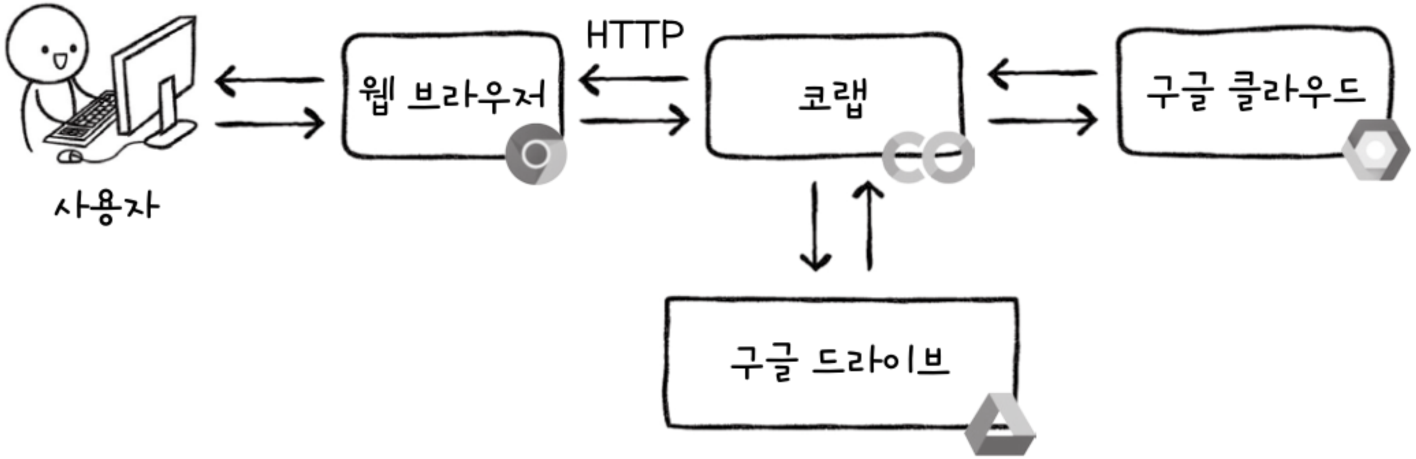
00

Google Colabratory

구글 코랩

구글 코랩

- 구글에서 교육과 과학연구를 목적으로 개발한 도구로 2017년 무료 공개
- 파이썬 코드를 실행하거나 텍스트를 저장 할 수 있고 그래프를 그릴 수 있음
- 온라인상에서 빅데이터 처리 및 인공지능 코드 처리 : 개인PC보다 속도가 빠름

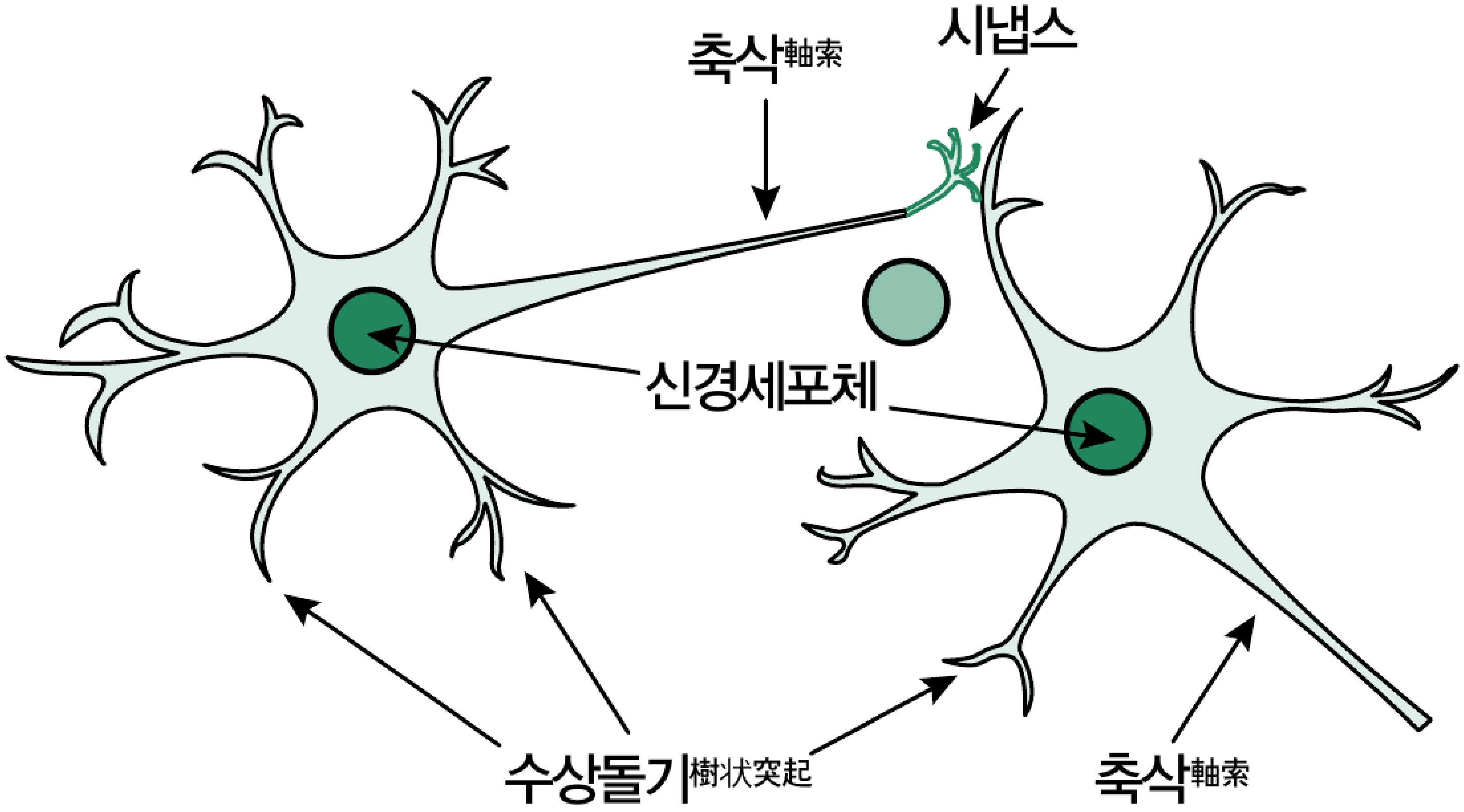


웹브라우저를 통해 제어하고 구글 클라우드에서 실행

- 구글 드라이브에 저장되고 불러올 수 있음
- 구글 계정 필요

01
딥러닝 시작하기

뉴런



뉴런 : 주로 신경세포체, 축삭, 수상돌기로 구성됨

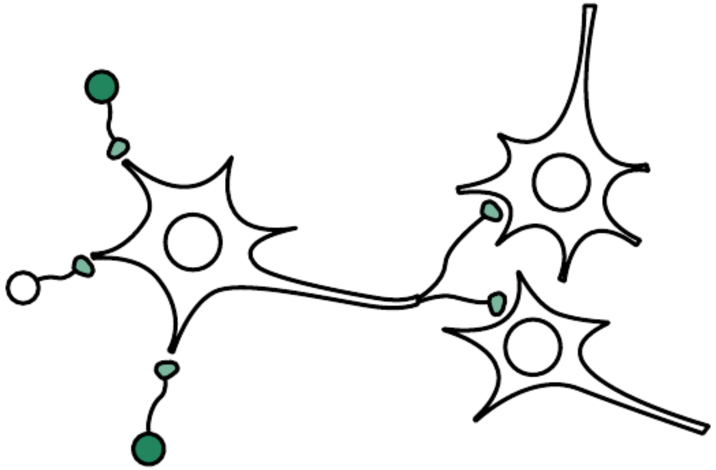
수상돌기 : 다른뉴런에게 정보를 받는 돌기

축삭 : 다른 뉴런에게 정보를 발송하는 돌기

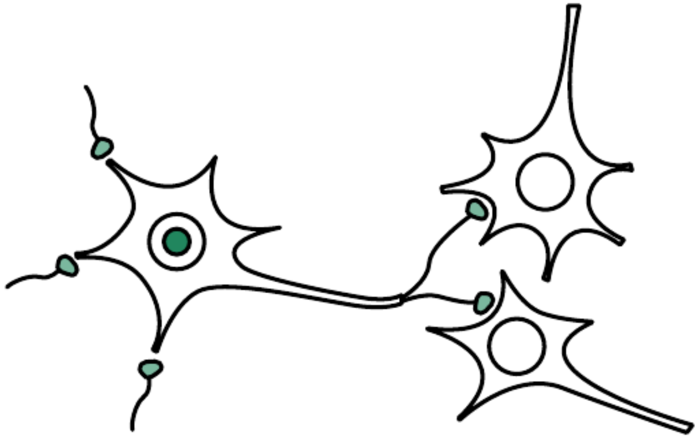
수상돌기가 받은 전기신호는 신경세포체에서 처리한 후 축삭을 지나 다음 뉴런으로 전달

뉴런은 시냅스를 매개로 결합해 네트워크를 형성

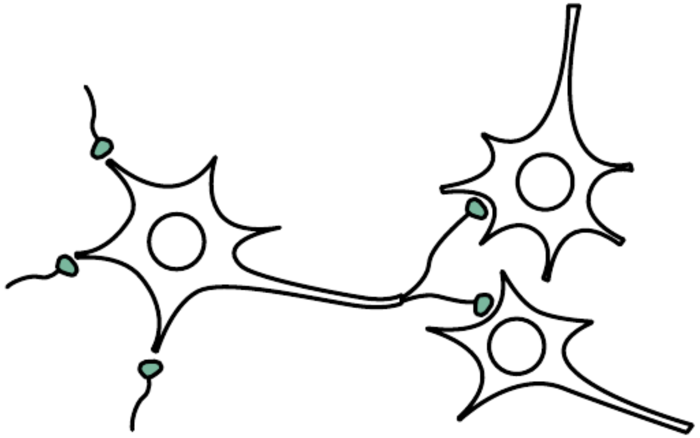
뉴런



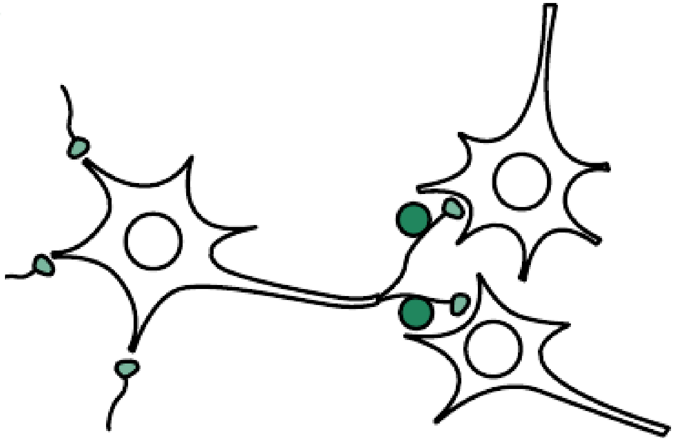
뉴런에 신호가 입력 됨



신경세포체는 신호의 합함



신호의 합이 임계값보다 작으면 무시

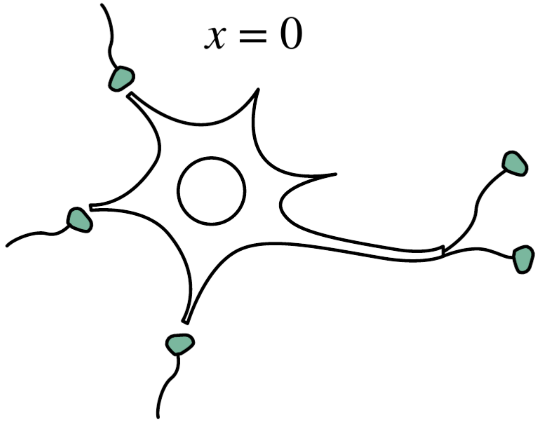


신호의 합이 임계값보다 크면 뉴런이
반응해 근처 뉴런에 신호 전달

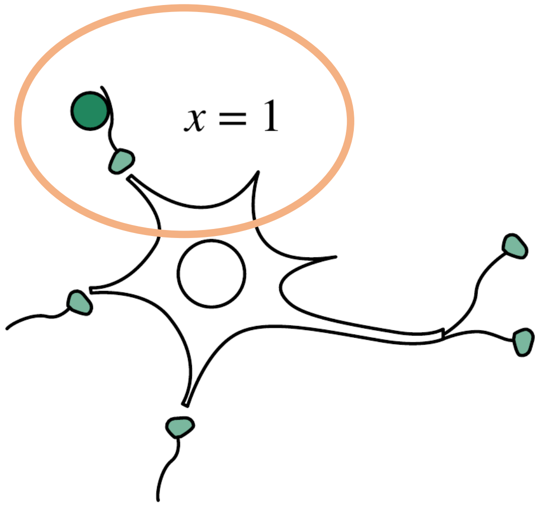
뉴런의 수학적 표현

- **다른 여러 뉴런의 '신호합'이 뉴런의 입력**
- **'신호합' 이 뉴런의 임계값보다 크면 반응**
- **뉴런의 출력 신호는 반응 유무에 따라 0과 1로 표현**
  - 복수의 출력 신호가 있더라도 반응 여부를 0과 1로 표현

입력

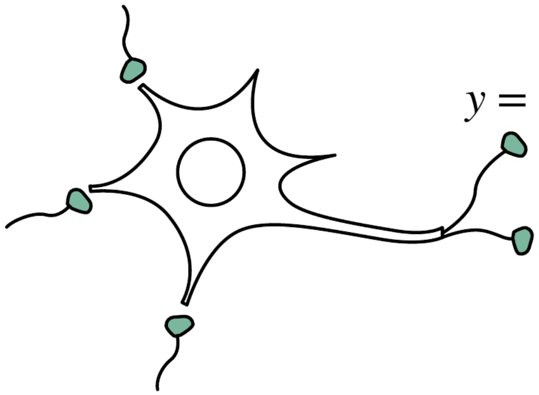


입력 신호 없음 x=0

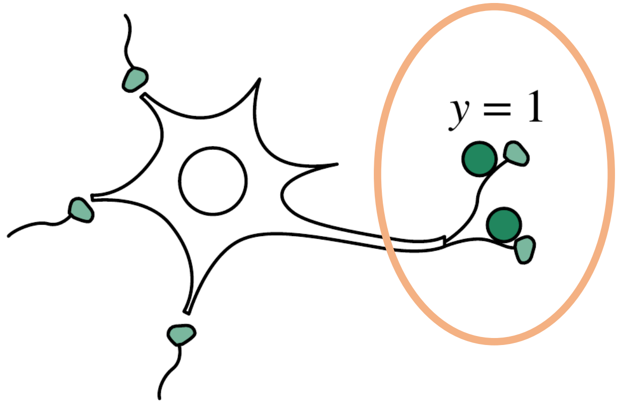


입력신호 있음 x=1

출력

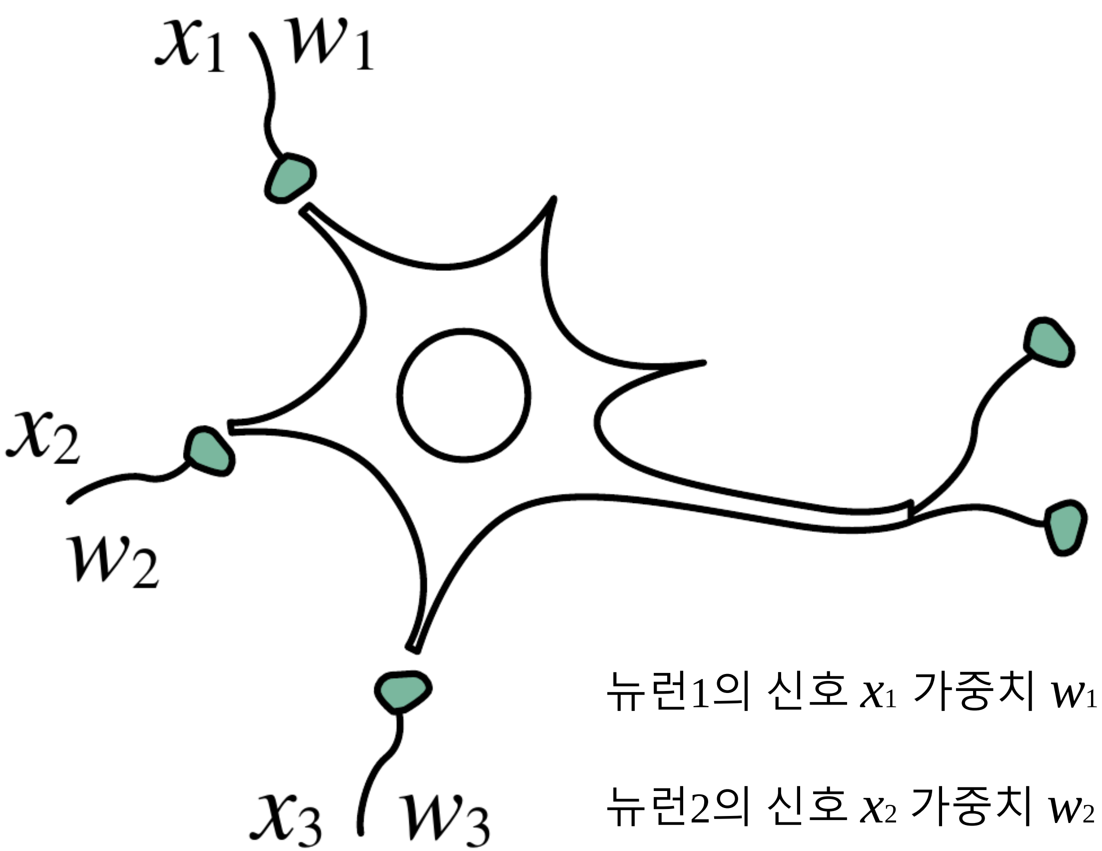


**출력 신호 없음 y=0
(반응 없음)**



**출력 신호 있음 y=1
(반응 있음)**

뉴런의 수학적 표현



뉴런1의 신호 x_1 가중치 w_1

뉴런2의 신호 x_2 가중치 w_2

뉴런3의 신호 x_3 가중치 w_3

뉴런의 반응 여부 수학적 표현

출력 신호 = $w_1x_1 + w_2x_2 + w_3x_3$

뉴런 : 신호의 합이 임계값 보다 크면 반응
작으면 반응하지 않음

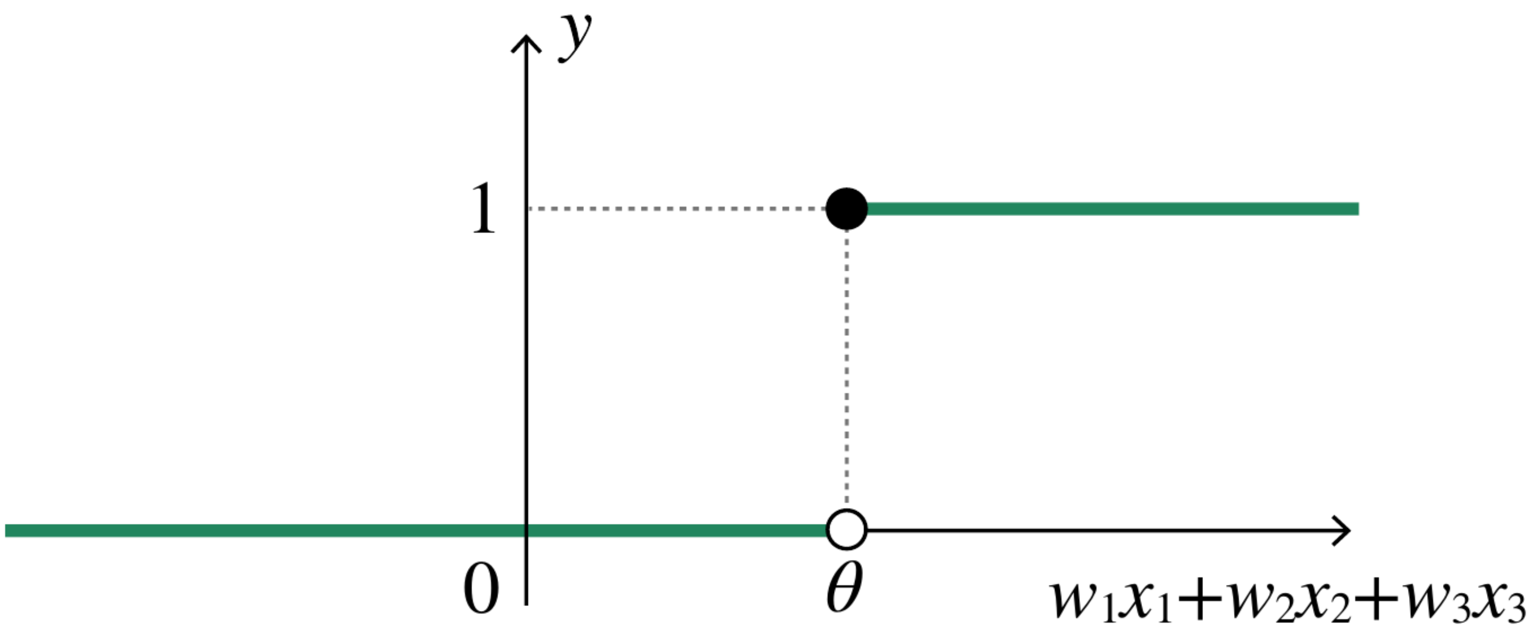
$$\begin{cases} y = 0 & w_1x_1 + w_2x_2 + w_3x_3 < \theta \\ y = 1 & w_1x_1 + w_2x_2 + w_3x_3 \geq \theta \end{cases}$$

θ 는 뉴런의 임계값

반응 조건

- 뉴런 : 신호의 합이 임계값 보다 크면 반응 작으면 반응하지 않음
- x축을 뉴런의 신호 합($w_1 x_1 + w_2 x_2 + w_3 x_3$) 이라고 가정

$$y = \begin{cases} 0, & w_1 x_1 + w_2 x_2 + w_3 x_3 < \theta \\ 1, & w_1 x_1 + w_2 x_2 + w_3 x_3 \geq \theta \end{cases}$$



반응조건

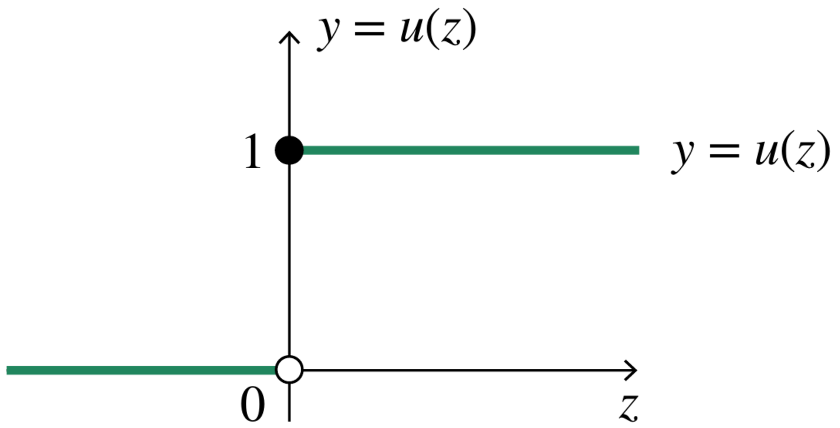
$$z = w_1x_1 + w_2x_2 + w_3x_3 - \theta$$

$$\begin{matrix} w_1x_1 + w_2x_2 + w_3x_3 < \theta \\ w_1x_1 + w_2x_2 + w_3x_3 \geq \theta \end{matrix} \Rightarrow \begin{matrix} w_1x_1 + w_2x_2 + w_3x_3 - \theta < 0 \\ w_1x_1 + w_2x_2 + w_3x_3 - \theta \geq 0 \end{matrix} \Rightarrow \begin{matrix} z < 0 \\ z \geq 0 \end{matrix} \Rightarrow u(z)$$

y	$w_1x_1+w_2x_2+w_3x_3$	$z = w_1x_1+w_2x_2+w_3x_3-\theta$	$u(z)$
0(반응하지 않음)	θ보다 작음	$z < 0$	0
1(반응함)	θ보다 큼	$z \geqq 0$	1

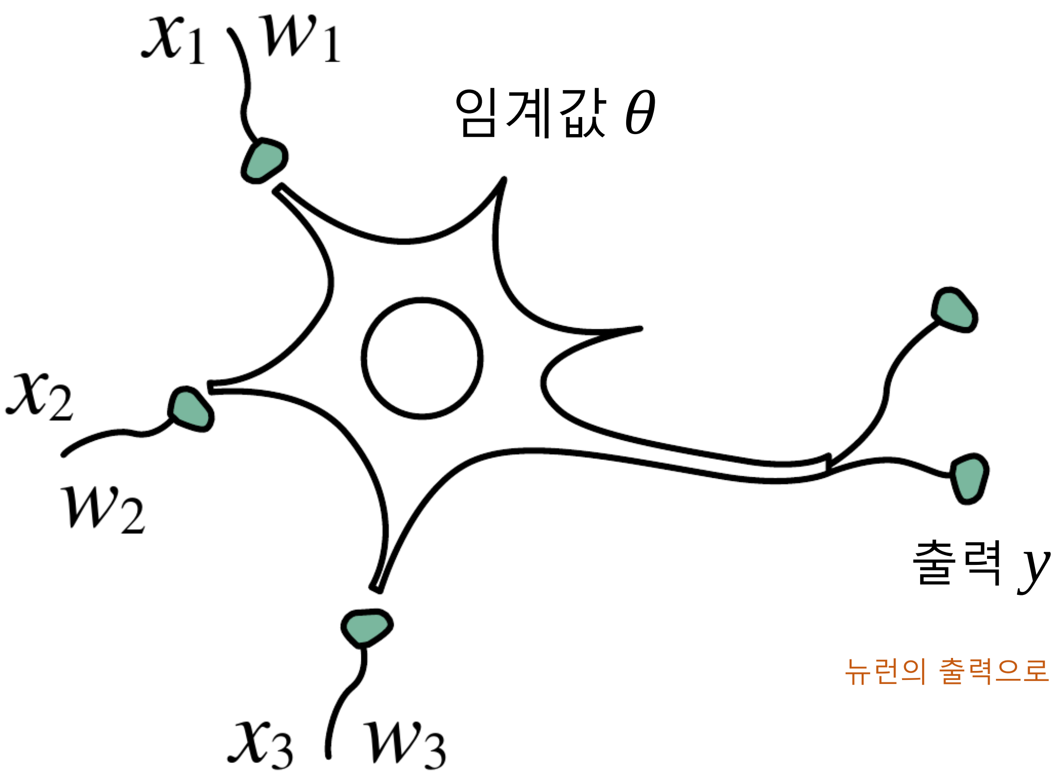
$$u(z) = \begin{cases} 0, & z < 0 \\ 1, & z \geq 0 \end{cases}$$

$$z = w_1x_1 + w_2x_2 + w_3x_3 - \theta$$

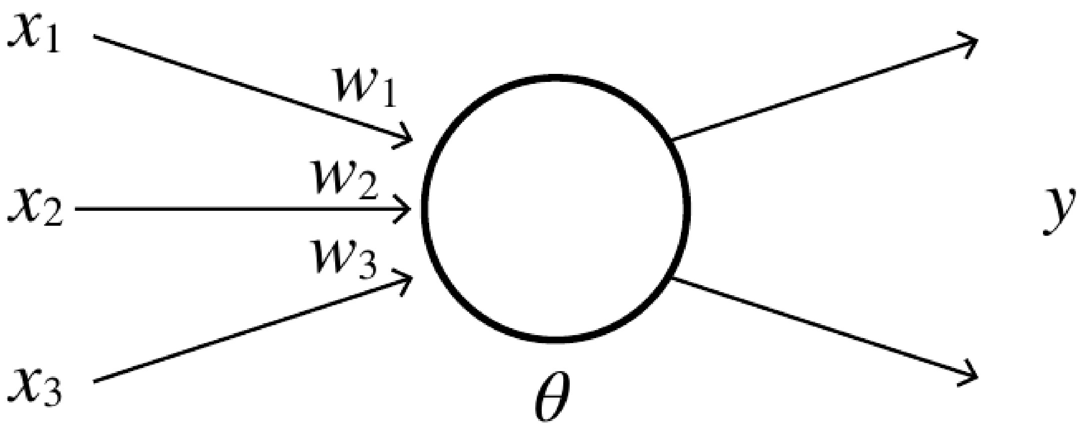


뉴런의 단순화

뉴런의 이미지(입력3개, 출력2개)



뉴런의 단순화(입력3개, 출력2개)



뉴런의 출력으로 2개의 화살표가 있더라도 출력은 y로 나타냄

뉴런1의 신호 x_1 가중치 w_1

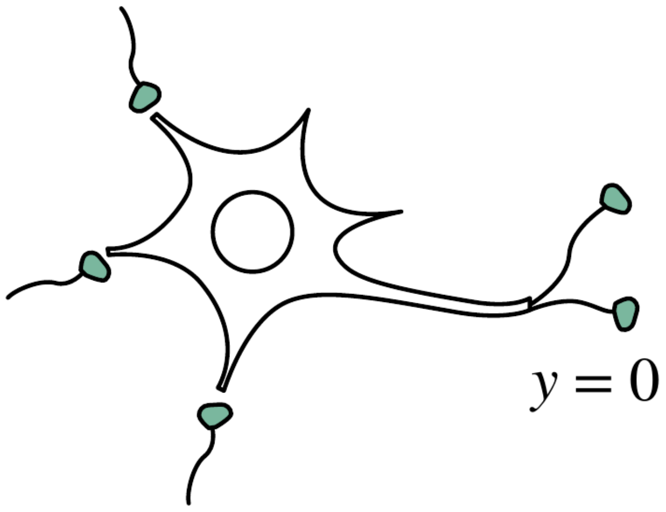
뉴런2의 신호 x_2 가중치 w_2 임계값 θ 출력 y

뉴런3의 신호 x_3 가중치 w_3

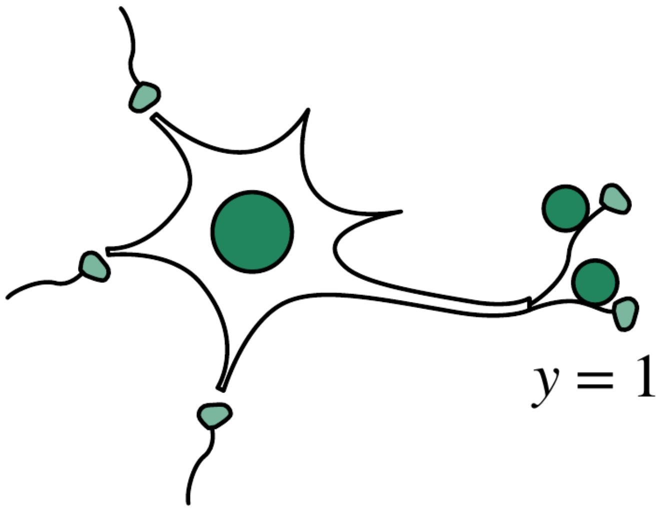
활성화 함수(Activation Function)

- **출력신호의 수학적 일반화**
  - 반응 여부(y)는 0과 1중 하나로 표현
  - '생물학적' 조건을 두지 않으면 0과 1로 표현하지 않아도 됨
  - 계단함수 u를 특정값에 영향을 받지 않는 함수 a로 일반화
- **활성화 함수(activation function)**
  - 함수 a : 사용자가 정의하는 함수 함수 a

$$y = a(w_1x_1 + w_2x_2 + w_3x_3 - \theta)$$

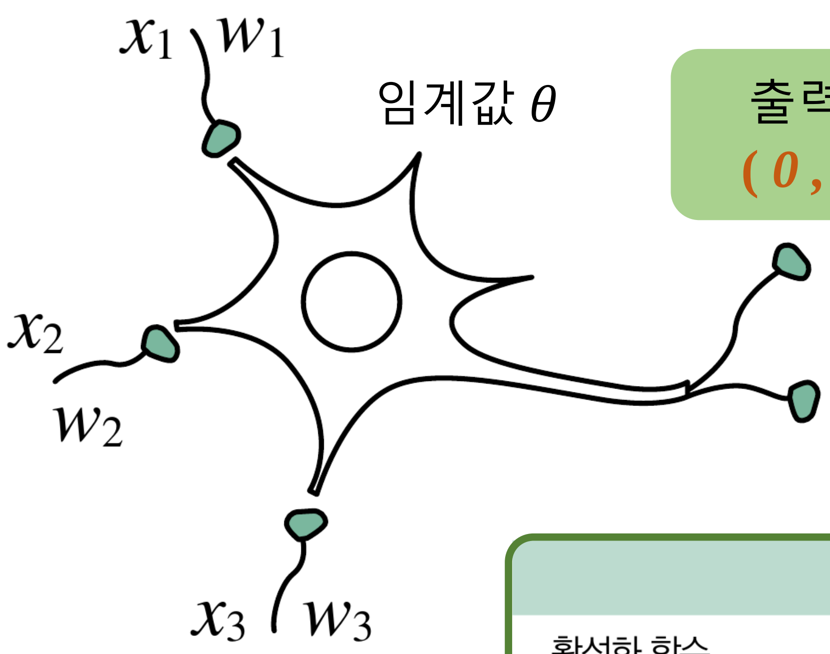


출력신호 없음 ($y = 0$)

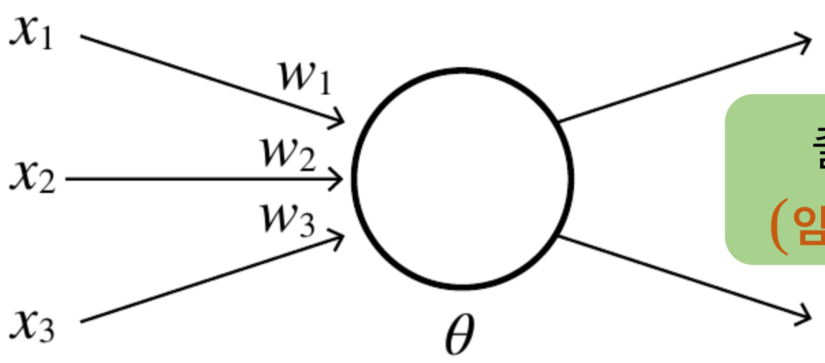


출력신호 있음 ($y = 1$)

활성화 함수(Activation Function)

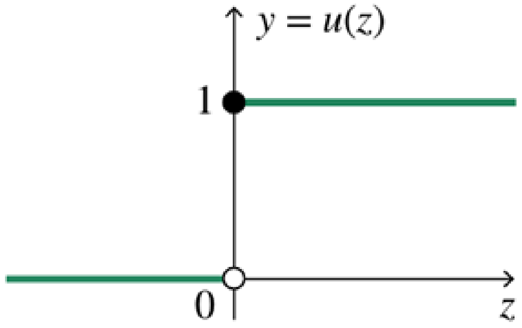


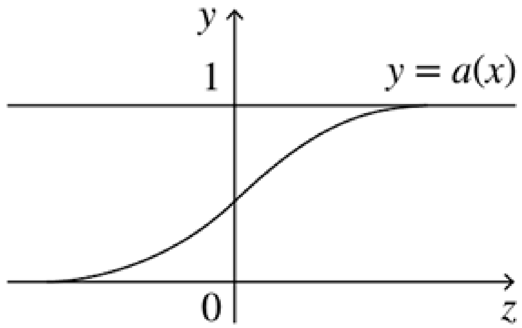
출력 y
$(0, 1)$



출력 y
(임의의 수)

	뉴런	유닛
활성화 함수	단위 계단 함수	사용자 정의 함수 (시그모이드 함수)
출력값 y	0 또는 1	활성화 함수를 사용할 수 있는 임의의 수
출력 신호 해석	반응 여부	유닛의 흥분도, 반응도, 활성도

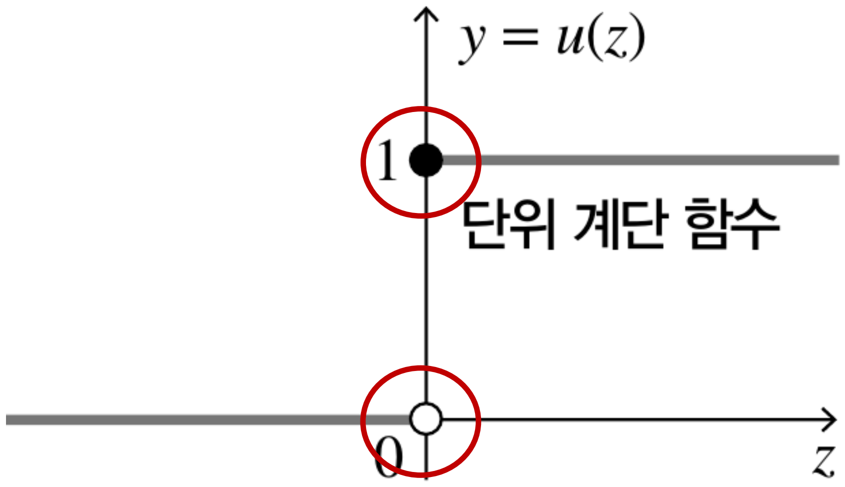


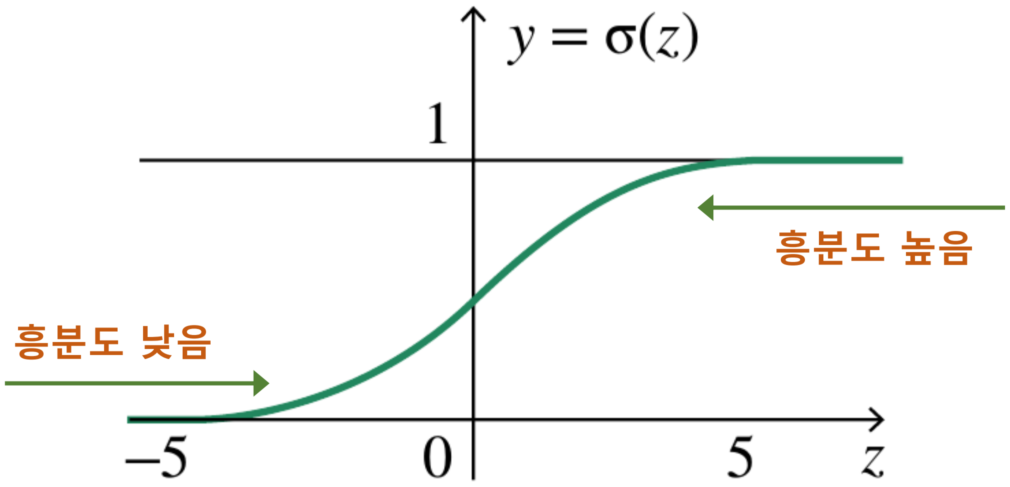


시그모이드(sigmoid)함수

- **시그모이드 함수($\sigma(z)$)**
  - 대표적인 활성화 함수
  - 출력은 0보다 크고 1보다 작은 임의의 값
  - 미분 가능 함수
  - 유닛의 흥분도 or 반응도를 나타냄

$$\sigma(z) = \frac{1}{1+e^{-z}} \quad (e = 2.718281\cdots)$$

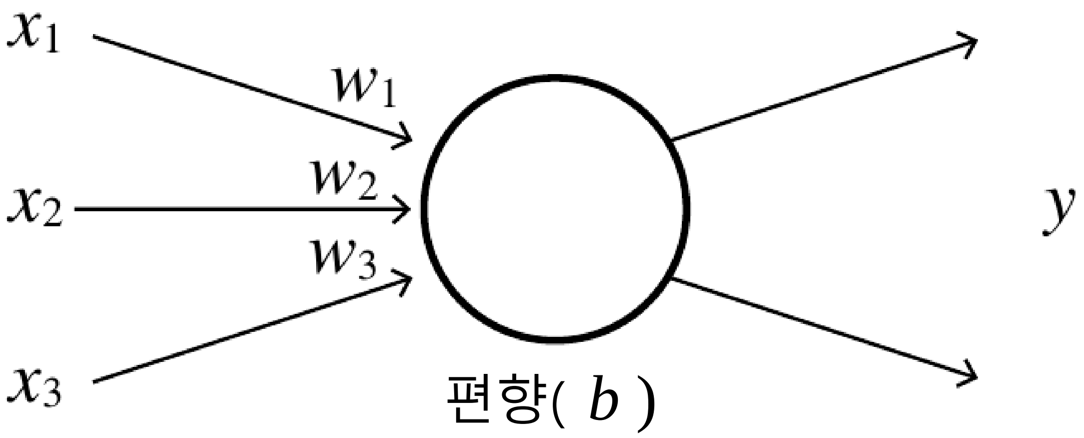




편향(bias)

- -θ를 b로 치환
  - 활성화 함수에서 θ는 뉴런의 출력 여부를 결정하는 임계값
  - θ의 크기에 따라 민감도 결정 (θ의 크기가 작으면 흥분도 높음(민감), 크면 흥분도 낮음(둔감))
  - θ 앞에 기호가 있으면 실수하기 쉬움 : -θ를 b로 치환

$$y = \begin{cases} 0, & w_1 x_1 + w_2 x_2 + w_3 x_3 < \theta \\ 1, & w_1 x_1 + w_2 x_2 + w_3 x_3 \geq \theta \end{cases}$$

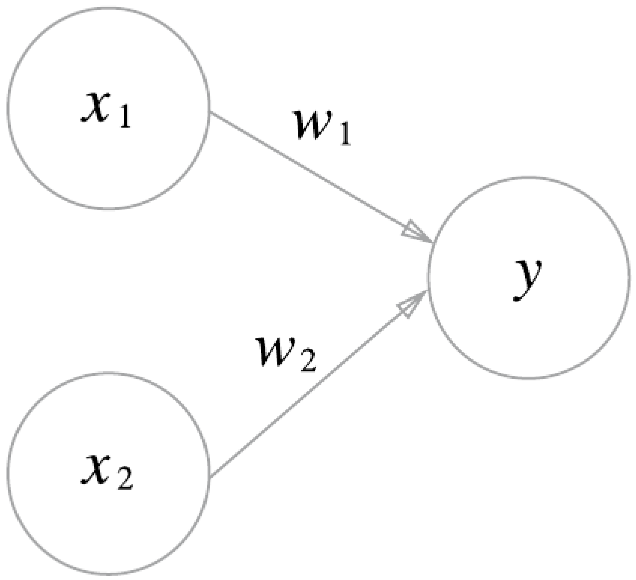


$$y = a(w_1 x_1 + w_2 x_2 + w_3 x_3 - \theta)$$

$$y = a(w_1 x_1 + w_2 x_2 + w_3 x_3 + b)$$

퍼셉트론(Perceptron)

- **신경망의 기원이 되는 알고리즘**
  - 프랑크 로젠블라트(Frank Rosenblatt)가 1957년 고안
- **다수의 신호를 입력으로 받아 하나의 신호를 출력**
  - 신호 : 흐름이 있는 어떤 것(전류, 강물 등)
  - 흐름을 만들고 정보를 앞으로 전달
  - 신호가 흐른다(1) / 신호가 흐르지 않는다(0)의 2가지 값을 가짐



입력이 2개인 퍼셉트론

$$y = \begin{cases} 0, & w_1 x_1 + w_2 x_2 < \theta \\ 1, & w_1 x_1 + w_2 x_2 \geq \theta \end{cases}$$

논리회로

AND 게이트

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

OR 게이트

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

NAND 게이트

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

퍼셉트론 - AND 게이트

- **AND게이트를 퍼셉트론으로 표현**
  - w_1, w_2 , θ의 값 정하기

$$y = \begin{cases} 0, & w_1 x_1 + w_2 x_2 < \theta \\ 1, & w_1 x_1 + w_2 x_2 \geq \theta \end{cases}$$

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

(w_1, w_2, θ)

예) : (0.5, 0.5, 0.7)
(0.5, 0.5, 0.8)
(1.0, 1.0, 1.0)

x_1	x_2	$w_1 x_1 + w_2 x_2$
0	0	$0.5 \cdot 0 + 0.5 \cdot 0 = 0 < 0.7$ (True)
0	1	$0.5 \cdot 0 + 0.5 \cdot 1 = 0.5 < 0.7$ (True)
1	0	$0.5 \cdot 1 + 0.5 \cdot 0 = 0.5 < 0.7$ (True)
1	1	$0.5 \cdot 1 + 0.5 \cdot 1 = 1 \geq 0.7$ (True)

AND게이트 구현

$$y = \begin{cases} 0, & w_1 x_1 + w_2 x_2 < \theta \\ 1, & w_1 x_1 + w_2 x_2 \ge \theta \end{cases}$$

$(w_1, w_2, \theta) = (0.5, 0.5, 0.7)$

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

print('AND(0,0):',AND(0,0))
print('AND(1,0):',AND(1,0))
print('AND(0,1):',AND(0,1))
print('AND(1,1):',AND(1,1))
```

NAND게이트를 퍼셉트론으로 표현

- w_1, w_2 , θ의 값 정하기

$$y = \begin{cases} 0, & w_1 x_1 + w_2 x_2 < \theta \\ 1, & w_1 x_1 + w_2 x_2 \geq \theta \end{cases}$$

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

$(w_1,\ w_2,\ \theta)$

예) : (-0.5, -0.5, -0.7)

x_1	x_2	$w_1 x_1 + w_2 x_2$
0	0	$-0.5 \cdot 0 + \ -0.5 \cdot 0 = 0 \geq -0.7$ (True)
0	1	$-0.5 \cdot 1 + \ -0.5 \cdot 0 = -0.5 \geq -0.7$ (True)
1	0	$-0.5 \cdot 0 + \ -0.5 \cdot 1 = -0.5 \geq -0.7$ (True)
1	1	$-0.5 \cdot 1 + \ -0.5 \cdot 1 = -1 < -0.7$ (True)

퍼셉트론 - NAND 게이트 구현

- **NAND게이트 구현**

$$y = \begin{cases} 0, & w_1 x_1 + w_2 x_2 < \theta \\ 1, & w_1 x_1 + w_2 x_2 \geq \theta \end{cases}$$

(w_1, w_2, θ) = (-0.5, -0.5, -0.7)

```python
def NAND(x1, x2):
```

NAND 코드 구현해 볼 것

```python
print('NAND(0,0):', NAND(0,0))
print('NAND(1,0):', NAND(1,0))
print('NAND(0,1):', NAND(0,1))
print('NAND(1,1):', NAND(1,1))
```

OR게이트를 퍼셉트론으로 표현

- $w_1,\ w_2\ ,\ \theta$의 값 정하기

$$y = \begin{cases} 0, & w_1 x_1 + w_2 x_2 < \theta \\ 1, & w_1 x_1 + w_2 x_2 \geq \theta \end{cases}$$

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

- OR게이트를 만족하는 $(w_1,\ w_2,\ \theta)$ 구하기

x_1	x_2	$w_1 x_1 + w_2 x_2$
0	0	
0	1	
1	0	
1	1	

퍼셉트론 - OR 게이트 구현

■ **OR게이트 구현**

$$y = \begin{cases} 0, & w_1 x_1 + w_2 x_2 < \theta \\ 1, & w_1 x_1 + w_2 x_2 \geq \theta \end{cases}$$

$(w_1,\ w_2,\ \theta)$

```
def OR(x1, x2):
```

OR 코드 구현해 볼 것

```
print('OR(0,0):', OR(0,0))
print('OR(1,0):', OR(1,0))
print('OR(0,1):', OR(0,1))
print('OR(1,1):', OR(1,1))
```

가중치와 편향 도입 예제

■ 가중치와 편향의 도입

$$y = \begin{cases} 0, & w_1 x_1 + w_2 x_2 + b < 0 \\ 1, & w_1 x_1 + w_2 x_2 + b \geq 0 \end{cases}$$

$$(w_1,\ w_2,\ \theta) = (0.5,\ 0.5,\ 0.7)$$

```
import numpy as np

x=np.array([[0,0],[1,0],[0,1],[1,1]])
w=np.array([0.5,0.5])
b=-0.7

# w1x1 + w2x2 + b
for i in range(4):
    result = np.sum(w*x[i]) + b
    y_value = 1 if result>0 else 0
    print("{:.2f} {}".format(result, y_value))
```

가중치와 편향 도입 예제

■ **AND 게이트 구현**

$$y = \begin{cases} 0, & w_1x_1 + w_2x_2 + b < 0 \\ 1, & w_1x_1 + w_2x_2 + b \geq 0 \end{cases}$$

$$(w_1,\ w_2,\ \theta) = (\ 0.5,\ 0.5,\ 0.7\ )$$

$$-\theta = b$$

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

```python
import numpy as np

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

print('AND(0,0):',AND(0,0))
print('AND(1,0):',AND(1,0))
print('AND(0,1):',AND(0,1))
print('AND(1,1):',AND(1,1))
```

NAND 게이트 구현

$$y = \begin{cases} 0, & w_1 x_1 + w_2 x_2 + b < 0 \\ 1, & w_1 x_1 + w_2 x_2 + b \geq 0 \end{cases}$$

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

```
import numpy as np

def NAND(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


print('NAND(0,0):',NAND(0,0))
print('NAND(1,0):',NAND(1,0))
print('NAND(0,1):',NAND(0,1))
print('NAND(1,1):',NAND(1,1))
```

NAND 코드 구현해 볼 것

가중치와 편향 도입 예제

■ **OR 게이트 구현**

$$y = \begin{cases} 0, & w_1x_1 + w_2x_2 + b < 0 \\ 1, & w_1x_1 + w_2x_2 + b \geq 0 \end{cases}$$

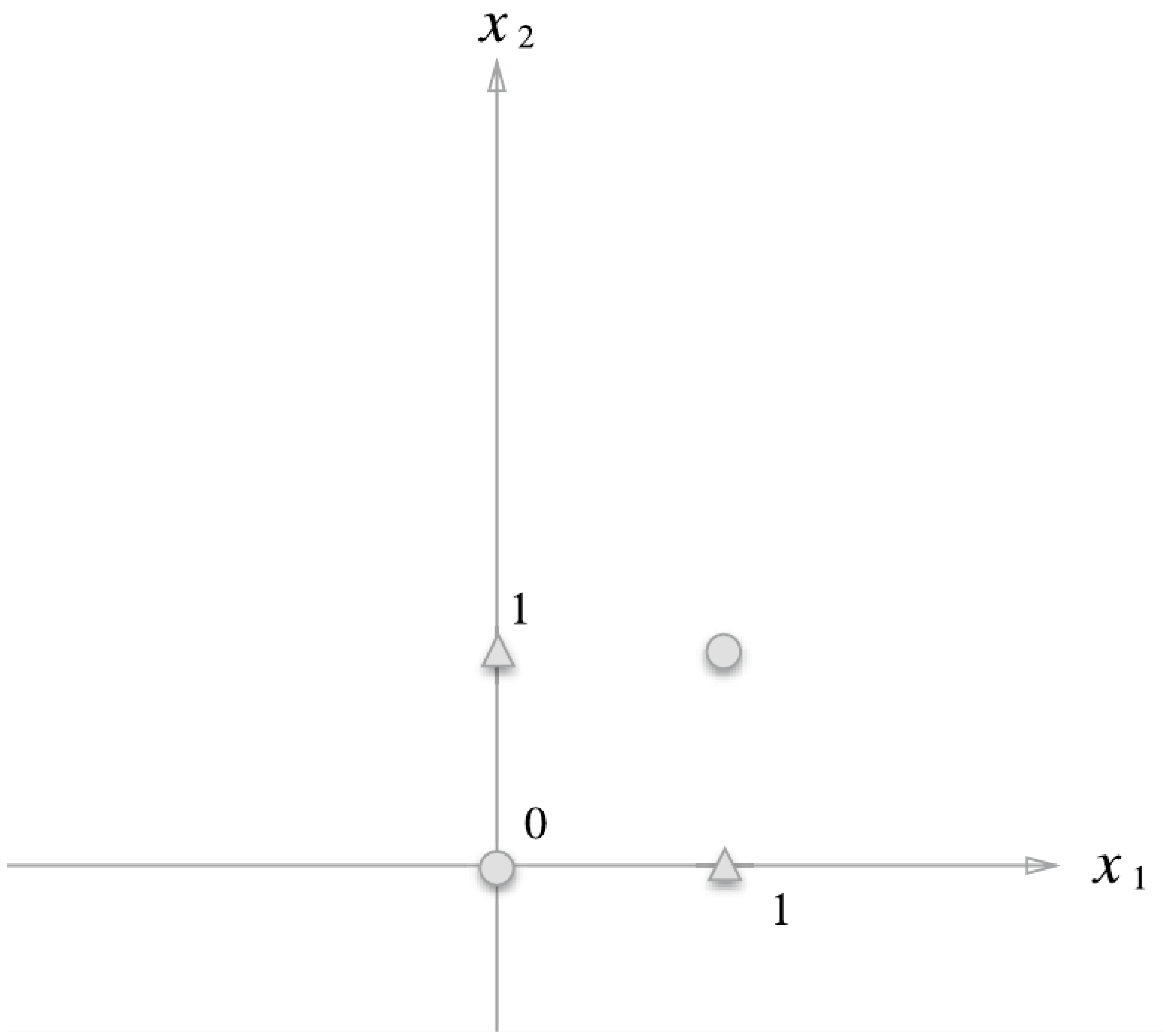
x_1	x_2	y
0	0	0
1	0	1
0	1	1
1	1	1

```
import numpy as np

def OR(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

print('OR(0,0):',OR(0,0))
print('OR(1,0):',OR(1,0))
print('OR(0,1):',OR(0,1))
print('OR(1,1):',OR(1,1))
```

OR 코드 구현해 볼 것

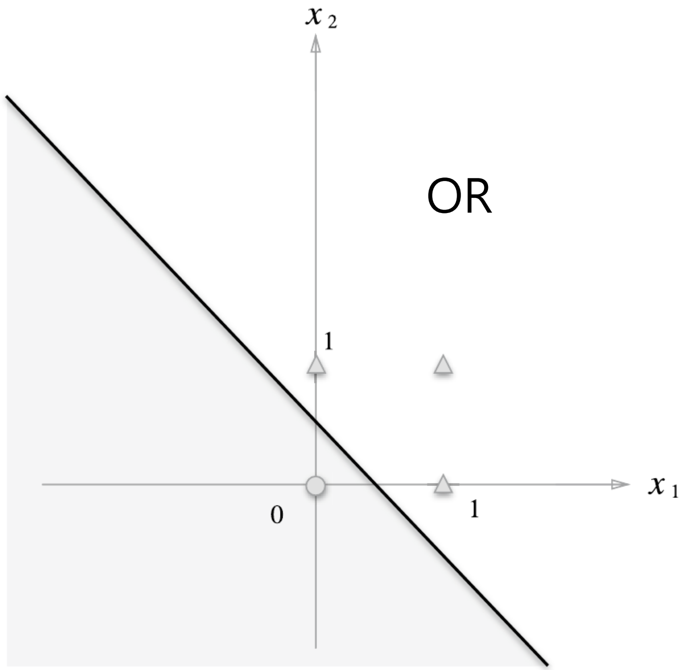
퍼셉트론의 한계

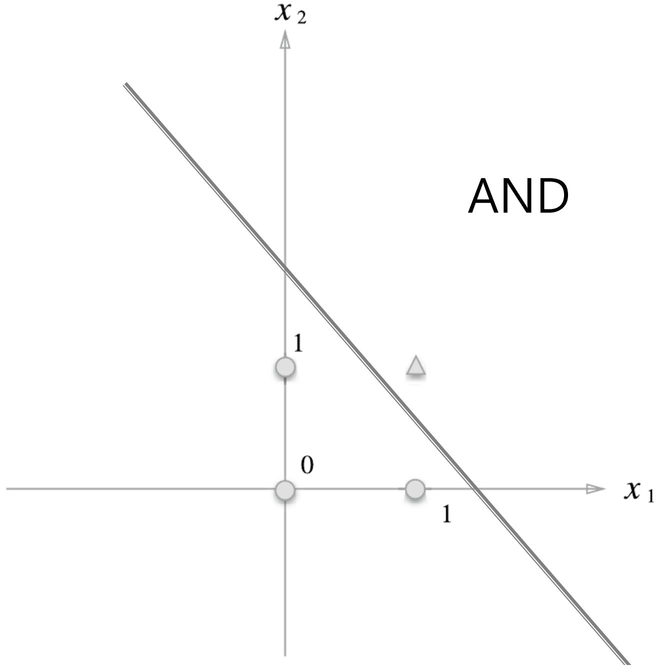
- 입력이 2개인 퍼셉트론으로 XOR 게이트 구현하기

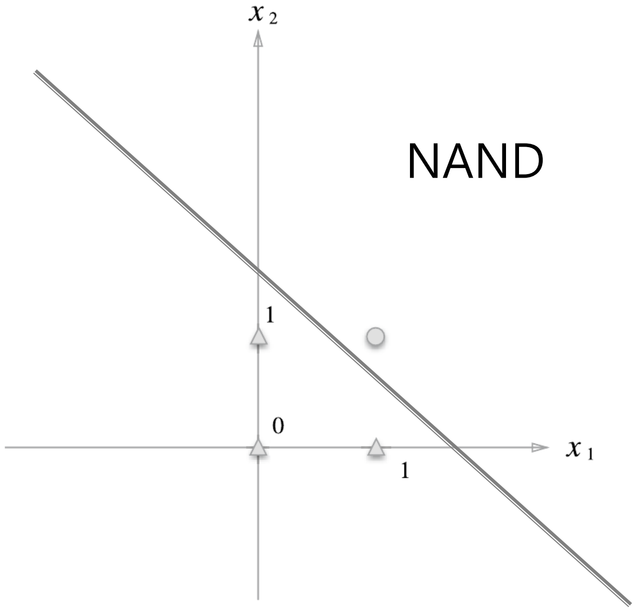


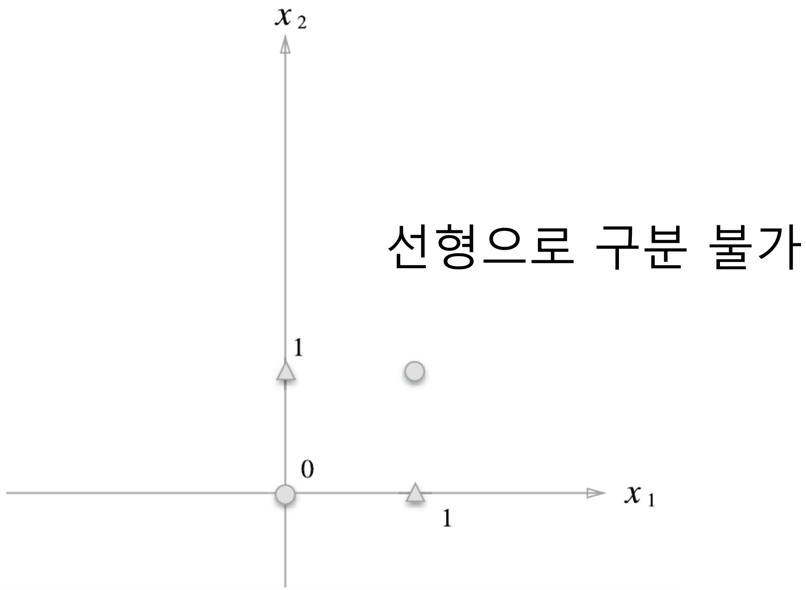
x_1	x_2	y
0	0	0
1	0	1
0	1	1
1	1	0

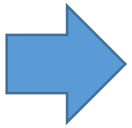
퍼셉트론의 한계



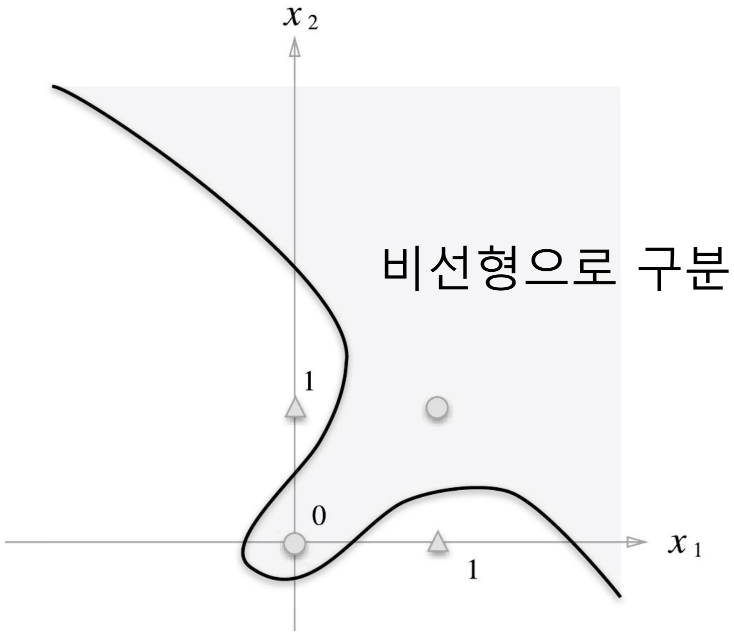




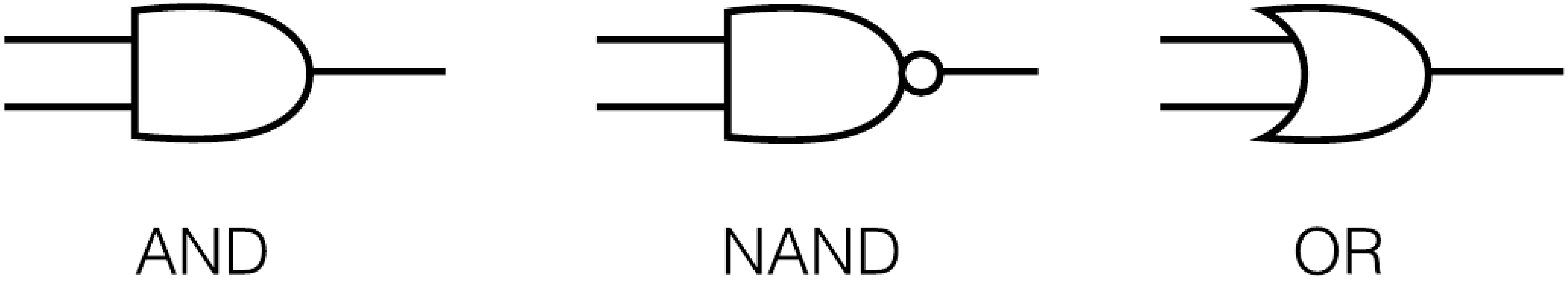


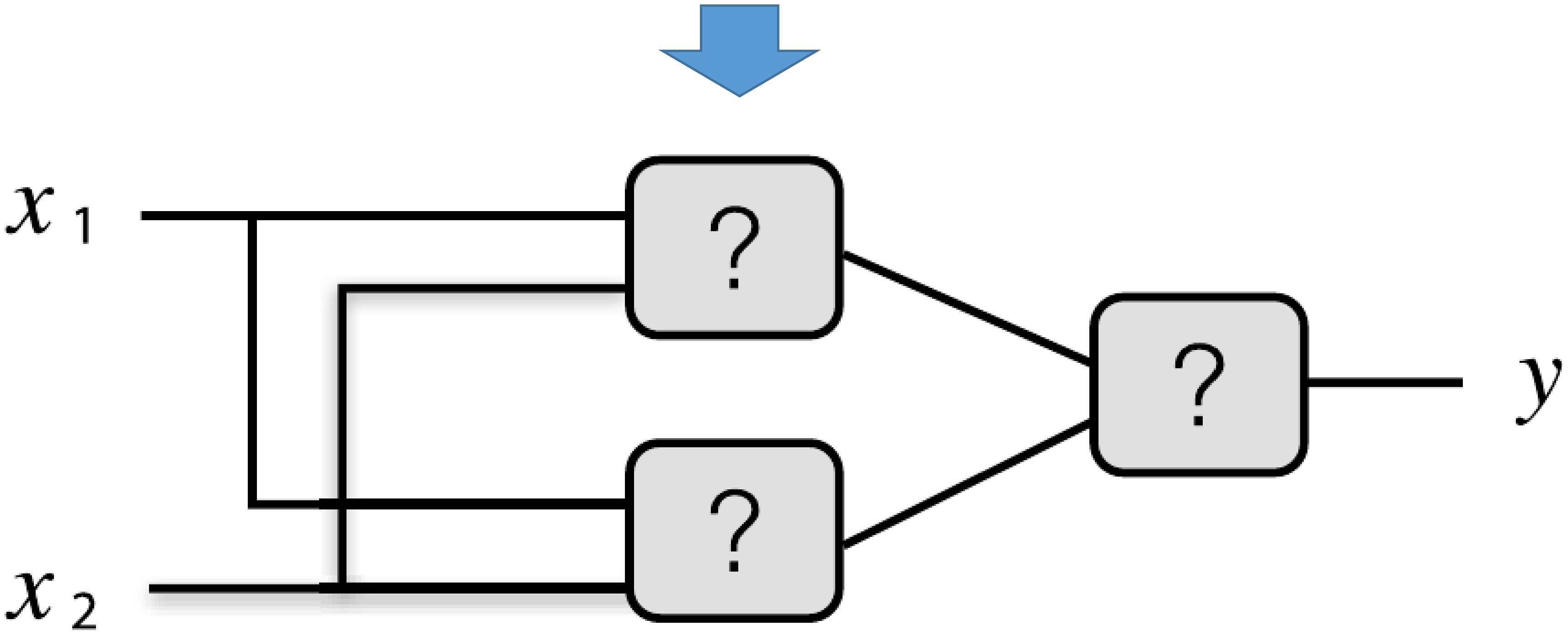


XOR



다층 퍼셉트론으로 구현(Multi-layer Perceptron)

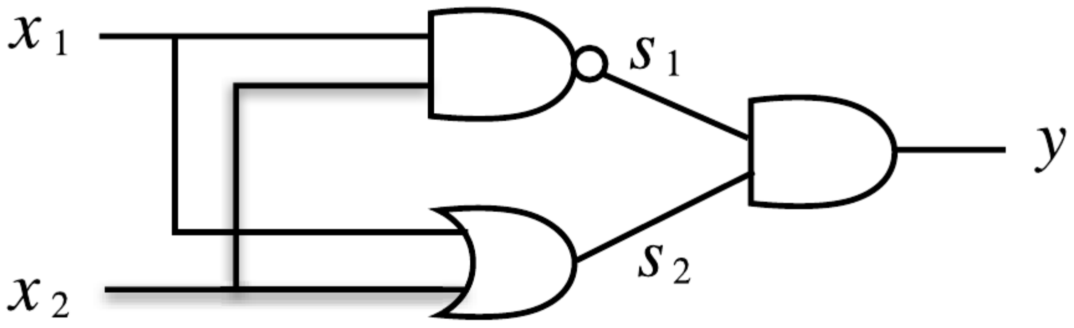


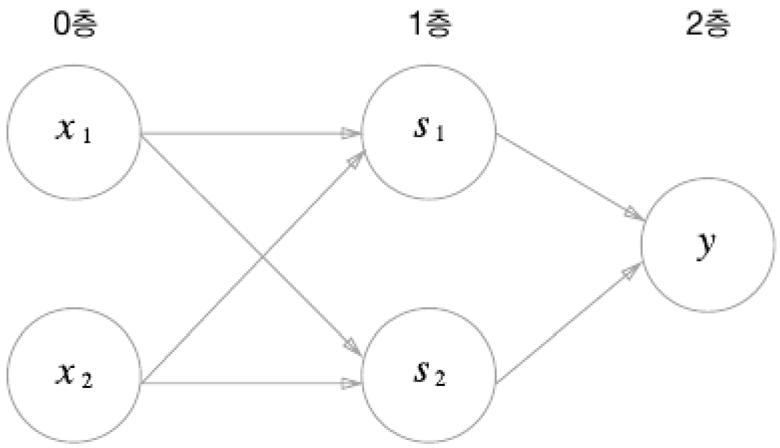


다층 퍼셉트론으로 구현(Multi-layer Perceptron)

- **XOR 게이트 구현하기**

x_1	x_2	s_1	s_2	y
0	0	1	0	0
1	0	1	1	1
0	1	1	1	1
1	1	0	1	0






```python
def XOR(x1,x2):
    s1 = NAND(x1,x2)
    s2 = OR(x1,x2)
    y=AND(s1,s2)
    return y

print('XOR(0,0):',XOR(0,0))
print('XOR(1,0):',XOR(1,0))
print('XOR(0,1):',XOR(0,1))
print('XOR(1,1):',XOR(1,1))
```

AND, NAND, OR 추가하여 결과 완성할것