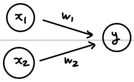
Ch. 2

Perceptron



x_1, x_2: 입력신호 , y: 출력 신호 , w: 가중치 (weight) (w가 클수록, 해당 신호가 중요하다)

θ: 임계값

$$\boxed{y = \begin{cases} 0 & (w_1x_1 + w_2x_2 \le \theta) \\ 1 & (w_1x_1 + w_2x_2 > \theta) \end{cases}}$$

- AND 게이트

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

: 모든 입력값이 1일 때만

$$(w_1, w_2, \theta) = (0.5, 0.5, 0.7) \Rightarrow \begin{cases} y = 0 & (0 \cdot 0.5 + 1 \cdot 0.5 = 0.5 < 0.7) \\ y = 1 & (1 \cdot 0.5 + 1 \cdot 0.5 = 1 > 0.7) \end{cases}$$

$$(w_1, w_2, \theta) = (1.0, 1.0, 1.0) \Rightarrow \begin{cases} y = 0 & (0 \cdot 1 + 1 \cdot 1 \le 1.0) \\ y = 1 & (1 \cdot 1.0 + 1 \cdot 1.0 > 1.0) \end{cases}$$

⋮

- NAND 게이트 (AND와 결과값 반대)

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

$$(w_1, w_2, \theta) = (-0.5, -0.5, -0.7) \Rightarrow \begin{cases} y = 0 & (1 \cdot (-0.5) + 1 \cdot (-0.5) = -1 < -0.7) \\ y = 1 & (1 \cdot (-0.5) + 0 \cdot (-0.5) = -0.5 > -0.7) \end{cases}$$

- OR 게이트

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

: 하나라도 1이면 1 출력

(In Python)

- AND

```
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
```

$\Rightarrow$ AND(0,0) $\rightarrow$ 0

AND(1,0), AND(0,1) $\rightarrow$ 0

AND(1,1) $\rightarrow$ 1

Intro to bias (b)

$$y = \begin{cases} 0 & (x_1 w_1 + x_2 w_2 \leq \theta) \\ 1 & (x_1 w_1 + x_2 w_2 > \theta) \end{cases} \Rightarrow y = \begin{cases} 0 & (b + x_1 w_1 + x_2 w_2 \leq 0) \\ 1 & (b + w_1 x_1 + w_2 x_2 > 0) \end{cases}, \quad b: \text{bias}$$

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

w * x = array([0, 0.5]) : element wise multi

np.sum(w * X) = 0.5 : array 원소들의 합

np.sum(w * X) + b = -0.2

- AND

```
def AND(x1, x2):
    X = np.array([x1, x2])
    w = np.array([w1, w2])
    b = -0.7
    tmp = np.sum(X * w) + b

    if tmp <= 0:
        return 0
    else:
        return 1
```

w_1, w_2 (weight) : determines the importance of each neurons

b (bias) : how easily y neuron is activated

- NAND

```
def NAND(x1, x2):
    x = np.array([x1, x2])
    w = np.array([-0.5, -0.5])
    b = 0.7
    tmp = np.sum(x * w) + b

    if tmp <= 0:
        return 0
    else:
        return 1
```

⇒ (Note: AND, NAND, OR 를 가르는건 w, b 값

⇒ 함수에서 w, b 값을 미리 정해줘야함)

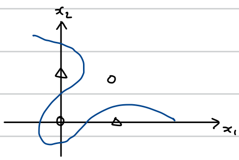
- OR

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

$0.5x_1 + 0.5x_2 - 0.2$

- XOR 게이트 : x_1, x_2 하나만 1일 때 1 출력

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



: non-linear ⇒ multi-layer perceptron is needed !

- Multi-Layer Perceptron

NAND (s_1)

x_1	x_2	s_1
0	0	1
0	1	1
1	0	1
1	1	0

OR (s_2)

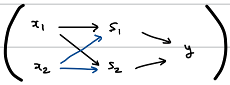
x_1	x_2	s_2
0	0	0
0	1	1
1	0	1
1	1	1

⇒ AND (y)

s_1	s_2	y
1	0	0
1	1	1
0	1	0

x_1, x_2 : 2개의 입력값을 받아 s_1, s_2에 각각 넣기

⇒ 각각 만든 s_1, s_2로 XOR 제작



(Python)

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

⇒ XOR(0,0) → 0

XOR(0,1), XOR(1,0) → 1

XOR(1,1) = 0