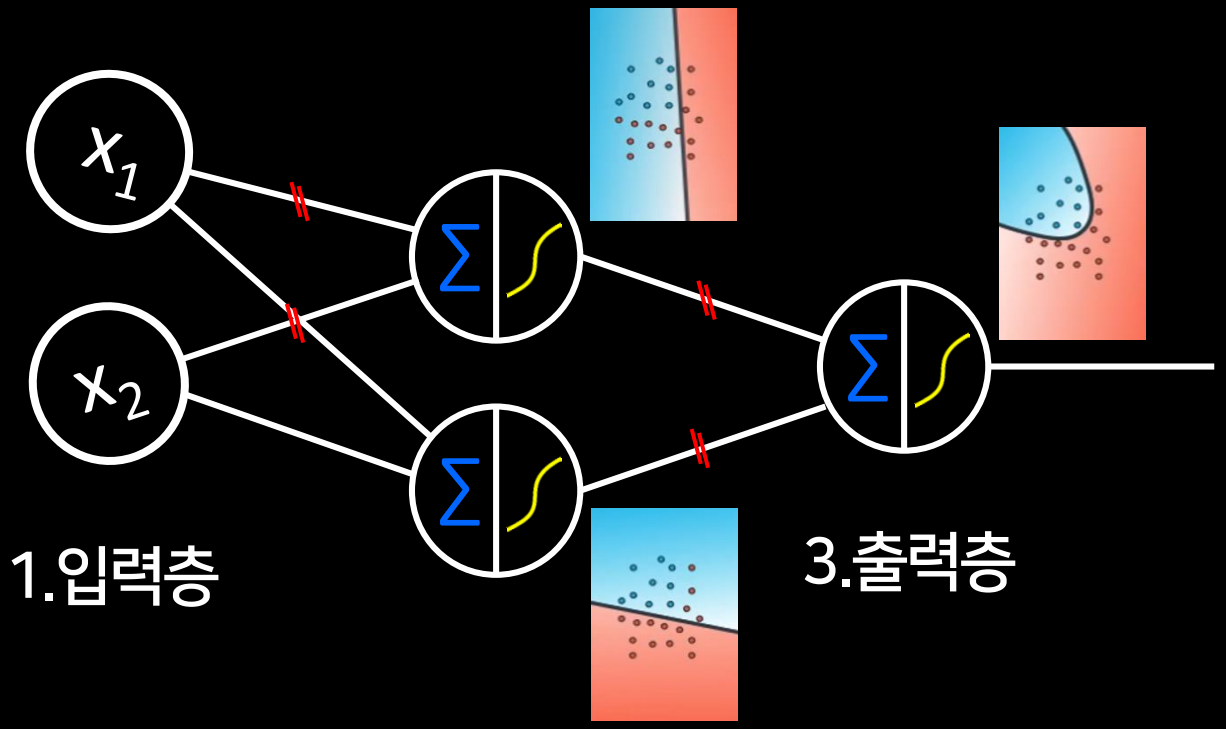
- 신경세포 하나
- 선형 결정 경계 1개
- 해결 불가능


x_1
x_2
Σ


x_1
x_2
Σ
Σ


x_1
x_2

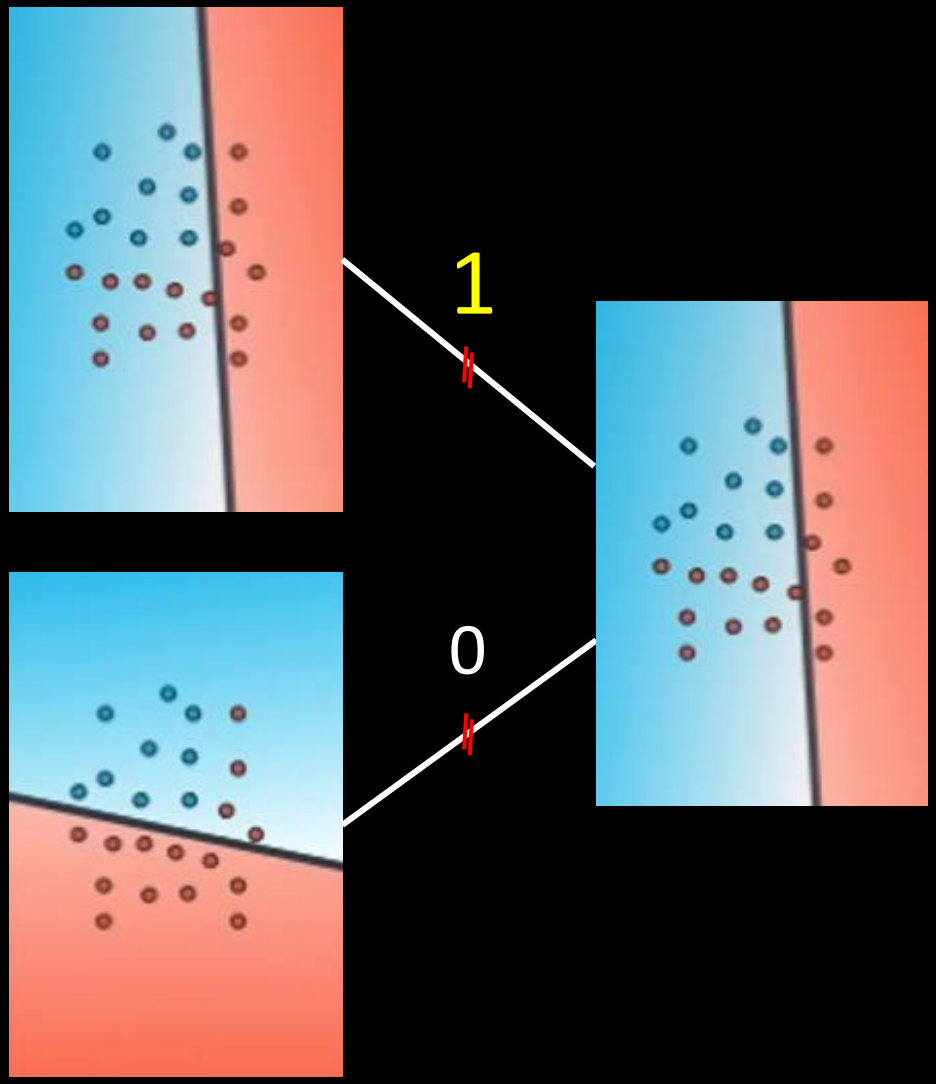
3층 신경망

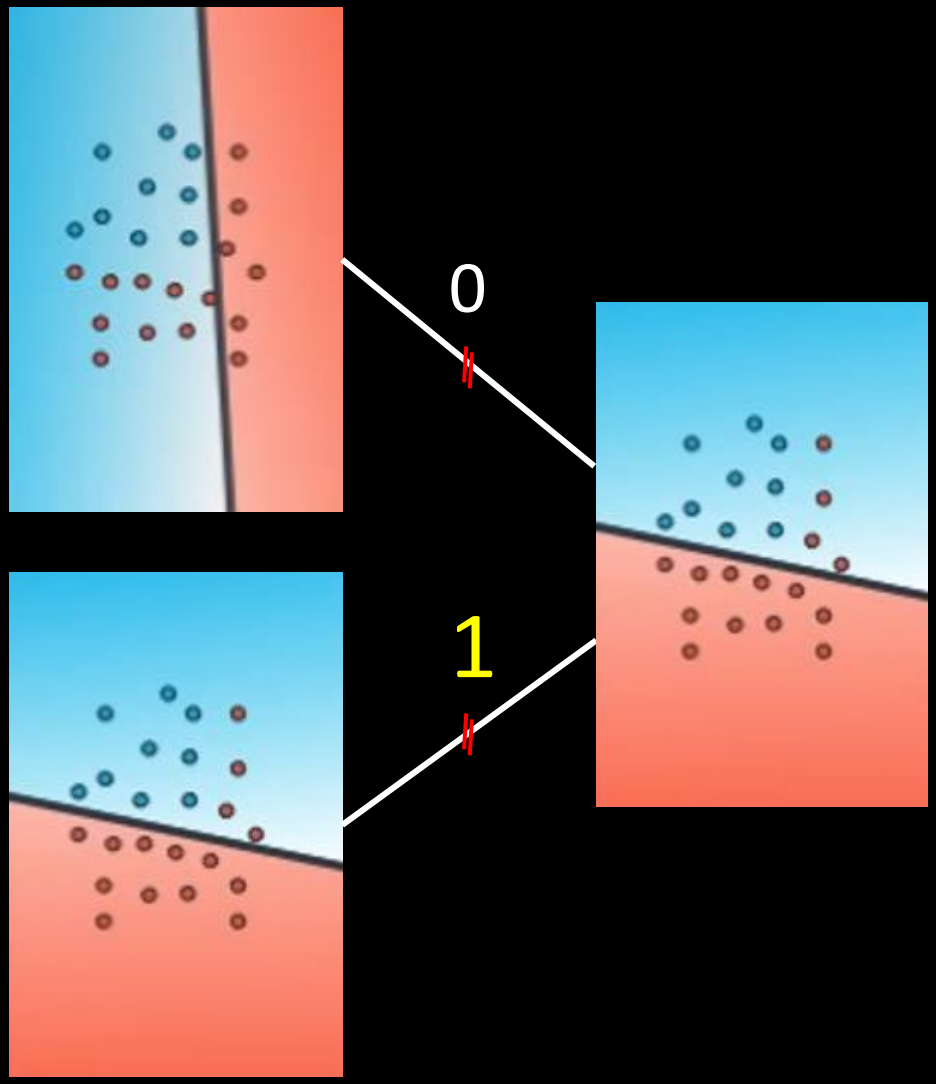


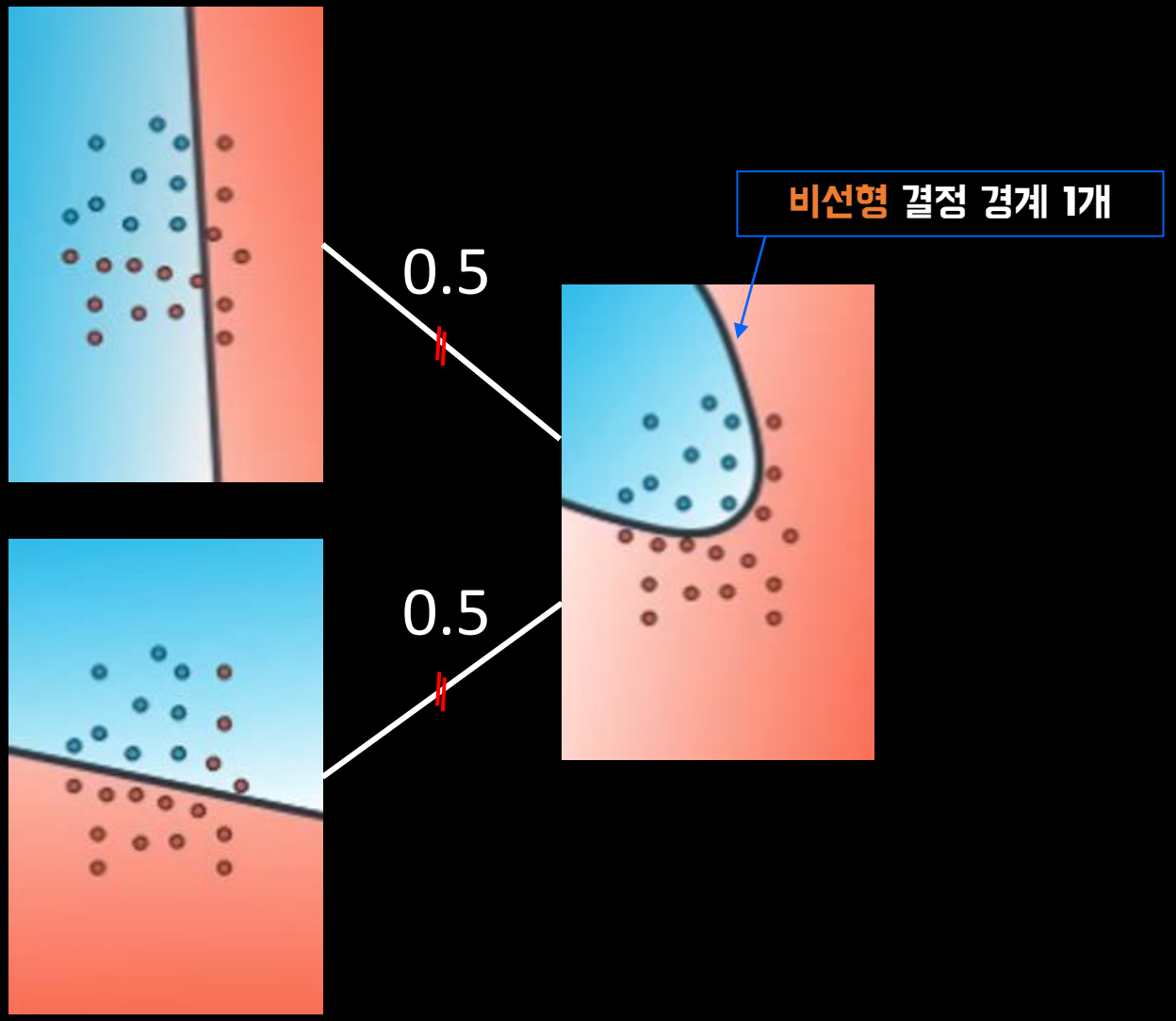
1.입력층

2.은닉층
(hidden)

3.출력층

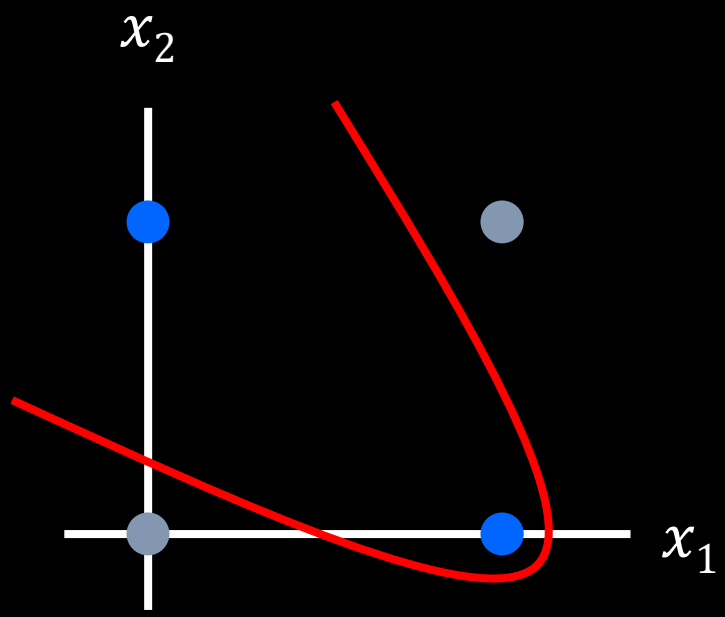

1
0


0
1

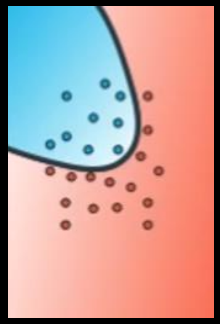

비선형 결정 경계 1개
0.5
0.5

3-layer NN for nonlinear decision boundary

XOR

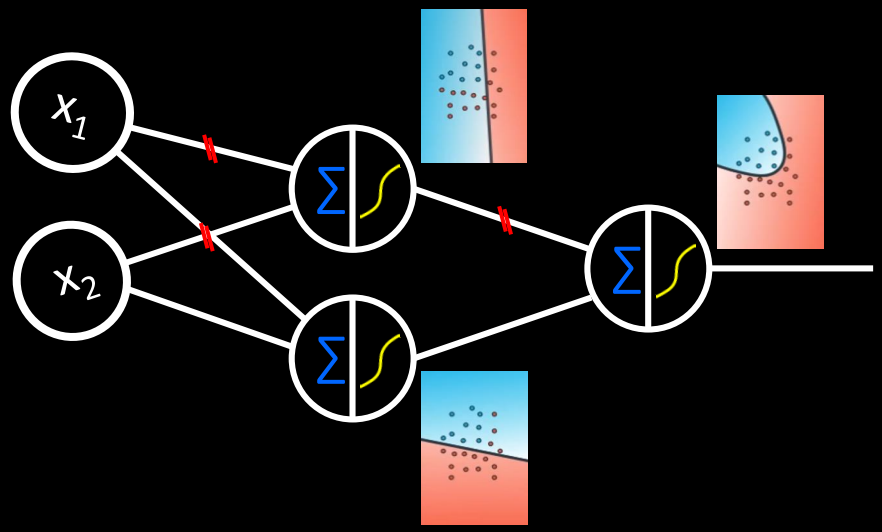


{View from above}



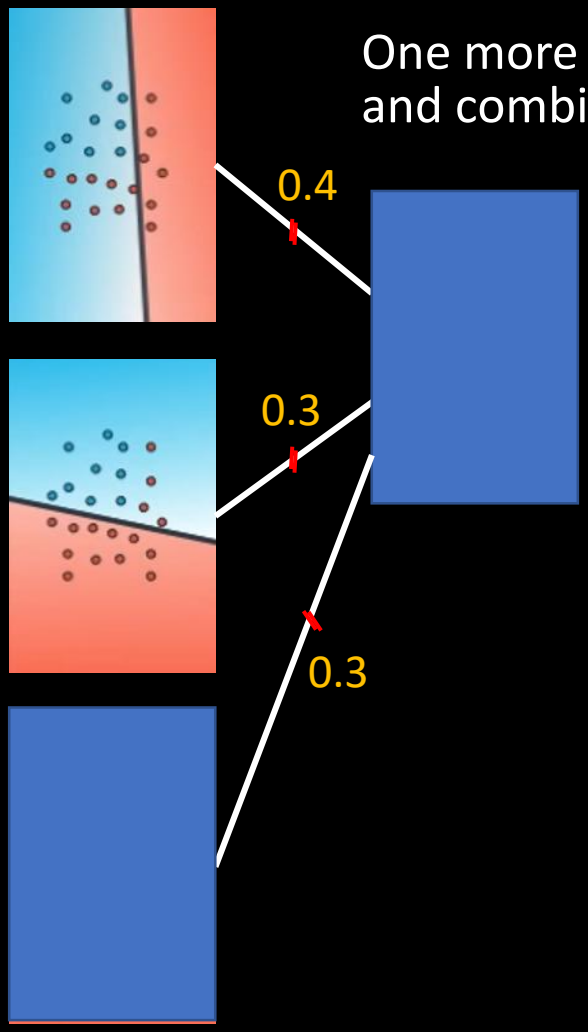
{Side view}

(Lab) 17.py



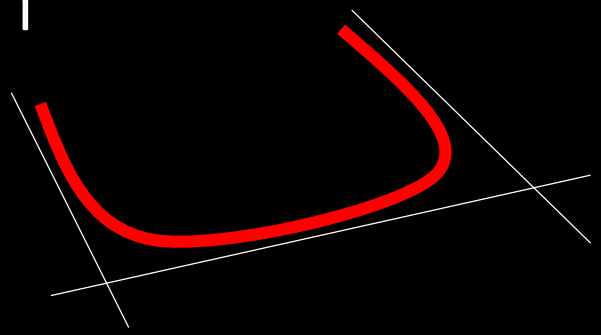
- 3-layer NN
- Input-Hidden-Output
- Nonlinear decision boundary

One more linear (go wider)
and combination



Nonlinear Decision Boundary

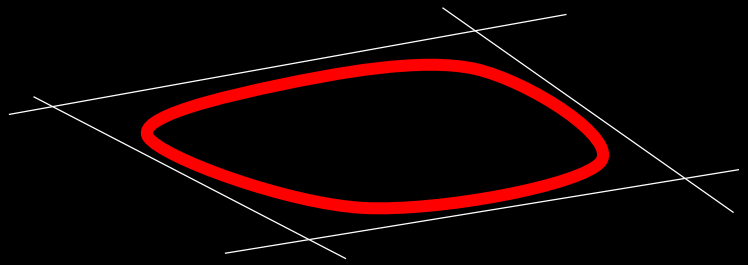
- Combination of three linear decision boundaries




JUPITERIMAGES

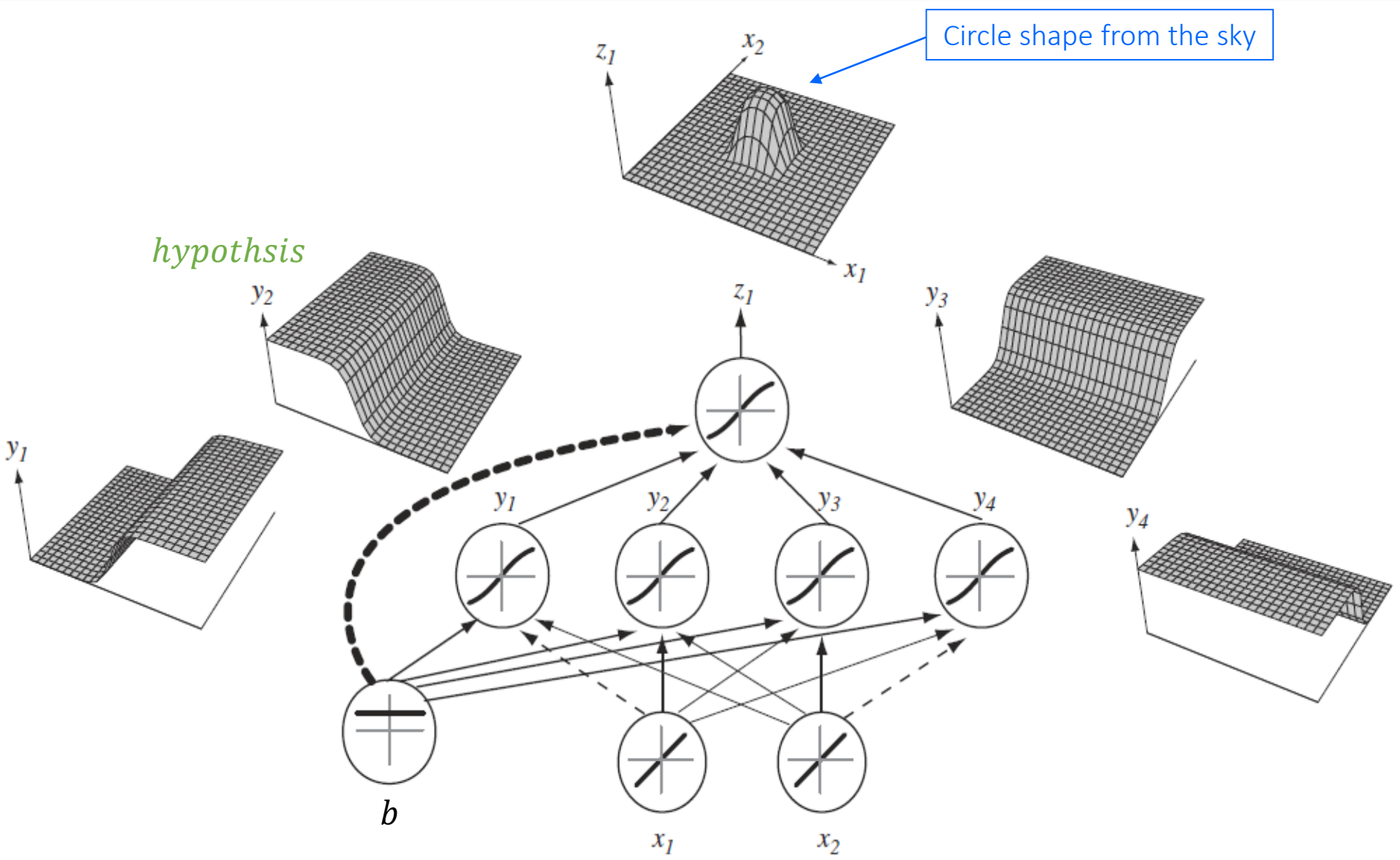
Nonlinear Decision Boundary

- Merging four linear decision boundaries



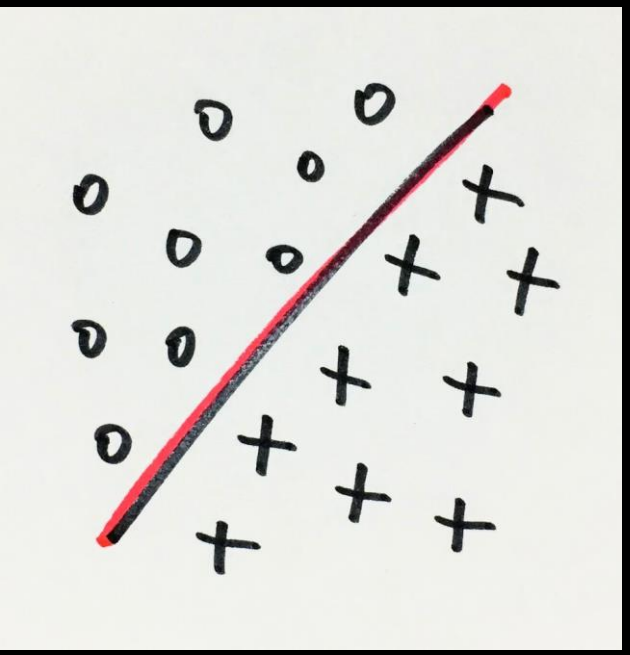
View from above

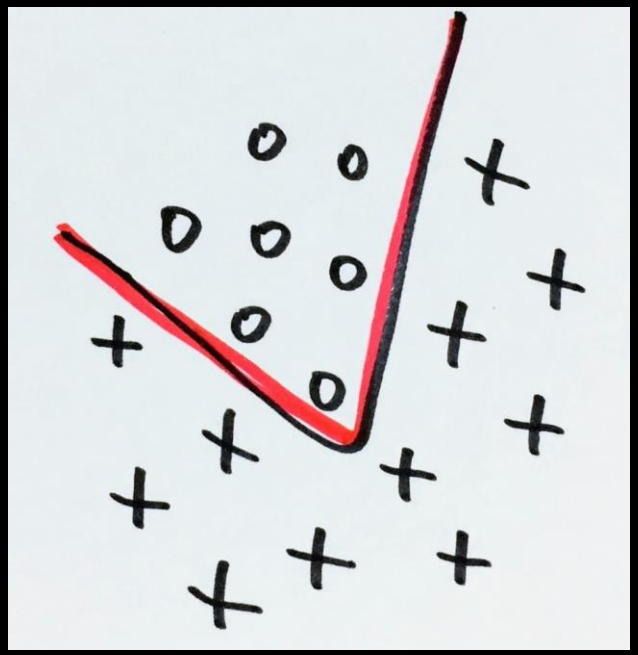
Nonlinear Decision Boundary

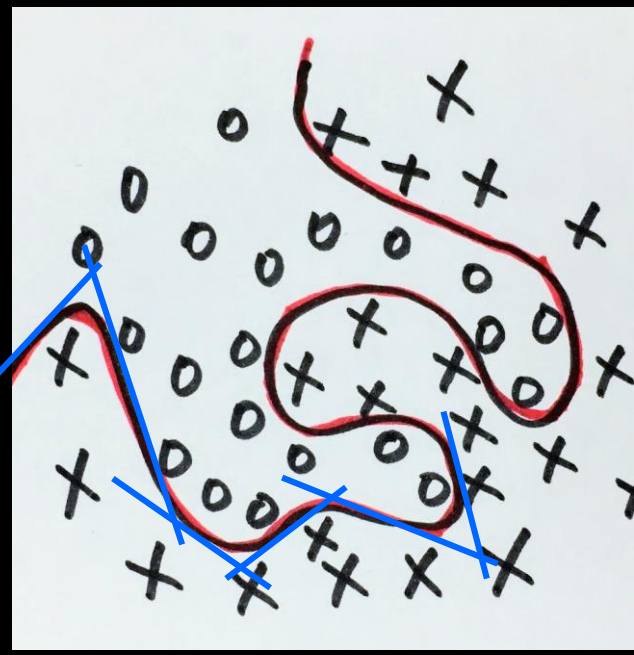


Side view

결정 경계 내 맘대로 (2 클래스)

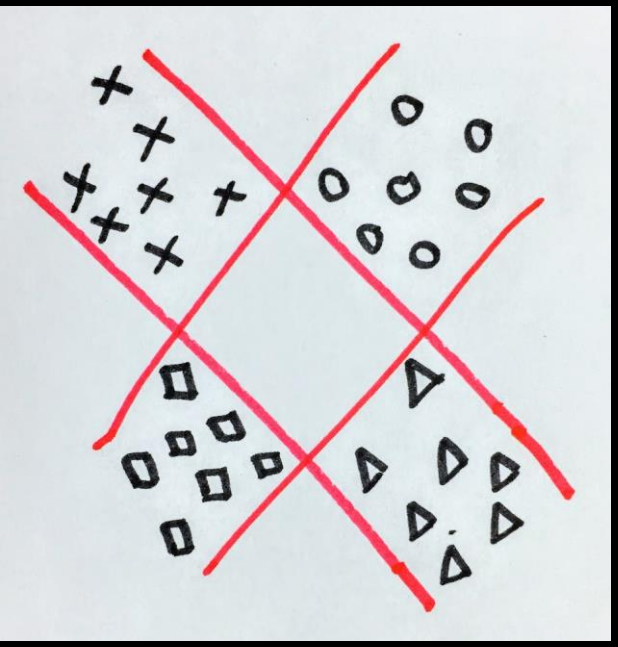


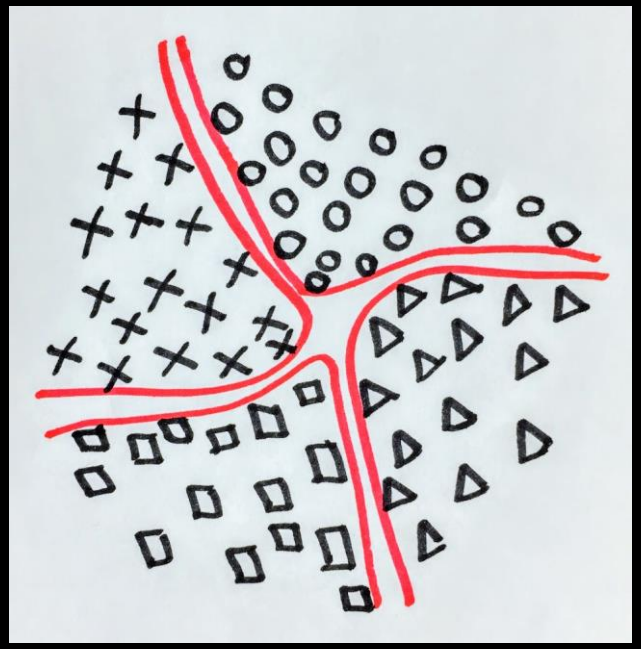


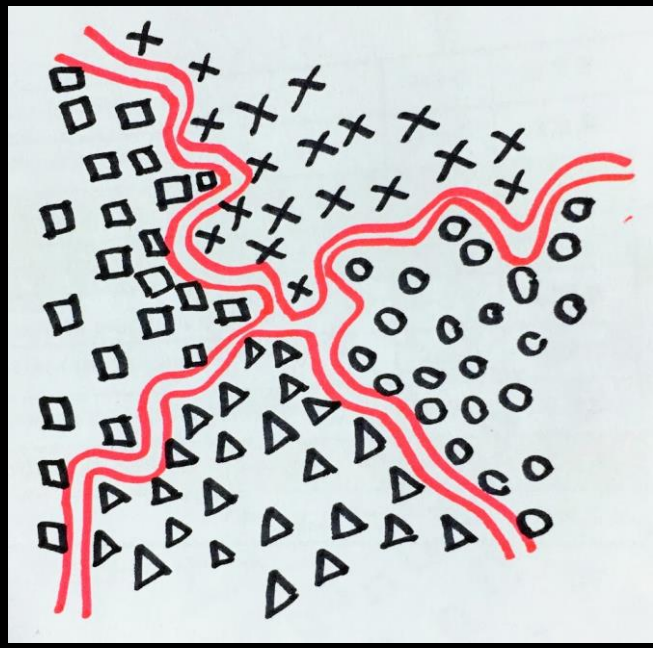


하늘에서 본 모습

결정 경계 내 맘대로 (4 클래스)







As you wish

go wider & deeper

- to make more complex nonlinear decision boundary
- We can classify anything we want.

The way of machine learning

- learning 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)