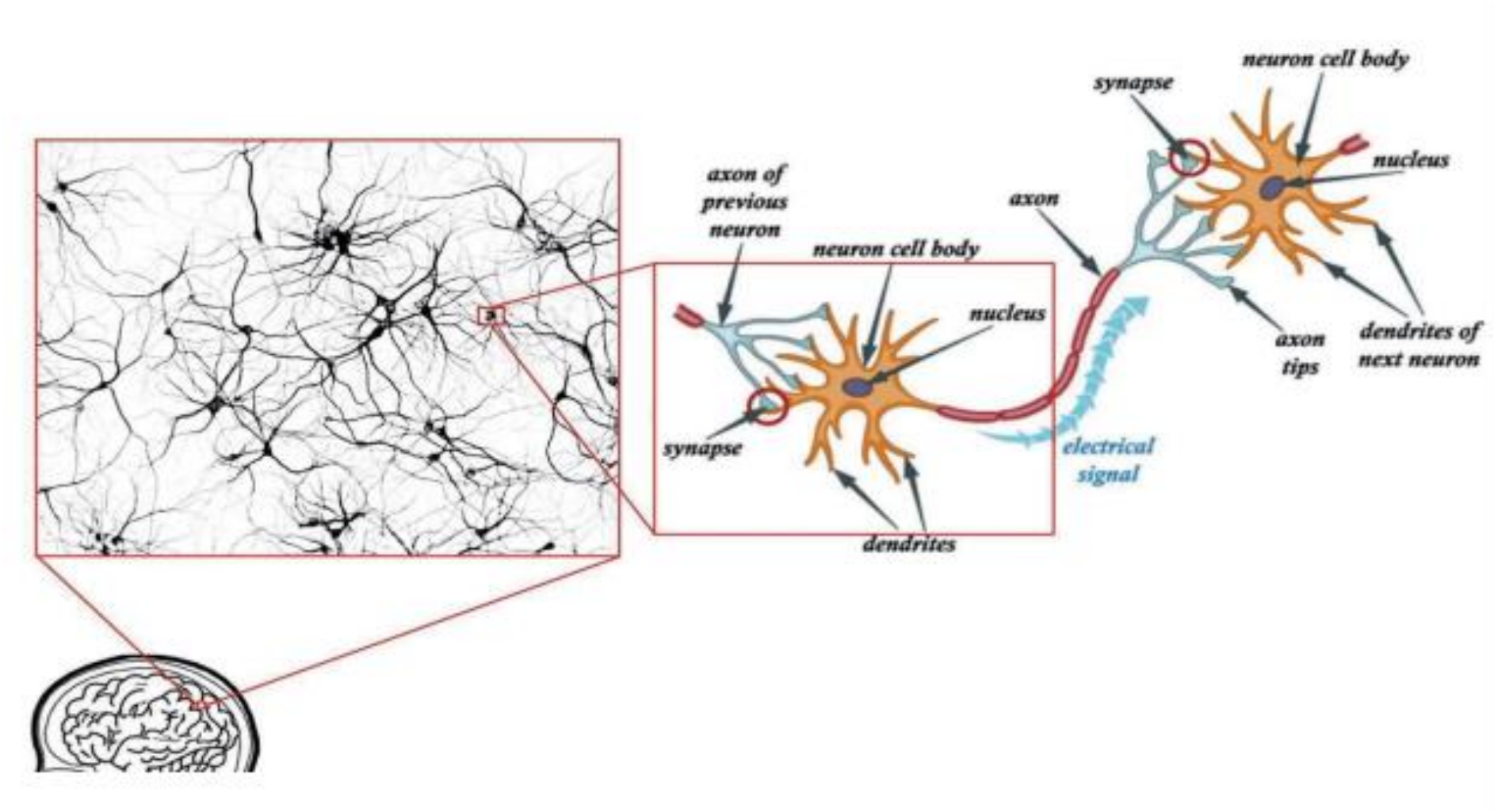
신경망 기초

- Biological Neural Network
- Artificial Neural Network
- Activation Function
- Single-Layer Perceptron
- Multiple Layer Perceptron
- Single-Layer Perceptron 학습

생물학적 신경망(biological neural network)



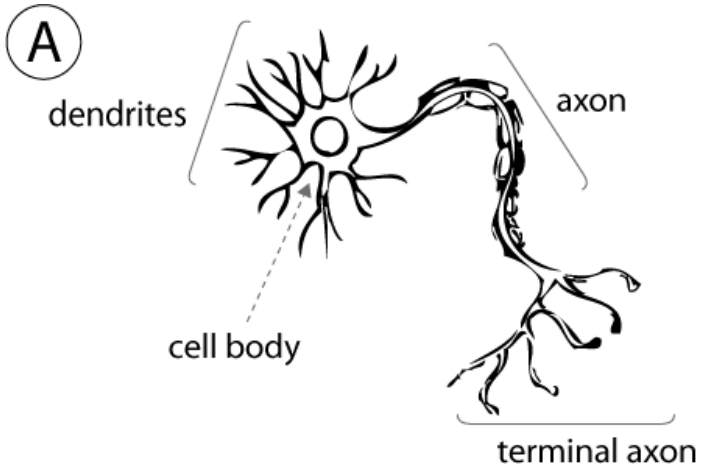
뇌는 수많은 뉴런(neuron)이 시냅스(synapse)를 통하여 네트워크를 구성
뉴런은 외부자극을 받아서 반응하는 기능을 담당

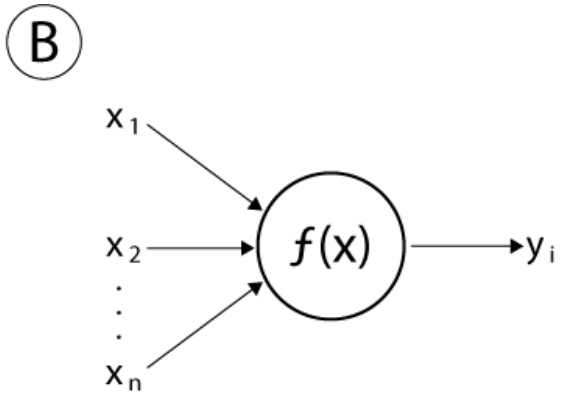
생물학적 신경망(biological neural network)

- 뉴런(Neuron)
  - 생물체들이 외부자극을 받아들이고 반응하는 신경계를 구성하는 신경세포
  - 인접한 다른 뉴런과 신경접합부(Synapse)를 통해 신호를 주고 받음으로써 다양한 정보를 받아들이고, 저장하고 계산하는 기능을 함
- 뉴런의 구조
  - 수상돌기(dendrite) : 신경세포가 신호를 받아들이는 가지
  - 세포체(soma, cell body): 신경세포의 중심부로 세포핵(nucleus) 을 가짐, 수상돌기로 들어온 신호를 합하여 값이 크면 '활동전위(action potential)'가 생겨 다음 신경세포에 전달되는 신호가 강해짐
  - 축삭(axon) : 세포체로부터 아주 길게 뻗어나가는 부분으로 수상돌기와 세포체를 거쳐 전달된 신호를 다른 신경세포에 전달하는 부분
  - 시냅스(synapse) : 인접한 두 신경 세포가 연접하면서 만드는 구조, 전기적인 신호로 전달된 신호는 신경전달물질이라는 화학적 신호로 바뀌어 시냅스를 통과
- 생물학적 신경망(biological neural network)
  - 인간의 신경계에서 약 860 억 개의 뉴런이 발견되며 시냅스와 연결되어 네트워크(neural network)를 구성

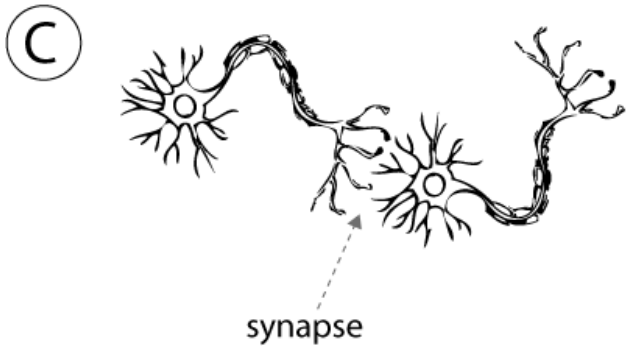
인공신경망(artificial neural network)

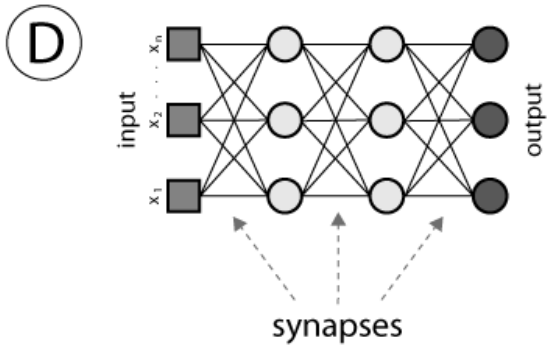
- 생물학적 신경 시스템이 정보를 처리하는 기능을 모방하여 인공적으로 구축한 신경망





(B) 신경세포를 노드로 표현
dendrite를 입력값 x_1 .. x_n으로 표현
cell body의 기능을 활성화함수 f(x)로 표현
axon 출력을 y_1 으로 표현



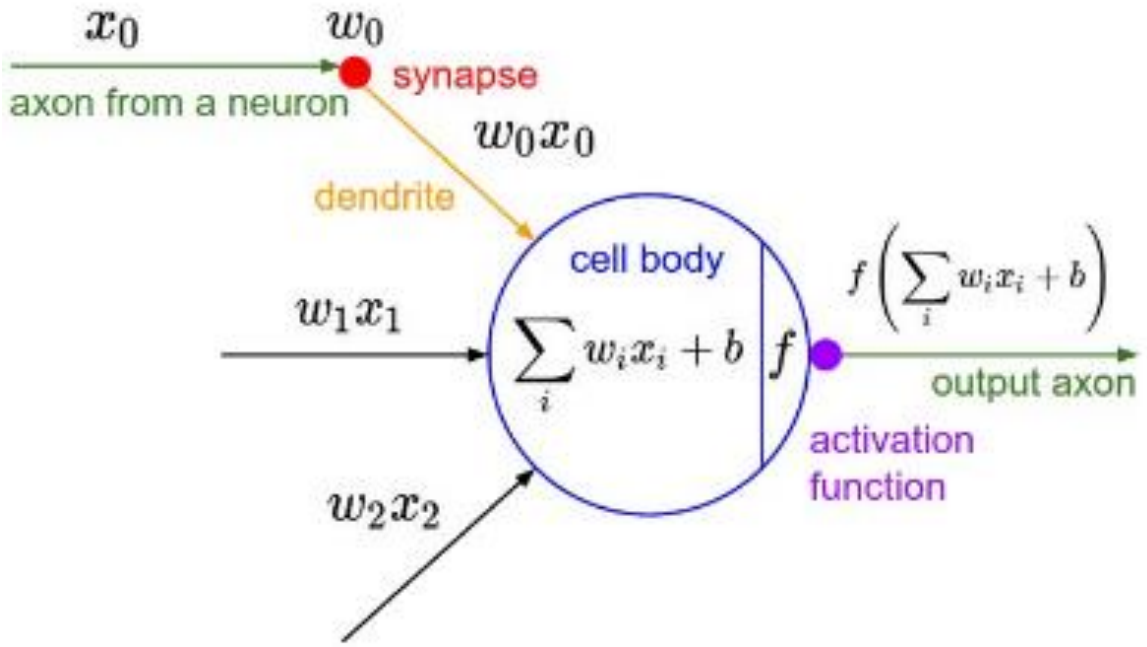


(D) 인공신경망 구조
신경세포사이의 연결부인 시냅스의 연결강도를 인공신경망의 노드들의 연결선으로 표현
연결강도가 클수록 이전 층에서의 입력값에 더 큰 영향을 줌

https://medium.com/@ivanliljeqvist/the-essence-of-artificial-neural-networks-5de300c995d6

인공신경망(artificial neural network)

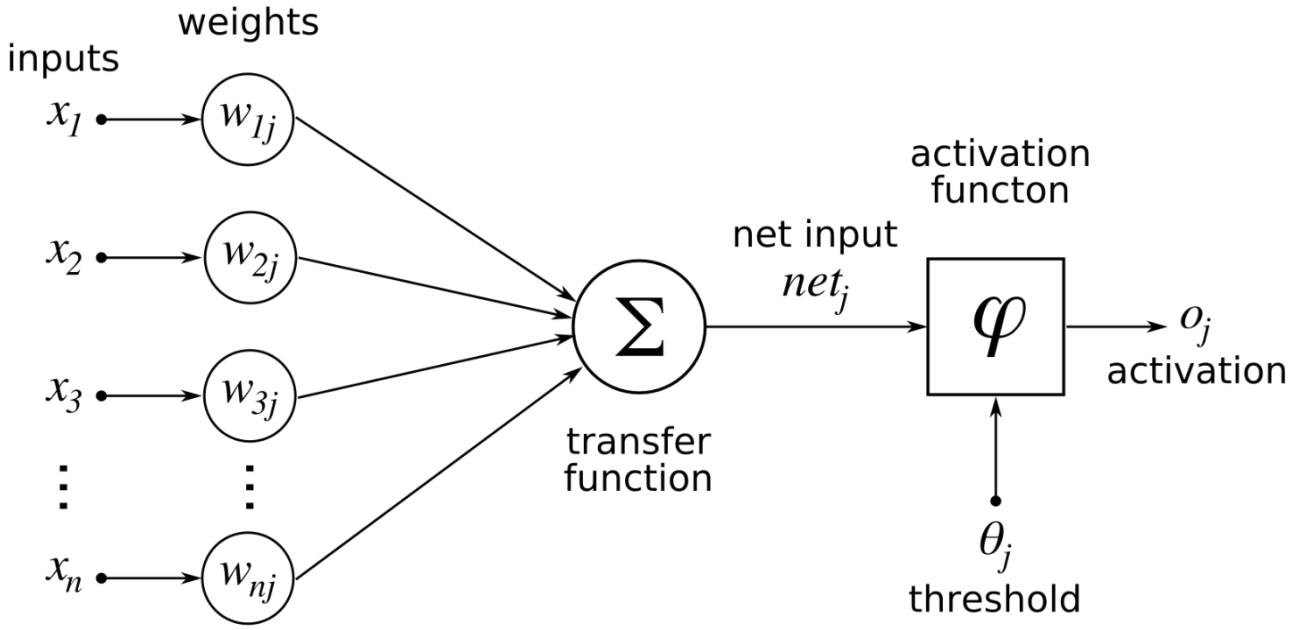
- 이전 뉴런의 신호는 입력 x 와 시냅스의 연결강도 w 를 곱하여 모든 뉴런들의 신호를 합한 후 활성화 함수를 적용한 결과를 출력



https://cs231n.github.io/neural-networks-1/

인공신경망(artificial neural network)

- Perception(퍼셉트론)
  - 초기 형태의 인공 신경망
  - 다수의 입력으로부터 하나의 결과를 내보내는 알고리즘

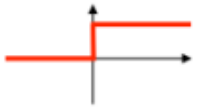
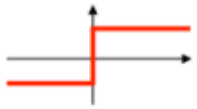
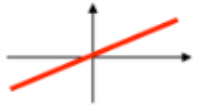
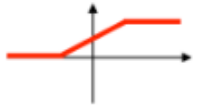
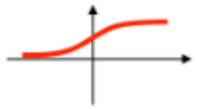
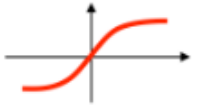
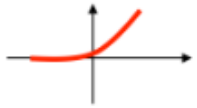


$$y = \begin{cases} 0 & (w_1x_1 + w_2x_2 \leq \theta) \\ 1 & (w_1x_1 + w_2x_2 > \theta) \end{cases}$$

뉴런에서 전달 받은 신호의 총합이 임계값 Θ를 넘을 때만 1을 출력

- transfer function (전달함수) : 입력 노드들의 입력값(x_i)과 연결강도(가중치, w_j)를 곱하여 합을 구하는 함수(weighted sum)
- activation function(활성 함수) : 임계값에 따라 활성화 여부를 결정하는 함수

활성함수(activation function)

Activation function	Equation	Example	1D Graph
Unit step (Heaviside)	$\phi(z) = \begin{cases} 0, & z < 0, \\ 0.5, & z = 0, \\ 1, & z > 0, \end{cases}$	Perceptron variant	
Sign (Signum)	$\phi(z) = \begin{cases} -1, & z < 0, \\ 0, & z = 0, \\ 1, & z > 0, \end{cases}$	Perceptron variant	
Linear	$\phi(z) = z$	Adaline, linear regression	
Piece-wise linear	$\phi(z) = \begin{cases} 1, & z \geq \frac{1}{2}, \\ z + \frac{1}{2}, & -\frac{1}{2} < z < \frac{1}{2}, \\ 0, & z \leq -\frac{1}{2}, \end{cases}$	Support vector machine	
Logistic (sigmoid)	$\phi(z) = \frac{1}{1 + e^{-z}}$	Logistic regression, Multi-layer NN	
Hyperbolic tangent	$\phi(z) = \frac{e^z - e^{-z}}{e^z + e^{-z}}$	Multi-layer Neural Networks	
Rectifier, ReLU (Rectified Linear Unit)	$\phi(z) = max(0, z)$	Multi-layer Neural Networks	
Rectifier, softplus	$\phi(z) = \ln(1 + e^z)$	Multi-layer Neural Networks	

Unit step : 전달받은 값 z가 음수이면 0, 0이면 0.5, 양수이면 1로 출력

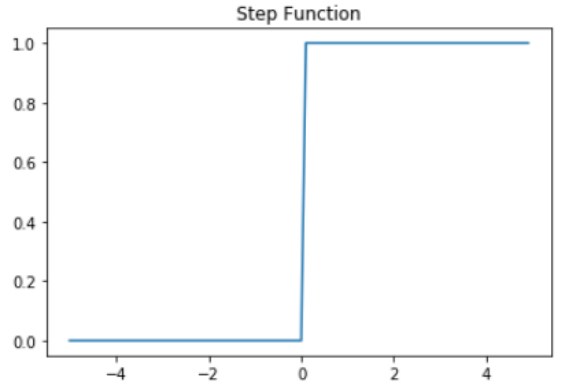
Logistic(sigmoid) : 전달받은 값 z를 비선형 함수인 시스모이드를 적용하여 0~1 출력, z값이 크거나 작을 때 출력에 영향을 덜 줄 수 있는 함수

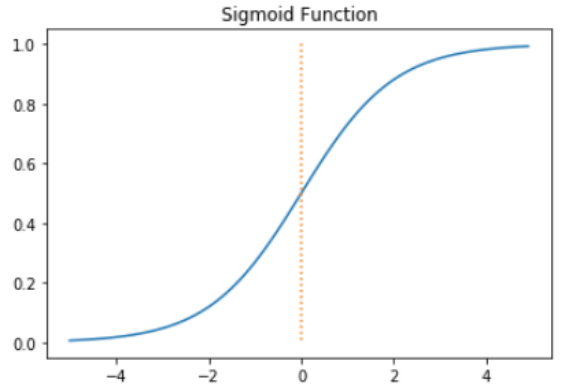
ReLU : 전달받은 값 z를 직선조합으로 비선형을 표현, (0~z) 출력, 딥러닝에서 주로 사용

활성함수(activation function)

- step, sigmoid 함수 구현 및 그래프 출력

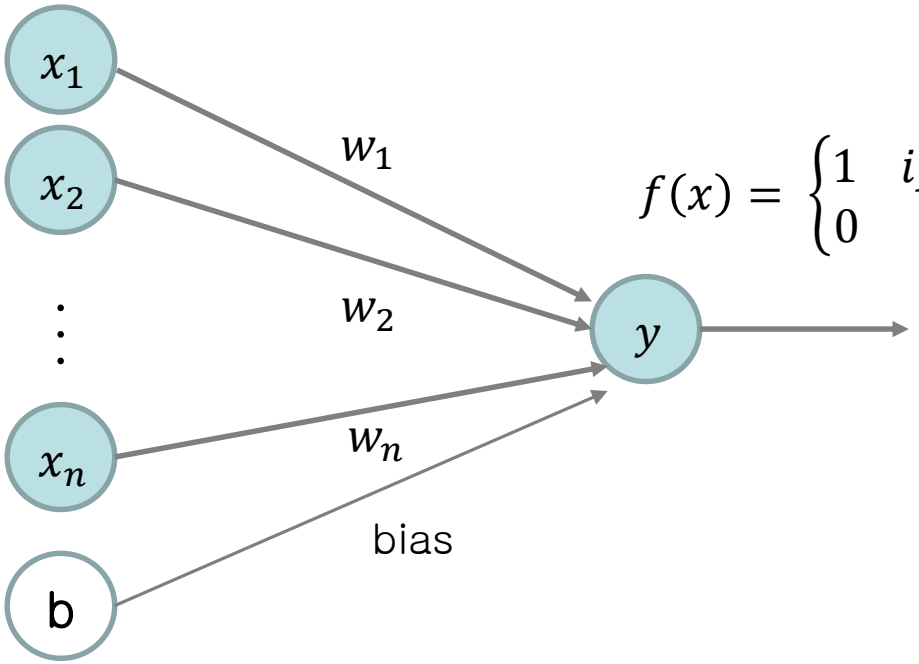
```
#step functrion
def step(x):
    return np.array(x > 0, dtype=np.int)

x = np.arange(-5.0, 5.0, 0.1) # -5.0부터 5.0까지 0.1 간격 생성
y = step(x)
plt.title('Step Function')
plt.plot(x,y)
plt.show()

#sigmoid function
def sigmoid(x):
    return 1/(1+np.exp(-x))
x = np.arange(-5.0, 5.0, 0.1)
y = sigmoid(x)

plt.plot(x, y)
plt.plot([0,0],[1.0,0.0], ':') # 가운데 점선 추가
plt.title('Sigmoid Function')
plt.show()
```





Single-Layer Perceptron

- 단층 퍼셉트론(Single-Layer Perceptron)
  - 입력단계(layer)와 출력단계로 구성된 퍼셉트론
  - 입력층(input layer)과 출력층(output layer)으로 구성
  - 입력벡터를 두 부류(0 또는 1)로 분류하는 이진 선형 분류기

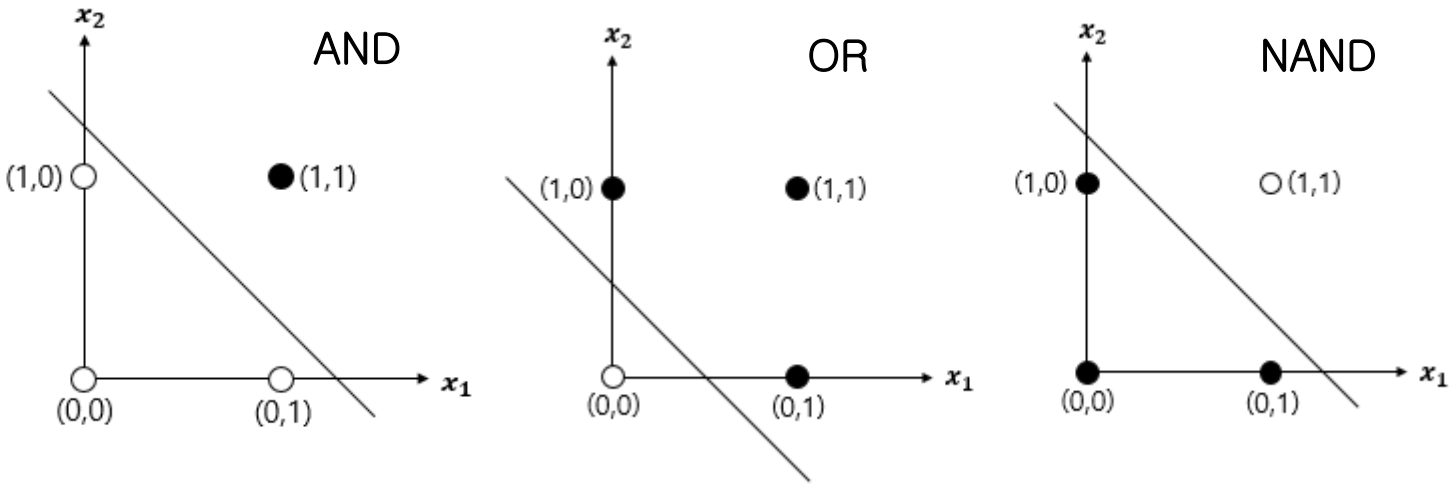


$$f(x) = \begin{cases} 1 & if\ w \cdot x + b \geq 0 \\ 0 & otherwise \end{cases} \qquad w \cdot x = \sum_{i=1}^{n} x_i w_i$$

Single-Layer Perceptron

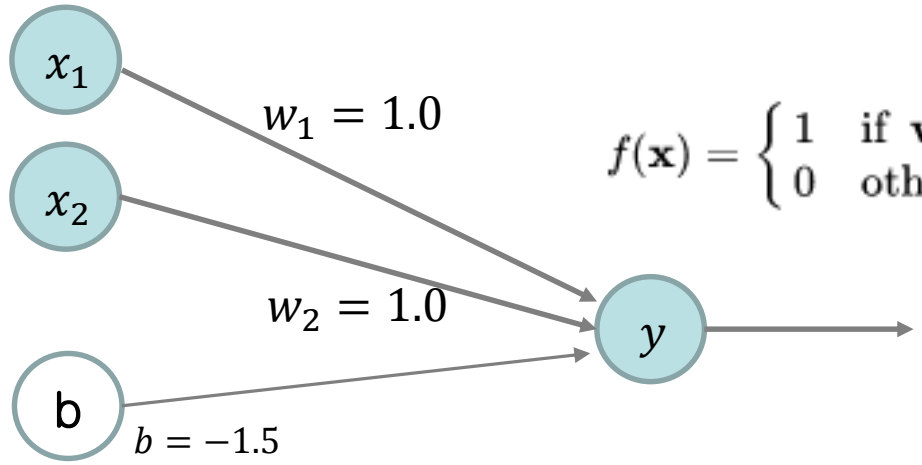
- AND, NAND, OR 연산에 대한 선형 분류 처리

input		output		
x_1	x_2	AND	OR	NAND
0	0	0	0	1
0	1	0	1	1
0	0	0	1	1
1	1	1	1	0

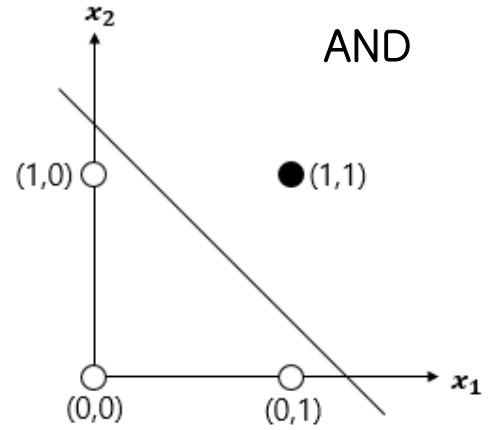


Single-Layer Perceptron 예

- AND 연산을 위한 단층 퍼셉트론
  - 입력이 1,1일 경우만 1 나머지는 0을 출력하는 선형분류



$$f(\mathbf{x}) = \begin{cases} 1 & \text{if } \mathbf{w} \cdot \mathbf{x} + b > 0, \\ 0 & \text{otherwise} \end{cases}$$



입력 (0,1) : $x_1 = 0,\ x_2 = 1$
연결강도(가중치) : $w_1 = 1.0,\ w_2 = 1.0$
바이어스 : $b = -1.5$
출력 : $s = \sum_{i=1}^{2} x_i w_i = 0 * 1.0 + 1 * 1.0 + (-1.5) = -0.5$
$y = \mathrm{f}(\mathrm{s}) = -0.5 < 0 = 0$

입력		출력	
x_1	x_2	s	y
0	0	−1.5	0
0	1	−0.5	0
1	0	−0.5	0
1	1	0.5	1

QUIZ!

문제1. NAND, OR 연산 결과를 출력해 봅시다
수동으로 찾은 가중치와 바이어스를 이용

logical operation	weights		bias
	w_1	w_2	b
NAND	−1	−1	1.5
OR	1	1	−0.5

입력		NAND 출력	
x_1	x_2	s	y
0	0		
0	1		
1	0		
1	1		

입력		OR 출력	
x_1	x_2	s	y
0	0		
0	1		
1	0		
1	1		

Single-Layer Perceptron

- AND, OR 함수 구현

```python
#단층퍼셉트론으로 AND, OR 연산을 위한 함수 구현
# step함수 사용
def step(s):
    if s <= 0:       #step function (0을 기준값으로 작으면 0 크면 1로 출력)
        return 0
    else:
        return 1

def AND_b(x1,x2):
    w1,w2,b = 1,1,-1.5
    s = x1*w1 + x2*w2 + b  #transfer function, weighted sum
    y = step(s)            #activation function, step function call
    return y

def AND(x):
    w = np.array([1,1])
    b = -1.5
    s = np.sum(w*x) + b     #transfer function, weighted sum
    y = step(s)             #activation function, step function call
    return y

def OR(x):
    w = np.array([1,1])
    b = -0.5
    s = np.sum(w*x) + b     #transfer function, weighted sum
    y = step(s)             #activation function, step function call
    return y
```

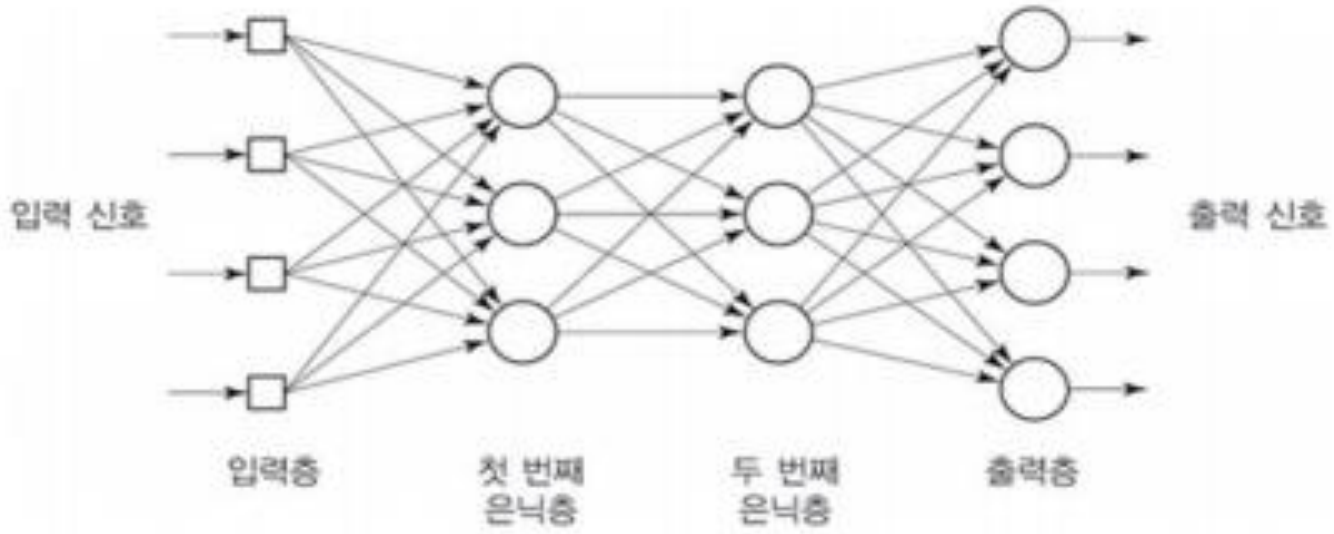
Single-Layer Perceptron

- AND, OR 연산처리

```
#단층퍼셉트론으로 AND, OR, NAND 연산처리

#입력 벡터
inputs = [[0, 0], [0, 1], [1, 0], [1, 1]]
display(inputs)

#입력벡터의 AND 연산 처리
print("AND_b")
for x in inputs:
    y = AND_b(x[0], x[1])
    print('({x}) -> {y}'.format(x=x, y=y))

#입력벡터의 AND 연산 처리
print("AND")
for x in inputs:
    y = AND(x)
    print('({x}) -> {y}'.format(x=x, y=y))

#입력벡터의 OR 연산 처리
print("OR")
for x in inputs:
    y = OR(x)
    print('({x}) -> {y}'.format(x=x, y=y))
```

Multiple Layer Perceptron

- 다층 퍼셉트론(MultiLayer Perceptron, MLP) :
  - 은닉층(hidden layer)을 두어 여러 층으로 구성된 퍼셉트론
  - 입력층으로부터 입력된 신호는 계산되어 첫번째 은닉층의 출력으로 나가고 두번째 은닉층의 입력 값이 되어 계산되고 출력층으로 신호가 전달
  - 심층 신경망(Deep Neural Network, DNN)
    : 복잡한 문제를 해결하기 위해 은닉층을 여러 개 사용
  - 비선형 분류기



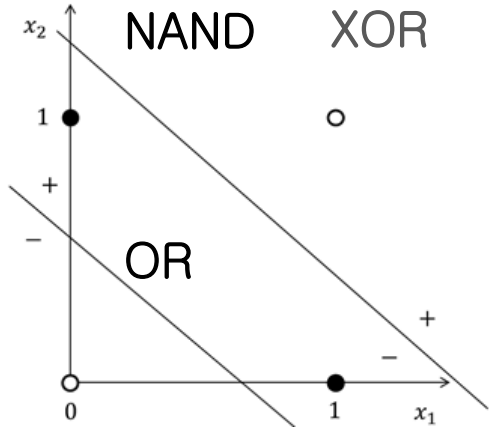
Multiple Layer Perceptron

- XOR 문제 해결을 위한 다층 퍼셉트론
  - NAND, OR, AND 연산을 조합하여 XOR 연산에 대한 비선형 분류 가능

    XOR = AND(NAND,OR)

input		output
x_1	x_2	XOR
0	0	0
0	1	1
1	0	1
1	1	0

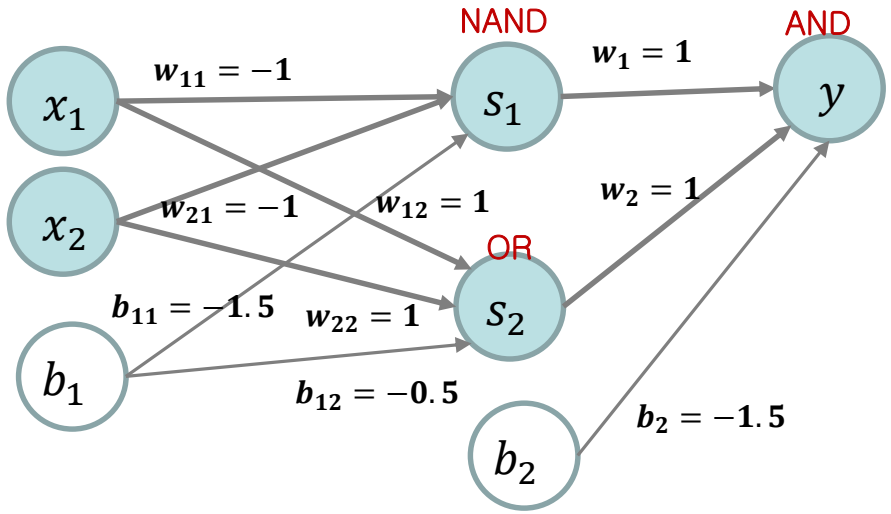
input		hidden		output
x_1	x_2	NAND	OR	AND
0	0	1	0	0
0	1	1	1	1
1	0	1	1	1
1	1	0	1	0



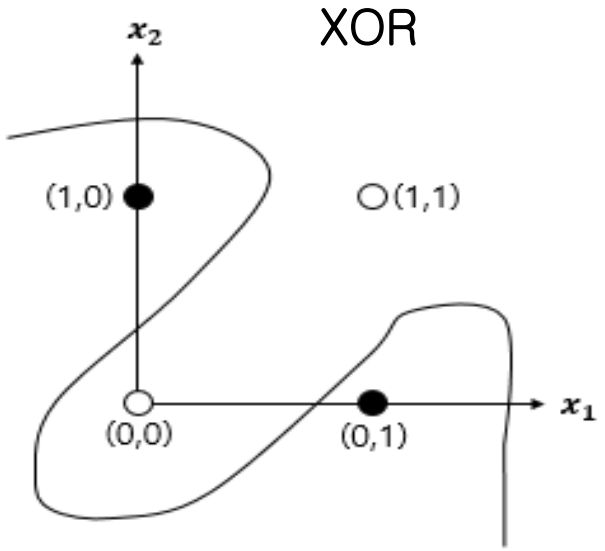
Multiple Layer Perceptron

- XOR 문제 해결을 위한 다층 퍼셉트론
  - NAND, OR, AND 문제 해결을 위해 수동으로 찾은 가중치와 바이어스 사용

logical operation	weights		bias
	w_1	w_2	b
NAND	−1	−1	1.5
OR	1	1	−0.5
AND	1	1	−1.5



Input layer hidden layer output layer



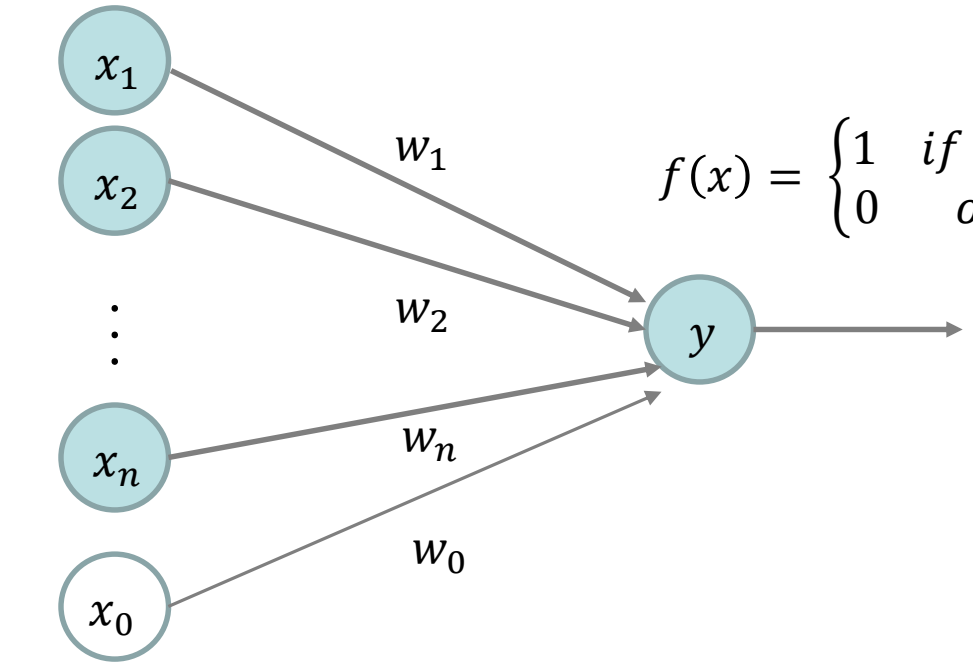
Multiple Layer Perceptron

- XOR 연산처리

```
#XOR 문제 해결을 위한 다층 퍼셉트론
#NAND, OR, AND 연산을 조합하여 XOR 연산처리
print("XOR")
for x in inputs:
    s1 = NAND(x)
    s2 = OR(x)
    s = np.array([s1,s2])
    y = AND(s)
    print('({x}) -> {y}'.format(x=x, y=y))
```

Single-Layer Perceptron 학습

- 입력벡터(학습데이터)를 이용하여 가중치(w_1, w_2), 바이어스(w_0)를 자동으로 찾는 방법



$$f(x) = \begin{cases} 1 & if\ w \cdot x \geq 0 \\ 0 & otherwise \end{cases} \qquad w \cdot x = \sum_{i=1}^{n} x_i w_i$$

Input layer

output layer

Single-Layer Perceptron 학습

- 학습 데이터

  입력벡터(x_i)와 목표값(d_i)으로 구성

  $$D = \{(\mathbf{x}_1, d_1), \ldots, (\mathbf{x}_s, d_s)\}$$

- 학습 알고리즘

  (1)가중치, 임계값(threshold)을 임의의 값으로 초기화 (-0.5~0.5)

  $w_i(t)$: t시점에서의 i번째 입력의 가중치, $w_0(t)$: 바이어스

  (2)입력벡터(학습데이터)에 대한 출력층의 결과값(y) 계산

  $$y_j(t) = f[\mathbf{w}(t) \cdot \mathbf{x}_j]$$
  $$= f[w_0(t)x_{j,0} + w_1(t)x_{j,1} + w_2(t)x_{j,2} + \cdots + w_n(t)x_{j,n}] = \begin{cases} 1 & w(t) \cdot x_j \geq threshold \\ 0 & w(t) \cdot x_j < threshold \end{cases}$$

  (3)출력층 결과와 목표값과 오차를 계산하여 가중치 변경 (학습)

  $$w_i(t+1) = w_i(t) + r \cdot (d_j - y_j(t))x_{j,i}$$

  r : 퍼셉트론의 학습률(Learning rate ,학습속도)

  (4)모든 학습데이터에 대해 (2)반복

  (5)모든 학습데이터가 허용오차보다 작으면 종료

  (6)허용오차가 큰 학습벡터에 대해 (2)반복

Single-Layer Perceptron 학습 예

- AND 연산 학습

(1)학습데이터

순번	입력		목표
t	x_1	x_2	d
1	0	0	0
2	0	1	0
3	1	0	0
4	1	1	1

(2)바이어스 입력 값

$x_0 = -1$

(3) 학습률

$\mathrm{r} = 0.05$

(4)가중치, 임계 값(th) 초기화

$w_0 = 0.3 \;\; w_1 = 0.4, \; w_2 = 0.1$

th = 0

(5-1) t=1 학습

$x_1 = 0, \; x_2 = 0$

$s = \sum_{i=0}^{2} x_i w_i = -1 * 0.3 + 0 \; * 0.4 + 0 \; * 0.1 = -0.3$

$y(1) = f(s) = -0.3 < 0 = 0$

$y(1)$와 $d(1)$ 은 같으므로 가중치 변경 없이 다음 데이터 학습

(5-2) t=2 학습

$x_1 = 0, \; x_2 = 1$

$s = \sum_{i=0}^{2} x_i w_i = -1 * 0.3 + 0 \; * 0.4 + 1 \; * 0.1 = -0.2$

$y(2) = f(s) = -0.2 < 0 = 0$

$y(2)$와 $d(2)$ 은 같으므로 가중치 변경 없이 다음 데이터 학습

Single-Layer Perceptron 학습 예

(5-3) t=3 학습

$x_1 = 1,\ x_2 = 0$

$s = \sum_{i=0}^{2} x_i w_i = -1 * 0.3 + 1 * 0.4 + 0 * 0.1 = 0.1$

$y(3) = f(s) = 0.2 \geq 0 = 1$

$y(3)$와 $d(3)$ 은 다르므로 학습(가중치 변경)

$$w_i(t+1) = w_i(t) + r \cdot (d_j - y_j(t))x_{j,i}$$

$w_0(4) = w_0(3) + 0.05 * (-1) * x_0 = 0.3 + 0.05 = 0.35$

$w_1(4) = w_1(3) + 0.05 * (-1) * x_1 = 0.4 - 0.05 = 0.35$

$w_2(4) = w_2(3) + 0.05 * (-1) * x_2 = 0.1 - 0 = 0.1$

(5-4) t=4 학습

$x_1 = 1,\ x_2 = 1$

$s = \sum_{i=0}^{2} x_i w_i = -1 * 0.35 + 1 * 0.35 + 1 * 0.1 = 0.1$

$y(4) = f(s) = 0.1 \geq 0 = 1$

$y(4)$와 $d(4)$ 는 같으므로 가중치 변경 없이 다음단계

Single-Layer Perceptron 학습 예

(6) t = 3 오차가 있으므로 학습

$x_1 = 1,\ x_2 = 0\ , \mathrm{d} = 0$

$s = \sum_{i=0}^{2} x_i w_i = -1 * 0.35 + 1\ * 0.35 + 0\ * 0.1 = 0.1$

$y(3)\ = f(s) = 0.1 \geq 0 = 1$

$y(3)$와 $d(3)$ 은 다르므로 학습(가중치 변경)

$$w_0(4) = w_0(3) + 0.05\ * (-1) * x_0 = 0.35 + 0.05 = 0.4$$
$$w_1(4) = w_1(3) + 0.05\ * (-1) * x_1 = 0.35 - 0.05 = 0.3$$
$$w_2(4) = w_2(3) + 0.05\ * (-1) * x_2 = 0.1 + 0 =\ 0.1$$

(7) t = 3 오차 확인

$x_1 = 1,\ x_2 = 0\ , \mathrm{d} = 0$

$s = \sum_{i=0}^{2} x_i w_i = -1 * 0.4 + 1\ * 0.3 + 0\ * 0.1 = -0.1$

$y(3)\ = f(s) = -0.1 < 0 = 0$

$y(3)$와 $d(3)$ 은 같으므로 다음단계

Single-Layer Perceptron 학습 예

(8) t = 4 오차 확인

$x_1 = 1,\ x_2 = 0\ , \mathrm{d} = 0$

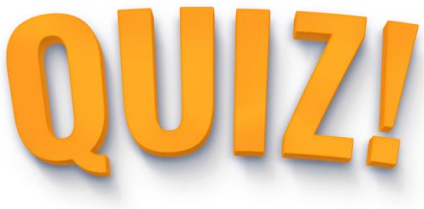
$s = \sum_{i=0}^{2} x_i w_i = -1 * 0.4 + 1\ * 0.3 + 1\ * 0.1 = 0$

$y(4) = f(s) = 0 \geq 0 = 1$

$y(4)$와 $d(4)$ 은 같으므로 학습 종료

(9)학습 후 최종 가중치

$w_0 = 0.4,\ w_1 = 0.3\ ,\ w_2 = \ 0.1$



OR, NAND 연산 학습을 통해 가중치 값을 생성해보세요.