

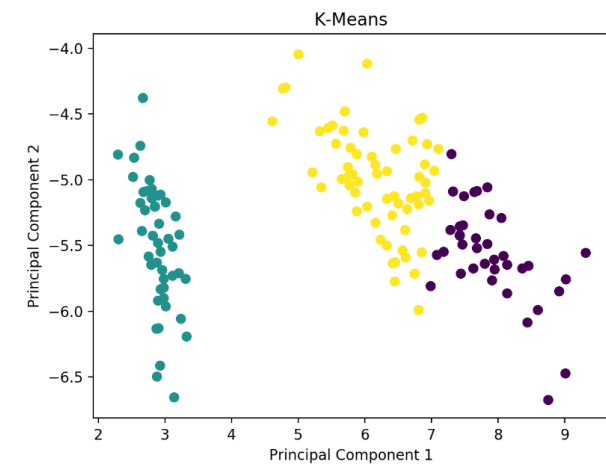
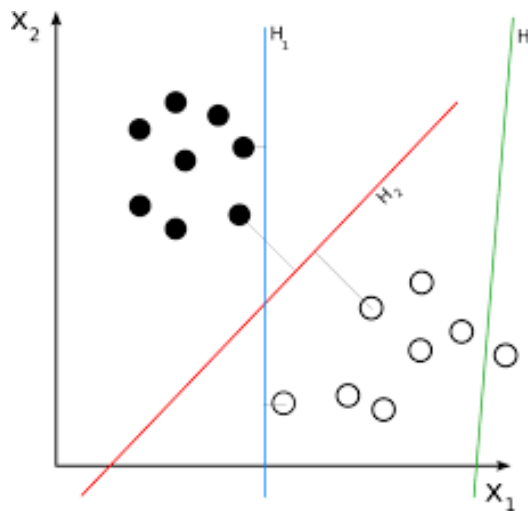
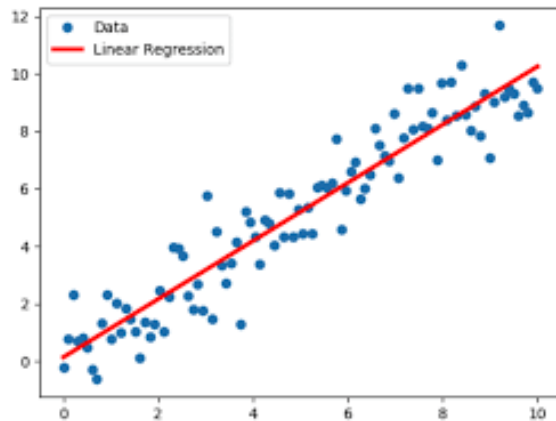
# | 신경망과 딥러닝

# Supervised Learning (지도학습)

- Linear Regression

- Binary Classification

- Multi-label Classification

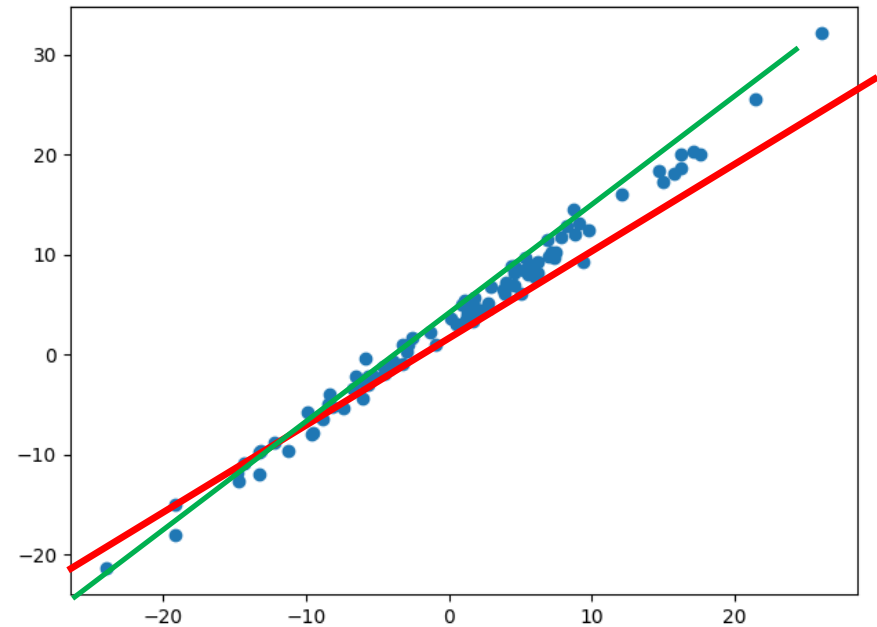


# Linear Regression (선형 회귀분석)

```
import numpy as np
from sklearn.linear_model import LinearRegression
```

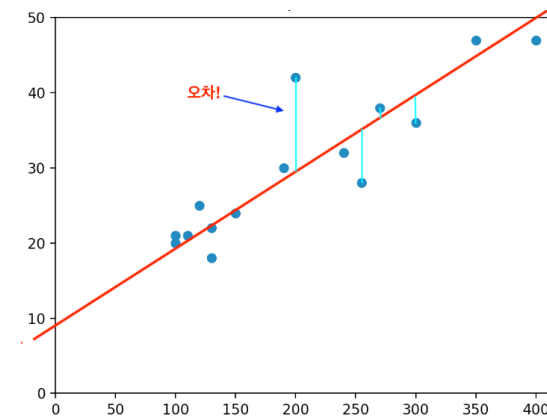
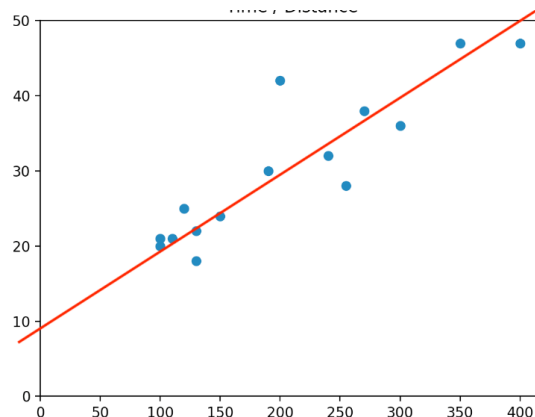
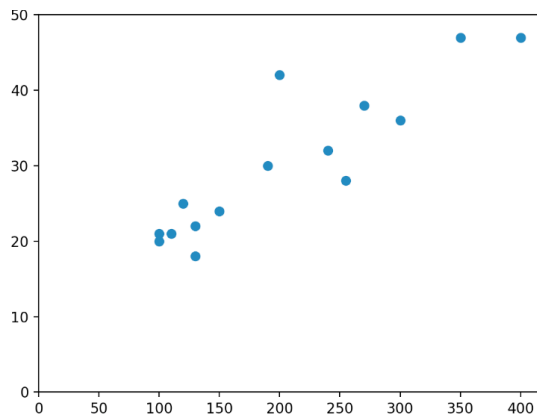
```
X = 10 * np.random.randn(100,1)
y = 3 + X + np.random.randn(100,1)
```

```
plt.plot(X, y, 'o')
plt.show()
```



# Linear Regression (선형 회귀분석)

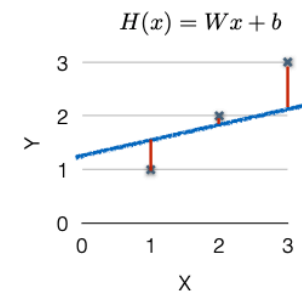
## Best one?



## Cost function (Loss Function)

$$\frac{(H(x^{(1)}) - y^{(1)})^2 + (H(x^{(2)}) - y^{(2)})^2 + (H(x^{(3)}) - y^{(3)})^2}{3}$$

$$cost = \frac{1}{m} \sum_{i=1}^m (H(x^{(i)}) - y^{(i)})^2$$

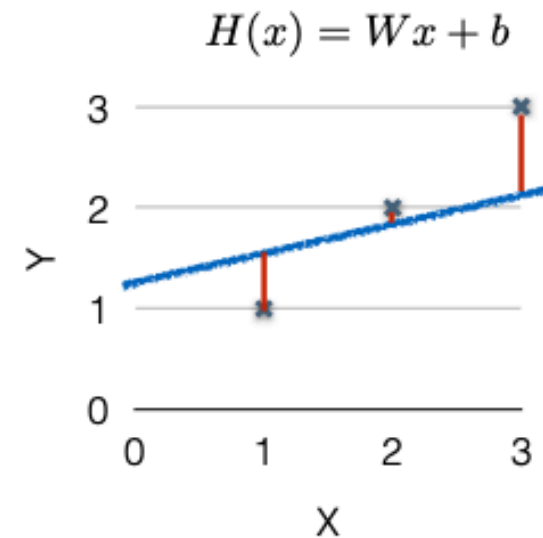


# Linear Regression (선형 회귀분석)

Cost Function → 최소화로....

$$\frac{(H(x^{(1)}) - y^{(1)})^2 + (H(x^{(2)}) - y^{(2)})^2 + (H(x^{(3)}) - y^{(3)})^2}{3}$$

$$cost = \frac{1}{m} \sum_{i=1}^m (H(x^{(i)}) - y^{(i)})^2$$



# Linear Regression (선형 회귀분석)

$$\text{cost}(W) = \frac{1}{m} \sum_{i=1}^m (Wx^{(i)} - y^{(i)})^2$$

x	Y
1	1
2	2
3	3

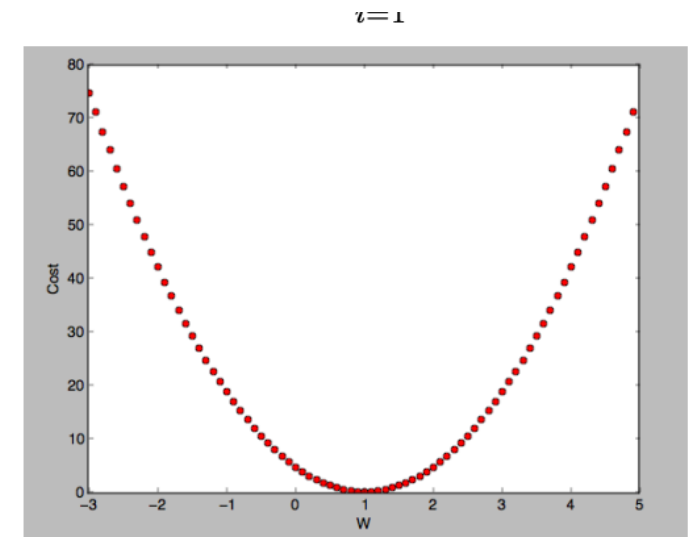
- $W=1, \text{cost}(W)=0$

$$\frac{1}{3}((1 * 1 - 1)^2 + (1 * 2 - 2)^2 + (1 * 3 - 3)^2)$$

- $W=0, \text{cost}(W)=4.67$

$$\frac{1}{3}((0 * 1 - 1)^2 + (0 * 2 - 2)^2 + (0 * 3 - 3)^2)$$

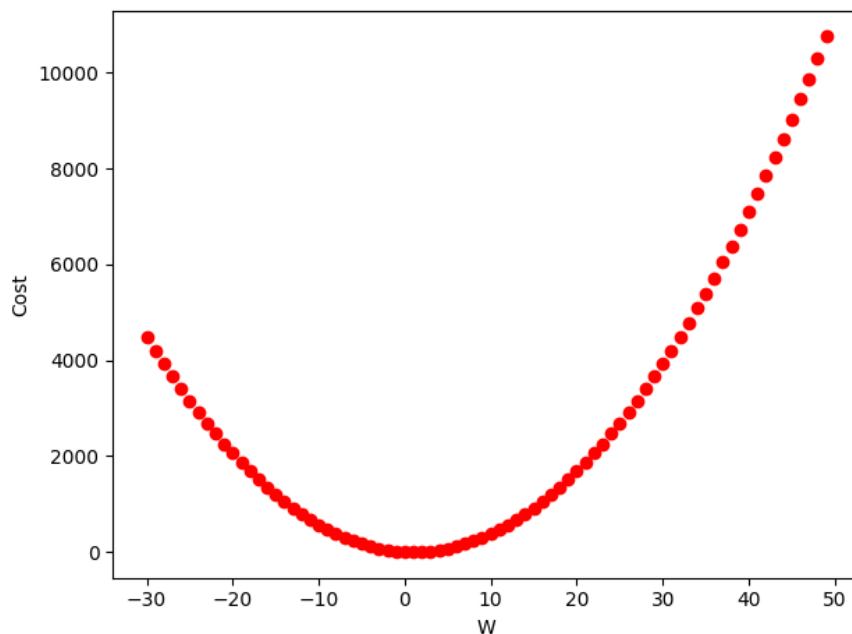
- $W=2, \text{cost}(W)=?$



# Linear Regression (선형 회귀분석)

## 경사 하강법

- cost함수 그래프를 그리면 위와 같은 2차

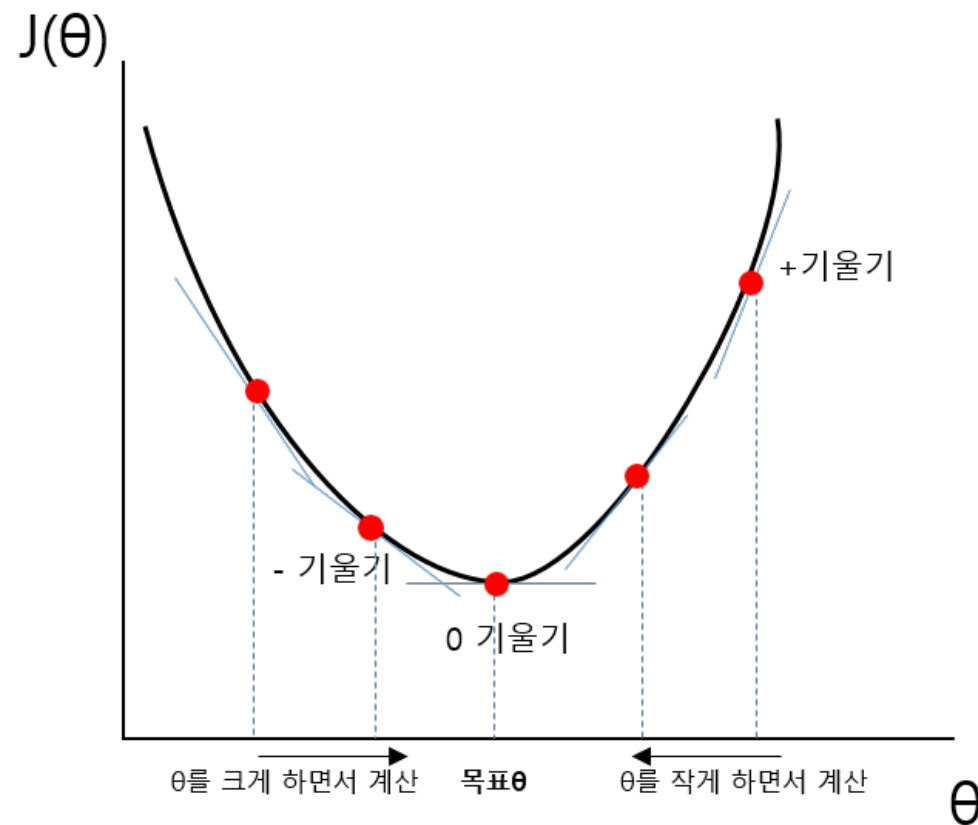


```
import numpy as np
import matplotlib.pyplot as plt
import tensorflow as tf
def forward(wa, x):
    return wa * x
```

```
X = np.array([1., 2., 3.])
Y = np.array([1., 2., 3.])
W_val = []
cost_val = []
m = n_samples = len(X)
for i in range(-30, 50):
    W_val.append(i)
    print(forward(i, X))
    cost_val.append(tf.reduce_sum(tf.pow(forward(i, X)-Y,
2)))/(m))
plt.plot(W_val, cost_val, 'ro')
plt.ylabel('Cost')
plt.xlabel('W')
plt.show()
```

# Linear Regression (선형 회귀분석)

## 경사 하강법





# Linear Regression (선형 회귀분석)

linear regression 문제를 표현하자면, 아래 식에서  $(\beta_0, \beta_1)$ 의 값을 추측

$$Y = f(X) = \beta_0 + \beta_1 X$$

추측값에는  $\hat{y}$

```
import numpy as np
from sklearn.linear_model import LinearRegression

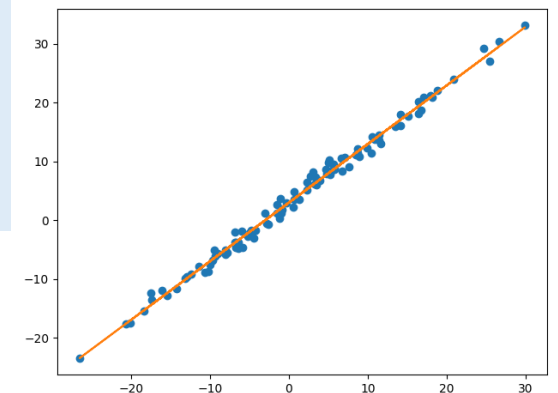
X = 10 * np.random.randn(100,1)
y = 3 + X + np.random.randn(100,1)

line_fitter = LinearRegression() #LinearRegression 모델을 생성
line_fitter.fit(X, y) #그 안에 X, y 데이터를 적용

#기울기: line_fitter.coef_
#절편: line_fitter.intercept_

y_predicted = line_fitter.predict(X) #  $\hat{y}$ 

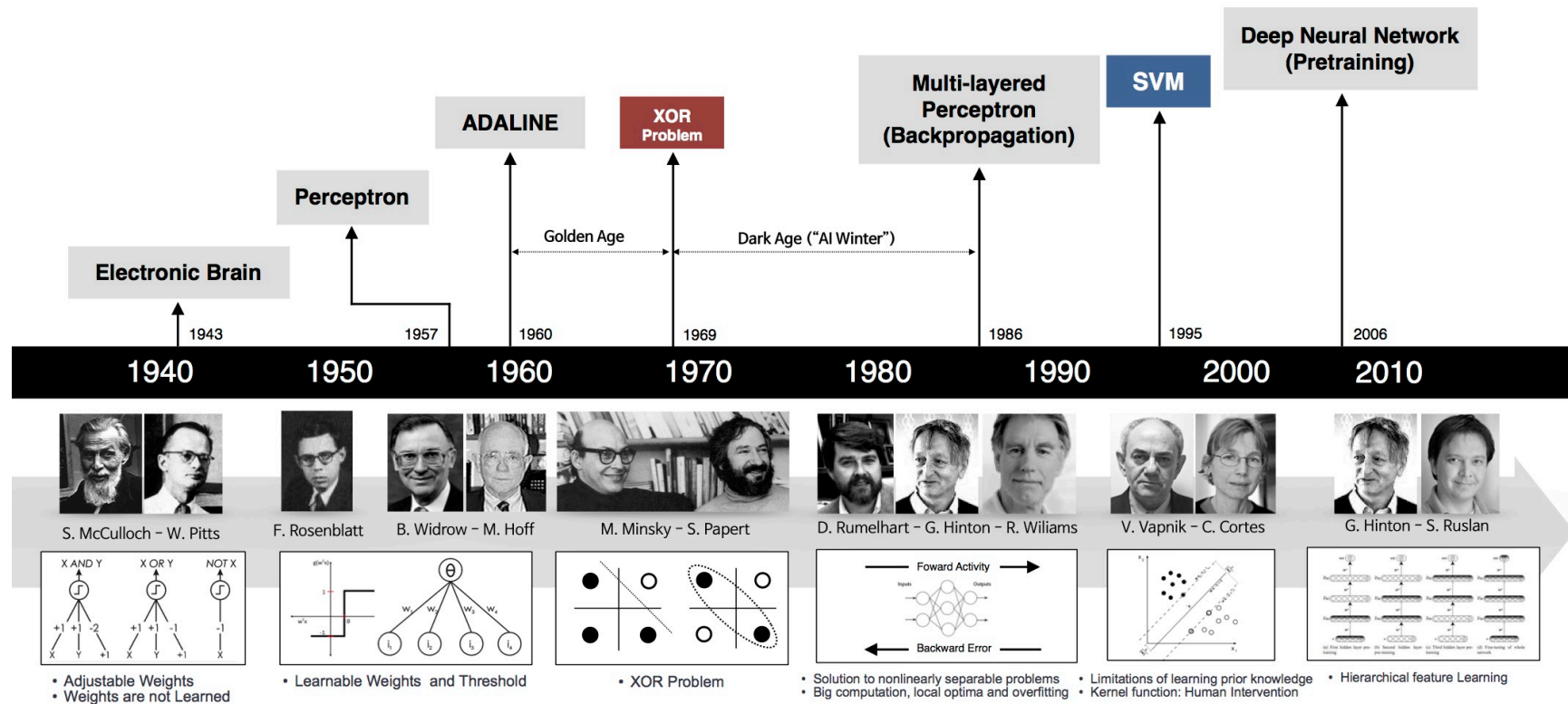
plt.plot(X, y, 'o')
plt.plot(X, y_predicted)
plt.show()
```



# 인공 신경망

## 뇌에 있는 생물학적 뉴런의 네트워크에 영감을 받은 머신 러닝 모델

1943년 신경생리학자 Warren McCulloch와 수학자 Walter Pitts가 '[A Logical Calculus of Ideas Immanent In Nervous Activity](#)' 처음 소개했으며, 명제 논리(propositional logic)를 사용해 동물 뇌의 생물학적 뉴런이 복잡한 계산을 위해 어떻게 상호작용하는지에 대해 간단한 계산 모델을 제시했다.

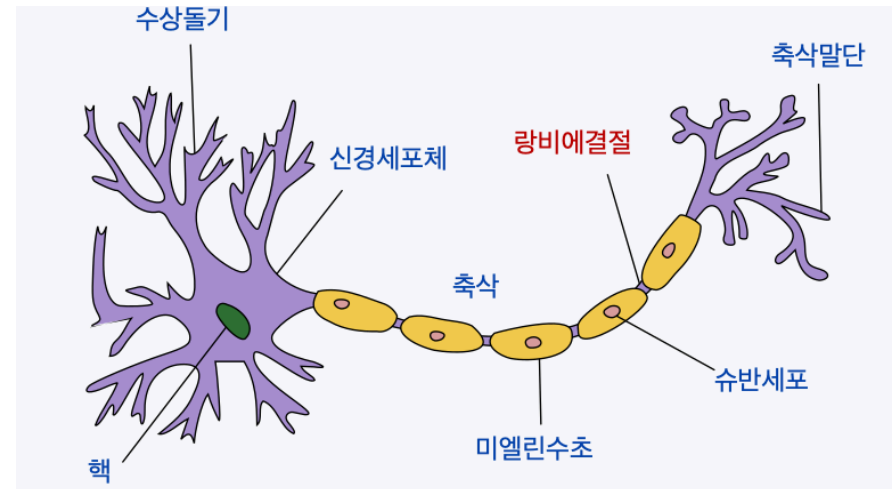
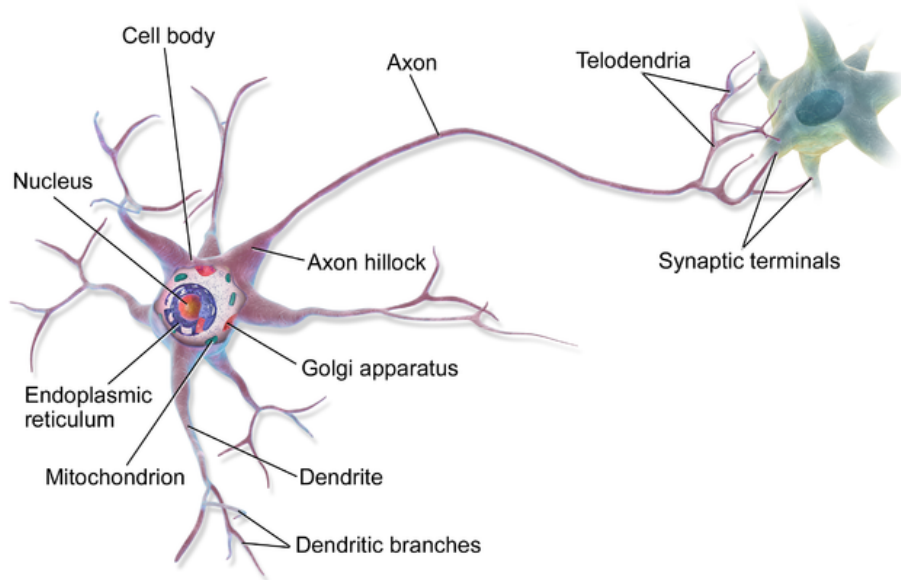


# 생물학적 뉴런

**Dendrite** : 수상돌기, 다른 뉴런으로부터 신호를 수용하는 부분

**Axon** : 축삭돌기, 신호를 내보내는 부분

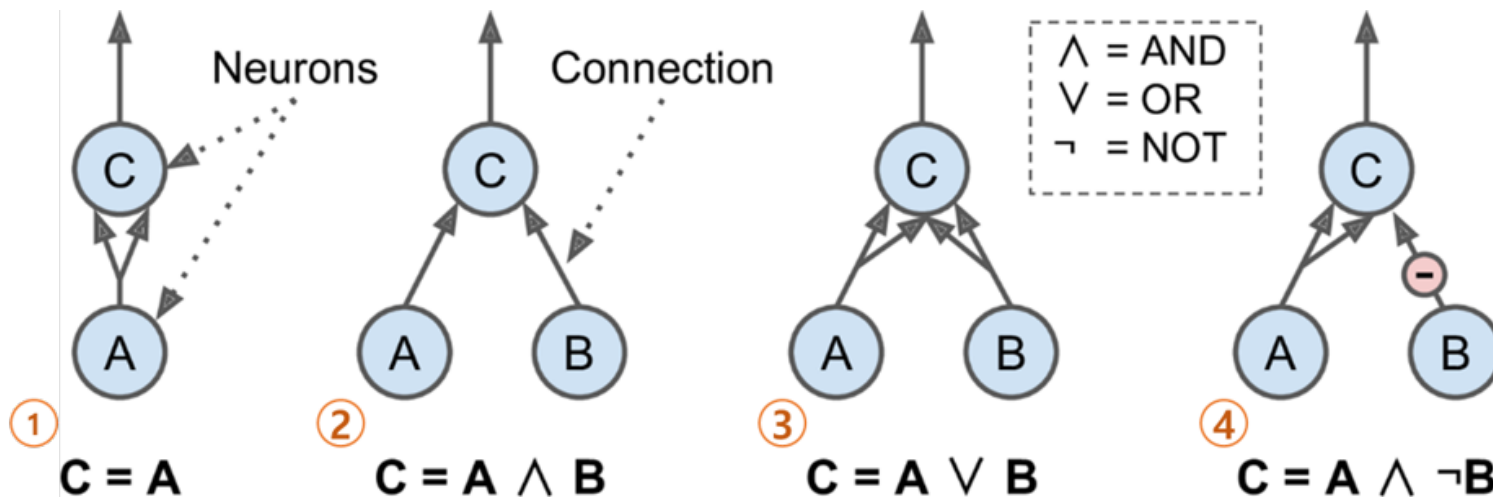
**Synaptic terminals** : 시냅스(synapse) 뉴런의 접합부, 다른 뉴런으로 부터 짧은 전기 자극 **신호**(signal)를 받음



[https://ko.wikipedia.org/wiki/신경\\_세포](https://ko.wikipedia.org/wiki/신경_세포)

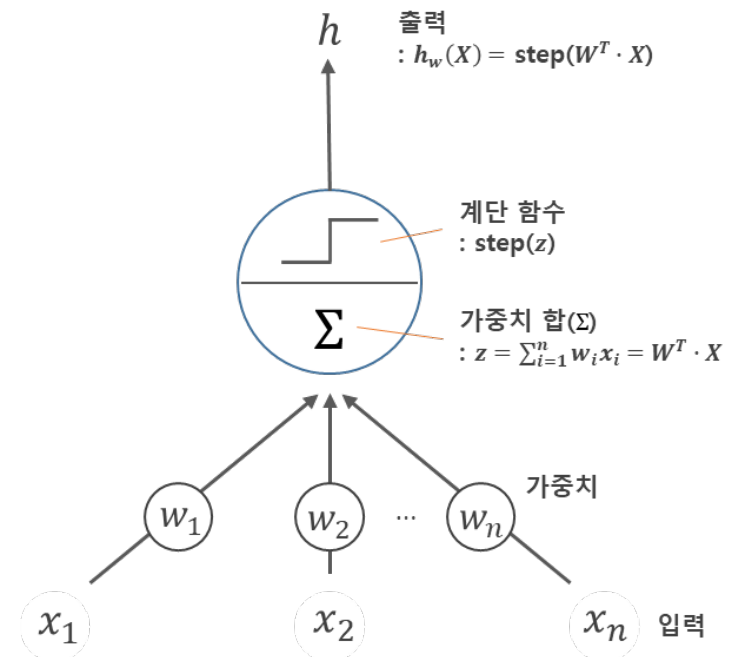
# 뉴런을 사용한 논리 연산

- ①은 항등함수를 의미하며, 뉴런 A가 활성화 되면 뉴런 C 또한 활성화된다.
- ②는 논리곱 연산을 의미하며, 뉴런 A와 B가 모두 활성화될 때만 뉴런 C가 활성화된다.
- ③은 논리합 연산을 의미하며, 뉴런 A와 B 둘 중 하나라도 활성화되면 C가 활성화된다.
- ④는 어떤 입력이 뉴런의 활성화를 억제할 수 있다고 가정하면, 위의 그림에서 처럼 뉴런 A가 활성화 되고 B가 비활성화 될 때 뉴런 C가 활성화된다.



# 퍼셉트론

- 1957년에 프랑크 로젠블라트가 고안한 신경망의 기원이 되는 알고리즘
- 신경망(딥러닝)의 기원이 되는 알고리즘
- 퍼셉트론(인공뉴런): 다수의 신호를 입력으로 받아 하나의 신호를 출력

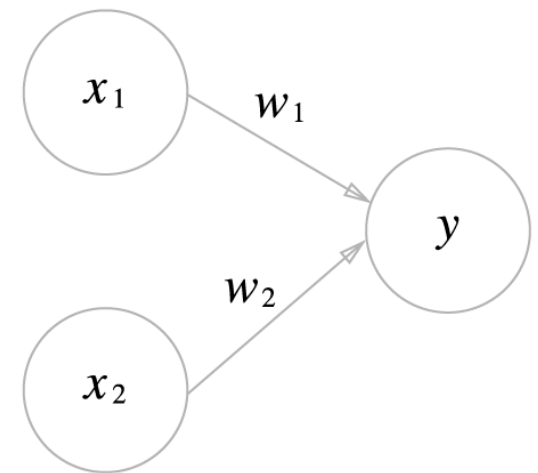


# | 신경망과 딥러닝2

# 퍼셉트론

## ■ 신호: 흐른다(1), 흐르지 않는다(0) 두 가지 값을 가짐.

- 입력신호:  $x_1, x_2$
- 출력신호:  $y$
- 가중치:  $w_1, w_2$  ( $w$ 는 weight의 머리글자)
- 퍼셉트론에서는 가중치가 클수록 강한 신호를 흘림.
- 전류에서 말하는 저항. 저항이 낮을 수록 큰 전류가 흐름.
- 뉴런, 노드:  $x_1, x_2, y$
- 입력신호가 뉴런에 보내질 때 각각 고유한 가중치가 곱해짐. ( $w_1x_1, w_2x_2$ )



## ■ 뉴런이 활성화: 뉴런에서 보내온 신호의 총합이 정해진 한계를 넘어설 때만 1을 출력

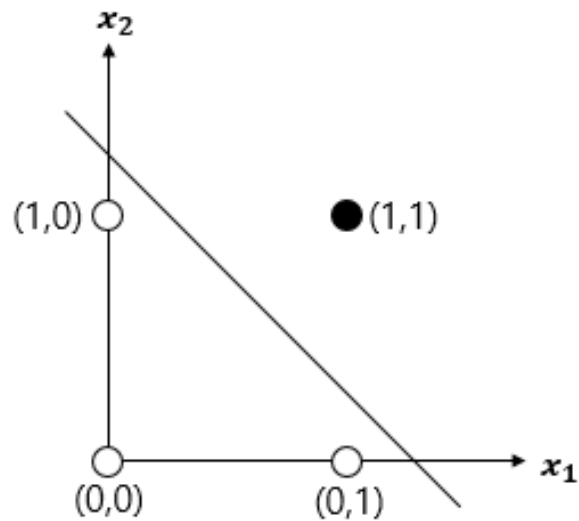
## ■ 임계값: 정해진 한계. 세타 기호로 표현

$$y = 1(w_1x_1 + w_2x_2 > \theta)$$

$$y = 0(w_1x_1 + w_2x_2 \leq \theta)$$

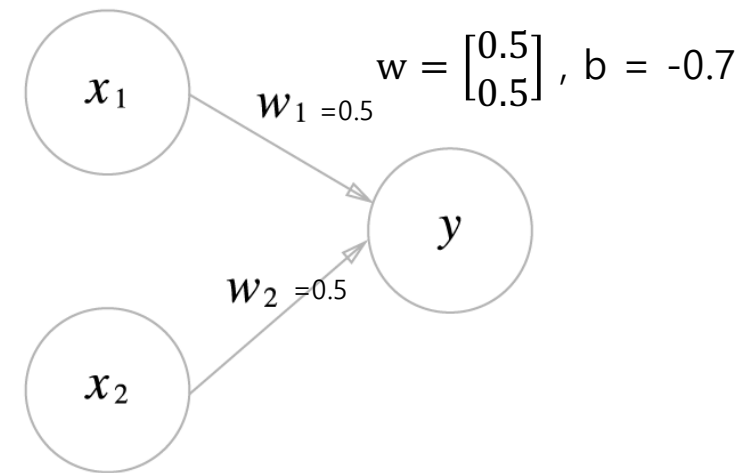
# 단순한 논리 회로

## AND 게이트



$x_1$	$x_2$	$y$
0	0	0
0	1	0
1	0	0
1	1	1

Numpy로 표현



$x_1$	$x_2$	$x_1*w_1 + x_2*w_2 + b_1$		$x_3$
0	0	$0 * 0.5 + 0 * 0.5 - 0.7 = -0.7$	$< 0$	0
1	0	$1 * 0.5 + 0 * 0.5 - 0.7 = -0.2$	$< 0$	0
0	1	$0 * 0.5 + 1 * 0.5 - 0.7 = -0.2$	$< 0$	0
1	1	$1 * 0.5 + 1 * 0.5 - 0.7 = 1.7$	$> 0$	1



# 단순한 논리 회로

## AND 게이트

```
def AND(x1, x2):  
    w1, w2, theta = 0.5, 0.5, -0.7  
    tmp = x1*w1 + x2*w2  
    if tmp <= theta:  
        return 0  
    elif tmp > theta:  
        return 1
```

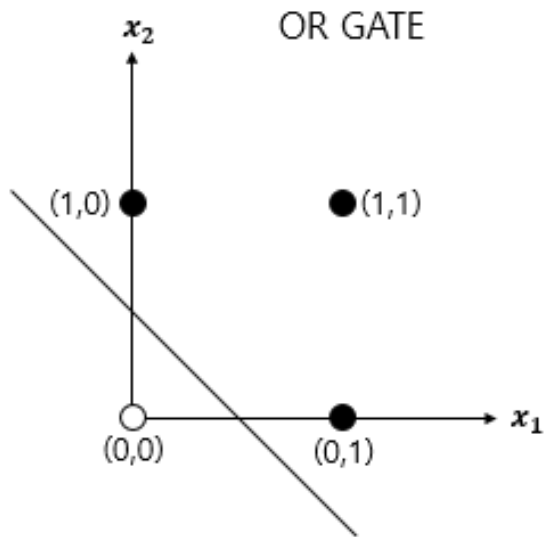
```
print(AND(0, 0)) # 0을 출력  
print(AND(1, 0)) # 0을 출력  
print(AND(0, 1)) # 0을 출력  
print(AND(1, 1)) # 1을 출력
```

```
def AND(x1, x2):  
    x = np.array([x1, x2])  
    w = np.array([0.5, 0.5])  
    b = -0.7  
    tmp = np.sum(w*x) + b  
    if tmp <= 0:  
        return 0  
    else:  
        return 1
```

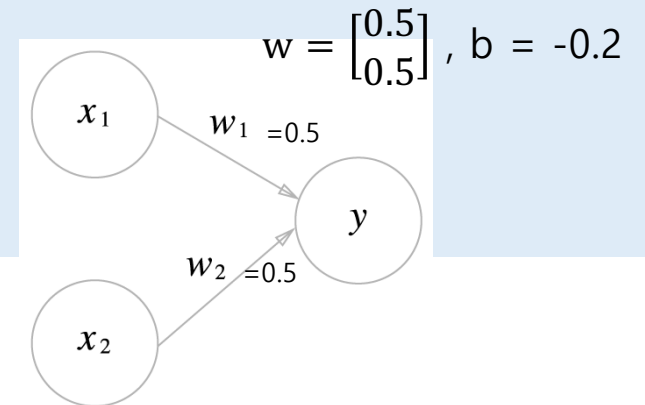
# 단순한 논리 회로

## OR 게이트

x1	x2	x3
0	0	0
1	0	1
0	1	1
1	1	1



```
def OR(x1, x2):  
    x = np.array([x1, x2])  
    w = np.array([0.5, 0.5]) # AND와는 가중치(w와 b)만 다르다.  
    b = -0.2  
    tmp = np.sum(w * x) + b  
    if tmp <= 0:  
        return 0  
    else:  
        return 1
```

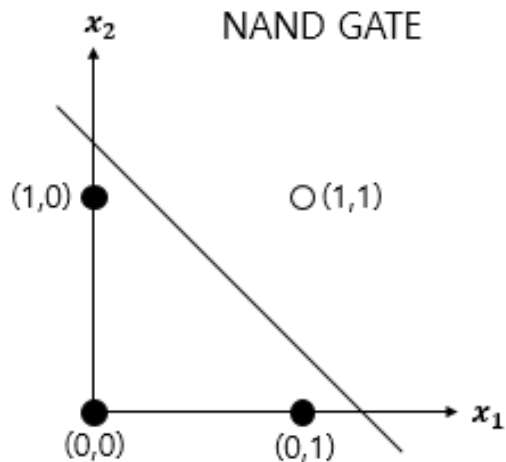


x1	x2	$x1*w1 + x2*w2+b1$		X3
0	0			
1	0			
0	1			
1	1			

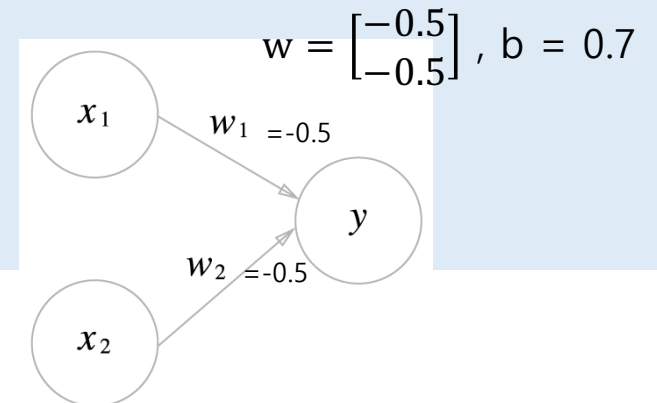
# 단순한 논리 회로

## NAND 게이트

x1	x2	x3
0	0	1
1	0	1
0	1	1
1	1	0



```
def NAND(x1, x2):  
    x = np.array([x1, x2])  
    w = np.array([-0.5, -0.5]) # AND와는 가중치(w와 b)만 다르다.  
    b = 0.7  
    tmp = np.sum(w * x) + b  
    if tmp <= 0:  
        return 0  
    else:  
        return 1
```



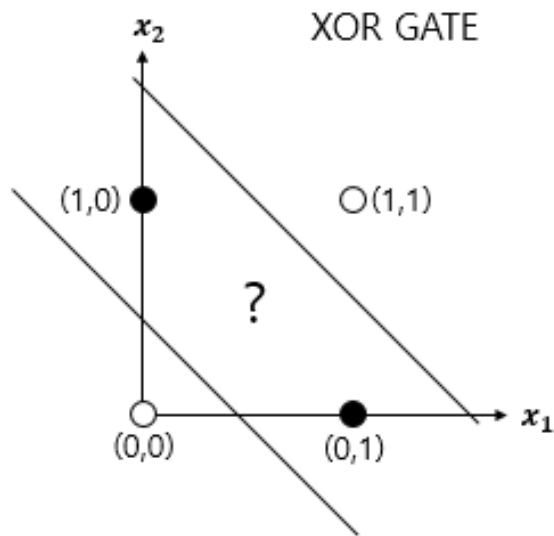
x1	x2	$x1*w1 + x2*w2 + b1$		X3
0	0			
1	0			
0	1			
1	1			

# 퍼셉트론의 한계

## 도전! XOR 게이트

XOR 게이트 배타적 논리합.  $x_1$ 과  $x_2$  한쪽이 1일 때만 1을 출력.

XOR 게이트를 직선 하나로 0과 1을 나누는 영역을 만들 수 있을까?



$x_1$	$x_2$	$y$
0	0	0
0	1	1
1	0	1
1	1	0

# 퍼셉트론의 한계

```
import numpy as np
import matplotlib.pyplot as plt
import matplotlib
```

```
x1 = np.arange(-1, 3, 0.1)
x2 = -x1 + 0.5
```

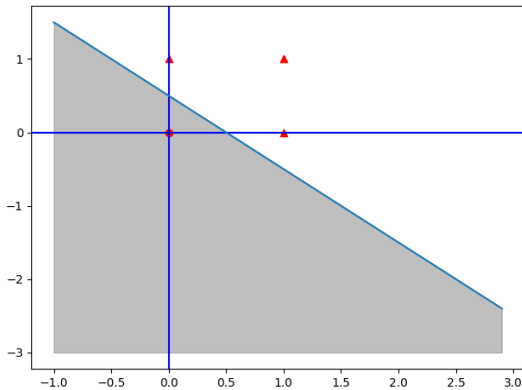
```
plt.axvline(x=0, color = 'b') # draw x = 0 axes
plt.axhline(y=0, color = 'b') # draw y = 0 axes
```

*# 그래프 그리기*

```
plt.plot(x1, x2, label="or")
plt.xlabel("X1") # x축 이름
plt.ylabel("X2") # y축 이름
plt.legend()
```

```
plt.fill_between(x1, x2, -3, color='grey', alpha='0.5')
```

```
plt.scatter([0],[0],marker='o',color='r')
plt.scatter([1,0,1],[0,1,1],marker='^',color='r')
plt.show()
```

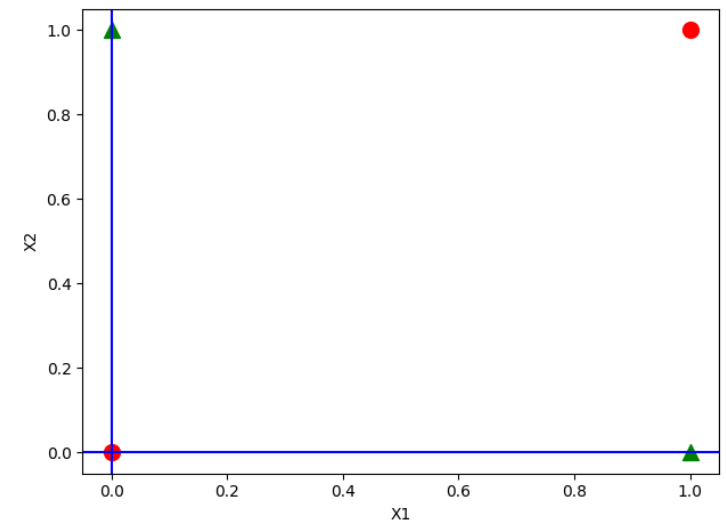


# 퍼셉트론의 한계

```
x1 = np.arange(-1, 3, 0.1)
x2 = -x1 + 0.5

plt.axvline(x=0, color = 'b') # draw x = 0 axes
plt.axhline(y=0, color = 'b') # draw y = 0 axes
plt.xlabel("X1") # x축 이름
plt.ylabel("X2") # y축 이름

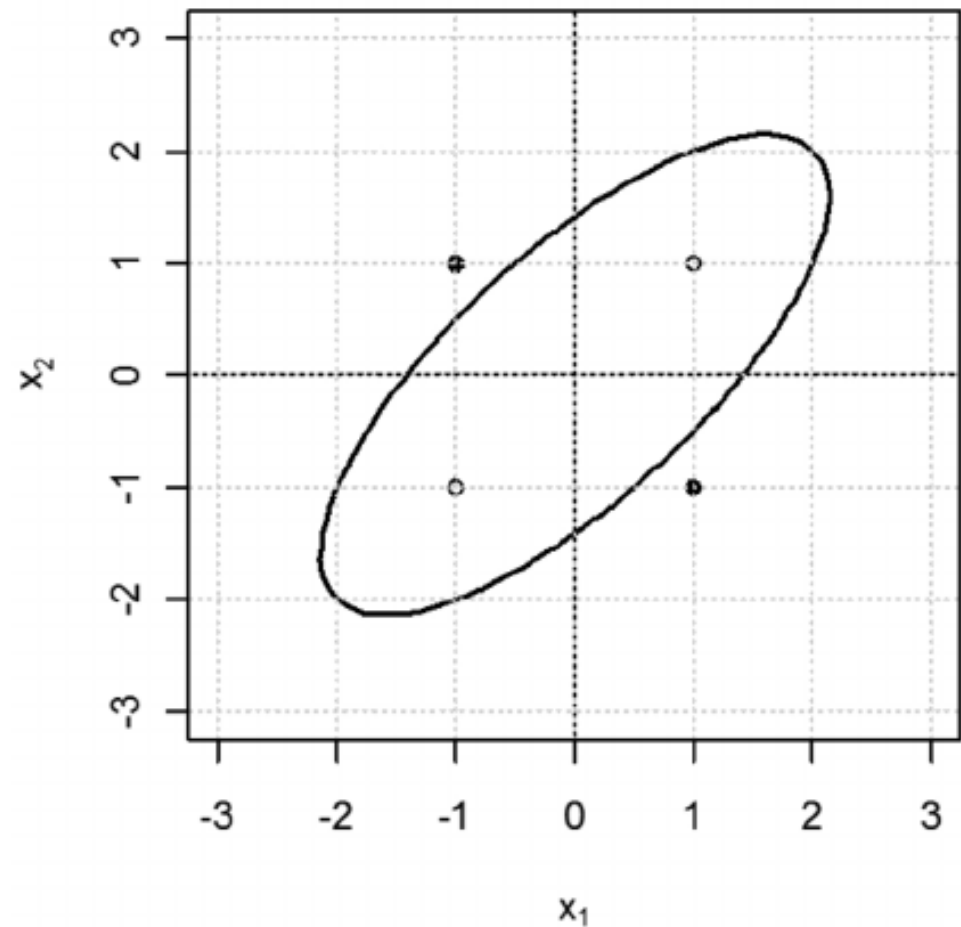
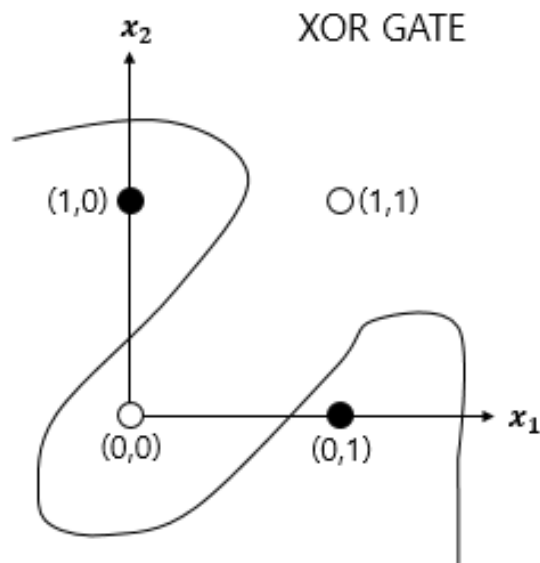
plt.scatter([0,1],[0,1],marker='o',color='r',s=100)
plt.scatter([1,0],[0,1],marker='^',color='g',s=100)
plt.show()
```



# 퍼셉트론의 한계

## 선형과 비선형

- 위와 같은 곡선의 영역을 비선형 영역.
- 직선의 영역을 선형 영역이라함.



# 퍼셉트론의 한계

## 다층 퍼셉트론이 출동




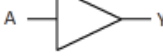




- 단층 퍼셉트론(single-layer perceptron)으로는 XOR 게이트를 표현할 수 없음
- 비선형 영역을 분리할 수 없음
- 다중 퍼셉트론(multi-layer perceptron):
  - 퍼셉트론의 층을 쌓음
  - XOR을 층을 하나 더 쌓아서 표현



# 퍼셉트론의 한계

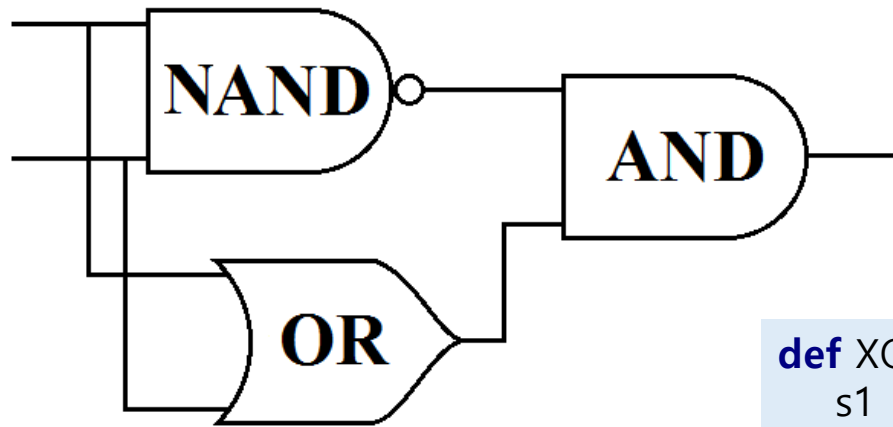
## 기존 게이트 조합

- XOR게이트를 만드는 방법은 다양.
- AND, NAND, OR 게이트를 조합하는 방법

게이트	기호	의미	진리표	논리식															
AND		입력신호가 모두 1일 때 1출력	<table><tr><th>A</th><th>B</th><th>Y</th></tr><tr><td>0</td><td>0</td><td>0</td></tr><tr><td>0</td><td>1</td><td>0</td></tr><tr><td>1</td><td>0</td><td>0</td></tr><tr><td>1</td><td>1</td><td>1</td></tr></table>	A	B	Y	0	0	0	0	1	0	1	0	0	1	1	1	$Y = A \cdot B$ $Y = AB$
A	B	Y																	
0	0	0																	
0	1	0																	
1	0	0																	
1	1	1																	
OR		입력신호 중 1개만 1이어도 1출력	<table><tr><th>A</th><th>B</th><th>Y</th></tr><tr><td>0</td><td>0</td><td>0</td></tr><tr><td>0</td><td>1</td><td>1</td></tr><tr><td>1</td><td>0</td><td>1</td></tr><tr><td>1</td><td>1</td><td>1</td></tr></table>	A	B	Y	0	0	0	0	1	1	1	0	1	1	1	1	$Y = A + B$
A	B	Y																	
0	0	0																	
0	1	1																	
1	0	1																	
1	1	1																	
NOT		입력된 정보를 반대로 변환하여 출력	<table><tr><th>A</th><th>Y</th></tr><tr><td>0</td><td>1</td></tr><tr><td>1</td><td>0</td></tr></table>	A	Y	0	1	1	0	$Y = A'$ $Y = \overline{A}$									
A	Y																		
0	1																		
1	0																		
BUFFER		입력된 정보를 그대로 출력	<table><tr><th>A</th><th>Y</th></tr><tr><td>0</td><td>0</td></tr><tr><td>1</td><td>1</td></tr></table>	A	Y	0	0	1	1	$Y = A$									
A	Y																		
0	0																		
1	1																		
NAND		NOT + AND, 즉 AND의 부정	<table><tr><th>A</th><th>B</th><th>Y</th></tr><tr><td>0</td><td>0</td><td>1</td></tr><tr><td>0</td><td>1</td><td>1</td></tr><tr><td>1</td><td>0</td><td>1</td></tr><tr><td>1</td><td>1</td><td>0</td></tr></table>	A	B	Y	0	0	1	0	1	1	1	0	1	1	1	0	$Y = \overline{A \cdot B}$ $Y = \overline{AB}$
A	B	Y																	
0	0	1																	
0	1	1																	
1	0	1																	
1	1	0																	
NOR		NOT + OR, 즉 OR의 부정	<table><tr><th>A</th><th>B</th><th>Y</th></tr><tr><td>0</td><td>0</td><td>1</td></tr><tr><td>0</td><td>1</td><td>0</td></tr><tr><td>1</td><td>0</td><td>0</td></tr><tr><td>1</td><td>1</td><td>0</td></tr></table>	A	B	Y	0	0	1	0	1	0	1	0	0	1	1	0	$Y = \overline{A + B}$
A	B	Y																	
0	0	1																	
0	1	0																	
1	0	0																	
1	1	0																	
XOR		입력신호가 모두 같으면 0, 한 개라도 틀리면 1출력	<table><tr><th>A</th><th>B</th><th>Y</th></tr><tr><td>0</td><td>0</td><td>0</td></tr><tr><td>0</td><td>1</td><td>1</td></tr><tr><td>1</td><td>0</td><td>1</td></tr><tr><td>1</td><td>1</td><td>0</td></tr></table>	A	B	Y	0	0	0	0	1	1	1	0	1	1	1	0	$Y = A \oplus B$ $Y = AB + \overline{A}\overline{B}$
A	B	Y																	
0	0	0																	
0	1	1																	
1	0	1																	
1	1	0																	
XNOR		NOT + XOR, 즉 XOR의 부정	<table><tr><th>A</th><th>B</th><th>Y</th></tr><tr><td>0</td><td>0</td><td>1</td></tr><tr><td>0</td><td>1</td><td>0</td></tr><tr><td>1</td><td>0</td><td>0</td></tr><tr><td>1</td><td>1</td><td>1</td></tr></table>	A	B	Y	0	0	1	0	1	0	1	0	0	1	1	1	$Y = A \odot B$ $Y = A \oplus B$ $Y = AB + \overline{A}\overline{B}$
A	B	Y																	
0	0	1																	
0	1	0																	
1	0	0																	
1	1	1																	

<https://funarock.tistory.com/m/328>

# 퍼셉트론의 한계



x1	x2	s1	s2	y
0	0	1	0	0
1	0	1	1	1
0	1	1	1	1
1	1	0	1	0

```
def XOR(x1, x2):  
    s1 = NAND(x1, x2)  
    s2 = OR(x1, x2)  
    y = AND(s1, s2)  
    return y  
print(XOR(0, 0)) # 0을 출력  
print(XOR(1, 0)) # 1을 출력  
print(XOR(0, 1)) # 1을 출력  
print(XOR(1, 1)) # 0을 출력
```

# 다층 퍼셉트론(MLP)

## AND, OR는 단층 퍼셉트론

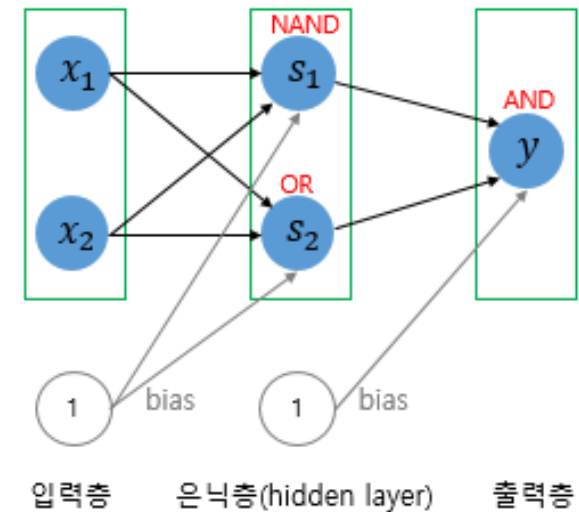
- 입력층과 출력층만 존재

## XOR은 2층 퍼셉트론

- 중간에 층을 더 추가

## 2층(다층) 퍼셉트론 서술

- 0층의 두 뉴런이 입력신호를 받아 1층의 뉴런으로 신호를 보냄
- 1층의 뉴런이 2층의 뉴런으로 신호를 보내고 2층의 뉴런은 이 입력신호를 바탕으로  $y$ 를 출력

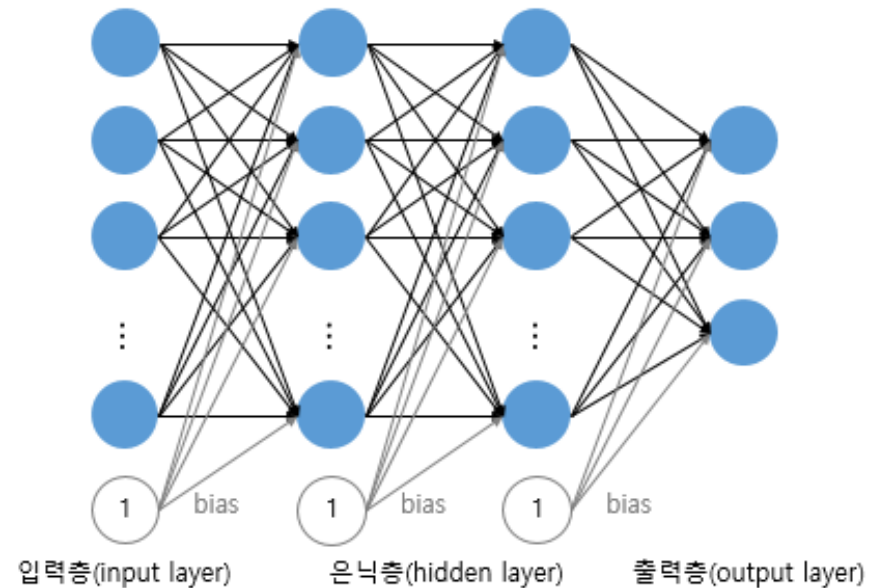


# 다층 퍼셉트론(MLP)

XOR 예제 : 은닉층 1개만으로 문제를 해결

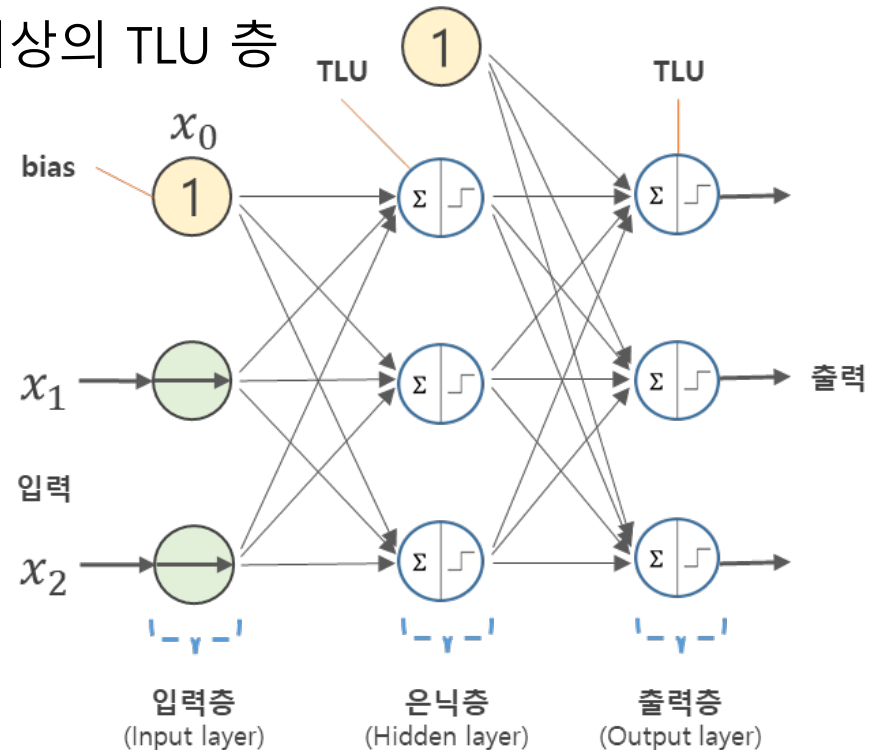
다층 퍼셉트론:

- 은닉층이 1개 이상인 퍼셉트론
- XOR 문제보다 더욱 복잡한 문제를 해결하기 위해서 다층 퍼셉트론은 중간에 수많은 은닉층 추가
- 은닉층이 2개 이상인 신경망
  - 심층 신경망(Deep Neural Network, DNN)



# 다층 퍼셉트론(MLP)

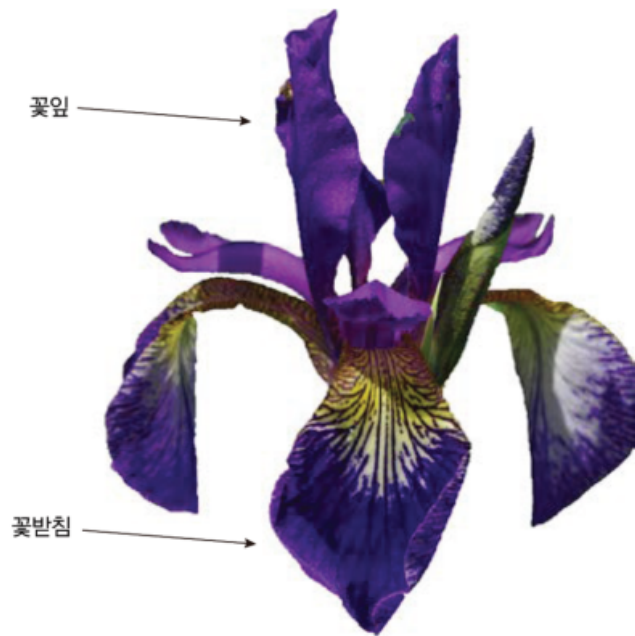
- 입력층, 은닉층(hidden layer)이라 부르는 하나 이상의 TLU 층
- 마지막으로 출력층(output layer)으로 구성



- MLP를 통해 XOR 문제를 해결
  - 층이 깊어질 수록 증가하는 가중치 매개변수의 수로 인해 다층 퍼셉트론을 학습시키기에는 오랜 시간이 걸리는 문제가 발생
    - (역전파(backpropagation) 알고리즘이 등장하면서 계산량을 획기적으로 줄일 수 있음) → 다음 수업

## 실습 - 붓꽃

Scikit-Learn은 하나의 TLU (Threshold Logic Unit )퍼셉트론을 구현한 [Perceptron](#) 클래스를 제공



꽃잎 petal  
꽃받침 sepal의 폭과  
꽃받침 sepal의 길이

```
from sklearn.datasets import load_iris  
from sklearn.linear_model import Perceptron
```

```
iris = load_iris()  
X = iris.data[:, (2, 3)] # petal length, width  
y = (iris.target == 0).astype(np.int)
```

```
per_clf = Perceptron(max_iter=100, random_state=42)  
per_clf.fit(X, y)
```

## Scikit-learn Perceptron

```
from sklearn.datasets import load_digits
from sklearn.linear_model import Perceptron
X, y = load_digits(return_X_y=True)
clf = Perceptron(tol=1e-3, random_state=0)
clf.fit(X, y)
Perceptron()
clf.score(X, y)
```

<code>decision_function(X)</code>	Predict confidence scores for samples.
<code>densify()</code>	Convert coefficient matrix to dense array format.
<code>fit(X, y[, coef_init, intercept_init, ...])</code>	Fit linear model with Stochastic Gradient Descent.
<code>get_params([deep])</code>	Get parameters for this estimator.
<code>partial_fit(X, y[, classes, sample_weight])</code>	Perform one epoch of stochastic gradient descent on given samples.
<code>predict(X)</code>	Predict class labels for samples in X.
<code>score(X, y[, sample_weight])</code>	Return the mean accuracy on the given test data and labels.
<code>set_params(**kwargs)</code>	Set and validate the parameters of estimator.
<code>sparsify()</code>	Convert coefficient matrix to sparse format.

[https://scikit-learn.org/stable/modules/generated/sklearn.linear\\_model.Perceptron.html](https://scikit-learn.org/stable/modules/generated/sklearn.linear_model.Perceptron.html)

## 실습 - 붓꽃

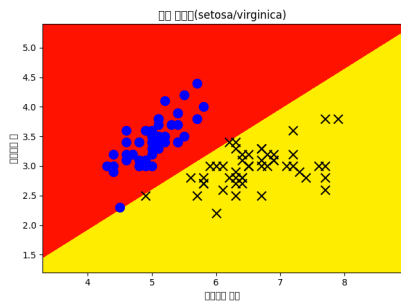


출처 : <http://mirlab.org/>의 아이리스 항목

<https://pinkwink.kr/1128?category=769346>



# 실습 - 붓꽃



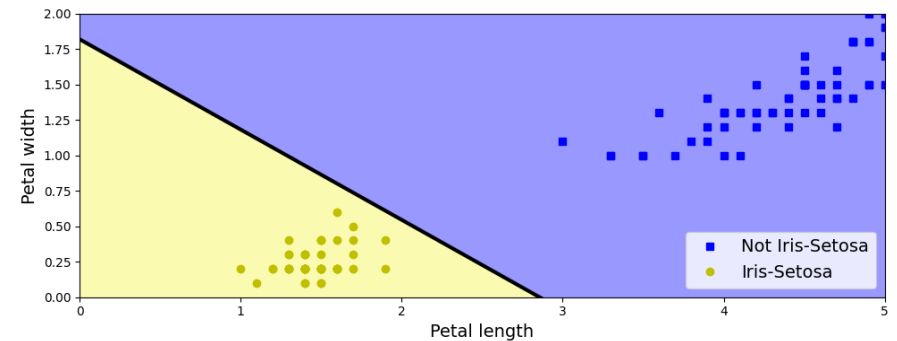
```
from sklearn.datasets import load_iris
from sklearn.linear_model import Perceptron
import numpy as np
import matplotlib as mpl
import matplotlib.pyplot as plt
iris = load_iris()
```

```
idx = np.in1d(iris.target, [0, 2])
X = iris.data[idx, :2]
y = (iris.target[idx] / 2).astype(np.int)
```


```
model = Perceptron(max_iter=300, shuffle=False, tol=0, n_iter_no_change=1e9).fit(X, y)
XX_min = X[:, 0].min() - 1
XX_max = X[:, 0].max() + 1
YY_min = X[:, 1].min() - 1
YY_max = X[:, 1].max() + 1
XX, YY = np.meshgrid(np.linspace(XX_min, XX_max, 1000),
                     np.linspace(YY_min, YY_max, 1000))
ZZ = model.predict(np.c_[XX.ravel(), YY.ravel()]).reshape(XX.shape)
plt.contourf(XX, YY, ZZ, cmap=mpl.cm.autumn)
plt.scatter(X[y == 0, 0], X[y == 0, 1], c='w', s=100, marker='o', edgecolor='k')
plt.scatter(X[y == 1, 0], X[y == 1, 1], c='k', s=100, marker='x', edgecolor='k')
plt.xlabel("꽃받침의 길이")
plt.ylabel("꽃받침의 폭")
plt.title("붓꽃 데이터(setosa/virginica)")
plt.xlim(XX_min, XX_max)
plt.ylim(YY_min, YY_max)
plt.grid(False)
plt.show()
```

# 실습 - 붓꽃

```
from sklearn.datasets import load_iris
from sklearn.linear_model import Perceptron
import numpy as np
import matplotlib.pyplot as plt
iris = load_iris()
print(iris.feature_names)
print(iris.target_names)
X, y = iris.data, iris.target
```



```
['sepal length (cm)', 'sepal width (cm)', 'petal length (cm)', 'petal width (cm)']
['setosa' 'versicolor' 'virginica']
```



이미지 및 코드 출처

<https://hunkim.github.io/>

Andrew Ng's ML class –

<https://class.coursera.org/ml-003/lecture>

<http://www.holehouse.org/mlclass/> (note)

Convolutional Neural Networks for Visual Recognition.

<http://cs231n.github.io/>

Hanson Machine Learning

<https://github.com/ExcelsiorCJH/Hands-On-ML/>