밑바닥부터 시작하는 딥러닝

CH3. 신경망

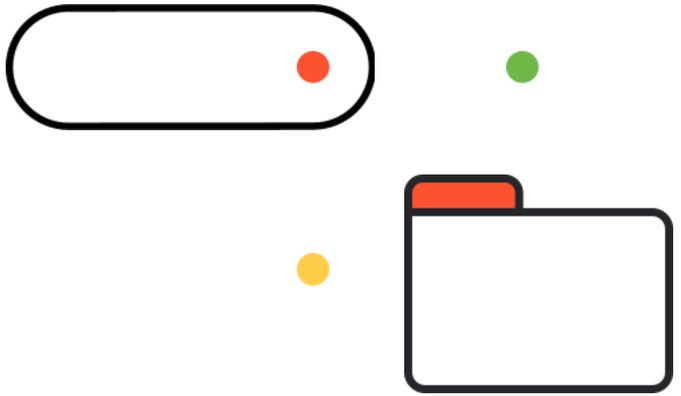
INDEX

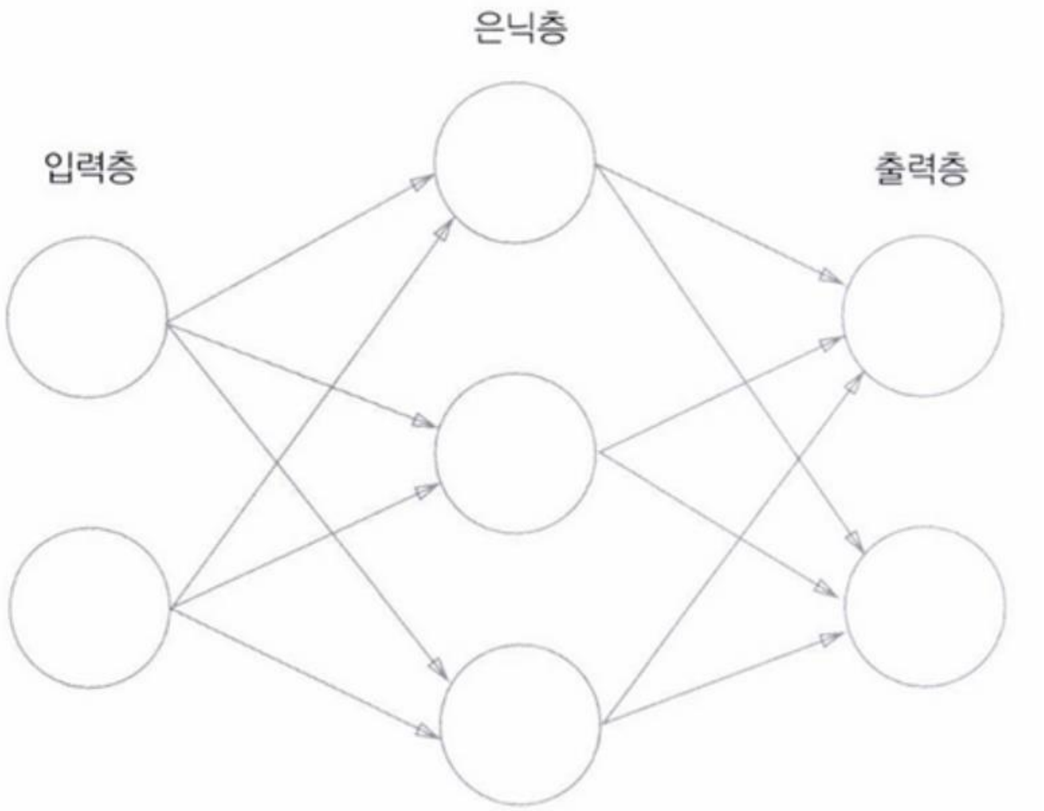
목차

presentation_title

01

신경망





presentation_photo

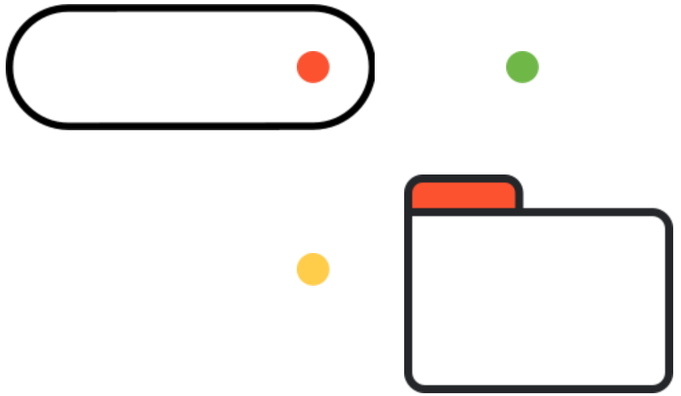
신경망의 구조

입력층, 출력층, 그리고 은닉층

여러개의 뉴런들의 복합적 구조
출력층 노드의 수는 클래스의 개수
은닉층으로 인한 복잡한 분류의 가능

02

활성화 함수



퍼셉트론과 신경망의 주된 차이

$$a = b + w_1 x_1 + w_2 x_2$$

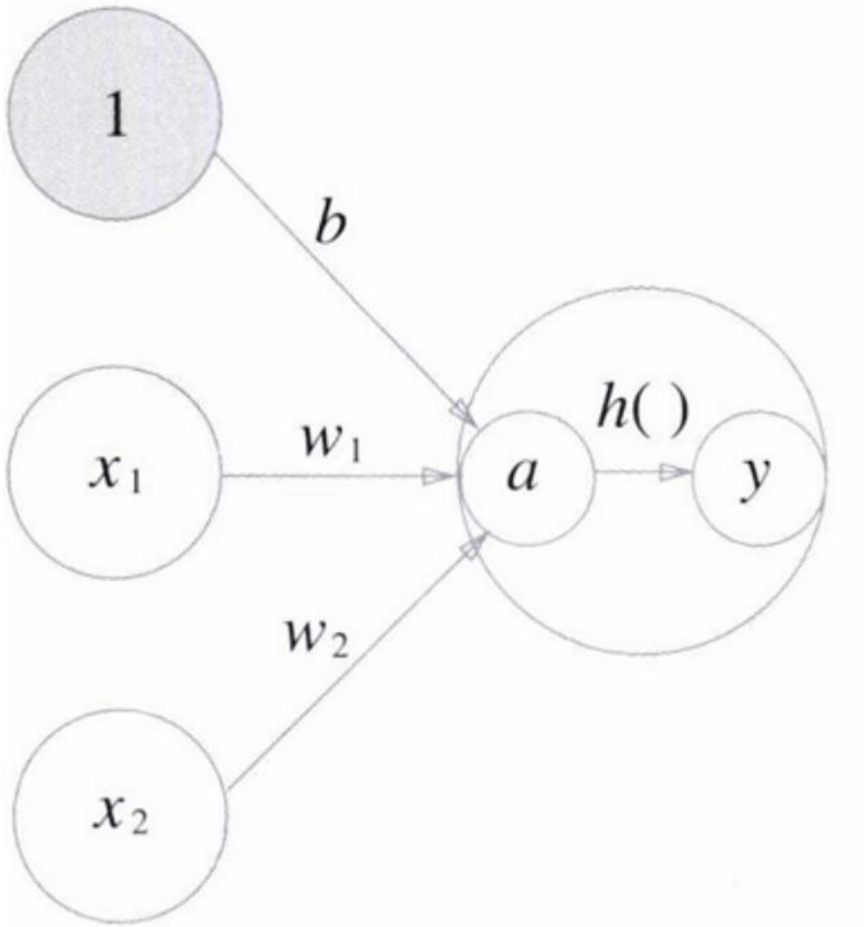
$$y = h(a)$$

presentation_photo

활성화 함수의 처리 과정

가중합과 비선형함수

입력 신호의 총합을 출력 신호로 변환하는 함수
입력 신호의 총합이 활성화를 일으키는지 결정



presentation_photo

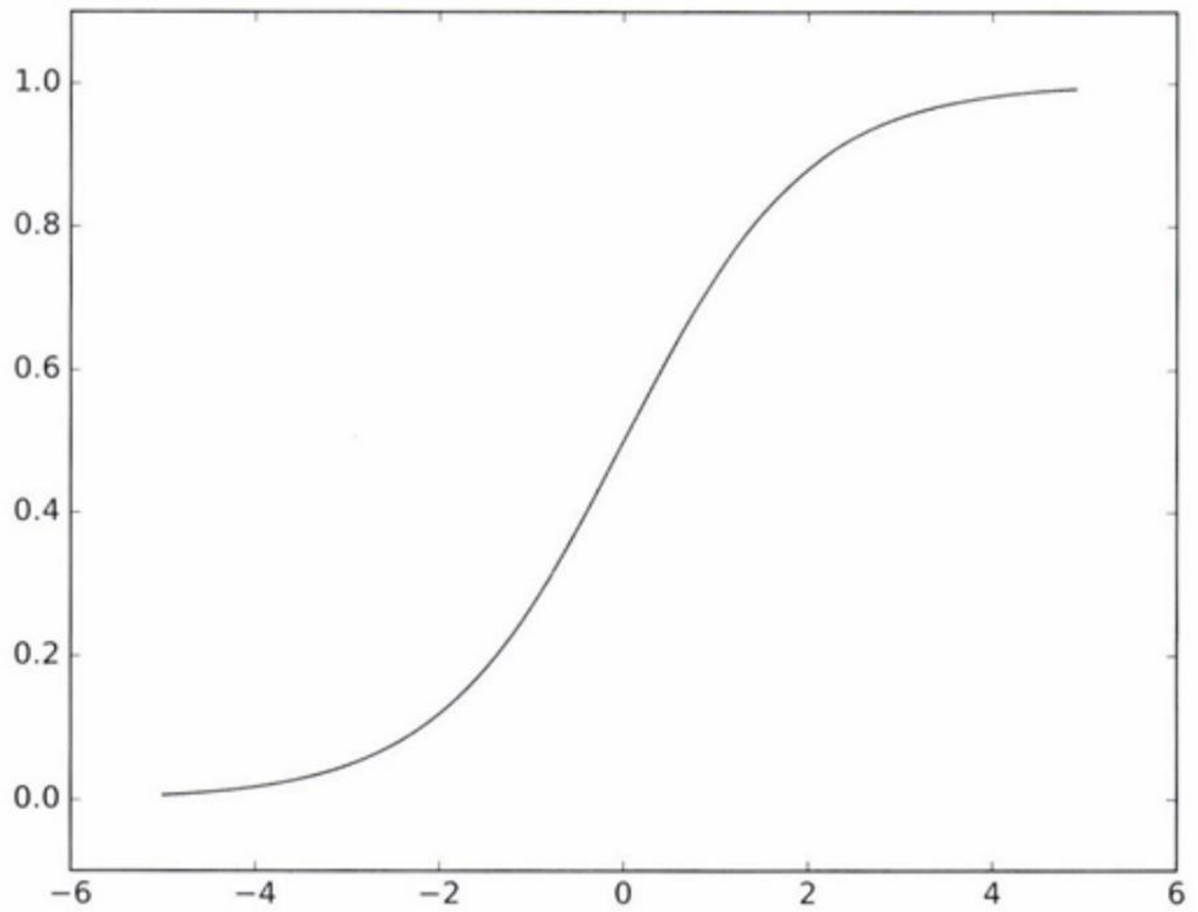
활성화 함수의 처리 과정

가중합과 비선형함수

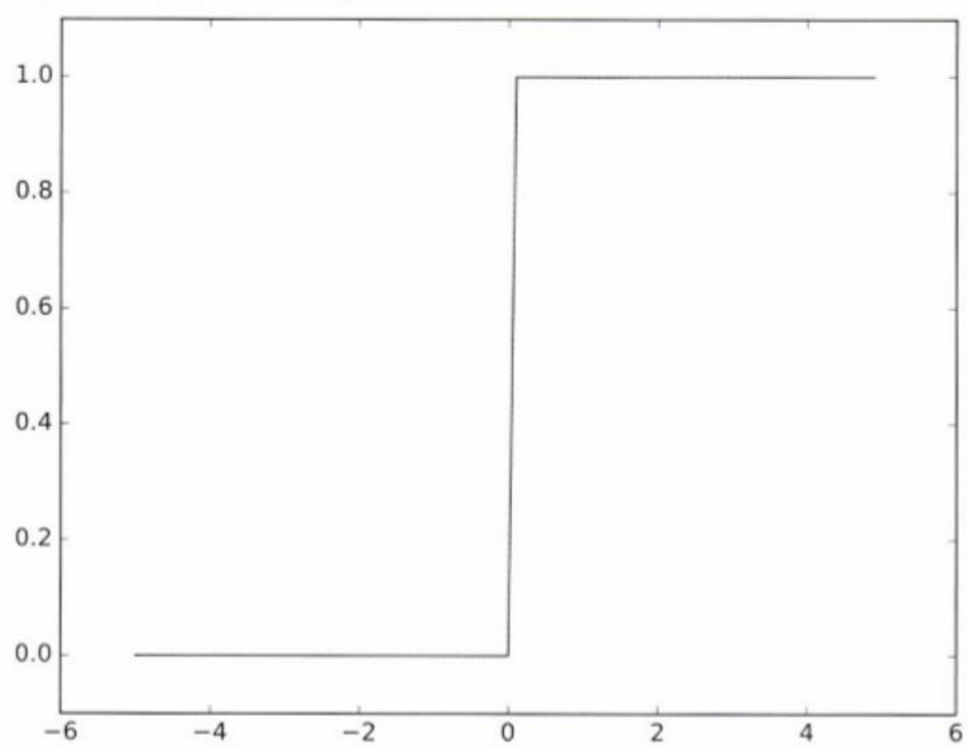
입력 신호의 총합을 출력 신호로 변환하는 함수
입력 신호의 총합이 활성화를 일으키는지 결정

시그모이드 함수

$$h(x) = \frac{1}{1 + \exp(-x)}$$



계단 함수



ReLU 함수

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

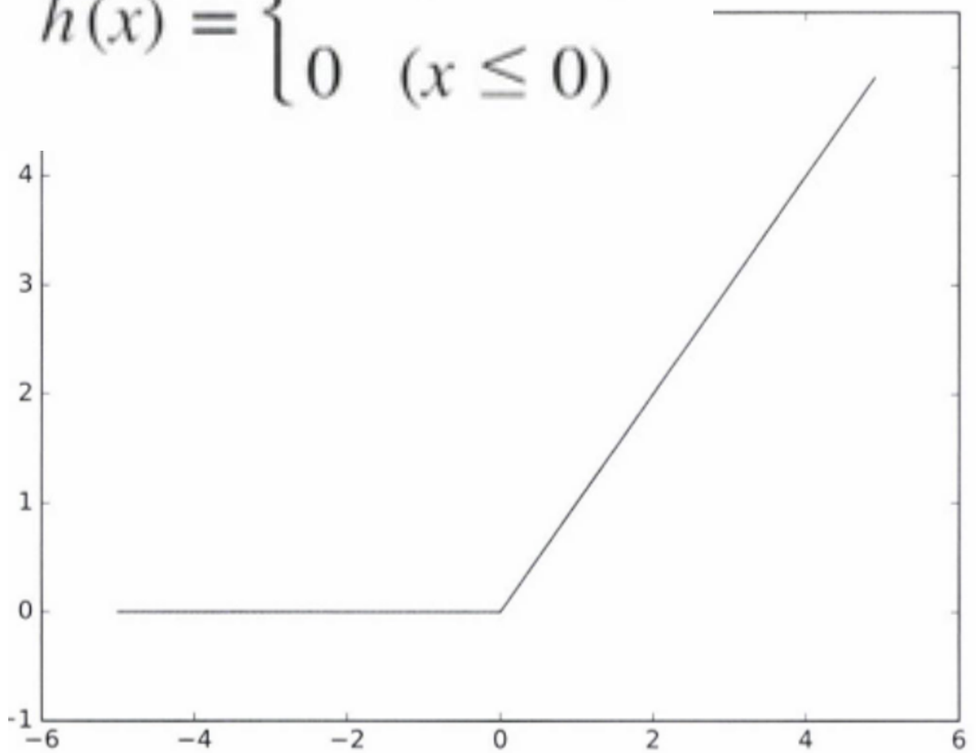
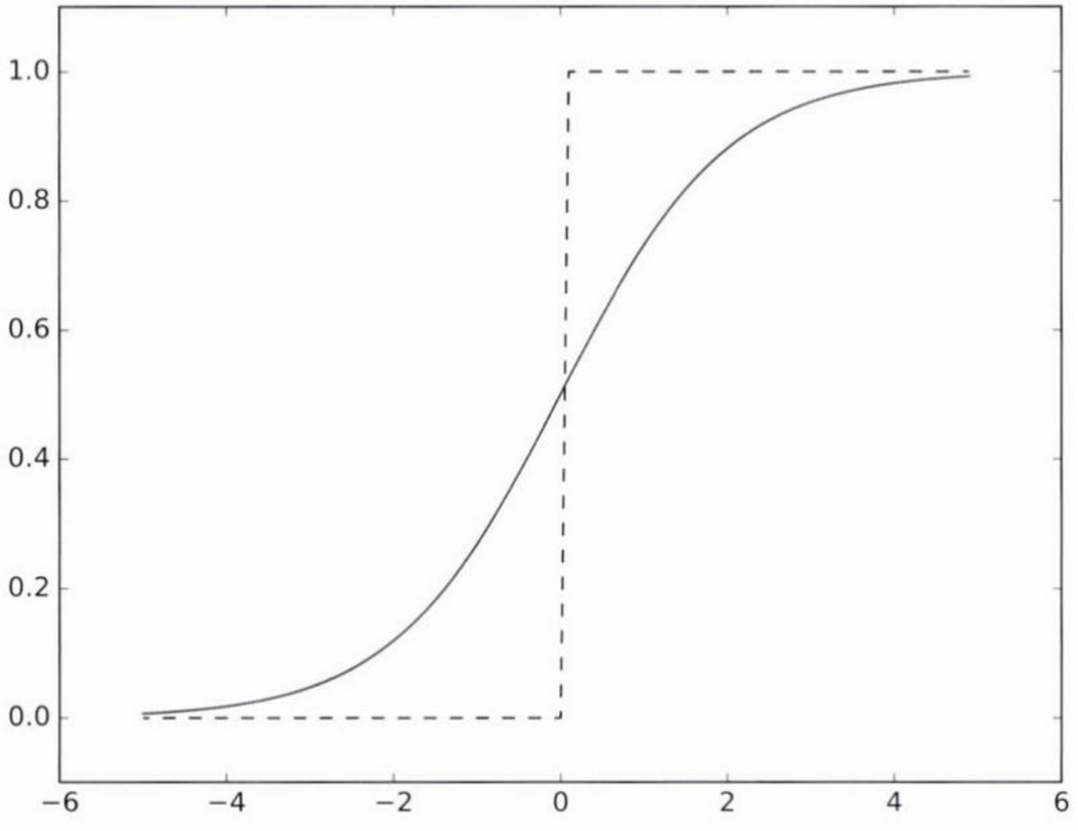


그림 3-8 계단 함수(점선)와 시그모이드 함수(실선)



계단 함수 vs 시그모이드 함수

공통점 - 출력 0에서 1 사이
- 입력이 작을 때는 출력이 0에 가깝거나 0이고 입력이 커지면 출력이 1에 가까워지거나 1 (같은 모양)
- 비선형 함수

차이점 - 계단 함수 : 0과 1 중 하나의 값
- 시그모이드 함수 : 연속적인 실수

선형 함수 : f(x) = ax + b
비선형 함수 : 직선 1개로는 그릴 수 없는 함수

신경망 활성화 함수 -> 비선형 함수
: 선형 함수를 이용하면 신경망의 층을 깊게 하는 의미가 없어짐.

$h(x) = cx$를 활성화 함수로 사용한 3층 네트워크

$y(x) = h(h(h(x)))$

$y(x) = c * c * c * x$

$y(x) = ax$와 똑같은 식 $a = c^3$

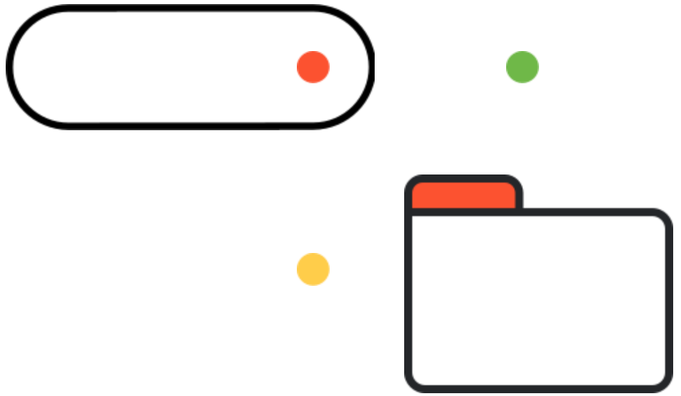
계단 함수 vs 시그모이드 함수

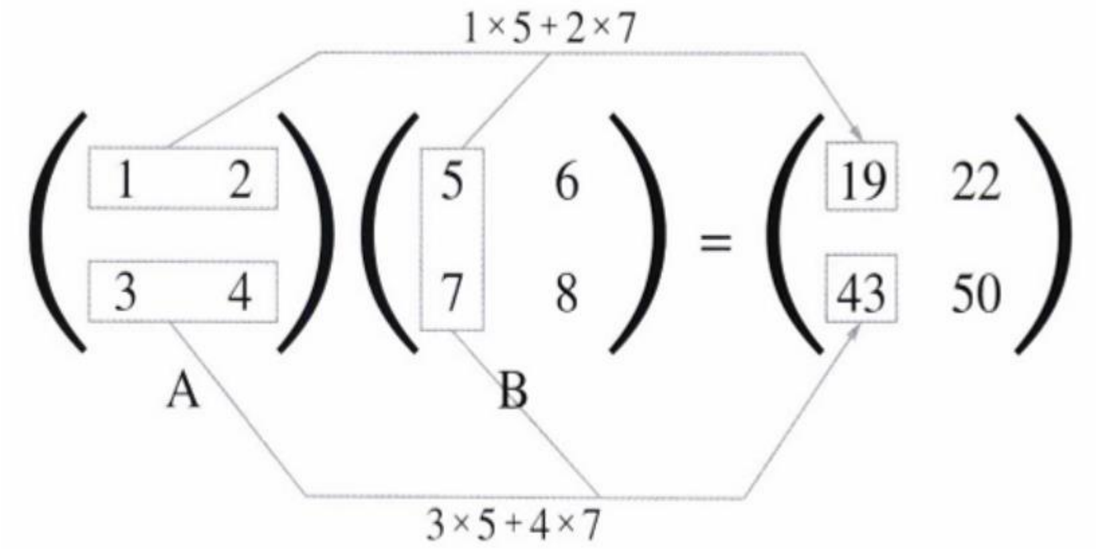
공통점
- 출력 0에서 1 사이
- 입력이 작을 때는 출력이 0에 가깝거나 0이고 입력이 커지면 출력이 1에 가까워지거나 1 (같은 모양)
- 비선형 함수

차이점
- 계단 함수 : 0과 1 중 하나의 값
- 시그모이드 함수 : 연속적인 실수

03

다차원 배열의 계산



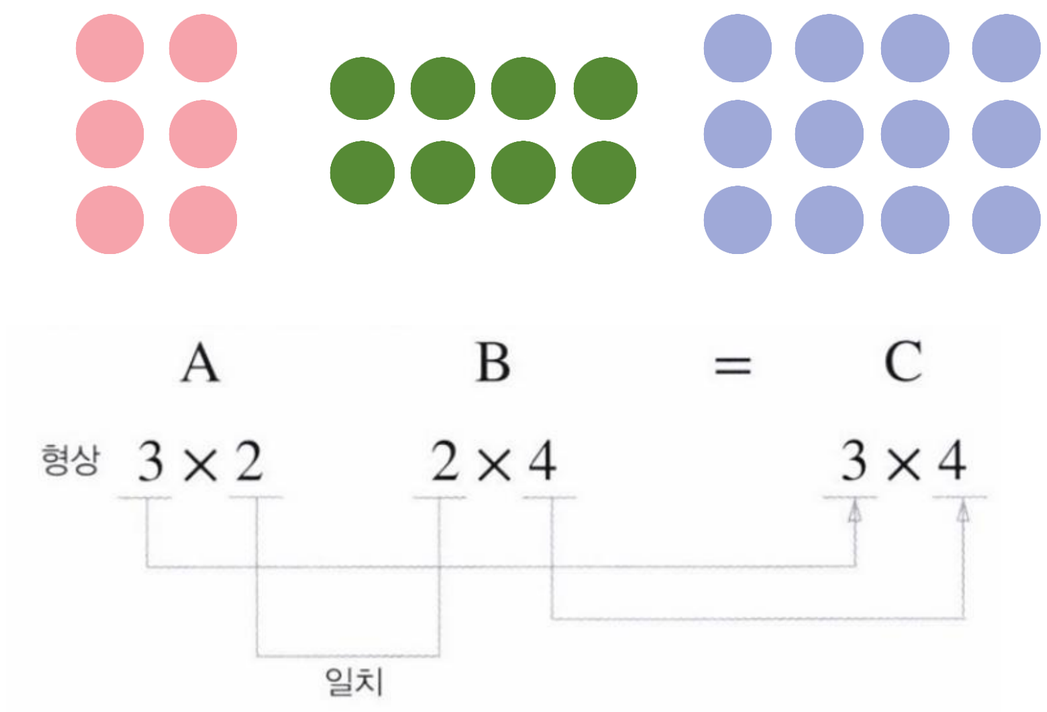


presentation_photo

행렬곱

기본 연산

앞행렬의 열수와 뒷행렬의 행수가 같아야함
가중합의 연산에서 기본이 되는 연산

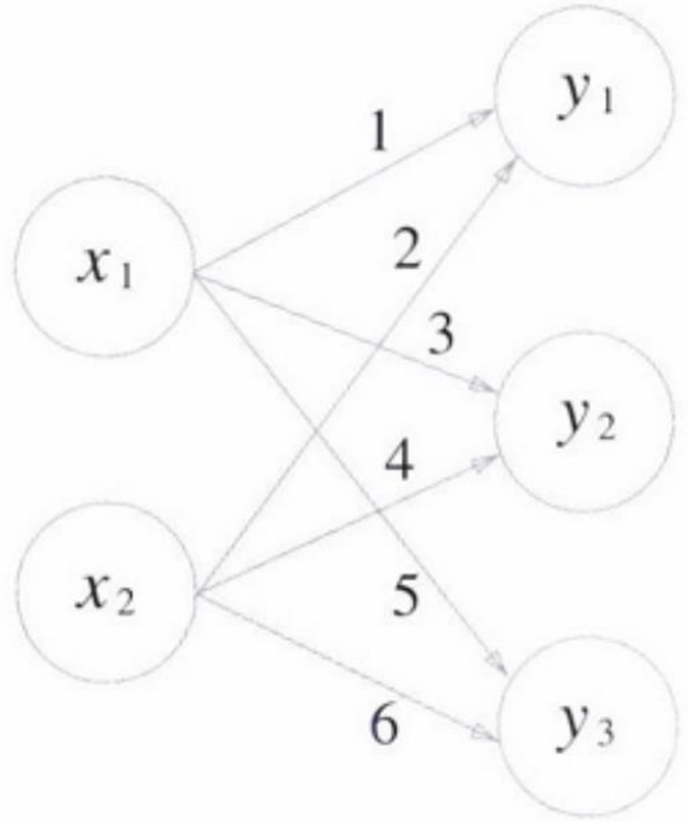


presentation_photo

행렬곱

기본 연산

앞행렬의 열수와 뒷행렬의 행수가 같아야함
가중합의 연산에서 기본이 되는 연산



$$\begin{pmatrix} 1 & 3 & 5 \\ 2 & 4 & 6 \end{pmatrix}$$

$$X \quad W = Y$$

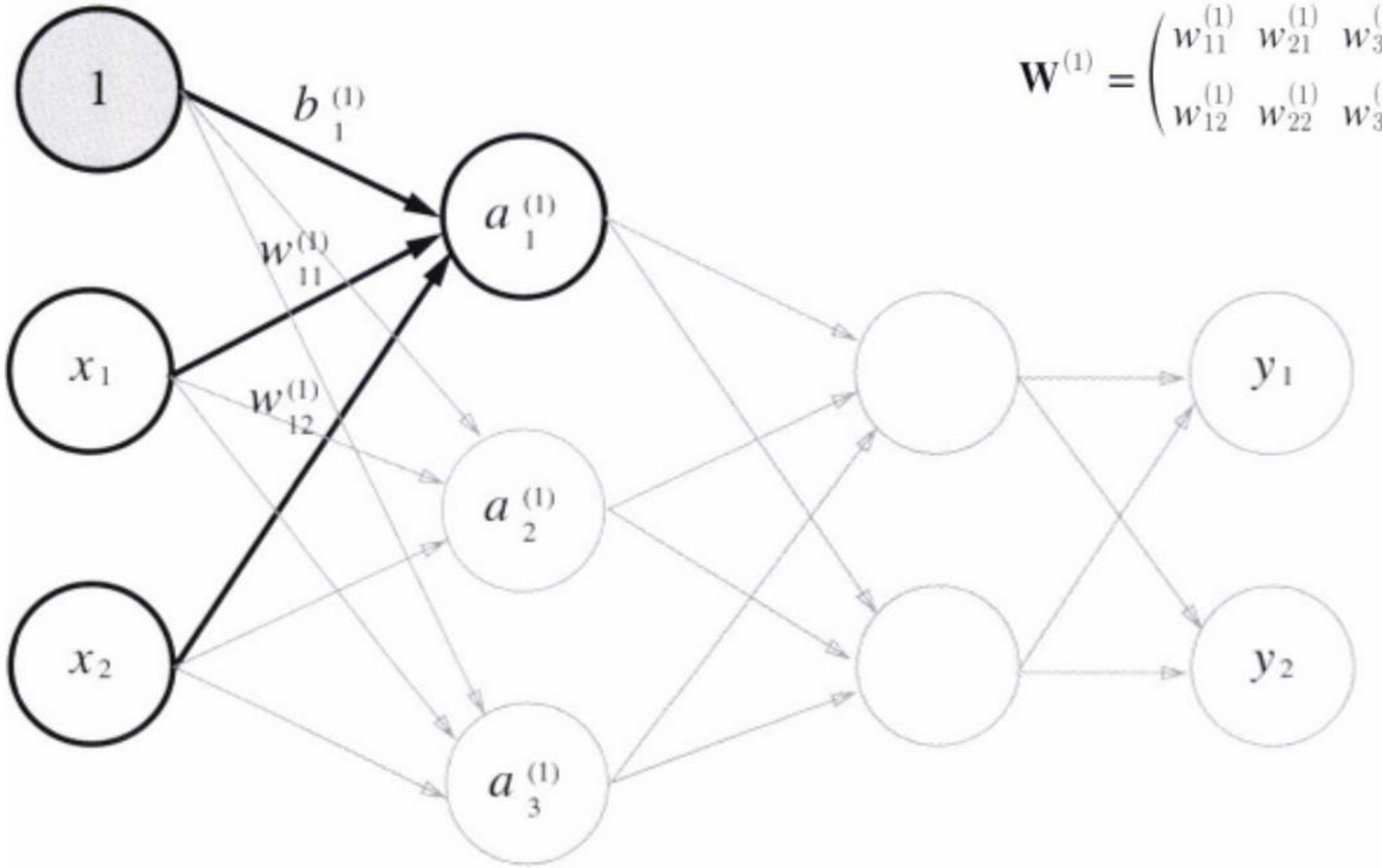
$$2 \quad 2 \times 3 \quad 3$$

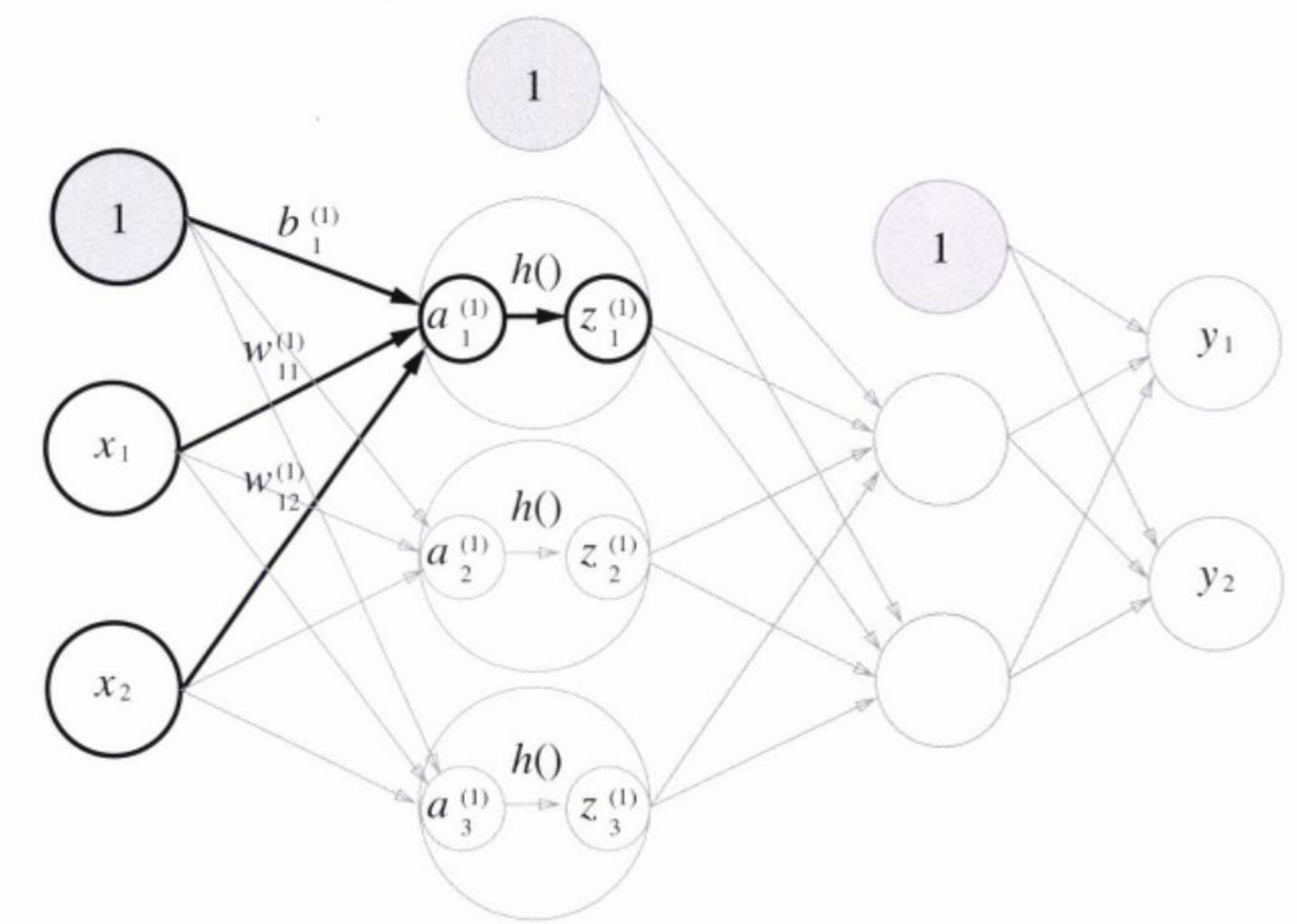
일치

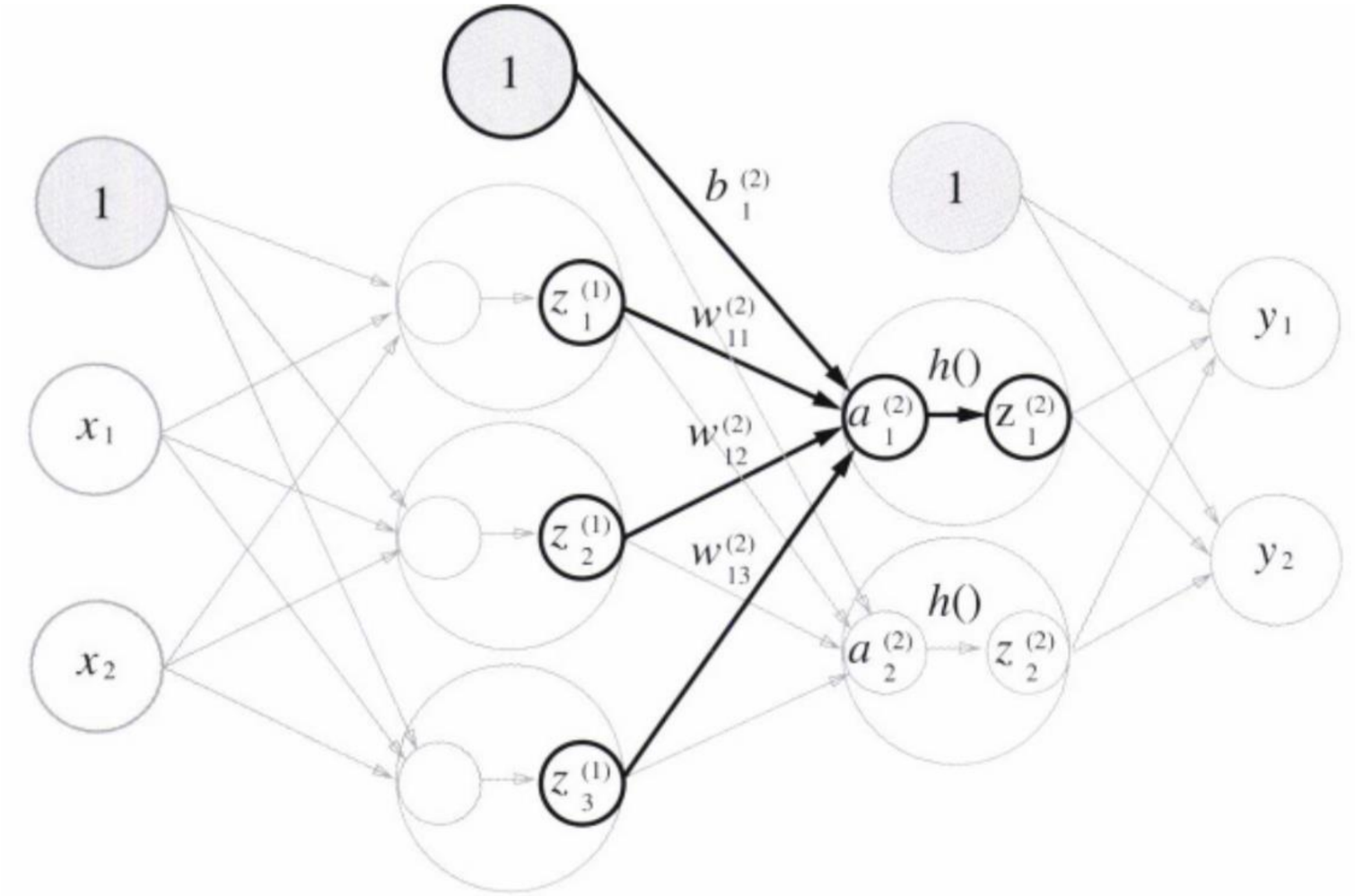
$$\mathbf{A}^{(1)} = \mathbf{X}\mathbf{W}^{(1)} + \mathbf{B}^{(1)}$$

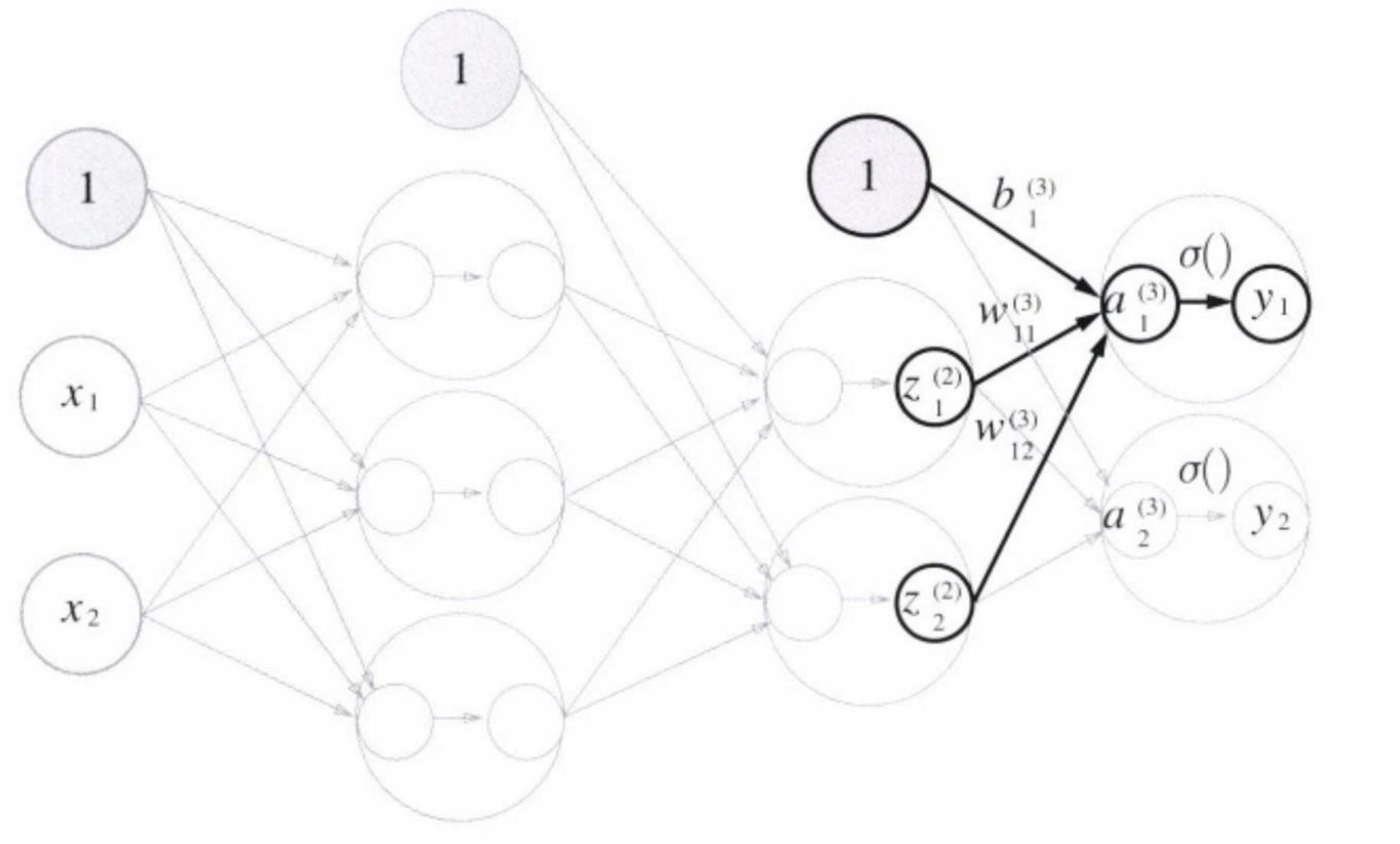
$$\mathbf{A}^{(1)} = \left(a_1^{(1)}\ a_2^{(1)}\ a_3^{(1)}\right),\ \mathbf{X} = \left(x_1\ x_2\right),\ \mathbf{B}^{(1)} = \left(b_1^{(1)}\ b_2^{(1)}\ b_3^{(1)}\right)$$

$$\mathbf{W}^{(1)} = \begin{pmatrix} w_{11}^{(1)} & w_{21}^{(1)} & w_{31}^{(1)} \\ w_{12}^{(1)} & w_{22}^{(1)} & w_{32}^{(1)} \end{pmatrix}$$




1
1
$b^{(1)}_1$
x_1
x_2
$w^{(1)}_{11}$
$w^{(1)}_{12}$
h()
$a^{(1)}_1$
$z^{(1)}_1$
h()
$a^{(1)}_2$
$z^{(1)}_2$
h()
$a^{(1)}_3$
$z^{(1)}_3$
1
y_1
y_2


1
1
x_1
x_2
$b^{(2)}_1$
$z^{(1)}_1$
$z^{(1)}_2$
$z^{(1)}_3$
$w^{(2)}_{11}$
$w^{(2)}_{12}$
$w^{(2)}_{13}$
1
h()
$a^{(2)}_1$
$z^{(2)}_1$
h()
$a^{(2)}_2$
$z^{(2)}_2$
y_1
y_2


1
1
x_1
x_2
1
$b^{(3)}_1$
$w^{(3)}_{11}$
$w^{(3)}_{12}$
$z^{(2)}_1$
$z^{(2)}_2$
$\sigma()$
$a^{(3)}_1$
y_1
$\sigma()$
$a^{(3)}_2$
y_2

```python
def init_network():
    network = {}
    network['W1'] = np.array([[0.1, 0.3, 0.5], [0.2, 0.4, 0.6]])
    network['b1'] = np.array([0.1, 0.2, 0.3])
    network['W2'] = np.array([[0.1, 0.4], [0.2, 0.5], [0.3, 0.6]])
    network['b2'] = np.array([0.1, 0.2])
    network['W3'] = np.array([[0.1, 0.3], [0.2, 0.4]])
    network['b3'] = np.array([0.1, 0.2])

    return network

def forward(network, x):
    W1, W2, W3 = network['W1'], network['W2'], network['W3']
    b1, b2, b3 = network['b1'], network['b2'], network['b3']

    a1 = np.dot(x, W1) + b1
    z1 = sigmoid(a1)
    a2 = np.dot(z1, W2) + b2
    z2 = sigmoid(a2)
    a3 = np.dot(z2, W3) + b3
    y = identity_function(a3)

    return y

def identity_function(x):
    return x

network = init_network()
x = np.array([1.0, 0.5])
y = forward(network, x)
print(y) # [ 0.31682708  0.69627909]
```

presentation_photo

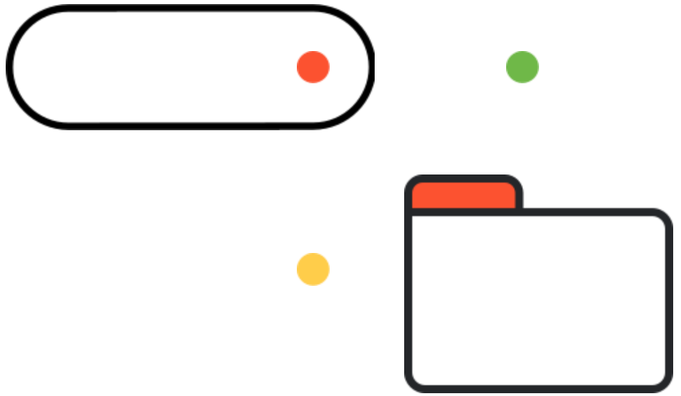
신경망 코드

3층 신경망

init_network()를 통한 초기값 설정
forward(network, x)를 통한 순전파 출력값 연산

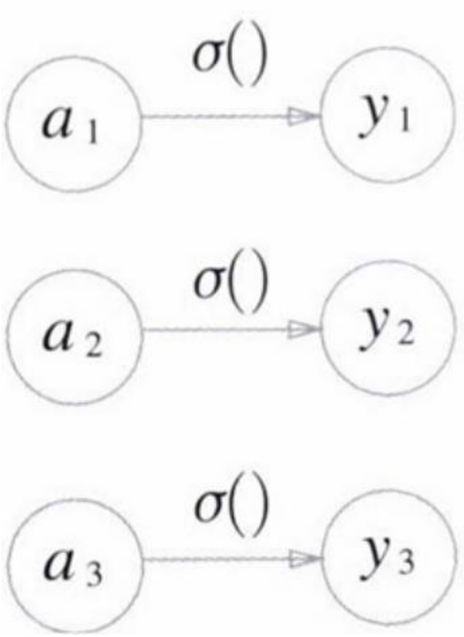
04

출력층 설계하기



일반적으로 회귀에는 항등함수, 분류에는 소프트맥스 함수 사용

항등 함수



소프트맥스 함수

$$y_k = \frac{\exp(a_k)}{\sum_{i=1}^{n}\exp(a_i)} = \frac{C\exp(a_k)}{C\sum_{i=1}^{n}\exp(a_i)}$$

$$= \frac{\exp(a_k + \log C)}{\sum_{i=1}^{n}\exp(a_i + \log C)}$$

$$= \frac{\exp(a_k + C')}{\sum_{i=1}^{n}\exp(a_i + C')}$$

구현 시 오버플로 문제
(큰 값 -> 결과 수치 '불안정')

어떤 정수를 더하거나 빼도 결과 동일

C' : 일반적으로 입력 신호 중 최댓값 이용

```
def softmax(a):
    c = np.max(a)
    exp_a = np.exp(a - c) # 오버플로 대책
    sum_exp_a = np.sum(exp_a)
    y = exp_a / sum_exp_a

    return y
```

소프트맥스 함수

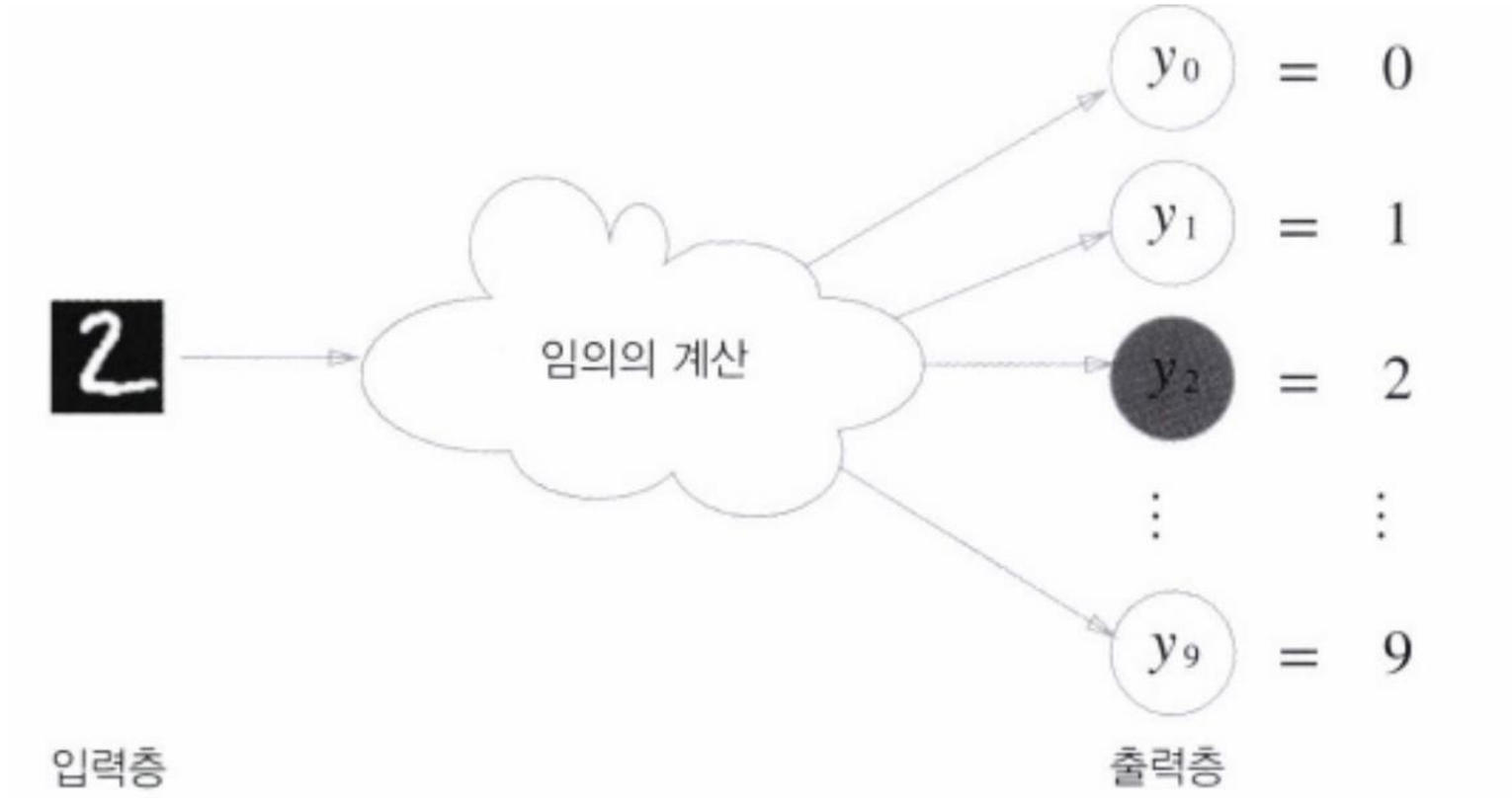
출력은 0에서 1 사이의 실수
출력의 총합은 1

'확률'

지수 함수 : 단조 증가 함수 -> 적용해도 각 원소의 대소 관계 동일

->> 추론 단계에서 출력층의 소프트맥스 함수는 생략 가능

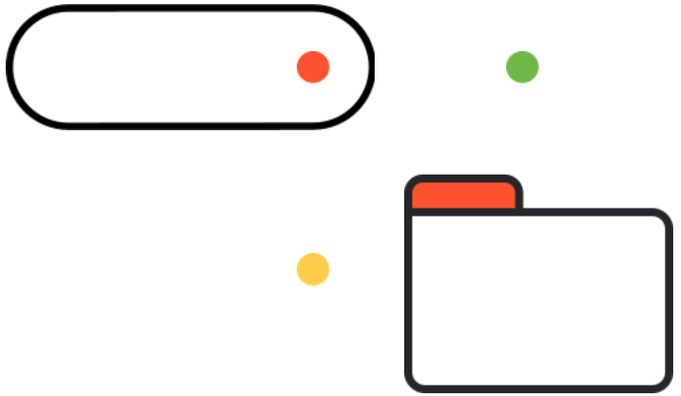
출력층 노드의 수는 분류하고 싶은 클래스의 수



뉴런의 회색 농도가 해당 뉴런의 출력 값의 크기 의미

05

손글씨 숫자 인식



presentation_photo

MNIST 이미지 데이터셋

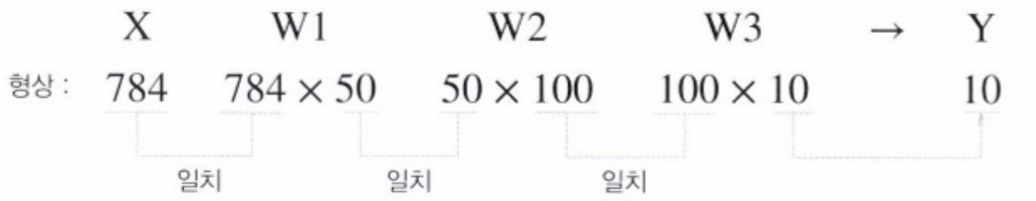
손글씨 데이터셋

60000개의 훈련이미지
10000개의 테스트 이미지
정답에 해당하는 레이블이 있는 지도학습 (분류)

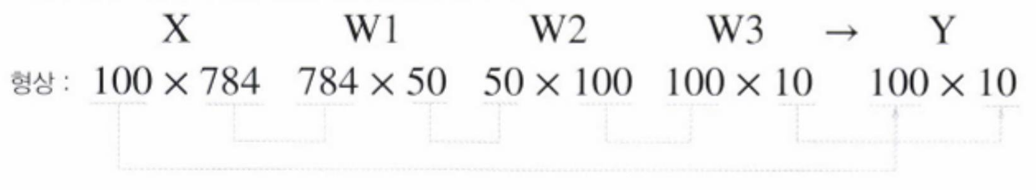
x의 형상

```
>>> x.shape
(10000, 784)
>>> x[0].shape
(784,)
>>> W1.shape
(784, 50)
>>> W2.shape
(50, 100)
>>> W3.shape
(100, 10)
```

이미지 1개



이미지 100개



배치: 하나로 묶은 입력 데이터

```
def get_data():
    (x_train, t_train), (x_test, t_test) = \
        load_mnist(normalize=True, flatten=True, one_hot_label=False)
    return x_test, t_test

def init_network():
    with open("sample_weight.pkl", 'rb') as f:
        network = pickle.load(f)

    return network

def predict(network, x):
    W1, W2, W3 = network['W1'], network['W2'], network['W3']
    b1, b2, b3 = network['b1'], network['b2'], network['b3']

    a1 = np.dot(x, W1) + b1
    z1 = sigmoid(a1)
    a2 = np.dot(z1, W2) + b2
    z2 = sigmoid(a2)
    a3 = np.dot(z2, W3) + b3
    y = softmax(a3)

    return y
```

presentation_photo

신경망으로 추론(분류)

MNIST 손글씨 데이터셋

normalize=T를 통한 정규화 (0~1)
flatten=T를 통한 일차원 배열로의 전환
one_hot_label=T를 하면 클래스에 해당하는 컬럼이 원핫인코딩됨 (열개 컬럼으로)

#입력층 뉴런 784개 (이미지 크기 28X28)
#출력층 뉴런 10개(0~9)
첫번째 은닉층 50개뉴런, 두번째 은닉층 100개 뉴런

Thank you