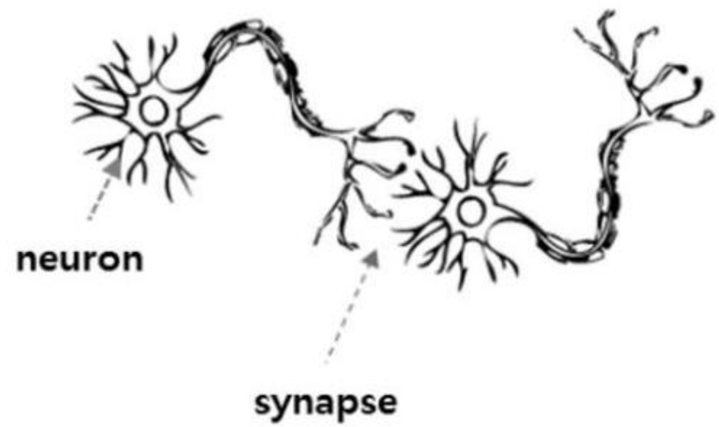
신경망의 이해

2019.11

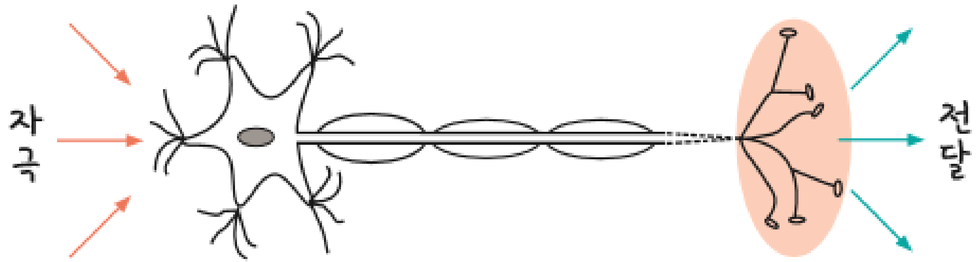
Deep Learning with Keras

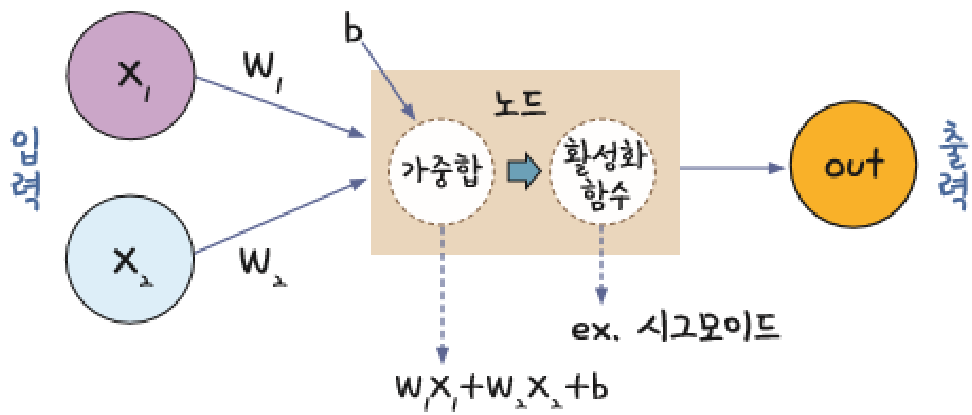
■ 뉴런(Neuron)이란?



- 인간의 뇌는 치밀하게 연결된 약 1000억 개의 뉴런으로 이루어져 있음
- 뉴런과 뉴런 사이에는 시냅스라는 연결 부위가 있는데, 신경 말단에서 자극을 받으면 시냅스에서 화학 물질이 나와 전위 변화를 일으킴
- 전위가 임계 값을 넘으면 다음 뉴런으로 신호를 전달하고, 임계 값에 미치지 못하면 아무것도 하지 않음 → 퍼셉트론의 개념과 유사!

■ 뉴런과 퍼셉트론의 비교





- 신경망을 이루는 가장 중요한 기본 단위는 퍼셉트론(perceptron)
- 퍼셉트론은 입력 값과 활성화 함수를 사용해 출력 값을 다음으로 넘기는 가장 작은 신경망 단위

■ 가중치, 가중합, 바이어스, 활성화 함수

❖ 용어 정리

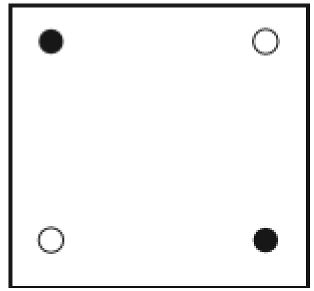
- 딥러닝답게 표현

$$y = ax + b \ (a\text{는 기울기},\ b\text{는 } y \text{ 절편})$$

$$\rightarrow\ y = wx + b \ (w\text{는 가중치},\ b\text{는 바이어스})$$

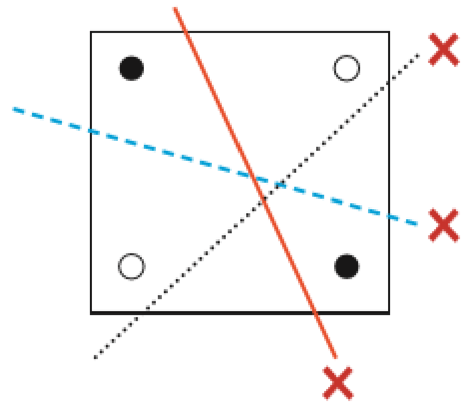
- 기울기 a는 **가중치**를 의미하는 w(weight)로 표기됨
- y 절편 b는 편향, 선입견이라는 뜻인 **바이어스**(bias)에서 따온 b
- **가중합**(weighted sum): 입력 값(x)과 가중치(w)의 곱을 모두 더한 다음 거기에 바이어스(b)를 더한 값
- 가중합의 결과를 놓고 1 또는 0을 출력해서 다음 단계로 보낼때 0과 1을 판단하는 함수가 있는데, 이를 **활성화 함수**(activation function) 라고 함.

■ 퍼셉트론의 과제



- 사각형 종이에 검은점 두 개와 흰점 두 개가 놓여 있음
- 이 네 점 사이에 직선을 하나 긋는다고 하자
- 이때 직선의 한쪽 편에는 검은점만 있고, 다른 한쪽에는 흰점만 있게끔 선을 그을 수 있을까?

■ 퍼셉트론의 과제



- 여러 개의 선을 아무리 그어보아도 하나의 직선으로는 흰점과 검은점을 구분할 수 없음

- 선형 회귀와 로지스틱 회귀를 통해 알아본 바와 같이 머신러닝은 결국 선이나 2차원 평면을 그리는 작업

■ XOR 문제

AND 진리표

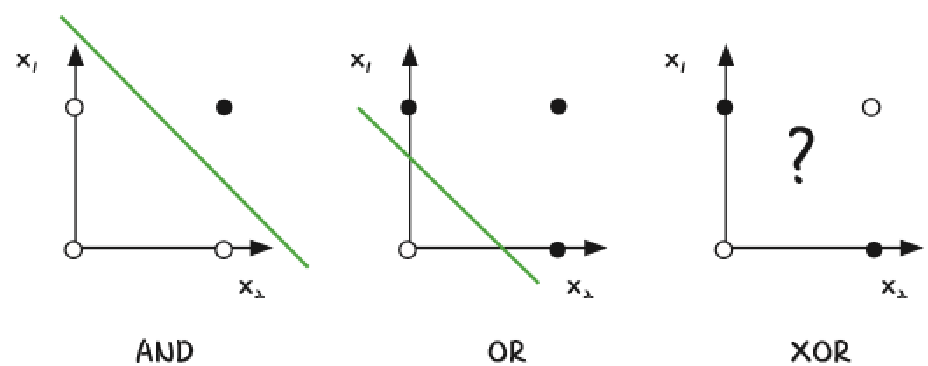
x_1	x_2	결괏값
0	0	0
0	1	0
1	0	0
1	1	1

OR 진리표

x_1	x_2	결괏값
0	0	0
0	1	1
1	0	1
1	1	1

XOR 진리표

x_1	x_2	결괏값
0	0	0
0	1	1
1	0	1
1	1	0



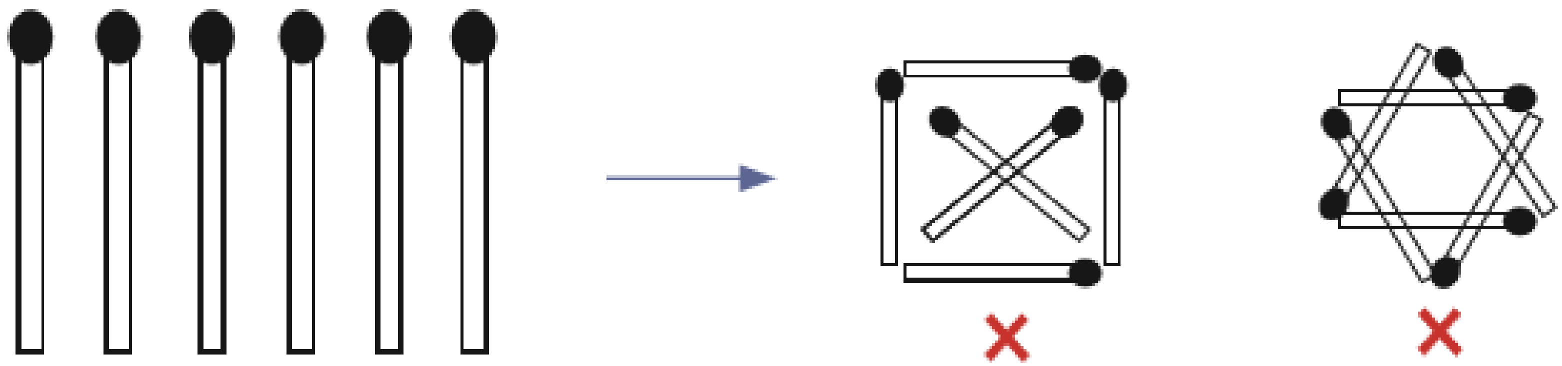
■ XOR 문제

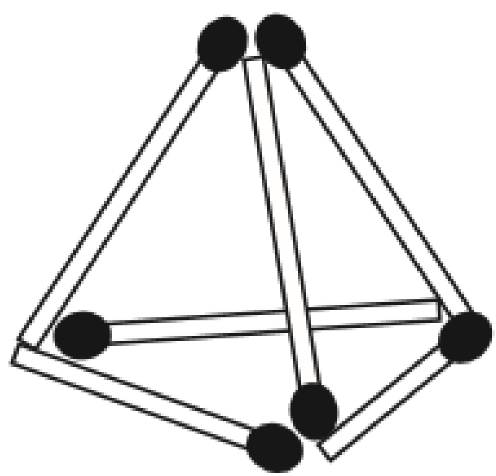
- 이는 인공지능 분야의 선구자였던 MIT의 마빈 민스키(Marvin Minsky) 교수가 1969년에 발표한 <퍼셉트론즈(Perceptrons)>라는 논문에 나오는 내용

- 10여 년이 지난 후에야 이 문제가 해결되는데, 이를 해결한 개념이 바로 다층 퍼셉트론(multilayer perceptron)

■ 다층 퍼셉트론

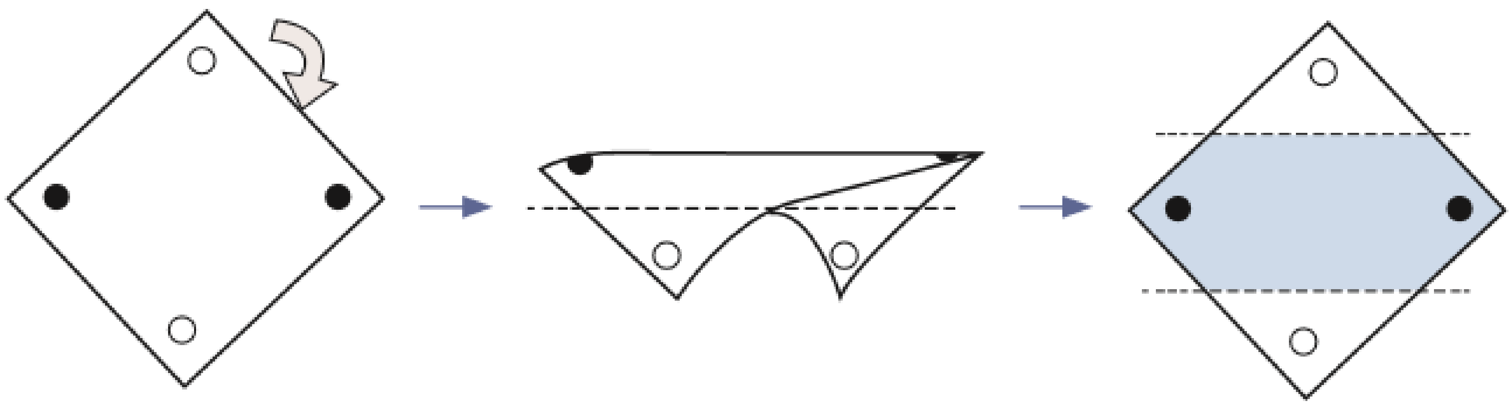
- 성냥개비 여섯 개로 정삼각형 네 개를 만들 수 있는가?





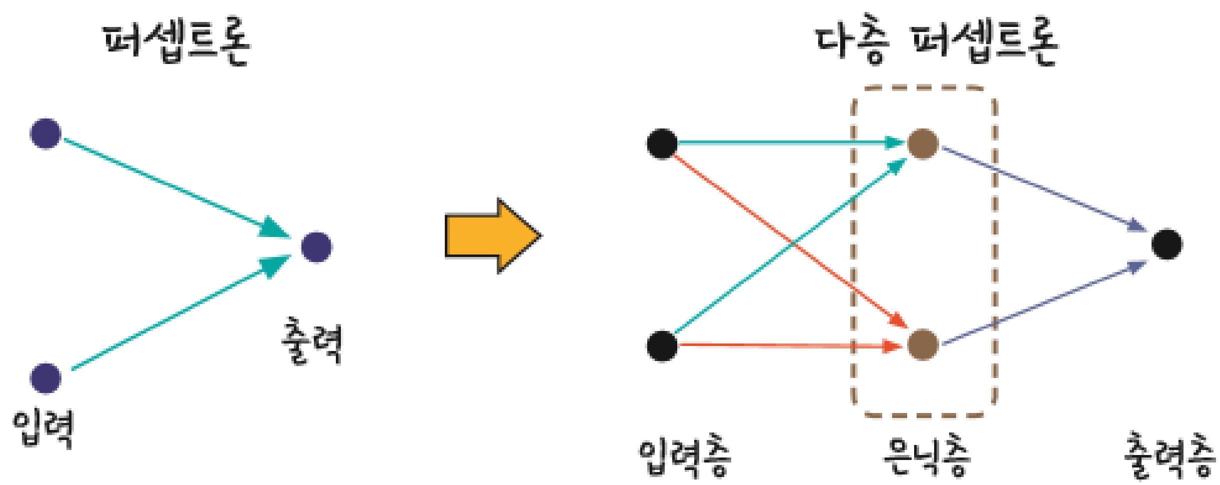
■ XOR 문제 해결

- XOR 문제 극복은 평면을 휘어주는것! 즉, 좌표 평면 자체에 변화를 주는 것



■ XOR 문제 해결

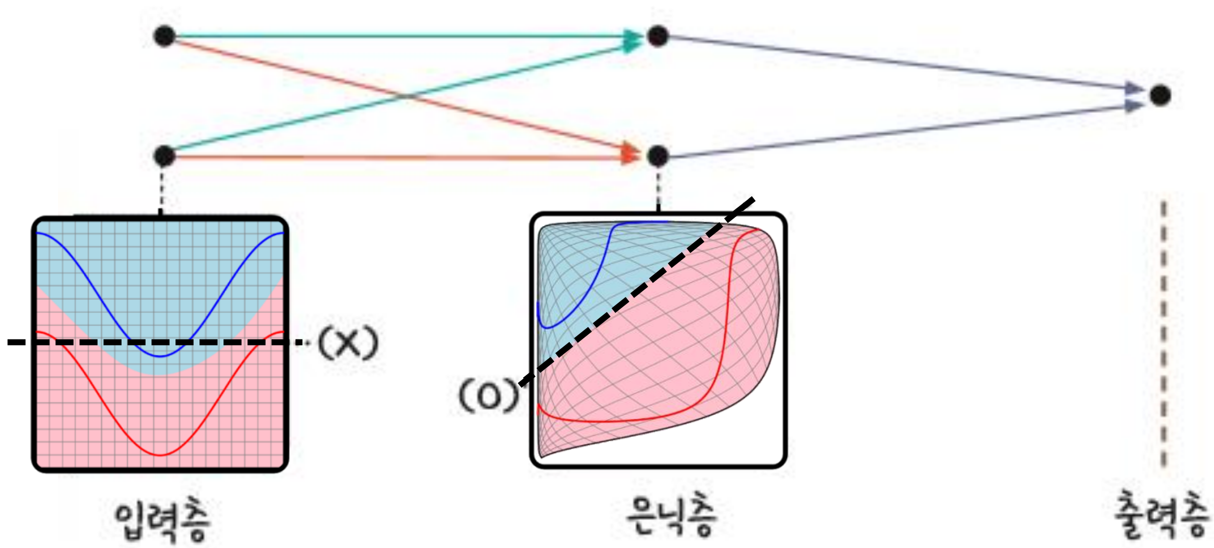
- XOR 문제를 해결하기 위해서 두 개의 퍼셉트론을 한 번에 계산할 수 있어야 함
- 이를 가능하게 하려면 숨어있는 층, 즉 은닉층(hidden layer)을 만들면 됨



■ XOR 문제 해결

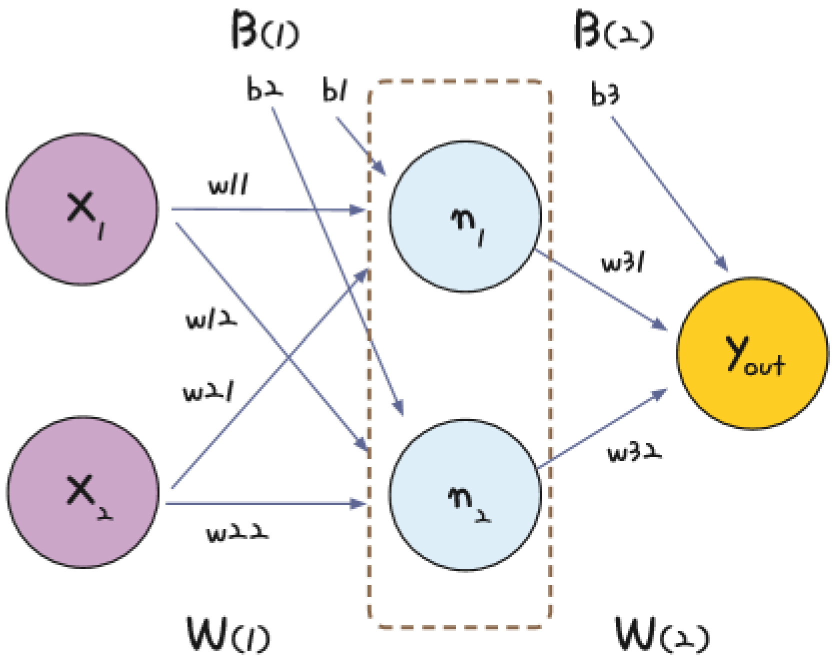
- 입력층과 은닉층의 그래프를 집어넣어 보면 아래 그림과 같음
- 은닉층이 좌표 평면을 왜곡시키는 결과를 가져옴

  → 두 영역을 가로지르는 선이 직선으로 바뀜



은닉층의 공간 왜곡(https://goo.gl/8qEGHD 참조)

■ 다층 퍼셉트론 설계



$$n_1 = \sigma(x_1 w_{11} + x_2 w_{21} + b_1)$$

n_2 = sigmoid($x_1 w_{12}$ + $x_2 w_{22}$ + b_2)

$$y_{\text{out}} = \sigma(n_1 w_{31} + n_2 w_{32} + b_3)$$

- 가중치 6개와 바이어스 3개가 필요함

$$W(1) = \begin{bmatrix} w_{11} & w_{12} \\ w_{21} & w_{22} \end{bmatrix} \quad B(1) = \begin{bmatrix} b_1 \\ b_2 \end{bmatrix}$$

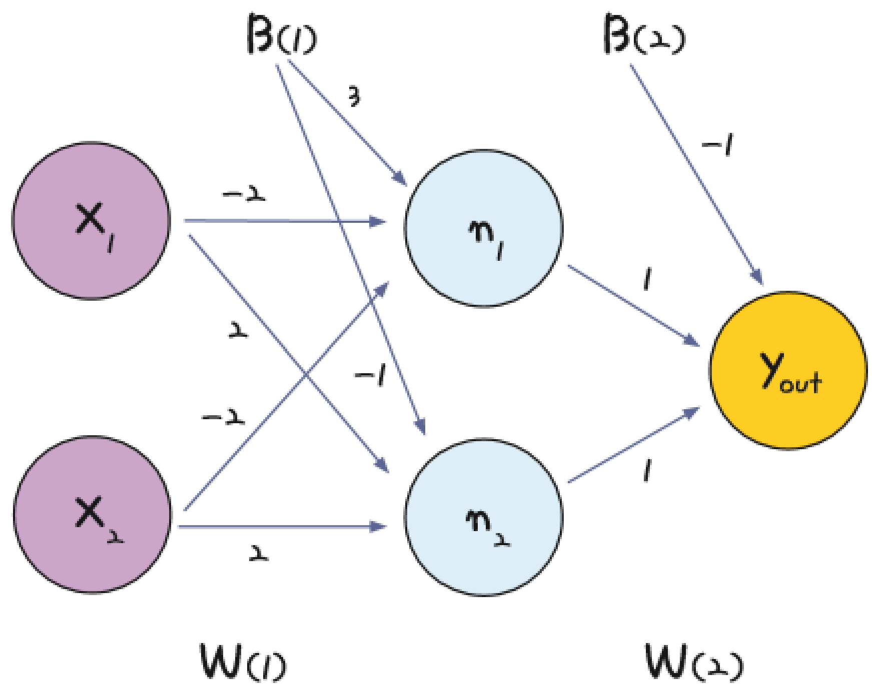
$$W(2) = \begin{bmatrix} w_{31} \\ w_{32} \end{bmatrix} \quad B(2) = [b_3]$$

■ XOR 문제의 해결

- XOR 문제를 해결을 위해 가중치 6개와 바이어스 3개가 필요함
- 이를 만족하는 가중치와 바이어스의 조합은 무수히 많음
- 예)

$$W(1) = \begin{bmatrix} -2 & 2 \\ -2 & 2 \end{bmatrix} \quad B(1) = \begin{bmatrix} 3 \\ -1 \end{bmatrix}$$

$$W(2) = \begin{bmatrix} 1 \\ 1 \end{bmatrix} \quad B(2) = [-1]$$



■ XOR 문제의 해결

- 검증

x_1	x_2	n_1	n_2	y_{out}	우리가 원하는 값
0	0	$\sigma(0*(-2)+0*(-2)+3)=1$	$\sigma(0*2+0*2-1)=0$	$\sigma(1*1+0*1-1)=0$	0
0	1	$\sigma(0*(-2)+1*(-2)+3)=1$	$\sigma(0*2+1*2-1)=1$	$\sigma(1*1+1*1-1)=1$	1
1	0	$\sigma(1*(-2)+0*(-2)+3)=1$	$\sigma(1*2+0*2-1)=1$	$\sigma(1*1+1*1-1)=1$	1
1	1	$\sigma(1*(-2)+1*(-2)+3)=0$	$\sigma(1*2+1*2-1)=1$	$\sigma(0*1+1*1-1)=0$	0

■ 코딩으로 확인하는 XOR 문제

▶ 11_다층퍼셉트론-XOR.ipynb

■ 코딩으로 확인하는 XOR 문제

- 실행 결과

```
입력 값: (0, 0) 출력 값: 0
입력 값: (1, 0) 출력 값: 1
입력 값: (0, 1) 출력 값: 1
입력 값: (1, 1) 출력 값: 0
```

- 우리가 원하는 XOR 문제의 정답이 도출됨
- 이렇게 퍼셉트론 하나로 해결되지 않던 문제를 은닉층을 만들어 해결
- 은닉층을 여러 개 쌓아올려 복잡한 문제를 해결하는 과정이 뉴런이 복잡한 과정을 거쳐 사고를 낳는 사람의 신경망을 닮음
- 그래서 이 방법을 인공 신경망이라 부르기 시작했고, 이를 간단히 줄여서 신경망이라고 통칭

■ 오차 역전파(Back Propagation) 개념

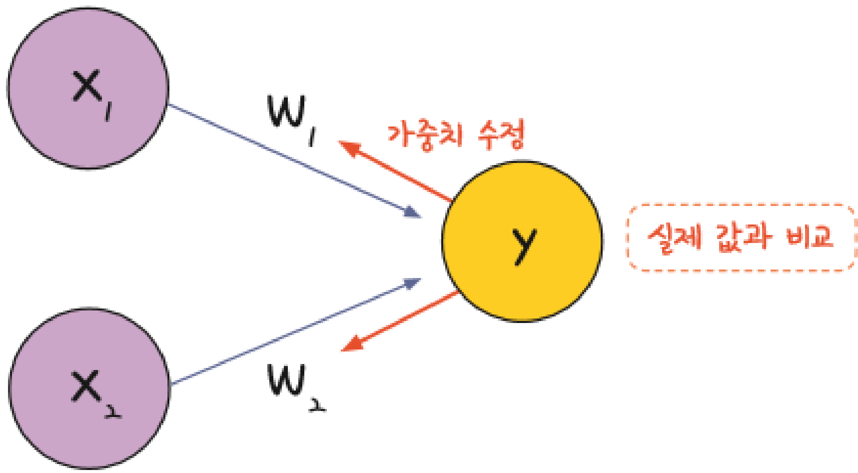
- 최적화의 계산 방향이 출력층에서 시작해 앞(뒤에서 앞으로)으로 진행됨
  → 오차 역전파(back propagation)라고 부름

- 경사 하강법 → 입력과 출력이 하나일 때, 즉 '단일 퍼셉트론'일 경우

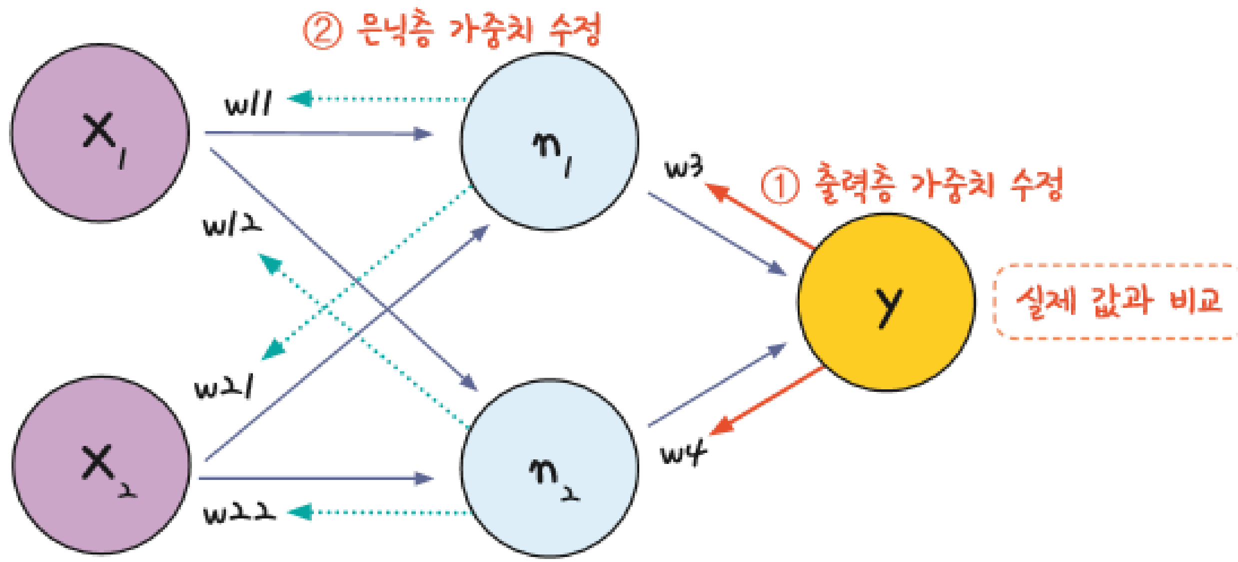
- 은닉층(Hidden Layer)이 있는 경우에는?
  구해야 할 가중치(w)와 바이어스(b)는?

- 단일 퍼셉트론에서 결과값을 얻으면 오차를 구해 이를 토대로 앞 단계에서 정한 가중치를 조정하는 것과 마찬가지로
- 다층 퍼셉트론 역시 결과값의 오차를 구해 이를 토대로 하나 앞선 가중치를 차례로 거슬러 올라가며 조정해 감

■ 오차 역전파 개념



◀ 단일 퍼셉트론에서의 오차 수정



다층 퍼셉트론에서의 오차 수정 ▶

■ 오차 역전파 개념

❖ 오차 역전파 구동 방식

1) 임의의 초기 가중치($w_{(1)}$)를 준 뒤 결과(y_{out})를 계산한다.
2) 계산 결과와 우리가 원하는 값 사이의 오차를 구한다.
3) 경사 하강법을 이용해 바로 앞 가중치를 오차가 작아지는 방향으로 업데이트한다.
4) 1)~3) 과정을 더이상 오차가 줄어들지 않을 때까지 반복한다.

❖ 해설

- '오차가 작아지는 방향으로 업데이트한다'는 의미는 미분 값이 0에 가까워지는 방향으로 나아간다는 말
- 즉, '기울기가 0이 되는 방향'으로 나아가야 하는데, 이 말은 가중치에서 기울기를 뺐을 때 가중치의 변화가 전혀 없는 상태를 말함

■ 미분 공식

1. $\frac{d}{dx}(c) = 0$

2. $\frac{d}{dx}[cf(x)] = cf'(x)$

3. $\frac{d}{dx}[f(x)+g(x)] = f'(x)+g'(x)$

4. $\frac{d}{dx}[f(x)-g(x)] = f'(x)-g'(x)$

5. $\frac{d}{dx}[f(x)g(x)] = f(x)g'(x)+g(x)f'(x)$ 곱셈공식

6. $\frac{d}{dx}[\frac{f(x)}{g(x)}] = \frac{g(x)f'(x)-f(x)g'(x)}{[g(x)]^2}$ 나눗셈공식

7. $\frac{d}{dx}f(g(x)) = f'(g(x))g'(x)$ 연쇄법칙(체인룰)

8. $\frac{d}{dx}(x^n) = nx^{n-1}$

■ Sigmoid 미분

$$F(x) = \frac{1}{1+e^{-x}} \qquad \begin{matrix} < f(x) \\ < g(x) \end{matrix}$$

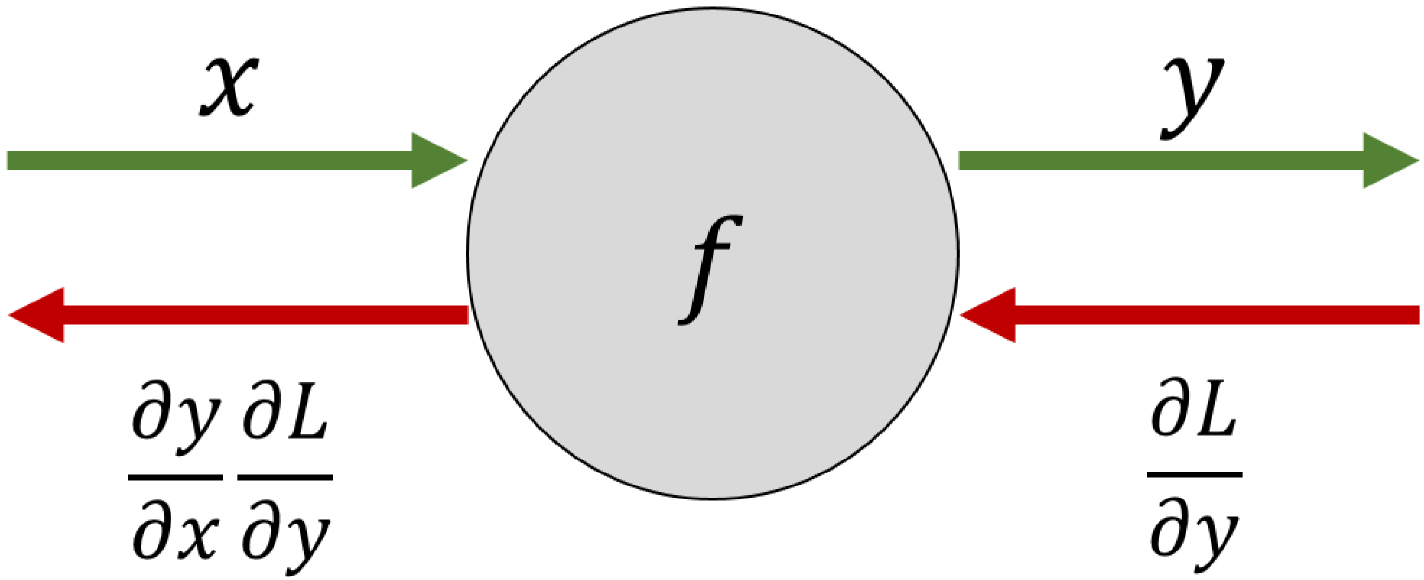
$$F'(x) = \frac{f'(x)g(x) - f(x)g'(x)}{g^2(x)}$$

$$= \frac{0-(-e^{-x})}{(1+e^{-x})^2} = \frac{1}{1+e^{-x}}\frac{e^{-x}}{1+e^{-x}}$$

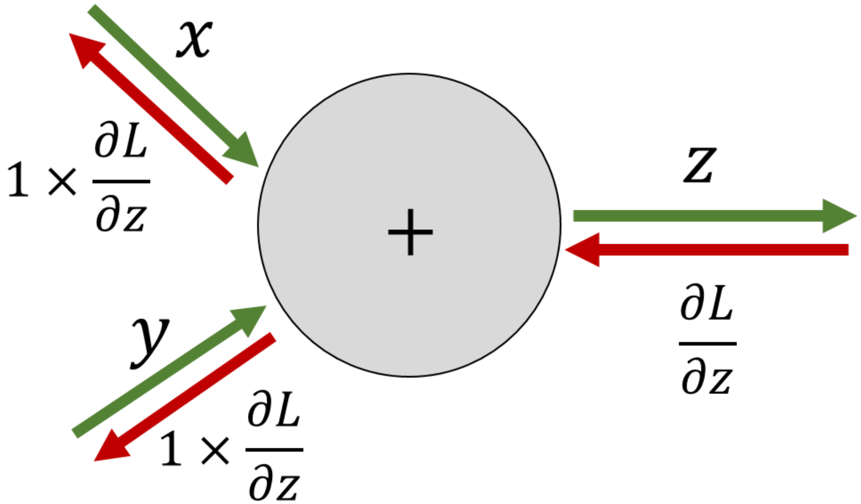
$$= F(x)\frac{1+e^{-x}-1}{1+e^{-x}} = F(x)\left(\frac{1+e^{-x}}{1+e^{-x}} - \frac{1}{1+e^{-x}}\right)$$

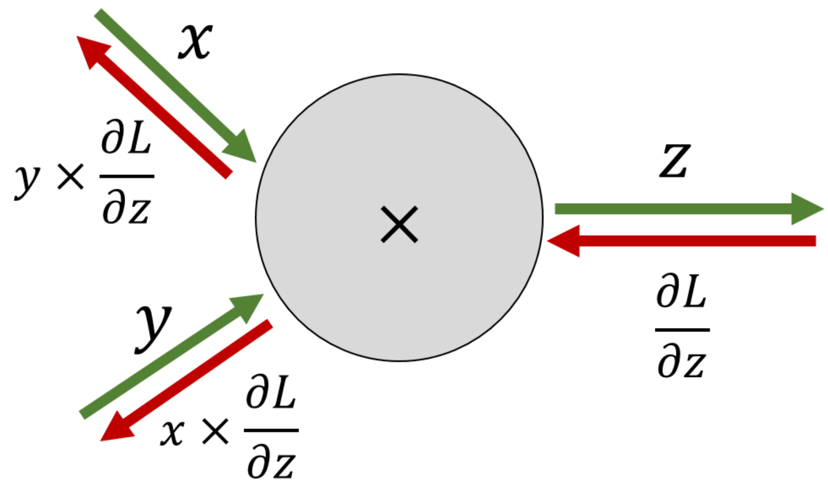
$$= F(x)(1-F(x))$$

■ **수식으로 표현**


x
f
y
$\frac{\partial y}{\partial x}\frac{\partial L}{\partial y}$
$\frac{\partial L}{\partial y}$

■ 수식으로 표현


x
$1 \times \frac{\partial L}{\partial z}$
$+$
z
$\frac{\partial L}{\partial z}$
y
$1 \times \frac{\partial L}{\partial z}$


x
$y \times \frac{\partial L}{\partial z}$
$\times$
z
$\frac{\partial L}{\partial z}$
y
$x \times \frac{\partial L}{\partial z}$

■ 수식으로 표현

1 $\dfrac{\partial L}{\partial \mathbf{X}} = \dfrac{\partial L}{\partial \mathbf{Y}} \cdot \mathbf{W}^{\mathrm{T}}$

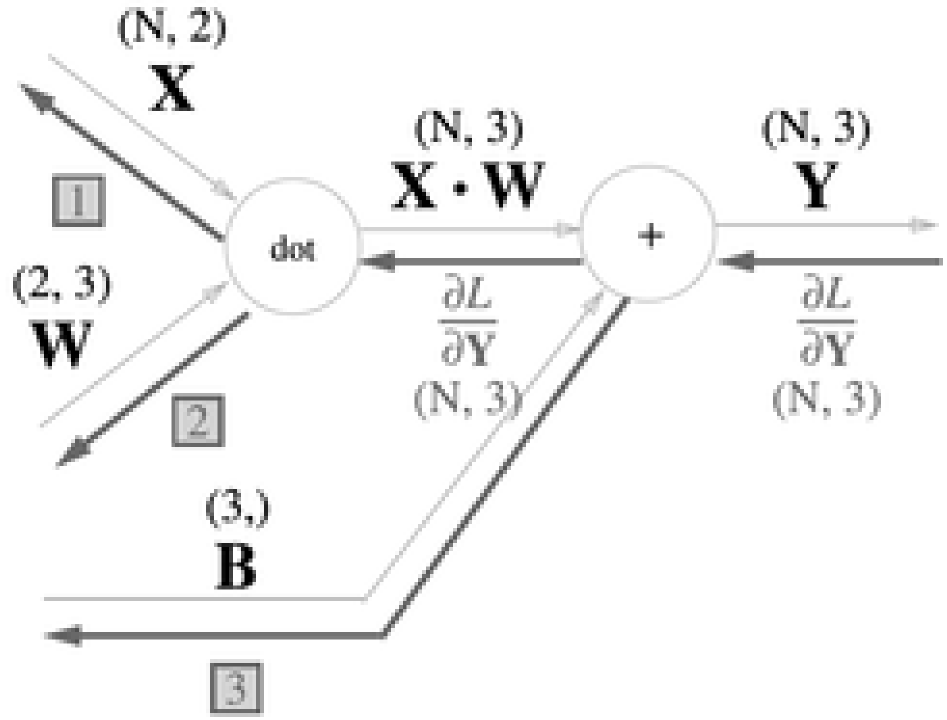
(N, 2) (N, 3) (3, 2)

2 $\dfrac{\partial L}{\partial \mathbf{W}} = \mathbf{X}^{\mathrm{T}} \cdot \dfrac{\partial L}{\partial \mathbf{Y}}$

(2, 3) (2, N) (N, 3)

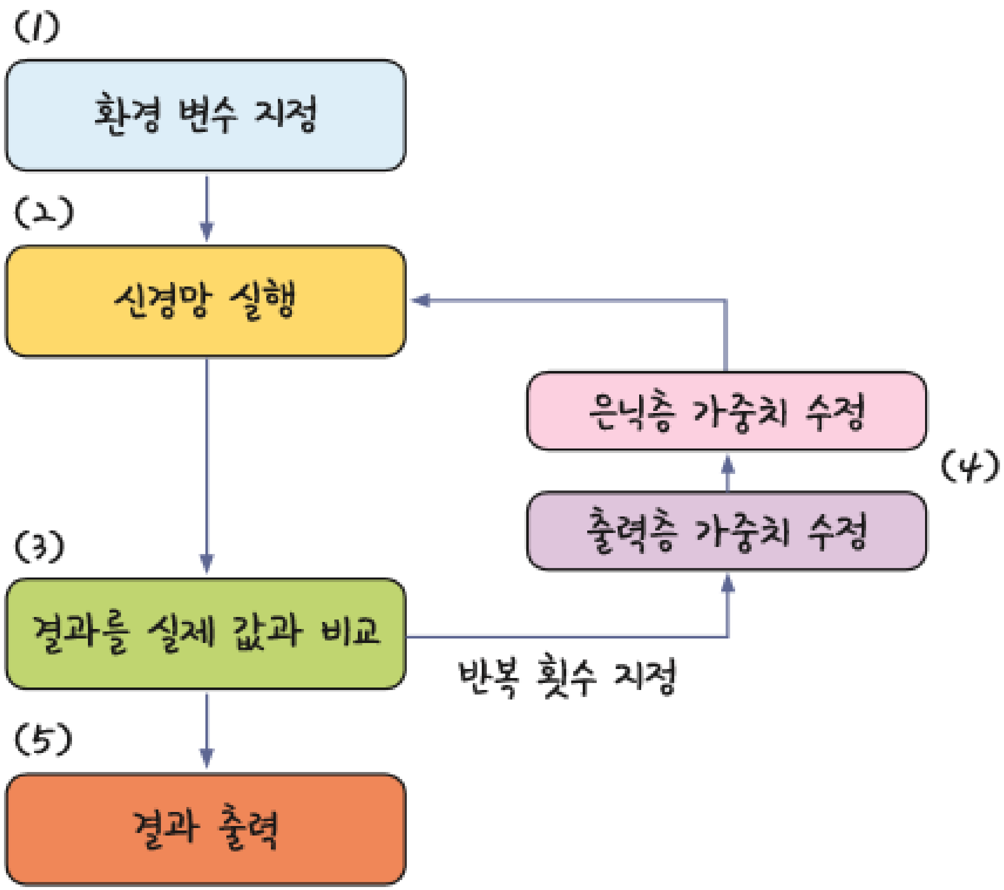
3 $\dfrac{\partial L}{\partial \mathbf{B}} = \dfrac{\partial L}{\partial \mathbf{Y}}$ 의 첫 번째 축(0축, 열방향)의 합

(3) (N, 3)



■ 코딩으로 확인하는 오차 역전파 (XOR 사례)

▶ 11_다층퍼셉트론-XOR.ipynb

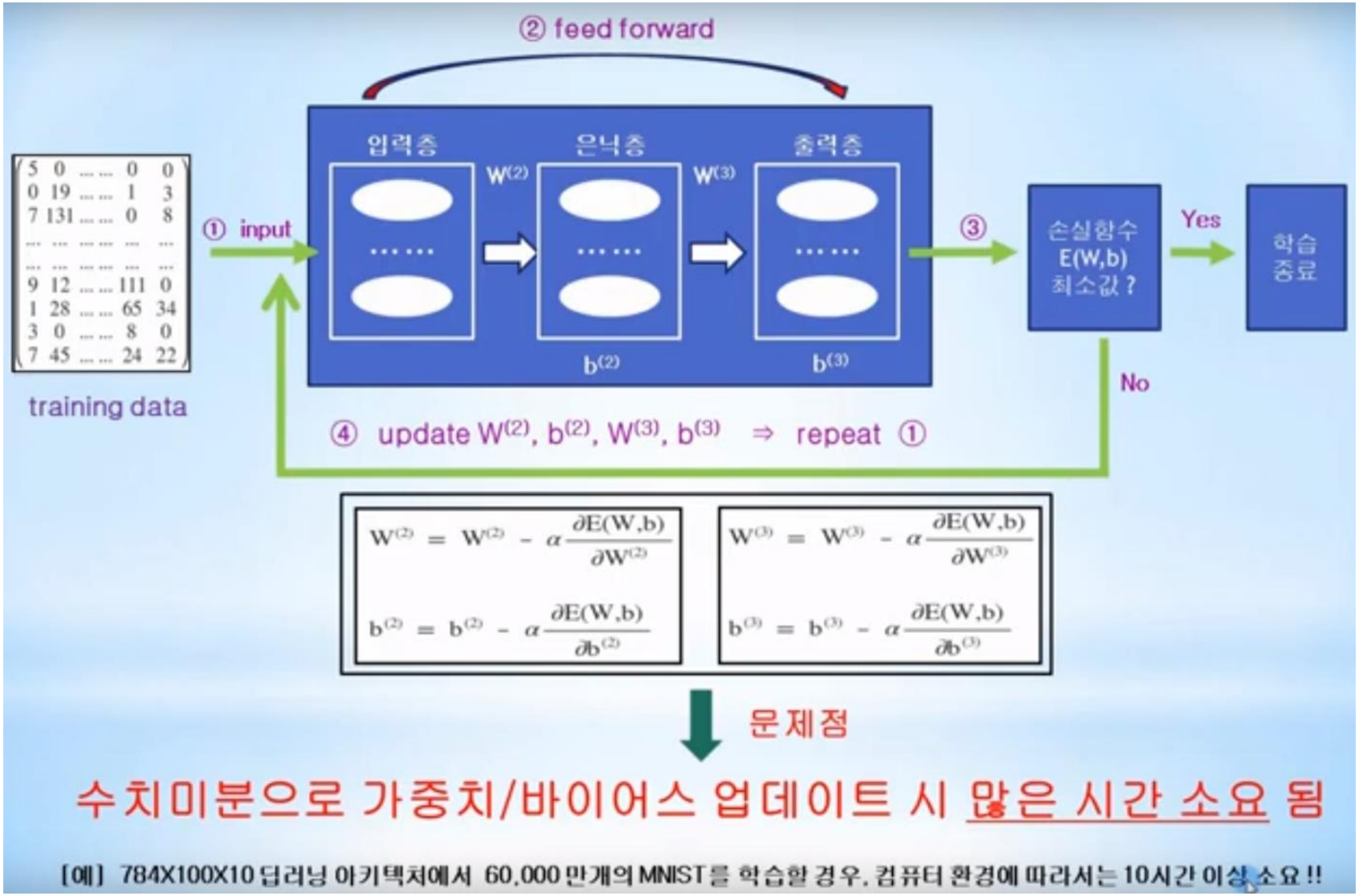


■ 코딩으로 확인하는 오차 역전파 (XOR 사례)

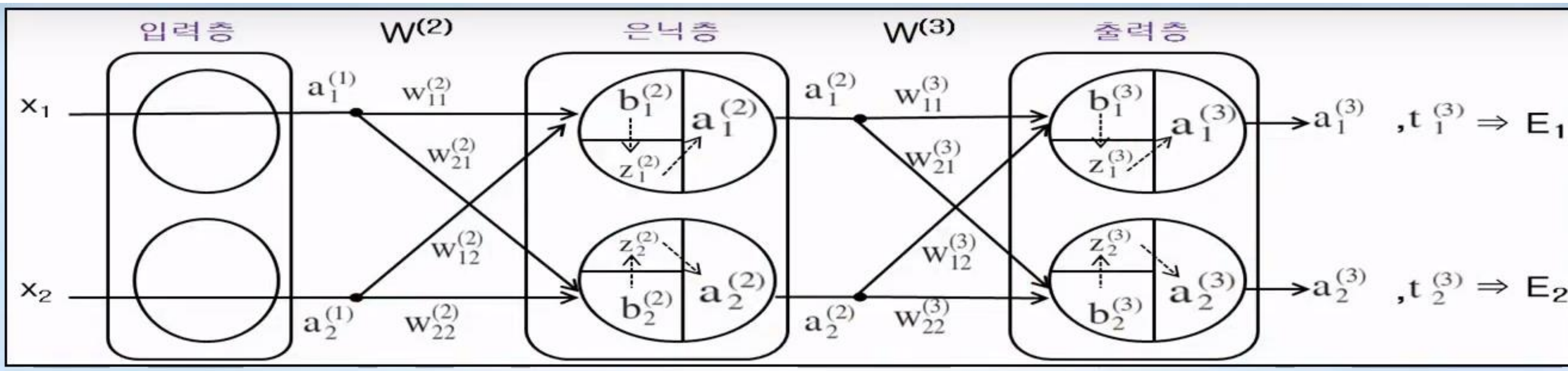
- 실행 결과

```
error: 0.67409
error: 0.00908
error: 0.00629
error: 0.00534
error: 0.00485
error: 0.00456
error: 0.00437
error: 0.00424
error: 0.00414
error: 0.00407
Input: [0, 0], Predict: [0.022050373480282319]
Input: [0, 1], Predict: [0.93979714252823832]
Input: [1, 0], Predict: [0.94211459973218525]
Input: [1, 1], Predict: [0.024052645439017865]
```

■ **오차 역전파 – 개념 및 문제점** ◀ 오차역전파 강의 YouTube (NeoWizard)

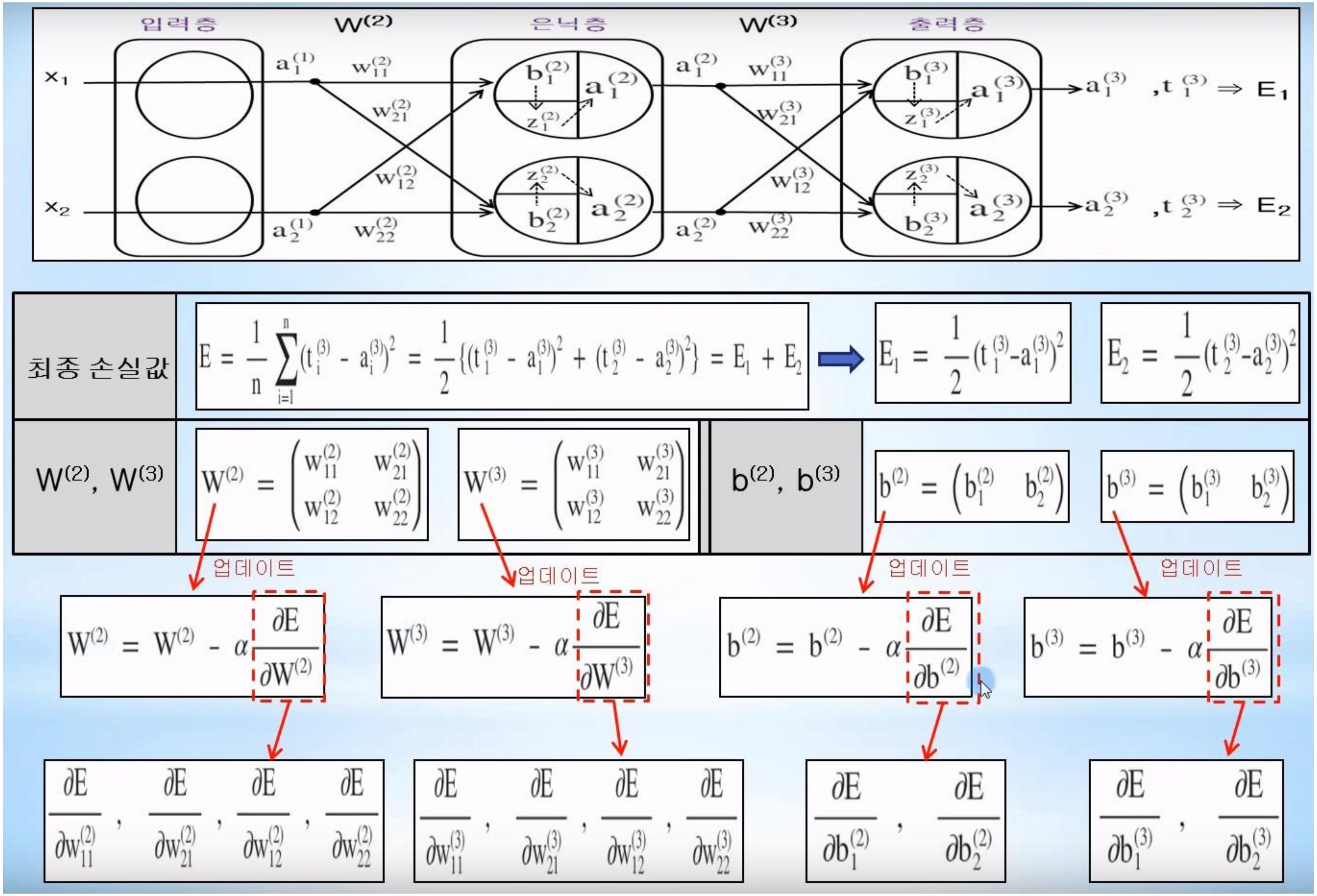


■ 오차 역전파 – 각 층의 선형회귀값(z) / 각 층의 출력값(a)

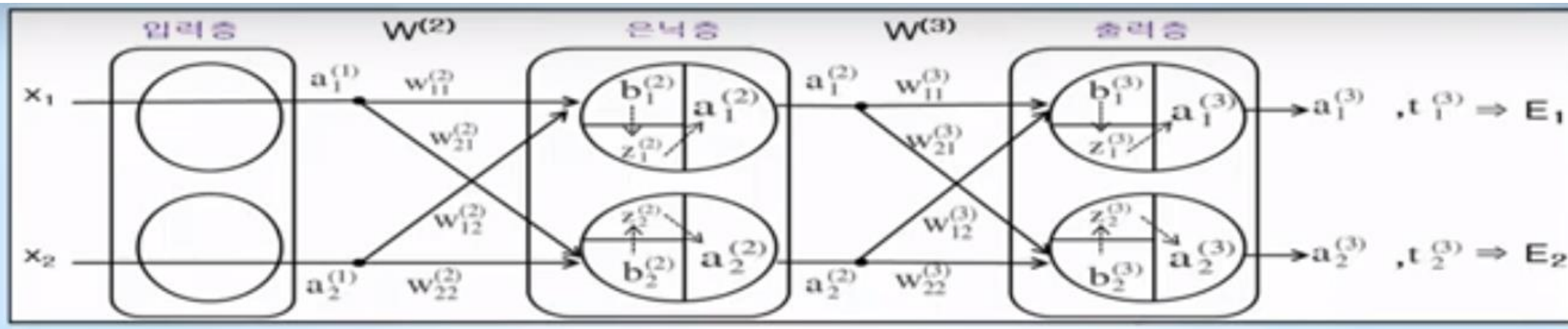


	선형 회귀 값 (z)	출력 값 (a)
입력층	입력 층에는 가중치가 없기 때문에 선형회귀 값은 적용하지 않음	$a_1^{(1)} = x_1$ $a_2^{(1)} = x_2$
은닉층	$z_1^{(2)} = a_1^{(1)}w_{11}^{(2)} + a_2^{(1)}w_{12}^{(2)} + b_1^{(2)}$	$a_1^{(2)} = \text{sigmoid}(z_1^{(2)})$
	$z_2^{(2)} = a_1^{(1)}w_{21}^{(2)} + a_2^{(1)}w_{22}^{(2)} + b_2^{(2)}$	$a_2^{(2)} = \text{sigmoid}(z_2^{(2)})$
출력층	$z_1^{(3)} = a_1^{(2)}w_{11}^{(3)} + a_2^{(2)}w_{12}^{(3)} + b_1^{(3)}$	$a_1^{(3)} = \text{sigmoid}(z_1^{(3)})$
	$z_2^{(3)} = a_1^{(2)}w_{21}^{(3)} + a_2^{(2)}w_{22}^{(3)} + b_2^{(3)}$	$a_2^{(3)} = \text{sigmoid}(z_2^{(3)})$

■ 오차 역전파 – 최종 손실(에러) 값 E / 가중치 W / 바이어스 b



■ 오차 역전파 – 선형회귀 값 Z / 출력 값 A / 손실 값 E / 가중치 W / 바이어스 b



입력층 선형회귀 값 (Z1)	입력 층에는 가중치가 없기 때문에 선형회귀 값은 적용하지 않음	입력층 출력 값 (A1)	$a_2^{(1)} = x_2$ $a_1^{(1)} = x_1$
은닉층 선형회귀 값 (Z2)	$z_1^{(2)} = a_1^{(1)}w_{11}^{(2)} + a_2^{(1)}w_{12}^{(2)} + b_1^{(2)}$ $z_2^{(2)} = a_1^{(1)}w_{21}^{(2)} + a_2^{(1)}w_{22}^{(2)} + b_2^{(2)}$	은닉층 출력 값 (A2)	$a_1^{(2)} = \mathrm{sigmoid}(z_1^{(2)})$ $a_2^{(2)} = \mathrm{sigmoid}(z_2^{(2)})$
출력층 선형회귀 값 (Z3)	$z_1^{(3)} = a_1^{(2)}w_{11}^{(3)} + a_2^{(2)}w_{12}^{(3)} + b_1^{(3)}$ $z_2^{(3)} = a_1^{(2)}w_{21}^{(3)} + a_2^{(2)}w_{22}^{(3)} + b_2^{(3)}$	출력층 출력 값 (A3)	$a_1^{(3)} = \mathrm{sigmoid}(z_1^{(3)})$ $a_2^{(3)} = \mathrm{sigmoid}(z_2^{(3)})$
$W^{(2)}$, $W^{(3)}$	$W^{(2)} = \begin{pmatrix} w_{11}^{(2)} & w_{21}^{(2)} \\ w_{12}^{(2)} & w_{22}^{(2)} \end{pmatrix}$ $W^{(3)} = \begin{pmatrix} w_{11}^{(3)} & w_{21}^{(3)} \\ w_{12}^{(3)} & w_{22}^{(3)} \end{pmatrix}$	$b^{(2)}$, $b^{(3)}$	$b^{(2)} = \begin{pmatrix} b_1^{(2)} & b_2^{(2)} \end{pmatrix}$ $b^{(3)} = \begin{pmatrix} b_1^{(3)} & b_2^{(3)} \end{pmatrix}$
최종 손실 값 (E)	$E = \frac{1}{n}\sum_{i=1}^{n}(t_i^{(3)} - a_i^{(3)})^2 = \frac{1}{2}\{(t_1^{(3)} - a_1^{(3)})^2 + (t_2^{(3)} - a_2^{(3)})^2\} = E_1 + E_2$ ➡		$E_1 = \frac{1}{2}(t_1^{(3)} - a_1^{(3)})^2$ $E_2 = \frac{1}{2}(t_2^{(3)} - a_2^{(3)})^2$

■ 오차 역전파 – 일반 공식 구하는 순서

➤ **오차역전파 공식 구하는 순서** (은닉층이 1개인 단순 신경망으로부터 공식 패턴 추출)

[1] 출력층의 출력 값과 정답(Target)을 이용하여, 출력층 손실 계산

출력층의 손실 = (출력층출력 – 정답) X 출력층출력(1 – 출력층출력)

[2] 은닉층 에서의 (가상의) 손실 loss_3, loss_2 등을 계산할 경우, 현재 층 (current layer) / 이전 층 (previous layer) / 다음 층 (next layer) 개념을 도입하여 동일한 패턴으로 반복 계산

[예] 은닉층2를 현재층(current layer)로 설정한다면, 이전층(previous layer) 은닉층1 / 다음층(next layer)은 출력층으로 가정함

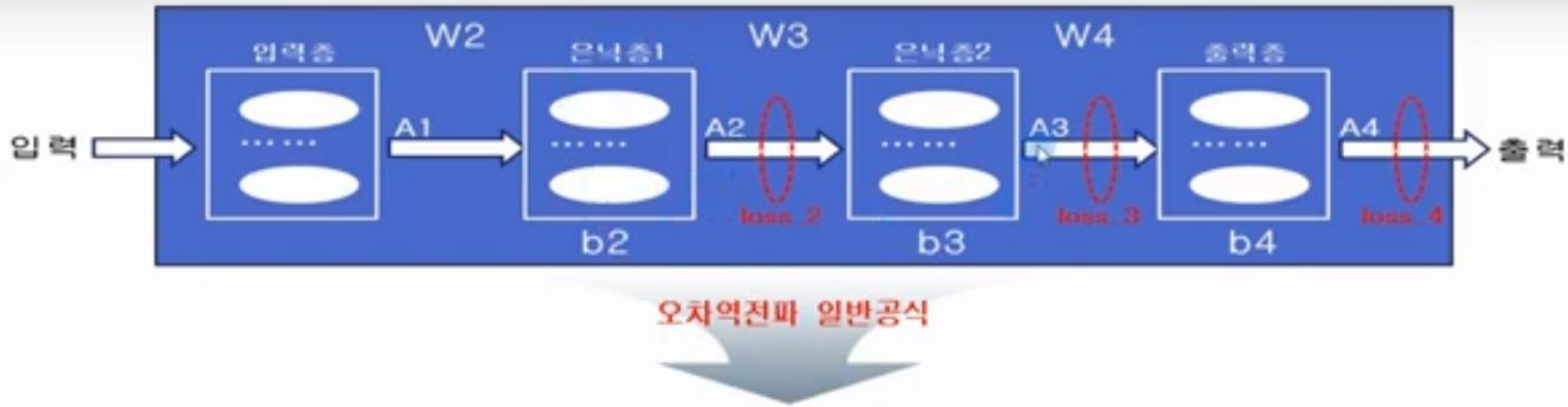
은닉층의 현재층 손실 = (다음층 손실 • 다음층에 적용되는 가중치W^T) X 현재층출력(1 – 현재층출력)

[3] 계산된 각 층의 출력 값과 손실을 이용하여,

현재층의 바이어스 변화율 $\partial E/\partial b$ = 현재층 손실

현재층에 적용되는 가중치 변화율 $\partial E/\partial W$ = (이전층 출력)^T • 현재층 손실

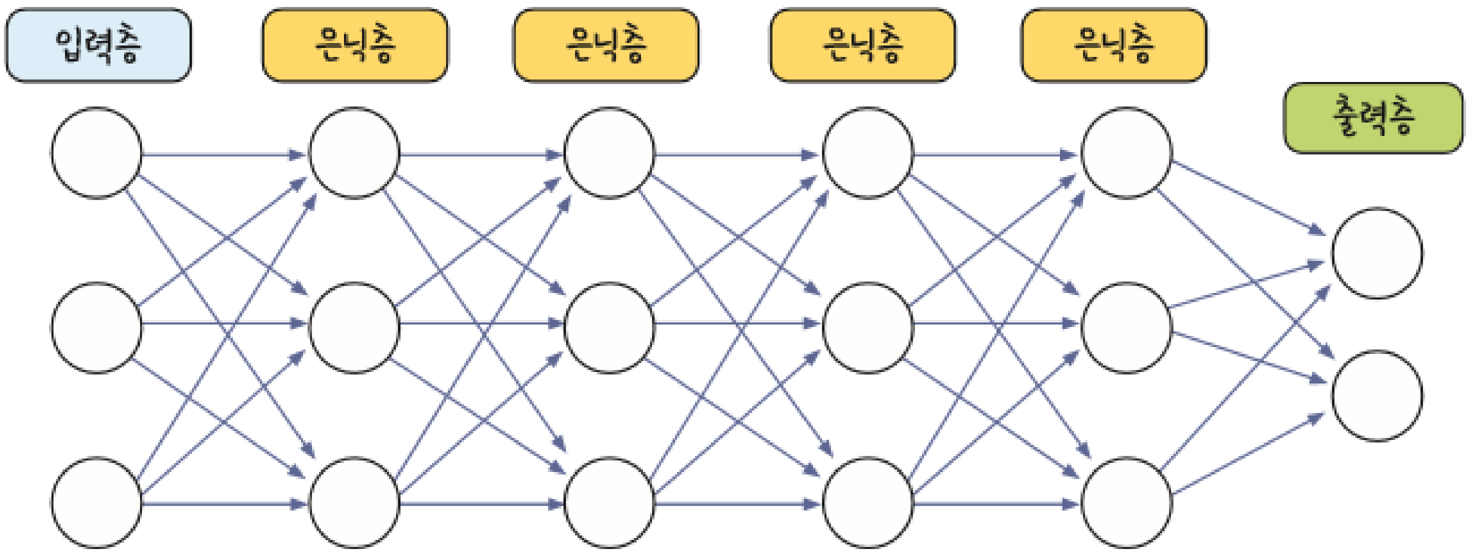
■ 오차 역전파 – 일반 공식



가중치 업데이트	$W4 = W4 - \alpha \frac{\partial E}{\partial W4} = W4 - \alpha \times (A3^T \bullet loss\_4)$
	$W3 = W3 - \alpha \frac{\partial E}{\partial W3} = W3 - \alpha \times (A2^T \bullet loss\_3)$
	$W2 = W2 - \alpha \frac{\partial E}{\partial W2} = W2 - \alpha \times (A1^T \bullet loss\_2)$
바이어스 업데이트	$b4 = b4 - \alpha \frac{\partial E}{\partial b4} = b4 - \alpha \times loss\_4$
	$b3 = b3 - \alpha \frac{\partial E}{\partial b3} = b3 - \alpha \times loss\_3$
	$b2 = b2 - \alpha \frac{\partial E}{\partial b2} = b2 - \alpha \times loss\_2$

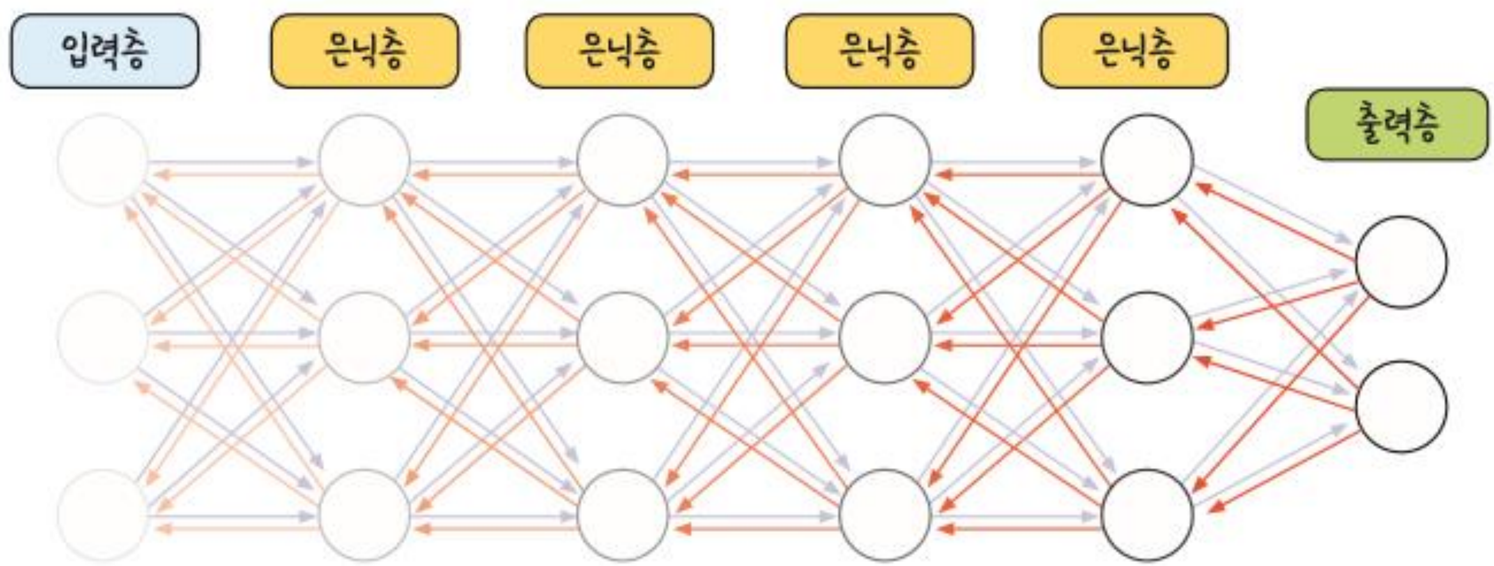
■ 기울기 소실 문제

- 다층 퍼셉트론이 오차 역전파를 만나 신경망이 되었고, 신경망은 XOR 문제를 가볍게 해결
- 하지만 기대만큼 결과가 좋아지지 않았음
- 이유가 무엇일까?



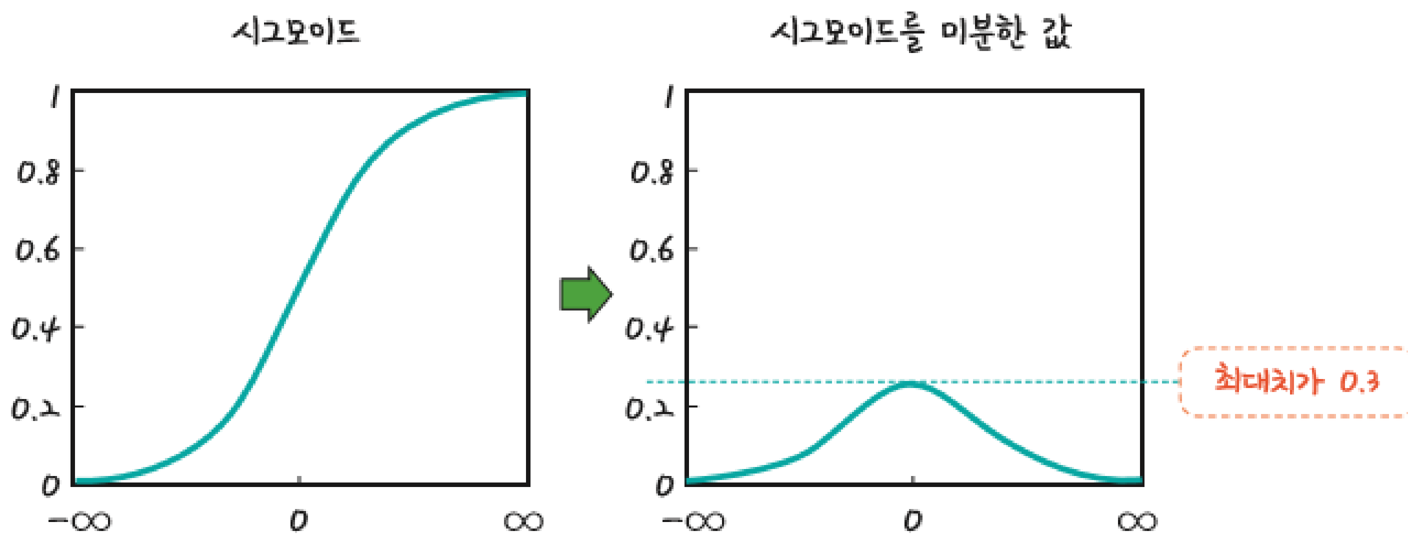
■ 기울기 소실 문제

- 오차 역전파는 출력층으로부터 하나씩 앞으로 되돌아가며 각 층의 가중치를 수정하는 방법
- 가중치를 수정하려면 미분 값, 즉 기울기가 필요하다고 배움
- 그런데 층이 늘어나면서 기울기가 중간에 0이 되어버리는 기울기 소실(vanishing gradient) 문제가 발생하기 시작

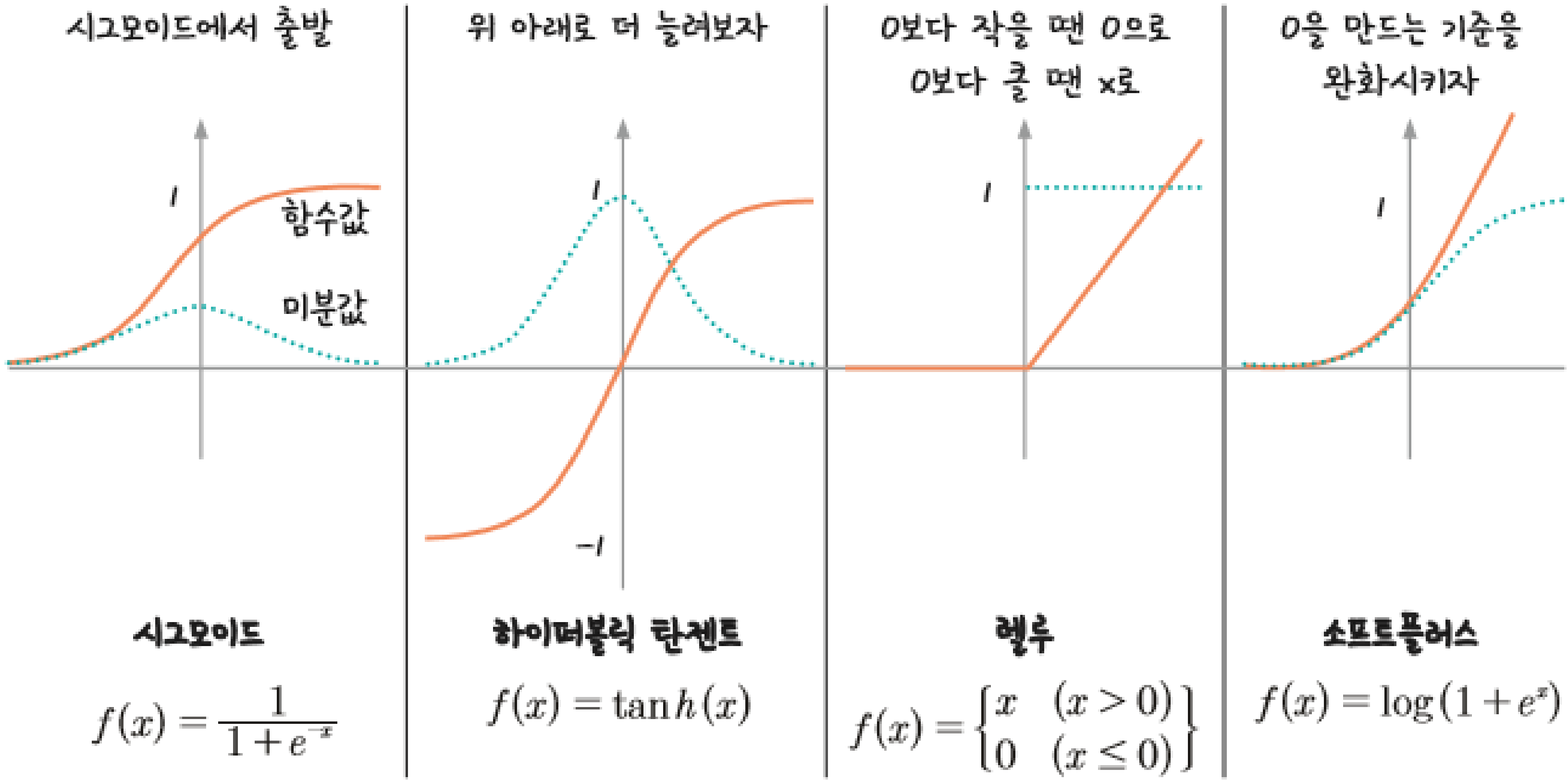


■ 기울기 소실 문제와 활성화 함수

- 이는 활성화 함수로 사용된 시그모이드 함수의 특성 때문임
- 아래 그림처럼 시그모이드를 미분하면 최대치가 0.25
- 1보다 작으므로 계속 곱하다 보면 0에 가까워짐
- 따라서 층을 거쳐 갈수록 기울기가 사라져 가중치를 수정하기가 어려워지는 것



■ 대체 활성화 함수


시그모이드에서 출발
위 아래로 더 늘려보자
0보다 작을 땐 0으로
0보다 클 땐 x로
0을 만드는 기준을
완화시키자
1
함수값
미분값
1
-1
1
1
시그모이드
$f(x)=\frac{1}{1+e^{-x}}$
하이퍼볼릭 탄젠트
$f(x)=\tanh(x)$
렐루
$f(x)=\begin{cases} x & (x>0) \\ 0 & (x\le 0) \end{cases}$
소프트플러스
$f(x)=\log(1+e^{x})$

■ 활성화 함수

❖ 하이퍼볼릭 탄젠트(tanh)

- 시그모이드 함수의 범위를 -1에서 1로 확장한 개념
- 미분한 값의 범위가 함께 확장되는 효과를 가져왔음
- 하지만 여전히 1보다 작은 값이 존재하므로 기울기 소실 문제는 사라지지 않음

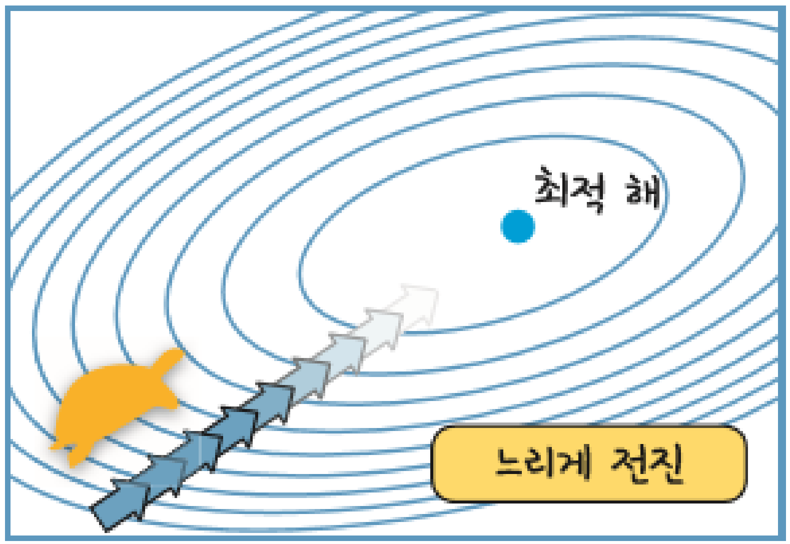
❖ 렐루(ReLU: Rectified Linear Unit)

- 토론토대학교의 제프리 힌튼 교수가 제안
- 시그모이드의 대안으로 떠오르며 현재 가장 많이 사용되는 활성화 함수
- 렐루는 x가 0보다 작을 때는 모든 값을 0으로 처리하고, 0보다 큰 값은 x를 그대로 사용하는 방법. 이 방법을 쓰면 x가 0보다 크기만 하면 미분 값이 1이 됨
- 따라서 여러 은닉층을 거치며 곱해지더라도 맨 처음 층까지 사라지지 않고 남아있을 수 있음: 딥러닝의 발전에 속도가 붙게 됨

■ 속도와 정확도 문제를 해결하는 고급 경사 하강법

❖ 경사 하강법

- 경사 하강법은 정확하게 가중치를 찾아가지만, 한 번 업데이트할 때마다 전체 데이터를 미분해야 하므로 계산량이 매우 많다는 단점이 있음

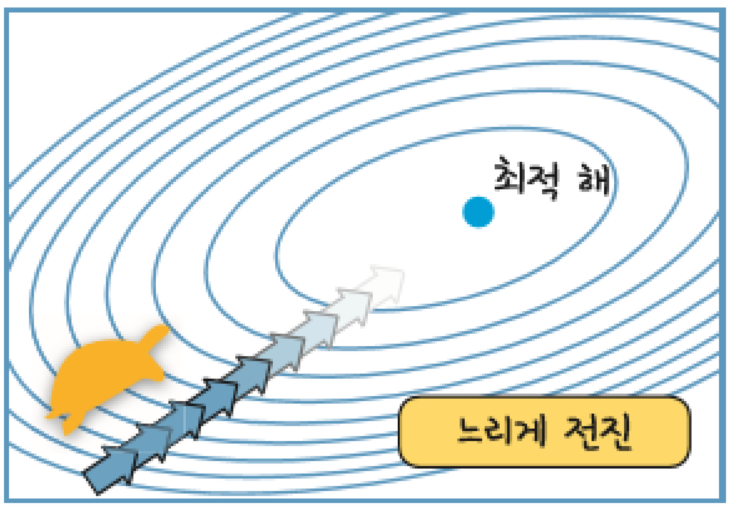


경사 하강법

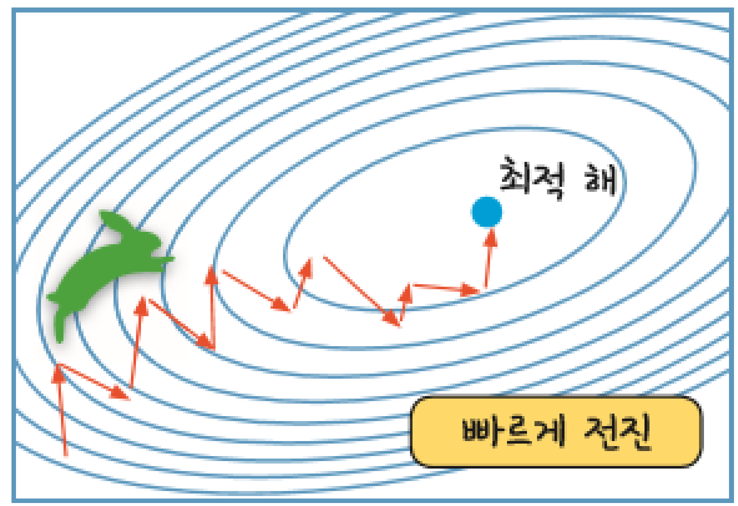
■ 속도와 정확도 문제를 해결하는 고급 경사 하강법

❖ 확률적 경사 하강법(SGD)

- 전체 데이터를 사용하는 것이 아니라, 랜덤하게 추출한 일부 데이터를 사용
- 일부 데이터를 사용하므로 더 빨리 그리고 자주 업데이트를 하는 것이 가능해짐



경사 하강법

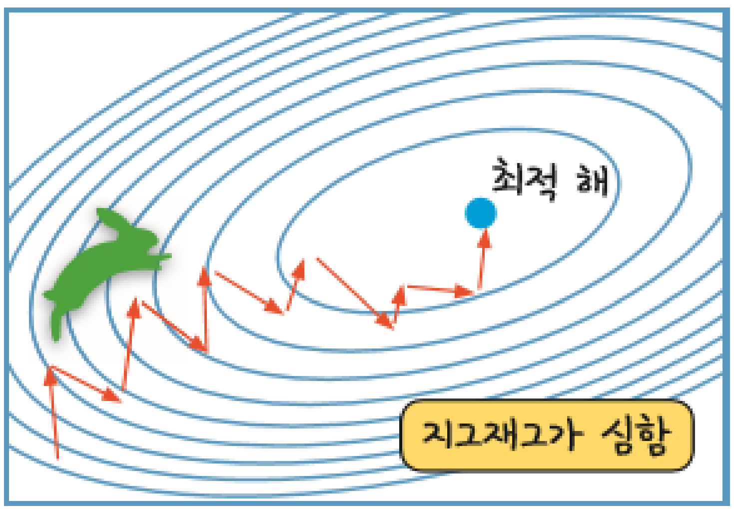


확률적 경사 하강법

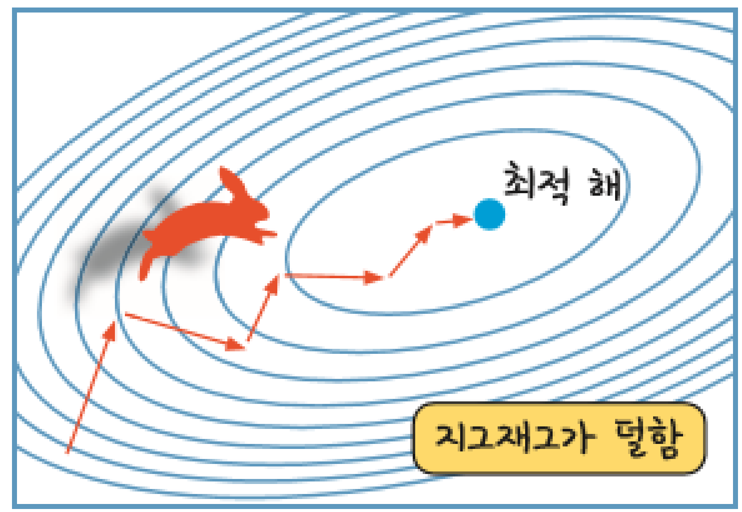
■ 속도와 정확도 문제를 해결하는 고급 경사 하강법

❖ 모멘텀

- 경사 하강법과 마찬가지로 매번 기울기를 구하지만, 이를 통해 오차를 수정하기 전 바로 앞 수정 값과 방향(+, -)을 참고하여 같은 방향으로 일정한 비율만 수정되게 하는 방법 (이동에 탄력을 더한다)



확률적 경사 하강법



모멘텀을 적용한 확률적 경사 하강법

■ 속도와 정확도 문제를 해결하는 고급 경사 하강법

고급 경사 하강법	개요	효과	케라스 사용법
1. 확률적 경사 하강법 (SGD)	랜덤하게 추출한 일부 데이터를 사용해 더 빨리, 자주 업데이트를 하게 하는 것	속도 개선	keras.optimizers.SGD(lr = 0.1) 케라스 최적화 함수를 이용합니다.
2. 모멘텀 (Momentum)	관성의 방향을 고려해 진동과 폭을 줄이는 효과	정확도 개선	keras.optimizers.SGD(lr = 0.1, momentum = 0.9) 모멘텀 계수를 추가합니다.
3. 네스테로프 모멘텀 (NAG)	모멘텀이 이동시킬 방향으로 미리 이동해서 그레이디언트를 계산. 불필요한 이동을 줄이는 효과	정확도 개선	keras.optimizers.SGD(lr = 0.1, momentum = 0.9, nesterov = True) 네스테로프 옵션을 추가합니다.

■ 속도와 정확도 문제를 해결하는 고급 경사 하강법

4. 아다그라드 (Adagrad)	변수의 업데이트가 잦으면 학습률을 적게 하여 이동 보폭을 조절하는 방법	보폭 크기 개선	keras.optimizers.Adagrad(lr = 0.01, epsilon = 1e − 6) 아다그라드 함수를 사용합니다. ※ 참고: 여기서 epsilon, rho, decay 같은 파라미터는 바꾸지 않고 그대로 사용하기를 권장하고 있습니다. 따라서 lr, 즉 learning rate(학습률) 값만 적절히 조절하면 됩니다.
5. 알엠에스프롭 (RMSProp)	아다그라드의 보폭 민감도를 보완한 방법	보폭 크기 개선	keras.optimizers.RMSprop(lr = 0.001, rho = 0.9, epsilon = 1e − 08, decay = 0.0) 알엠에스프롭 함수를 사용합니다.
6. 아담(Adam)	모멘텀과 알엠에스프롭 방법을 합친 방법	정확도와 보폭 크기 개선	keras.optimizers.Adam(lr = 0.001, beta_1 = 0.9, beta_2 = 0.999, epsilon = 1e − 08, decay = 0.0) 아담 함수를 사용합니다.