

딥러닝팀

1팀

김예찬
박시언
박윤아
정승민
김민

CONTENTS

1. 머신러닝

2. 퍼셉트론

3. 신경망

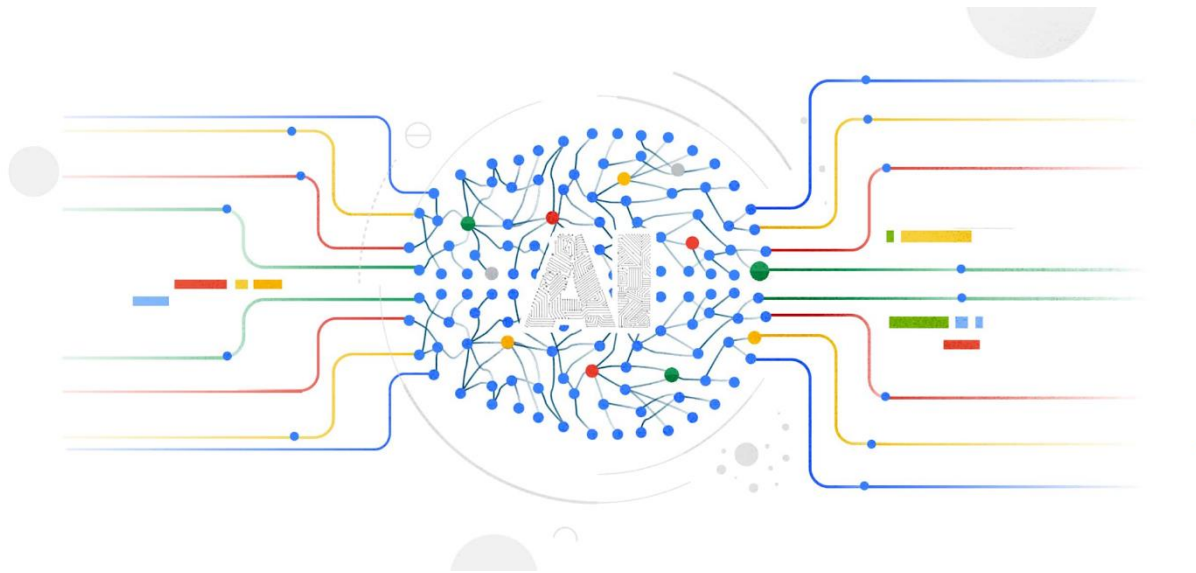
4. 성능 향상 기법

5. 마무리

1

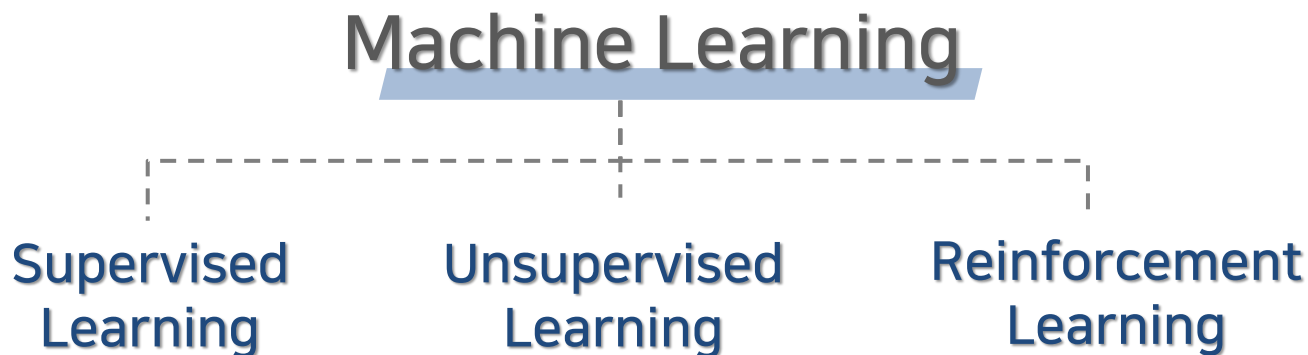
머신러닝

머신러닝이란? Machine learning



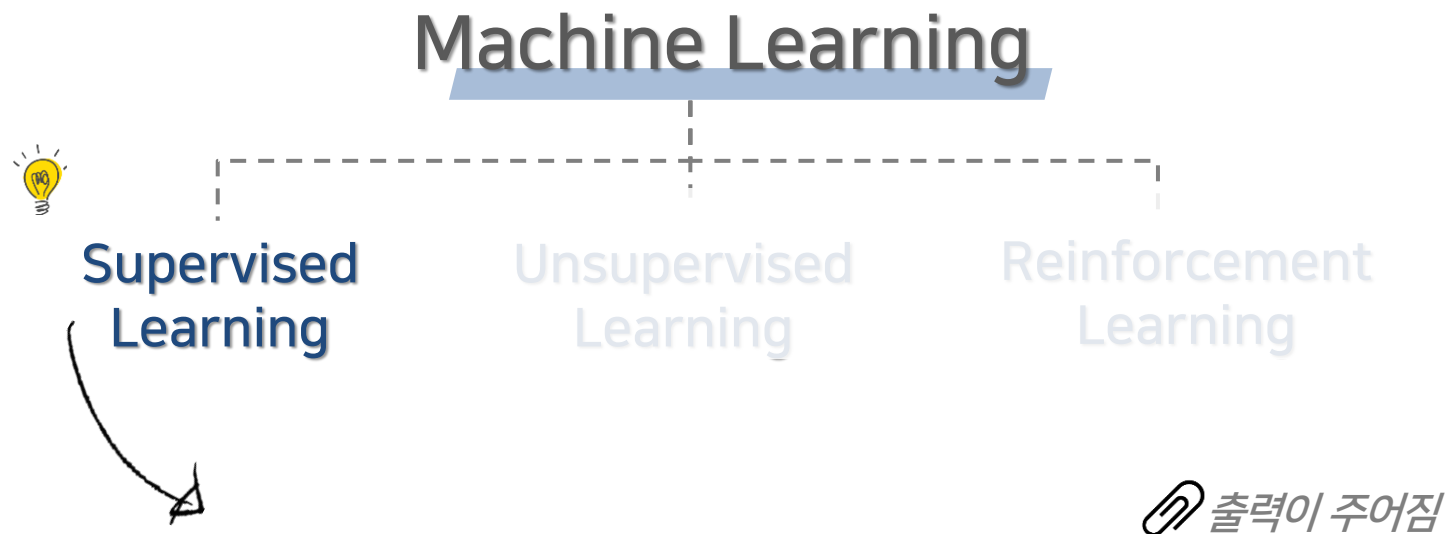
데이터로부터 **학습**을 통해 성능을 개선할 수 있는
컴퓨터 알고리즘의 한 분야

머신러닝의 종류



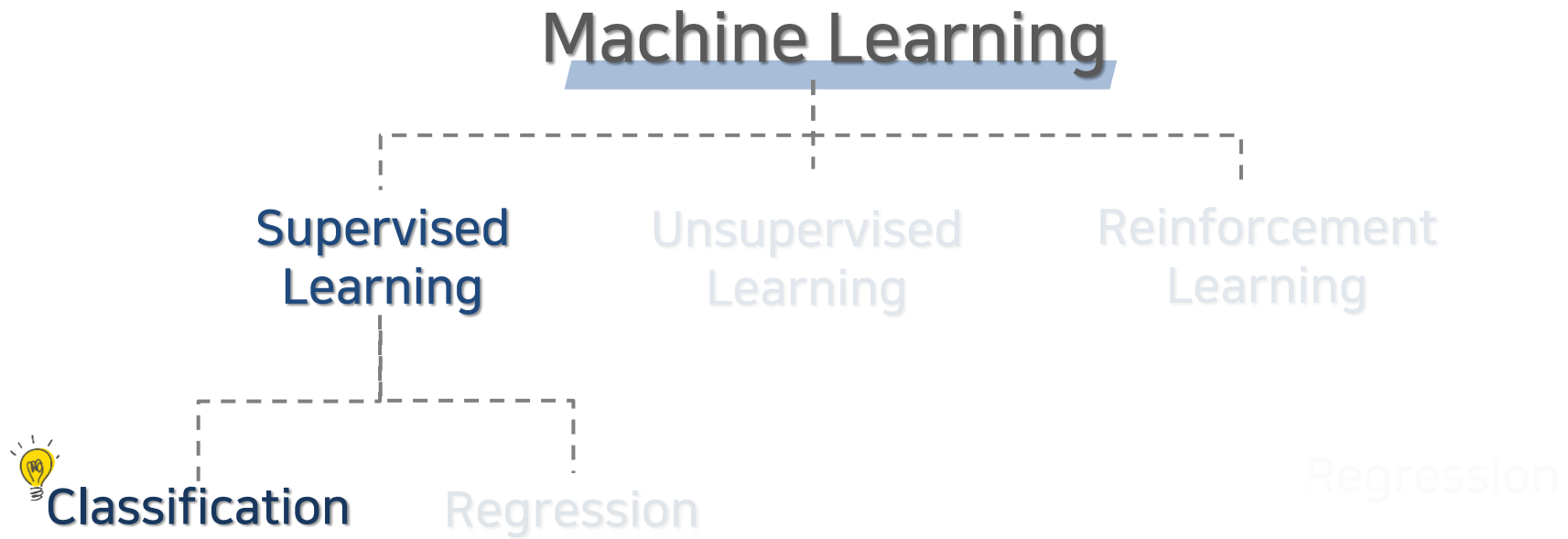
머신러닝에는
지도학습, 비지도학습과 강화학습이 있음

머신러닝의 종류



지도학습은 **입력**과 **출력**이 데이터로 주어졌을 때
그 입력과 출력 간의 함수관계를 유추하는 형태로 학습이 이루어짐

머신러닝의 종류



분류 문제는 예측 대상이 **범주형 자료**로 주어지는 예측과제

Ex) 로지스틱 회귀모형, SVM, 랜덤포레스트 분류모델

머신러닝의 종류

Machine Learning

Supervised
LearningUnsupervised
LearningReinforcement
Learning

Classification



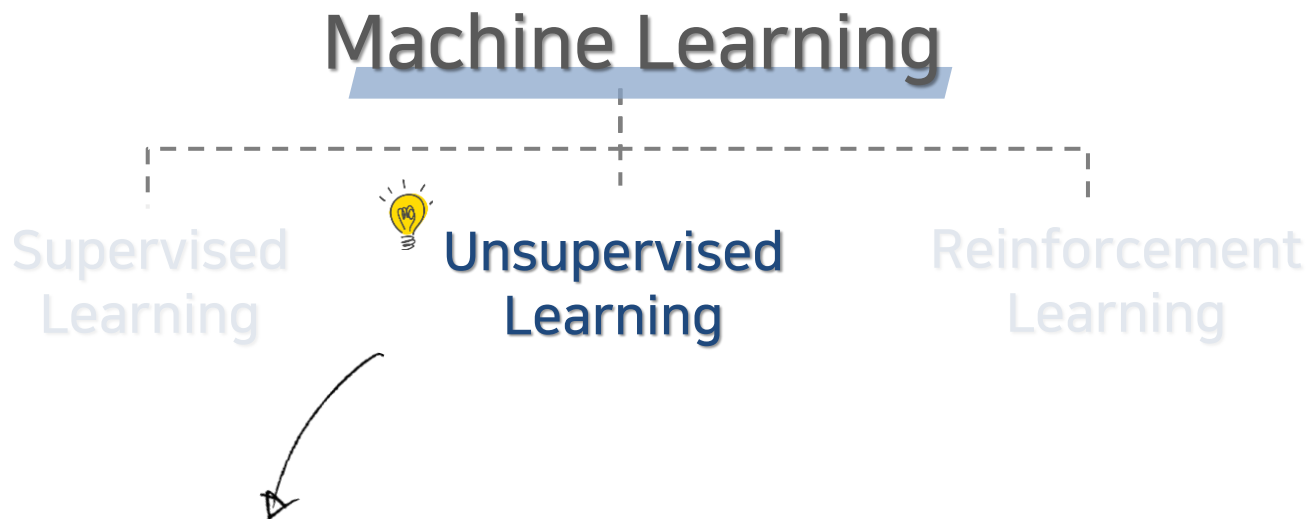
Regression

Regression

회귀문제는 예측 대상이 **연속형 자료**로 주어지는 예측과제

Ex) 라쏘, 릿지 회귀모형

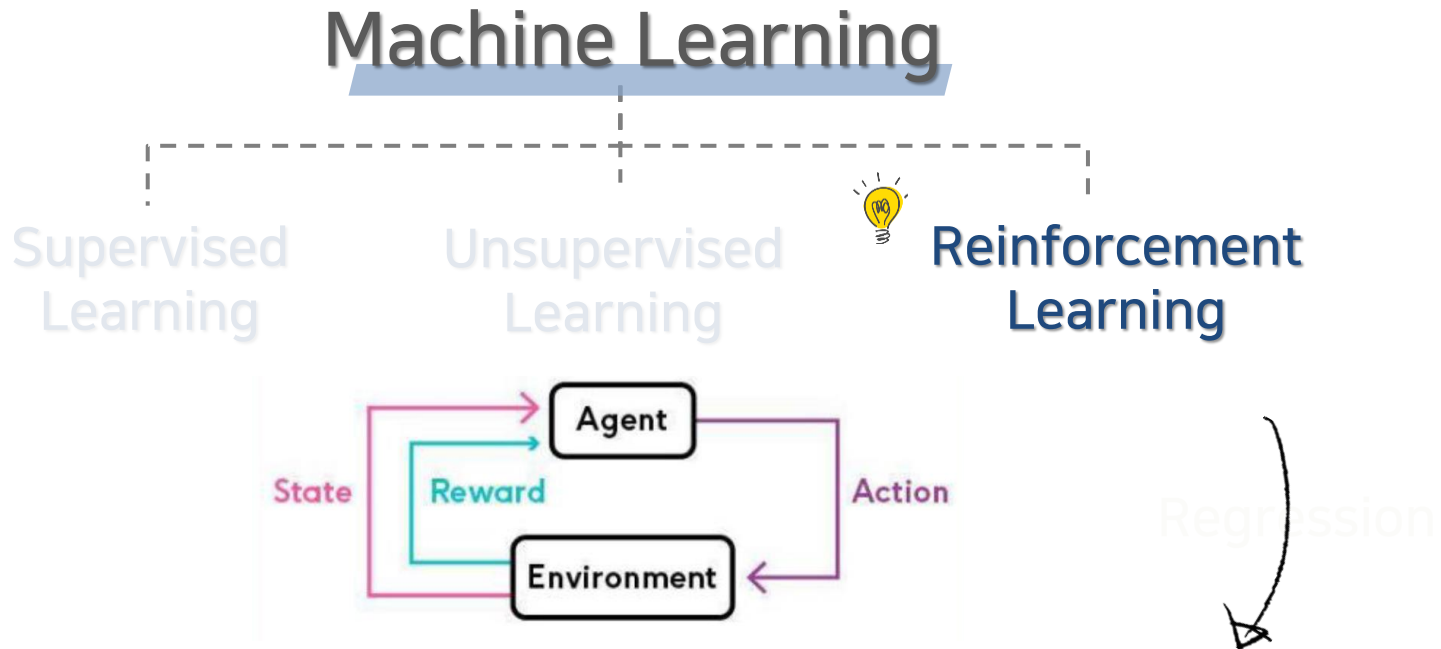
머신러닝의 종류



비지도학습은 **출력 값**을 알려주지 않고
스스로 **규칙성**을 찾아내어 모델을 구축하고 학습하는 것을 의미
Ex) 군집화, 이상탐지, 차원축소

이상탐지는 딥러닝에서도 자주 다뤄지는 주제!

머신러닝의 종류




강화학습은 시행착오를 통해 **보상**하는 행동을 학습하는 모델로,
지도학습, 비지도 학습과 완전히 분리됨

딥러닝 vs 머신러닝

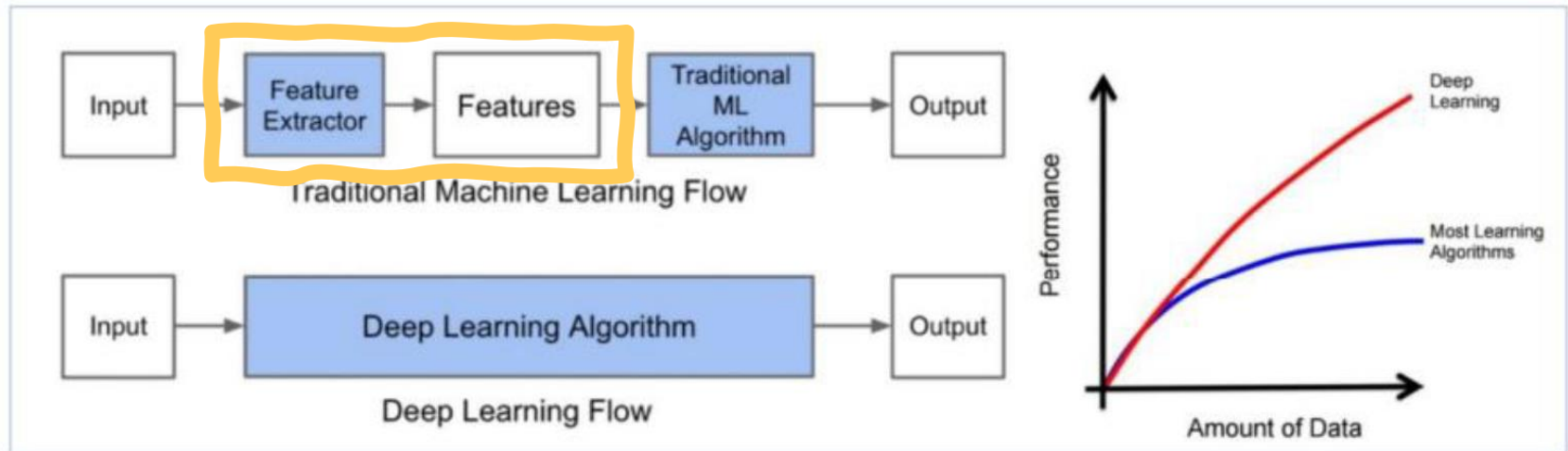


Feature Engineering

 Ex) 사람을 분류하기 위한 데이터에서 눈, 코, 입과 같이 사람을 구분할 수 있는 특징을 찾는 것

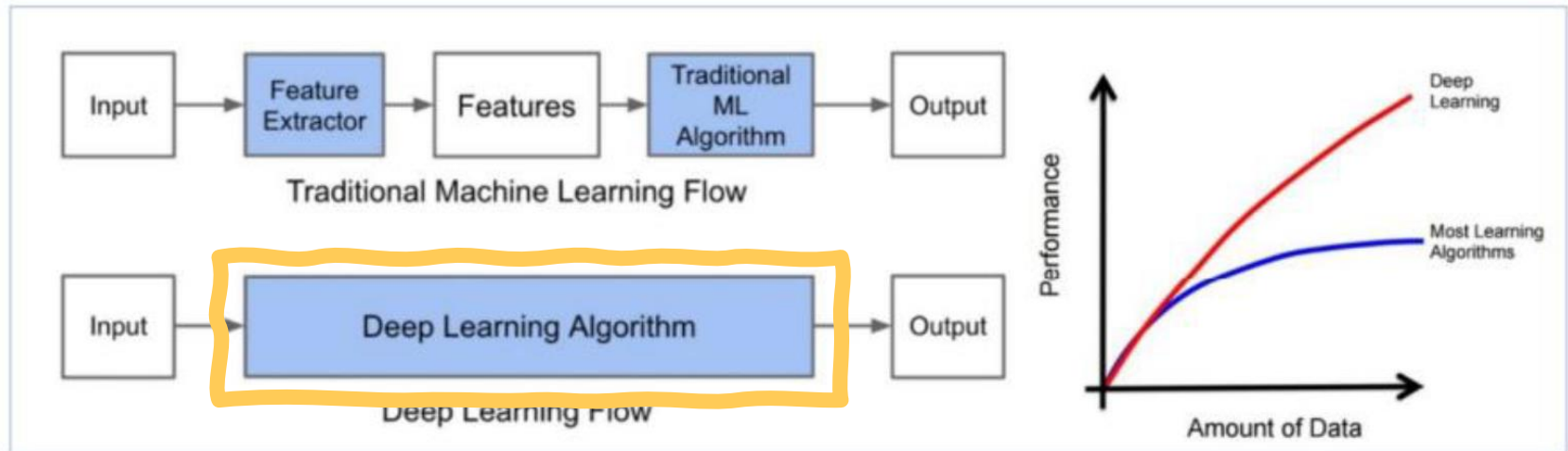
모델 성능과 해석력을 높이기 위해 데이터를 가공하는 것
사람이 직접 하는 것 > 많은 시간과 지식이 필요

딥러닝 vs 머신러닝



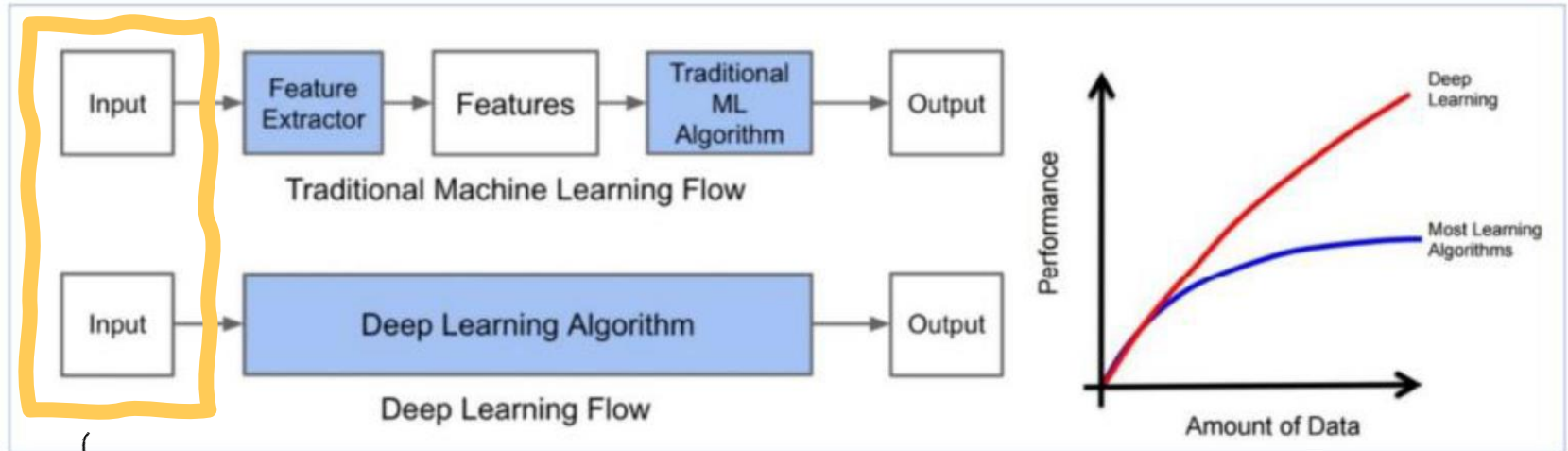
머신러닝에서는 Feature Engineering 과정에 따라
결과가 천차만별로 달라짐

딥러닝 vs 머신러닝



딥러닝은 머신러닝과 달리 **Feature Engineering**의 중요성이 낮아,
사람이 개입하지 않아도 된다는 강점을 가짐

딥러닝과 머신러닝 차이



Ex) csv데이터, 스프레드시트 데이터

머신러닝은 **정형데이터**를, 딥러닝은 **비정형데이터**를 주로 다룸

Ex) 동영상, 이미지

딥러닝의 활용 사례

딥러닝은 지도학습, 비지도학습, 강화학습 과제에 모두 적용 가능

지도학습


얼굴 인식
질병 진단

비지도학습

이미지 생성
음성 합성

강화학습

DQN 알고리즘,
REINFORCE
알고리즘

 Reinforce 알고리즘은
알파고의 학습원리

2

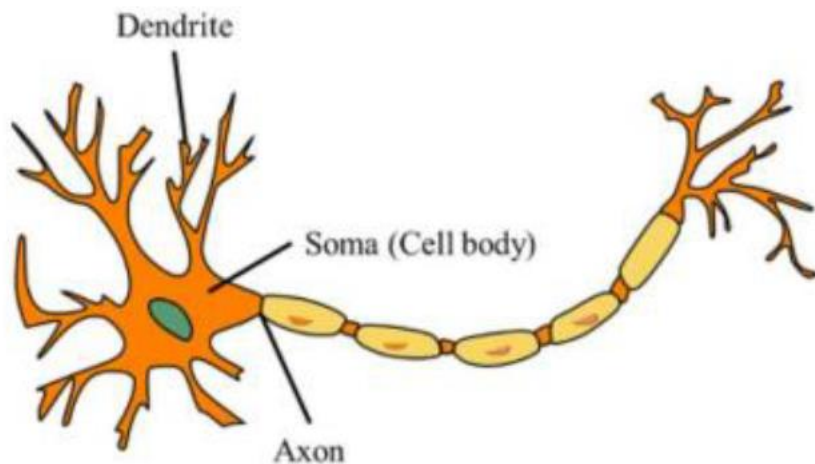
퍼셉트론

딥러닝의 정의와 작동방식에 대해 알아보자!

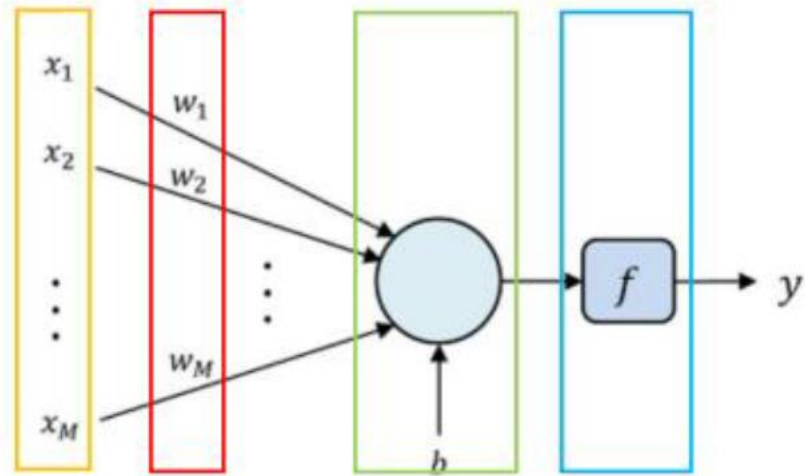
퍼셉트론 Perceptron

📎 딥러닝의 기본 단위!

다수의 입력 데이터에 대해 하나의 출력을 반환하는 형태의 알고리즘



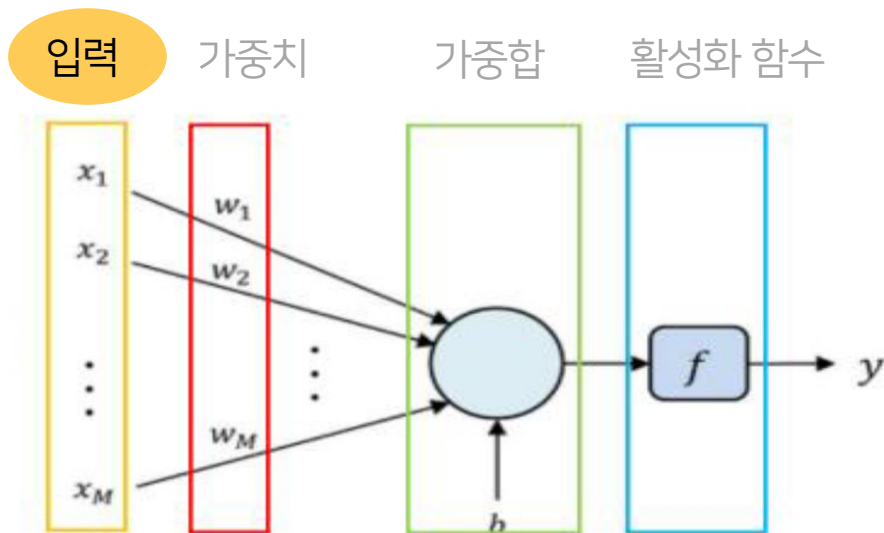
<신경 세포>



<퍼셉트론 도식화>

신경세포와 퍼셉트론의 형태가 유사함

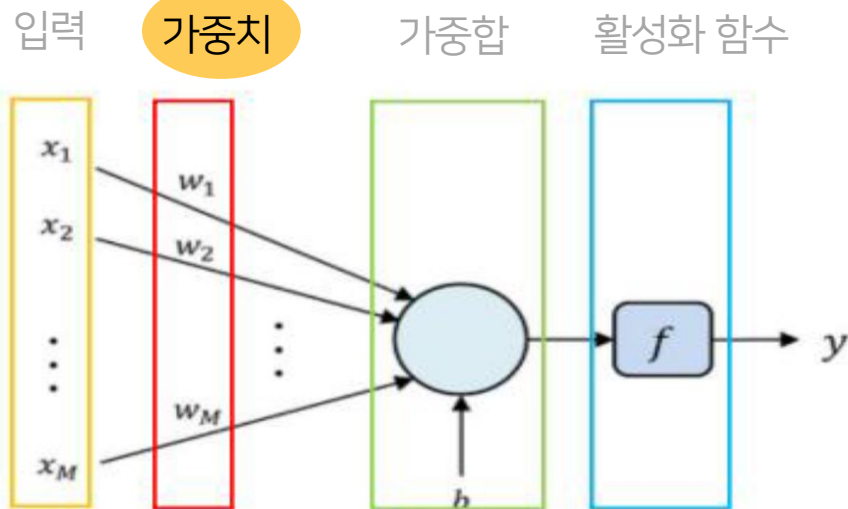
퍼셉트론 Perceptron



$$y = \begin{cases} 0 & (\sum_{i=1}^n w_i x_i + b \leq \theta) \\ 1 & (\sum_{i=1}^n w_i x_i + b \geq \theta) \end{cases}$$

임계값 \nearrow

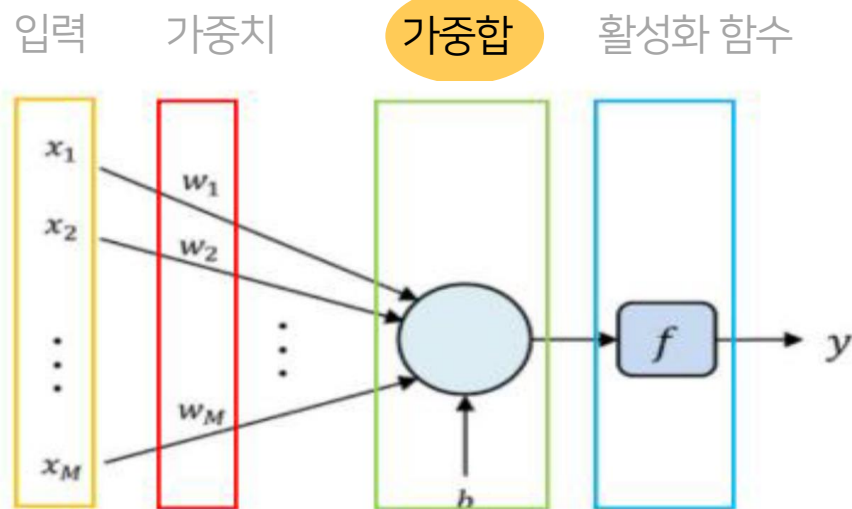
퍼셉트론 Perceptron



임계값

$$y = \begin{cases} 0 & (\sum_{i=1}^n w_i x_i + b \leq \theta) \\ 1 & (\sum_{i=1}^n w_i x_i + b \geq \theta) \end{cases}$$

퍼셉트론 Perceptron

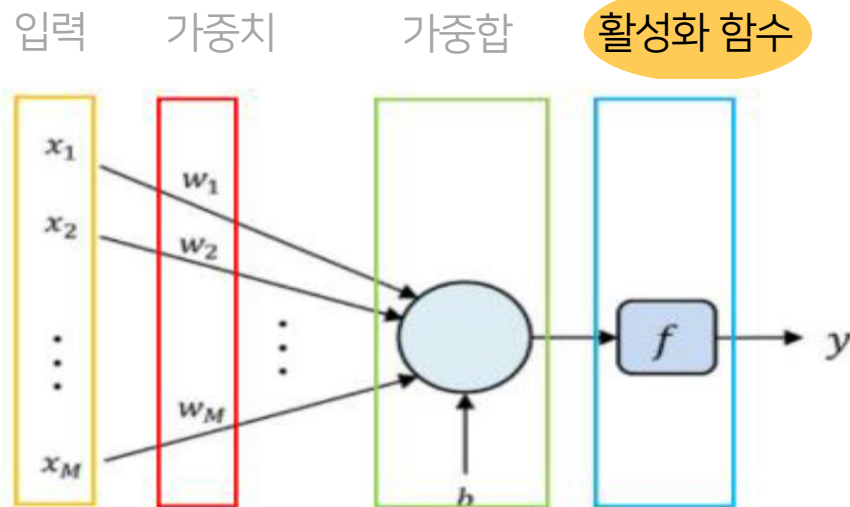


임계값 (Threshold)

$$y = \begin{cases} 0 & \left(\sum_{i=1}^n w_i x_i + b \leq \theta \right) \\ 1 & \left(\sum_{i=1}^n w_i x_i + b \geq \theta \right) \end{cases}$$

입력과 가중치를 곱하여 합하는 구간

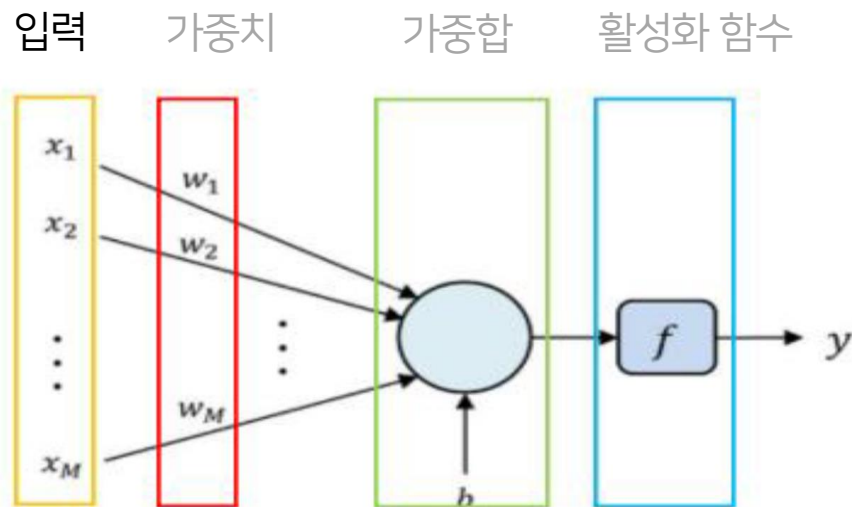
퍼셉트론 Perceptron



$$y = \begin{cases} 0 & \left(\sum_{i=1}^n w_i x_i + b \leq \theta \right) \\ 1 & \left(\sum_{i=1}^n w_i x_i + b \geq \theta \right) \end{cases}$$

← 임계값

퍼셉트론 Perceptron



임계값

$$y = \begin{cases} 0 & (\sum_{i=1}^n w_i x_i + b \leq \theta) \\ 1 & (\sum_{i=1}^n w_i x_i + b \geq \theta) \end{cases}$$

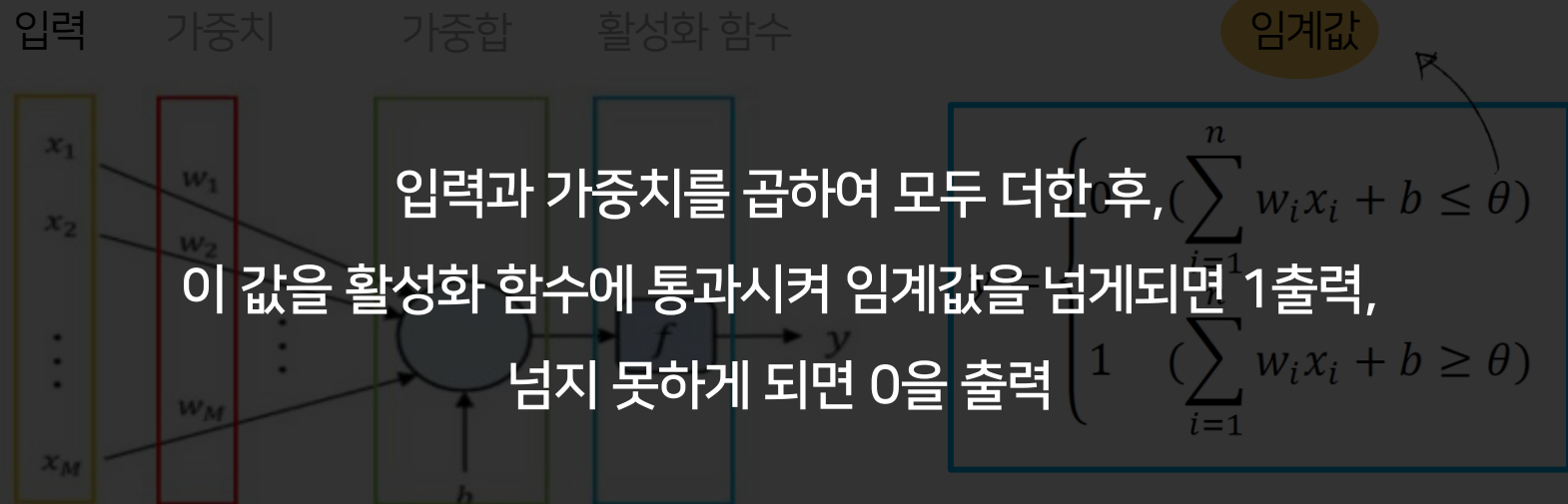
퍼셉트론 Perceptron



💡 딥러닝의 기본 단위!

다수의 입력 데이터에 대해 하나의 출력을 반환하는 형태의 알고리즘

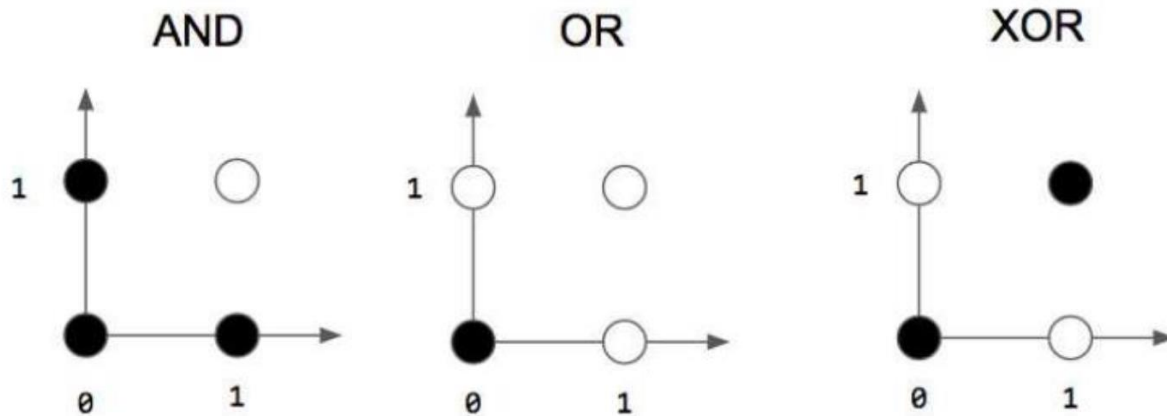
퍼셉트론의 연산과정



퍼셉트론의 한계

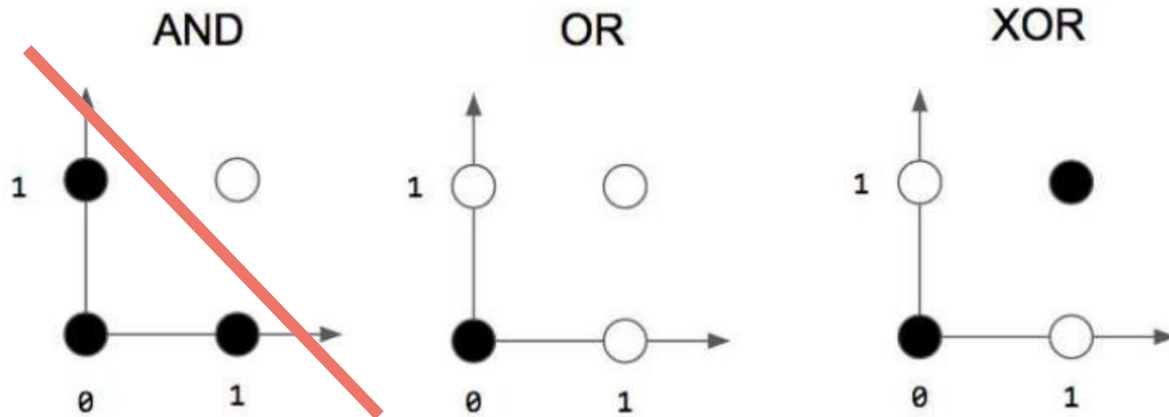
퍼셉트론을 이용하여 아래 문제를 해결해보자

Q. 선형구분선을 그어 검은점과 흰 점을 구분하자!



퍼셉트론의 한계

Q. 선형구분선을 그어 검은점과 흰 점을 구분하자!

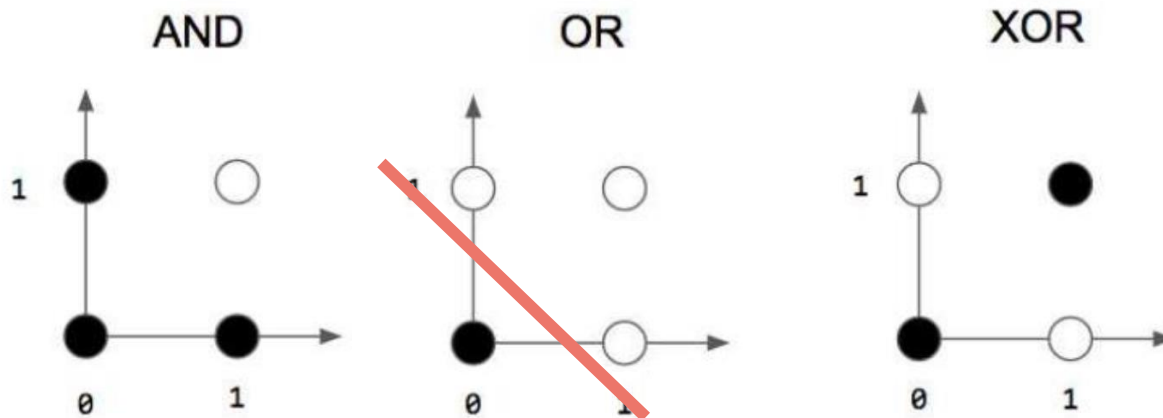


퍼셉트론의 매개변수에 임의의 값 대입 시

$ax_1 + bx_2 > c$ 로 직선의 방정식의 형태로 나옴

퍼셉트론의 한계

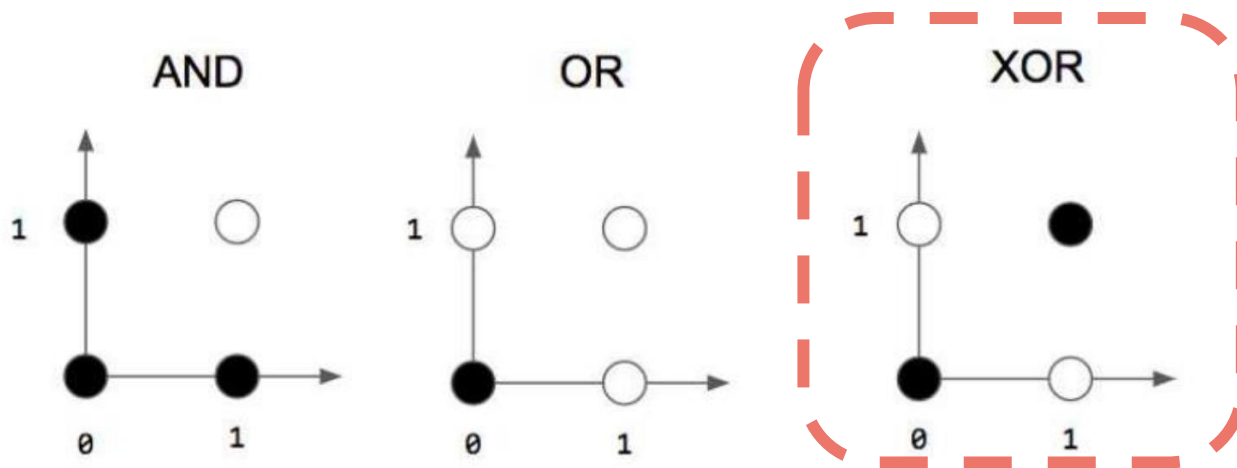
Q. 선형구분선을 그어 검은점과 흰 점을 구분하자!



즉 퍼셉트론은 **선형적인 문제**만을 해결할 수 있음

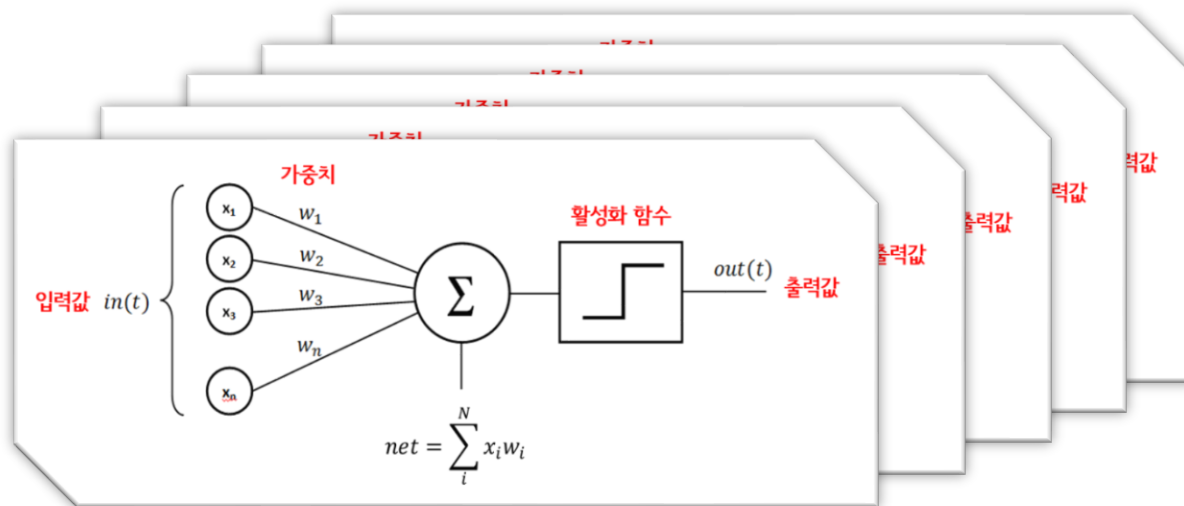
퍼셉트론의 한계

Q. 선형구분선을 그어 검은점과 흰 점을 구분하자!



퍼셉트론 하나로는 비선형적인 XOR 문제를 해결할 수 없음 !

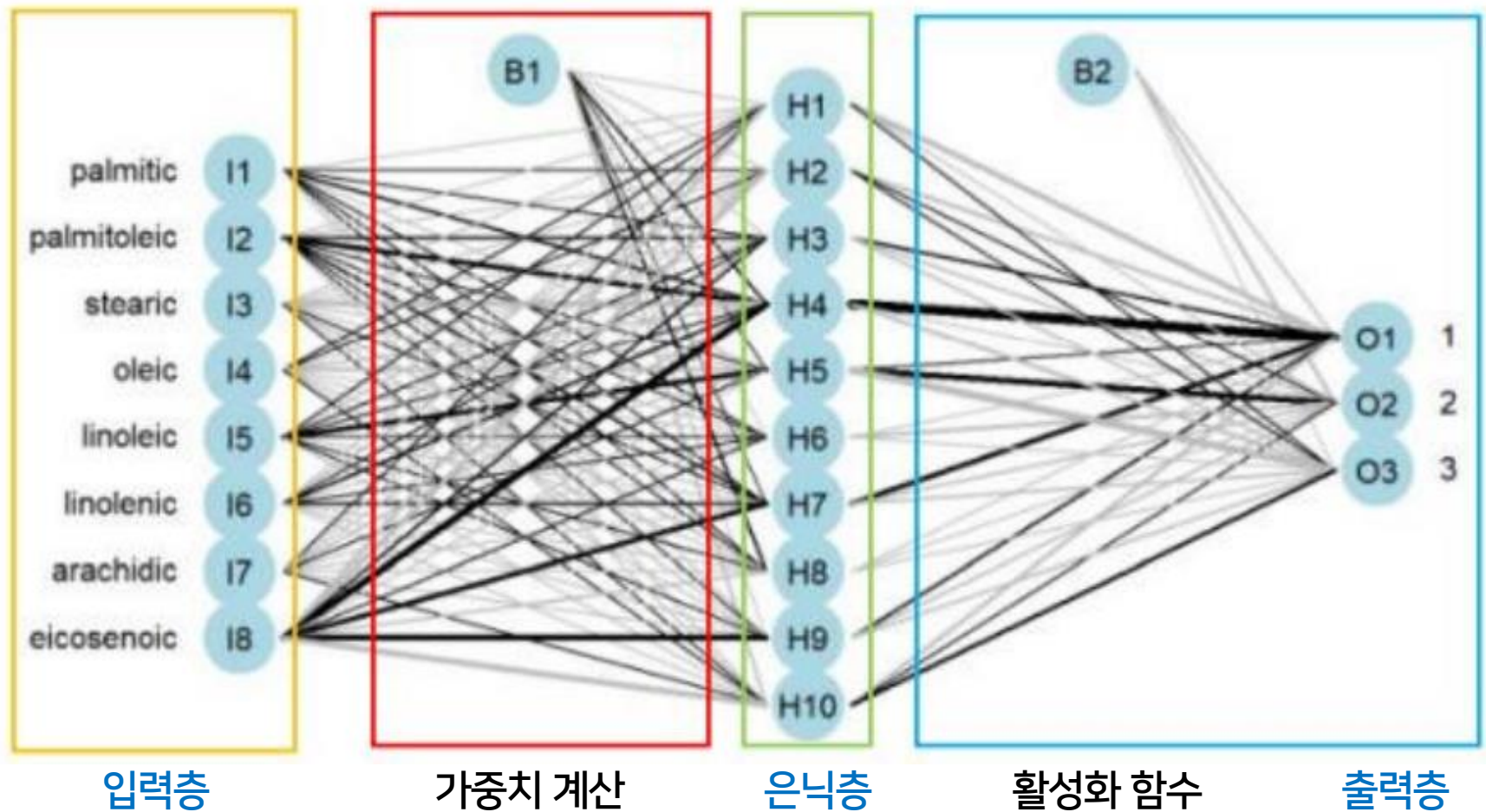
다층 퍼셉트론



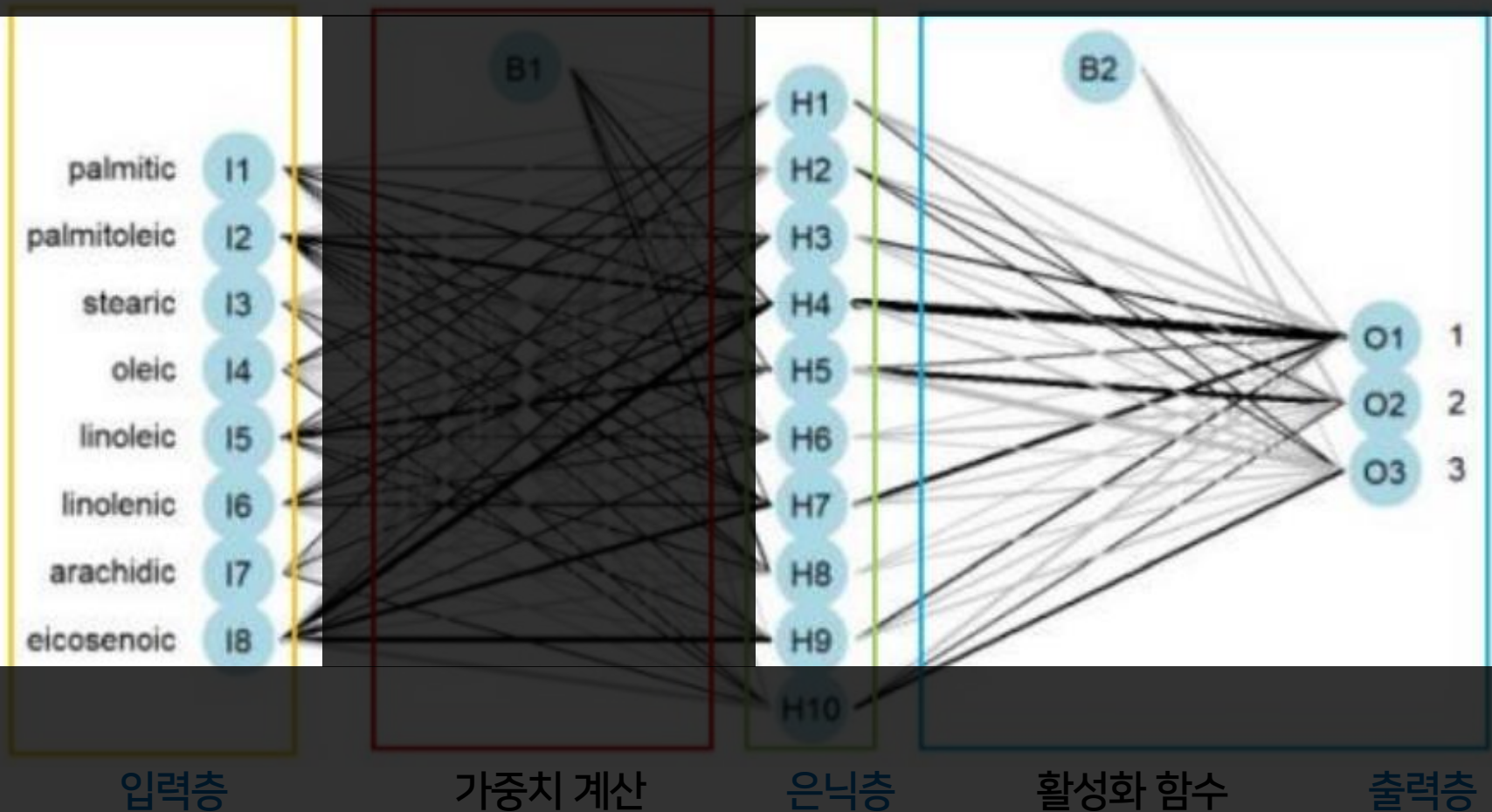
퍼셉트론 하나로는 XOR문제를 해결할 수 없었지만,
여러 퍼셉트론을 쌓아올리면 XOR 문제를 해결할 수 있음

다층 퍼셉트론

동작 방식

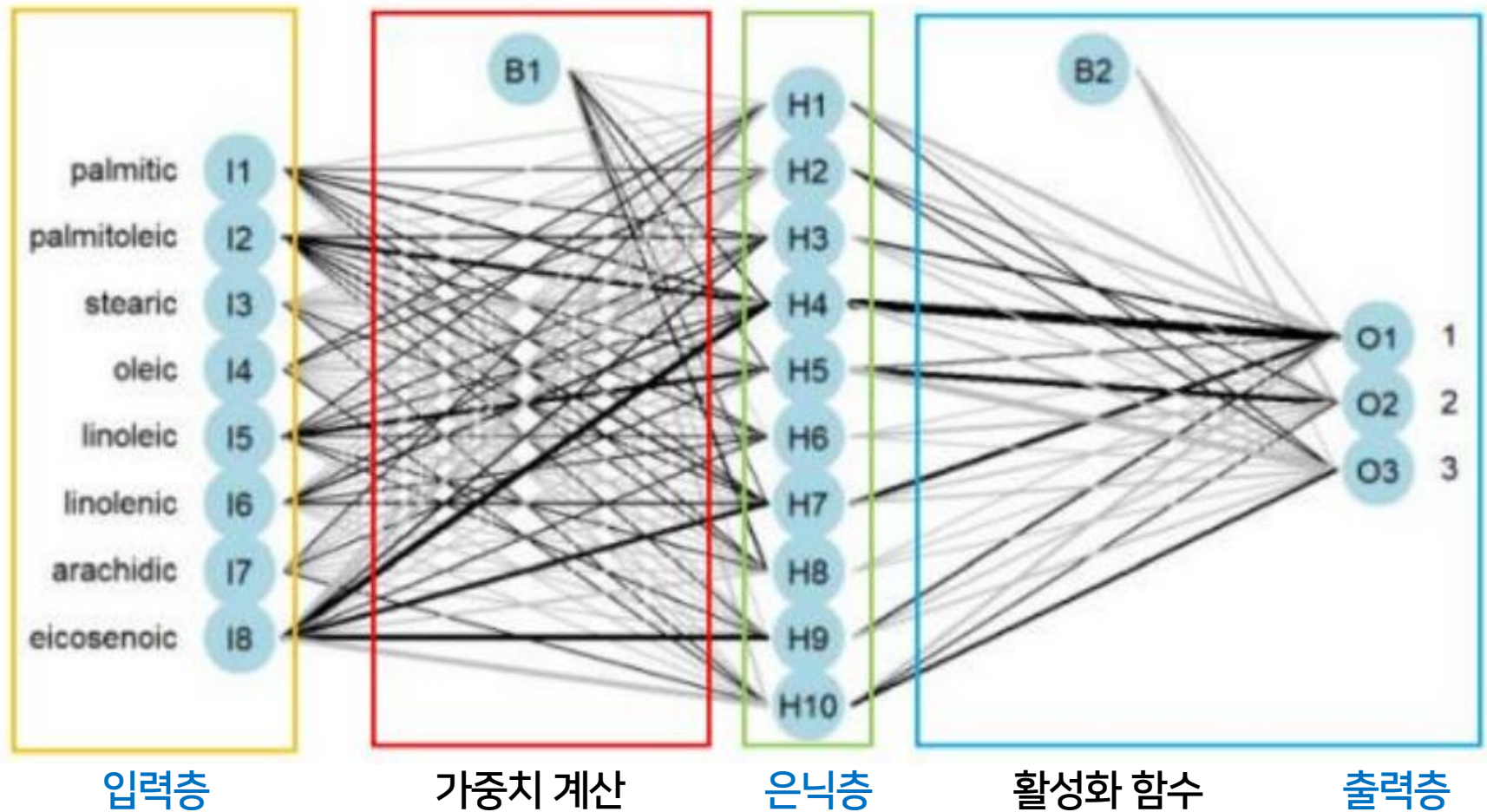


다층 퍼셉트론 노드(Node) 입력과 출력에 해당하는 파란 원 동작 방식



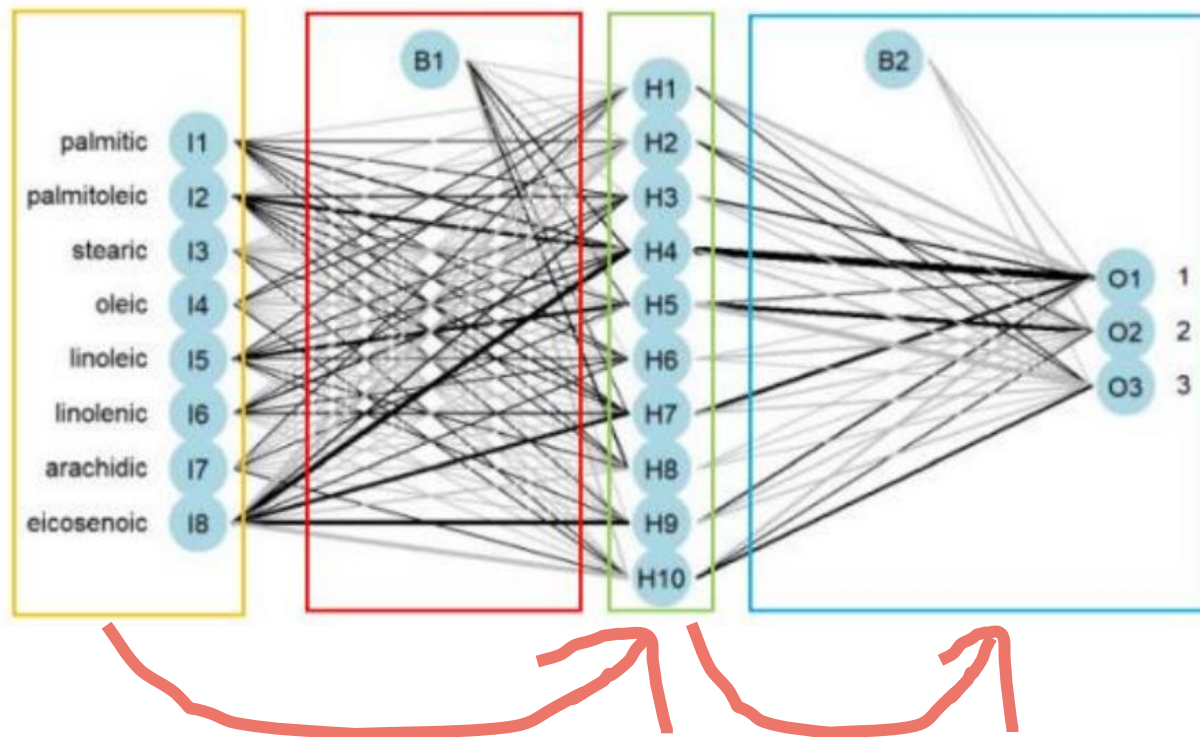
다층 퍼셉트론

동작 방식



다층 퍼셉트론

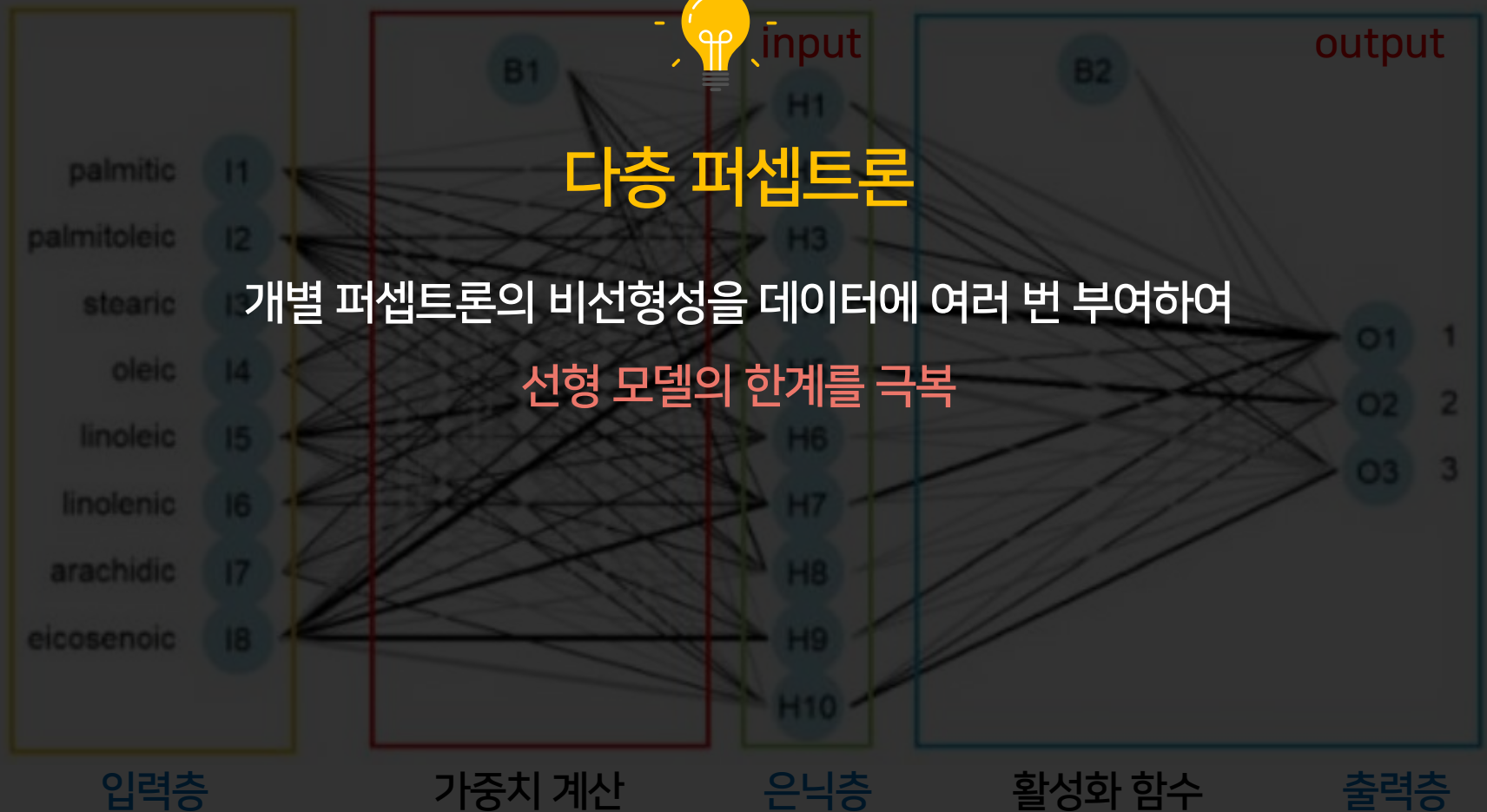
동작 방식



I에서 H로 한번, H에서 O로 한번 **총 두번에 걸쳐 퍼셉트론을 쌓음**

다층 퍼셉트론

다층 퍼셉트론의 동작 방식



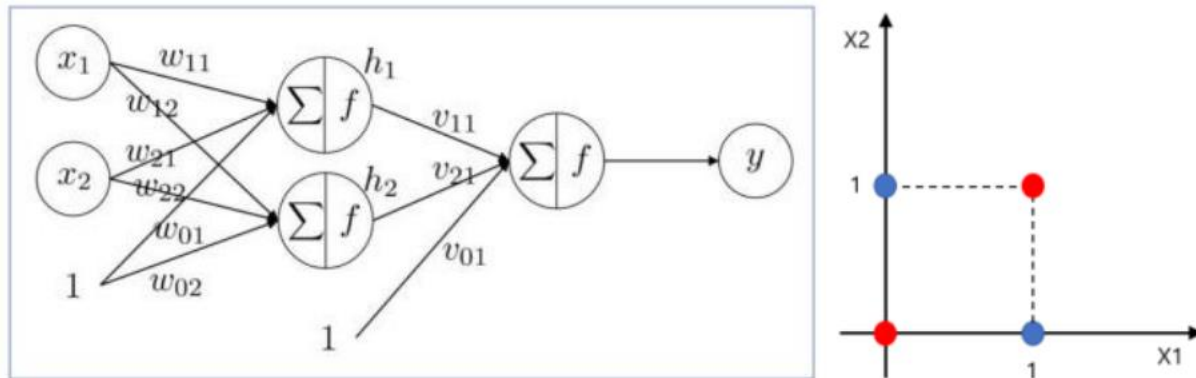
다층 퍼셉트론

개별 퍼셉트론의 비선형성을 데이터에 여러 번 부여하여

선형 모델의 한계를 극복

다층 퍼셉트론

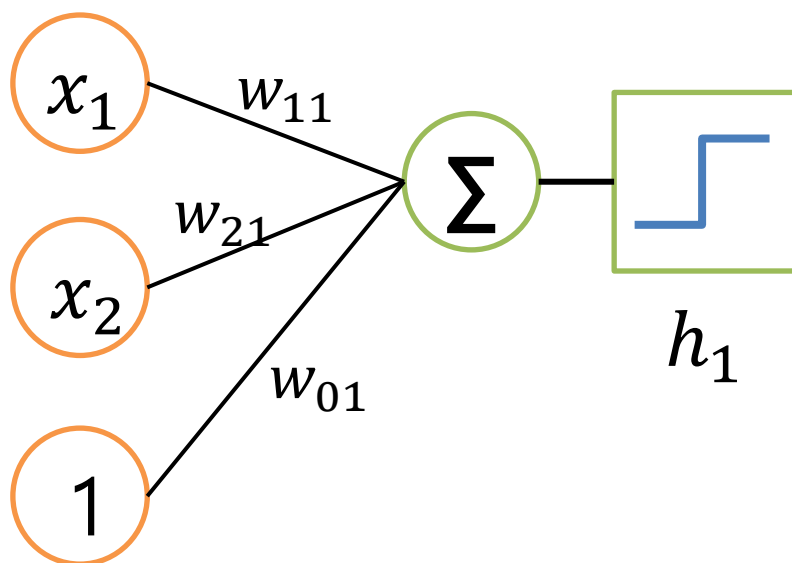
XOR 문제 해결



위 그림과 같은 다층 퍼셉트론은 입력층과 은닉층의 노드가 2개, 출력층의 노드가 1개이며, 활성화 함수가 계단함수인 이진 분류 모델

다층 퍼셉트론

XOR 문제 해결



$$w_{11} = 1.0, w_{21} = 1.0, w_{01} = -1.5$$

x_1	x_2	Σ	y_1
0	0	-1.5	0
0	1	-0.5	0
1	0	-0.5	0
1	1	0.5	1

오른쪽 표는 주어진 입력층에서 은닉층으로 연결된 2 개의 퍼셉트론 중 첫번째 퍼셉트론의 가중치를 차례대로 1, 1, -1.5 로 놓은 결과

다층 퍼셉트론

XOR 문제 해결

$$\begin{array}{c}
 x_1 \quad x_2 \quad \text{bias} \\
 \begin{array}{|c|c|c|}
 \hline
 0 & 0 & 1 \\
 \hline
 0 & 1 & 1 \\
 \hline
 1 & 0 & 1 \\
 \hline
 1 & 1 & 1 \\
 \hline
 \end{array} \\
 (4,3)
 \end{array}
 \times
 \begin{array}{c}
 \begin{array}{|c|}
 \hline
 1 \\
 \hline
 1 \\
 \hline
 -1.5 \\
 \hline
 \end{array} \\
 (3,1)
 \end{array}
 \begin{array}{c}
 w_{11} \\
 w_{21} \\
 w_{01}
 \end{array}
 =
 \begin{array}{c}
 \Sigma \\
 \begin{array}{|c|}
 \hline
 -1.5 \\
 \hline
 -0.5 \\
 \hline
 -0.5 \\
 \hline
 0.5 \\
 \hline
 \end{array} \\
 (4,1)
 \end{array}$$

다층 퍼셉트론

XOR 문제 해결

$$w_{11} = 1.0, w_{21} = 1.0, \\ w_{01} = -1.5$$

x_1	x_2	Σ	y_1
0	0	-1.5	0
0	1	-0.5	0
1	0	-0.5	0
1	1	0.5	1

$$w_{12} = 1.0, w_{22} = 1.0, \\ w_{02} = -0.5$$

x_1	x_2	Σ	y_2
0	0	-0.5	0
0	1	0.5	1
1	0	0.5	1
1	1	1.5	1

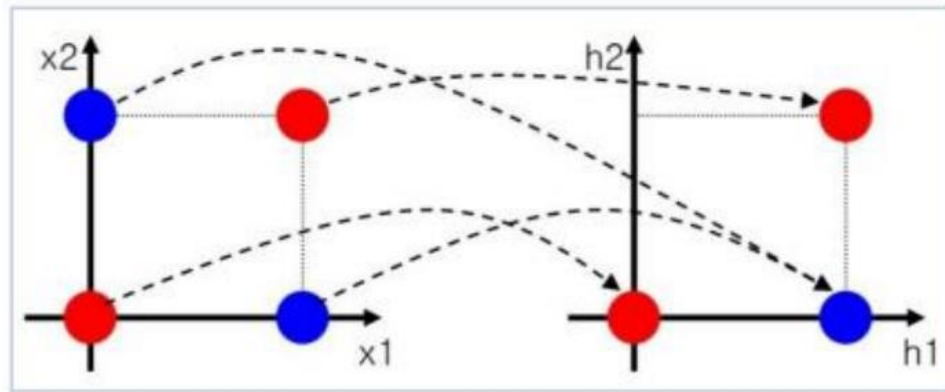
$$v_{11} = -1.0, v_{21} = 1.0, \\ v_{01} = -0.5$$

y_1	y_2	Σ	y
0	0	-0.5	0
0	1	0.5	1
0	1	0.5	1
1	1	-0.5	0

결과적으로 총 9개의 가중치의 값을 위 표에 주어진 것처럼 설정하면
 파란색 원의 좌표를 넣었을 때 1을, 빨간색 원의 좌표를 넣었을 때
 0을 정확하게 반환하고 있음

다층 퍼셉트론

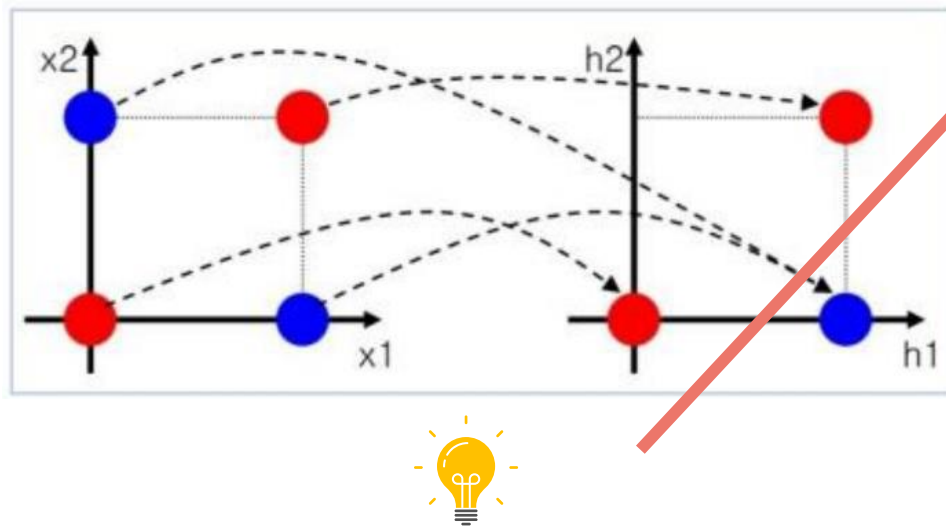
XOR 문제 해결



파란색 점에 해당하는 왼쪽 2 차원 평면에서의 (1, 0)과 (0, 1)은
오른쪽 2 차원 평면에서의 (0, 1)로 모두 옮겨짐
빨간색 점들 역시 h_1 축과 h_2 축에서 해당하는 위치로 옮겨짐

다층 퍼셉트론

XOR 문제 해결



왼쪽 2 차원 평면에서는 선형적인 구분선을 그을 수 없었던 반면,
오른쪽의 2 차원 평면에서는 **선형적인 구분선을 그을 수 있게 됨**

3

신경망

신경망

유사점

인공 신경망(Neural Network)은 다층 퍼셉트론과 유사한 개념

차이점

다층 퍼셉트론 : 순전파 vs 인공 신경망 : 순전파+역전파

순전파

유사점

인공 신경망(Neural Network)은 다층 퍼셉트론과 유사한 개념

차이점

다층 퍼셉트론 : 순전파

인공 신경망 : 순전파+역전파

순전파란?

입력에 가중치가 곱해지고 해당 값들이 모두 더해져 활성화 함수를 거쳐 출력되는 과정 전반

역전파란?

역방향으로 이루어지는 전파

순전파

유사점

인공 신경망(Neural Network)은 다층 퍼셉트론과 유사한 개념

차이점

다층 퍼셉트론 : 순전파

인공 신경망 : 순전파+역전파

순전파란?

입력에 가중치가 곱해지고 해당 값들이 모두 더해져 활성화 함수를 거쳐 출력되는 과정 전반

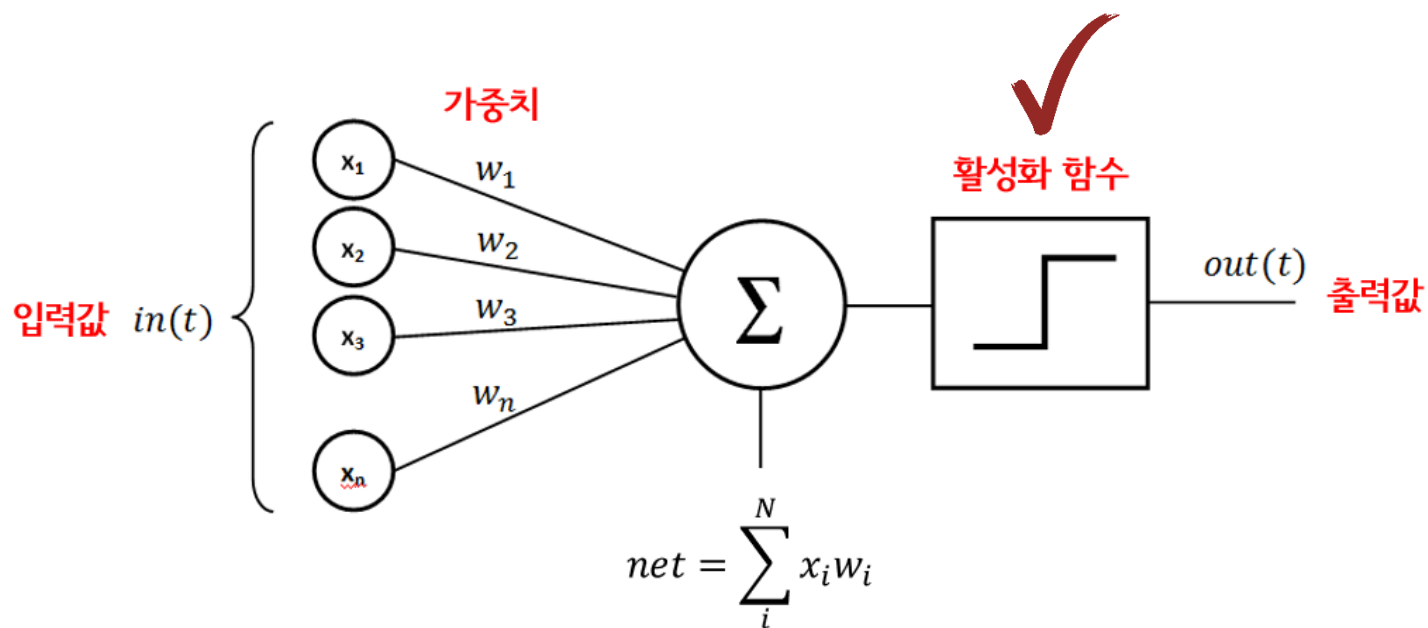
역전파란?

모델이 학습을 해 나가는 과정이며, 오차를 가중치에 반영하기 위해 역방향으로 이루어지는 전파

활성화 함수 Activation function

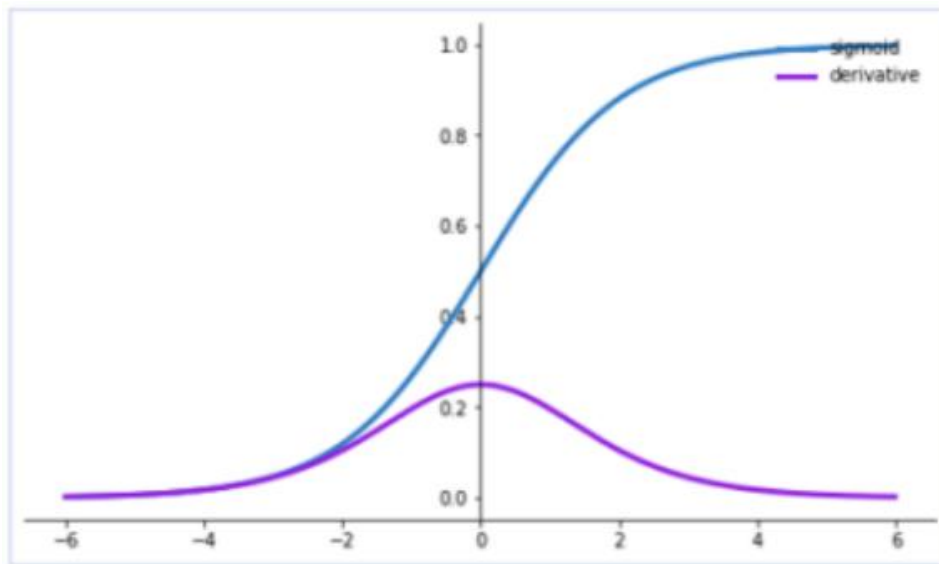
입력과 가중치의 선형결합에 **비선형성**을 부여

Ex) Sigmoid, tanh, ReLU, Leaky ReLU, PReLU, ELU, softmax



활성화 함수 Activation function

시그모이드 함수



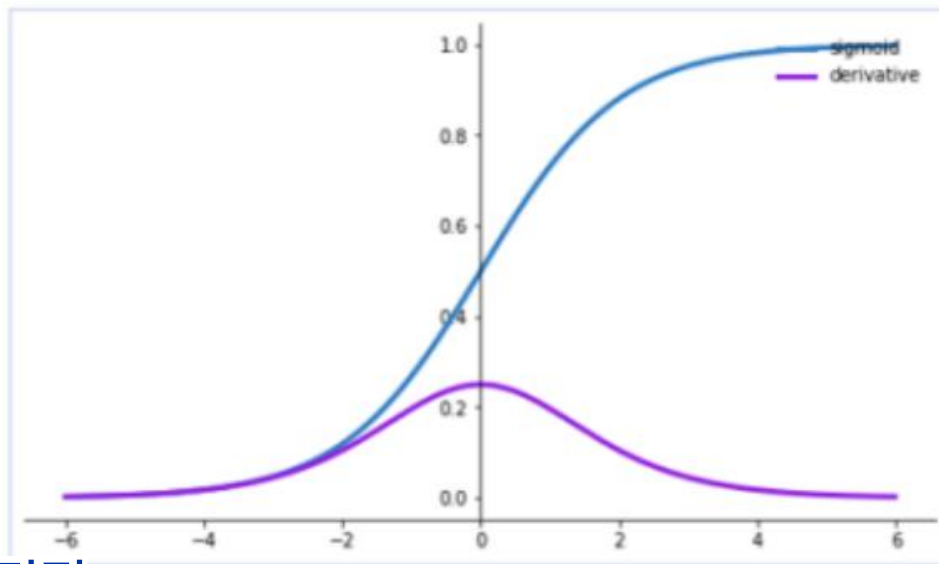
$$\sigma(x) = \frac{1}{1 + e^x}$$

특징

- 0과 1사이의 값을 가질 수 있음
- 이진분류에 사용
- 미분 가능 > 역전파 가능

활성화 함수 Activation function

시그모이드 함수



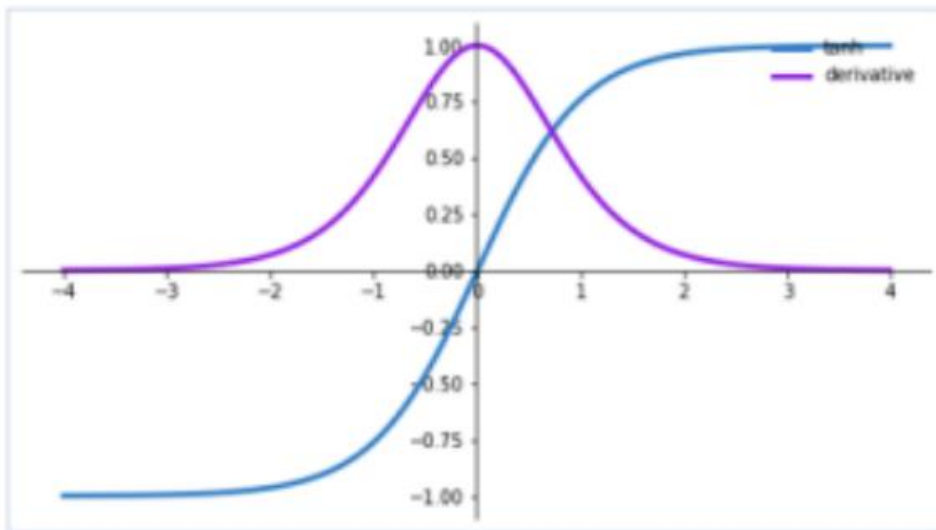
$$\sigma(x) = \frac{1}{1 + e^x}$$

단점

- 대부분의 값이 0 또는 1에 수렴, 중간 값이 나오는 경우는 적음
- 중양에서 조금만 벗어나도 미분값이 0으로 수렴
 - 지수함수로 인한 많은 연산량
- 중양값이 0.5 > 최적화가 어려움

활성화 함수 Activation function

하이퍼볼릭 탄젠트 함수



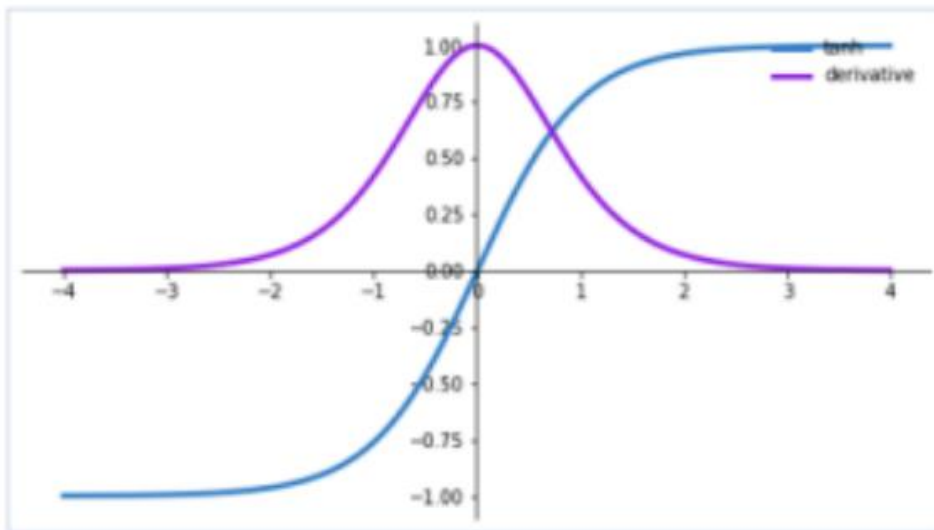
$$\tanh(x) = \frac{e^x - e^{-x}}{e^x + e^{-x}}$$

특징

- -1과 1사이의 값을 가질 수 있음
- 중앙값이 0 = 음수 출력 가능
- 출력 범위가 크기 때문에 효율적 학습 가능

활성화 함수 Activation function

하이퍼볼릭 탄젠트 함수



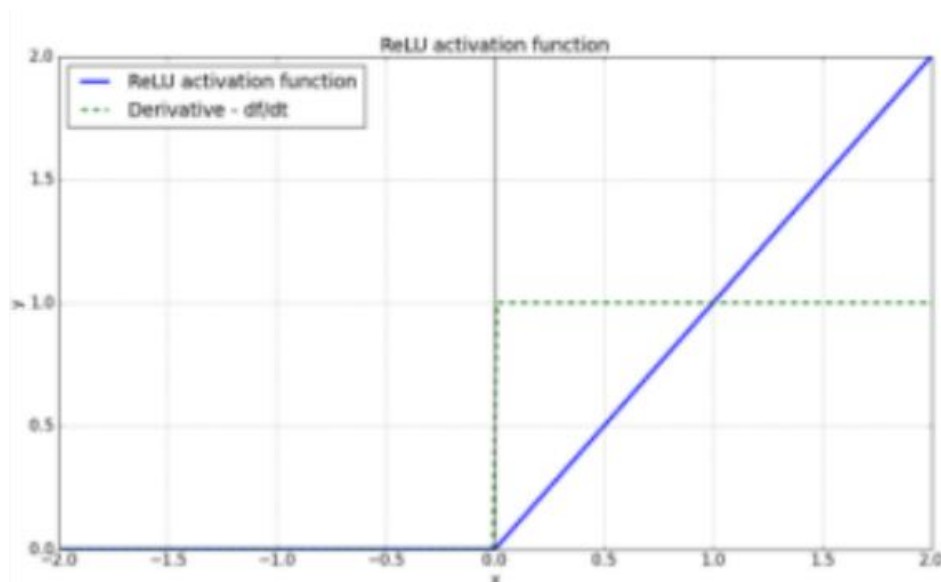
$$\tanh(x) = \frac{e^x - e^{-x}}{e^x + e^{-x}}$$

단점

- 중앙에서 조금만 벗어나도 미분값이 0으로 수렴
- 지수함수로 인한 많은 연산량

활성화 함수 Activation function

ReLU 함수



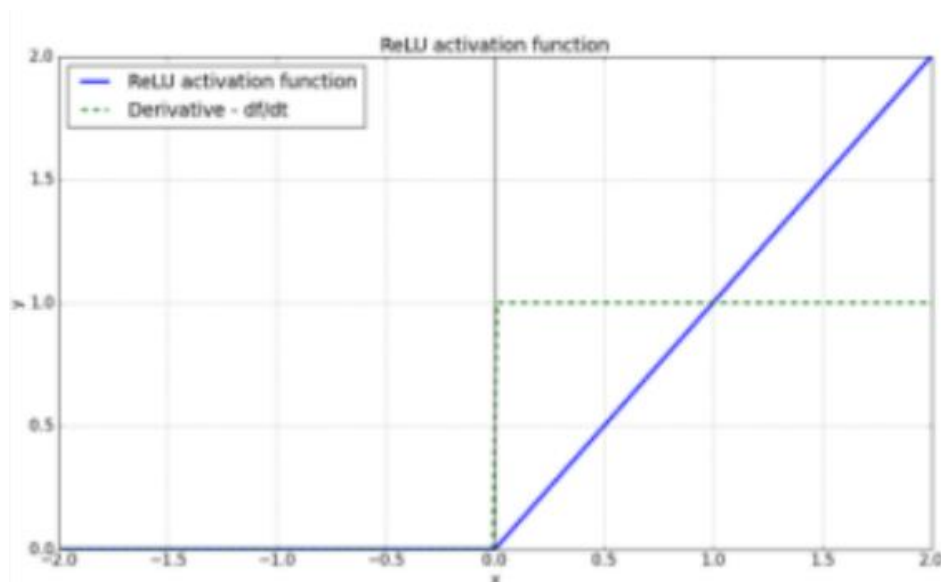
$$f(x) = \max(0, x)$$

특징

- 입력값이 음수면 0, 양수면 자기 자신을 반환
- 기울기 소실 문제 해결 (특정 수로 수렴하지 않기 때문)
- 계산이 쉬움 > 학습 속도 감소

활성화 함수 Activation function

ReLU 함수



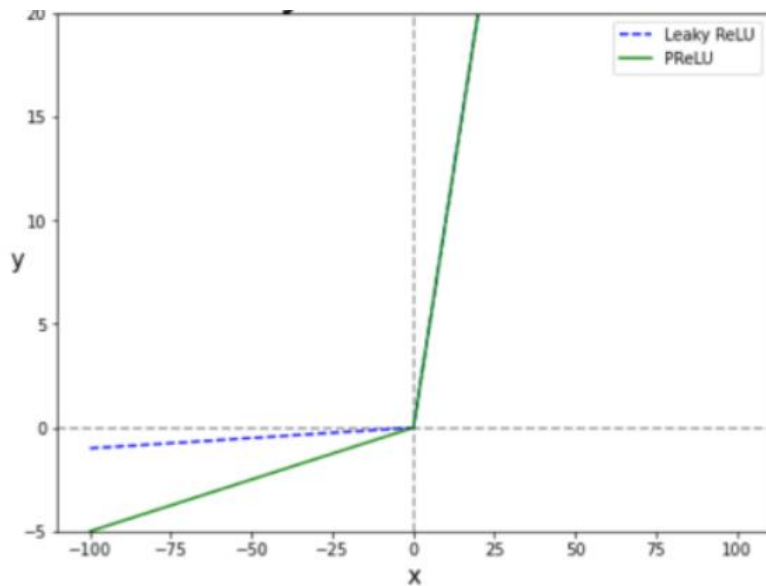
$$f(x) = \max(0, x)$$

단점

미분값이 0인 경우 학습이 이뤄지지 않음 (knock out 문제)

활성화 함수 Activation function

Leaky ReLU & PReLU



$$f(x) = \max(0.01x, x)$$
$$f(x) = \max(ax, x)$$

특징

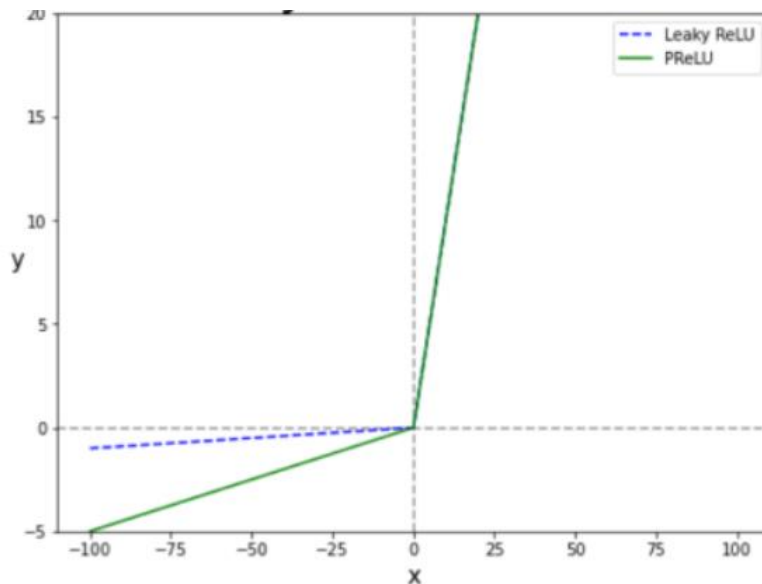
Leaky ReLU : 음수 x 0.01

PReLU : 음수를 하이퍼 파라미터 α 로 지정

- knock out 해결

활성화 함수 Activation function

Leaky ReLU & PReLU



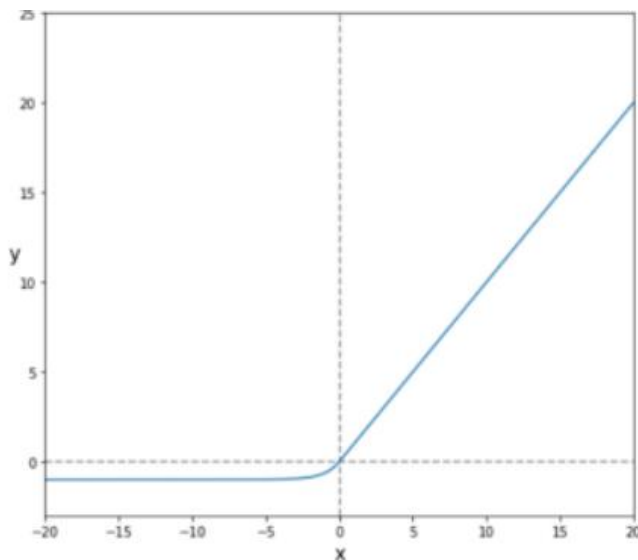
$$f(x) = \max(0.01x, x)$$
$$f(x) = \max(ax, x)$$

단점

음수에서 선형성 발생 > 복잡한 분류에서 사용하기 어려움

활성화 함수 Activation function

ELU 함수



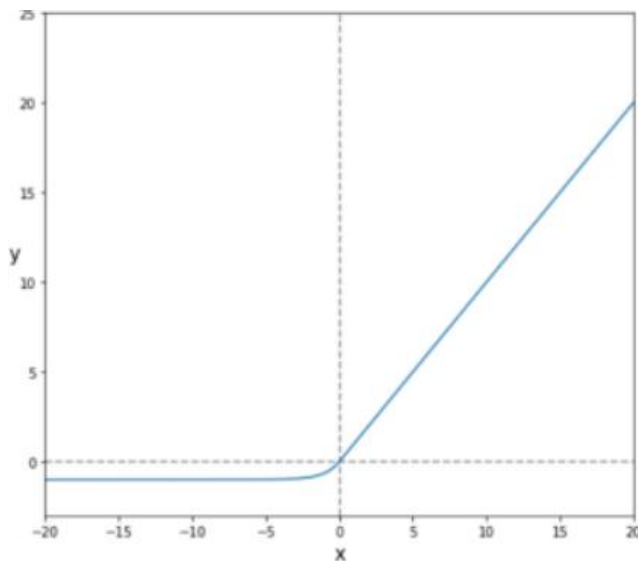
$$f(x) = \begin{cases} x & (x > 0) \\ a(e^x - 1) & (x \leq 0) \end{cases}$$

특징

- 음수에서 지수함수 사용
- ReLU 함수의 knock out 해결
- 음수에서도 비선형적이기 때문에 복잡한 분류에도 사용 가능

활성화 함수 Activation function

ELU 함수



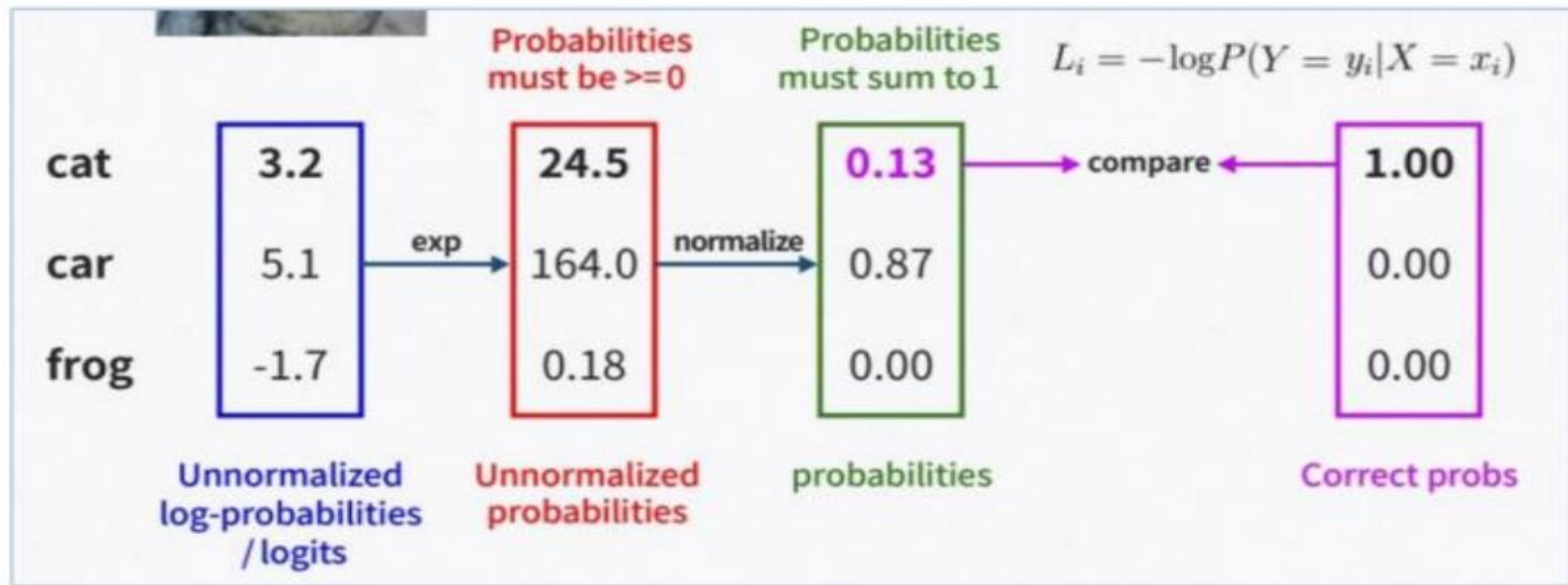
$$f(x) = \begin{cases} x & (x > 0) \\ a(e^x - 1) & (x \leq 0) \end{cases}$$

단점

- 크게 성능이 향상되는 건 아님
- 지수함수로 인한 연산량 증가

활성화 함수 Activation function

Softmax 함수



-출력층에서 활용, 다중분류문제에 사용

-출력 값을 0에서 1사이로 변환

-지수함수 통과를 통해 양수로 변환, 정규화를 통해 [0,1]로 변환

손실 함수 Loss function

Loss function

모델의 출력과 정답 레이블 사이의 차이를 계산하는 역할
Loss를 모델에 학습시켜 가중치를 조정해가며 가장 데이터를 잘 fit하는
모델 파라미터(=가중치)를 결정하게 함

평균 제곱 오차

Mean Squared Error

교차 엔트로피 오차

Cross Entropy Loss

손실 함수

평균 제곱 오차 Mean Squared Error

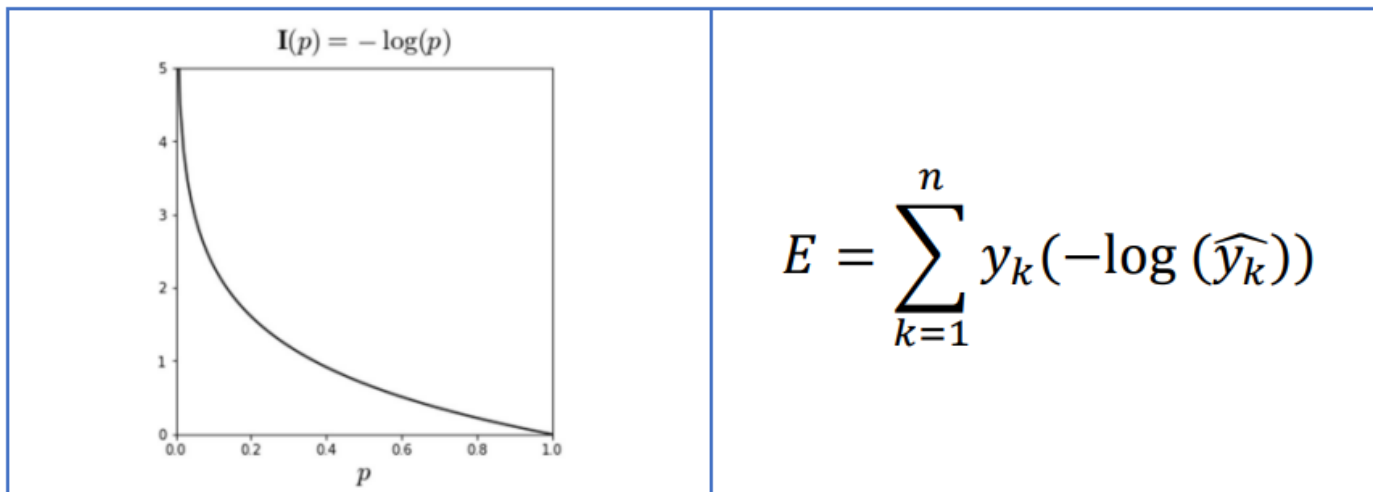
$$E = \frac{1}{2} \sum_{k=1}^n (\hat{y}_k - y_k)^2$$

-회귀문제에서 주로 쓰이는 손실 함수

-1/2는 역전파를 할 때 미분을 깔끔하게 하기 위해

손실 함수

교차 엔트로피 오차 Cross Entropy Loss



-분류 문제에서 사용되는 손실 함수

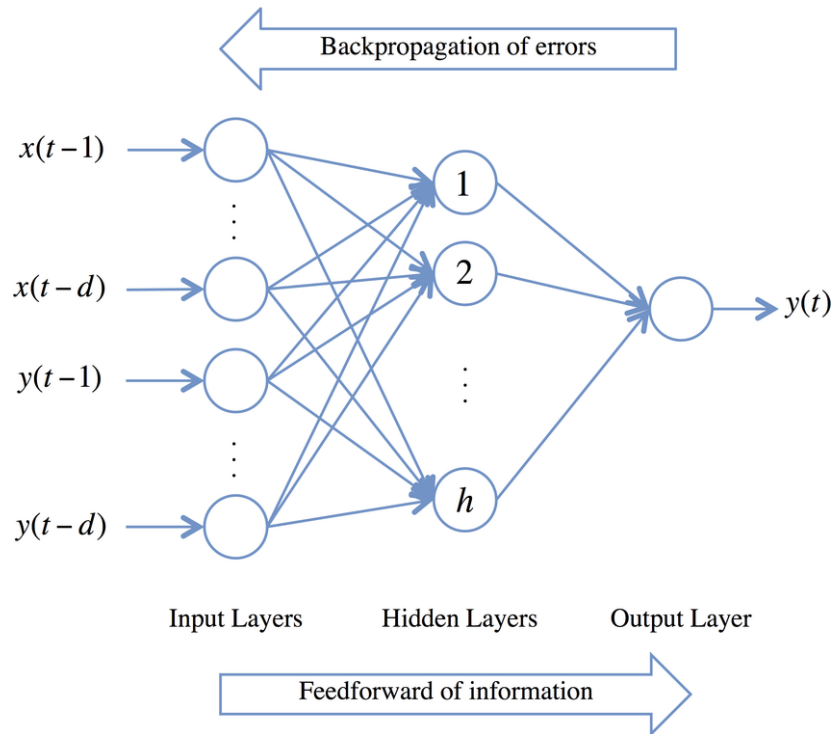
-Log함수에서 0에 가까울수록 큰 값, 1에 가까울 수록 0에 수렴하는 값 반환

-정답과 모델의 예측 결과의 차이가 클 수록 오차가 커짐

역전파 Backpropagation

Backpropagation

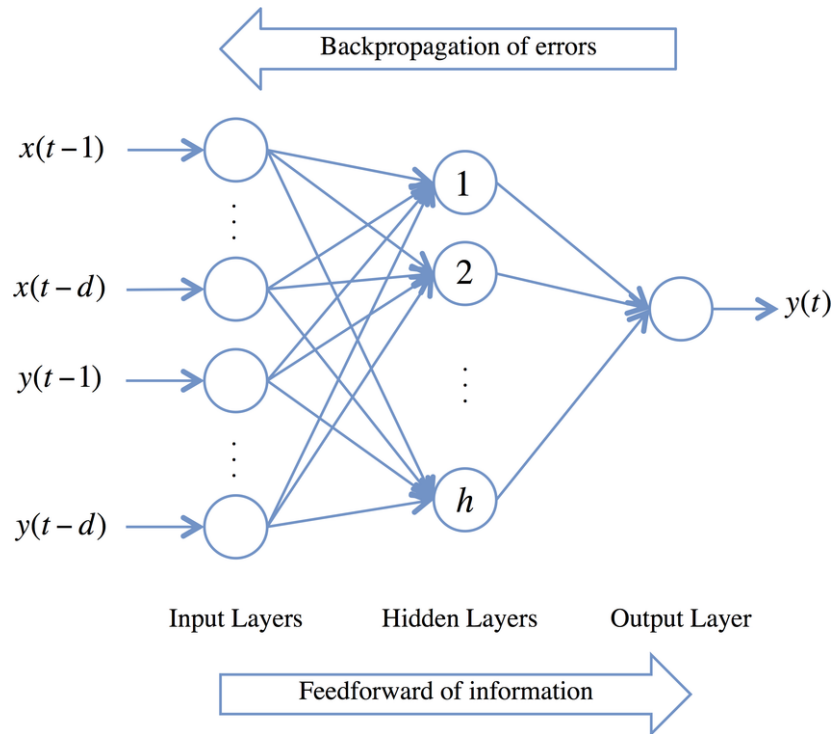
모델이 학습을 해 나가는 과정



역전파 Backpropagation

Backpropagation

모델이 학습을 해 나가는 과정

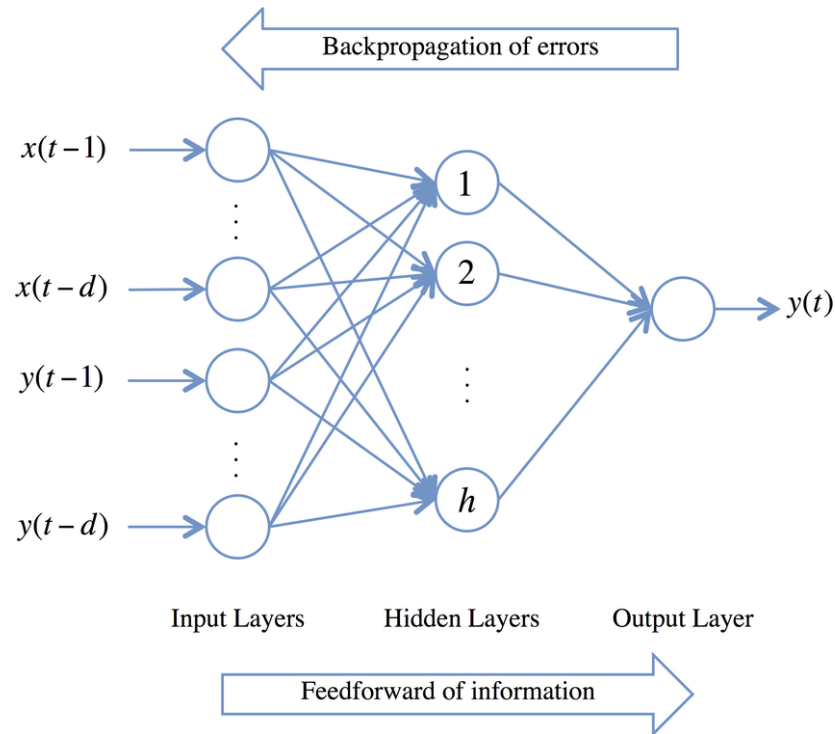


순전파를 통해
정답과 예측값의
차이를 계산

역전파 Backpropagation

Backpropagation

모델이 학습을 해 나가는 과정

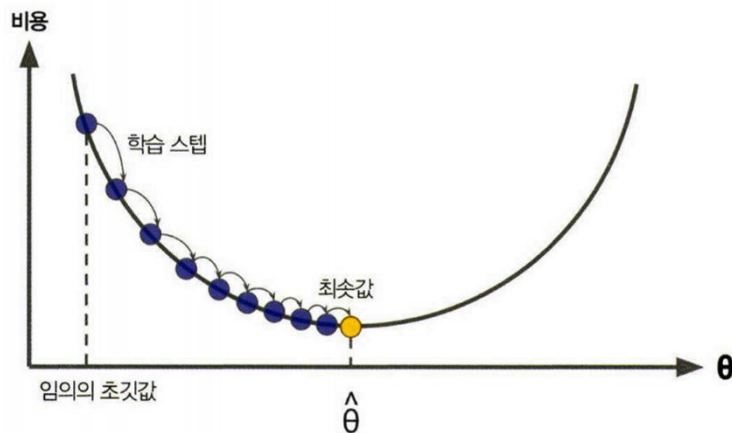


역전파를 통해
가중치와 편향을
업데이트

순전파를 통해
정답과 예측값의
차이를 계산

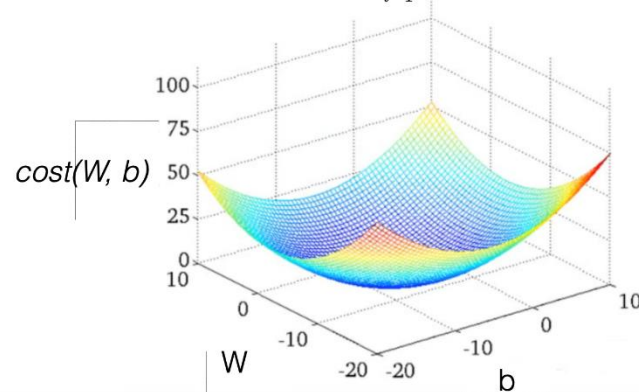
Optimizer

경사 하강법 (Gradient Descent)



Convex function

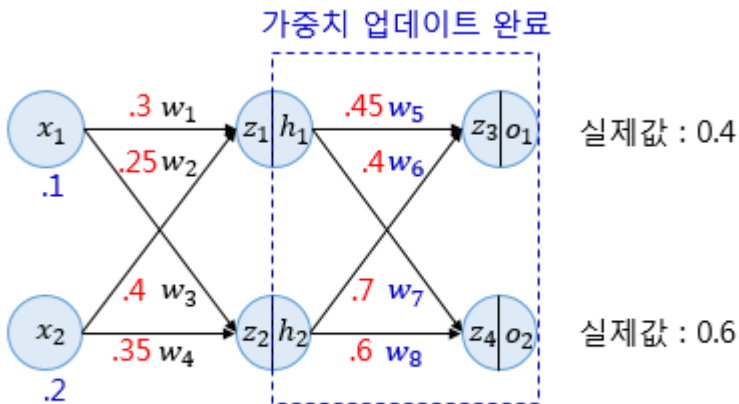
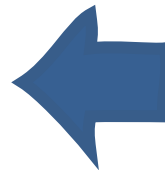
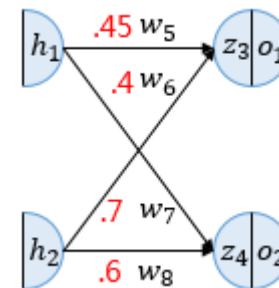
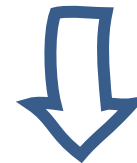
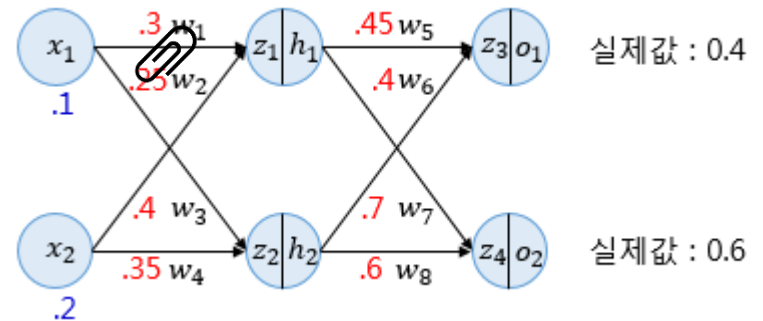
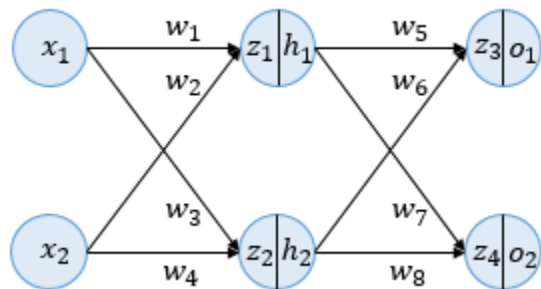
$$cost(W, b) = \frac{1}{m} \sum_{i=1}^m (H(x^{(i)}) - y^{(i)})^2$$



벡터의 편미분 값, 즉 기울기를 작게 만드는 방향으로
나아가는 형태의 최적화 방법

경사하강법 (Gradient Descent)

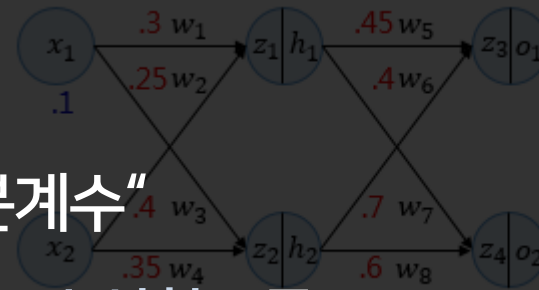
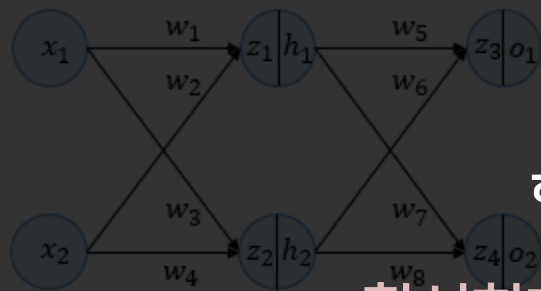
학습순서: 순전파 ~ 역전파



역전파(Back Propagation)

경사하강법 (Gradient Descent)

순전파 ~ 역전파



실제값 : 0.4

실제값 : 0.6

핵심은 “미분계수”

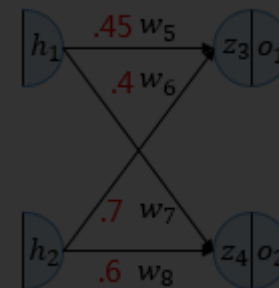
활성화함수, 가중합, 손실함수를

Chain Rule을 이용하여 미분



실제값 : 0.4

실제값 : 0.6

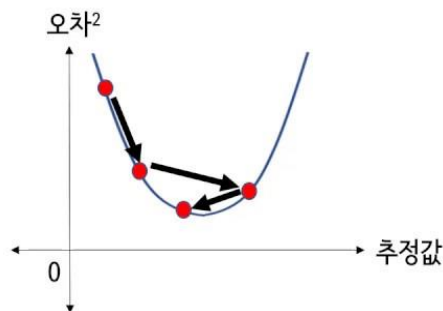


역전파(Back Propagation)

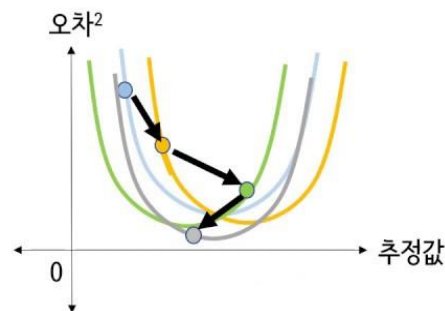
Optimizer

확률적 경사하강법 Stochastic Gradient Descent

경사하강법



확률적 경사하강법



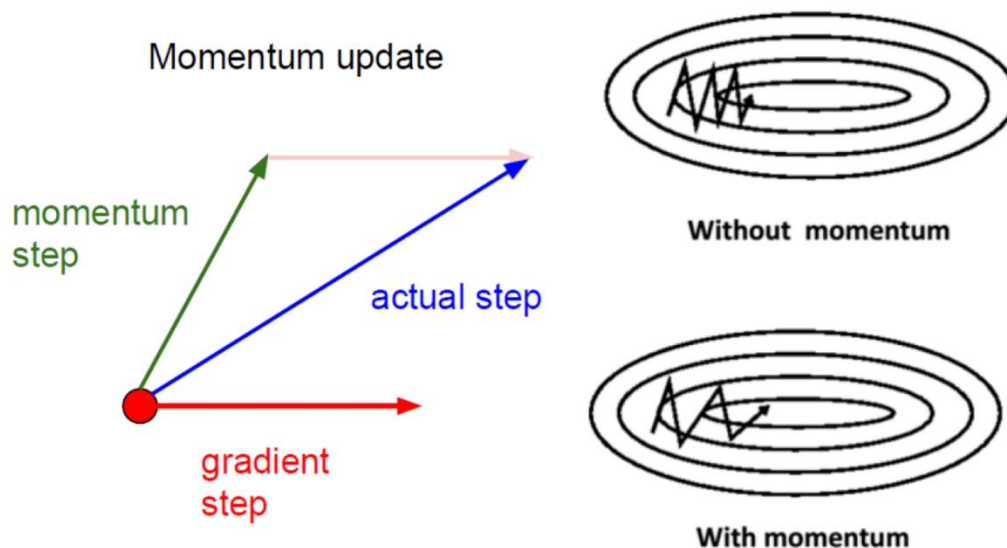
모든데이터를 사용하지 않고, 일정 수의 데이터만을 활용

-모든 데이터를 사용할 때보다 변동이 심하다는 단점

But, 효율성과 속도의 측면에서 경사하강법 보다 훨씬 뛰어남

Optimizer

모멘텀 (Momentum)

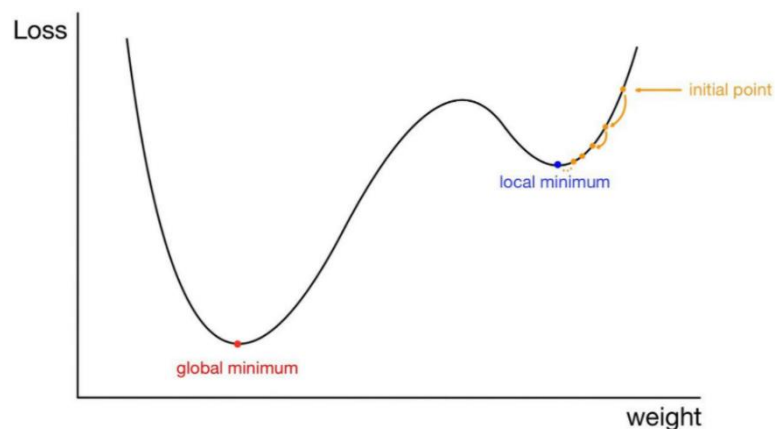


 Momentum은 "관성" 을 의미

최적화를 할 때, 미분 값이 클 경우 일종의 가속도를 부여하여
더 큰 보폭으로 움직임 > Local minima 문제 해결에 도움이 됨

Optimizer에서 만나는 여러가지 문제들

Local Minima



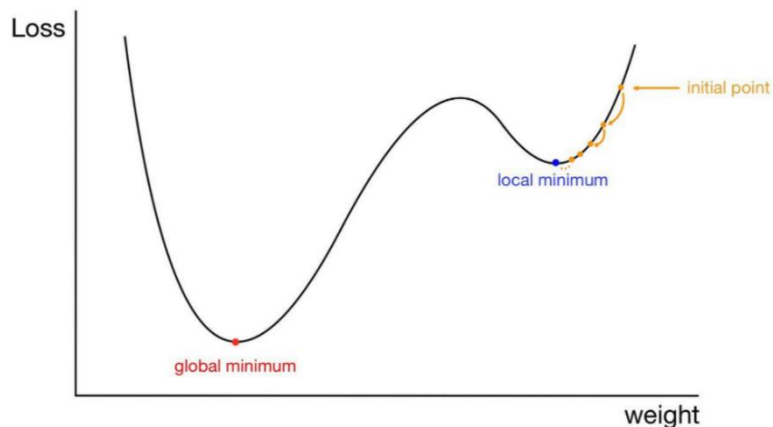
Loss 함수가 **최소**가 아닌
극소에서 학습이 종료



실제로 고차원 공간에서는
발생하기 힘들

Optimizer에서 만나는 여러가지 문제들

Local Minima



Loss 함수가 **최소**가 아닌
극소에서 학습이 종료

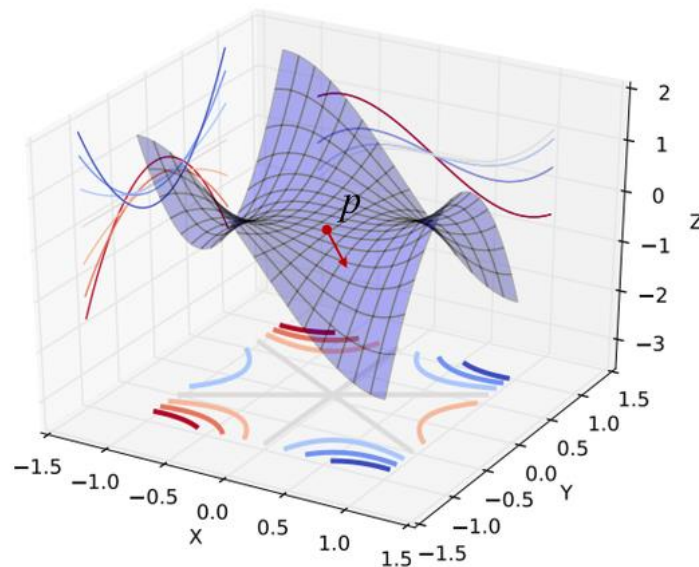


실제로 고차원 공간에서는
발생하기 힘들

 고차원에서는 워낙 가중치들이 많기 때문

Optimizer에서 만나는 여러가지 문제들

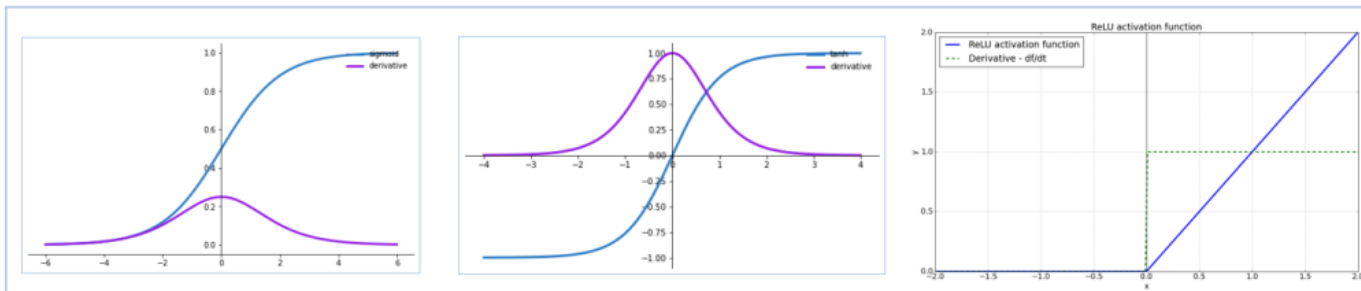
Saddle Point



고차원 함수에서 나타날 수 있는 문제로, 특정 점이 어떤 방향에서는
최소값이지만 어떤 방향에서는 **최대값**이 되는 문제

Optimizer에서 만나는 여러가지 문제들

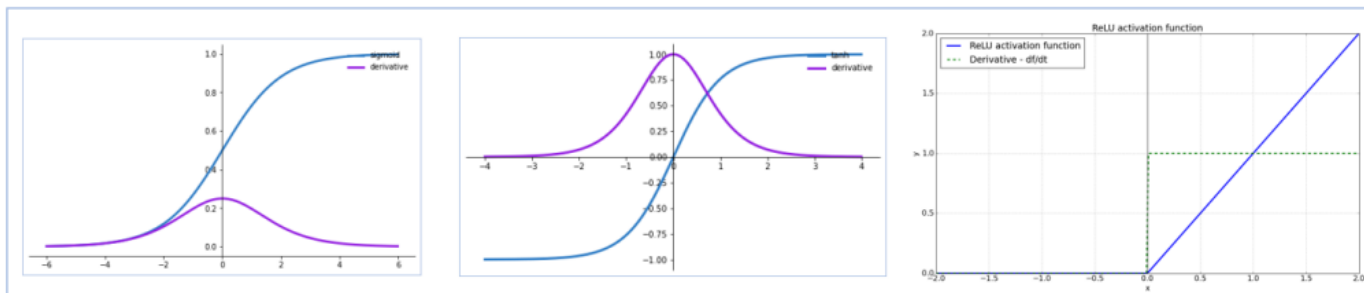
기울기 소실 문제 (Gradient Vanishing Problem)



역전파 과정에서
지속적으로
활성화함수의
도함수가 곱해짐

Optimizer에서 만나는 여러가지 문제들

기울기 소실 문제 (Gradient Vanishing Problem)



역전파 과정에서
지속적으로
활성화함수의
도함수가 곱해짐

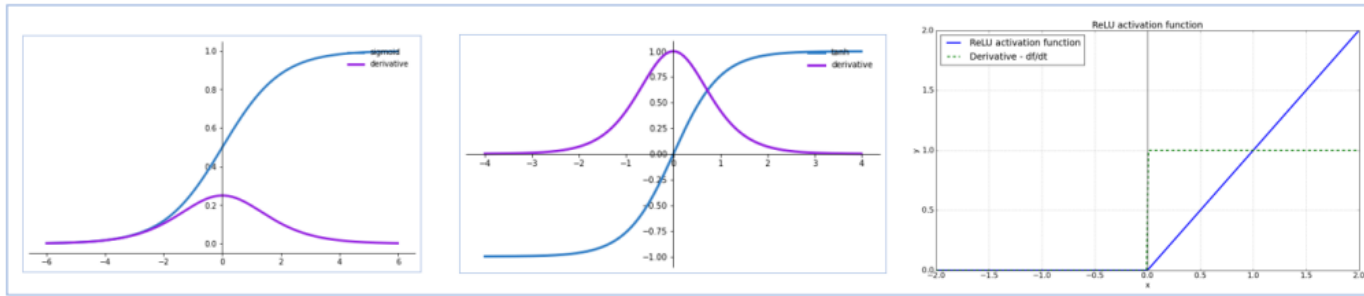


문제 발생



Optimizer에서 만나는 여러가지 문제들

기울기 소실 문제 (Gradient Vanishing Problem)



역전파 과정에서
지속적으로
활성화함수의
도함수가 곱해짐

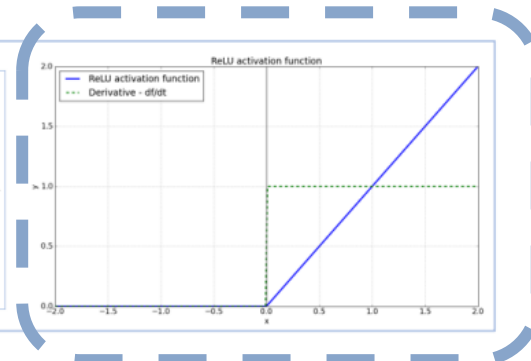
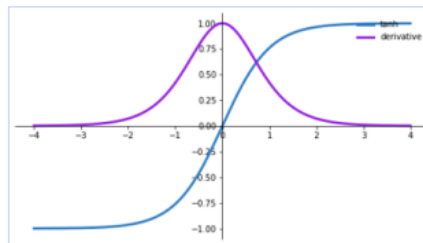
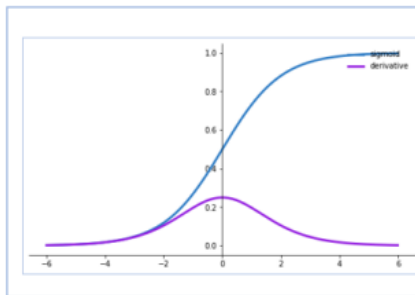
문제 발생



그 결과 기울기가
점점 0에 수렴

Optimizer에서 만나는 여러가지 문제들

기울기 소실 문제 (Gradient Vanishing Problem)



역전파 과정에서
지속적으로
활성화함수의
도함수가 곱해짐

문제 발생



그 결과 기울기가
점점 0에 수렴

해결책 제시



도함수의 그래프가
계단함수 형태인
ReLU의 도입



4

성능 향상 기법

가중치 초기화 Weight Initialization

신경망이 파라미터를 최적화 하는 과정에서, 동일하게 경사 하강법을 통해 내려간다고 하더라도 **가중치에 따라 도달하는 최저점은 다름**



"학습 시작 시점의 가중치를 정해주는 것 " 이 매우 중요

가중치 초기화 Weight Initialization

신경망이 파라미터를 최적화 하는 과정에서, 동일하게 경사 하강법을 통해 내려간다고 하더라도 **가중치에 따라 도달하는 최저점은 다름**



"학습 시작 시점의 가중치를 정해주는 것 " 이 매우 중요

가중치 초기화 Weight Initialization

신경망이 파라미터를 최적화 하는 과정에서, 동일하게 경사 하강법을 통해 내려간다고 하더라도 **가중치에 따라 도달하는 최저점은 다름**



"학습 시작 시점의 가중치를 정해주는 것 " 이 매우 중요



가중치 초기화 (Weight Initialization)

가중치 초기화 Weight Initialization

Zero Initialization

파라미터의 값을 모두 0으로 설정

파라미터의 값이 모두 같다면
역전파의 갱신이 무의미

>딥러닝에서 잘 사용하지 않음

Random Initialization

파라미터 값을 정규분포를 따르는
랜덤으로 설정

역전파 과정에서 출력값이
점점 치우침

Gradient Vanishing 문제가 발생

가중치 초기화 Weight Initialization

Zero Initialization

파라미터의 값을 모두 0으로 설정

파라미터의 값이 모두 같다면
역전파의 갱신이 무의미

>딥러닝에서 잘 사용하지 않음

Random Initialization

파라미터 값을 정규분포를 따르는
랜덤으로 설정

역전파 과정에서 출력값이
점점 치우침

Gradient Vanishing 문제가 발생

가중치 초기화 Weight Initialization

Xavier Initialization, He Initialization

Xavier Initialization

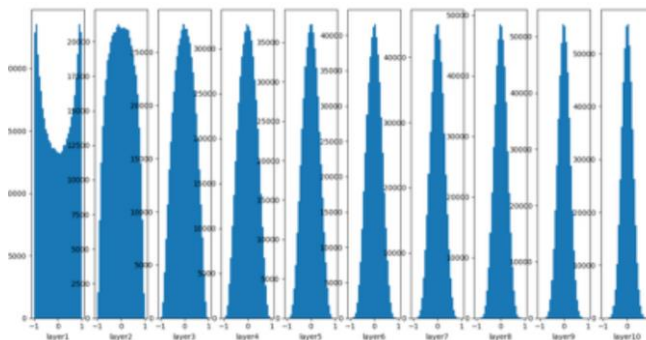
-이전 은닉층의 개수가 n 개이고 현재 은닉층의 개수가 m 개일 때,

$w_i \sim N(0, \frac{2}{\sqrt{n+m}})$ 로 가중치를 초기화하는 방법

-시그모이드 함수일때 주로 사용

-층마다 노드 개수를 다르게 설정하더라도 이에 맞게 가중치가 초기화 됨

>고정된 표준편차를 사용하는 것보다 더 강건(robust)



위 그래프는 활성화 함수로 sigmoid,
가중치 초기화로 Xavier Initialization을 사용했을 때의 그래프

가중치 초기화 Weight Initialization

Xavier Initialization, He Initialization

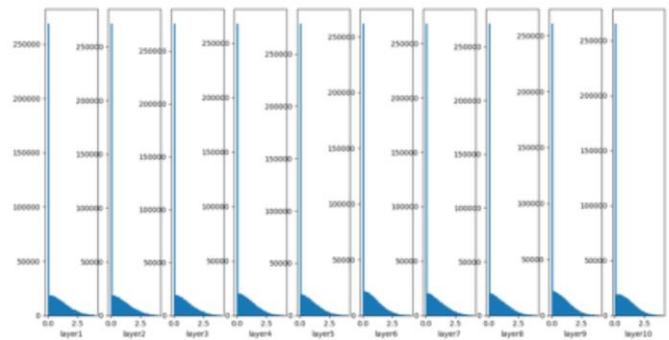
He Initialization

$-w_i \sim N(0, \sqrt{\frac{2}{n}})$ 로 가중치를 초기화하는 방법

-활성화함수가 ReLU 함수일 때, Xavier 정규화의 경우

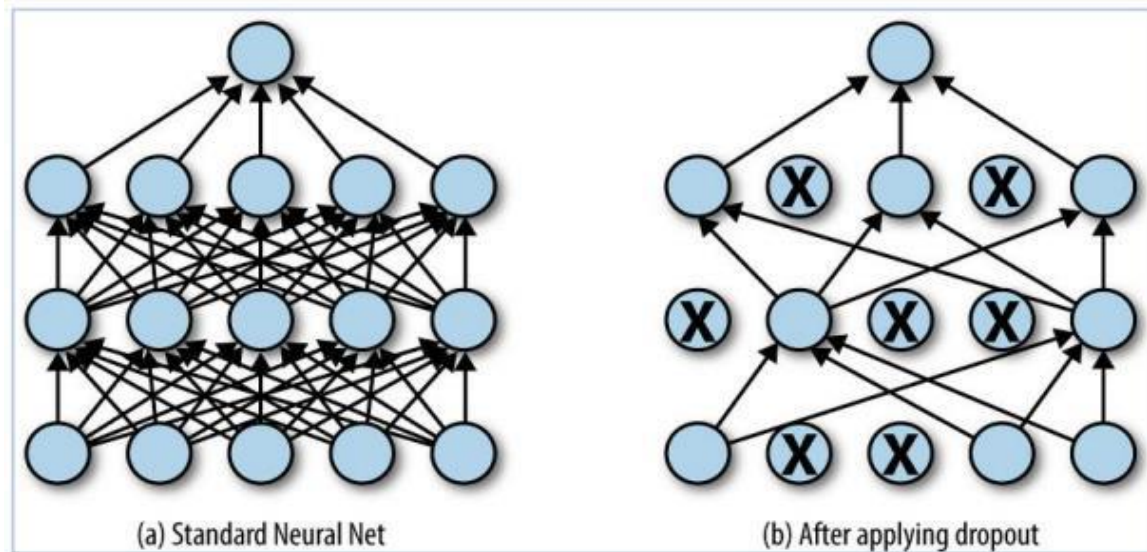
층이 깊어질수록 활성화값의 분포가 치우치기 때문에

> He 초기화를 사용



위 그래프는 활성화 함수로 ReLU,
가중치 초기화로 He Initialization을 사용했을 때의 그래프

Drop out

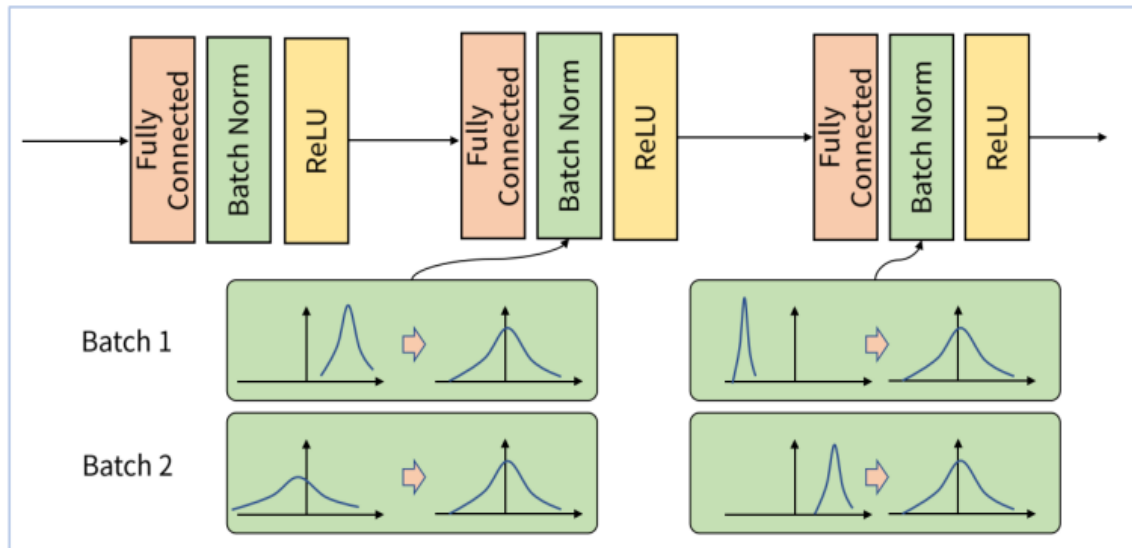


💡 과적합을 방지할 수 있는 방법 중 하나!

- Batch마다 학습할 때 일정한 비율의 노드(Node)를 버리고 학습하는 방법
- 여러 모델을 앙상블한 것과 유사한 효과를 낼 수 있음

Batch Normalization

과정

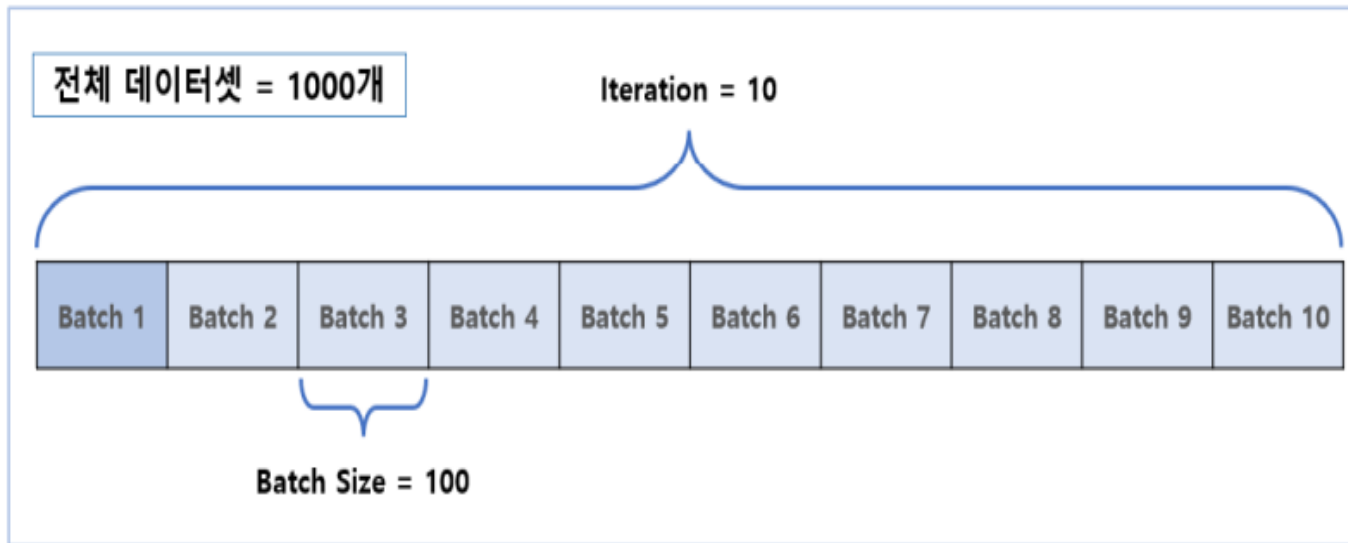


Batch Normalization

Batch마다 다른 분포로 인해 가중치 학습이 batch에 영향을 받는 것을 방지하기 위해 batch별 데이터를 평균과 분산으로 정규화해주는 것

Batch Normalization

Batch



 "Batch"란? 전체 데이터셋을 여러 개로 분할한 것

Batch의 크기에 따라 Iteration이 결정

위 그림의 경우 batch size가 100이므로 iteration의 수는 10

Layer Normalization

!

RNN처럼 sequential 데이터를 처리하는 경우에는
Batch normalization을 적용시키기 어려움

Layer Normalization

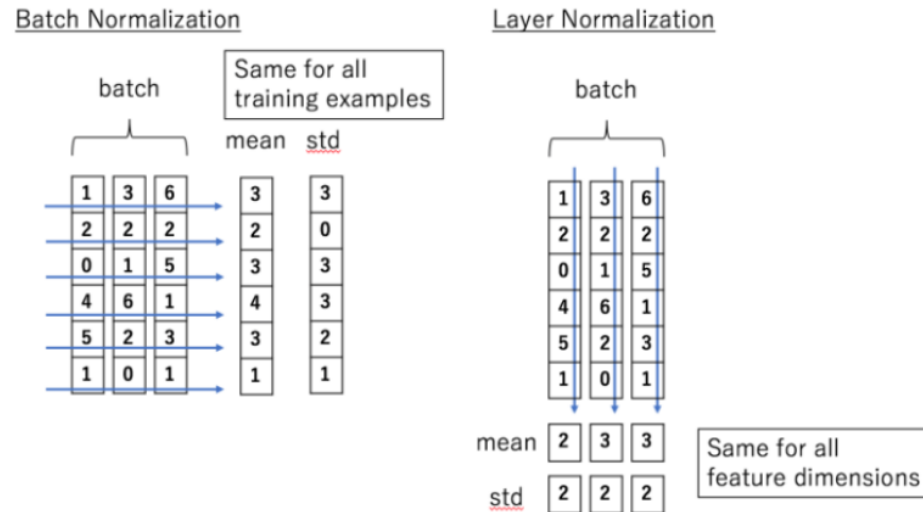


RNN처럼 sequential 데이터를 처리하는 경우에는
Batch normalization을 적용시키기 어려움



Feature차원에서 정규화를 진행하는
Layer Normalization을 진행

Layer Normalization



Sequence에 따라 파라미터 크기가 달라질 수 있는 Batch Normalization과 다르게

Layer Normalization은 Sequence에 강건

5

마무리, 다음주차 예고

1주차 Check point

- 지도학습, 비지도학습과 강화학습의 개념
 - 머신러닝과 딥러닝의 차이
 - 퍼셉트론 > 다층 퍼셉트론
 - 순전파와 역전파의 과정
 - optimizer 개념
- 성능을 향상을 위한 여러가지 방법

다음주차에는...

- ✓ 이미지 데이터의 특징
 - ✓ CNN의 개념
 - ✓ CNN의 발전 과정
 - ✓ 컴퓨터 비전



“다음주에도 딥예찬과 함께하도록 하지”



THANK YOU

