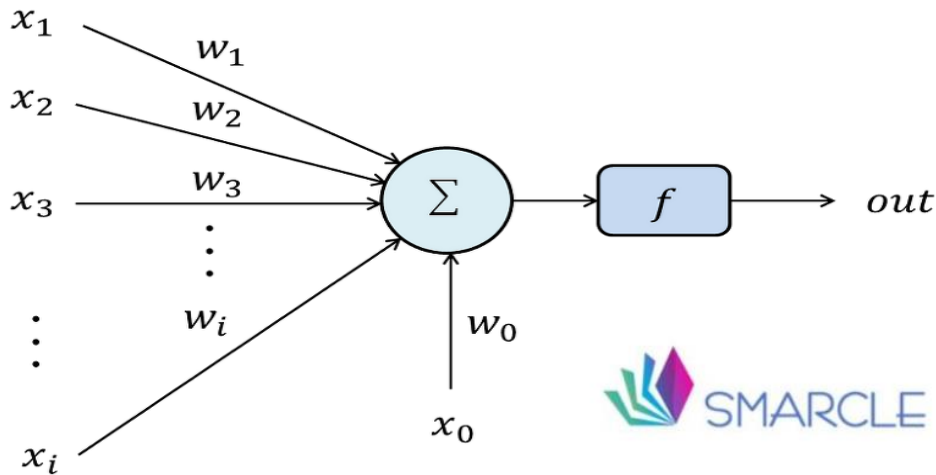


신경망의 이해

2021 SMARCLE Winter team 4

17 송원진
17 신도현
18 권수지
20 이유빈



목차

6장. 퍼셉트론

7장. 다층 퍼셉트론

8장. 오차 역전파

9장. 신경망 to 딥러닝



TABLE OF CONTENTS



01

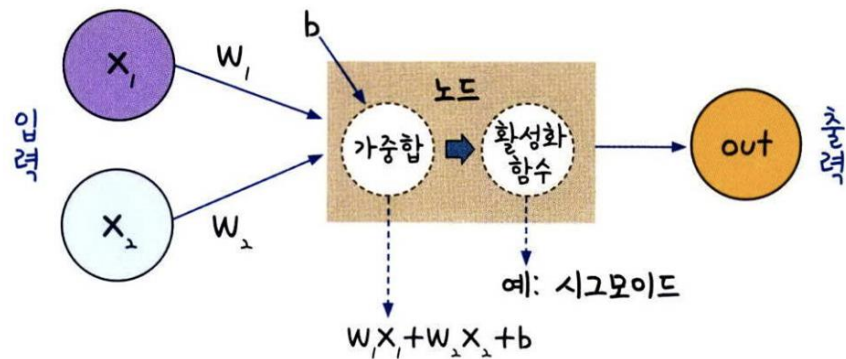
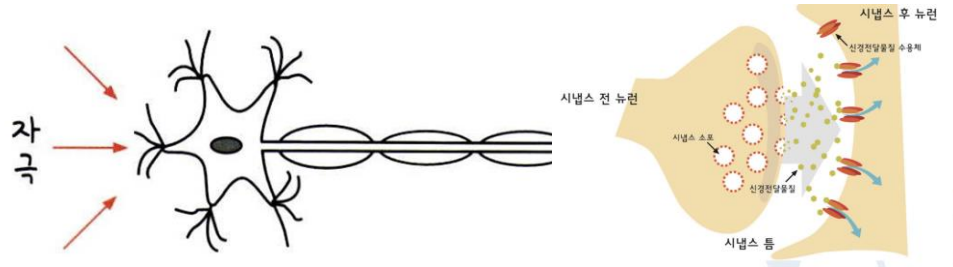
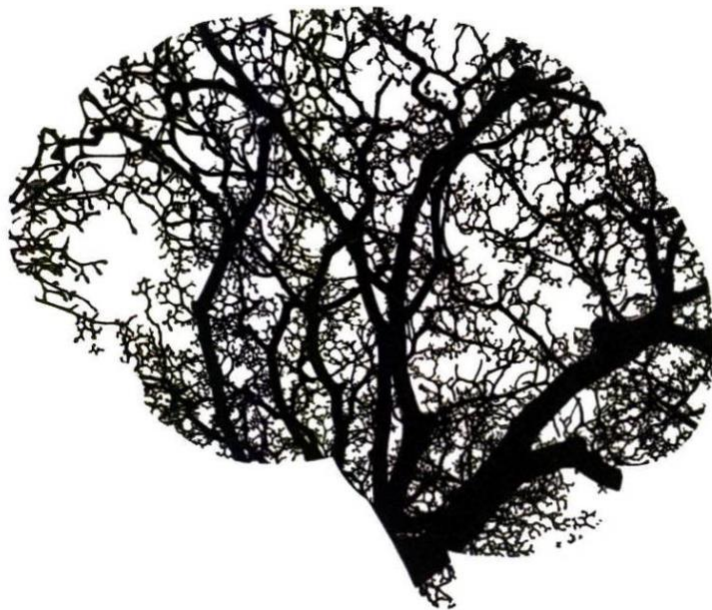
Lore
ante
Sene
Done

04

Lore
ante
Sene
Done

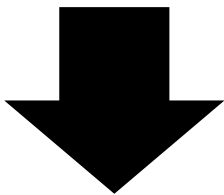


6. 퍼셉트론

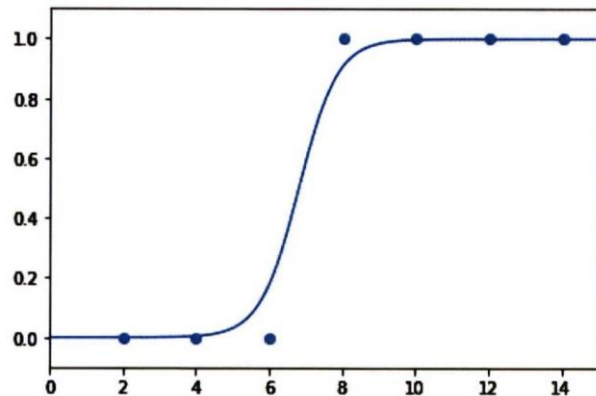


$$y = ax + b \text{ (} a \text{는 기울기, } b \text{는 } y \text{ 절편)}$$

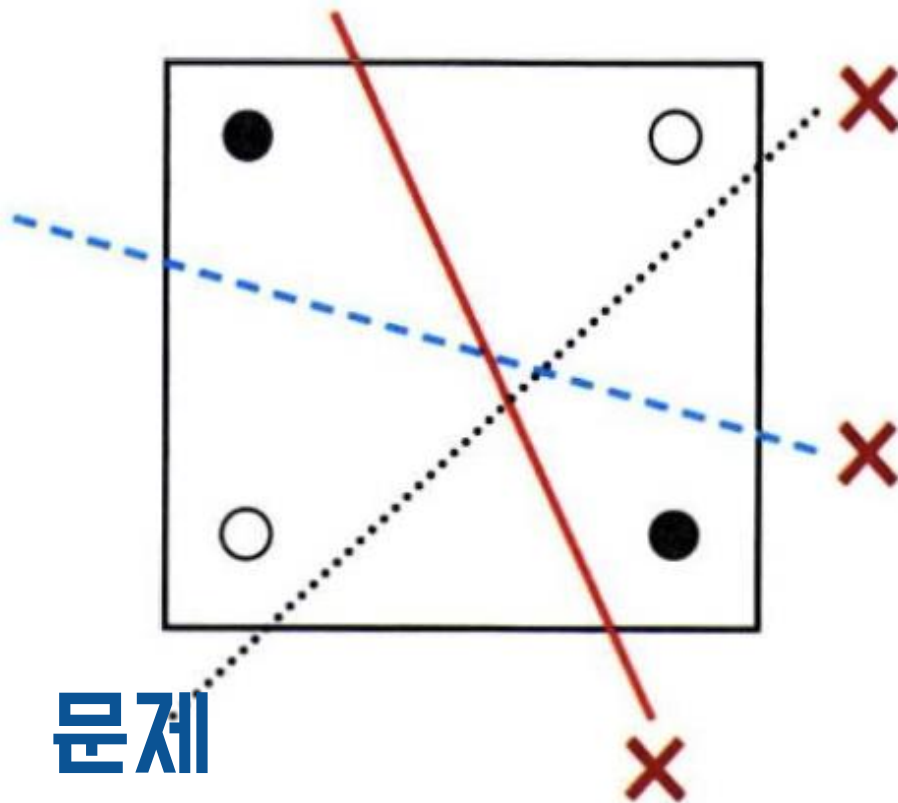
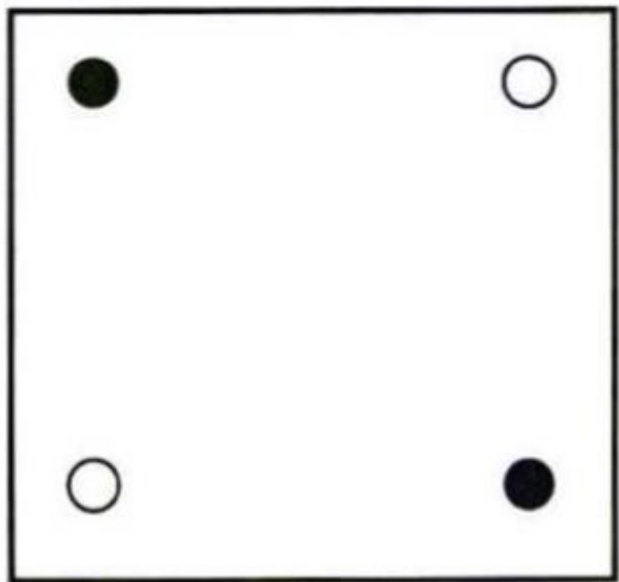
가중합 $\rightarrow y = wx + b$ (w 는 가중치, b 는 바이어스)



활성화 함수



퍼셉트론의 한계



XOR 문제

AND 진리표

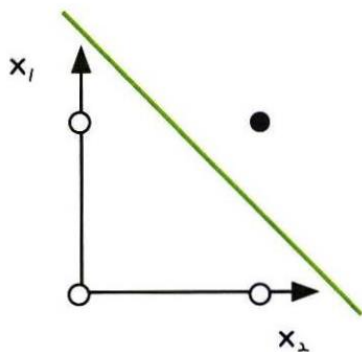
x_1	x_2	결괏값
0	0	0
0	1	0
1	0	0
1	1	1

OR 진리표

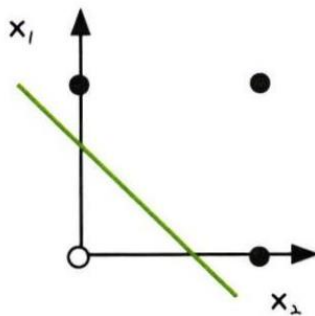
x_1	x_2	결괏값
0	0	0
0	1	1
1	0	1
1	1	1

XOR 진리표

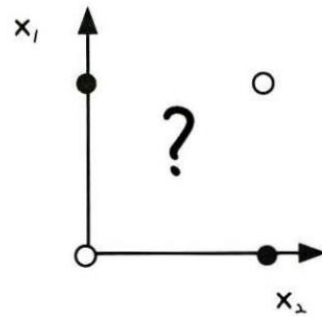
x_1	x_2	결괏값
0	0	0
0	1	1
1	0	1
1	1	0



AND



OR



XOR



7. 다층 퍼셉트론

성냥개비 6개 --> 정삼각형 4개를 만들
어라!?

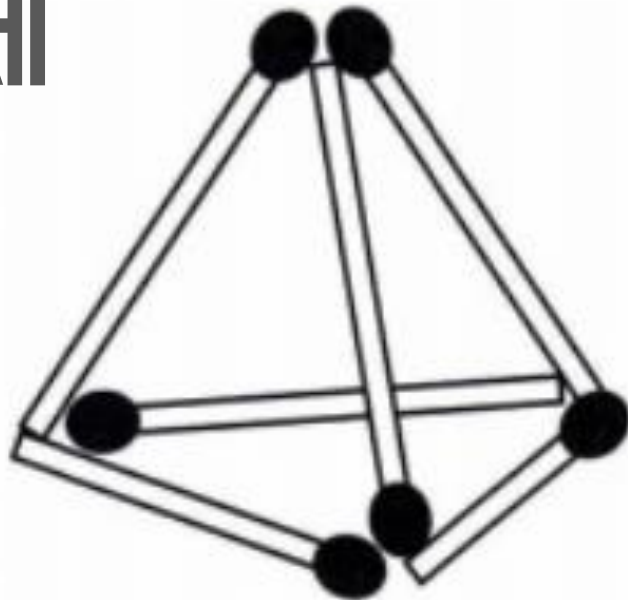


X

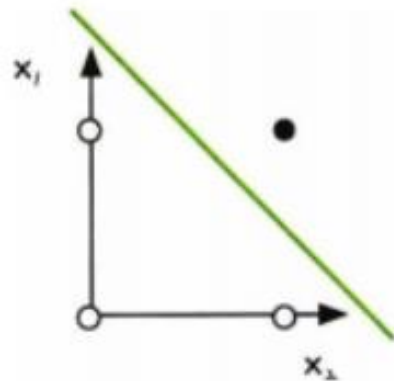


X

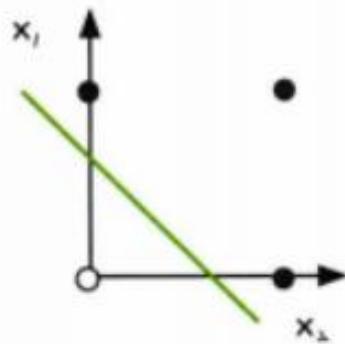
고정관념의 문제



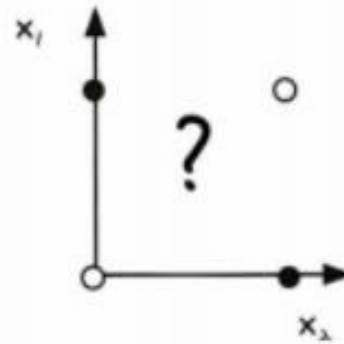
퍼셉트론의 한계



AND

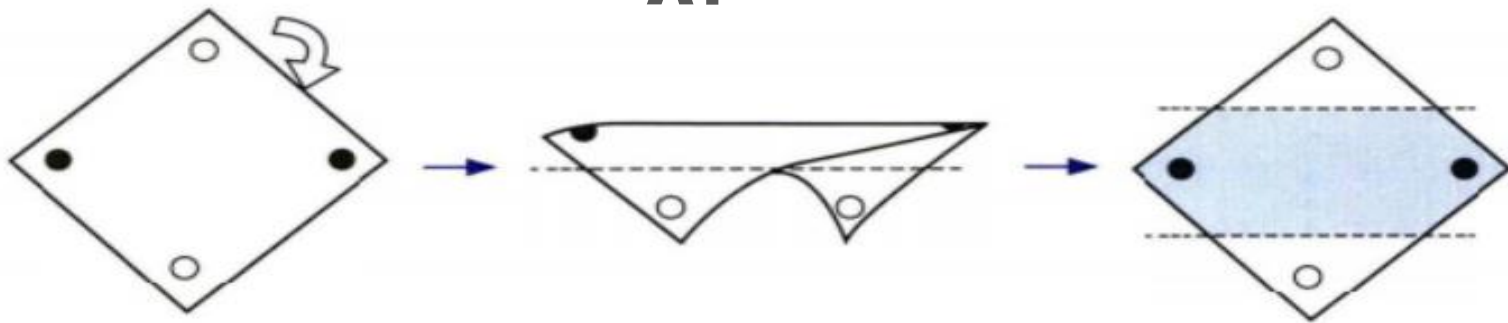


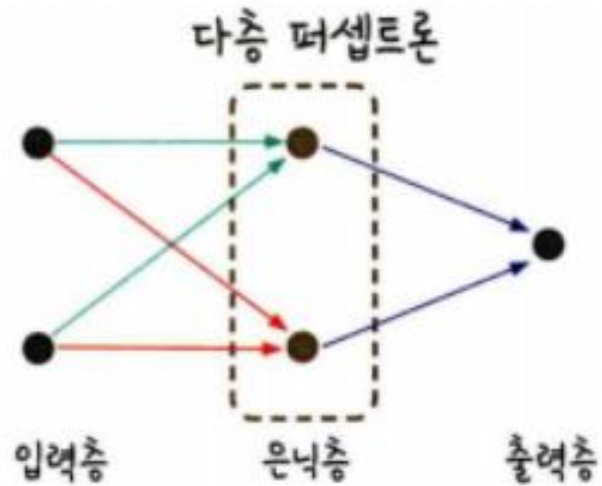
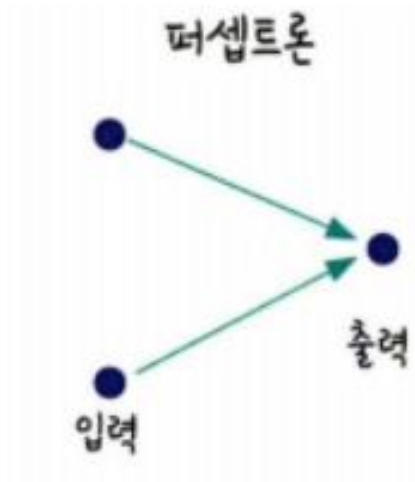
OR

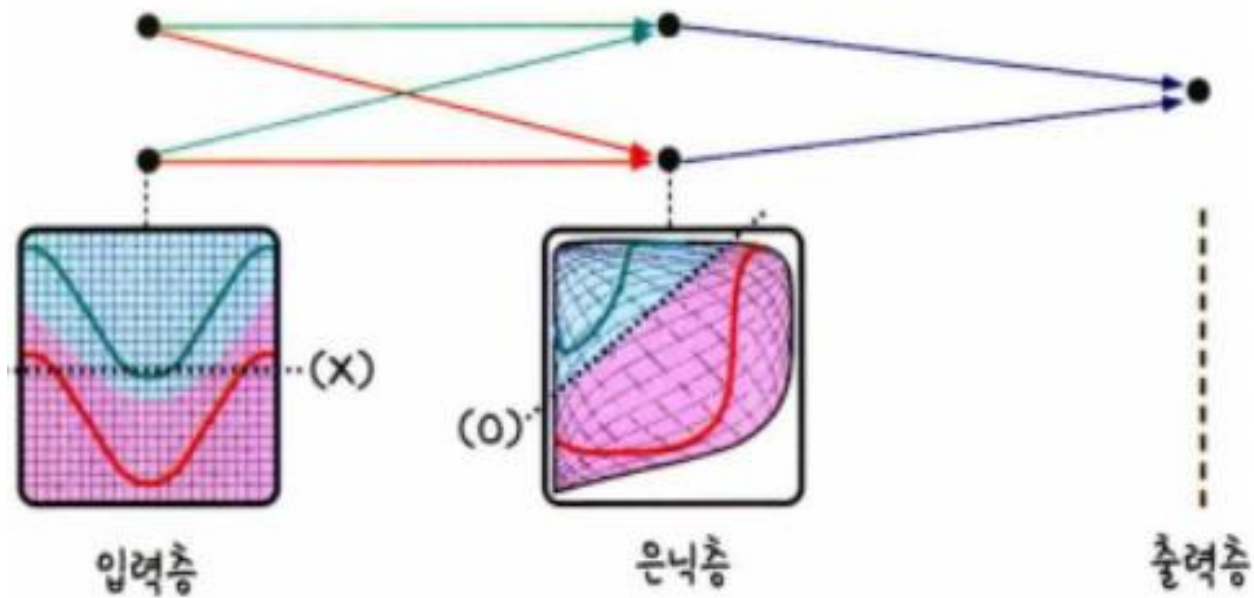


XOR

종이(좌표평면)를 휘어버리 자





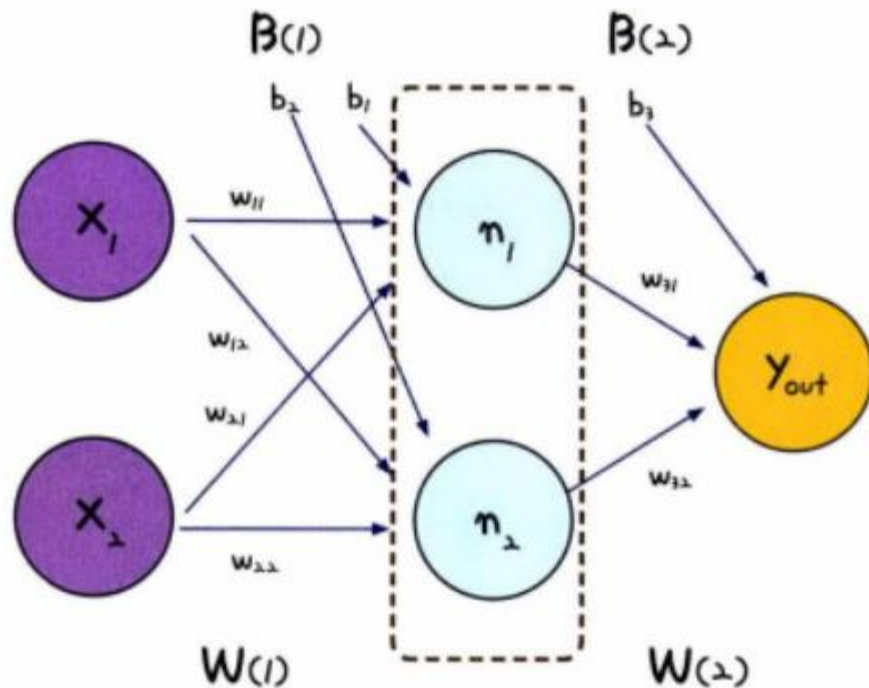


다층 퍼셉트론의 설계

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

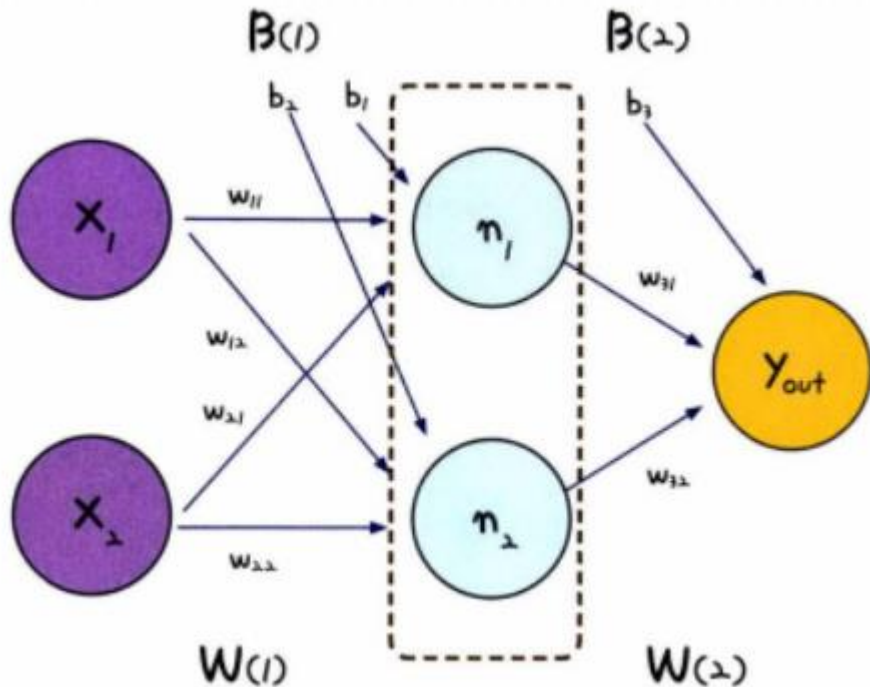
$$n_2 = \sigma(x_1 w_{12} + x_2 w_{22} + b_2)$$

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



가중치(W) & 바이어스(b)

$$W(1) = \begin{bmatrix} w_{11} & w_{12} \\ w_{21} & w_{22} \end{bmatrix} \quad B(1) = \begin{bmatrix} b_1 \\ b_2 \end{bmatrix}$$
$$W(2) = \begin{bmatrix} w_{31} \\ w_{32} \end{bmatrix} \quad B(2) = [b_3]$$



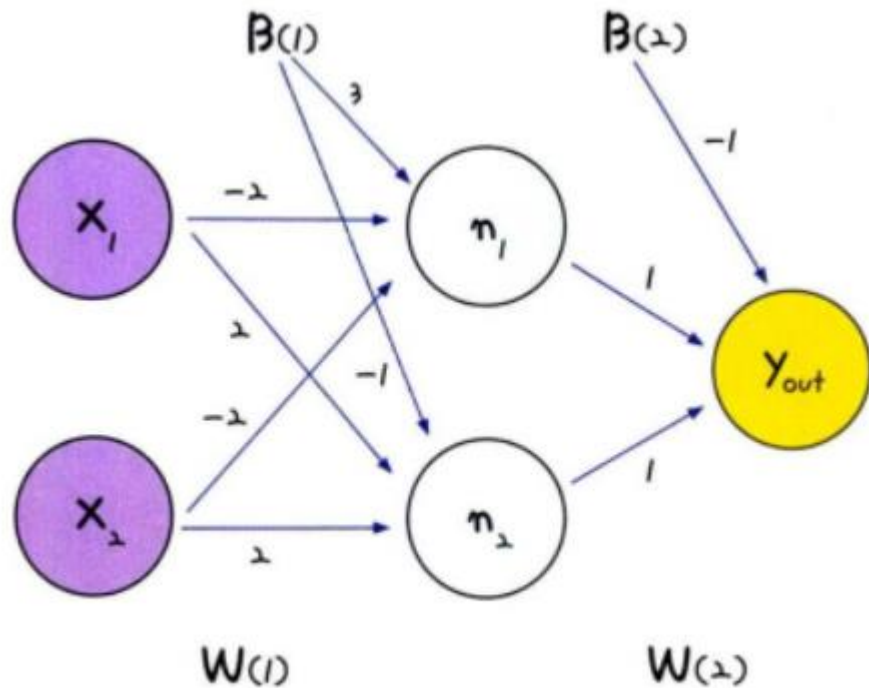
XOR 문제 해결

XOR 진리표

x_1	x_2	결괏값
0	0	0
0	1	1
1	0	1
1	1	0

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

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



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

실습 코드 - (XOR 문제 해결)

```
[28] import numpy as np
```

```
[29] #가중치와 바이어스  
w11 = np.array([-2,-2])  
w12 = np.array([2,2])  
w2 = np.array([1,1])  
b1 = 3  
b2 = -1  
b3 = -1
```

실습 코드 - (XOR 문제 해결)

```
[30] #퍼셉트론
def MLP(x,w,b):
    y = np.sum(w*x) + b
    if y <= 0:
        return 0
    else:
        return 1
```

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

```
[31] #OR 게이트
def OR(x1,x2):
    return MLP(np.array([x1,x2]),w12,b2)

#NAND 게이트
def NAND(x1,x2):
    return MLP(np.array([x1,x2]),w11,b1)

#AND 게이트
def AND(x1,x2):
    return MLP(np.array([x1,x2]),w2,b3)

#XOR 게이트
def XOR(x1,x2):
    return AND(NAND(x1,x2),OR(x1,x2))
```

실습 코드 - (XOR 문제 해결)



```
if __name__ == '__main__':  
    for x in [(0,0),(1,0),(0,1),(1,1)]:  
        y = XOR(x[0],x[1])  
        print("입력 값: " +str(x) + "출력 값: "+str(y))
```

```
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
```

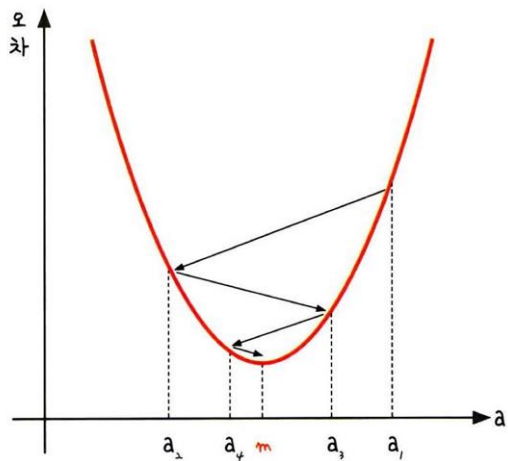
어떻게 W (가중치) & b (바이어스) 값을 줄여나갈 수 있을까?



8. 오차 역전파

단일 퍼셉트론
다중 퍼셉트론

확장



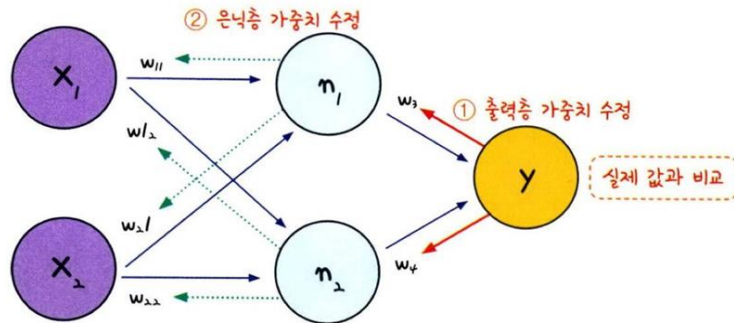
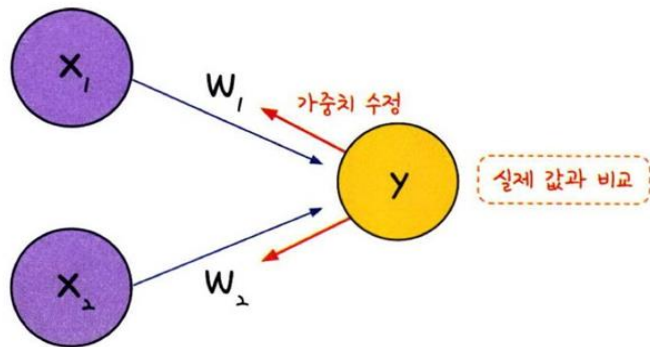
경사하강법

$$W(t+1) = W_t - \frac{\partial \text{오차}}{\partial W}$$

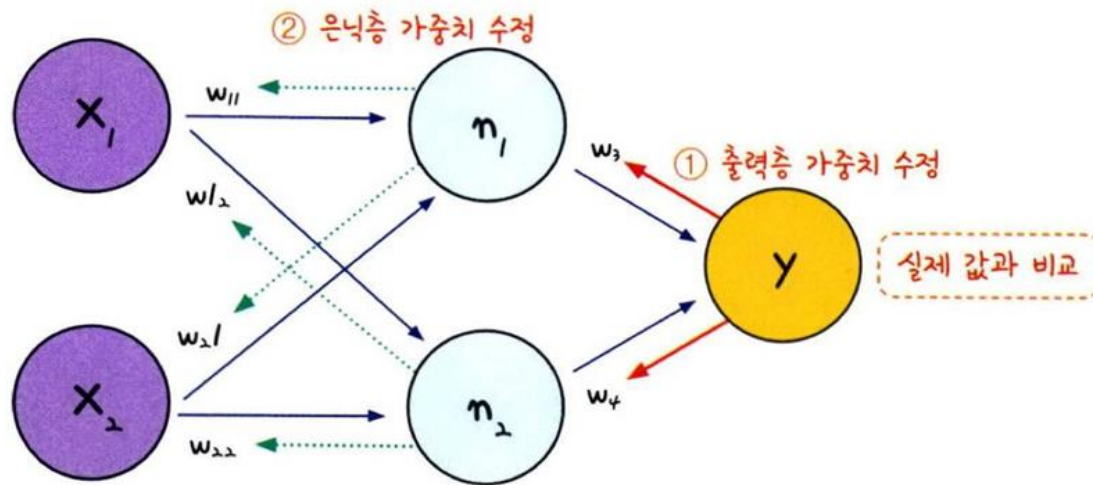
오차 역전파

단일 퍼셉트론
다중 퍼셉트론

확장



오차 역전파

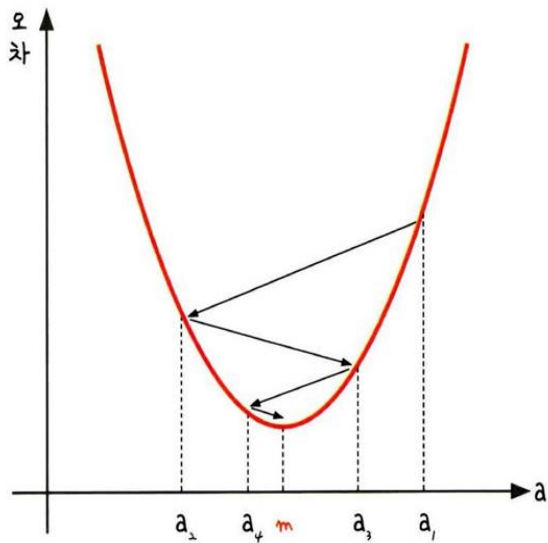


오차 역전파 (가중치 수정 작업 수식)

새 가중치는 현 가중치에서 '가중치에 대한 기울기'를 뺀 값!

$$W(t+1) = W_t - \frac{\partial \text{오차}}{\partial W}$$

오차 역전파 (수식에 대한 이해)



새 가중치는 현 가중치에서 '가중치에 대한 기울기'를 뺀 값

$$W(t+1) = W_t - \frac{\partial \text{오차}}{\partial W}$$

Ex

가중치 a & $a-1$ ($0 < a < 1$)

구해야하는 결과값 100

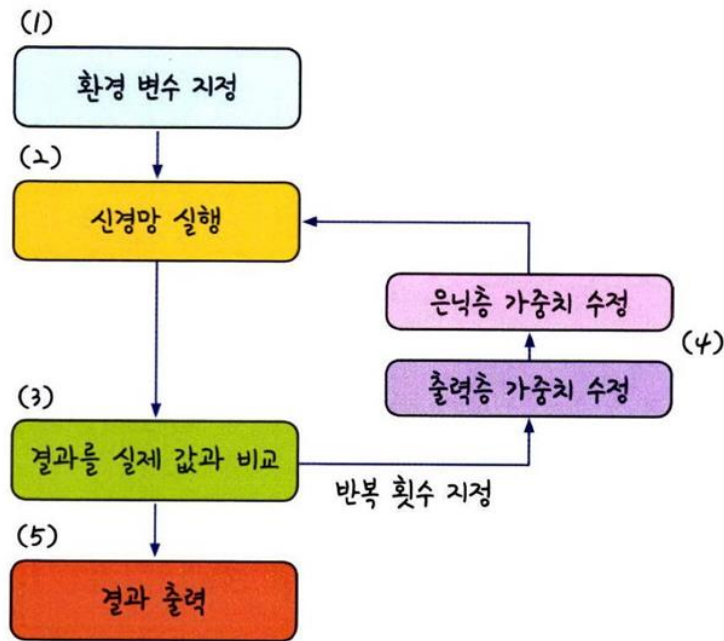
1 & 0 -> 130 ; 0.2

0.8 & 0.2 -> 90 ; 0.08

0.88 & 0.12 -> 97 ; 0.001

신경망의 구현 과정

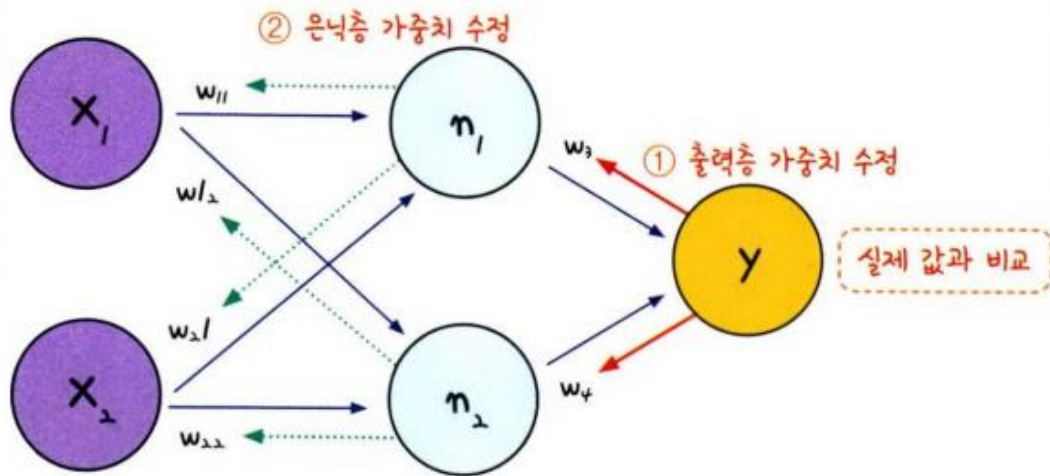
- 1 | 환경 변수 지정: 환경 변수에는 입력 값과 타깃 결괏값이 포함된 데이터셋, 학습률 등이 포함됩니다. 또한, 활성화 함수와 가중치 등도 선언되어야 합니다.
- 2 | 신경망 실행: 초깃값을 입력하여 활성화 함수와 가중치를 거쳐 결괏값이 나오게 합니다.
- 3 | 결과를 실제 값과 비교: 오차를 측정합니다.
- 4 | 역전파 실행: 출력층과 은닉층의 가중치를 수정합니다.
- 5 | 결과 출력



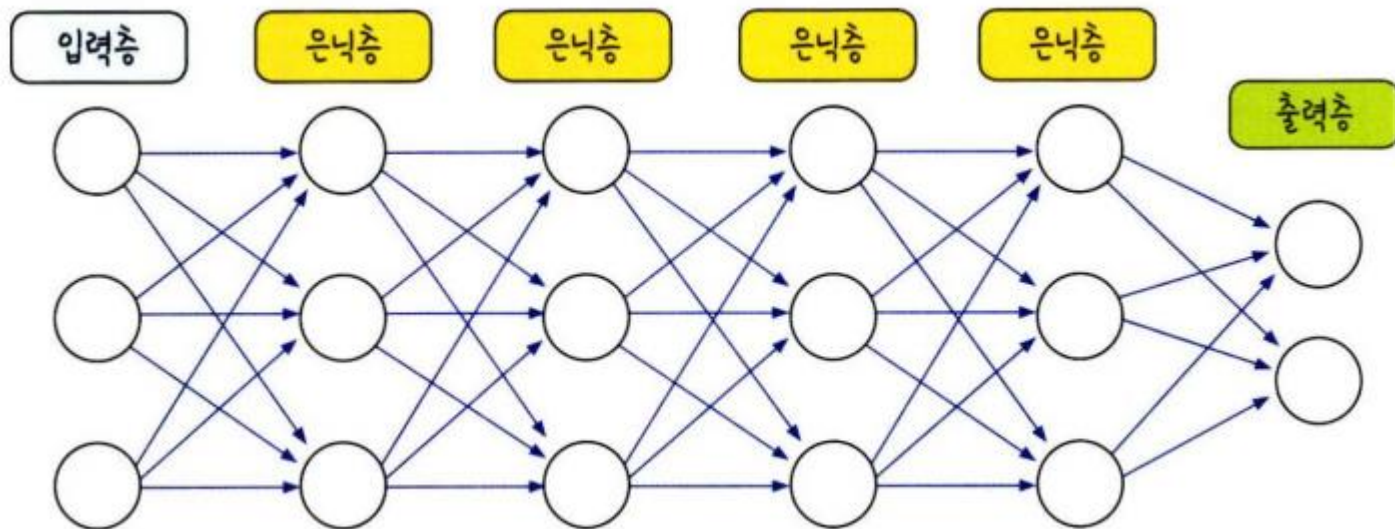


9. 신경망 to 딥러닝

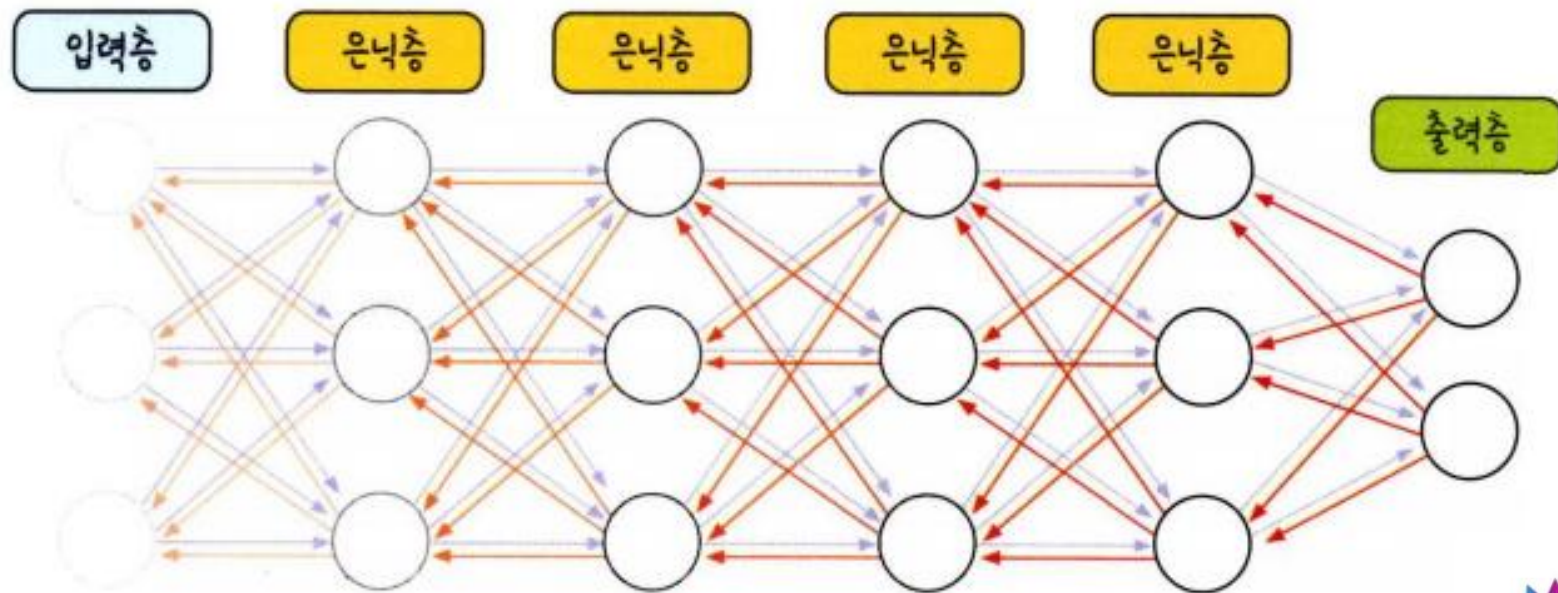
preview) 오차역전파

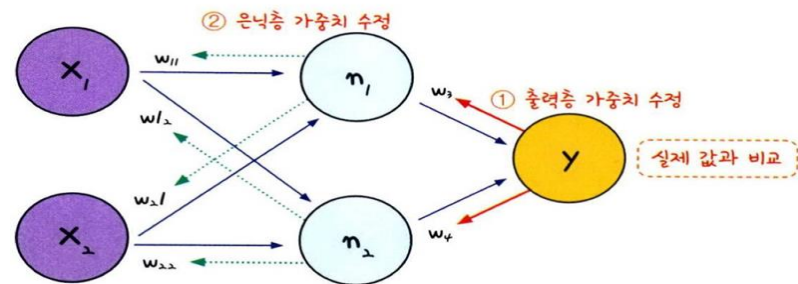


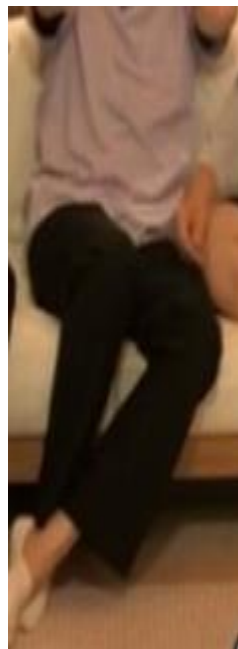
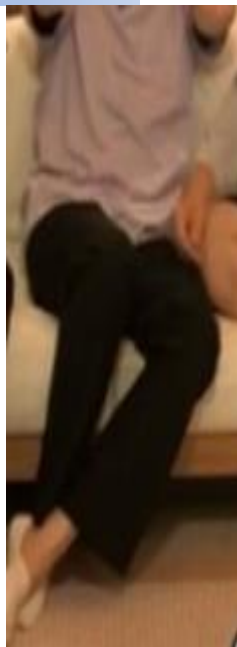
신경망 구성

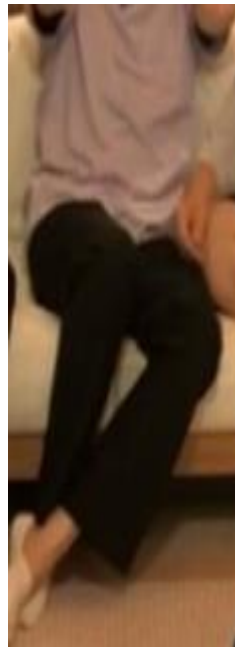
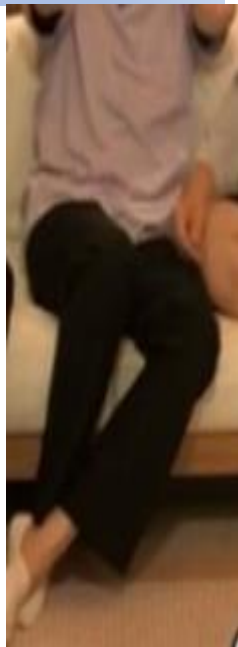


기울기 소실(Vanishing gradient)

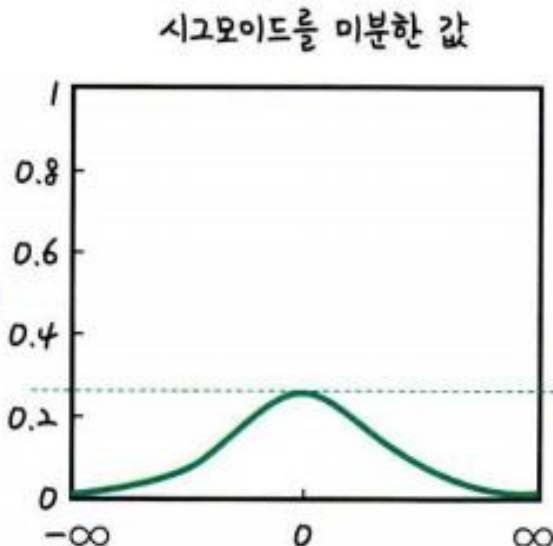
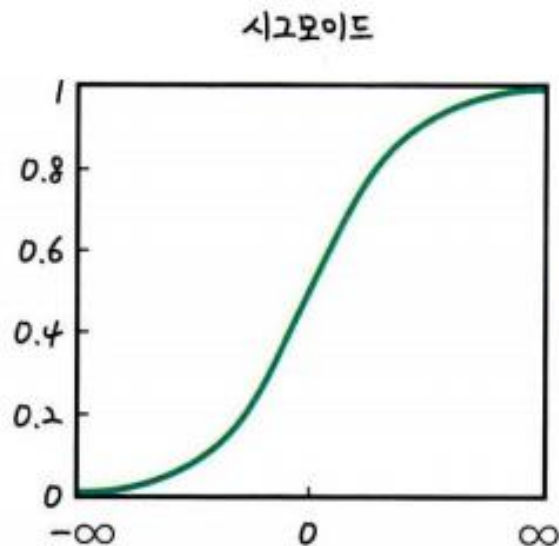








한 번 미분했을 뿐인데...



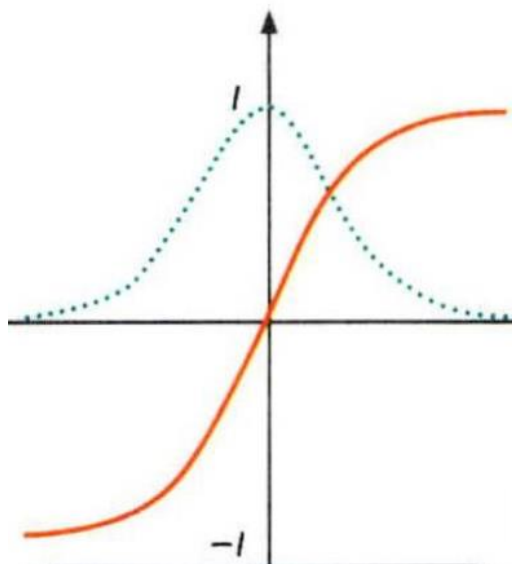
최대치가 0.3

하이퍼볼릭 탄젠트 (hyperbolic tangent)

- sigmoid 보완하고자 제안된 함수
- 범위를 (-1.1) 사이로 확장
- 전반적인 성능 개선
- 아직도 기울기 소실은 발생

$$\tanh(x) = \frac{e^x - e^{-x}}{e^x + e^{-x}}$$

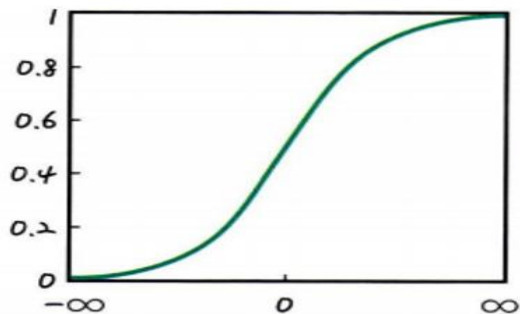
위 아래로 더 늘려보자



하이퍼볼릭 탄젠트 함수

$$f(x) = \tanh(x)$$

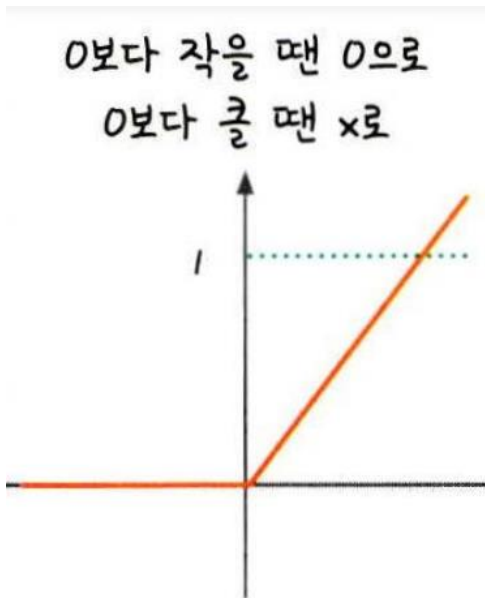
시그모이드



렐루 함수 (ReLU : Rectified Linear Unit)

- 가장 인기있는 활성화 함수
- 미분 값이 0 또는 1
- exp()를 실행하지 않아 6배 정도 빠르게 학습이 진행됨

$$\text{ReLU}(x) = \max(0, x)$$

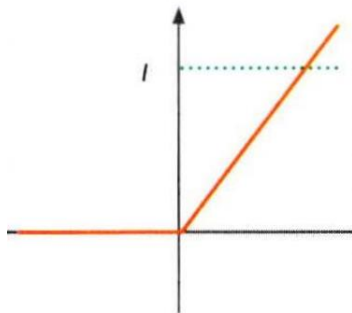


렐루 함수

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

소프트플러스 함수 (Softplus)

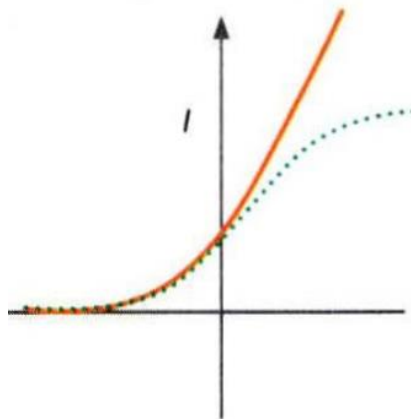
0보다 작을 땐 0으로
0보다 클 땐 x로



렐루 함수

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

0을 만드는 기준을
완화시키자



소프트플러스 함수

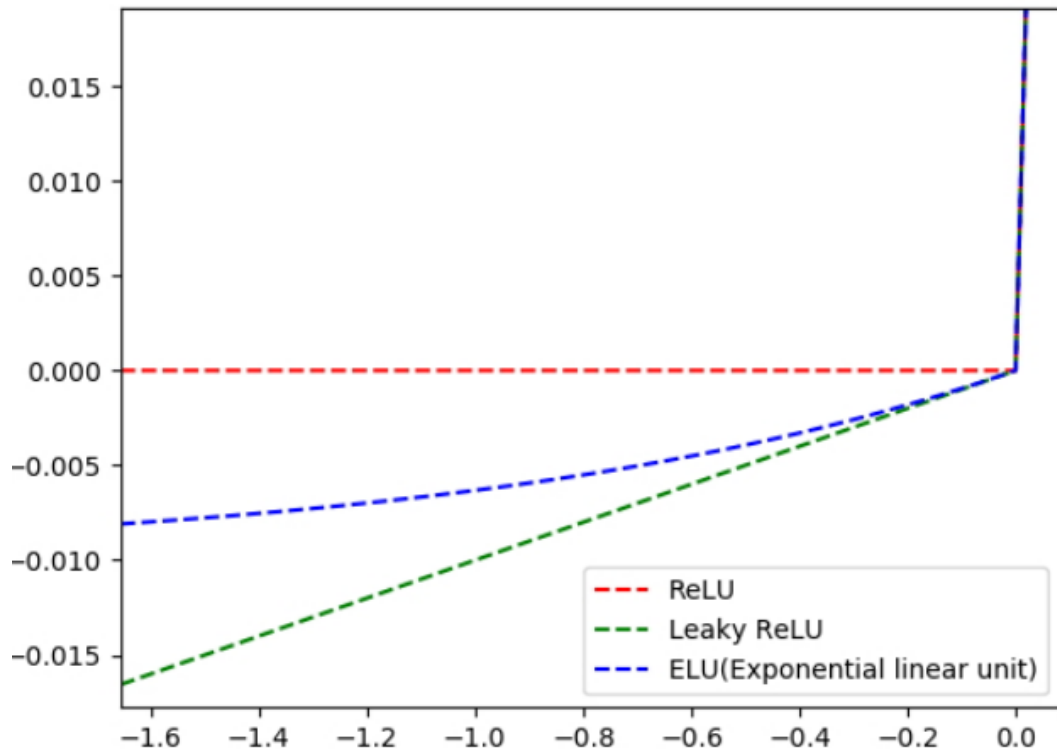
$$f(x) = \log(1 + e^x)$$

Leaky ReLU

“dying ReLU” 현상을 해결하기
위한 함수

차이점: $x < 0$ 인 경우
작은 기울기(0.01)를 부여함

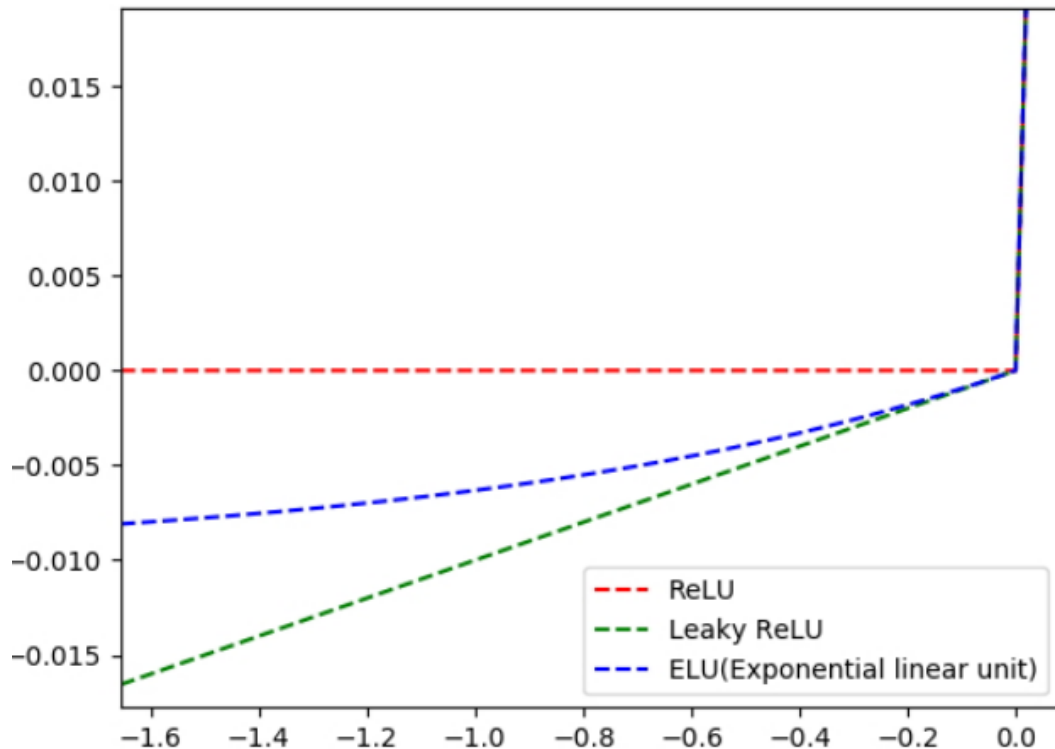
$$\text{Leaky ReLU}(x) = \max(0.01 * x, x)$$



ELU

- 역시 dying ReLU를 해결함
- 출력값이 보다 '정규화'에 가까움
- 단점) $\exp()$ 계산 비용이 높음

$$f(x) = \begin{cases} x & \text{if } x > 0 \\ \alpha(e^x - 1) & \text{if } x \leq 0 \end{cases}$$



결론

sigmoid 함수의 한계를 극복하기 위해 다양한 함수의 발생

- 먼저, ReLU 함수를 사용하기 (가장 많이 사용되는 1 pick)
- 다음으로 Leaky ReLU, ELU 등등을 시도
- 성능이 좋아 질 수 있는 **가능성**이 있음 (반드시는 X)
- 앞으로 심층 신경망(Deep Neural Network)에서는 Sigmoid는 피한다

ELU, Leaky ReLU > ReLU > tanh > sigmoid

근거(back data)

The compatibility of activation functions and initialization.
Dataset: CIFAR-10

Init method	maxout	ReLU	tanh	Sigmoid
LSUV	93.94	92.11	89.28	n/c
OrthoNorm	93.78	91.74	89.48	n/c
OrthoNorm-MSRA scaled	–	91.93	–	n/c
Xavier	91.75	90.63	89.82	n/c
MSRA	n/c†	90.91	89.54	n/c

n/c symbol stands for “failed to converge, Architecture FitNets-17

경사 하강법의 개선

기존의 경사 하강법

: **전체 데이터**를 사용(이용)하여

구하고자 하는 가중치를
정확하게 추적하여 찾아감

BUT 전체 데이터를 사용해서
→ 계산량이 너무 많다

확률적 경사 하강법(SGD)

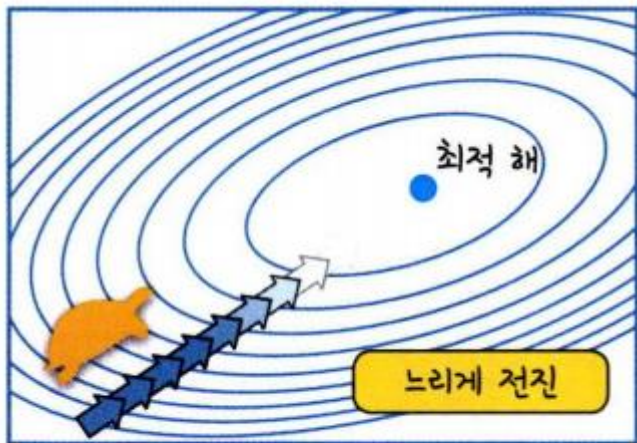
: **랜덤 추출된 일부 데이터** 사용(이용)

하여 구하고자 하는 가중치를 더욱
빨리 추적하여 찾아감

빠른 속도로 업데이트도 가능

경사 하강법의 개선

경사 하강법



경사 하강법

확률적 경사 하강법(SGD)



확률적 경사 하강법


momentum

미국[mou | mentəm]  영국식[mə | mentəm] 


명사


1

The fight for his release gathers **momentum** each day. 

그의 석방을 위한 투쟁에 나날이  이 불고 있다.

2

The vehicle gained **momentum** as the road dipped. 

도로가 내리막이 되자 차에  붙었다.

momentum

미국[mou | mentəm]  영국식[mə | mentəm] 


명사

1 (일의 진행에 있어서의) 탄력[가속도]

The fight for his release gathers **momentum** each day. 

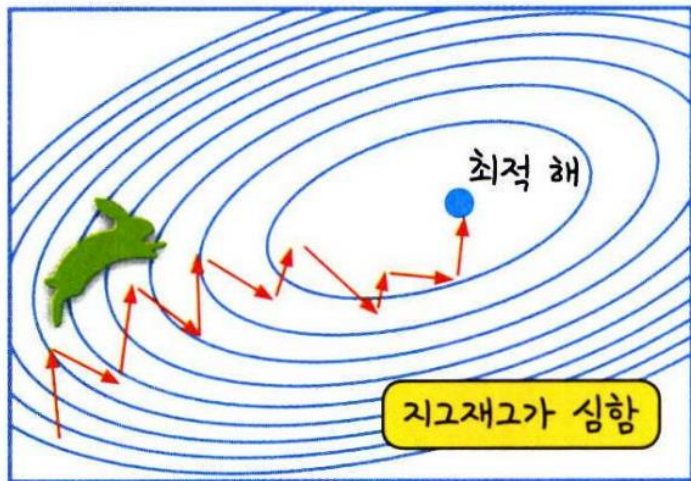
그의 석방을 위한 투쟁에 나날이 탄력[가속도]이 붙고 있다.

2 가속도

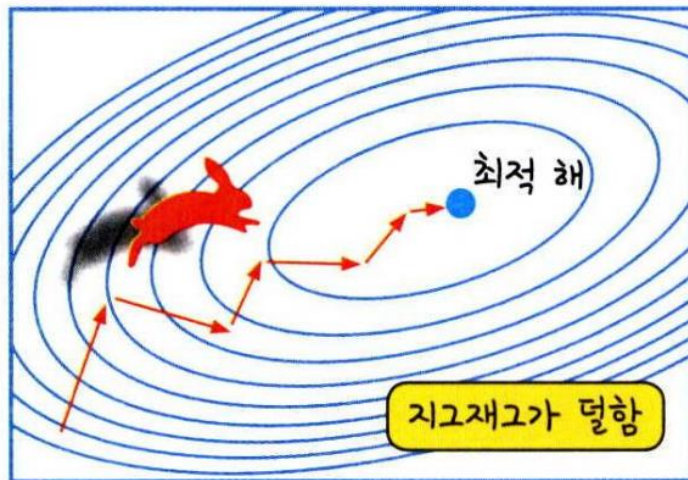
The vehicle gained **momentum** as the road dipped. 

도로가 내리막이 되자 차에 가속도가 붙었다.

모멘텀 확률 경사 하강법



확률적 경사 하강법



모멘텀을 적용한 확률적 경사 하강법

고급 경사 하강법	개요	효과	케라스 사용법
확률적 경사 하강법 (SGD)	랜덤하게 추출한 일부 데이터를 사용해 더 빨리, 자주 업데이트를 하게 하는 것	속도 개선	<code>keras.optimizers.SGD(lr = 0.1)</code> 케라스 최적화 함수를 이용합니다.
모멘텀 (Momentum)	관성의 방향을 고려해 진동과 폭을 줄이는 효과	정확도 개선	<code>keras.optimizers.SGD(lr = 0.1, momentum = 0.9)</code> 모멘텀 계수를 추가합니다.
네스테로프 모멘텀 (NAG)	모멘텀이 이동시킬 방향으로 미리 이동해서 그래디언트를 계산. 불필요한 이동을 줄이는 효과	정확도 개선	<code>keras.optimizers.SGD(lr = 0.1, momentum = 0.9, nesterov = True)</code> 네스테로프 옵션을 추가합니다.
아다그라드 (Adagrad)	변수의 업데이트가 잦으면 학습률을 적게 하여 이동 보폭을 조절하는 방법	보폭 크기 개선	<code>keras.optimizers.Adagrad(lr = 0.01, epsilon = 1e - 6)</code> 아다그라드 함수를 사용합니다. ※ 참고: 여기서 <code>epsilon</code> , <code>rho</code> , <code>decay</code> 같은 파라미터는 바꾸지 않고 그대로 사용하기를 권장하고 있습니다. 따라서 <code>lr</code> , 즉 <code>learning rate</code> (학습률) 값만 적절히 조절하면 됩니다.
알엠에스프롭 (RMSProp)	아다그라드의 보폭 민감도를 보완한 방법	보폭 크기 개선	<code>keras.optimizers.RMSprop(lr = 0.001, rho = 0.9, epsilon = 1e - 08, decay = 0.0)</code> 알엠에스프롭 함수를 사용합니다.


고급 경사 하강법	개요	효과	케라스 사용법
아담(Adam)	모멘텀과 알엠에스프롭 방법을 합친 방법	정확도와 보폭 크기 개선	<pre>keras.optimizers.Adam(lr = 0.001, beta_1 = 0.9, beta_2 = 0.999, epsilon = 1e - 08, decay = 0.0)</pre> 아담 함수를 사용합니다.

아담(Adam): 현재 가장 많이 사용되는 고급
경사법

```
from tensorflow.keras import optimizers
from tensorflow.keras import activations
from tensorflow.keras import losses
```

* loss: '훈련 손실값' -> 작을수록(0에 수렴할수록) 모델에 수렴한

```
losses.MSE() # MSE
losses.binary_crossentropy() # 로지스틱 회귀(이진 분류) 에서
losses.categorical_crossentropy() # 다중 분류에서
```



```
activations.relu() # ReLu  
activations.sigmoid() # 시그모이드  
activations.tanh() # 하이퍼볼릭 탄젠트
```

```
optimizers.Adam(learning_rate=0.002) # Adam  
optimizers.SGD(learning_rate=0.002) # 확률적 경사하강법 SGD  
optimizers.RMSprop(learning_rate=0.001) # RMSprop
```