딥러닝팀

CV팀
강서진
송다은
최종혁
이나연
이민호

INDEX

1

머신러닝과 딥러닝 개요

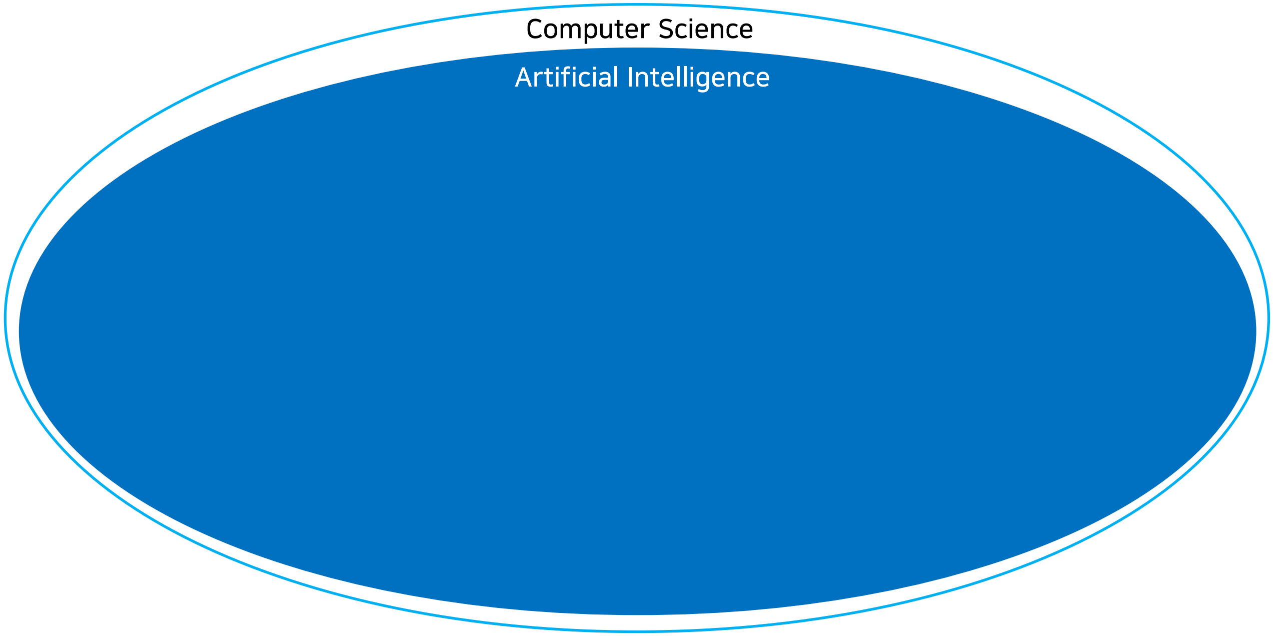
1 머신러닝과 딥러닝 개요

인공지능

인공지능 (AI : Artificial Intelligence)

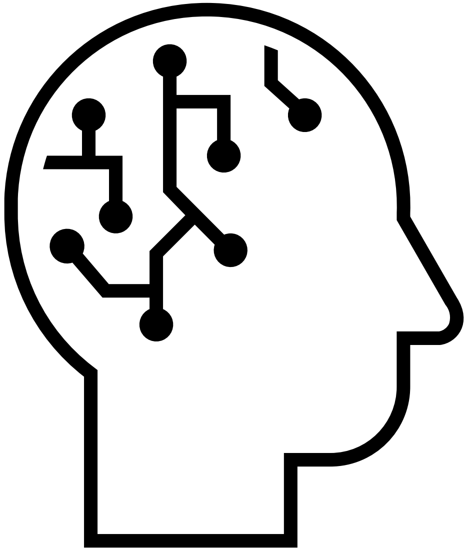
사람이 할 경우 지능이 필요할 것으로 생각되는 작업을

기계가 수행하도록 하는 것



인공지능

강인공지능



인간수준으로 스스로 학습하는 인공지능

약인공지능

특정한 문제를 해결하는 인공지능

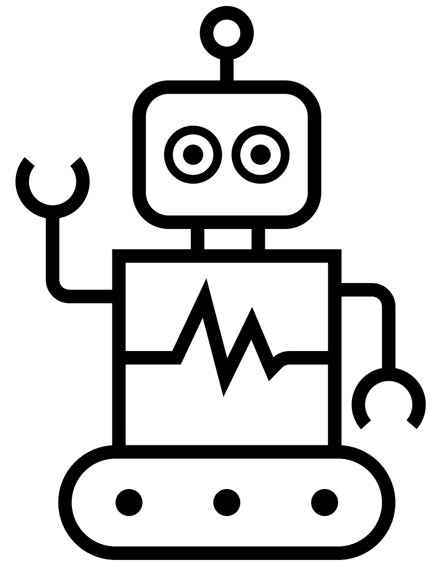
(Narrow AI)

인공지능

강인공지능

인간수준으로 스스로 학습하는 인공지능

약인공지능



특정한 문제를 해결하는 인공지능

(Narrow AI)

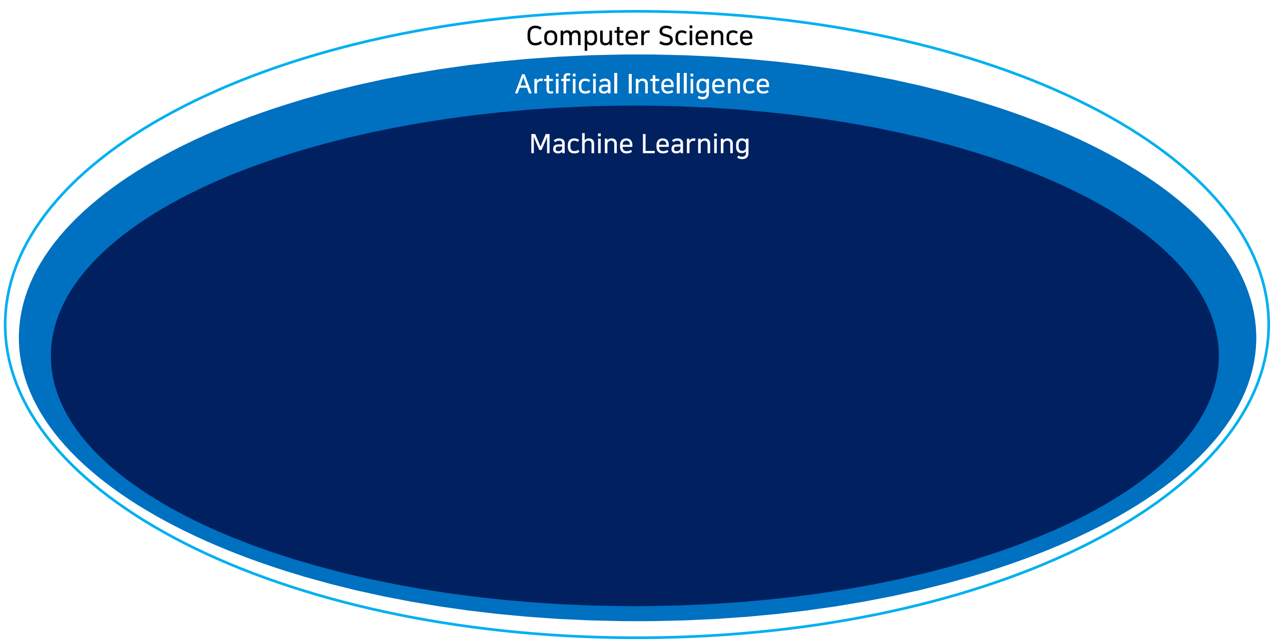
1 머신러닝과 딥러닝 개요

머신러닝

머신러닝 Machine Learning

인공지능의 하위 분야로, 익숙한 규칙 기반 인공지능 방식과는 달리

컴퓨터가 **주어진 데이터를 기반으로 스스로 학습하여 성능을 향상**시키는 방식 사용

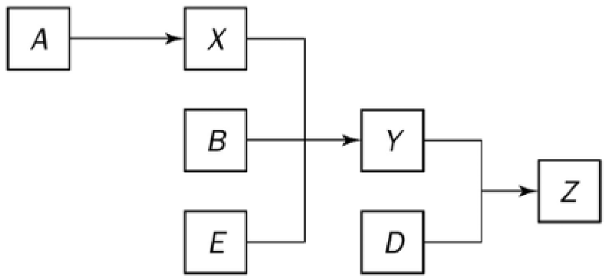


머신러닝

규칙 1: IF *Y가 참이다*
AND *D가 참이다*
THEN *Z가 참이다*

규칙 2: IF *X가 참이다*
AND *B가 참이다*
AND *E가 참이다*
THEN *Y가 참이다*

규칙 3: IF *A가 참이다*
THEN *X가 참이다*



[그림 2-5] 추론 사슬의 예

규칙 기반(Rule-based) 시스템 관련 예시

대표적인 응용 분야는 전문가 시스템(Expert System)

머신러닝

```
if num_room > 3:
    if (latitude, longitude) in Seoul:
        if build_year >= 1990 and build_year < 1995:
            if window == 'North':
                if school_nearby:
                    # Logic for specific conditions
                    ...
    else if (latitude, longitude) in Suwon:
        ...
        ...
        ...
        ...
```

규칙 3: IF *A가 참이다*
THEN *X가 참이다*

[그림 2-5] 추론 사슬의 예

규칙 기반(Rule-based) 시스템 관련 예시

대표적인 응용 분야는 전문가 시스템(Expert System)

그러나 **방대한 데이터**를 다루려고 하면

개별적인 조건들을 모두 분기해가며 → 현실적으로 불가능!

예측 모델을 만들어야 한다는 문제에 직면

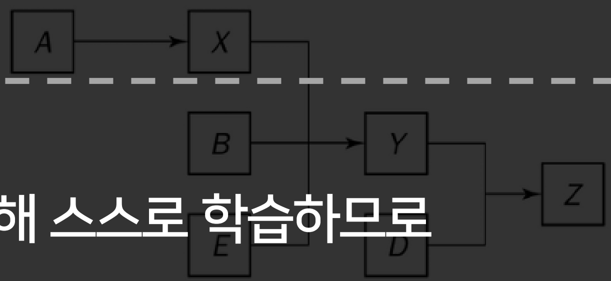
머신러닝

규칙 기반 시스템의 단점 해결

규칙 1: IF Y가 참이다
AND D가 참이다
THEN Z가 참이다

규칙 2: IF X가 참이다
AND B가 참이다
AND E가 참이다
THEN Y가 참이다

규칙 3: IF A가 참이다
THEN X가 참이다



머신러닝은 데이터를 통해 스스로 학습하므로
인간이 명시하지 않아도 경험을 통해
지속적으로 성능을 개선시킬 수 있음

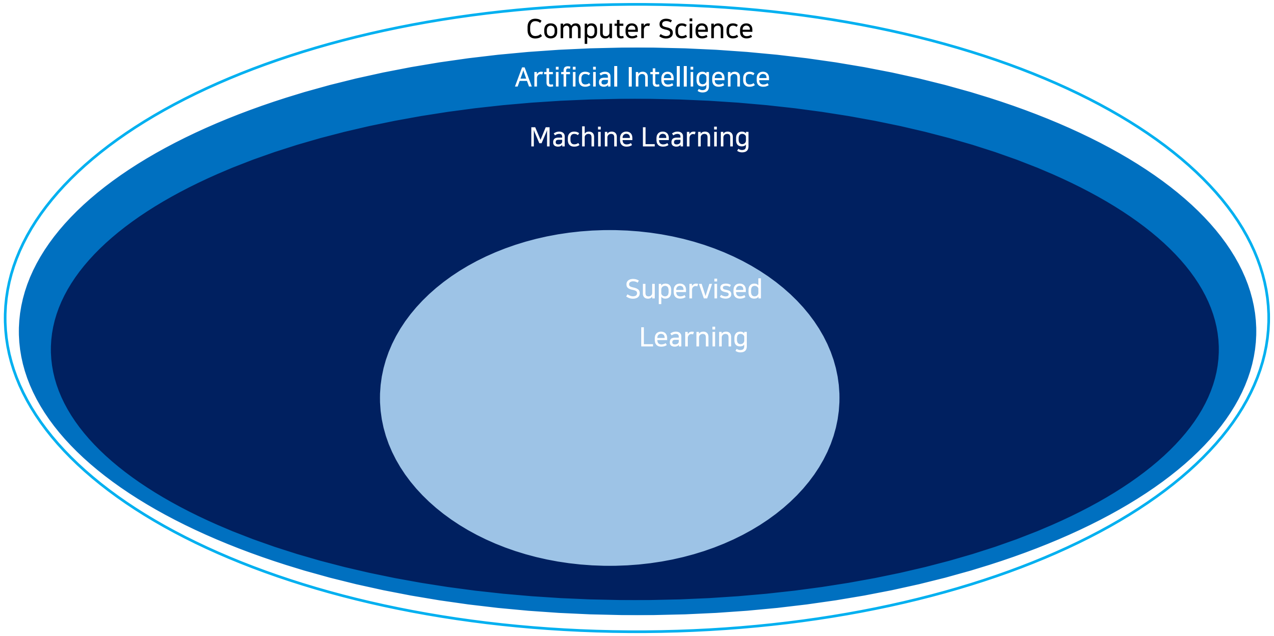
규칙 기반(Rule-based) 시스템 관련 예시
대표적인 응용 분야는 전문가 시스템(Expert System)

머신러닝

지도학습 Supervised Learning

데이터와 레이블이 함께 주어진 경우,

데이터와 레이블의 관계를 학습하는 방식

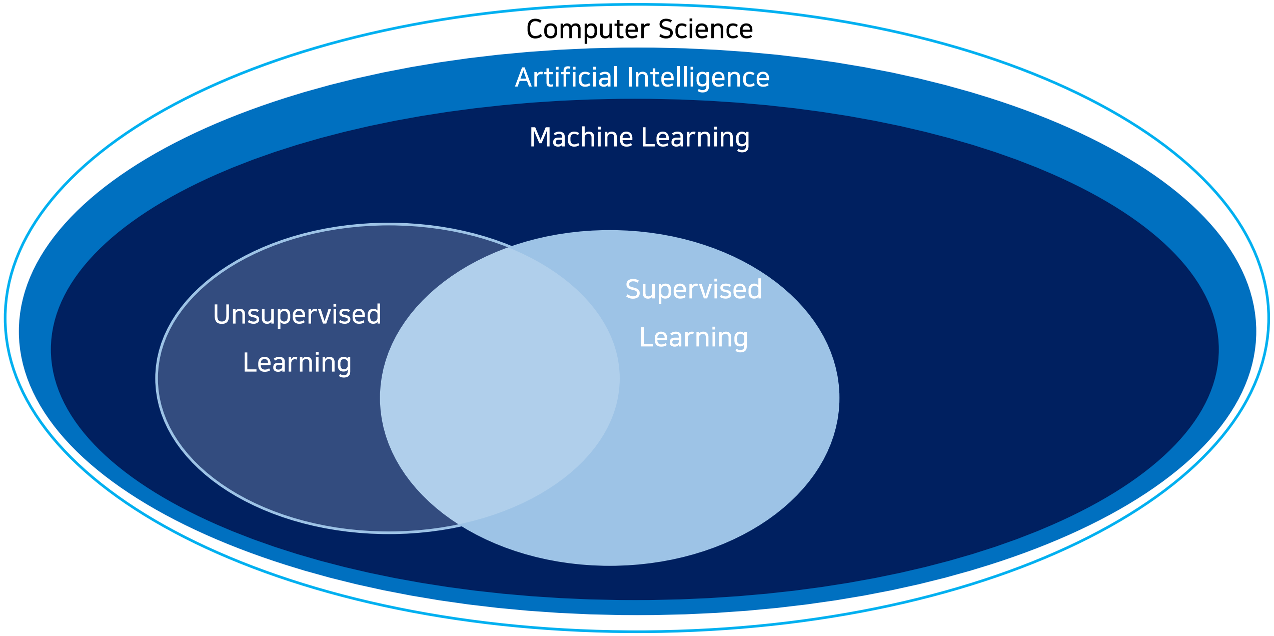


머신러닝

비지도학습 Unsupervised Learning

레이블 없이 데이터만 주어진 경우,

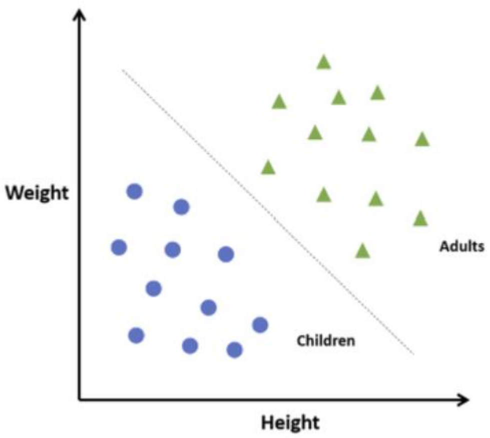
데이터의 특징, 패턴, 구조 등을 파악하는 방식



1 머신러닝과 딥러닝 개요

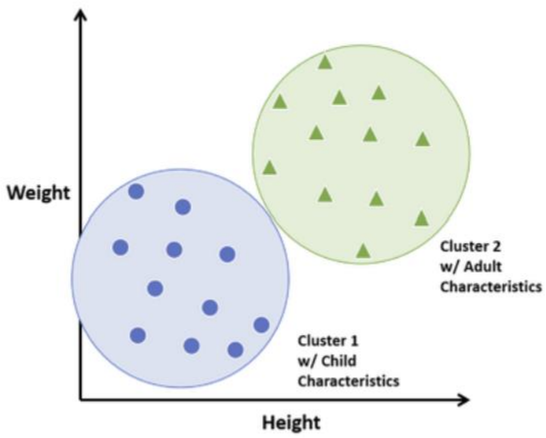
머신러닝

지도학습



Ex) 분류, 회귀, SVM 등

비지도학습



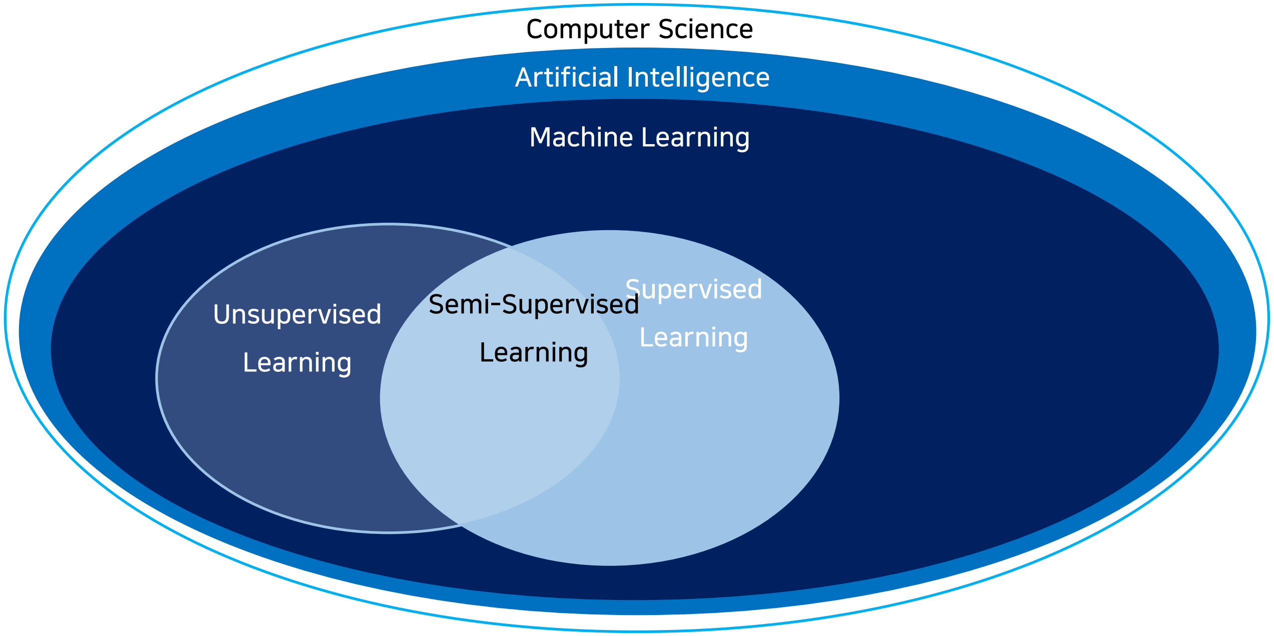
Ex) 클러스터링, PCA, 이상치 탐지 등

머신러닝

준지도학습 Semi-Supervised Learning

레이블이 있는 데이터와 레이블이 없는 데이터를 함께 사용하여

입출력 간의 관계를 학습



머신러닝

지도학습

$$D_L = \{(x_1, y_1), (x_2, y_2), \cdots, (x_n, y_n)\}$$

x_n : 데이터

y_n : 레이블

비지도학습

$$D_U = \{x_1, x_2, \cdots, x_n\}$$

x_n : 데이터

준지도학습

$$D_S = \{D_L, D_U\}$$

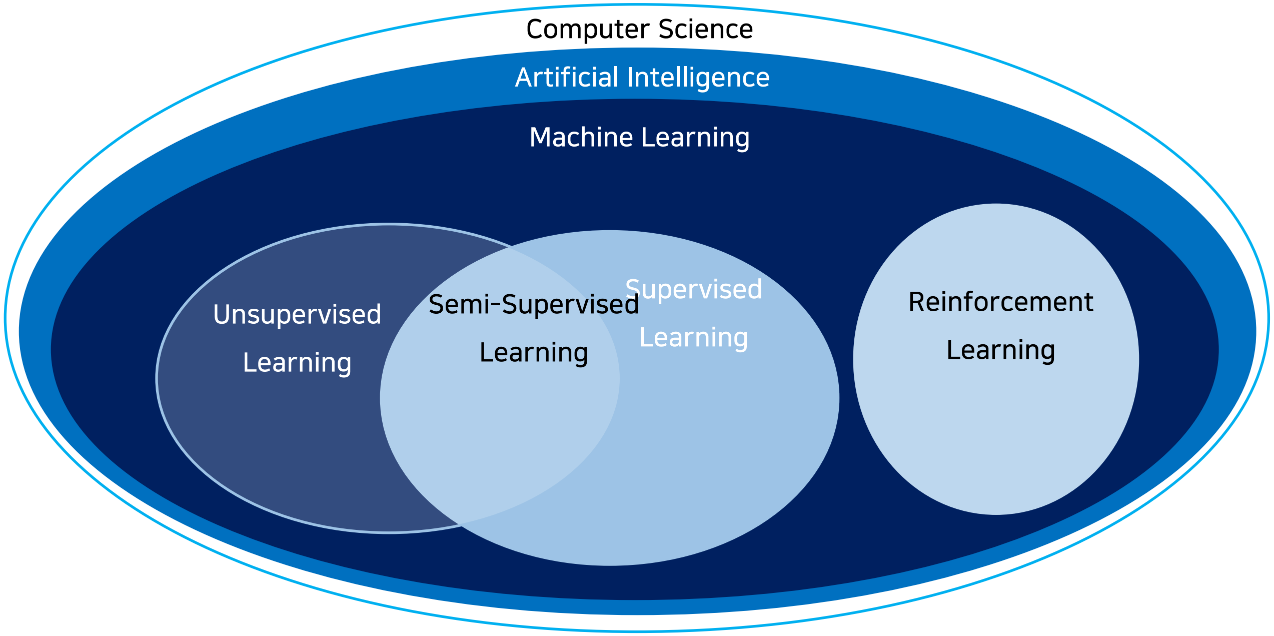
D_L: 레이블 O 데이터 집합

D_U: 레이블 X 데이터 집합

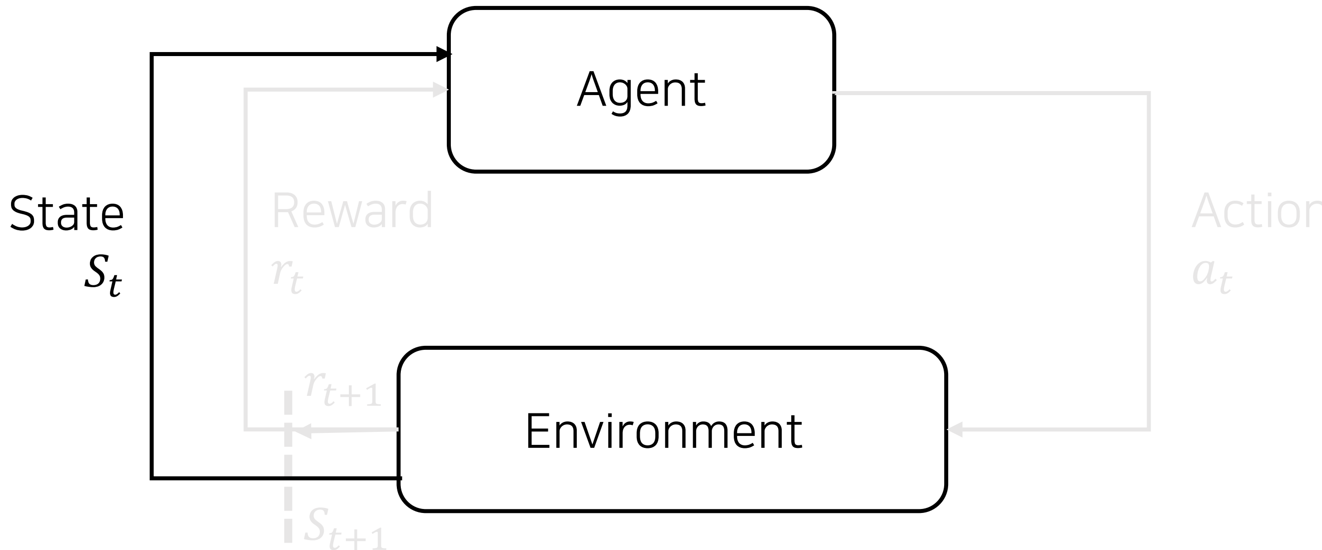
머신러닝

강화학습 Reinforcement Learning

어떤 Environment 안에서 정의된 Agent가 현재의 State를 인식하여, 선택 가능한 Action들 중 Reward를 최대화하는 Policy를 선택하는 방법

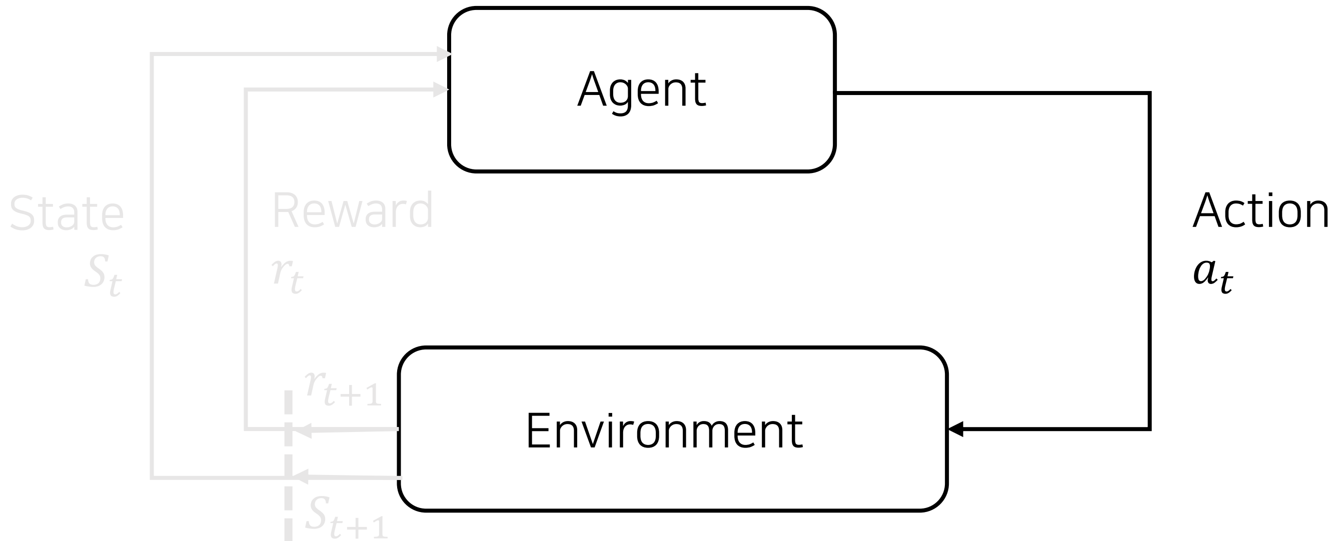


강화학습



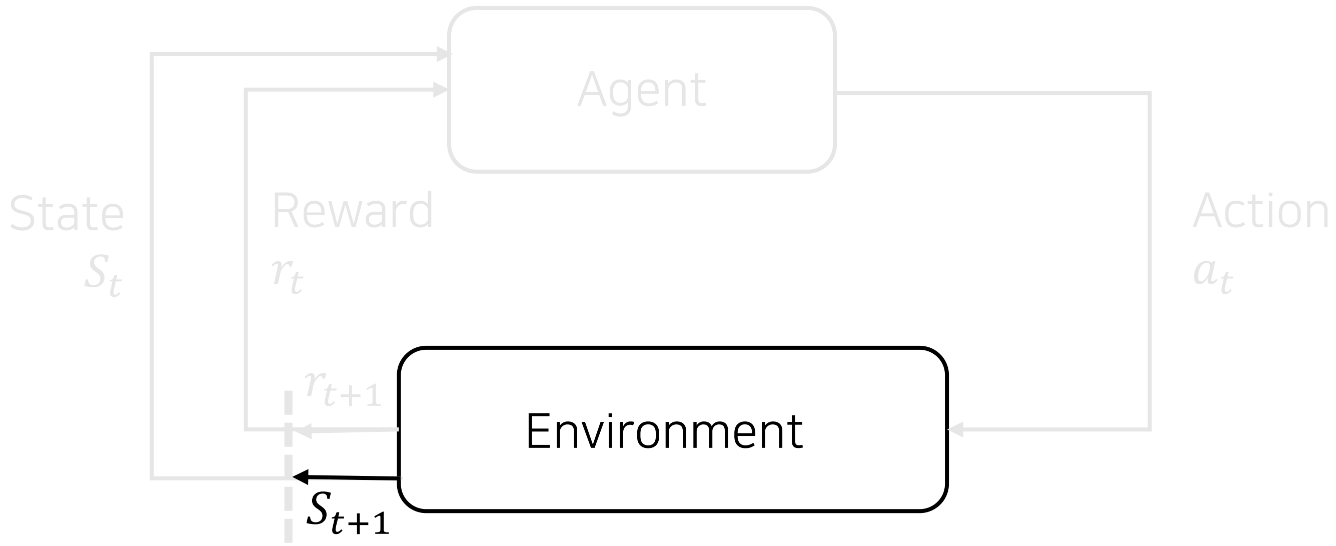
1 현재 Environment에서 관찰한 State S_t가 Agent에 주어짐

강화학습



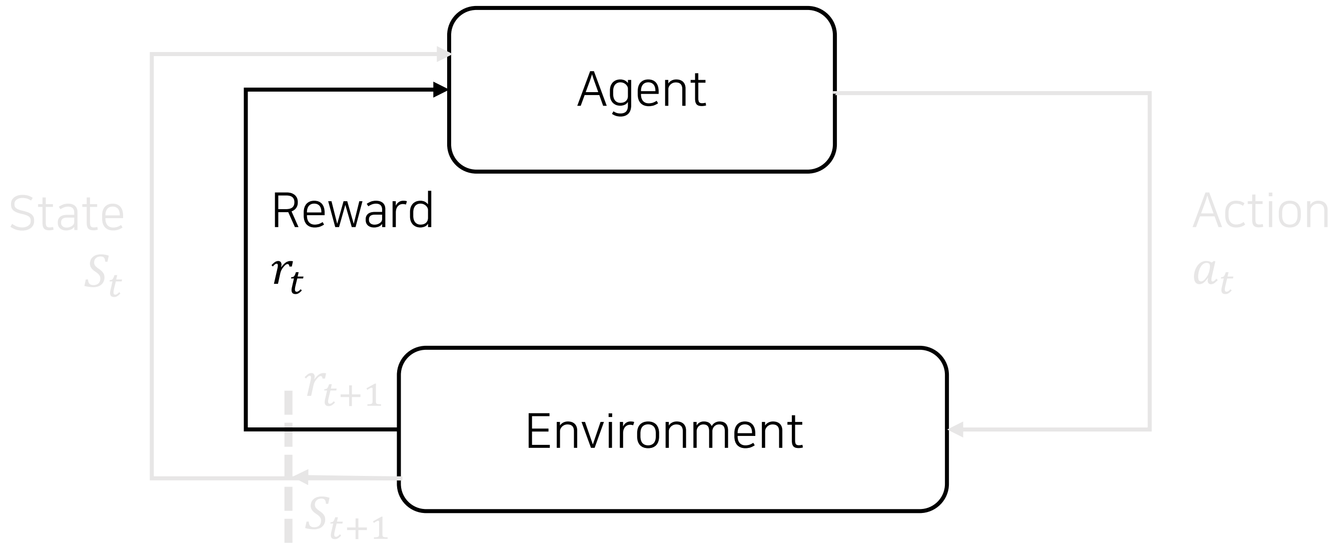
2 Agent는 인식한 상태에 따라 Action a_t 를 수행

강화학습



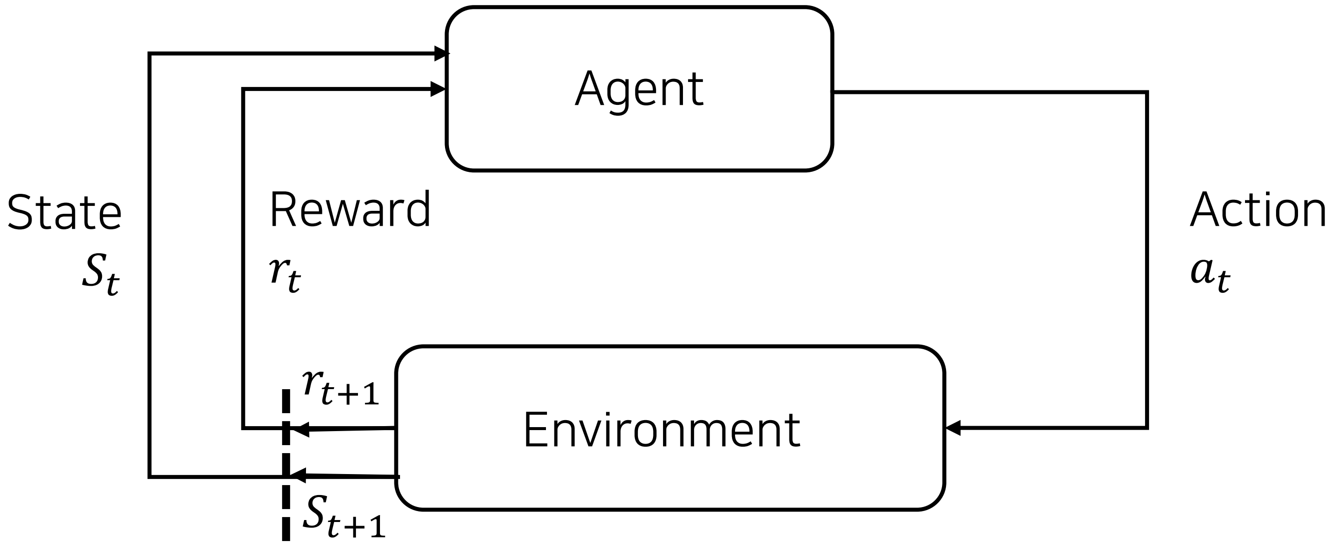
3 그 a_t에 따라 Environment의 State가 S_t 에서 S_{t+1} 로 변화

강화학습



4 변화한 Environment에 따라 Reward r_t 가 Agent에 주어짐

강화학습



단계를 반복하며 최종적인 Reward를 최대화하도록 Action을 조절

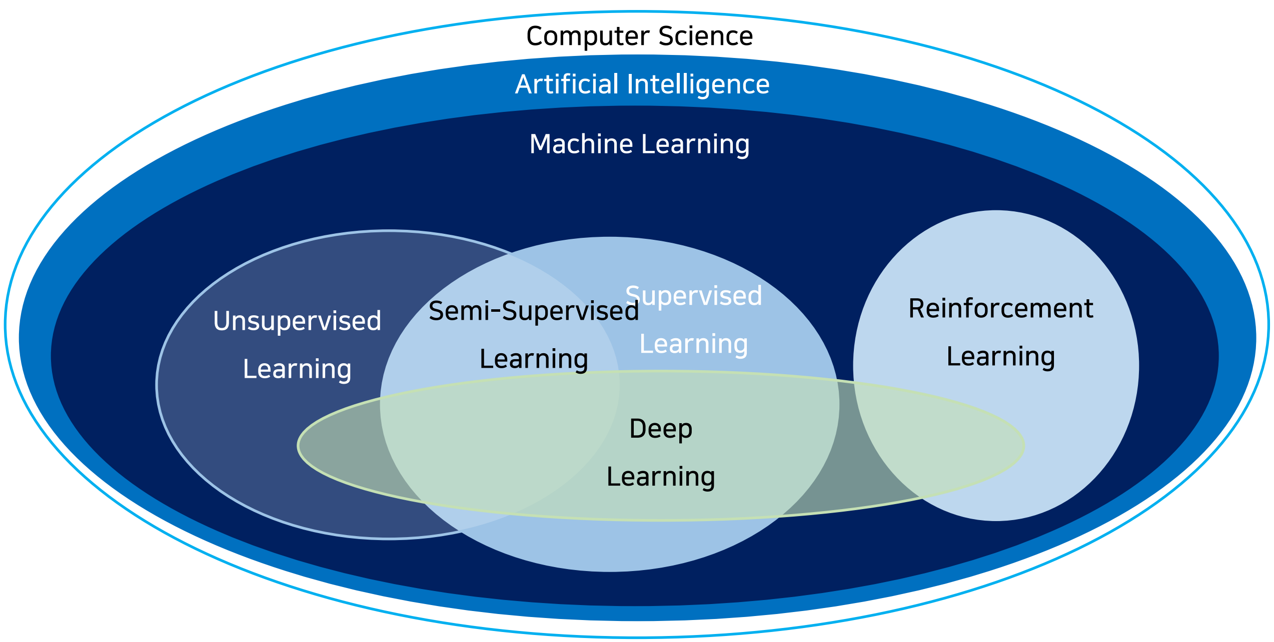
위 5가지 요소만 정의할 수 있다면, 데이터가 주어지지 않더라도 학습 가능

딥러닝

딥러닝 Deep Learning

머신러닝 신경망 학습 분야 중, 심층 인공신경망을 사용한 머신러닝 기법

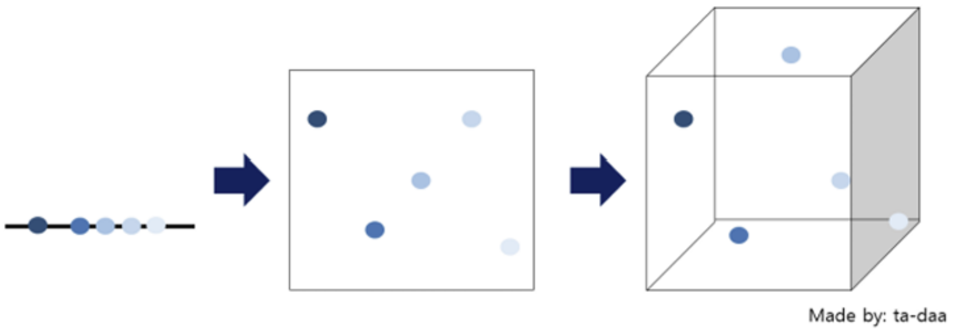
층을 깊게 쌓아 만든 신경망 모델



딥러닝의 필요성 | ① 차원의 저주

차원의 저주

데이터 공간이 고차원으로 커질수록 데이터 밀도가 급격히 감소하여
데이터 분석 또는 머신러닝 모델의 성능을 떨어뜨리는 현상



딥러닝의 필요성 | ① 차원의 저주

차원의 저주

데이터 공간이 고차원으로 커질수록 데이터 밀도가 급격히 감소하여
데이터 분석 또는 머신러닝 모델의 성능을 떨어뜨리는 현상

차원이 점점 늘어날수록 관측치들 간의 거리가 기하급수적으로 멀어지므로,
비슷한 패턴을 찾기 힘들어져 전체 공간을 포괄하지 못하게 됨

딥러닝의 필요성 | ① 차원의 저주

기존의 **머신러닝**은 이 한계점을 어떻게 **처리**했을까?

차원 축소 기법을 활용해서 저차원으로 변환

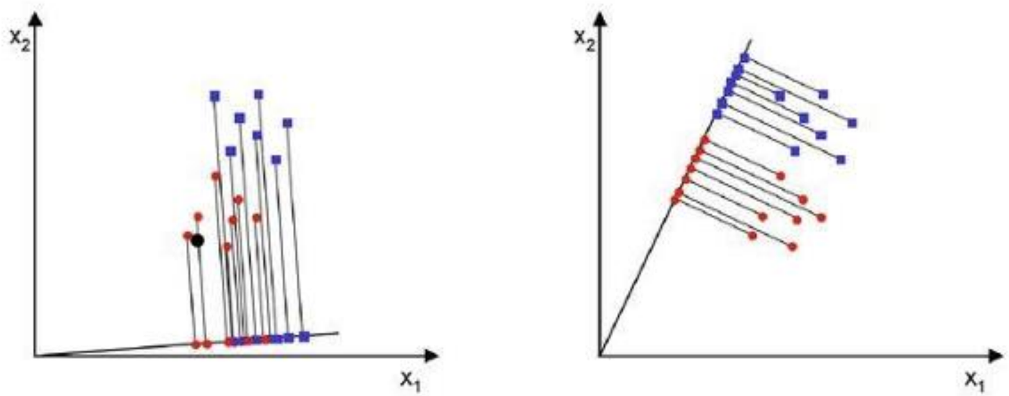
특징선택을 통해 중요한 특징만을 선택하여 차원을 줄임

딥러닝의 필요성 | ① 차원의 저주

Ex) LDA

데이터를 최고로 잘 분류할 수 있는 축으로 데이터를 사영시킴.

기존보다 적은 연산량으로 더욱 좋은 성능을 낼 수 있음!



딥러닝의 필요성 | ① 차원의 저주

기존 머신러닝

최적의 자원을 찾기 위해

다양한 방법을 사용

딥러닝

차원에 대해 매우 강건하다는 특징

딥러닝의 필요성 | ① 차원의 저주

기존 머신러닝

최적의 자원을 찾기 위해

다양한 방법을 사용

딥러닝

차원에 대해 매우 강건하다는 특징

1 머신러닝과 딥러닝 개요

딥러닝의 필요성 | ① 차원의 저주

기존 머신러닝

최적의 자원을 찾기 위해
다양한 방법을 사용해 옴

자동 특징 추출

Automatic Feature Extraction

사람의 지도 없이도
훈련에 사용된 데이터에서
자율적으로 판별에 사용될
적합한 특징을 찾아내는 것

딥러닝

차원에 대해 매우 강건하다는 특징

딥러닝의 필요성 | ② 다양체 가설

다양체

고차원 공간에 존재하지만, 그보다 낮은 차원에서
잘 설명될 수 있는 어떤 구조를 가진 데이터의 집합

즉, 데이터가 n차원 공간에 있음에도
대부분이 유효하지 않은 입력들로 구성되어 있을 수 있다는 것

딥러닝의 필요성 | ② 다양체 가설

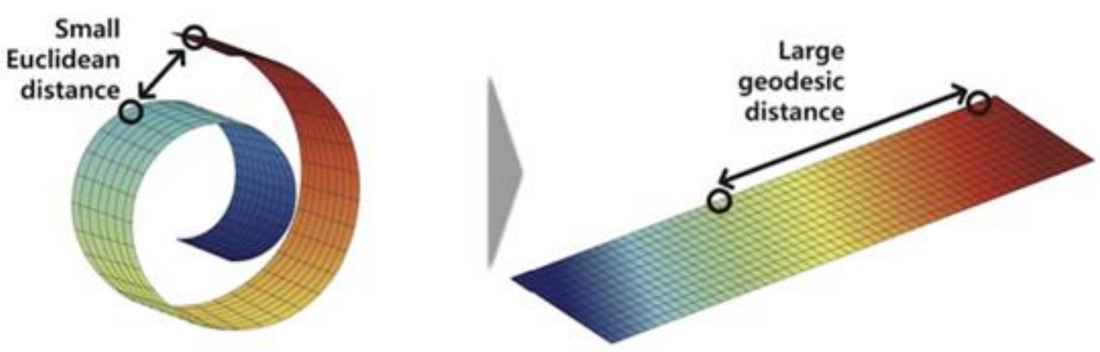
다양체 가설 Manifold Hypothesis

실세계의 고차원 공간 데이터가

고차원에 놓여있는 저차원의 잠재 다양체로 구성된다는 가설

고차원 공간

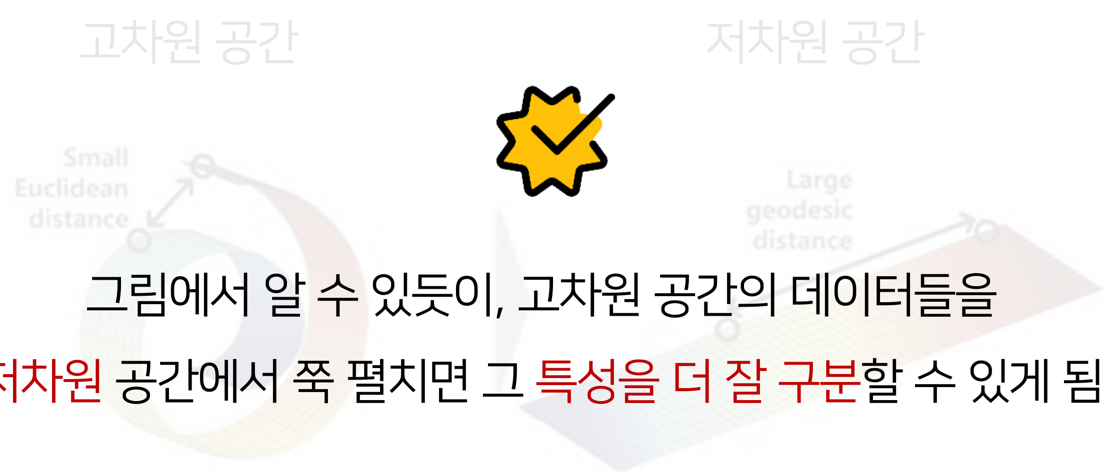
저차원 공간



딥러닝의 필요성 | ② 다양체 가설

다양체 가설 Manifold Hypothesis

실세계의 고차원 공간 데이터가
고차원에 놓여있는 저차원의 잠재 다양체로 구성된다는 가설



그림에서 알 수 있듯이, 고차원 공간의 데이터들을
저차원 공간에서 쭉 펼치면 그 특성을 더 잘 구분할 수 있게 됨

딥러닝의 필요성 | ② 다양체 가설

Ex) 이미지 데이터

이미지의 모든 픽셀 조합이 유의미한 이미지를 생성하지는 않으며
오히려 의미 있는 이미지는 저차원의 다양체에 놓여있을 가능성이 더 많음

이미지의 RGB 값을
랜덤하게 선택하여
이미지를 생성하면
단지 노이즈일 뿐!

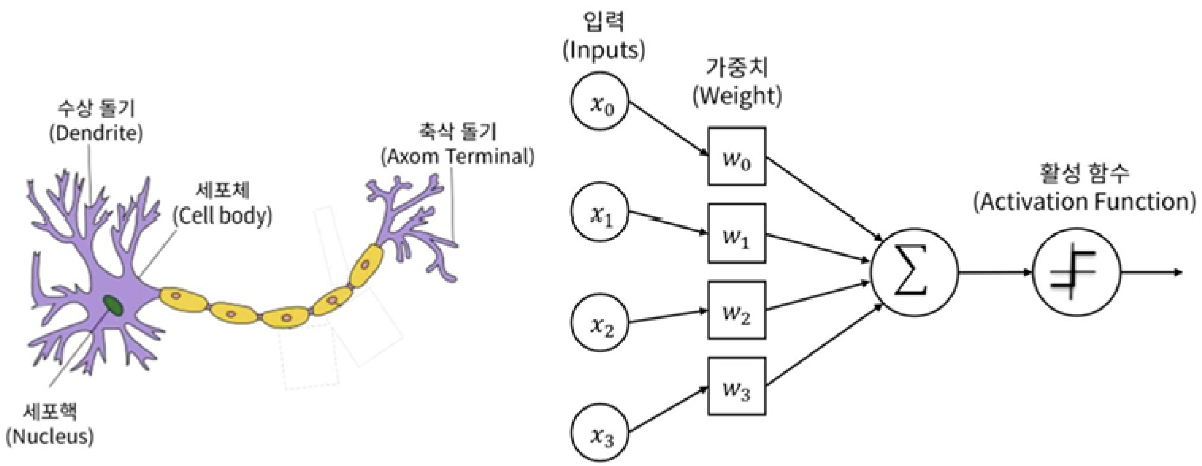
2

인공신경망

인공신경망

인공신경망 Artificial Neural Network

사람의 뉴런 구조를 본떠 만든 머신러닝 모델

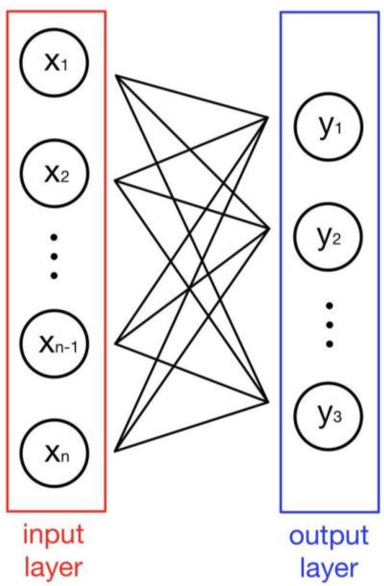


다층 퍼셉트론 | ① 은닉층

퍼셉트론

컴퓨터 소프트웨어로 인공신경망을 구현하기 위해 사용하는 기본 단위

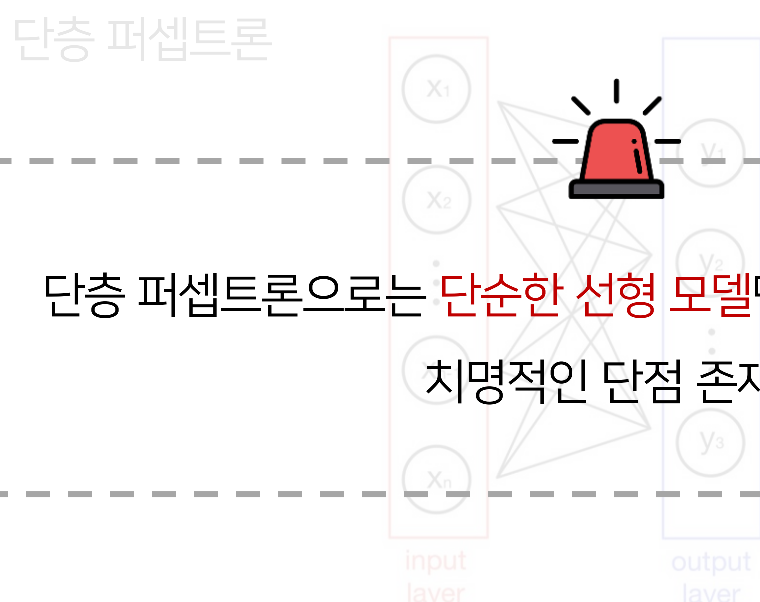
단층 퍼셉트론



다층 퍼셉트론 | ① 은닉층

퍼셉트론

컴퓨터 소프트웨어로 인공신경망을 구현하기 위해 사용하는 기본 단위



단층 퍼셉트론으로는 단순한 선형 모델만을 구현할 수 있다는 치명적인 단점 존재

다층 퍼셉트론 | ① 은닉층

비선형적 현상을 어떻게 모델링할 것인가?

퍼셉트론

컴퓨터 소프트웨어로 인공신경망을 구현하기 위해 사용하는 기본 단위

하나 이상의 은닉층을 모델에 통합하려고 시도함

입력층과 출력층 사이에 은닉층(Hidden Layers)을 쌓아 해결

단층 퍼셉트론

단층 퍼셉트론으로는 단순한 선형 모델만을 구현할 수 있다는

치명적인 단점 존재

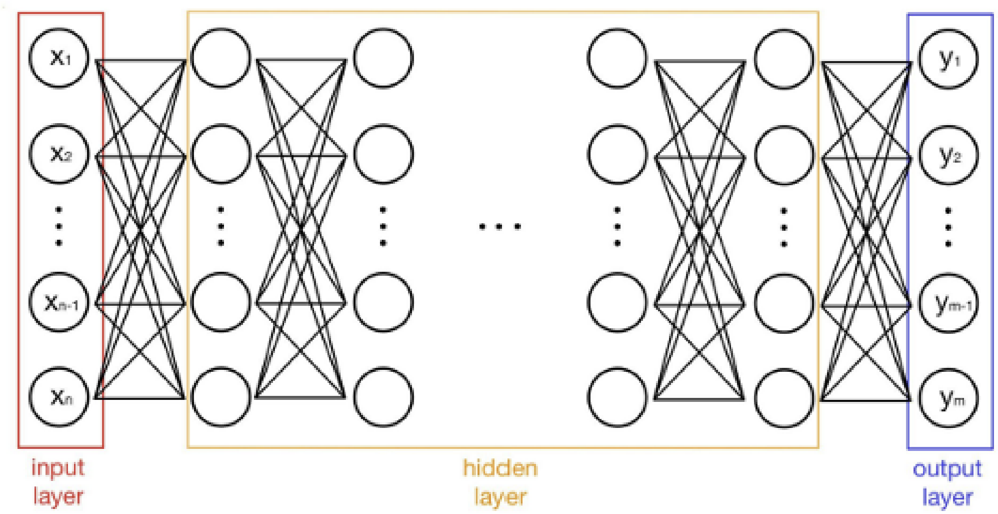
이러한 구조를 2차원적으로 연결!

다층 퍼셉트론 | ① 은닉층

다층 퍼셉트론 Multilayer Perceptron (MLP)

입력층과 출력층 사이에 은닉층을 쌓은 구조를 2차원적으로 연결.

완전 연결 신경망(fully-connected neural network) 이라고도 함.



다층 퍼셉트론 | ② 보편적 근사 정리

보편적 근사 정리 Universal Approximators

" 은닉층을 깊게 쌓은 심층 신경망은 얼마나 강력한가? " 에 대한 답변

단일 은닉층 네트워크여도, 충분한 노드와 적절한 가중치 집합이 주어진다면 어떤 함수든 모델링 가능. 그러나 함수 학습의 어려움으로 인해, 더 깊은 네트워크를 사용함으로써 많은 함수를 훨씬 더 간결하게 근사하고자 할 뿐!

- Simonyan and Ziserman(2014)

활성화함수

활성화함수 Activation Function

신경망을 비선형(non-linear)으로 만드는 함수

다층 퍼셉트론의 각 은닉층들에 활성화함수를 추가함

활성화함수

활성화함수 Activation Function

신경망을 비선형(non-linear)으로 만드는 함수

왜 모델에 비선형성을 여러 번 추가할까?

세상의 데이터들은 선형모델로 표현할 수 없는 경우가 대다수.

실제 데이터들의 결정경계(Decision Boundary)가 선형이 되기 힘듦

다층 퍼셉트론의 각 은닉층들에 활성화함수를 추가함

활성화함수를 통해 입력 데이터 공간을 왜곡시켜

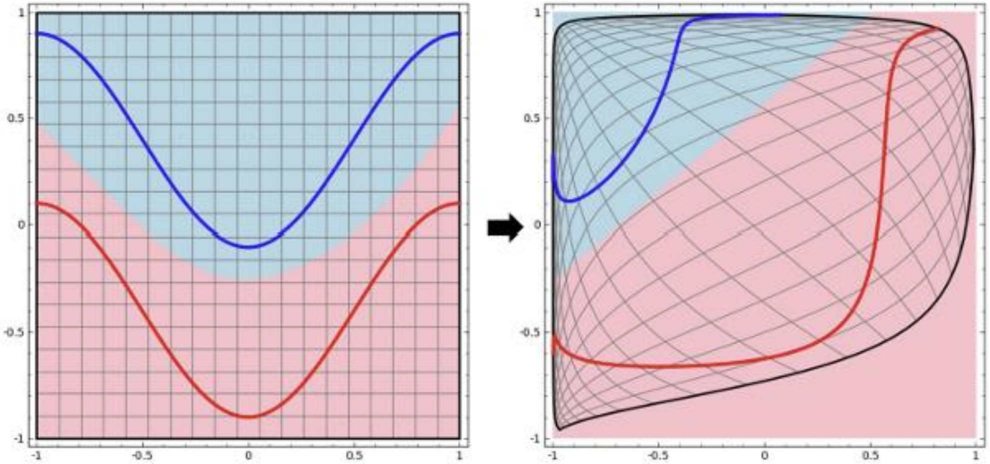
이후 layer에서 결정경계를 찾는 것을 용이하게 만들어 줌

활성화함수

즉, 퍼셉트론을 여러 층으로 구성한다는 것은

비선형적인 특징을 잘 표현할 수 있도록 표현력을 높여주며

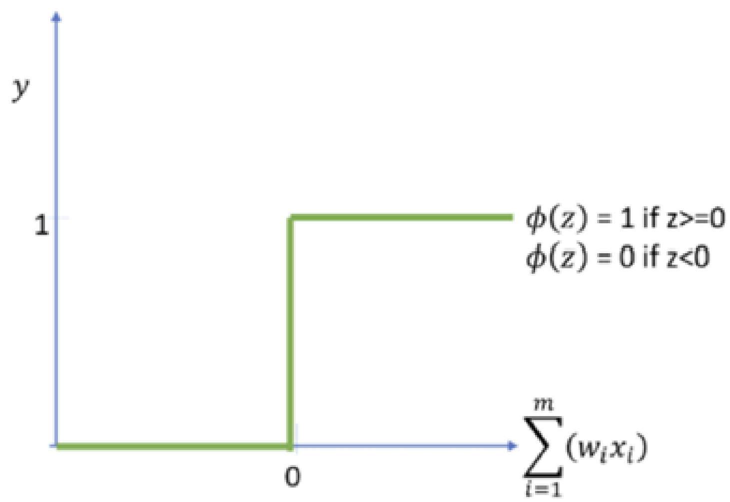
이는 선형의 결정 경계를 찾기 쉽도록 공간을 잘 왜곡시키는 과정!



활성화 함수 | ① Sign

Sign 함수

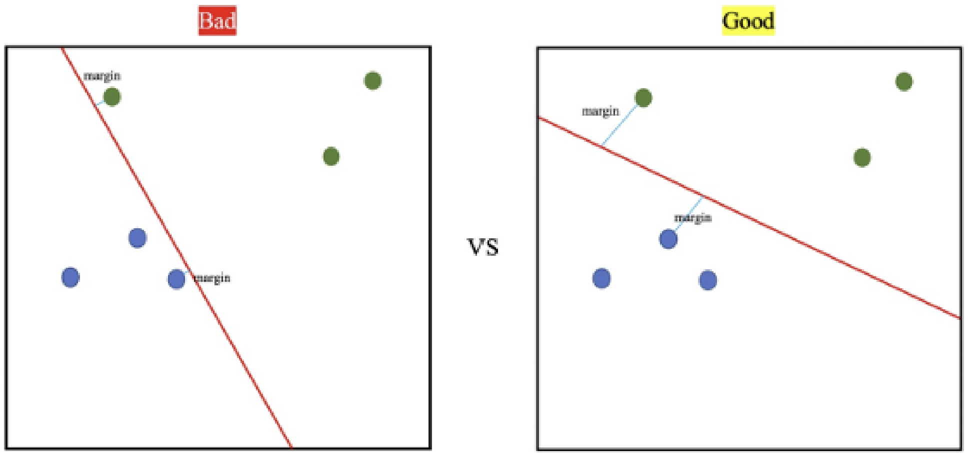
단순히 부호를 나타내는 함수로 고전적인 퍼셉트론에서 사용



활성화 함수 | ① Sign

Sign 함수의 문제점

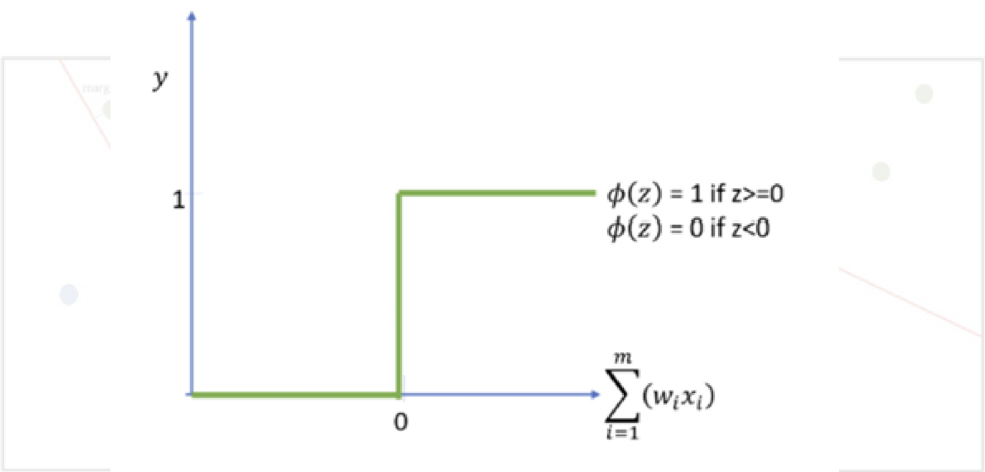
- 데이터와 Decision Boundary 간의 거리 정보를 고려하지 않음
- 미분이 불가능하거나 값이 0이라서 Gradient Descent 사용 불가



활성화 함수 | ① Sign

Sign 함수의 문제점

- 데이터와 Decision Boundary 간의 거리 정보를 고려하지 않음
- 미분이 불가능하거나 값이 0이라서 Gradient Descent 사용 불가

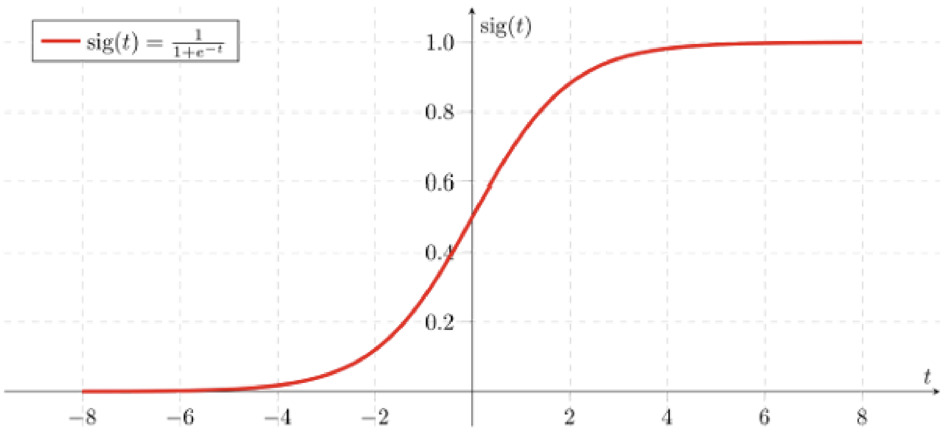


활성화 함수 | ② Sigmoid

Sigmoid 함수

실수 범위의 input을 0 ~ 1사이의 연속적인 실수 값으로 스쿼싱

입력이 작으면 0에 가까운 값, 크면 1에 가까운 값 출력



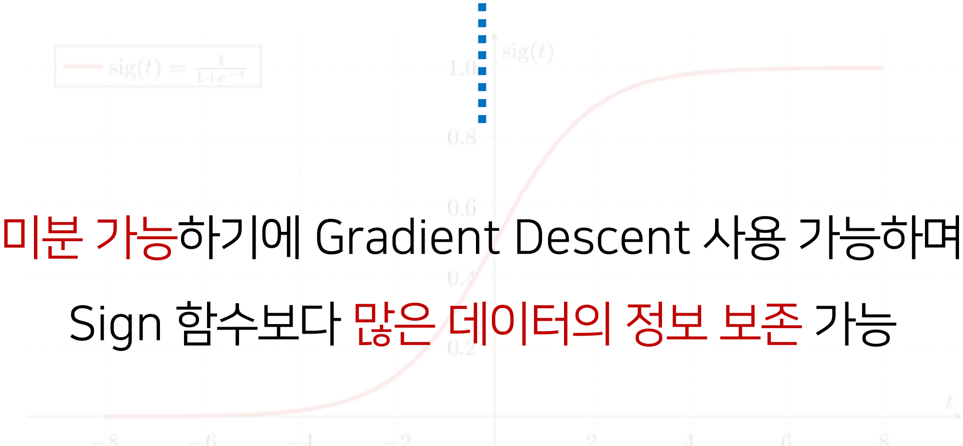
$$sigmoid(x) = \frac{1}{1 + \mathrm{e}^{-x}}$$

활성화 함수 | ② Sigmoid

Sigmoid 함수

실수 범위의 input을 0 ~ 1사이의 연속적인 실수 값으로 스쿼싱

입력이 작으면 0에 가까운 값, 크면 1에 가까운 값 출력



미분 가능하기에 Gradient Descent 사용 가능하며

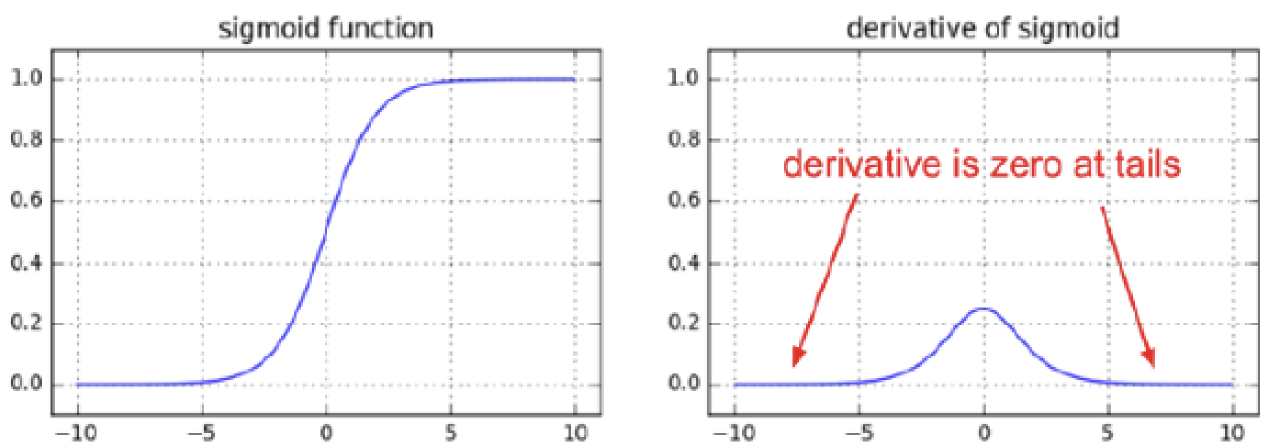
Sign 함수보다 많은 데이터의 정보 보존 가능

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

활성화 함수 | ② Sigmoid

Sigmoid 함수의 문제점

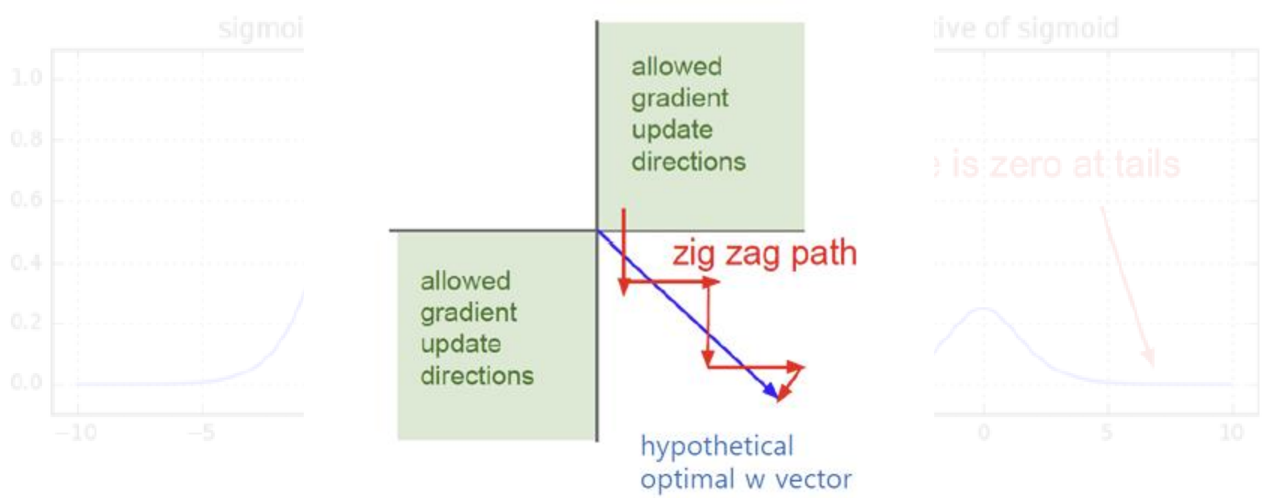
- Gradient Vanishing 현상이 발생하여 추가적인 학습 불가
- 중앙의 값이 0이 아니기 때문에 학습 방향이 치우치는 zig zag 문제



활성화 함수 | ② Sigmoid

Sigmoid 함수의 문제점

- Gradient Vanishing 현상이 발생하여 추가적인 학습 불가
- 중앙의 값이 0이 아니기 때문에 수렴 속도가 느려지는 zig zag 문제



활성화 함수 | ② Sigmoid

Output layer에서의 Sigmoid 함수

0과 1 사이의 유사확률로 해석이 가능

따라서 출력층에서 Binary classification을 수행할 때 널리 이용됨.

활성화 함수 | ② Sigmoid

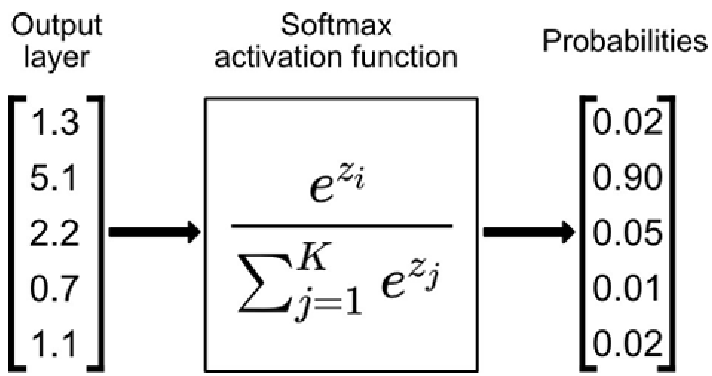
Softmax 함수

분류 문제에서 Multi-classification인 경우, 출력층에서 사용

$$\mathit{softmax}(x_i) = \frac{\mathrm{e}^{x_i}}{\Sigma_j \mathrm{e}^{x_j}}$$

전체 합이 1이 되도록 만들어진 다중 분류 함수

활성화 함수 | ② Sigmoid



출력층의 활성화함수로 사용하여 다중 분류 시행

↓

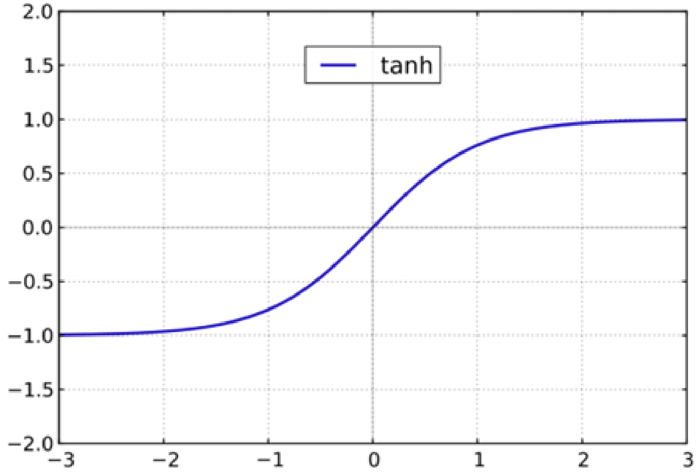
확률이 제일 높은 항목의 클래스로 분류 결과 예측

활성화 함수 | ③ Tanh

Tanh 함수

Sigmoid 함수의 대칭점 값이 0이 아니어서 발생한

zig zag 문제를 해결하기 위해 고안



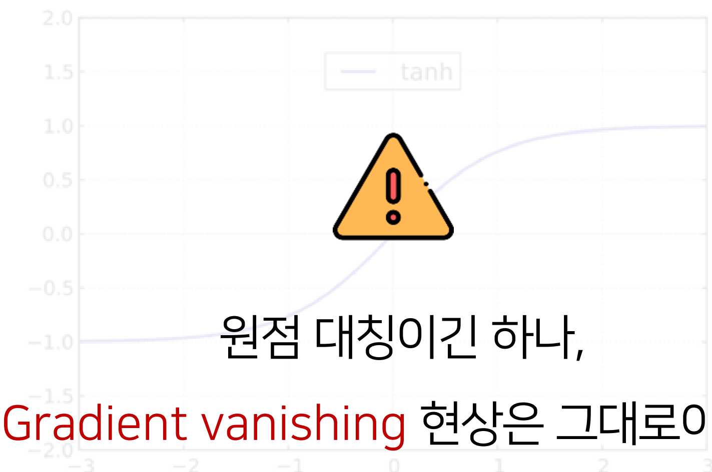
$$\tanh(x) = \frac{e^{x} - e^{-x}}{e^{x} + e^{-x}}$$

활성화 함수 | ③ Tanh

Tanh 함수

Sigmoid 함수의 대칭점 값이 0이 아니어서 발생한

zig zag 문제를 해결하기 위해 고안



원점 대칭이긴 하나,

Gradient vanishing 현상은 그대로이기에

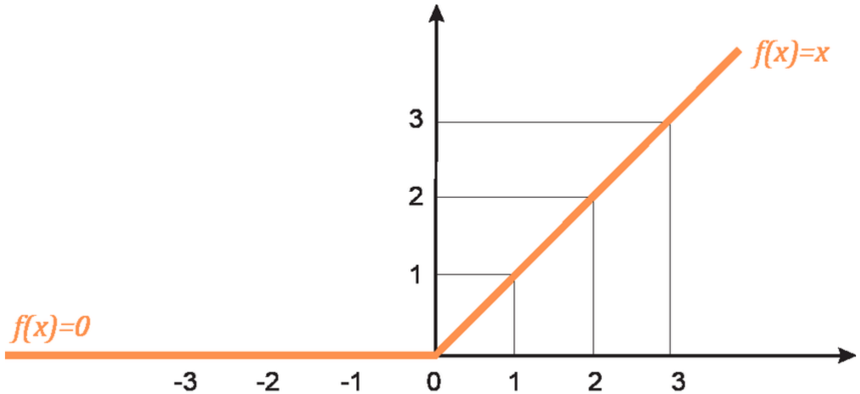
잘 사용되지 않고 있음!

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

활성화함수의 종류

ReLU 함수

양수 영역에서 $y = x$, 음수 영역에서 0의 값을 가지는 함수



활성화함수의 종류

ReLU의 장점

- Gradient Vanishing 문제 해결
- 구현의 단순함
- 빠른 연산속도

ReLU의 단점

음수 영역의 경우

출력 값과 미분 값이 0

↓

Dying ReLU

활성화함수의 종류

ReLU의 장점

- Gradient Vanishing 문제 해결
- 구현의 단순함
- 빠른 연산속도

ReLU의 단점

음수 영역의 경우

출력 값과 미분 값이 0

↓

Dying ReLU

활성화함수의 종류

ReLU의 단점 해결

음수 영역에서의 Dying ReLU 현상을 해결하기 위해 다양한 함수(Leaky ReLU, PReLU, ELU, SELU) 들이 제시되고 있음

ReLU의 단점

음수 영역의 경우

출력 값과 미분 값이 0

↓

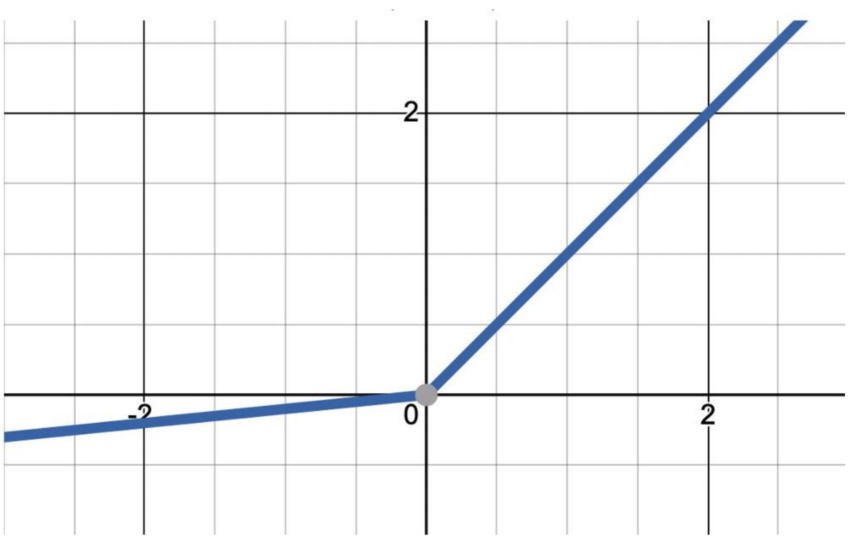
Dying ReLU

활성화함수의 종류

Leaky ReLU 함수

음수 영역에 **조금의 기울기**를 주어 Dying ReLU 현상을 해결

$$Leaky\ ReLU(x) = \max(0.01x, x)$$

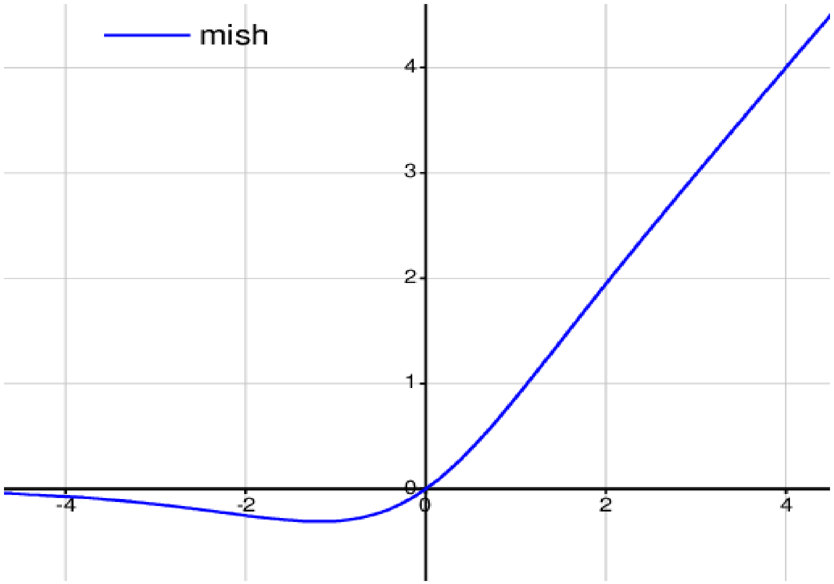


활성화함수의 종류

Mish 함수

이미지 분류와 객체탐지에서 뛰어난 성능을 보이는 최신 활성화함수

$$Mish(x) = x * tanh\big(softplus(x)\big) = x * tanh(ln(1 + e^{x}))$$



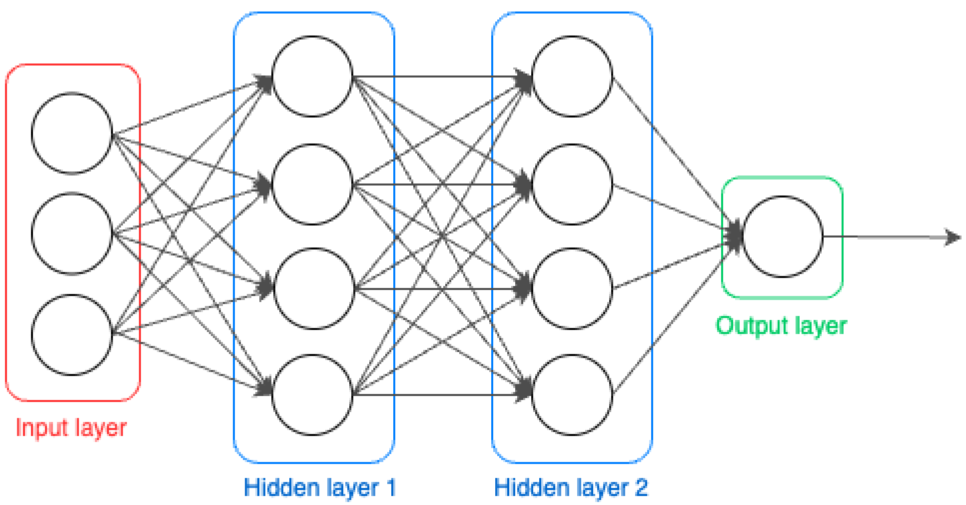
3

신경망 학습

순전파

순전파 Forward Propagation

입력 데이터를 기반으로 신경망을 따라 입력층부터 출력층까지 차례대로 변수를 계산 후 결과를 도출하는 **추론 과정**



순전파

순전파 Forward Propagation

입력 데이터를 기반으로 신경망을 따라 입력층부터 출력층까지
차례대로 변수를 계산 후 결과를 도출하는 **추론 과정**

다중 퍼셉트론을 통한 추론

① 입력층에 대해서 가중치 행렬 곱 연산

$$\begin{bmatrix} w_{00} & w_{01} & w_{02} \\ w_{10} & w_{11} & w_{12} \\ w_{20} & w_{21} & w_{22} \end{bmatrix} * \begin{bmatrix} x_0 \\ x_1 \\ x_2 \end{bmatrix} + b = \begin{bmatrix} w_{00} * x_0 + & w_{01} * x_1 + & w_{02} * x_2 + b \\ w_{10} * x_0 + & w_{11} * x_1 + & w_{12} * x_2 + b \\ w_{20} * x_0 + & w_{21} * x_1 + & w_{22} * x_2 + b \end{bmatrix}$$

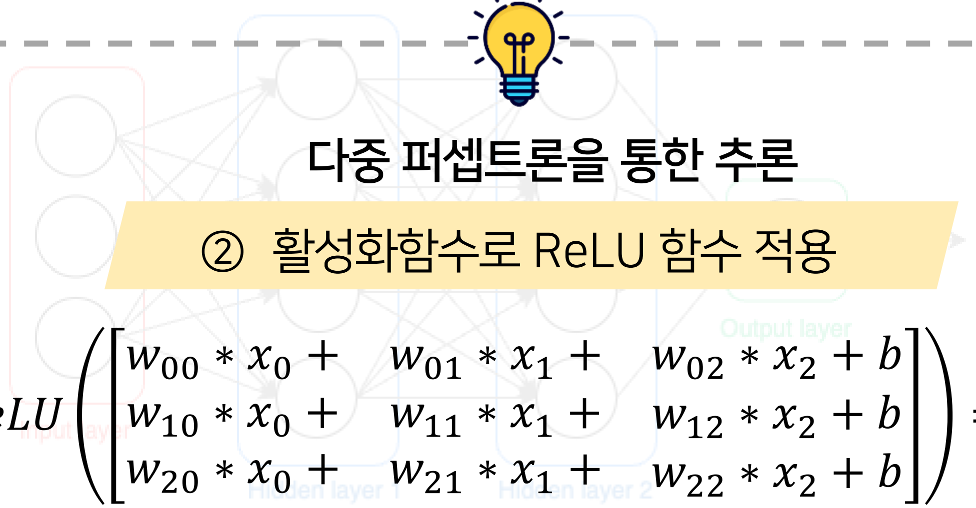
순전파

순전파 Forward Propagation

입력 데이터를 기반으로 신경망을 따라 입력층부터 출력층까지
차례대로 변수를 계산 후 결과를 도출하는 **추론 과정**

다중 퍼셉트론을 통한 추론

② 활성화함수로 ReLU 함수 적용



$$ReLU\left(\begin{bmatrix} w_{00} * x_0 + & w_{01} * x_1 + & w_{02} * x_2 + b \\ w_{10} * x_0 + & w_{11} * x_1 + & w_{12} * x_2 + b \\ w_{20} * x_0 + & w_{21} * x_1 + & w_{22} * x_2 + b \end{bmatrix}\right) = \begin{bmatrix} h_0 \\ h_1 \\ h_2 \end{bmatrix}$$

순전파

순전파 Forward Propagation

입력 데이터를 기반으로 신경망을 따라 입력층부터 출력층까지
차례대로 변수를 계산 후 결과를 도출하는 **추론 과정**

다중 퍼셉트론을 통한 추론

③ 은닉층에 대해서 가중치 행렬 곱 연산

$$\begin{bmatrix} w'_{00} & w'_{01} & w'_{02} \\ w'_{10} & w'_{11} & w'_{12} \end{bmatrix} * \begin{bmatrix} h_0 \\ h_1 \\ h_2 \end{bmatrix} + b' = \begin{bmatrix} w'_{00} * h_0 + & w'_{01} * h_1 + & w'_{02} * h_2 + b' \\ w'_{10} * h_0 + & w'_{11} * h_1 + & w'_{12} * h_2 + b' \end{bmatrix}$$

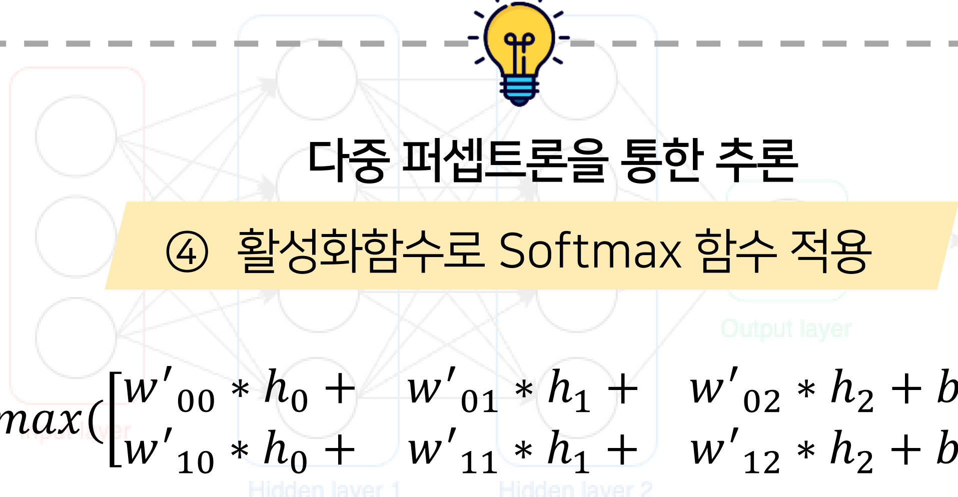
순전파

순전파 Forward Propagation

입력 데이터를 기반으로 신경망을 따라 입력층부터 출력층까지
차례대로 변수를 계산 후 결과를 도출하는 **추론 과정**

다중 퍼셉트론을 통한 추론

④ 활성화함수로 Softmax 함수 적용



$$softmax\left(\begin{bmatrix} w'_{00} * h_0 + & w'_{01} * h_1 + & w'_{02} * h_2 + b' \\ w'_{10} * h_0 + & w'_{11} * h_1 + & w'_{12} * h_2 + b' \end{bmatrix}\right) = \begin{bmatrix} \widehat{y_0} \\ \widehat{y_1} \end{bmatrix}$$

손실함수 계산

Mean Squared Error

회귀 문제에서 적용

$$MSE = \frac{1}{2}\left((y_0 - \hat{y}_0)^2 + (y_1 - \hat{y}_1)^2\right)$$

$$MSE = \frac{1}{2}\left((y - \hat{y})^T (y - \hat{y})\right)$$

Cross Entropy

분류 문제에서 적용

$$CE = -\sum_{i=1}^{C} y_i \log(\hat{y}_i)$$

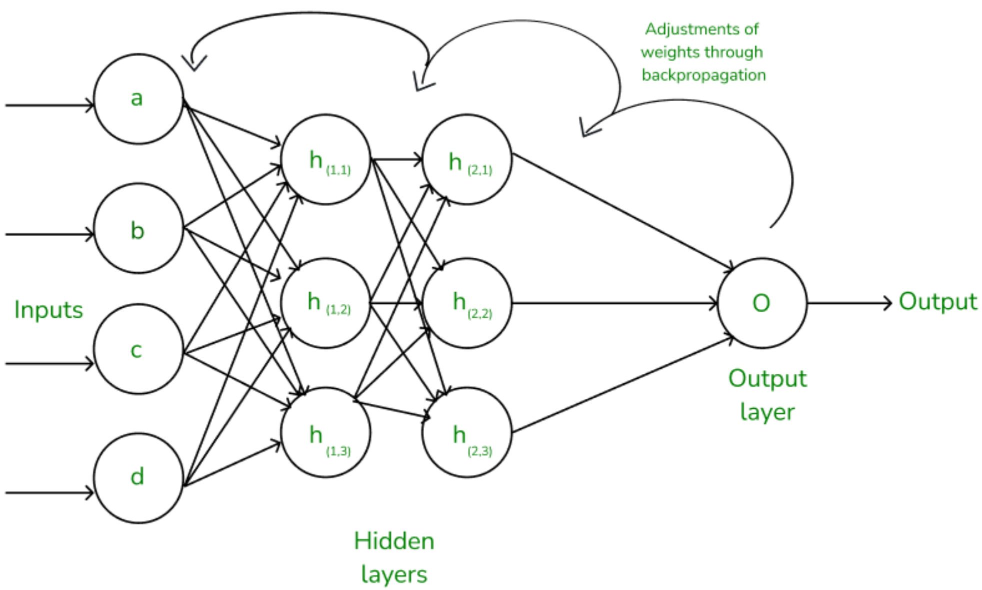
$$BCE = -ylog(\hat{y}) - (1 - y)\log(1 - \hat{y})$$

역전파

역전파 Backward Propagation

모델이 예측을 위한 **최적의 파라미터**를 학습해나가는 과정

손실함수로 구한 예측 값과 정답의 차이를 최소화 시키는 것이 목적

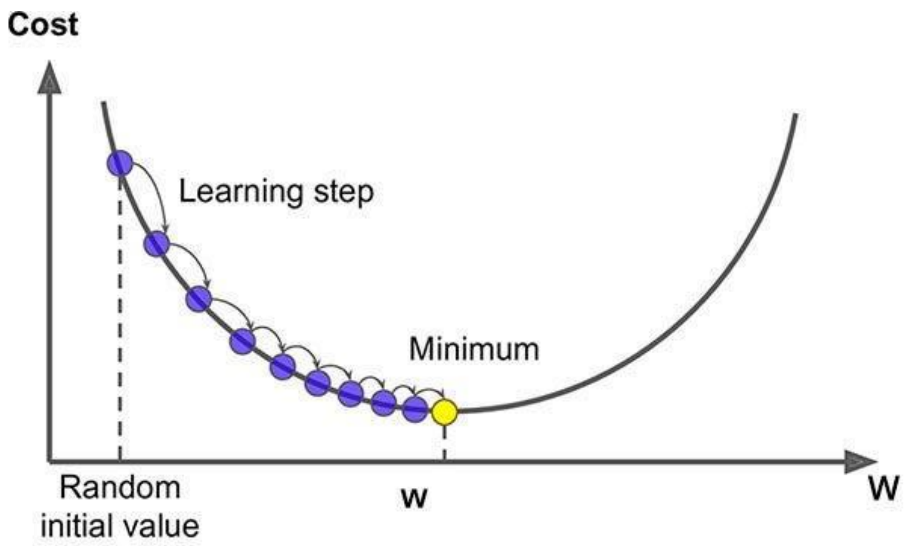


역전파

경사하강법 Gradient Descent

Gradient를 계산하여 **파라미터의 최적 값**을 찾아가는 최적화방법

가중치와 편향을 업데이트하면서 모델 학습



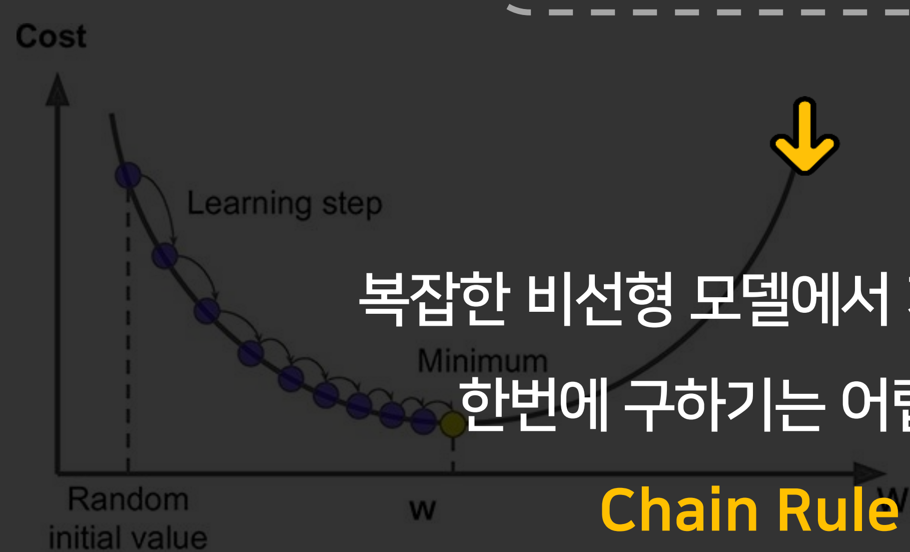
✓ $$w = w - \eta \frac{\partial C}{\partial w}$$

역전파

경사하강법 Gradient Descent

Gradient를 계산하여 **파라미터의 최적 값**을 찾아가는 최적화방법

가중치와 편향을 업데이트하면서 모델 학습



$$w = w - \eta \frac{\partial C}{\partial w}$$

한번에 구하기는
쉽지 않음ㅠㅠ

복잡한 비선형 모델에서 가중치와 편향을

한번에 구하기는 어렵기 때문에,

Chain Rule 사용!

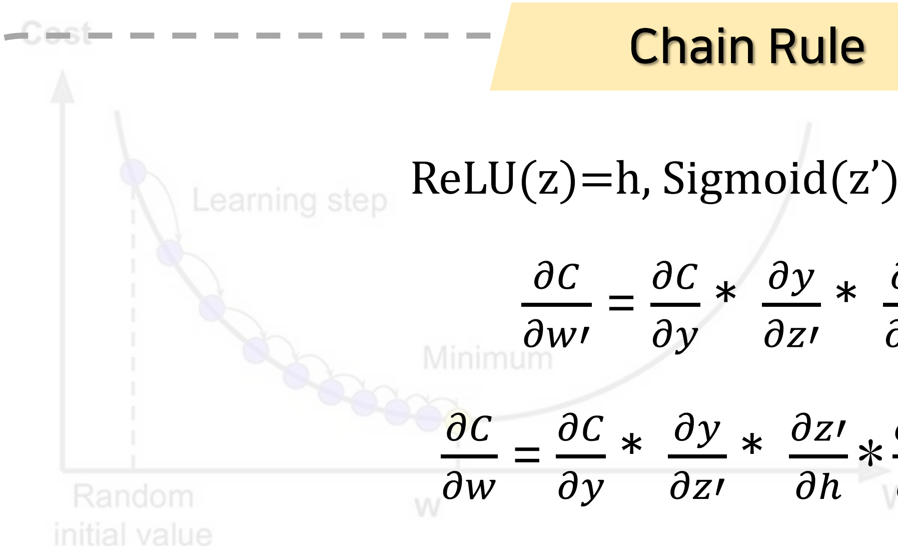
역전파

경사하강법 Gradient Descent

Gradient를 이용하여 **파라미터의 최적 값**을 찾아가는 최적화방법

가중치와 편향을 업데이트하면서 모델 학습

Chain Rule



ReLU(z)=h, Sigmoid(z')=y 일때,

$$\frac{\partial C}{\partial w'} = \frac{\partial C}{\partial y} * \frac{\partial y}{\partial z'} * \frac{\partial z'}{\partial w'}$$

$$\frac{\partial C}{\partial w} = \frac{\partial C}{\partial y} * \frac{\partial y}{\partial z'} * \frac{\partial z'}{\partial h} * \frac{\partial h}{\partial z} * \frac{\partial z}{\partial w}$$

$$w = w - \eta \frac{\partial C}{\partial w}$$

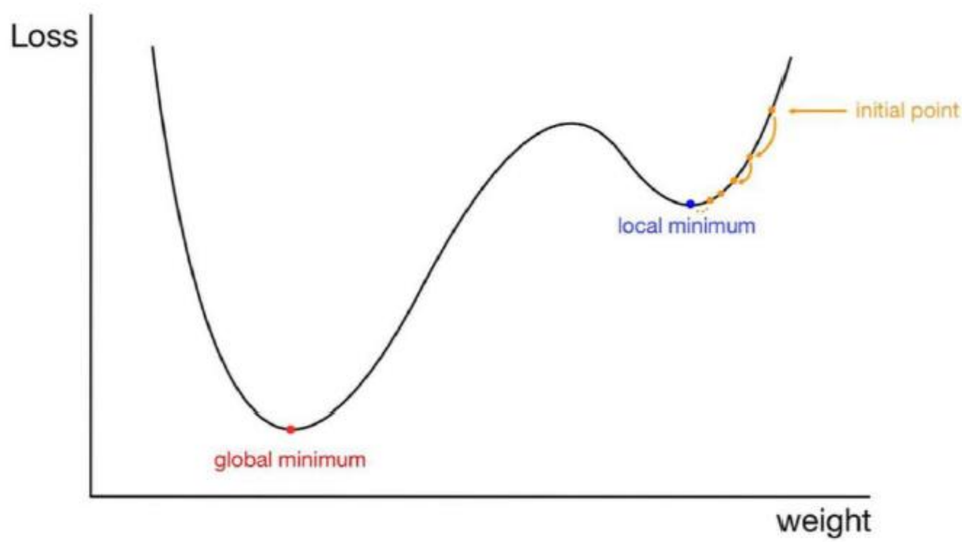
4

Gradient Descent

Gradient Descent의 문제

Local Minimum

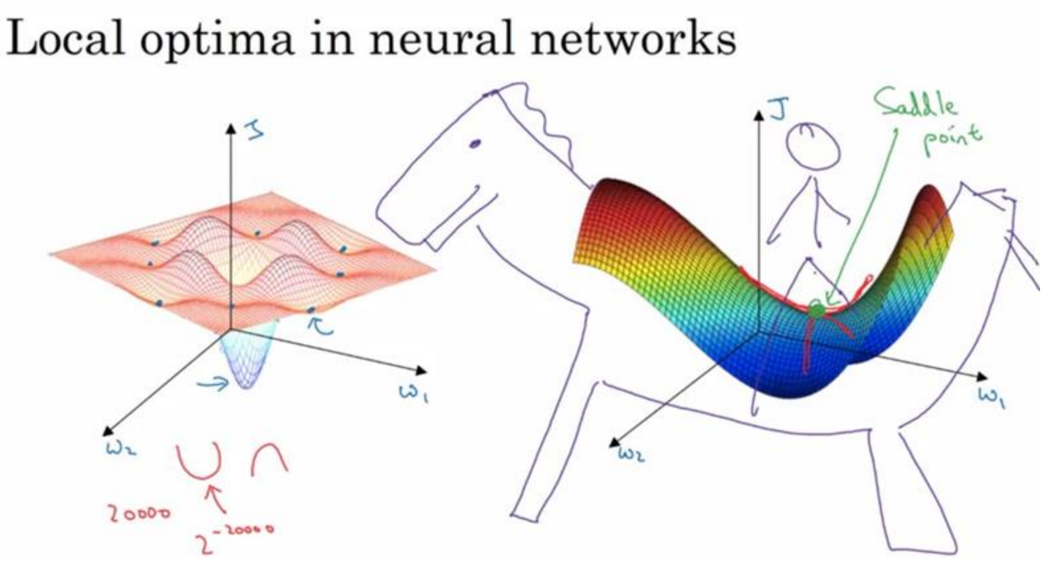
Local minimum을 찾아 수렴하게 되어 Global minimum을 찾지 못하는 문제



Gradient Descent의 문제

Saddle Point

극값이 아닌 **안장점**에 수렴하여 학습이 종료



Gradient Descent

기존 Gradient Descent의 문제점

모든 데이터를 통과해서
업데이트를 진행

느린 속도

Random성 부족으로 인한
Local Minima로
최적해를 결정할 가능성 높음

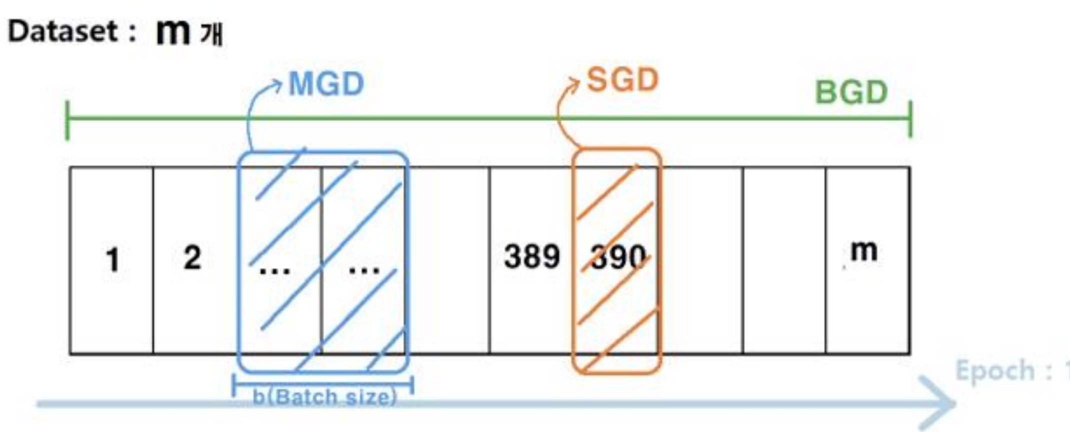
Mini Batch을
이용하여 SGD 진행

Stochastic(Mini Batch) Gradient Descent

Stochastic Gradient Descent (SGD)

데이터 전체(Full Batch)가 아니라

무작위로 추출한 데이터의 일부만을 사용하는 방법



Stochastic(Mini Batch) Gradient Descent

Stochastic Gradient Descent (SGD)

데이터 전체(Full Batch)가 아니라

무작위로 추출한 데이터의 일부만을 사용하는 방법

SGD : 반복적으로 **하나의 Sample에** 대해 무작위 추출 후 최적화를 수행

↓

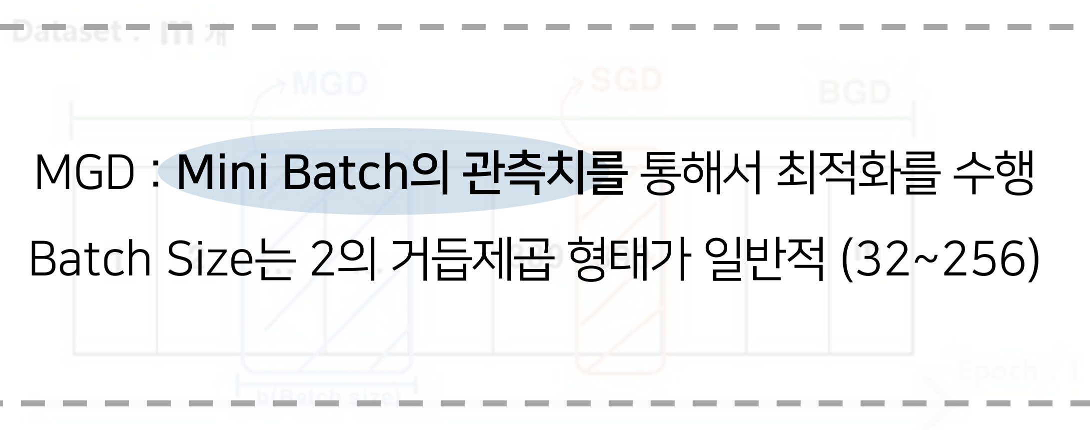
캐시적인 문제와 몇몇 알고리즘에서의 오작동 문제

Stochastic(Mini Batch) Gradient Descent

Stochastic Gradient Descent (SGD)

데이터 전체(Full Batch)가 아니라

무작위로 추출한 데이터의 일부만을 사용하는 방법



MGD : **Mini Batch의 관측치**를 통해서 최적화를 수행

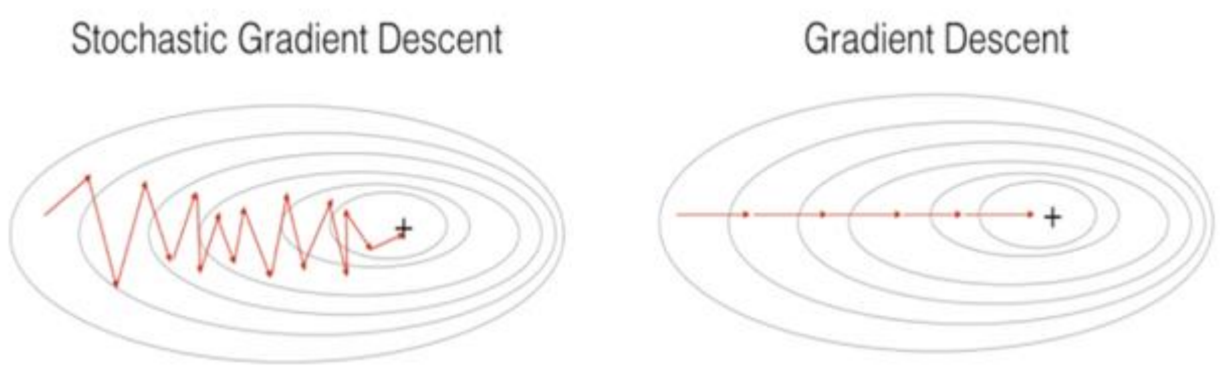
Batch Size는 2의 거듭제곱 형태가 일반적 (32~256)

Stochastic(Mini Batch) Gradient Descent

Stochastic Gradient Descent (SGD)

노이즈에 의해 Local Minimum에서 탈출할 수 있음

일부 데이터를 사용한 만큼 속도 및 메모리 측면에서 이점



Optimizers | Momentum

Momentum

$$v_t = \gamma v_{t-1} + \eta \nabla f(x_{t-1})$$

$$x_t = x_{t-1} - v_t$$

v_t: t번째 time stop에서의 x의 이동벡터

γ: 관성계수. 일반적으로 0.9로 설정

η: 학습률

4 Gradient Descent

Optimizers | Momentum

Momentum

$$v_t = \gamma v_{t-1} + \eta \nabla f(x_{t-1})$$

$$x_t = x_{t-1} - v_t$$

Local minima와 Saddle point 문제를 해결하기 위해,

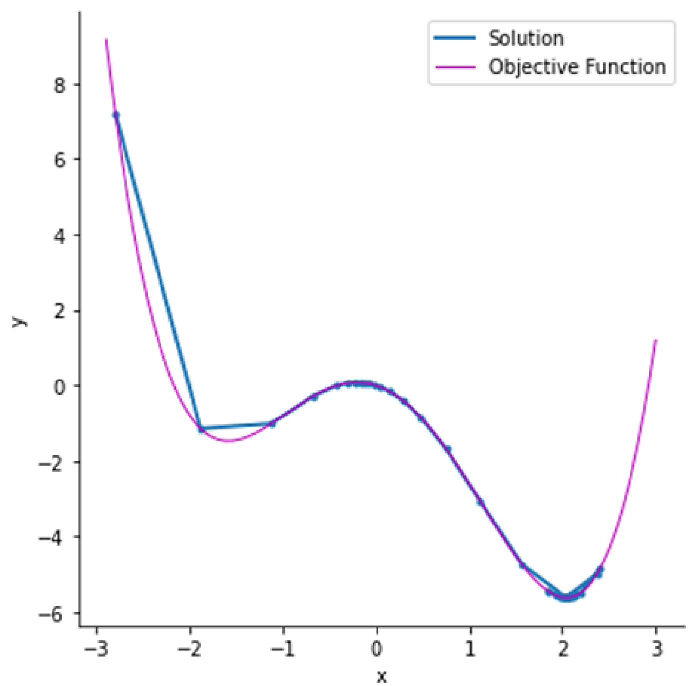
단순한 Gradient Descent 대신 복잡한 Optimizer 사용

학습 과정에서 **기울기 정보**에 **관성**을 추가하고

가중 평균 개념을 통해 **가속효과**를 내는 방식

Optimizers | Momentum

Momentum



가중 평균 개념을 통해 가속효과를 내는 방식

반복 학습을 통해 Local minimum 탈출 가능!

Optimizers | Nesterov Momentum

Nesterov Momentum

$$v_t = \gamma v_{t-1} + \eta \nabla f(x_{t-1} - \gamma v_{t-1})$$

$$x_t = x_{t-1} - v_t$$

→ **Momentum만큼 미리 간 후** Gradient를 구하는 방식

단순 Momentum보다 **공격적인** 방식!

Optimizers | Nesterov Momentum

Nesterov Momentum

$$v_t = \gamma v_{t-1} + \eta \nabla f(x_{t-1} - \gamma v_{t-1})$$

$$x_t = x_{t-1} - v_t$$

v_t : t번째 time stop에서의 x의 이동벡터

γ : 관성계수. 일반적으로 0.9로 설정

η: 학습률

Optimizers | Nesterov Momentum

Nesterov Momentum

$$v_t = \gamma v_{t-1} + \eta \nabla f(x_{t-1} - \gamma v_{t-1})$$

$$x_t = x_{t-1} - v_t$$

Momentum 에서 확장된 형태이므로 변수 동일!

Optimizers | Nesterov Momentum

Nesterov Momentum

$$v_t = \gamma v_{t-1} + \eta \nabla f(x_{t-1} - \gamma v_{t-1})$$

$$x_t = x_{t-1} - v_t$$

☝ 정확한 방향으로 수렴 가능

✌ 최적해를 지나칠 가능성을 줄임

Optimizers | Nesterov Momentum

Nesterov Momentum

$$v_t = \gamma v_{t-1} + \eta \nabla f(x_{t-1} - \gamma v_{t-1})$$

$$x_t = x_{t-1} - v_t$$

매 업데이트마다 관성만큼 미리 가는

과정을 고려하여 Gradient 계산

비효율성 + 계산량의 증가

정확한 방향으로 수렴 가능

최적해를 지나칠 가능성

Optimizers | Adagrad

Adagrad

$$g_t = g_{t-1} + \left(\nabla f(x_{t-1})\right)^2$$

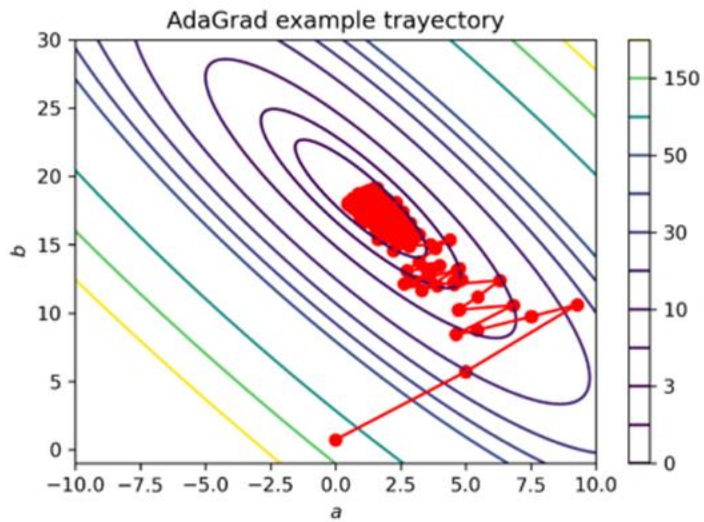
$$x_t = x_{t-1} - \frac{\eta}{\sqrt{g_t + \epsilon}} \cdot \nabla f(x_{t-1})$$

g_t : t번째 time step까지 기울기 누적합

ϵ: 수치적 안정성을 위한 작은 값으로 분모가 0이 되는 것을 방지

η: 학습률(learning rate)

Optimizers | Adagrad



각 Feature들의 **Gradient 크기를 기반**으로 학습률을 조정

학습률을 기울기 누적합으로 나누어 계산

→ 학습이 많이 된 변수일수록 **학습률 감소**

Optimizers | Adagrad

Adagrad의 문제점

분모값인 기울기 누적합이 학습이 진행될수록 커지기 때문에

결국에는 학습률이 0으로 수렴하여 **더 이상 학습 진행 X**

기울기 업데이트 과정에서 지수이동평균 추가!

Optimizers | RMSprop

RMSProp

$$g_t = \gamma g_{t-1} + (1-\gamma)\left(\nabla f(x_{t-1})\right)^2$$

$$x_t = x_{t-1} - \frac{\eta}{\sqrt{g_t + \epsilon}} \cdot \nabla f(x_{t-1})$$

g_t: t번째 time step까지 기울기 누적합

ϵ: 수치적 안정성을 위한 작은 값으로 분모가 0이 되는 것을 방지

η: 학습률(learning rate)

γ: 지수이동평균의 업데이트 계수

Optimizers | RMSprop

RMSProp

$$g_t = \gamma g_{t-1} + (1-\gamma)\left(\nabla f(x_{t-1})\right)^2$$

$$x_t = x_{t-1} - \frac{\eta}{\sqrt{g_t + \epsilon}} \cdot \nabla f(x_{t-1})$$

Adagrad에서 지수이동평균 계수가 추가된 형태

g_t가 무한히 커지는 것을 방지함으로써

AdaGrad의 한계 극복

g_t : t번째 time step까지 기울기 누적합

ϵ: 수치적 안정성을 위한 작은 값으로 분모가 0이 되는 것을 방지

η: 학습률(learning rate)

γ : 지수이동평균의 업데이트 계수

Optimizers | Adam

Adam

$$m_t = \beta_1 m_{t-1} + (1-\beta_1)\nabla f(x_{t-1})$$

$$g_t = \beta_2 g_{t-1} + (1-\beta_2)\left(\nabla f(x_{t-1})\right)^2$$

$$\widehat{m_t} = \frac{m_t}{1-\beta_1^t} \quad , \quad \widehat{g_t} = \frac{g_t}{1-\beta_2^t}$$

$$x_t = x_{t-1} - \frac{\eta}{\sqrt{\widehat{g_t}} + \epsilon} \cdot \widehat{m_t}$$

가장 널리 이용되는 Optimizer

Adam = Momentum(m_t) + RMSProp(g_t)

Optimizers | Adam

Adam

$$m_t = \beta_1 m_{t-1} + (1-\beta_1)\nabla f(x_{t-1})$$

$$g_t = \beta_2 g_{t-1} + (1-\beta_2)\left(\nabla f(x_{t-1})\right)^2$$

β_1 : 1차 모멘트의 지수이동평균 감쇠율. 보통 0.9로 설정

β_2 : 2차 모멘트의 지수이동평균의 감쇠율. 보통 0.999로 설정

Optimizers | Adam

Adam

$$\widehat{m_t} = \frac{m_t}{1-\beta_1^t} \quad , \quad \widehat{g_t} = \frac{g_t}{1-\beta_2^t}$$

$$x_t = x_{t-1} - \frac{\eta}{\sqrt{\widehat{g_t} + \epsilon}} \cdot \widehat{m_t}$$

ϵ, η : RMSProp과 동일한 지표

$\widehat{m_t}, \widehat{g_t}$ = 학습 초기시 m_t, g_t가 0이 되는 것을 방지하기 위한 보정 값

→ 편향 보정을 통해 보다 효율적인 학습 가능

다음주 예고

1. 이미지 데이터

2. CNN

3. CNN 모델

4. 객체 탐지와 이미지 분할

감사합니다