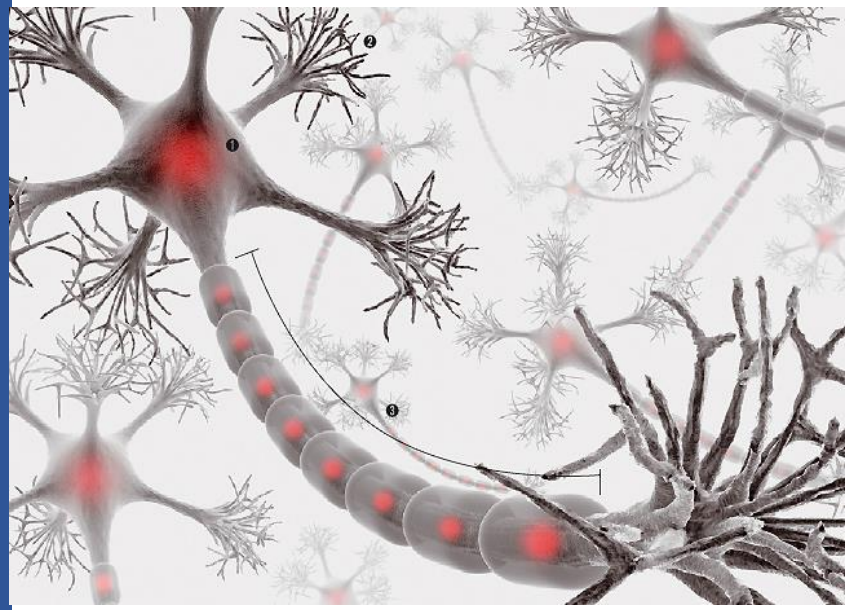
# Numpy로 신경망 만들기

**학습 목표**

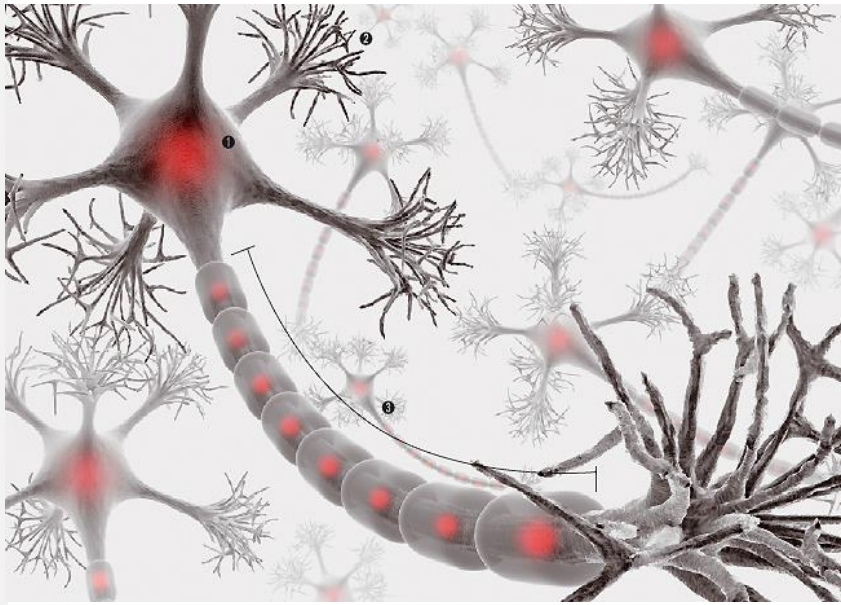
- Numpy를 이용해서 DNN 구성 및 Forward Pass 구현해 본다.

**주요 내용**

- Classification
- Regression

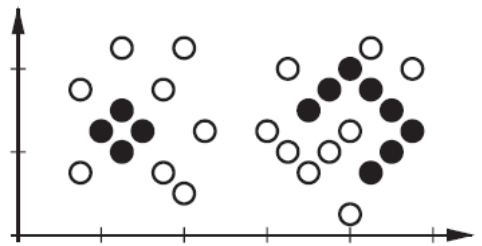


# 1 Classification

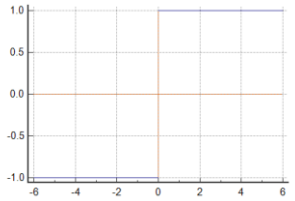


# Classification

**분류 문제**



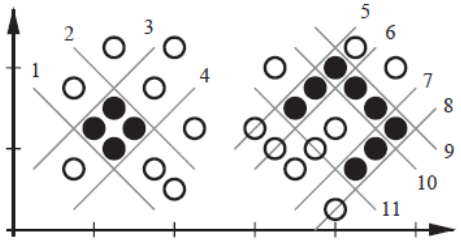
**Activation function :**



$$f(n) = \begin{cases} 1 & \text{if } n \geq 0 \\ -1 & \text{otherwise} \end{cases}$$

3계층 네트워크 필요

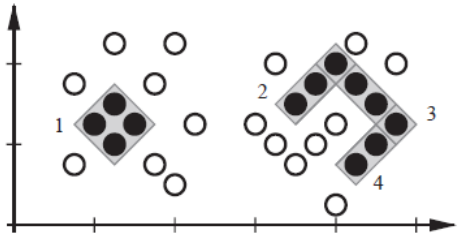
**첫번째 계층**




$$(\mathbf{W}^1)^T = \begin{bmatrix} 1 & -1 & 1 & -1 & 1 & -1 & 1 & -1 & -1 & 1 & 1 \\ 1 & -1 & -1 & 1 & -1 & 1 & -1 & 1 & -1 & 1 & 1 \end{bmatrix}$$

$$(\mathbf{b}^1)^T = \begin{bmatrix} -2 & 3 & 0.5 & 0.5 & -1.75 & 2.25 & -3.25 & 3.75 & 6.25 & -5.75 & -4.75 \end{bmatrix}$$

**두번째 계층**




$$\mathbf{W}^2 = \begin{bmatrix} 1 & 1 & 1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 1 & 0 & 0 & 1 & 0 & 1 \\ 0 & 0 & 0 & 0 & 1 & 0 & 0 & 1 & 1 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & 1 & 1 & 0 & 1 \end{bmatrix}, \mathbf{b}^T = \begin{bmatrix} -3 \\ -3 \\ -3 \\ -3 \end{bmatrix}$$

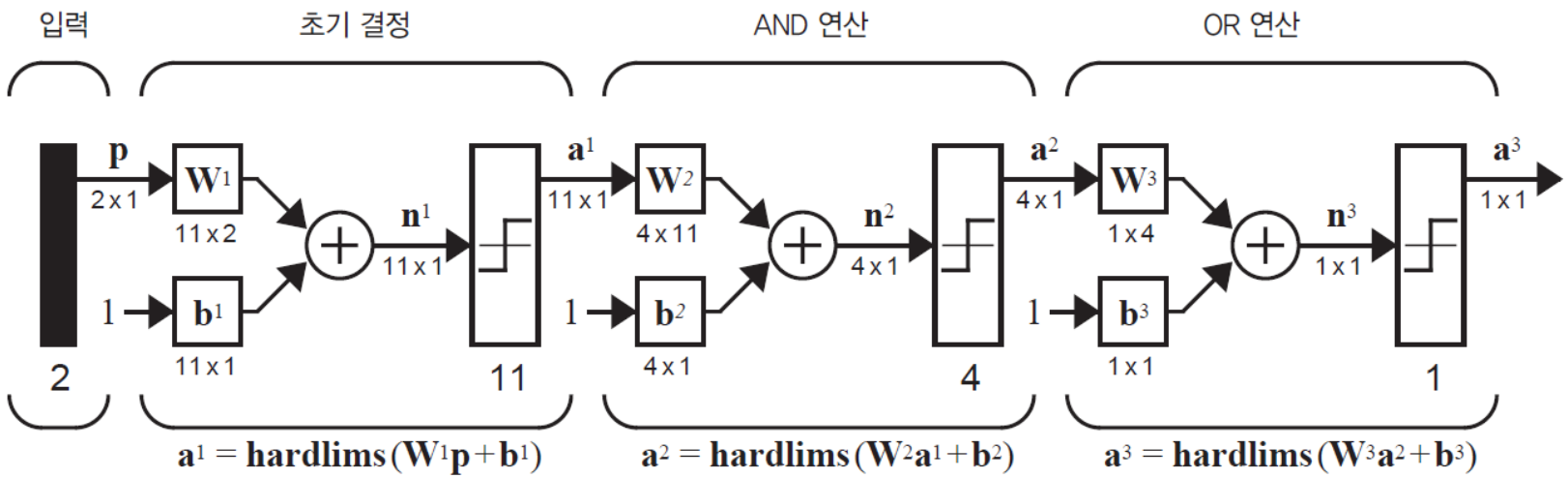
11개 뉴런의 출력을 두 번째 계층의 AND 뉴런을 이용해 그룹으로 결합

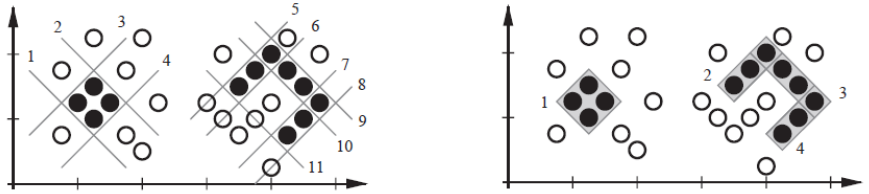
**세번째 계층**

세 번째 계층에서는 OR 연산을 사용해 두 번째 계층의 네 결정 영역을 하나의 영역으로 결합

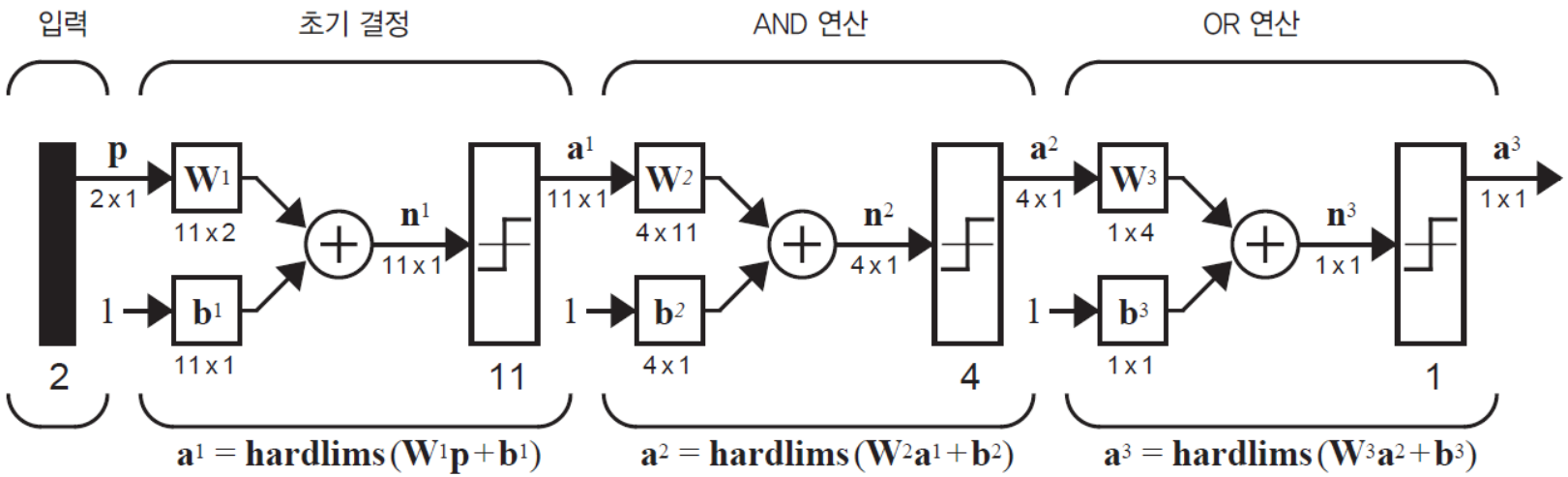
$$\mathbf{W}^3 = \begin{bmatrix} 1 & 1 & 1 & 1 \end{bmatrix}, \mathbf{b}^3 = \begin{bmatrix} 3 \end{bmatrix}$$

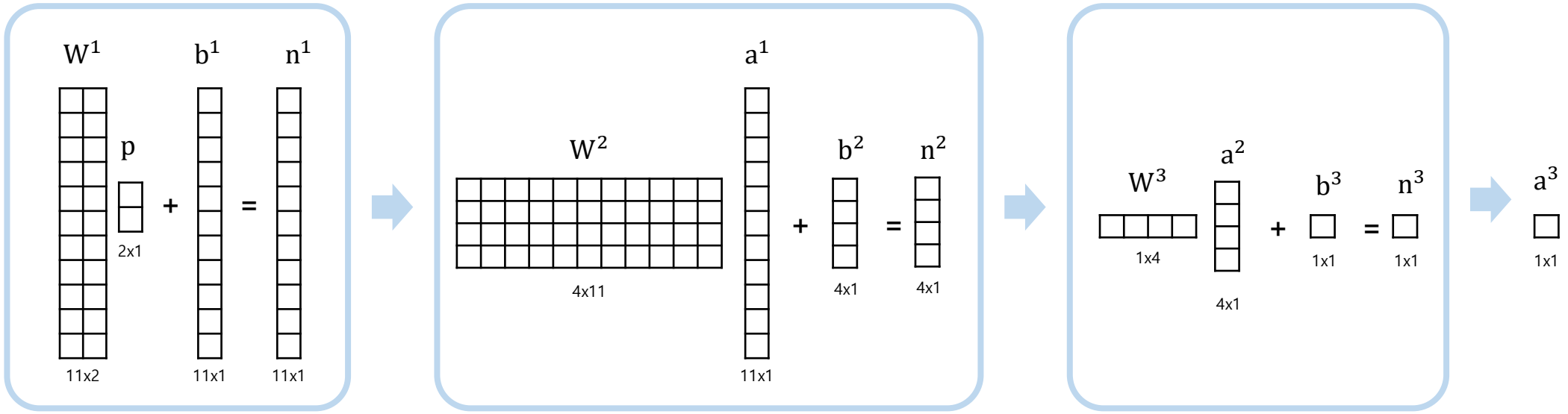
# Classification



# Classification

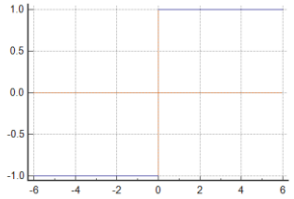


# Activation Function (문제)

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

# activation function을 정의하시오
def activation(n):
    # your code
    return x
```

**Activation function :**




$$f(n) = \begin{cases} 1 & \text{if } n \geq 0 \\ -1 & \text{otherwise} \end{cases}$$

# 가중치, 편향 (문제)

```
# W1, W2, W3, b1, b2, b3를 정의하시오.

W1 = np.array([[1, -1, 1, -1, 1, -1, 1, -1, -1, 1, 1],
               [1, -1, -1, 1, -1, 1, -1, 1, -1, 1, 1]], float).transpose()
b1 =  # your code
W2 = # your code
b2 =  # your code
W3 = # your code
b3 =  # your code
```

$$(\mathbf{W}^1)^T = \begin{bmatrix} 1 & -1 & 1 & -1 & 1 & -1 & 1 & -1 & -1 & 1 & 1 \\ 1 & -1 & -1 & 1 & -1 & 1 & -1 & 1 & -1 & 1 & 1 \end{bmatrix}$$

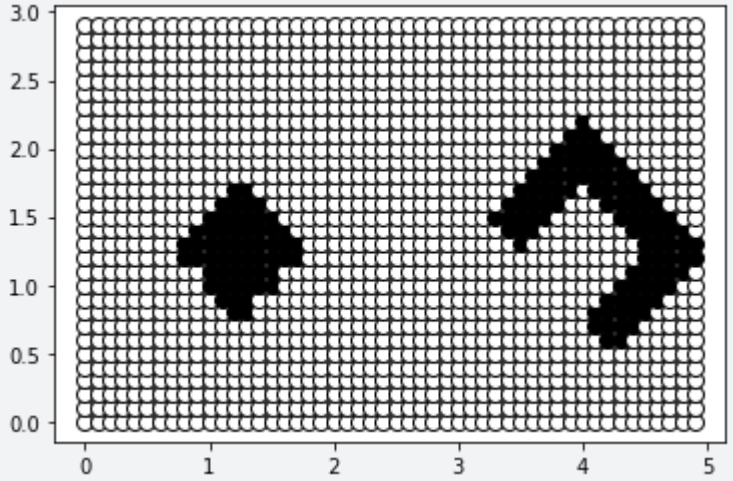
$$(\mathbf{b}^1)^T = \begin{bmatrix} -2 & 3 & 0.5 & 0.5 & -1.75 & 2.25 & -3.25 & 3.75 & 6.25 & -5.75 & -4.75 \end{bmatrix}$$

$$\mathbf{W}^2 = \begin{bmatrix} 1 & 1 & 1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 1 & 0 & 0 & 1 & 0 & 1 \\ 0 & 0 & 0 & 0 & 1 & 0 & 0 & 1 & 1 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & 1 & 1 & 0 & 1 \end{bmatrix}, \quad \mathbf{b}^T = \begin{bmatrix} -3 \\ -3 \\ -3 \\ -3 \end{bmatrix}$$

$$\mathbf{W}^3 = \begin{bmatrix} 1 & 1 & 1 & 1 \end{bmatrix}, \quad \mathbf{b}^3 = \begin{bmatrix} 3 \end{bmatrix}$$

# Forward Pass (문제)

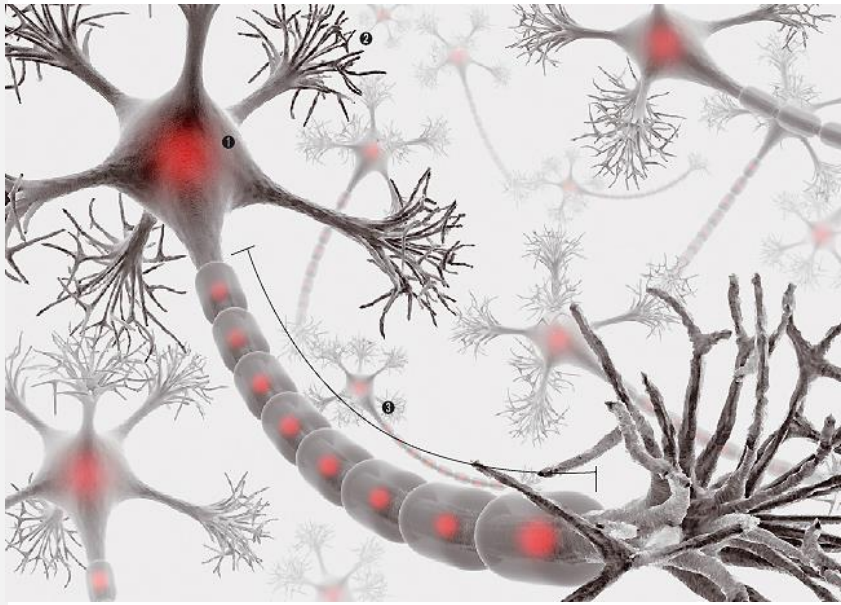
```
# forward pass를 구현하시오 (Hint : np.dot 사용)
x_precision = 50
y_precision = 30
for i in range(x_precision):
    for j in range(y_precision):
        x=i/x_precision*5
        y=j/y_precision*3

        input = np.array([x,y], float).transpose()

        # forward pass를 구현하시오 (Hint : np.dot 사용)
        a1 = # your code
        a2 = # your code
        a3 = # your code
        if a3==1:
            plt.plot(x,y,'o',mec='k',mfc='k',ms=8)
        else :
            plt.plot(x,y,'o',mec='k',mfc='w',ms=8)
```




mec : markeredgecolor, mfc : markerfacecolor

$$\mathbf{a}^1 = \mathbf{hardlims}(\mathbf{W}^1\mathbf{p}+\mathbf{b}^1) \qquad \mathbf{a}^2 = \mathbf{hardlims}(\mathbf{W}^2\mathbf{a}^1+\mathbf{b}^2) \qquad \mathbf{a}^3 = \mathbf{hardlims}(\mathbf{W}^3\mathbf{a}^2+\mathbf{b}^3)$$

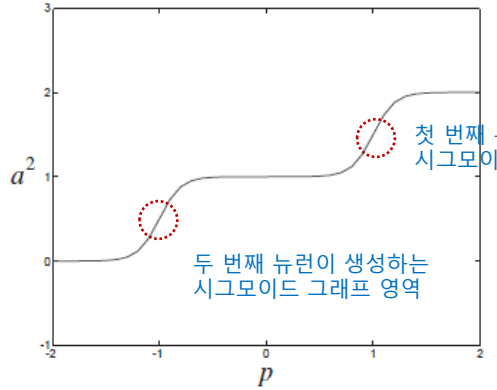
# 2 Function Approximation



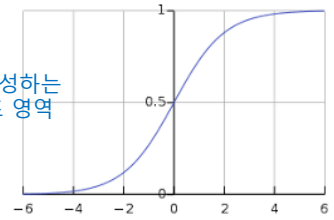
# Function Approximation

**다음과 같은 함수를 근사해 보시오.**

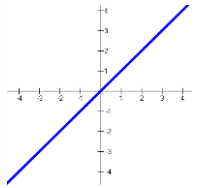
## 함수 근사 문제




**Activation function :**

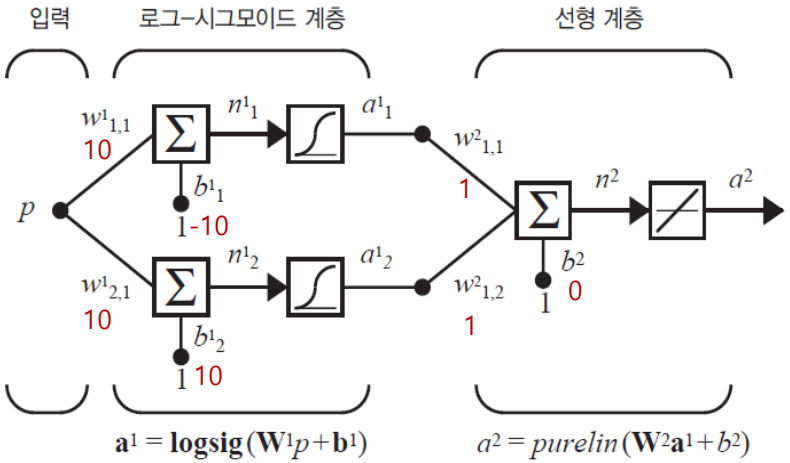


$$f^1(n) = \frac{1}{1+e^{-n}}$$



$$f^2(n) = n$$

## 함수 근사를 위한 2계층 네트워크

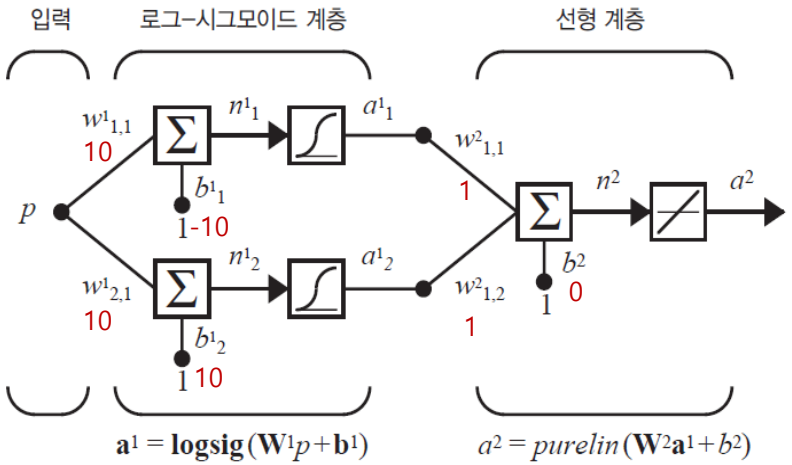



$$n^1_1 = w^1_{1,1}p + b^1_1 = 0 \quad \Rightarrow \quad p = -\frac{b^1_1}{w^1_{1,1}} = -\frac{-10}{10} = 1$$

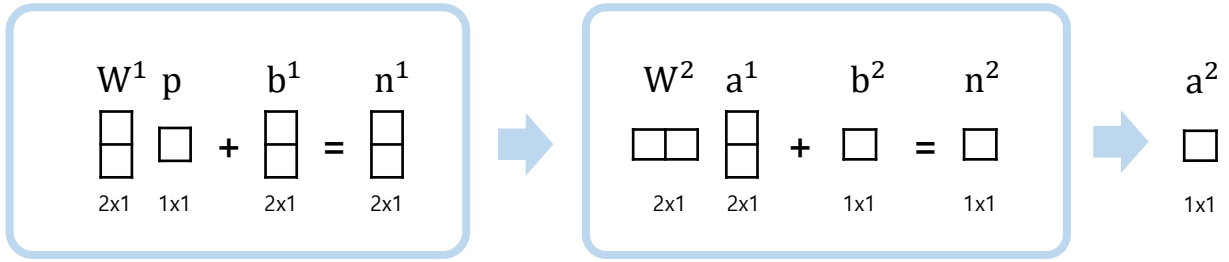
$$n^1_2 = w^1_{2,1}p + b^1_2 = 0 \quad \Rightarrow \quad p = -\frac{b^1_2}{w^1_{2,1}} = -\frac{10}{10} = -1$$

계단의 중심은 첫 번째 계층 뉴런의 네트 입력이 0인 지점에 있다.

# Function Approximation



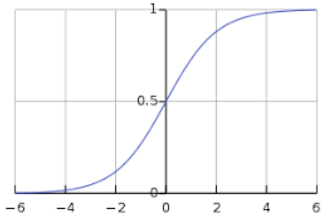

현재 표기법이 열 벡터 기준



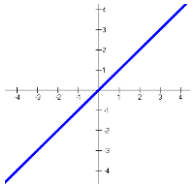
# Activation Function (문제)

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

# activation function을 정의하시오
# Sigmoid Activation Function
def activation_1(n):
    # your code
    return x

# Identity Activation Function
def activation_2(n):
    # your code
    return x
```

**Activation function :**




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

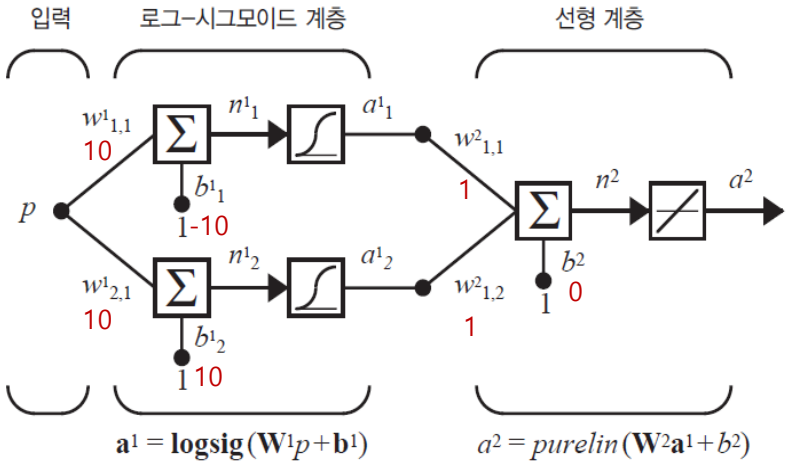


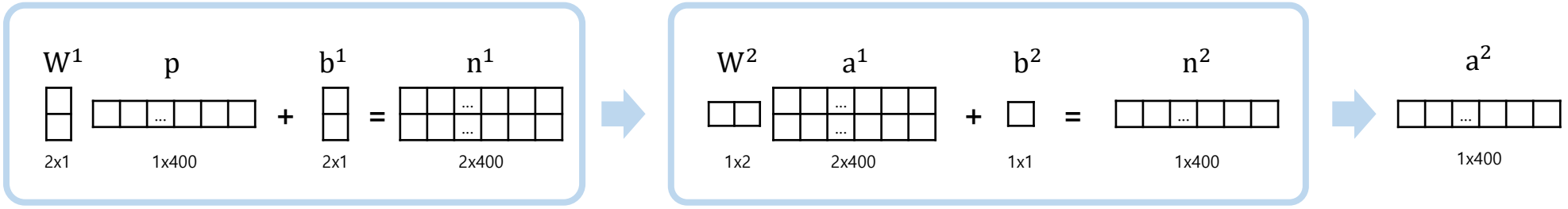
$$f^2(n) = n$$

# 가중치, 편향 (문제)

## 가중치, 편향 정의

```
p = np.arange(-2, 2, 0.01)
p = p[np.newaxis, : ] # 입력 데이터는 (1, 400) 형태

 # W1, W2, b1, b2를 정의하시오.
W1 = # your code
b1 =  # your code

W2 = # your code
b2 =  # your code
```

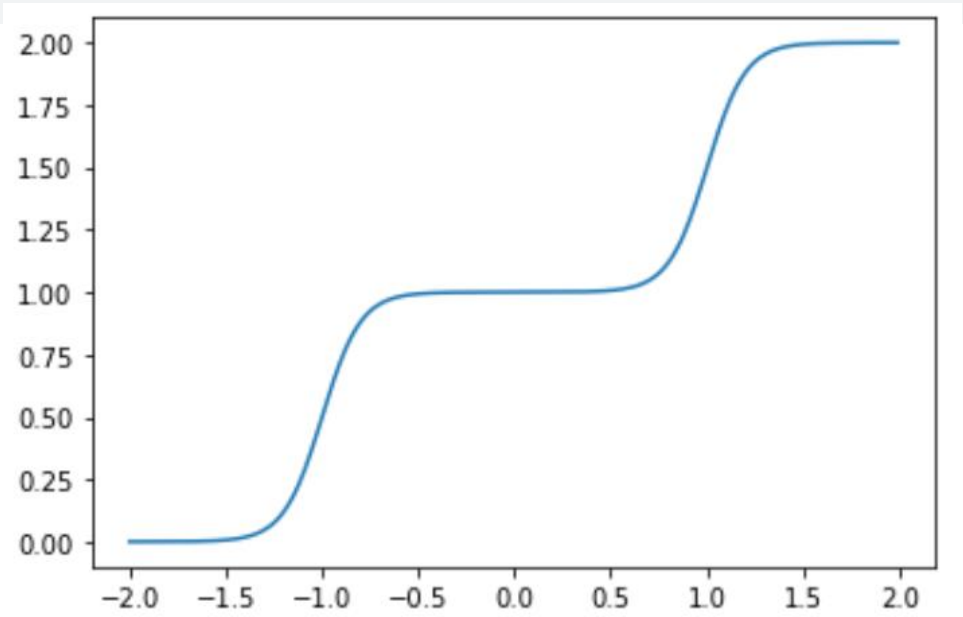



입력 데이터는 전체 sample 400개를 배치로 넣어서 계산

# Forward pass (문제)

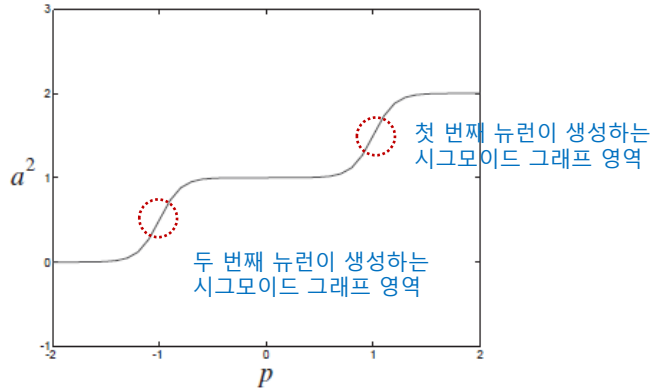
```
# forward pass를 구현하시오 (Hint : np.dot 사용)
a1 = # your code
a2 = # your code

plt.plot(p.flatten(), a2.flatten())
plt.axis([-2, 2, -1, 3])
plt.show()
```



# Function Approximation

**함수의 파라미터를 바꿔보면서 네트워크에 미치는 영향을 확인해 보시오.**

### 네트워크 반응



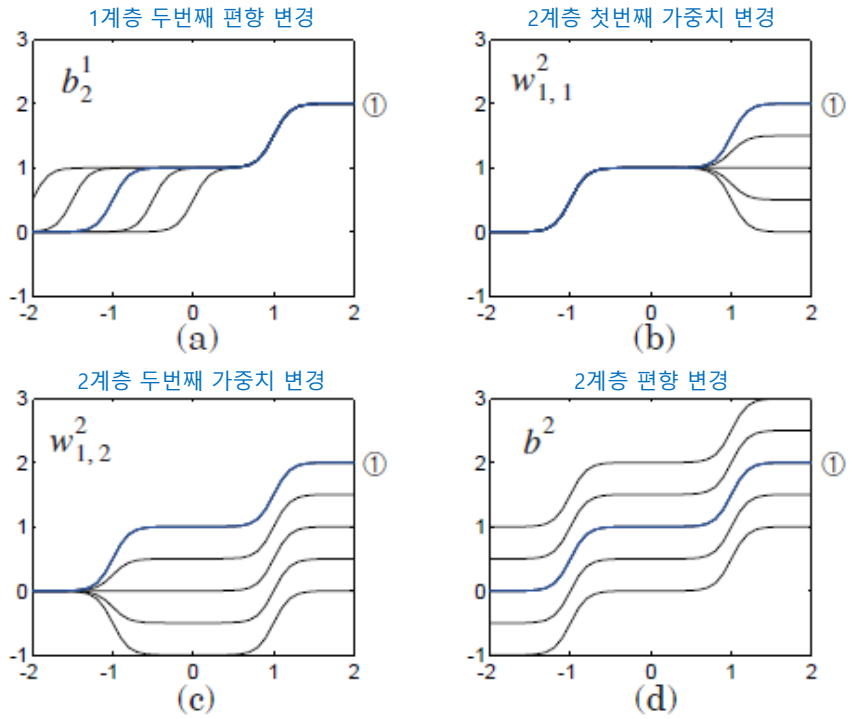

$$n_1^1 = w_{1,1}^1 p + b_1^1 = 0 \quad \Rightarrow \quad p = -\frac{b_1^1}{w_{1,1}^1} = -\frac{-10}{10} = 1$$

$$n_2^1 = w_{2,1}^1 p + b_2^1 = 0 \quad \Rightarrow \quad p = -\frac{b_2^1}{w_{2,1}^1} = -\frac{10}{10} = -1$$

계단의 중심은 첫 번째 계층 뉴런의 네트 입력이 0인 지점에 있다.

**파라미터 변경**

### 네트워크 반응에 미치는 영향



# Thank you!