AI and Deep Learning

Multi-Layer Neural Networks and Non-linear Decision Boundary

Jeju National University

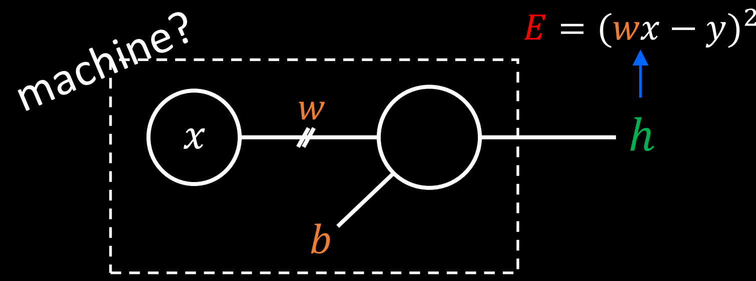
Yungcheol Byun

Agenda

- Machine Learning revisited
- Linear decision boundary
- XOR problem
- Non-linearity and multi-layer neural networks
- Complex nonlinear boundary as you like

Machine Learning

A neuron



① Parameters(w, b) initialization randomly

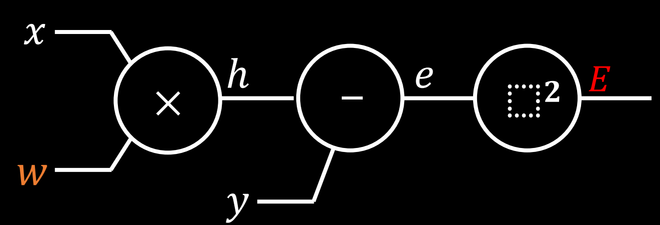
② Building computation graph of E by TensorFlow framework

③ Foreword propagation by putting values into the graph and calculate E

④ Back-propagation to get the influence of w, b on the error (applying chain rules)

⑤ Update w, b to adjust a decision boundary

⑥ go to ③



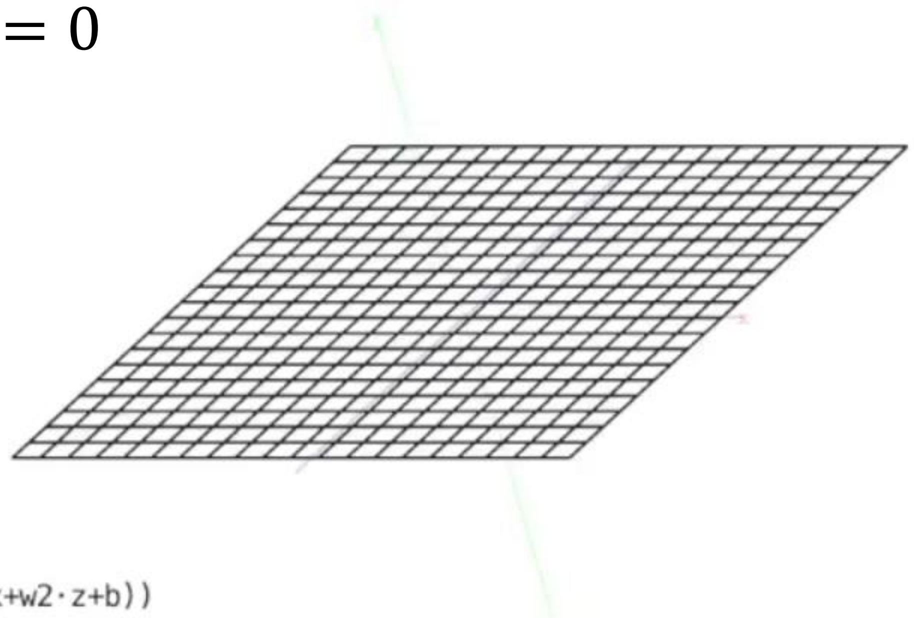
How to change the decision boundary

sigmoid(w1·length + w2·width + b)

db slope   db shift

$$w_1 x_1 + w_2 x_2 + b = 0$$

db rotation



surface(f(x,z)=sig(w1·x+w2·z+b))

slope → w1 = 0.00

rotation → w2 = 0.00

shift → b = 0.00

Machine 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]))        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)
```

Machine Learning

```
sess = tf.Session()
sess.run(tf.global_variables_initializer())  ①
②
for i in range(1001):
    sess.run(train)  ③④⑤

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

Learning finished after 1001 times updates

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

*hypo = tf.sigmoid(x_data * w)*

Testing new data

- After learning,
- a neuron can classify new input data correctly.

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

- When the computation graph is created, data is set only one time.
- **Still old data in the computation graph.**
- No feeding the new data[-2,4] into the precreated computational graph
- Therefore, not working!

Place Holder

- Marking certain place holders in a computational graph
- and then replace it with new data when it runs(evaluation).

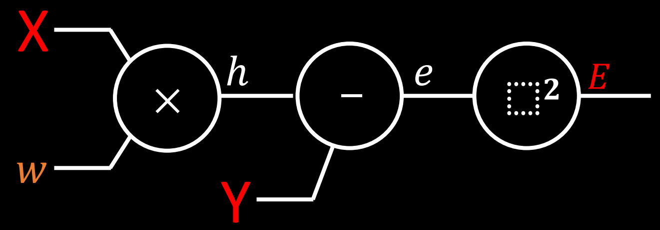
sess.run ( )

Place Holder

```
import tensorflow as tf

#----- training data
x_data = [[-2], [-1], [1], [2]]
y_data = [[0], [0], [1], [1]]

X = tf.placeholder (tf.float32)
Y = tf.placeholder (tf.float32)

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

1. Place holders : X, Y
2. x_data → X, y_data → Y
3. Feeding real data when hypo, cost, and train operations are executed.

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}))
```

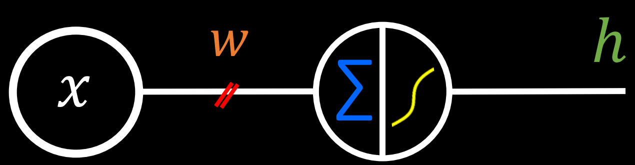
Place Holder

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

Lab 15.py

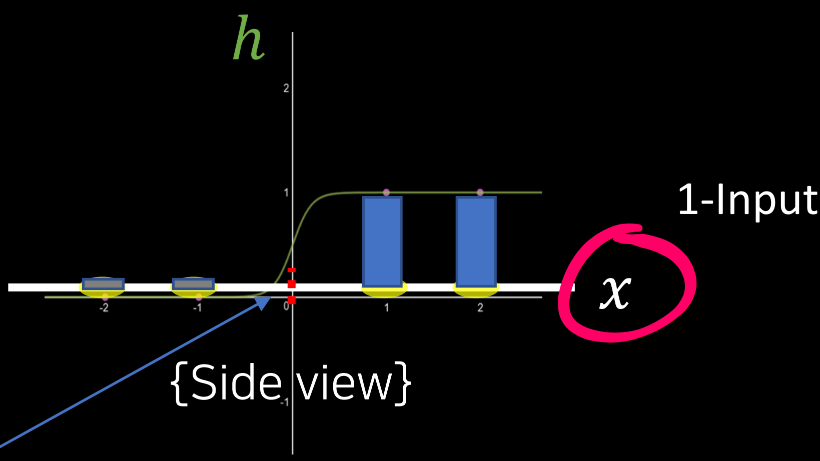
- Classification into one of four classes
- Using placeholders

1-Input Neuron

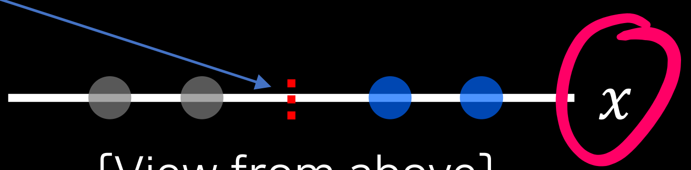


Decision boundary : Value

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

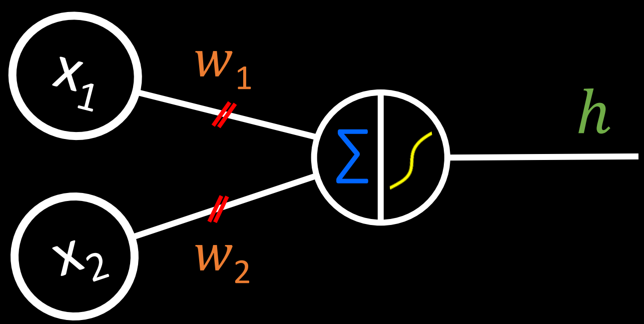


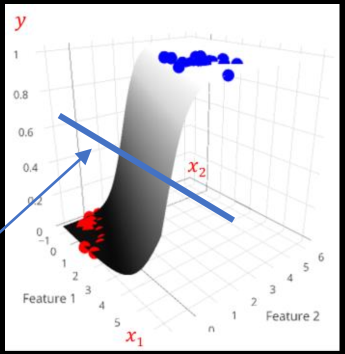
{Side view}



{View from above}

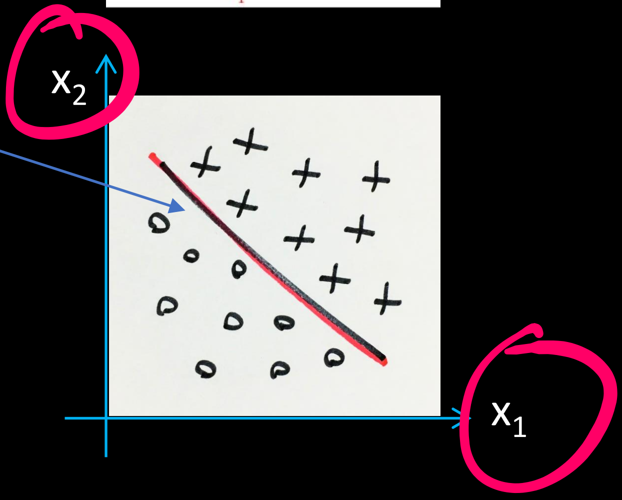
2-Input Neuron



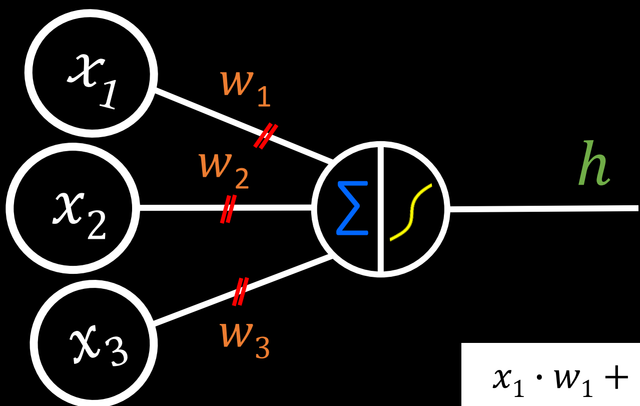


Decision boundary : Line

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

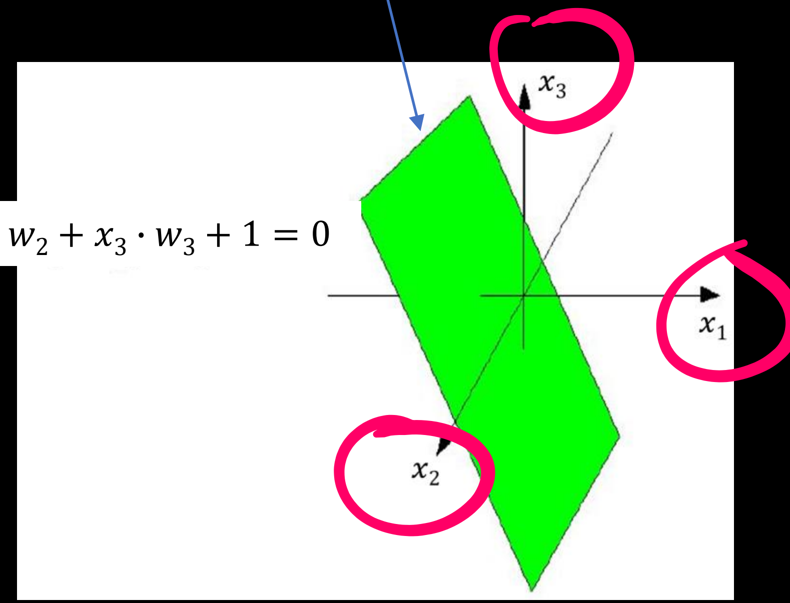


3-Input Neuron



Decision boundary : Plane

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



More than 4 inputs?

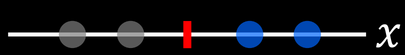
$$x_1w_1 + \; x_2w_2 + x_3w_3 + x_4w_4 + b = 0$$

→ hyperplane

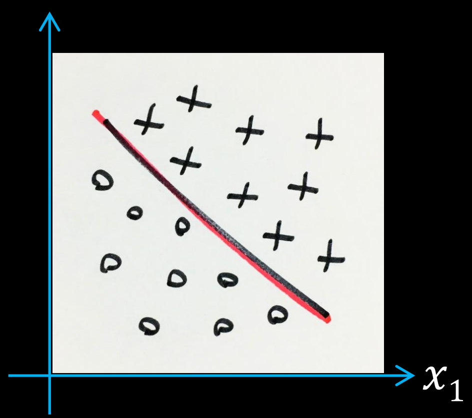
초평면

Linear Decision Boundary

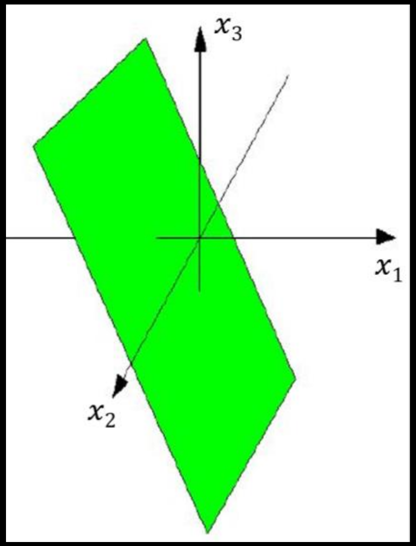
value



line

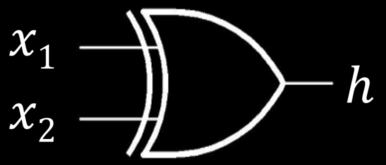


plane



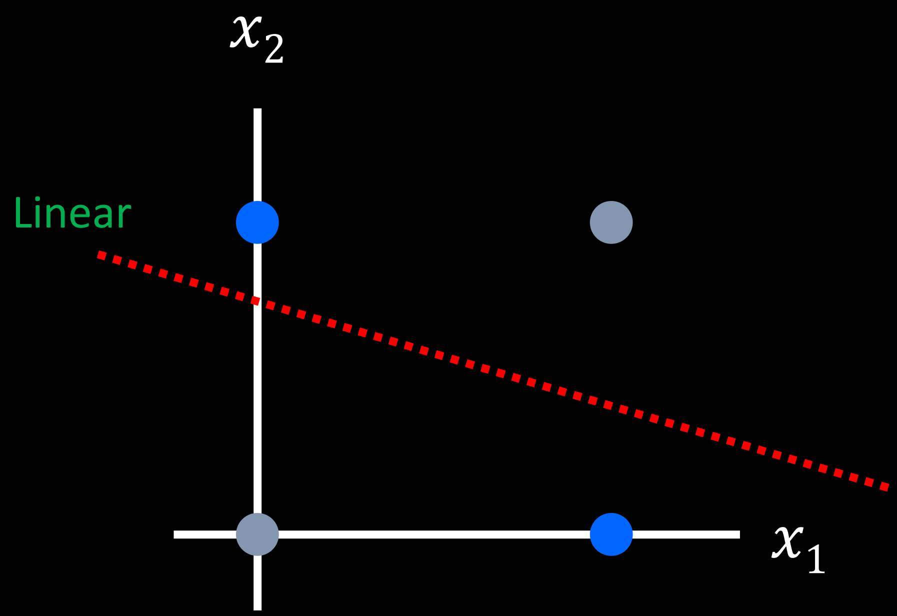
hyperplane

XOR



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

Truth table



View from above

XOR

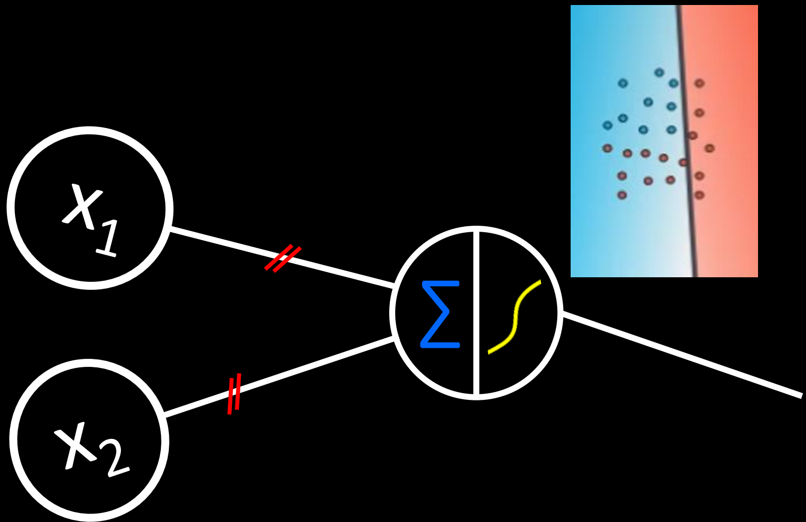
Binary Cross-Entropy Loss

- Number of class : 2 (0 or 1)
- 1-decision boundary for 2-class classification
- However, impossible to classify using a single linear decision boundary

Lab 16.py

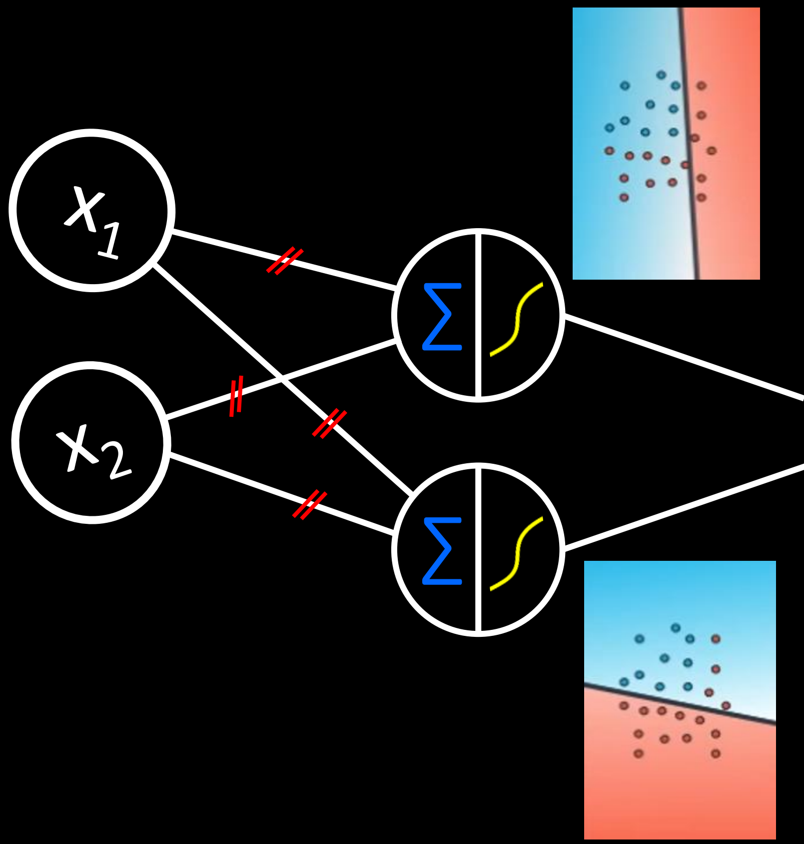
- XOR problem
- A neuron, 1 linear decision boundary
- Cannot be solved!

How to solve



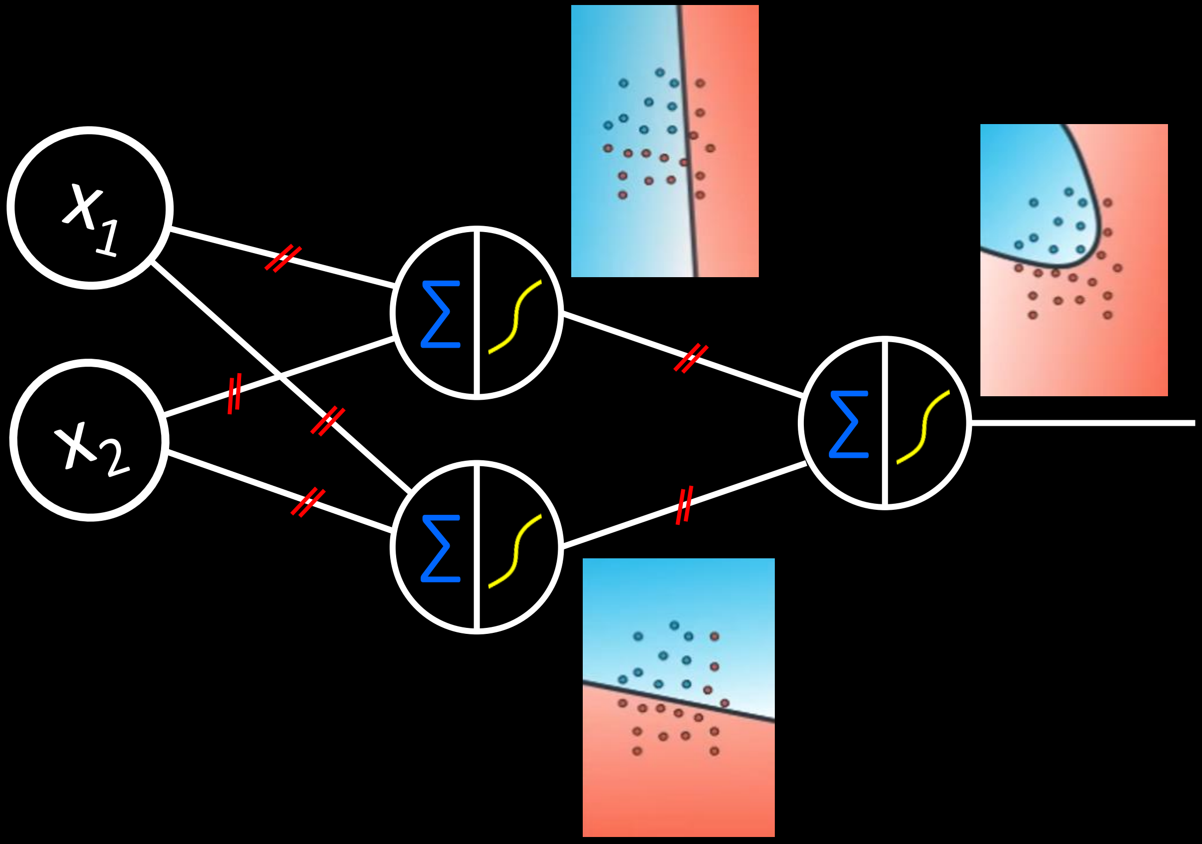
by Luis Serrano, A friendly introduction to Deep Learning, UDACITY

How to solve



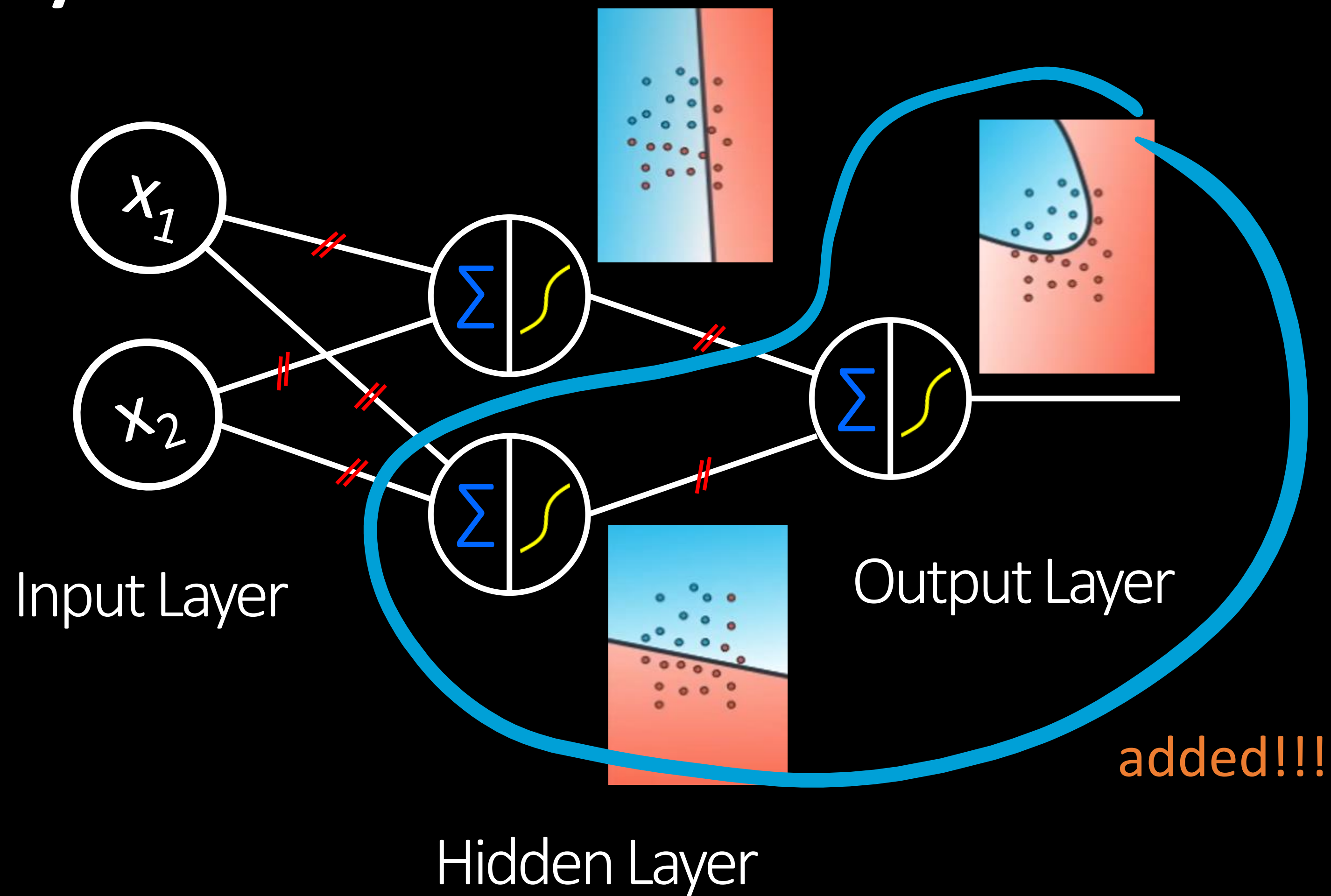
by Luis Serrano, A friendly introduction to Deep Learning, UDACITY

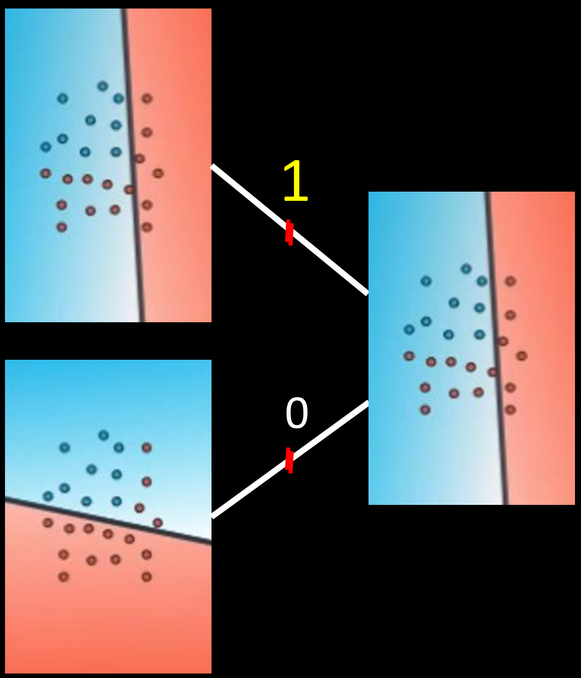
How to solve

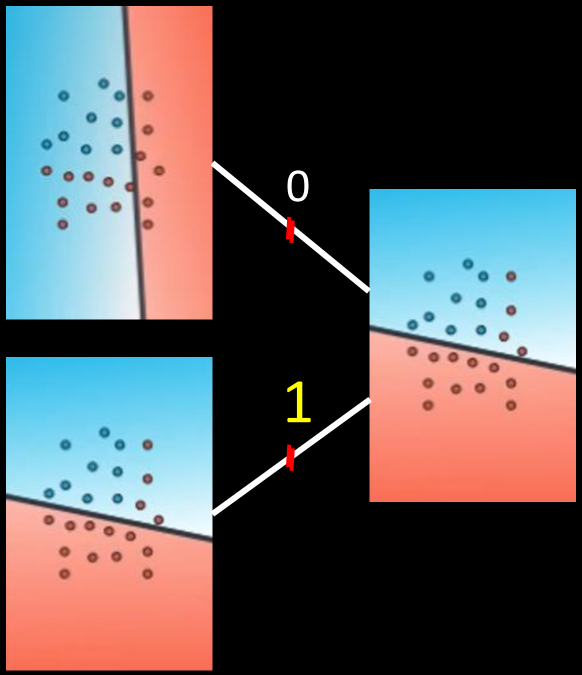


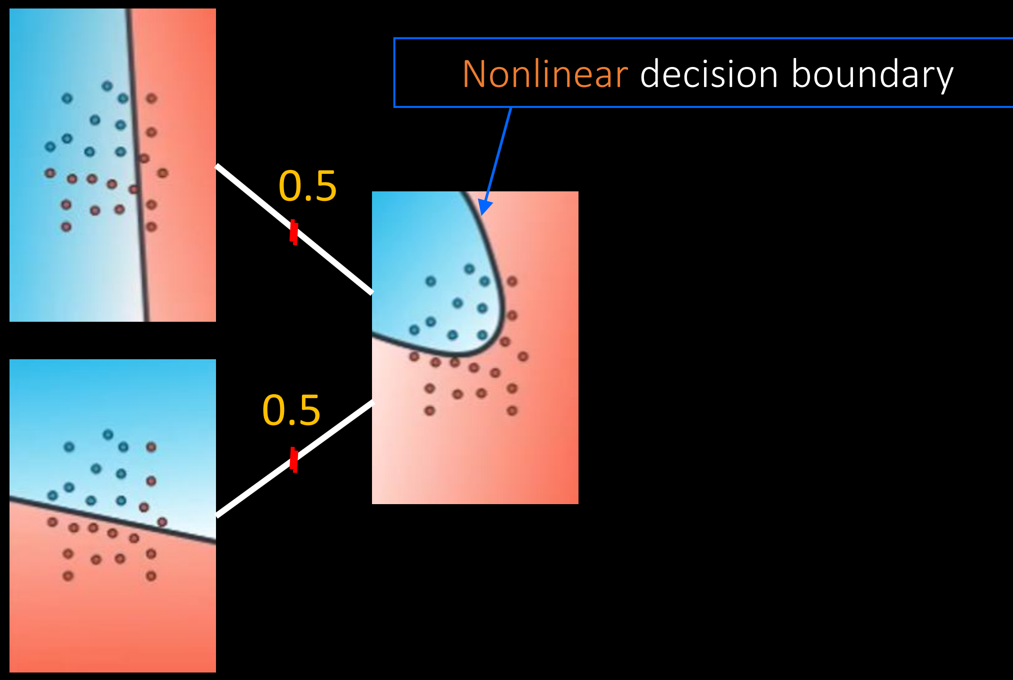
by Luis Serrano, A friendly introduction to Deep Learning, UDACITY

3-layer Neural Network




1
0

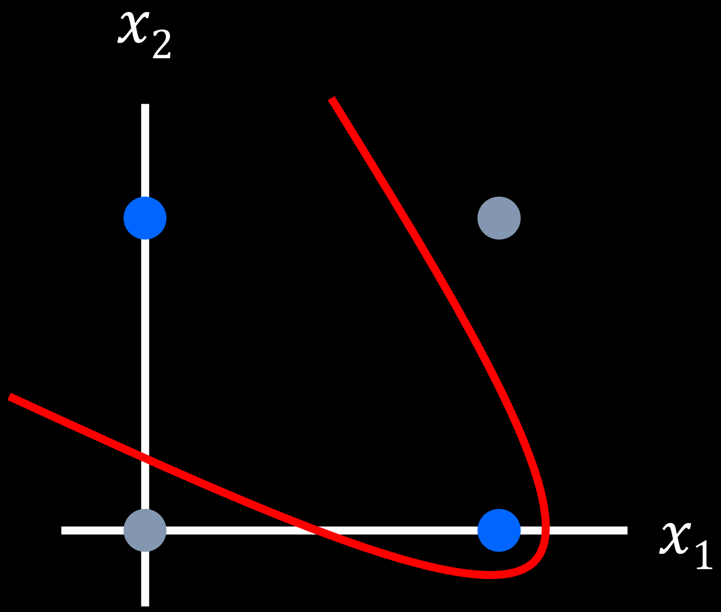

0
1



What is a linear decision boundary?

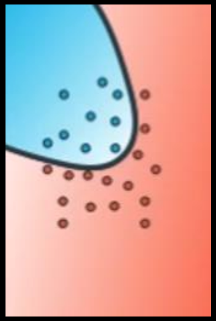
3-layer NN for nonlinear decision boundary

XOR

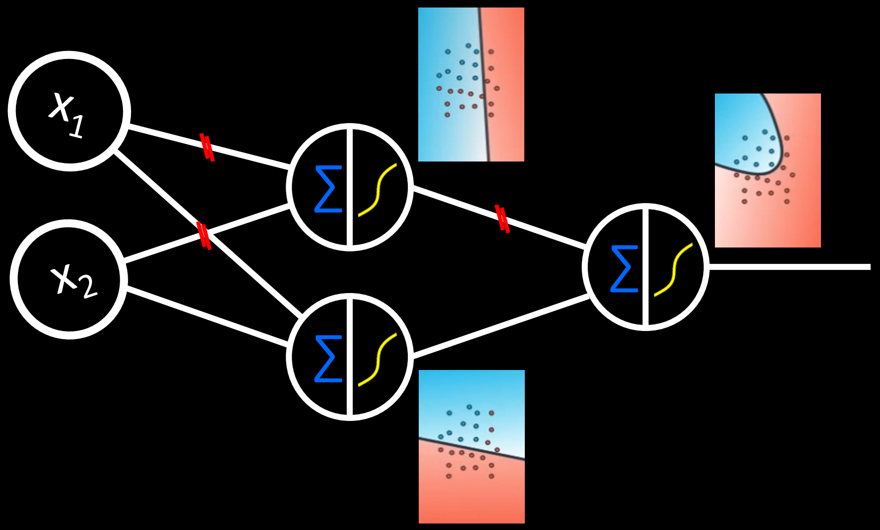


View from above

View from above



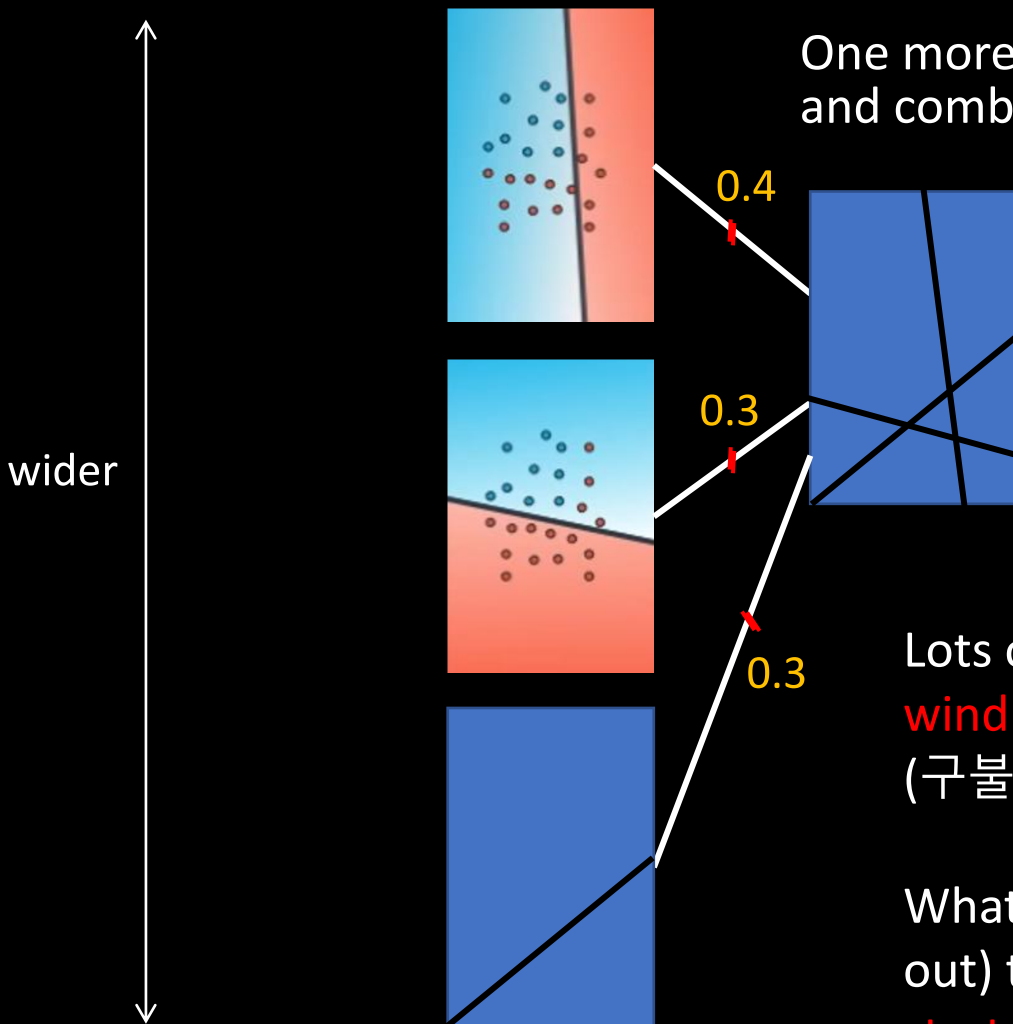
Side view



Lab 17.py

- Solving XOR gate problem using 3-layer neural network
- The way to create non-linear decision boundary

deeper



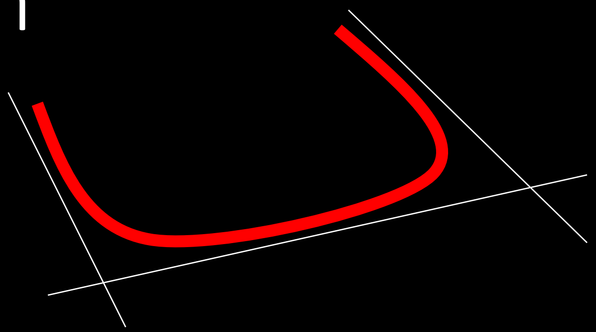
One more linear (going wider) and combination

Lots of connections give us a winding decision boundary (구불 구불한 결정경계).

What happens if we cut(drop-out) the connections? stretching the decision boundary (펴진 결정경계).

Nonlinear Decision Boundary

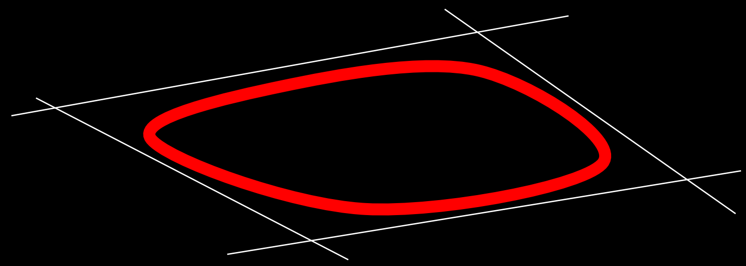
- Combination of three linear decision boundaries



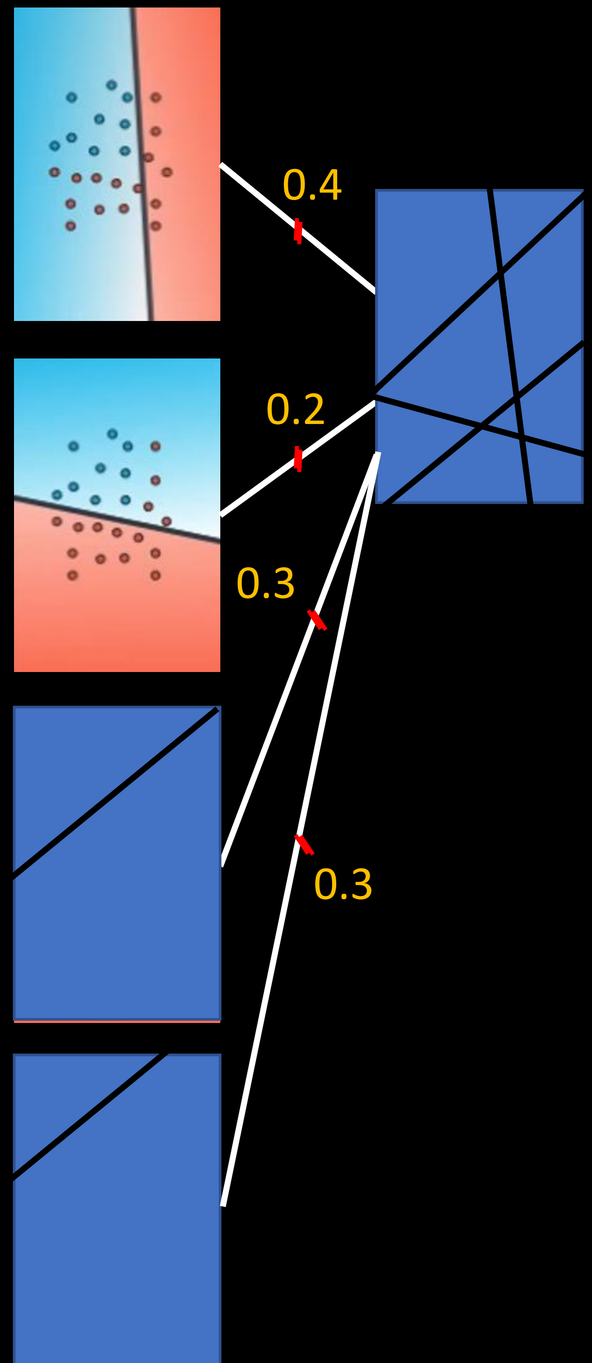

JUPITERIMAGES

Nonlinear Decision Boundary

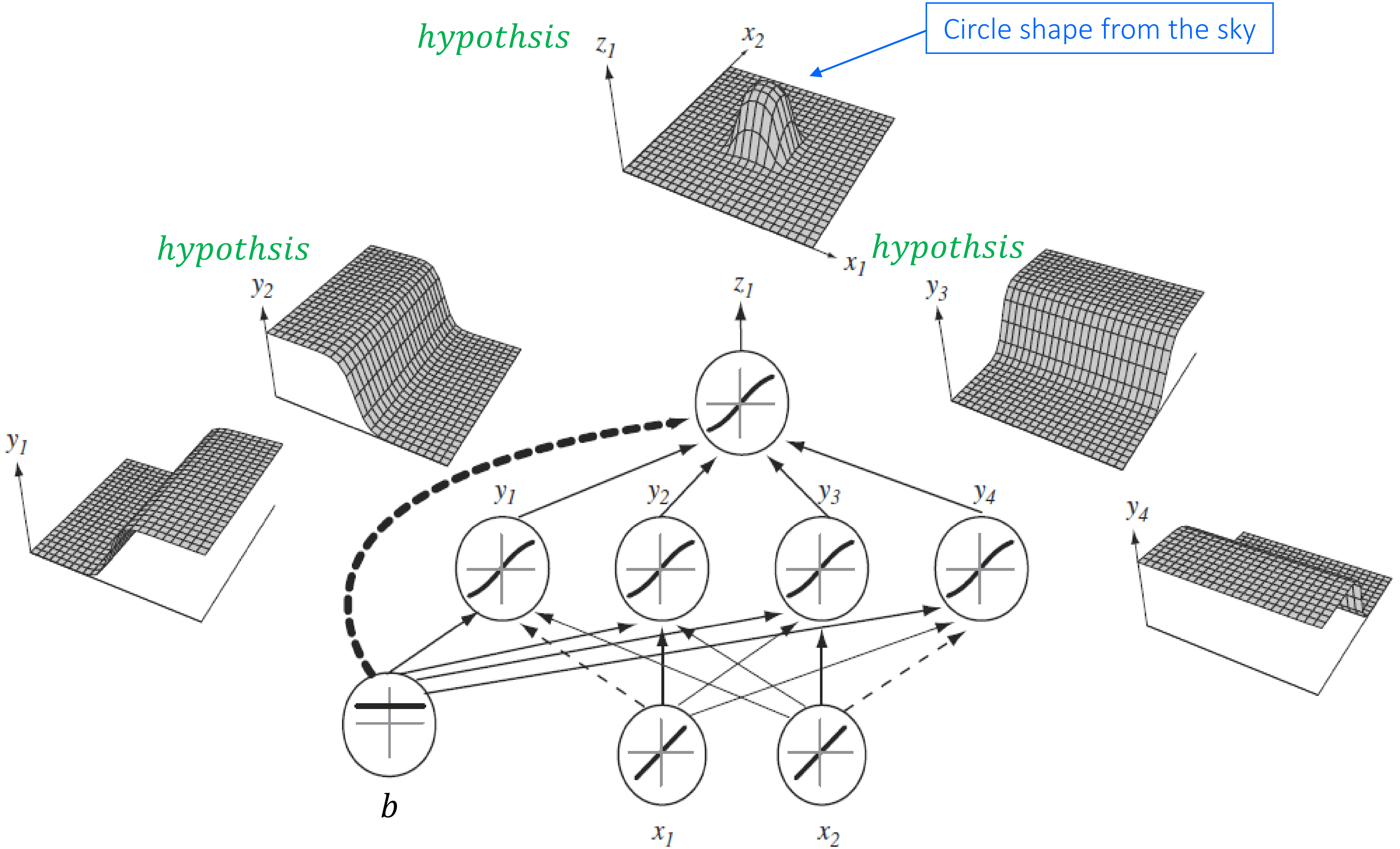
- Merging four linear decision boundaries



View from above


0.4
0.2
0.3
0.3

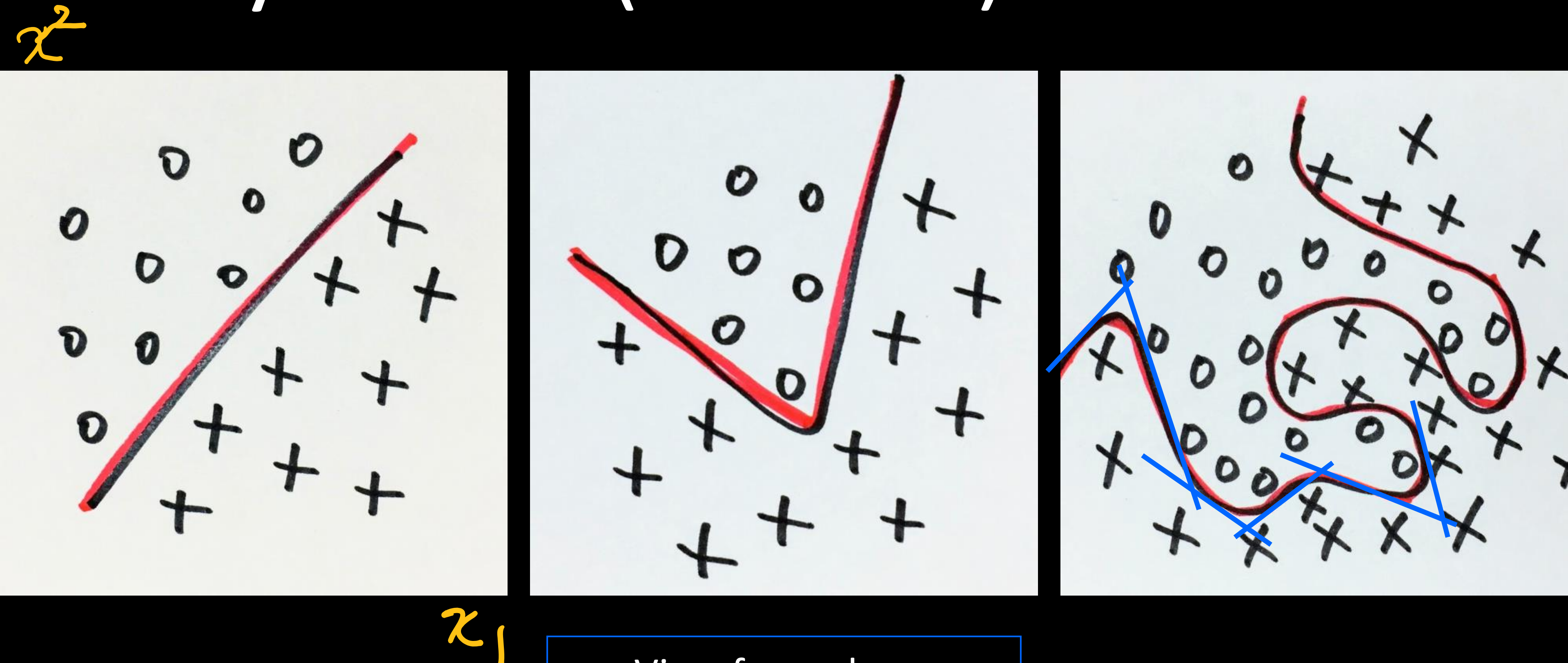
Nonlinear Decision Boundary



View from above

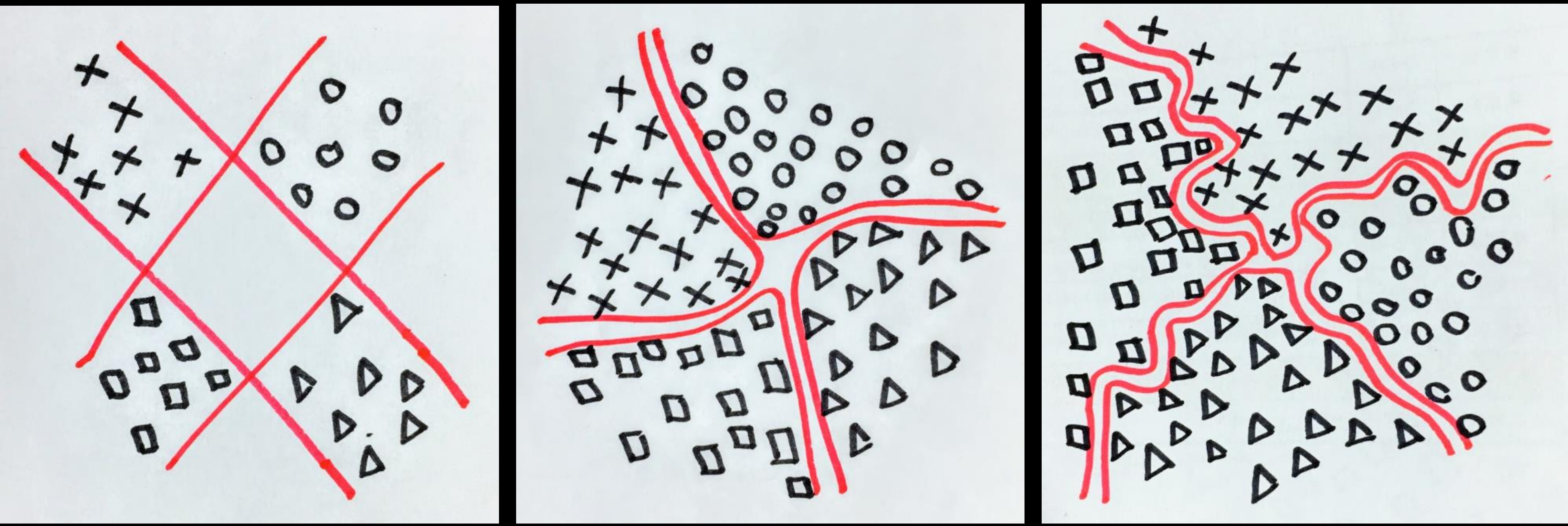
Side view

As you wish (2 classes)



View from above

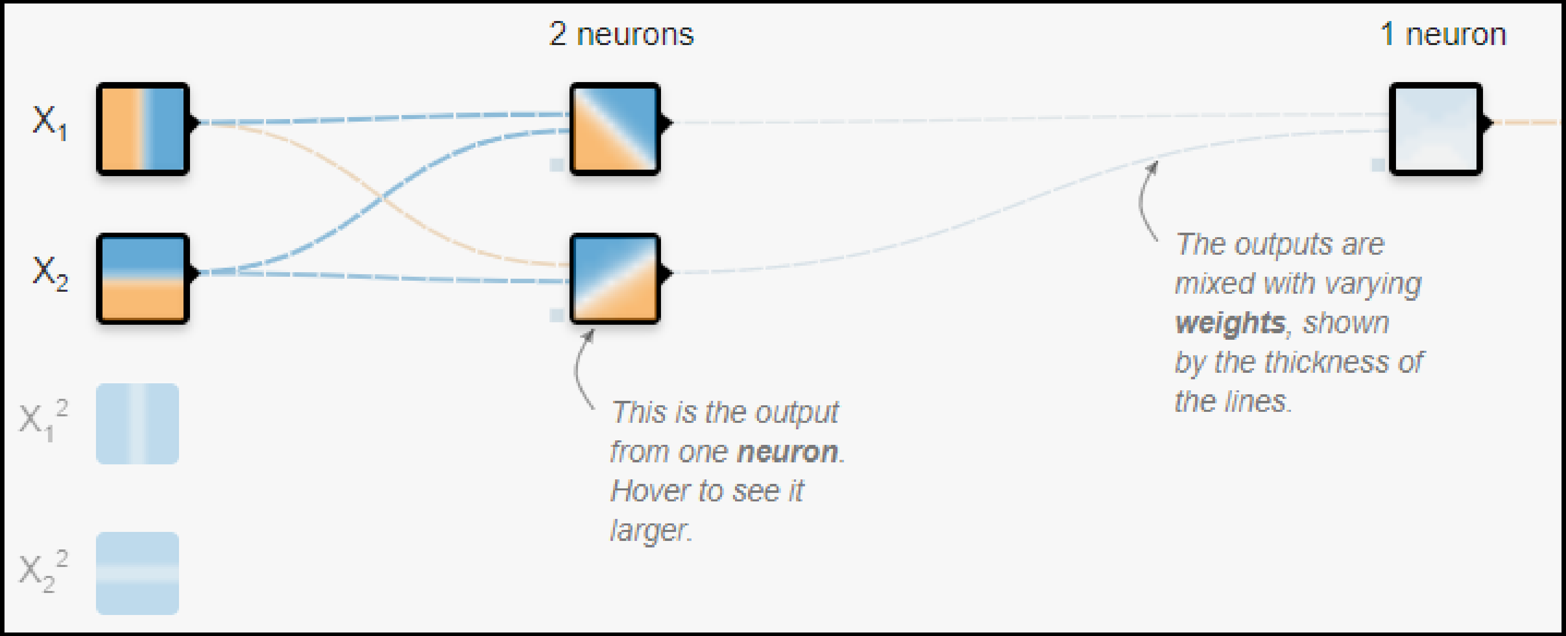
As you wish (4 classes)



From the above

More neurons (wider & deeper),
more complex(detail)
decision boundary

http://playground.tensorflow.org



As you wish

go wider & deeper

- to make more complex nonlinear decision boundaries.
- We can classify anything we imagine.

The way of machine learning

- Learning over and over again just like human being
- If it misrecognizes, just say ‘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)