

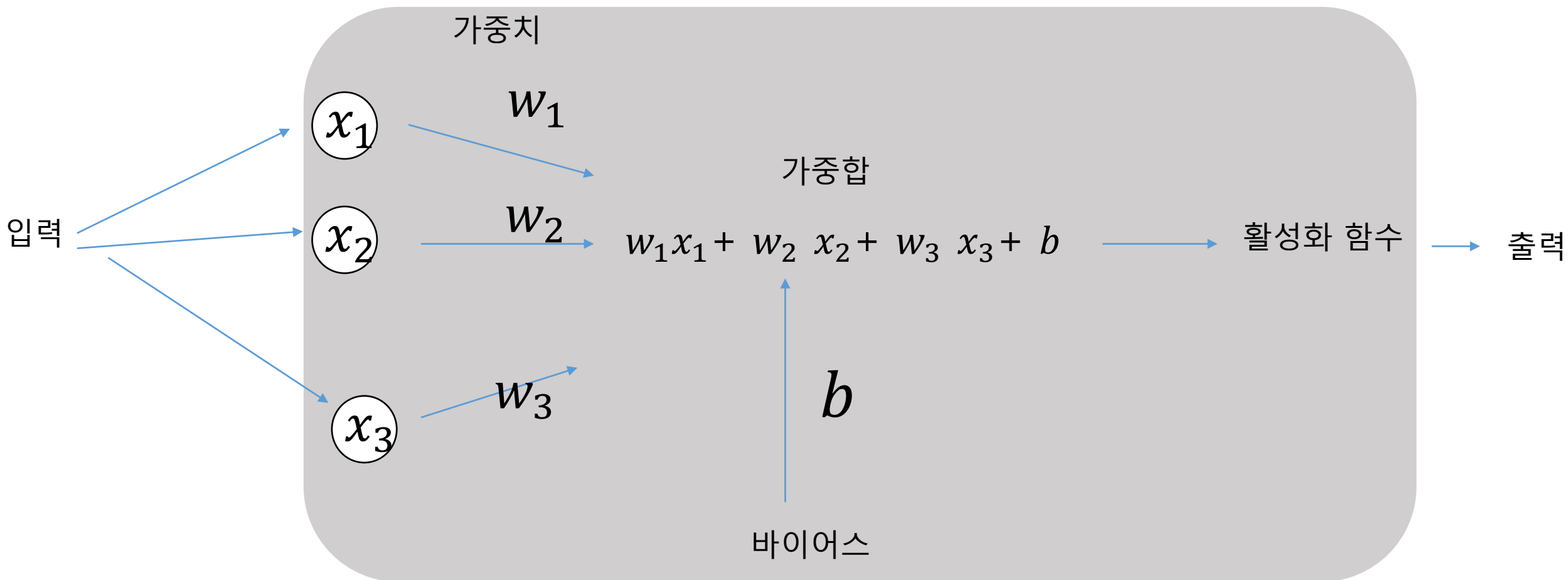
AI 프로그래밍

- 2024

7주차

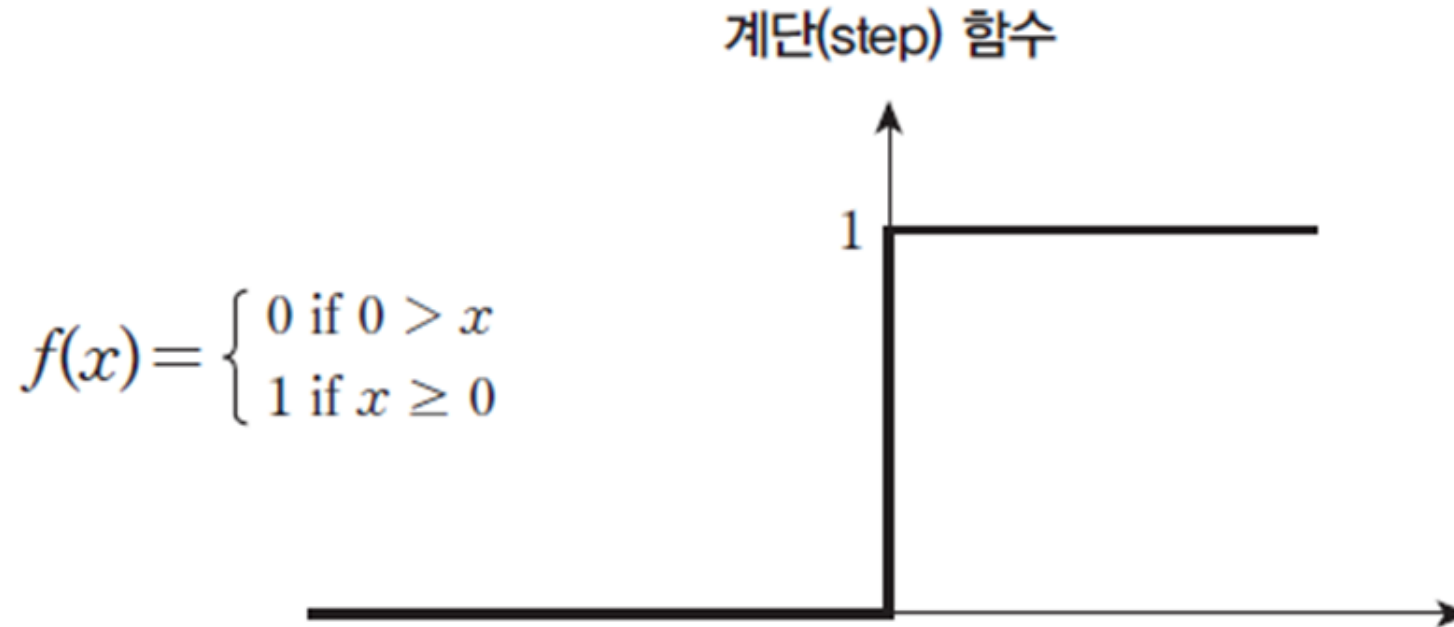
퍼셉트론 (입력 3개)

인하공전 컴퓨터 정보과



퍼셉트론 (Perceptron)

인하공전 컴퓨터 정보과

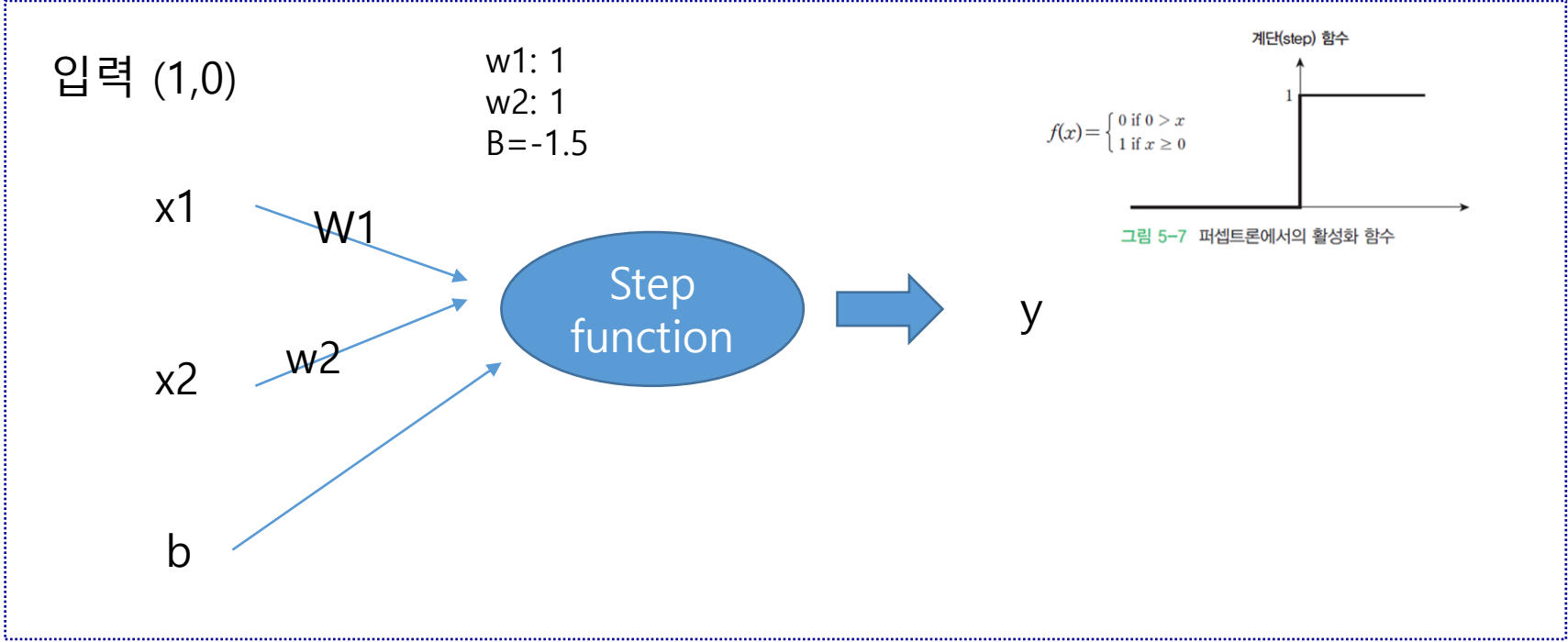


-
- 가중치는 입력 신호가 출력에 미치는 **중요도**를 조절
- 바이어스는 뉴런의 활성화 기준
- **가중합 (입력 2개) : $w_1 \cdot x_1 + w_2 \cdot x_2 + b$**

- 뉴런에서는 입력 신호의 가중치 합이 어떤 임계값을 넘는 경우에만 뉴런이 활성화되어서 1을 출력한다. 그렇지 않으면 0을 출력한다.

$$y = \begin{cases} 1 & \text{if } (w_1x_1 + w_2x_2 + b \geq 0) \\ 0 & \text{otherwise} \end{cases}$$

퍼셉트론 (Perceptron)



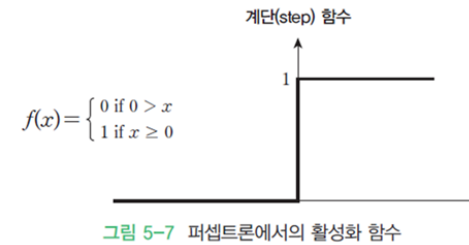
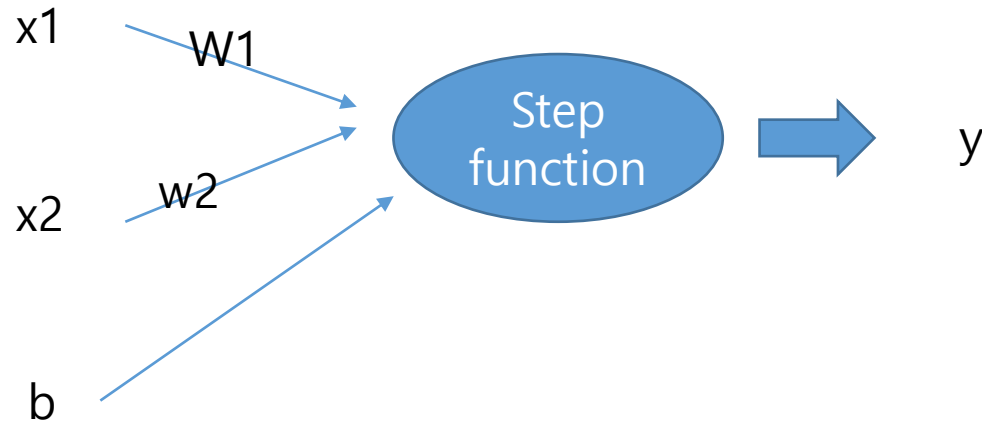
출력값 y 는?

퍼셉트론 (Perceptron)

인하공전 컴퓨터 정보과

입력 (1,0)

w1: 1
w2: 1
B=-1.5



출력값 y 는? $x_1 * w_1 + x_2 * w_2 + b =$

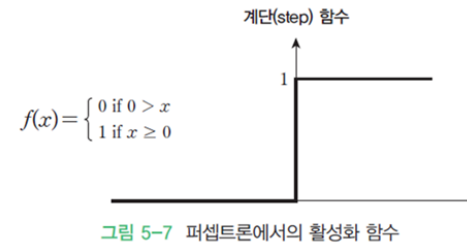
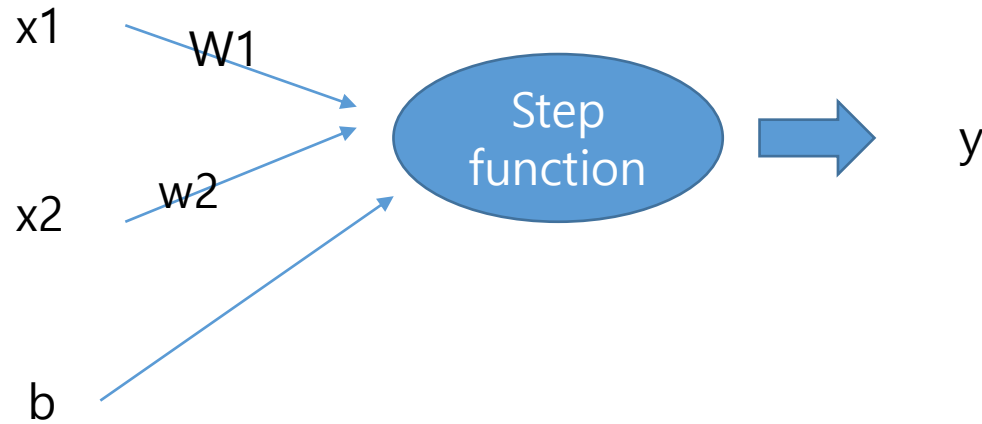
-> step function ->

퍼셉트론 (Perceptron)

인하공전 컴퓨터 정보과

입력 (1,1)

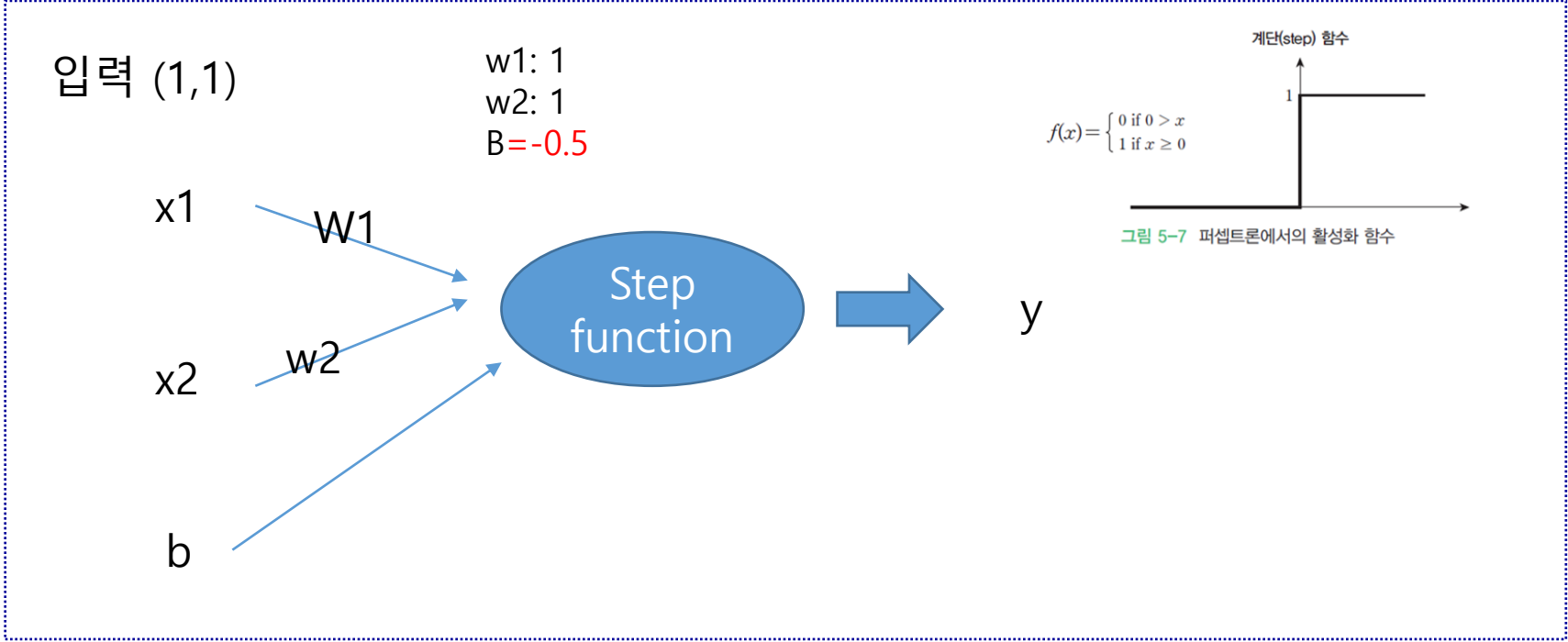
w1: 1
w2: 1
B=-1.5



출력값 y 는? $x_1 * w_1 + x_2 * w_2 + b =$

-> step function ->

퍼셉트론 (Perceptron)



출력값 y 는? $x1*w1+x2*w2+b=$ -> step function ->

퍼셉트론 (Perceptron) – 논리 연산 수행

인하공전 컴퓨터 정보과

AND 연산

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

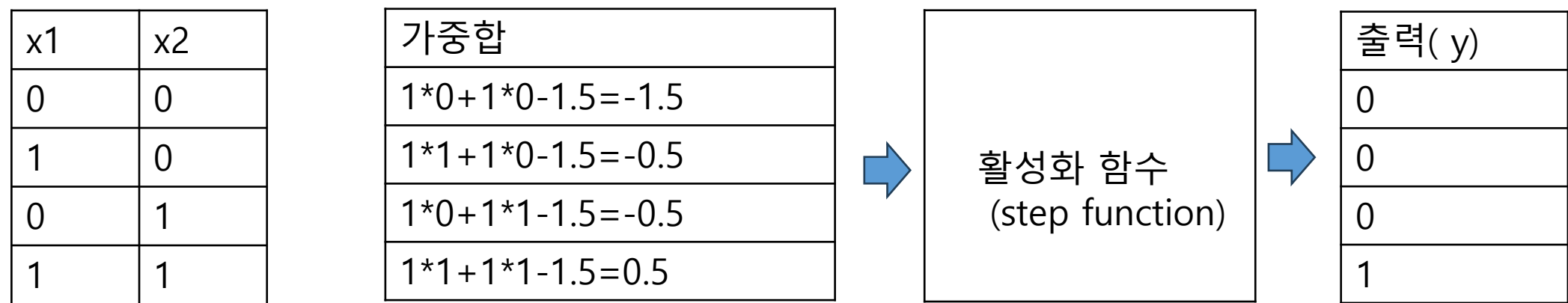
만족시키는 w_1, w_2, b 찾기

$w_1=1, w_2=1, b=-1.5$ 를 테스트 해보기

퍼셉트론 (Perceptron) – 논리 연산 수행

$w_1=1, w_2=1, b=-1.5$

가중합: $x_1*w_1+x_2*w_2+b$



퍼셉트론 (Perceptron) 구현

인하공전 컴퓨터 정보과

```
epsilon = 0.0000001

def perceptron(x1, x2):
    w1, w2, b = 1.0, 1.0, -1.5
    sum = x1*w1+x2*w2+b
    if sum > epsilon :           # 부동소수점 오차를 방지하기 위하여
        return 1
    else :
        return 0

print(perceptron(0, 0))
print(perceptron(1, 0))
print(perceptron(0, 1))
print(perceptron(1, 1))
```

w1=1.0, w2=1.0, b=-0.5 일때 출력 구하기

- 1) 입력이 (0,0) 일 때
- 2) 입력이 (1,0) 일 때
- 3) 입력이 (0,1) 일 때
- 4) 입력이 (1,1) 일 때

과제 1) code upload(*.py), 출력값 입력

퍼셉트론 (Perceptron) 학습

인하공전 컴퓨터 정보과

- 퍼셉트론 도 scikit에서 학습 이 가능

퍼셉트론 (Perceptron) 학습 -scikit learn

인하공전 컴퓨터 정보과

```
from sklearn.linear_model import Perceptron

# 논리적 AND 연산 샘플과 정답이다.
X = [[0,0],[0,1],[1,0],[1,1]]          # 항상 2차원 배열이어야 한다.
y = [0, 0, 0, 1]

# 퍼셉트론을 생성한다. tol은 종료 조건이다. random_state는 난수의 시드이다.
clf = Perceptron(tol=1e-3, random_state=0)

# 학습을 수행한다.
clf.fit(X, y)
|
# 테스트를 수행한다.
print(clf.predict(X))
```

과제 2) code upload(*.py), 출력 값 입력

퍼셉트론 (Perceptron) 학습 OR 연산

인하공전 컴퓨터 정보과

x_1	x_2	Y
0	0	0
1	0	1
0	1	1
1	1	1

OR

SCIKIT LEARN 학습 한 후 예측 결과를 제출

과제 3) 결과값 제출

퍼셉트론 (Perceptron) 학습 XOR 연산

인하공전 컴퓨터 정보과

x1	x2	y
0	0	0
1	0	1
0	1	1
1	1	0

XOR 연산을 수행하는 퍼셉트론 학습

과제4) 결과값 제출

- XOR 연산은 퍼셉트론으로 학습이 불가능 하다

- 퍼셉트론은 간단한 xor문제를 해결 못함.
- 해결 하기 위해선 => 다층 퍼셉트론 필요 (Multilayer perceptron)

```
arr1 = np.array([[1, 2], [3, 4], [5, 6]])  
arr2 = np.array([[2, 2], [2, 2], [2, 2]])  
result = arr1 * arr2  
result  
array([[ 2,  4],  
       [ 6,  8],  
       [10, 12]])
```

```
import numpy as np
```

```
a = np.array([[1, 2, 3], [4, 5, 6]])
```

```
b = np.array([[3, 2, 1, 3], [2, 0, 3, 2], [1, 3, 1, 1]])
```

```
c1=a@b
```

```
c2=np.dot(a,b)
```

```
c3=a.dot(b)
```

```
C4=np.matmul(a,b)
```

c1,c2,c3,c4 모두 행렬 곱셈을 나타냄

1	2	3
4	5	6

dot

3	2	1	3
2	0	3	2
1	3	1	1

1차원 행렬 곱셈

인하공전 컴퓨터 정보과

3	2	1
---	---	---

dot

4	5	6
---	---	---

$$=3*4+2*5+1*6=28$$

- $A=[1,2,3]$

- $B=[4,5,6]$

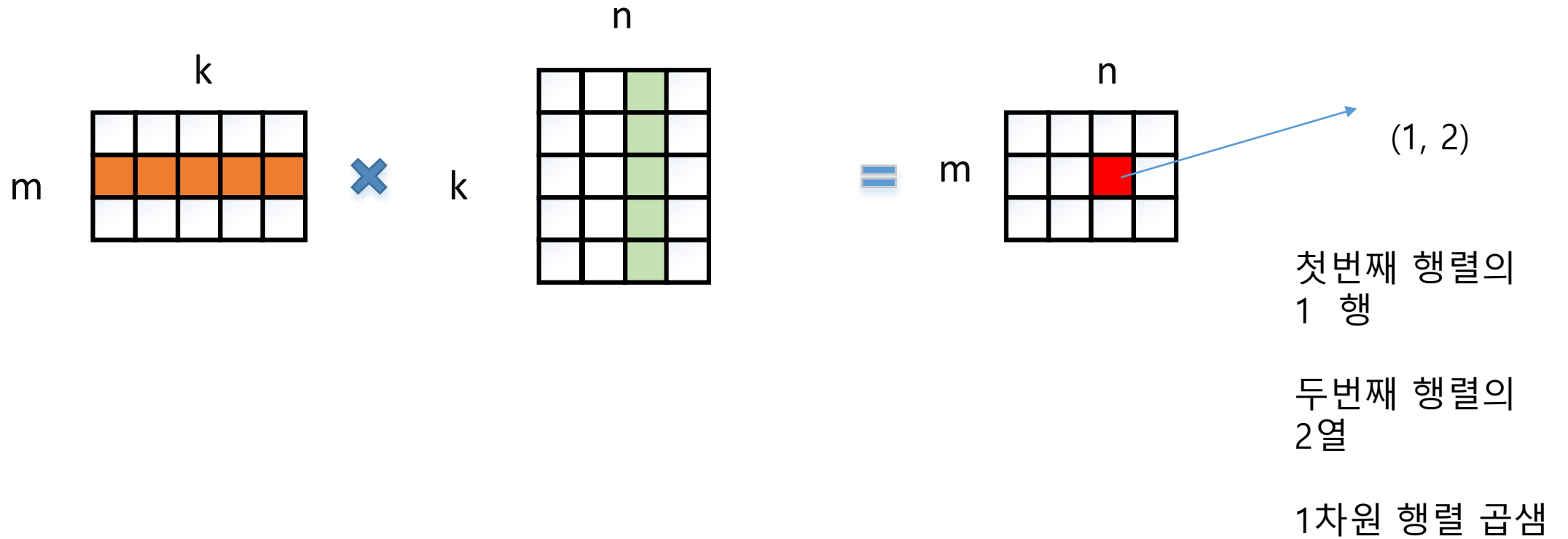
- $A*B$ 일반 곱셈

$\Rightarrow A*B \Rightarrow [4,10,18]$

- $A*B$ 행렬 곱셈

$\text{np.dot}(A,B)$

- $1*4+2*5+3*6=4+10+18=32$



$$(m,k) \times (k,n) = (m, n)$$

1	2	3
4	5	6

dot

3	2	1	3
2	0	3	2
1	3	1	1

=

1)	2)	3)	10
28	26	25	28

(2,3)

(3,4)

(2,4)

1) $1*3+2*2+3*1=10$

1	2	3
4	5	6

dot

3	2	1	3
2	0	3	2
1	3	1	1

=

1)	2)	3)	10
28	26	25	28

(2,3)

(3,4)

(2,4)

2) $1*2+2*0+3*3=11$

1	2	3
4	5	6

(2,3)

dot

3	2	1	3
2	0	3	2
1	3	1	1

(3,4)

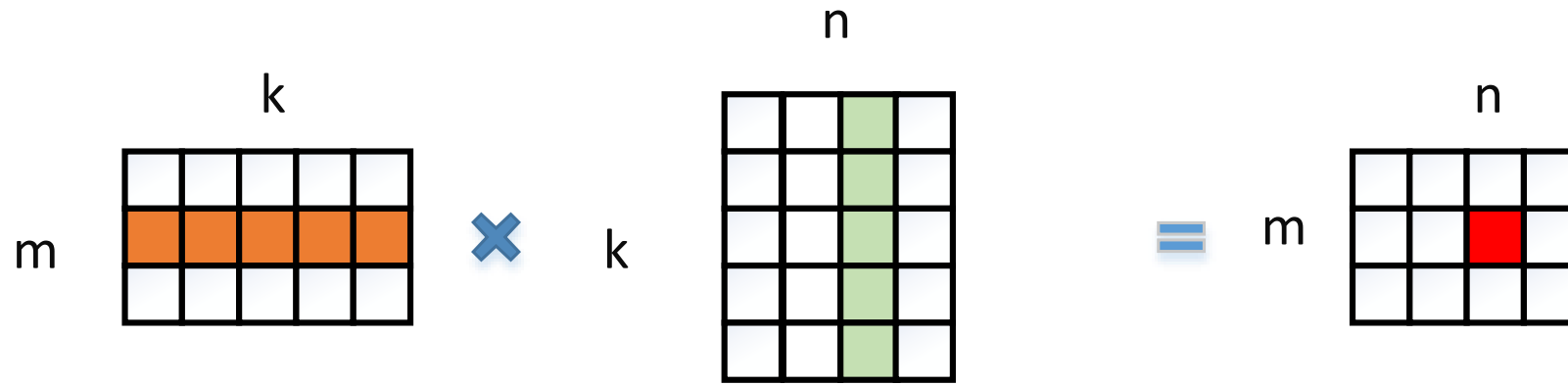
=

1)	2)	3)	10
28	26	25	28

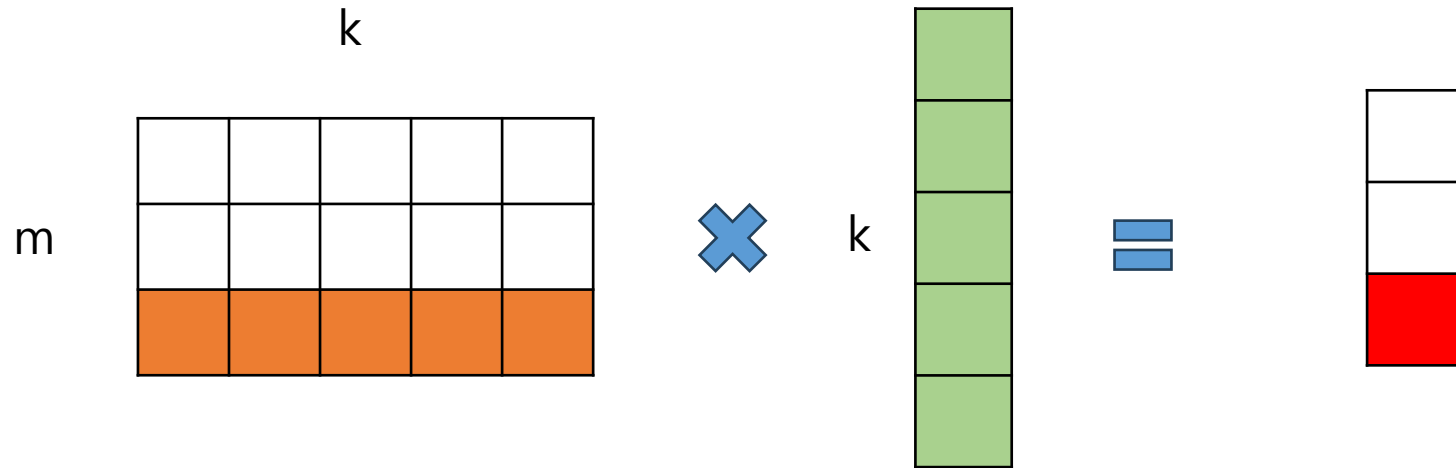
(2,4)

3)

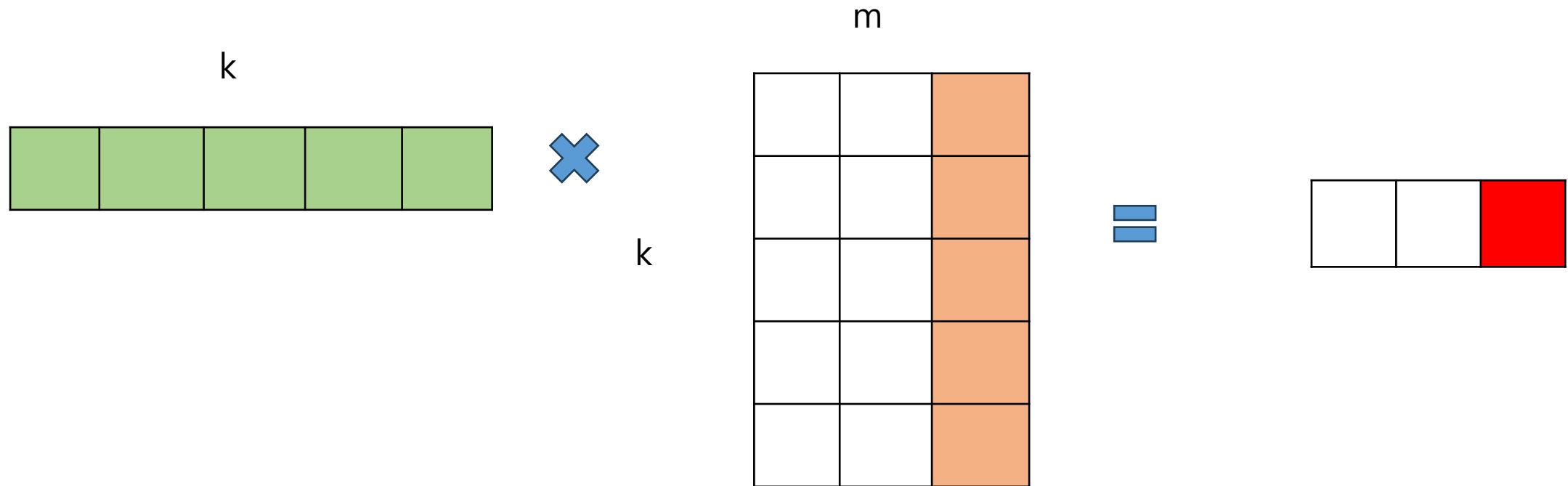
```
>>> arr1 = np.array([[1, 2, 3], [4, 5, 6], [7, 8, 9]])
>>> arr2 = np.array([[2, 2], [2, 2], [2, 2]])
>>> result = arr1 @ arr2          # arr1.dot(arr2) 도 가능
array([[12, 12],
       [30, 30],
       [48, 48]])
```



$$(m,k) \times (k,n) = (m, n)$$



$$(m,k) \times (k) = (m,)$$



$$(k) \times (k,m) = (m,)$$

- $A=[1,2,3]$

- $B=[4,5,6]$

- $A*B$ 일반 곱셈

$\Rightarrow A*B \Rightarrow [4,10,18]$

- $A*B$ 행렬 곱셈

- $1*4+2*5+3*6=4+10+18=32$

$$W=[w_1, w_2], \quad X=[x_1, x_2]$$

$$WX=w_1x_1+w_2x_2$$

$$WX=\text{np. matmult}(W, X)$$

$$W=[1, 2], \quad X=[3, 4] \quad WX= ?$$

$$\text{np. matmult}(W, X)$$

$$W = \begin{bmatrix} w_1 & w_2 \\ w_3 & w_4 \end{bmatrix} \quad X = \begin{bmatrix} x_1 \\ x_2 \end{bmatrix}$$

$$W = \begin{bmatrix} 1 & 2 \\ 3 & 4 \end{bmatrix} \quad X = \begin{bmatrix} 5 \\ 6 \end{bmatrix}$$

$$WX = \begin{bmatrix} w_1x_1 + w_2x_2 \\ w_3x_1 + w_4x_2 \end{bmatrix}$$

$$WX=?$$

$$\text{np. matmult}(W, X)$$

$W=[w_1, w_2], \quad X=[x_1, x_2]$

$WX=w_1*x_1+w_2*x_2$

$WX=np. \text{matmult}(W, X)$

$W=[1, 2], X=[3, 4] \quad WX= ?$

$np. \text{matmult}(W, X)$

```
import numpy as np
W=np.array([ 1 ,2 ])
X=np.array([3,4])
```

```
print(W*X)      => [3  8]
print(np. matmult(W,X))  => 11
```

$W= \begin{bmatrix} w_1 & w_2 \\ w_3 & w_4 \end{bmatrix} \quad X= \begin{bmatrix} x_1 \\ x_2 \end{bmatrix}$

$W= \begin{bmatrix} 1 & 2 \\ 3 & 4 \end{bmatrix} \quad X= \begin{bmatrix} 5 \\ 6 \end{bmatrix}$

$WX= \begin{bmatrix} w_1x_1 + & w_2x_2 \\ w_3x_1 + & w_4x_2 \end{bmatrix}$

$WX=?$

$np. \text{matmult}(W, X)$

```
import numpy as np
W=np.array([[ 1 ,2 ], [3, 4]])
X=np.array([5,6])
print(np. matmult(W, X) )
```

```
[17 39]
```


행렬 곱셈 연습 (과제 5)

인하공전 컴퓨터 정보과

5-1. $W=[1, 2], \quad X=[0,3]$

1) $W*X$ 의 결과는?

2) WX 의 결과는?

5-2. $W = \begin{bmatrix} 1 & 2 \\ 0 & 4 \end{bmatrix} \quad X = \begin{bmatrix} 5 \\ 2 \end{bmatrix}$

1) WX 의 결과는?

5-3. $W = \begin{bmatrix} 7 & 3 \\ 1 & 2 \end{bmatrix} \quad X = \begin{bmatrix} 0 \\ 4 \end{bmatrix}$

1) WX 의 결과는?

5-4. $W=[3, 1], \quad X=[2,1]$

1) $W*X$ 의 결과는?

2) WX 의 결과는?

5-5. $W = \begin{bmatrix} 1 & 2 \\ 0 & 4 \end{bmatrix} \quad X = \begin{bmatrix} 5 \\ 2 \end{bmatrix}$

1) WX 의 결과는?

결과 입력

$WX = \text{np.matmul}(W, X)$

$$W=[w1, w2], \quad X=[x1,x2]$$

$$WX=w1*x1+w2*x2$$

$$W= \begin{bmatrix} w1 & w2 \\ w3 & w4 \end{bmatrix} \quad X= \begin{bmatrix} x1 \\ x2 \end{bmatrix}$$

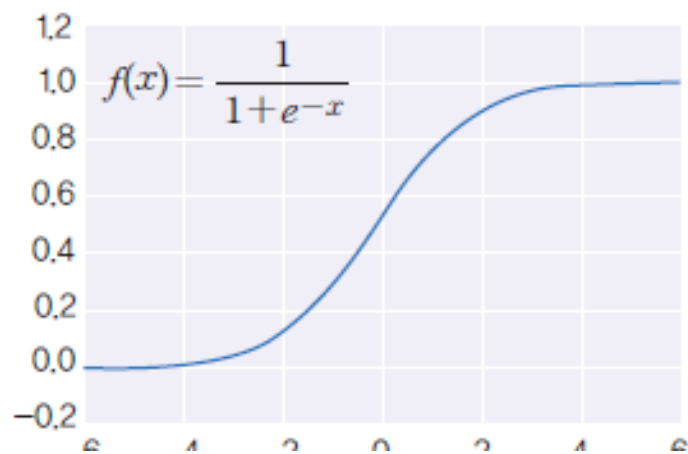
$$WX= \begin{bmatrix} w1x1 + & w2x2 \\ wx1 + & w4x2 \end{bmatrix}$$

$$w1*x1+w2*x2+b \quad \Rightarrow WX+b$$

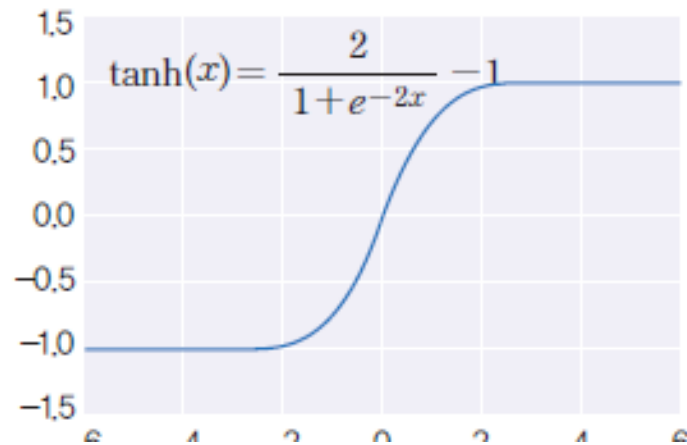
Np dot 하고 그냥 곱셈 차이

- 활성화 함수(activation function)은 입력의 총합을 받아서 출력값을 계산하는 함수이다
- MLP에서는 다양한 활성화 함수를 사용한다.

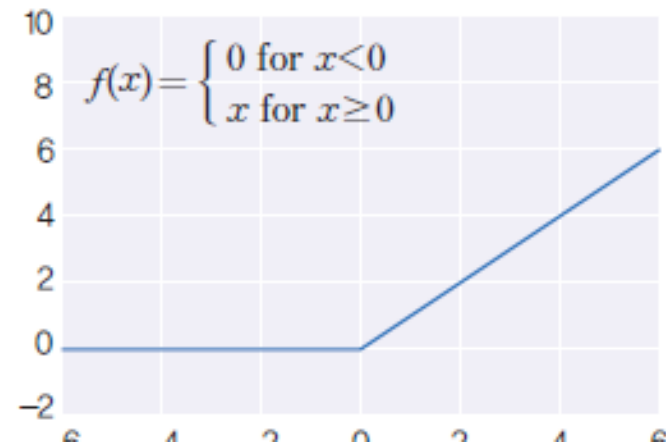
Sigmoid



TanH

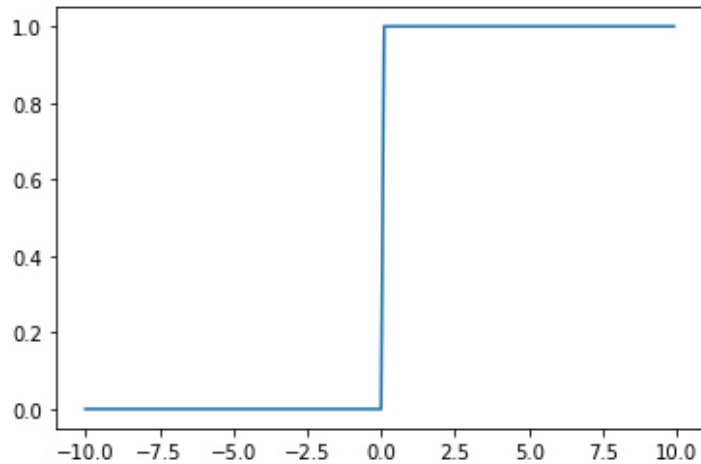


ReLU



- 계단 함수는 입력 신호의 총합이 0을 넘으면 1을 출력하고, 그렇지 않으면 0을 출력하는 함수이다.

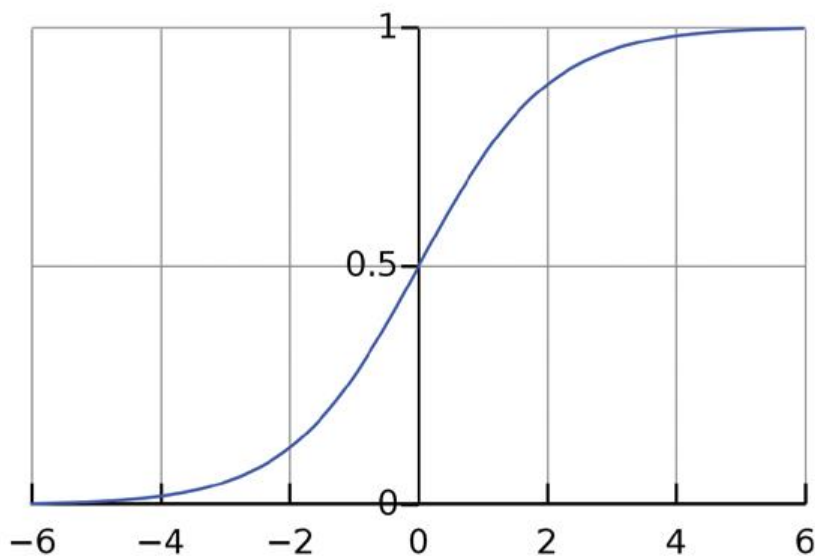
$$f(x) = \begin{cases} 1 & x > 0 \\ 0 & x \leq 0 \end{cases}$$



활성화 함수 (시그모이드)

- 1980년대부터 사용돼온 전통적인 활성화 함수이다. 시그모이드는 다음과 같이 s자와 같은 형태를 가진다.

$$f(x) = \frac{1}{1 + e^{-x}}$$

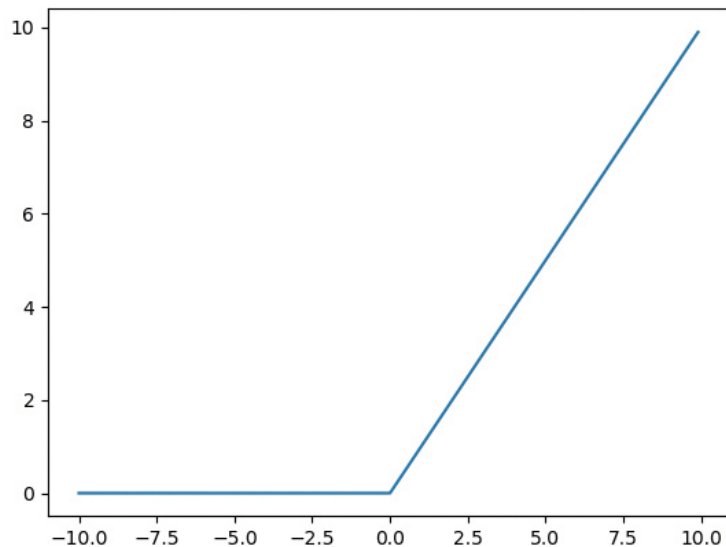


활성화 함수 (Rectified Linear Unit function)

인하공전 컴퓨터 정보과

- ReLU 함수는 최근에 가장 인기 있는 활성화 함수이다. ReLU 함수는 입력이 0을 넘으면 그대로 출력하고, 입력이 0보다 적으면 출력은 0이 된다.

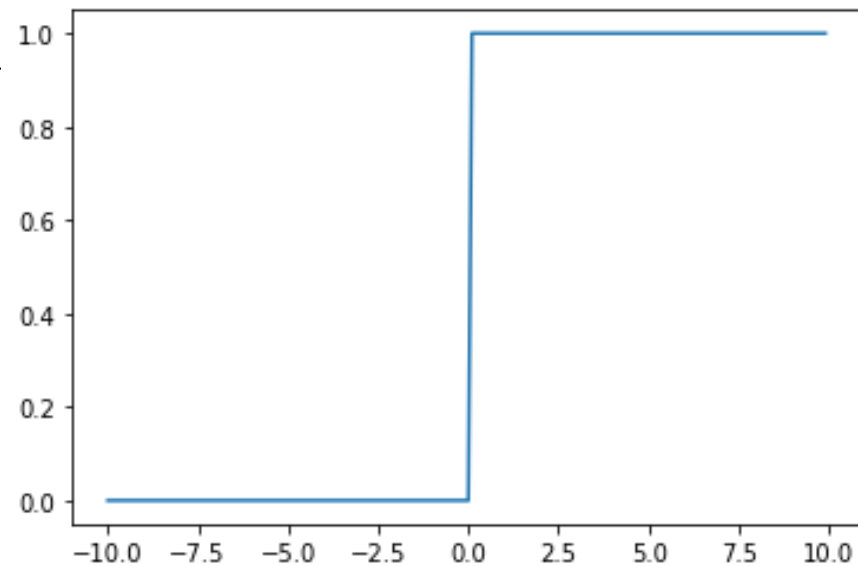
$$f(x) = \begin{cases} x & x > 0 \\ 0 & x \leq 0 \end{cases}$$



Step 함수

```
def step(x):  
    result = x > 0.000001          # True 또는 False  
    return result.astype(np.int)    # 정수로 반환
```

```
import numpy as np  
import matplotlib.pyplot as plt  
  
x = np.arange(-10.0, 10.0, 0.1)  
y = step(x)  
plt.plot(x, y)  
plt.show()
```



시그모이드 함수

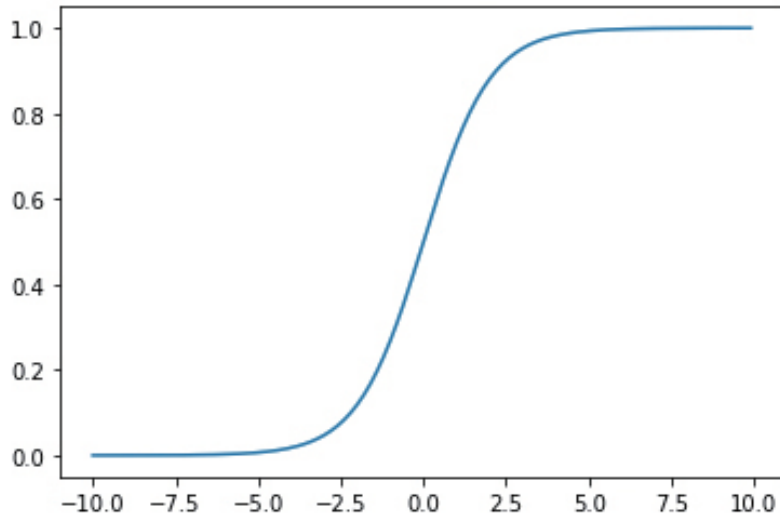
인하공전 컴퓨터 정보과

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

```
def sigmoid(x):
    return 1.0 / (1.0 + np.exp(-x))
```

```
x = np.arange(-10.0, 10.0, 0.1)
y = sigmoid(x)
plt.plot(x, y)
plt.show()
```

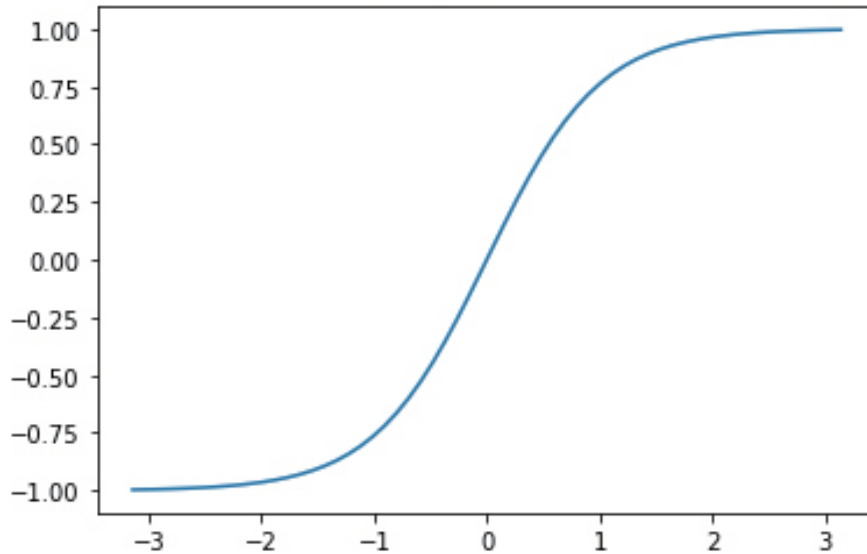
$$f(x) = \frac{1}{1 + e^{-x}}$$



Tanh 함수

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

x = np.linspace(-np.pi, np.pi, 60)
y = np.tanh(x)
plt.plot(x, y)
plt.show()
```

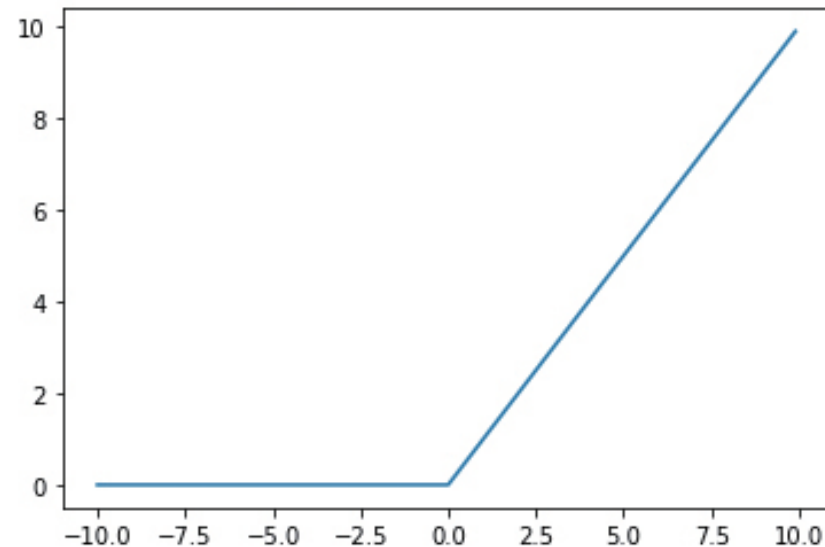


relu 함수

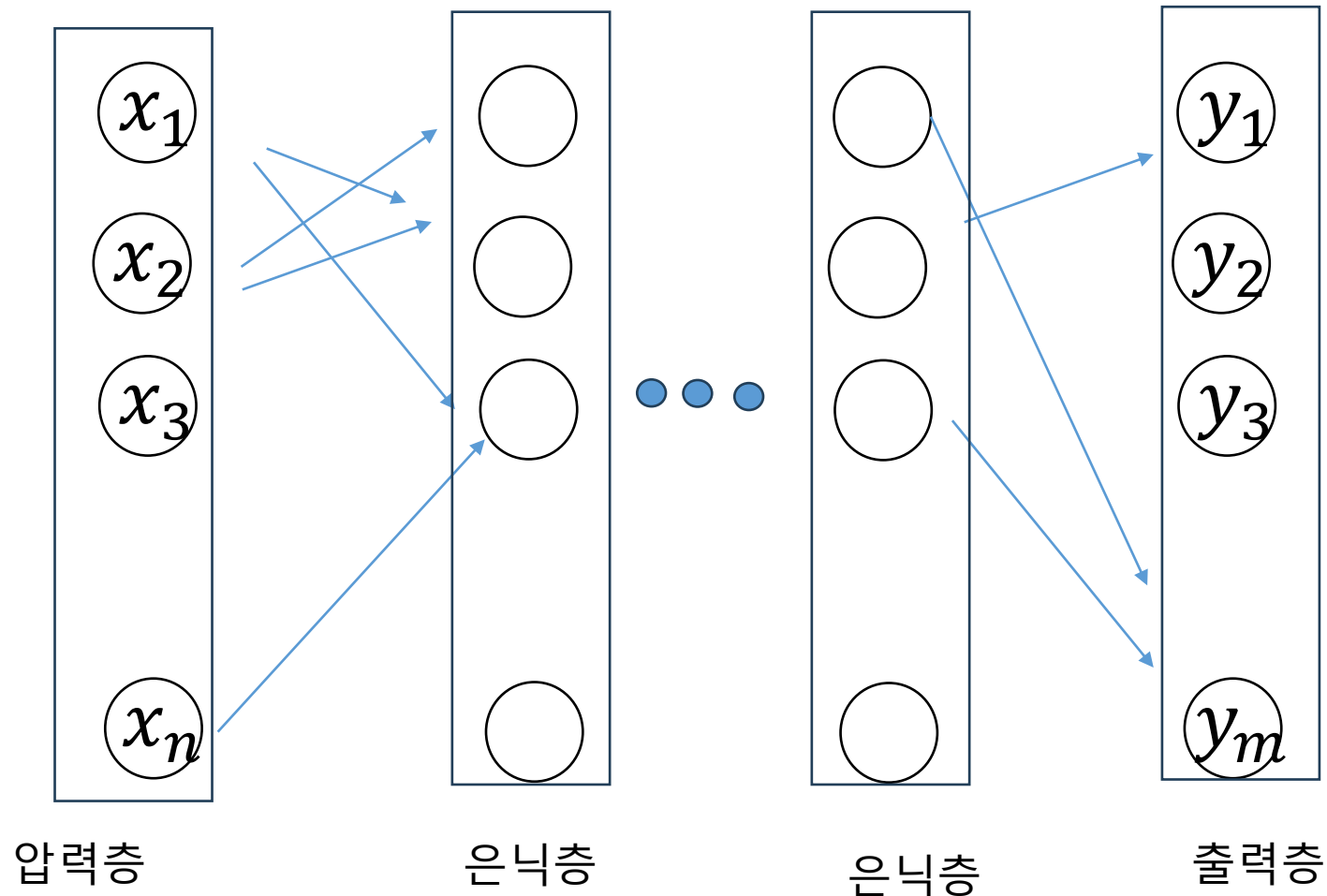
```
import numpy as np
import matplotlib.pyplot as plt

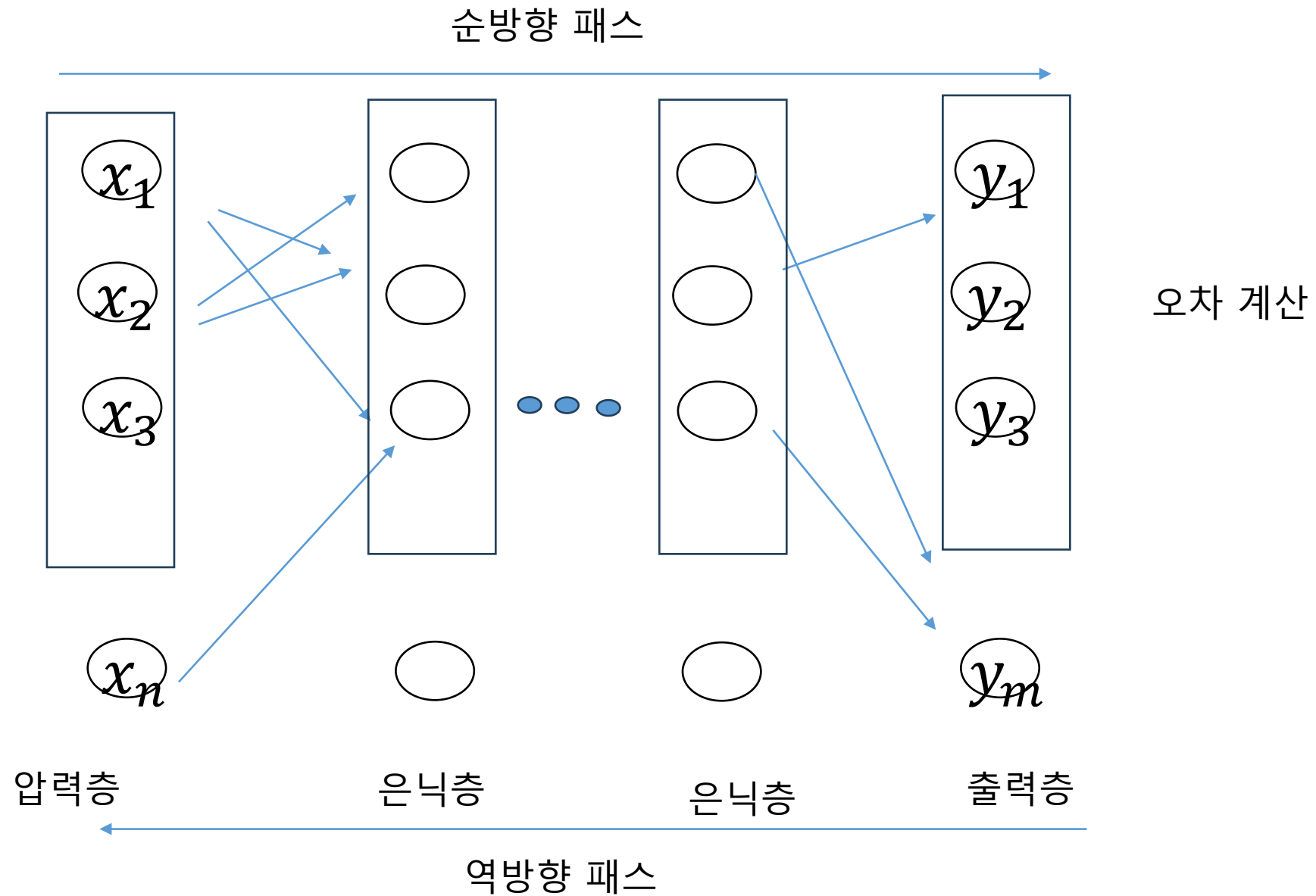
def relu(x):
    return np.maximum(x, 0)

x = np.arange(-10.0, 10.0, 0.1)
y = relu(x)
plt.plot(x, y)
plt.show()
```



- 다층 퍼셉트론(multilayer perceptron: MLP): 입력층과 출력층 사이에 은닉층(hidden layer)을 존재

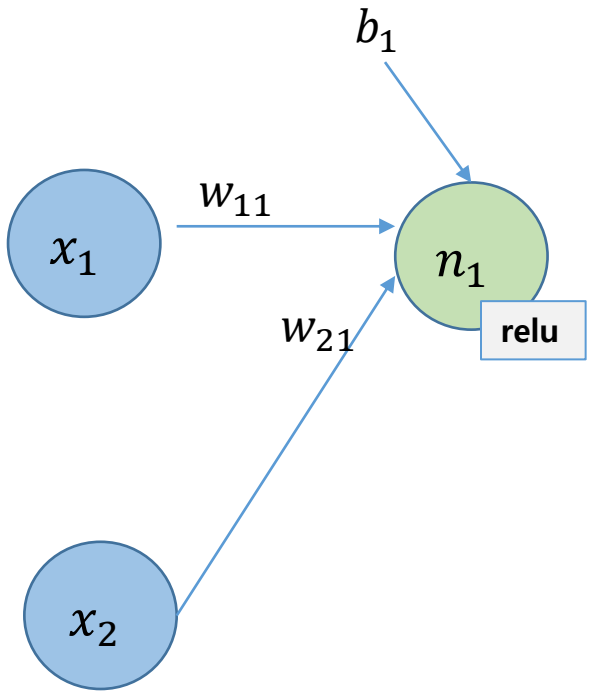
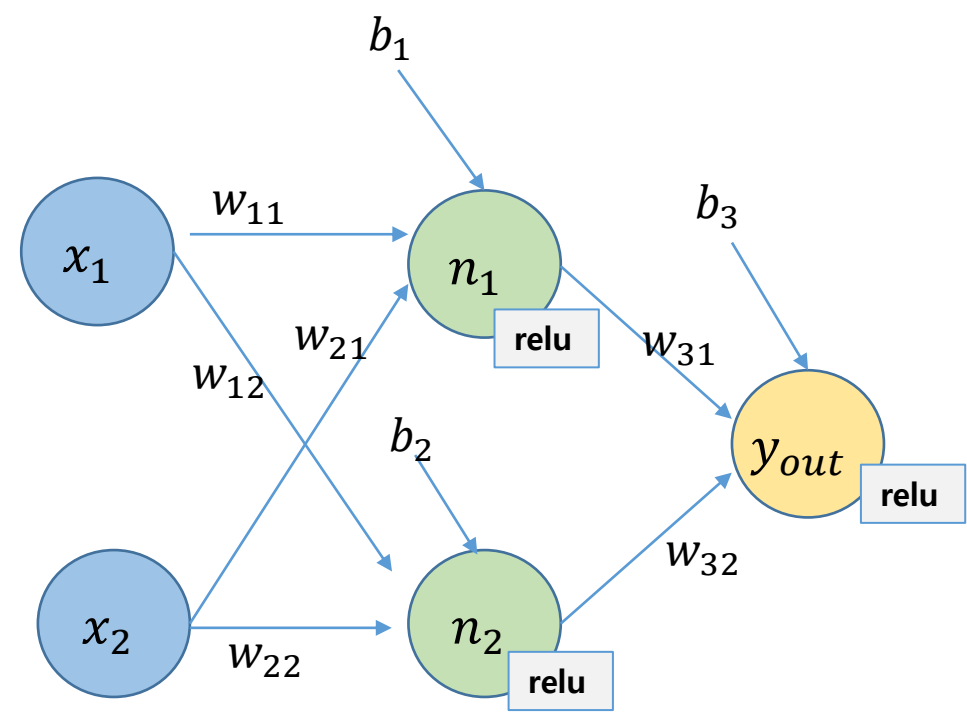




다중 퍼셉트론 (Multi Layer Perceptron: MLP)

인하공전 컴퓨터 정보과

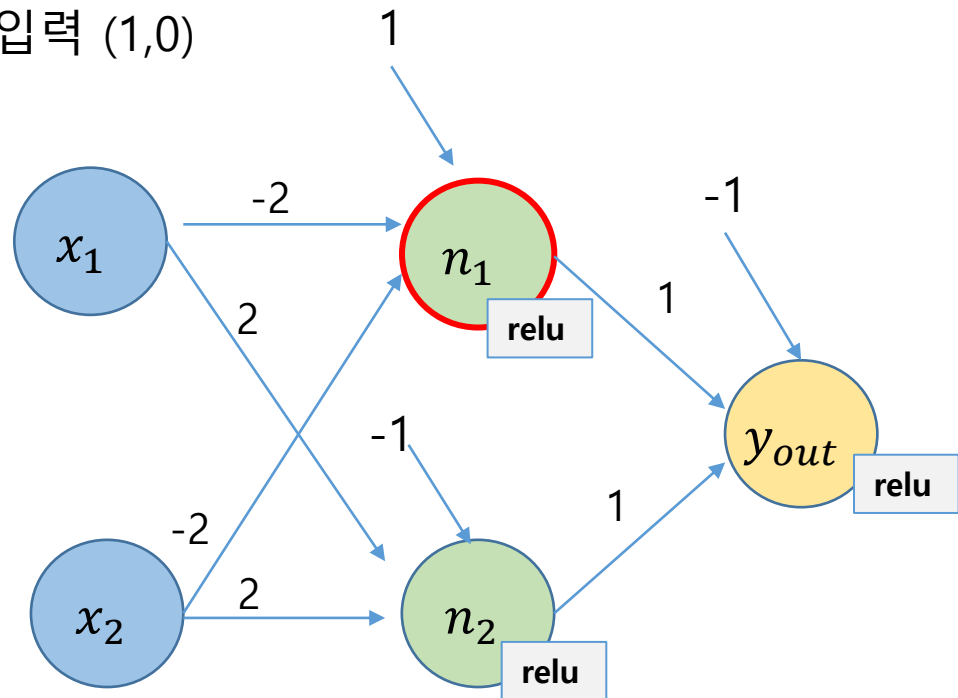
입력 (1,0)



다중 퍼셉트론 (Multi Layer Perceptron: MLP)

인하공전 컴퓨터 정보과

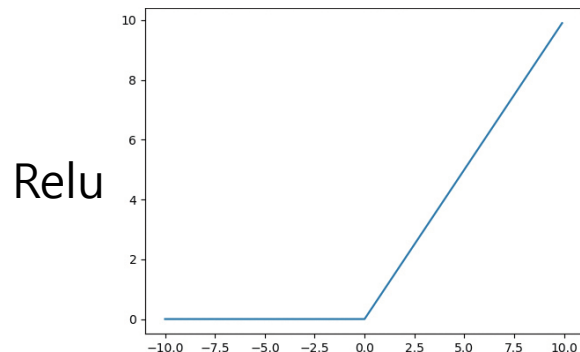
입력 (1,0)



다음 다중 퍼셉트론의 출력은?

w_{12}

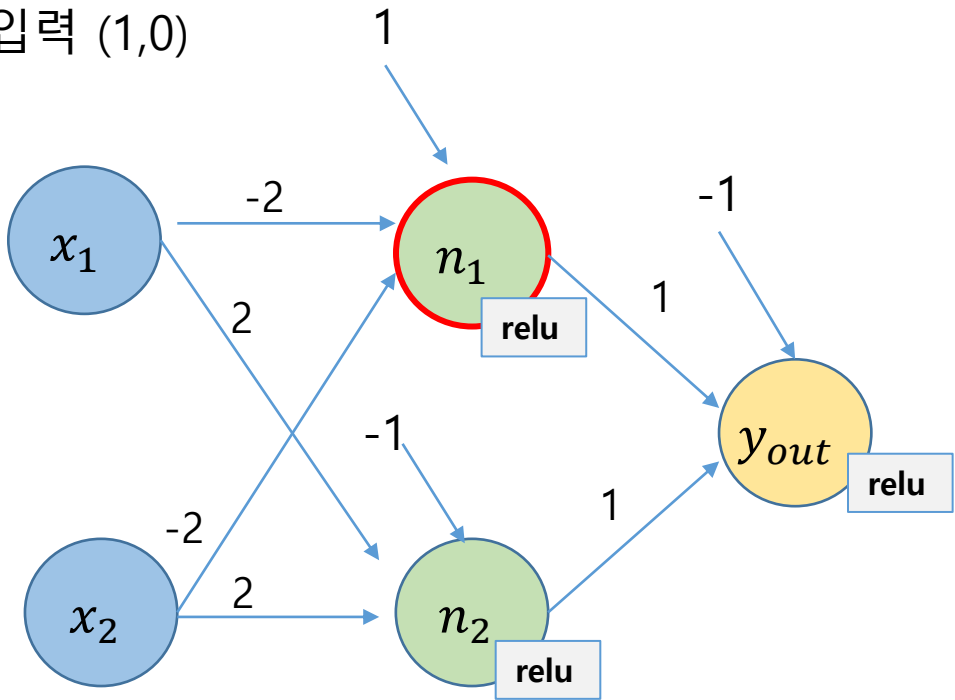
$$f(x) = \begin{cases} x & x > 0 \\ 0 & x \leq 0 \end{cases}$$



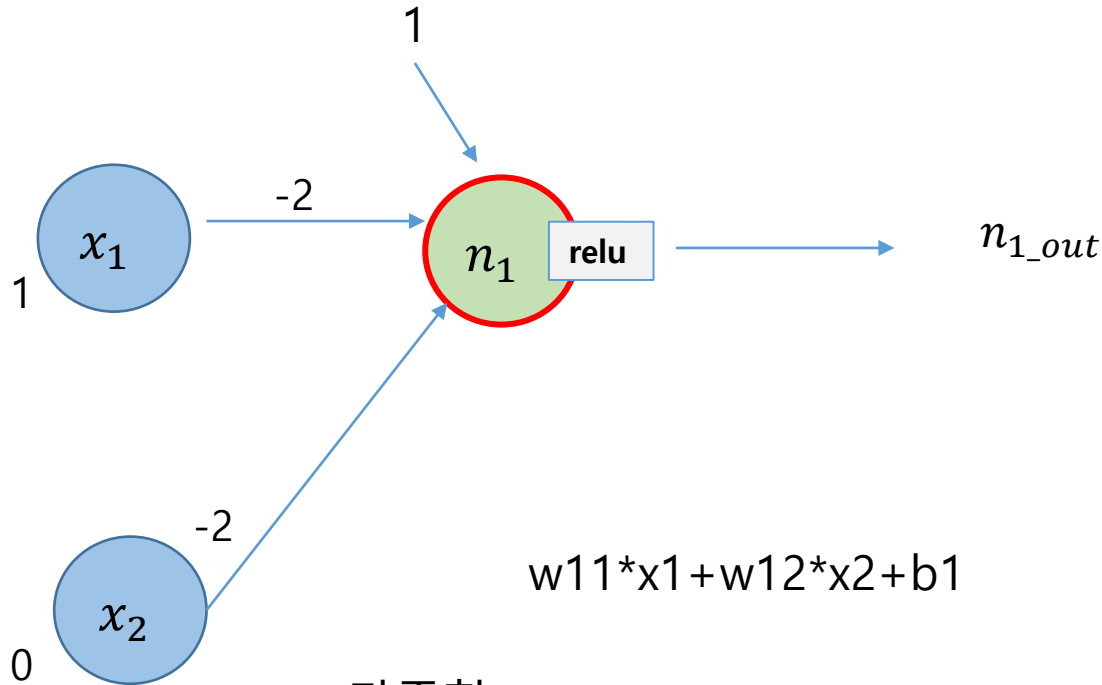
다중 퍼셉트론 (Multi Layer Perceptron: MLP)

인하공전 컴퓨터 정보과

입력 (1,0)



다음 다중 퍼셉트론의 출력은?



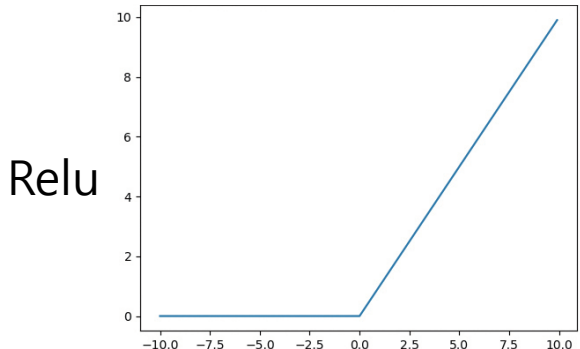
$$w_{11} \cdot x_1 + w_{12} \cdot x_2 + b_1$$

가중합: $(-2) \cdot 1 + (-2) \cdot 0 + 1 = -1$

$\text{Relu}(-1) = 0$

$n_{1_out} = 0$

$$f(x) = \begin{cases} x & x > 0 \\ 0 & x \leq 0 \end{cases}$$

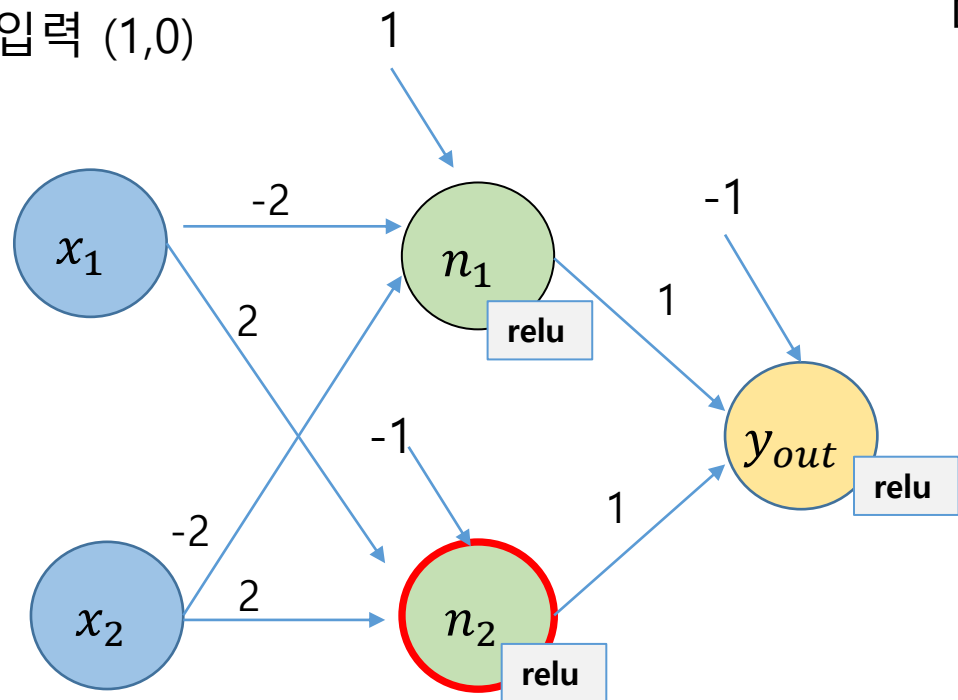


다중 퍼셉트론 (Multi Layer Perceptron: MLP)

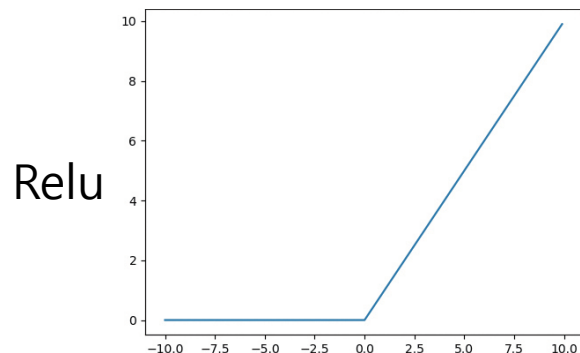
인하공전 컴퓨터 정보과

입력 (1,0)

다음 다중 퍼셉트론의 출력은?



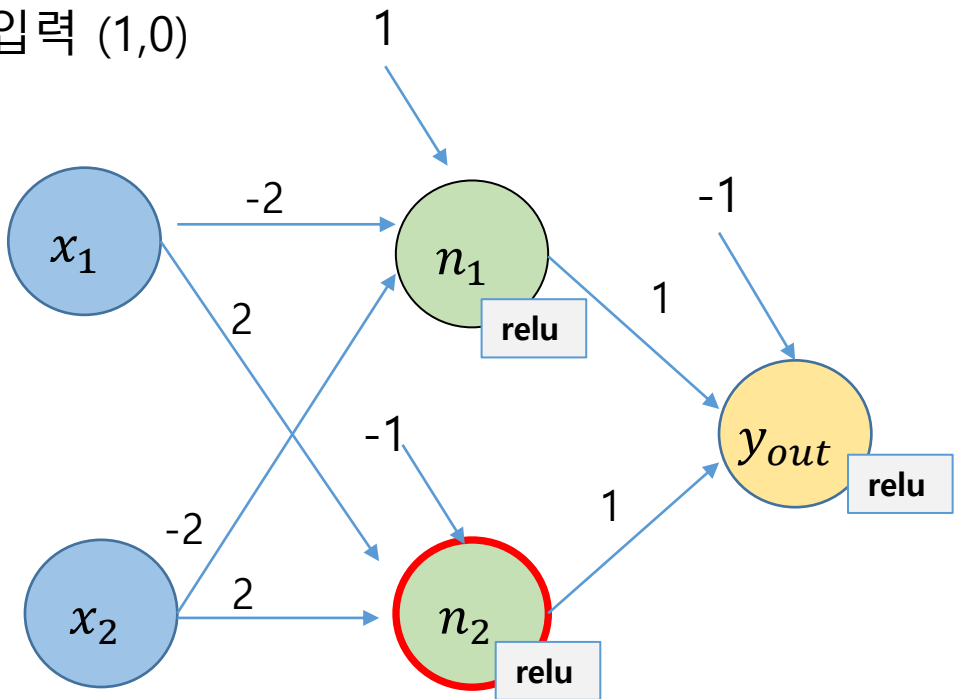
$$f(x) = \begin{cases} x & x > 0 \\ 0 & x \leq 0 \end{cases}$$



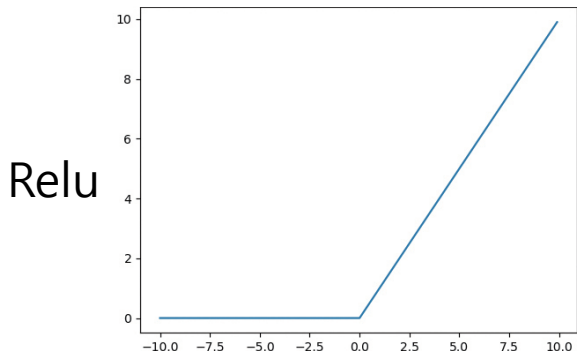
다중 퍼셉트론 (Multi Layer Perceptron: MLP)

인하공전 컴퓨터 정보과

입력 (1,0)



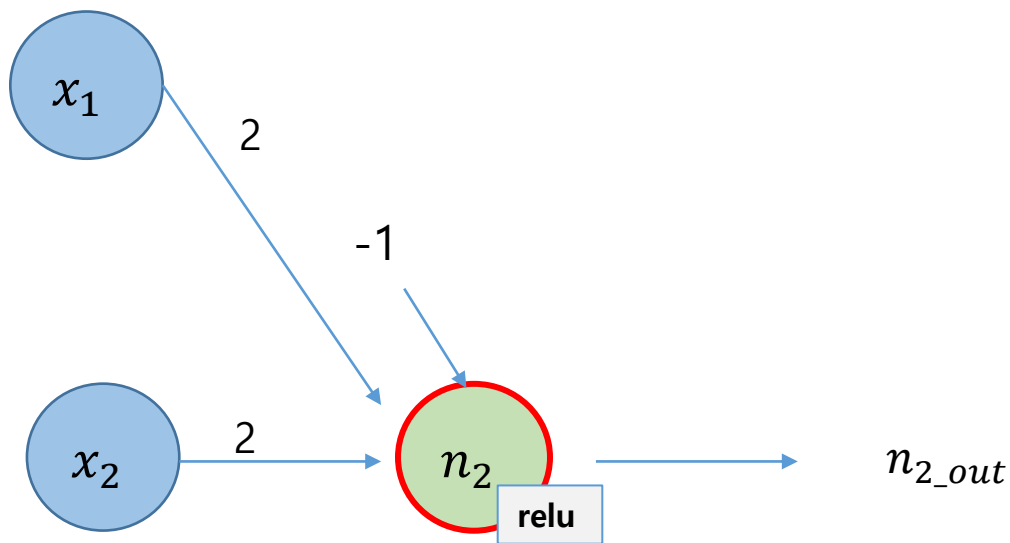
$$f(x) = \begin{cases} x & x > 0 \\ 0 & x \leq 0 \end{cases}$$



참고자료 : 딥러닝 express

다음 다중 퍼셉트론의 출력은?

입력 (1,0)



$$w_{21} \cdot x_1 + w_{22} \cdot x_2 + b_2$$

$$\text{가중합: } 1 \cdot (2) + 2 \cdot (0) - 1 = 1$$

$$\text{Relu}(1) = 1$$

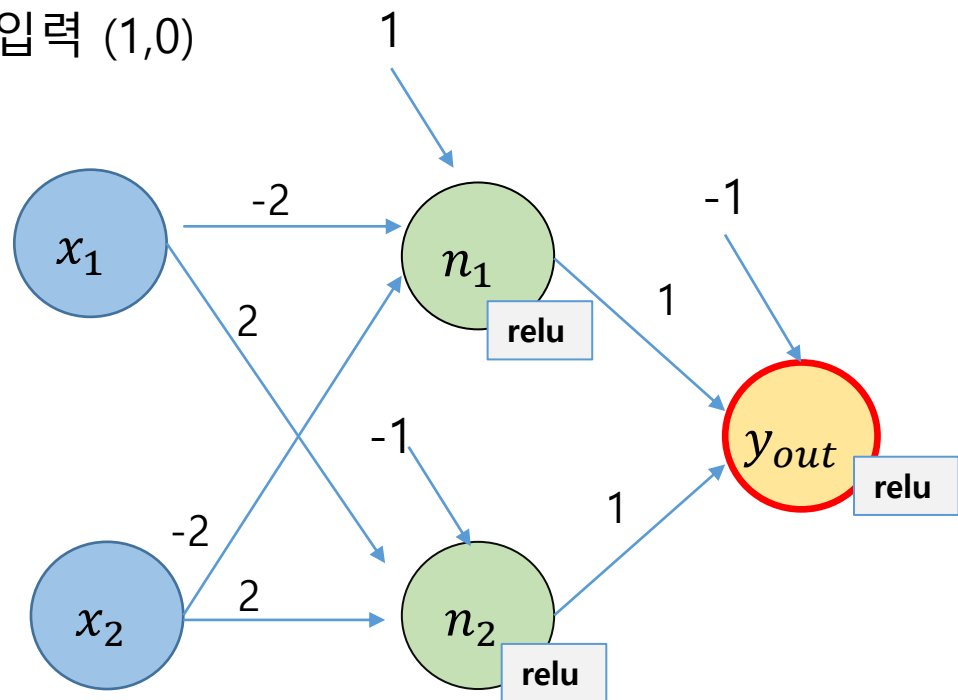
$$n_{2_out} = 1$$

다중 퍼셉트론 (Multi Layer Perceptron: MLP)

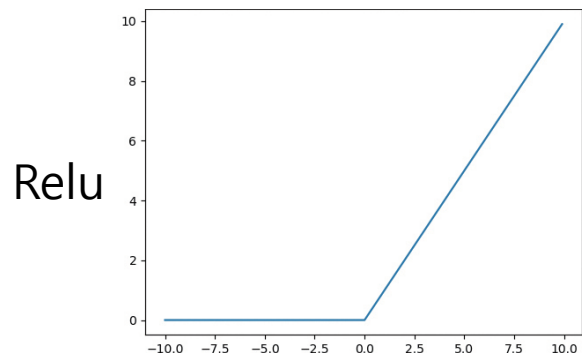
인하공전 컴퓨터 정보과

입력 (1,0)

다음 다중 퍼셉트론의 출력은?



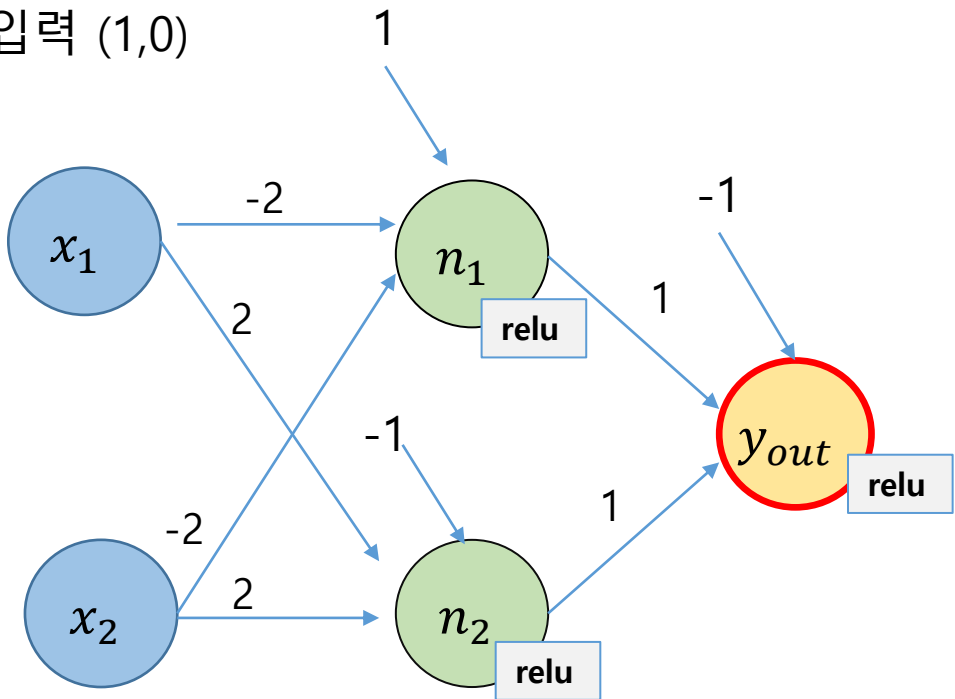
$$f(x) = \begin{cases} x & x > 0 \\ 0 & x \leq 0 \end{cases}$$



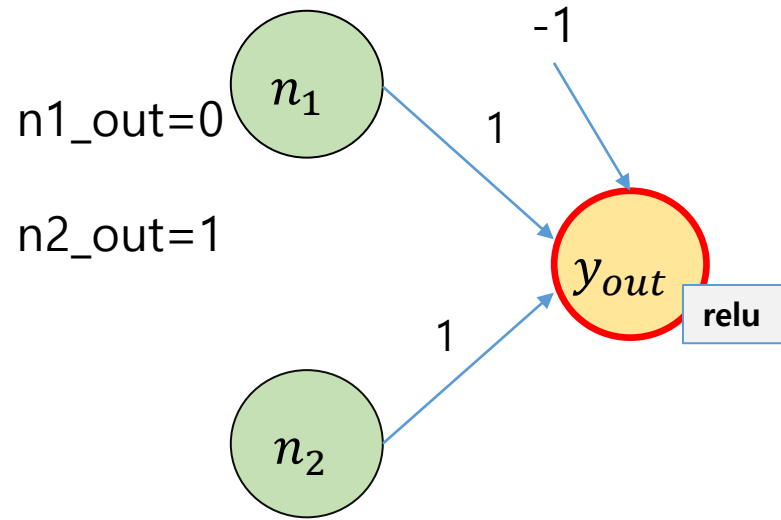
다중 퍼셉트론 (Multi Layer Perceptron: MLP)

인하공전 컴퓨터 정보과

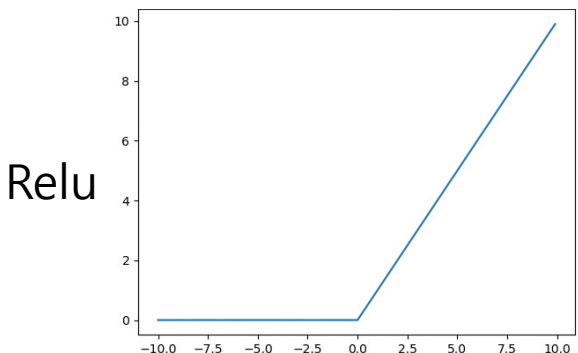
입력 (1,0)



다음 다중 퍼셉트론의 출력은?



$$f(x) = \begin{cases} x & x > 0 \\ 0 & x \leq 0 \end{cases}$$



$$w31 \cdot n1_{out} + w32 \cdot n2_{out} + b3$$

가중합: $1 \cdot (0) + 1 \cdot (1) - 1 = 0$

$\text{Relu}(0) = 0$

$Y_{out} = 0$

■ 순방향 패스

- 순방향 패스란 입력 신호가 입력층 유닛에 가해지고 이들 입력 신호가 은닉층을 통하여 출력층으로 전파되는 과정을 의미한다.

역방향 패스 (오차 역전파)

신경망 내부의 가중치는 오차 역전파 방법을 사용해 수정함

■ 경사하강법

- 손실 함수를 입력 변수를 기준으로 미분하여 w , b 를 업데이트
- 반복 하면서 오차를 최소화 하는 방향으로 w 와 b 를 업데이트

```
iteration 0: loss 17.17 w 0.10 b 0.08
iteration 50: loss 0.61 w 1.73 b 1.71
iteration 100: loss 0.33 w 1.73 b 2.05
iteration 150: loss 0.24 w 1.63 b 2.23
iteration 200: loss 0.19 w 1.53 b 2.36
iteration 250: loss 0.16 w 1.47 b 2.45
iteration 300: loss 0.15 w 1.41 b 2.52
iteration 350: loss 0.14 w 1.37 b 2.58
iteration 400: loss 0.13 w 1.34 b 2.62
iteration 450: loss 0.13 w 1.32 b 2.65
##### final w,b 1.3033228991130752
2.6760184293088694
```

1.25 2.74

- 오차 역전파(back propagation) :

다층 퍼셉트론에서의 최적화 과정

- 오차 역전파 구동 방식은 다음과 같이 정리할 수 있음

1 | 임의의 초기 가중치(w)를 준 뒤 결과(y_{out})를 계산함

2 | 계산 결과와 우리가 원하는 값 사이의 오차를 구함

3 | 경사 하강법을 이용해 바로 앞 가중치를 오차가 작아지는 방향으로 업데이트함

4 | 위 과정을 더이상 오차가 줄어들지 않을 때까지 반복함

동영상

- 역전파 알고리즘은 입력이 주어지면 순방향으로 계산하여 출력을 계산한 후에 실제 출력과 우리가 원하는 출력 간의 오차를 계산한다. 이 오차를 역방향으로 전파하면서 오차를 줄이는 방향으로 가중치를 변경한다.

- 신경망에서 학습을 시킬 때 는 실제 출력과 원하는 출력 사이의 오차를 이용한다.
- 오차를 계산하는 함수를 손실함수(loss function)라고 한다.

평균 제곱 오차(MSE)

인하공전 컴퓨터 정보과

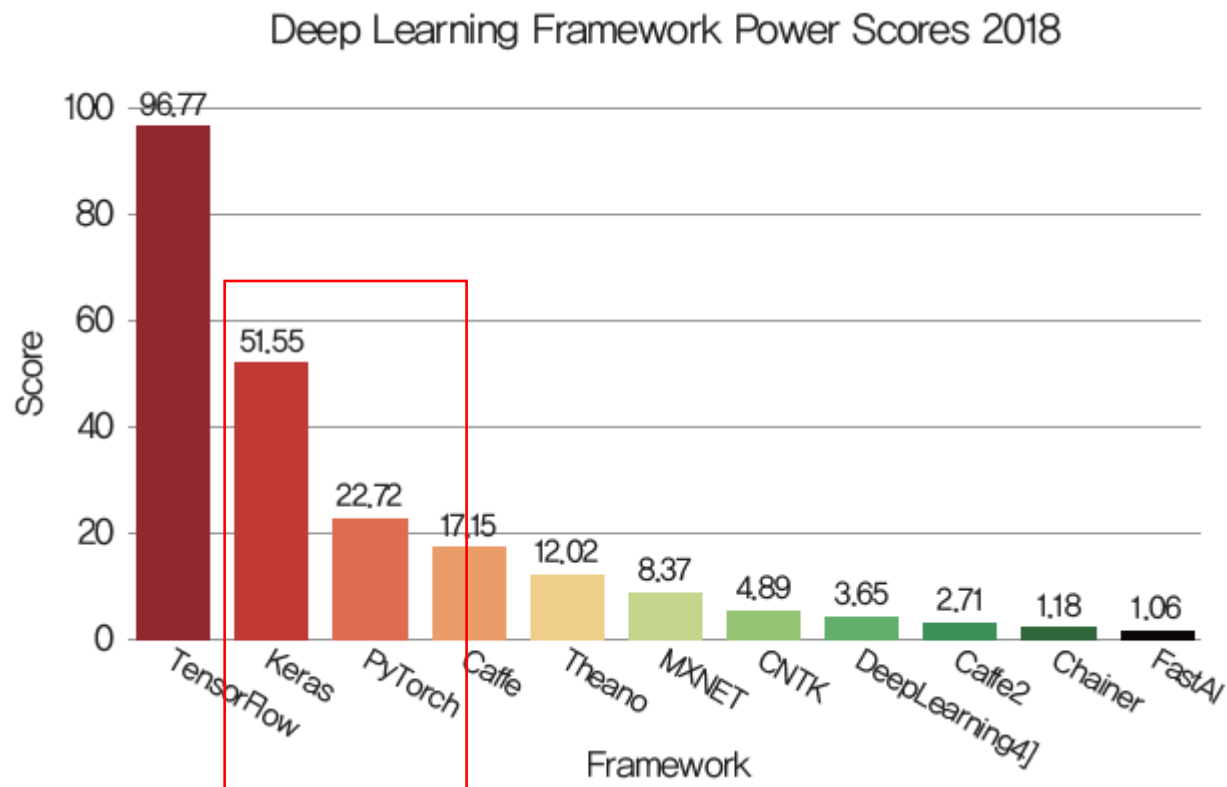
- 예측값과 정답 간의 평균 제곱 오차

- **심층 신경망 (Deep Neural Network)을 사용하는 학습 방법**
- 심층 신경망(DNN: Deep Neural Networks)은 MLP(다층 퍼셉트론)에서 은닉층의 개수를 증가시킨 것이다.
- 은닉층을 하나만 사용하는 것이 아니고 여러 개를 사용한다.
- 최근에 딥러닝은 컴퓨터 시각, 음성 인식, 자연어 처리, 소셜 네트워크 필터링, 기계 번역 등에 적용되어서 인간 전문가에 필적하는 결과를 얻고 있다.

딥러닝 (Deep learning) 실습-keras 라이브러리

인하공전 컴퓨터 정보과
인하공전 컴퓨터 정보과

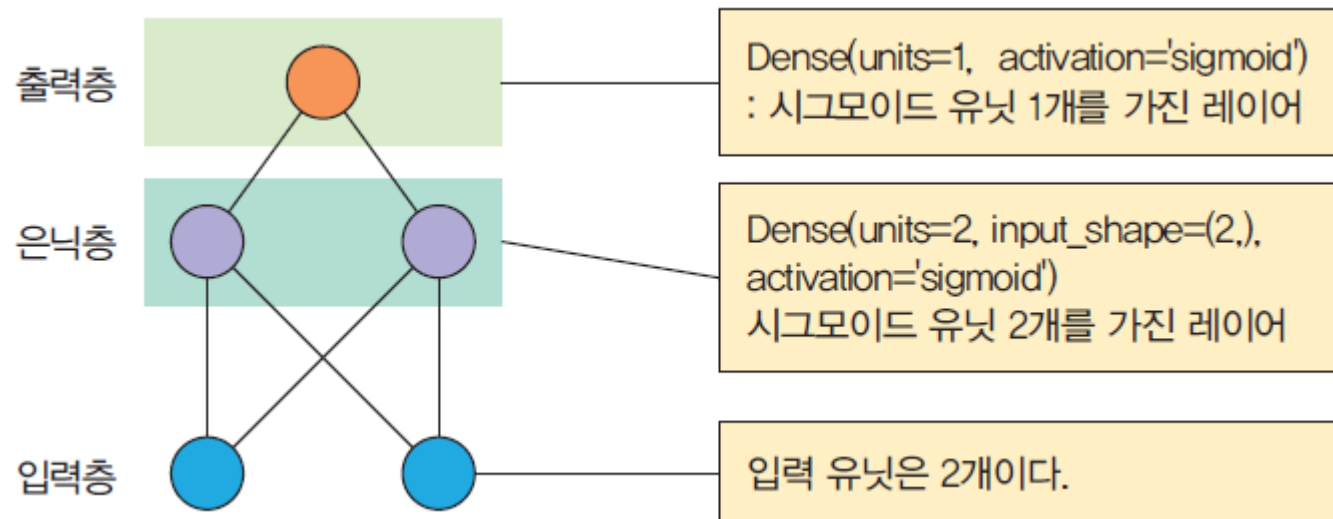
- 케라스로 모델 설계 하기 (keras 라이브러리 이용)
- Tensorflow + keras 사용
- `Import tensorflow.keras`



- 케라스의 핵심 데이터 구조
 - 모델(model): 레이어를 구성하는 방법을 나타낸다.
- 가장 간단한 모델 유형은 Sequential 선형 스택 모델이다. Sequential 모델은 레이어를 선형으로 쌓을 수 있는 신경망 모델이다

케라스로 신경망을 작성

인하공전 컴퓨터 정보과



케라스로 신경망을 작성 하는 절차

```
model=keras.Sequential()
```

```
model.add(layers.Dense(units=2,activation='sigmoid',input_shape=(2,)))
```

```
model.add(layers.Dense(units=1,activation='sigmoid'))
```

```
model.compile(optimizer=keras.optimizers.SGD(learning_rate=0.7),loss='mse')
```

```
model.fit(X,y,epochs=300,batch_size=1)
```

```
model.summary()
```

```
print(model.predict(X))
```


Sequential model 을 생성

```
model=keras.Sequential()
```

Layer 추가

```
model.add(layers.Dense(units=2,activation='sigmoid',input_shape=(2,)))
```

```
model.add(layers.Dense(units=1,activation='sigmoid'))
```

학습시 option 들 결정

```
model.compile(optimizer=keras.optimizers.SGD(learning_rate=0.7),loss='mse')
```

```
model.fit(X,y,epochs=300,batch_size=1)
```

학습

```
model.summary()
```

```
print(model.predict(X))
```

예측

OR 연산을 수행하는 신경망 실습

인하공전 컴퓨터 정보과

```
from tensorflow import keras
from keras import layers
import numpy as np
```

```
X=np.array([[1,1],[1,0],[0,1],[0,0]])
y=np.array([[1],[1],[1],[0]])
```

```
model=keras.Sequential()
```

```
model.add(layers.Dense(units=2,activation='sigmoid',input_shape=(2,)))
model.add(layers.Dense(units=1,activation='sigmoid'))
```

```
model.compile(optimizer=keras.optimizers.SGD(learning_rate=0.7),loss='mse')
model.fit(X,y,epochs=300,batch_size=1)
```

```
model.summary()
```

```
print(model.predict(X))
```

OR 연산을 수행하는 신경망 실습

Model: "sequential_9"

Layer (type)	Output Shape	Param #
dense_18 (Dense)	(None, 2)	6
dense_19 (Dense)	(None, 1)	3

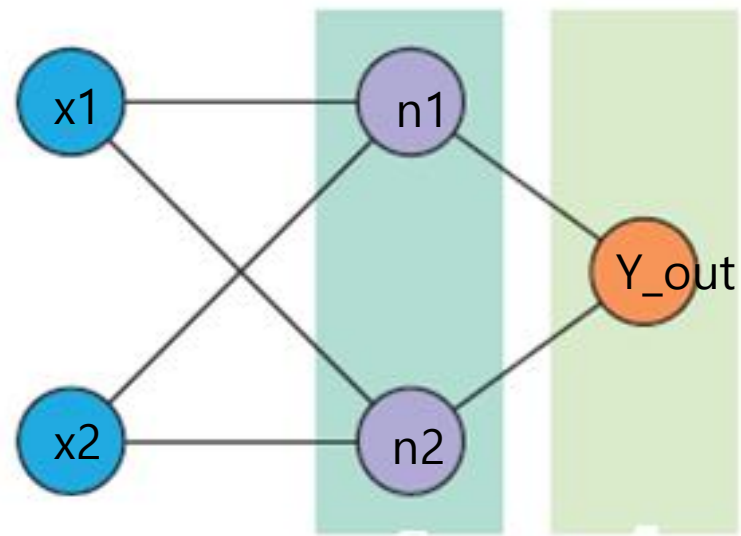
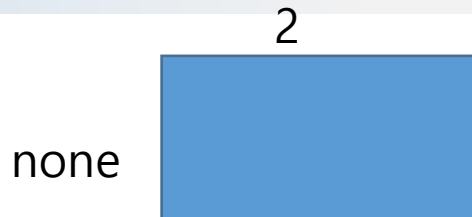
Total params: 9

Trainable params: 9

Non-trainable params: 0

```
[[0.10639304]
 [0.9426358 ]
 [0.9383687 ]
 [0.97790086]]
```


Output shape



XOR를 학습하는 MLP keras 구현

인하공전 컴퓨터 정보과

앞의 OR 연산을 수행하는 신경망 코드를 이용하여
XOR 연산을 수행 하도록 학습 시키고 예측 값을 제출

	x1	x2		y
샘플 #1	0	0		0
샘플 #2	0	1		1
샘플 #3	1	0		1
샘플 #4	1	1		0

과제 1)

▪ 96 page의 code를 이용하여 다음 수행 결과를 입력 하고 code를 upload (*.py) 하시오.

- 1) OR 연산 학습, epoch= 300, 예측 결과
- 2) OR 연산 학습, epoch= 1000, 예측 결과
- 3) XOR 연산 학습, epoch= 300, 예측 결과
- 4) XOR 연산 학습, epoch= 1000, 예측 결과

수고하셨습니다

jhmin@inhatec.ac.kr