딥러닝팀

CV팀
강철석
박상훈
박채원
방건우
이현진

INDEX

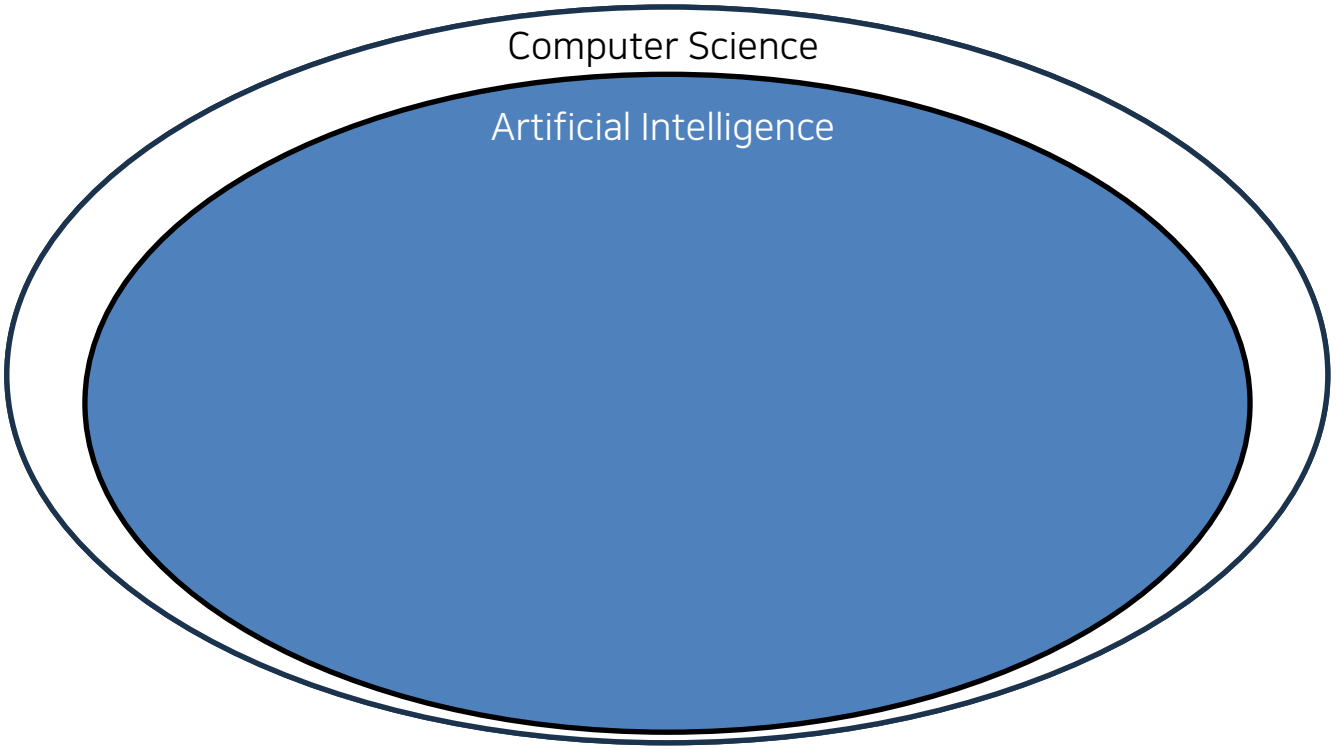
1

머신러닝과 딥러닝

인공지능

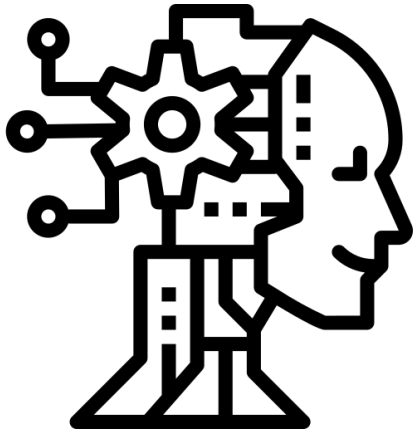
인공지능 *Artificial Intelligence*

기계가 사람처럼 **지능**이 필요한 일을 하도록 만드는 과학



인공지능 – 강인공지능과 약인공지능

강인공지능



사람처럼 생각하고 행동하는 인공지능

약인공지능

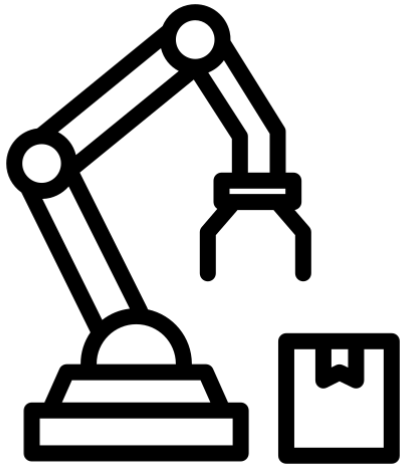
한 task에 집중하여 수행하는 인공지능

인공지능 – 강인공지능과 약인공지능

강인공지능

사람처럼 생각하고 행동하는 인공지능

약인공지능

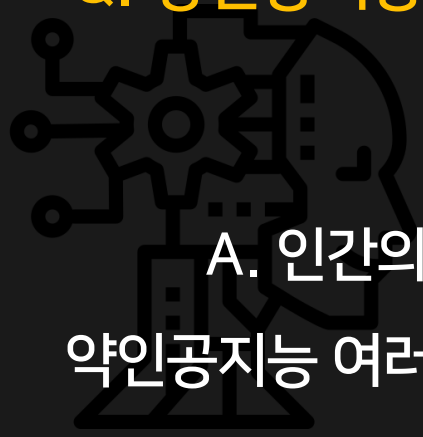


한 task에 집중하여 수행하는 인공지능

인공지능 – 강인공지능과 약인공지능

강인공지능 | 약인공지능

Q. 강인공지능은 우리랑 상관없는 것 아닌가요?



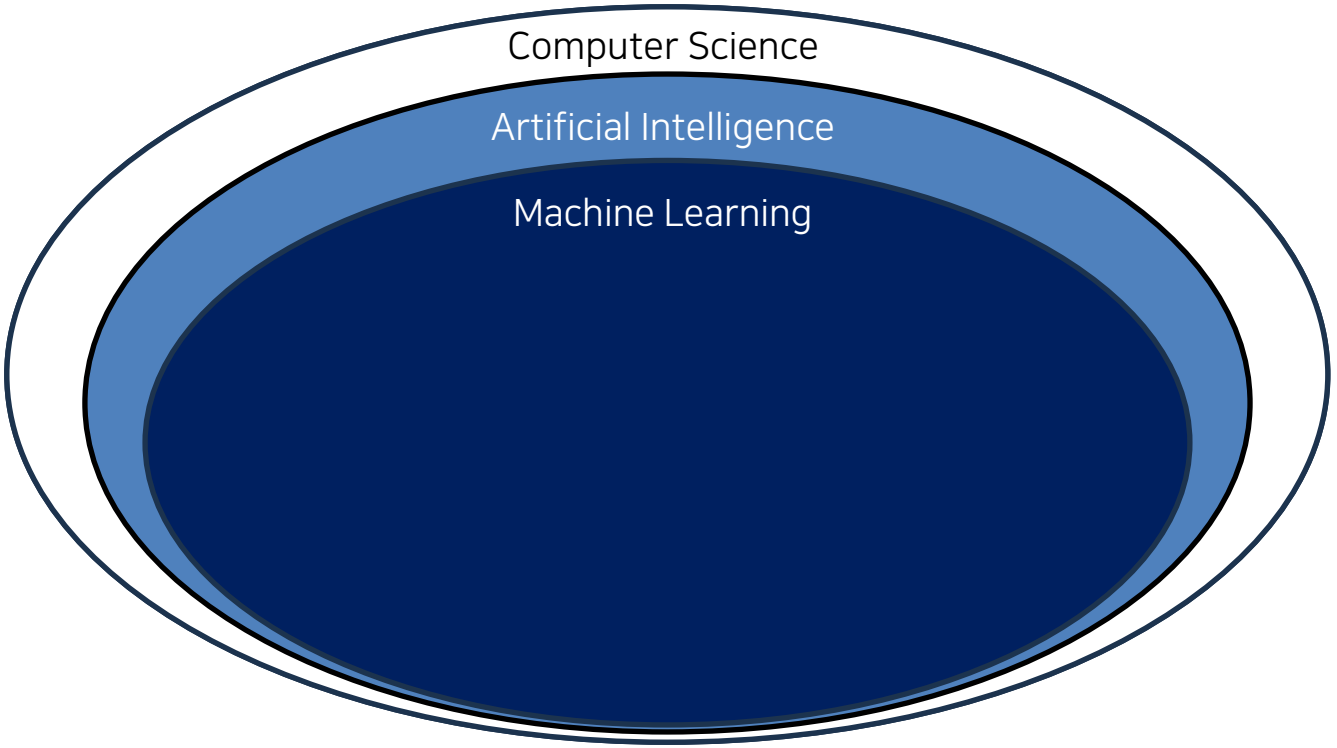
A. 인간의 지능을 모방하려는 시도에서
약인공지능 여러 분야의 좋은 알고리즘들이 나왔음!

사람처럼 생각하고 행동하는 인공지능 | 한 task에 집중하여 수행하는 약인공지능

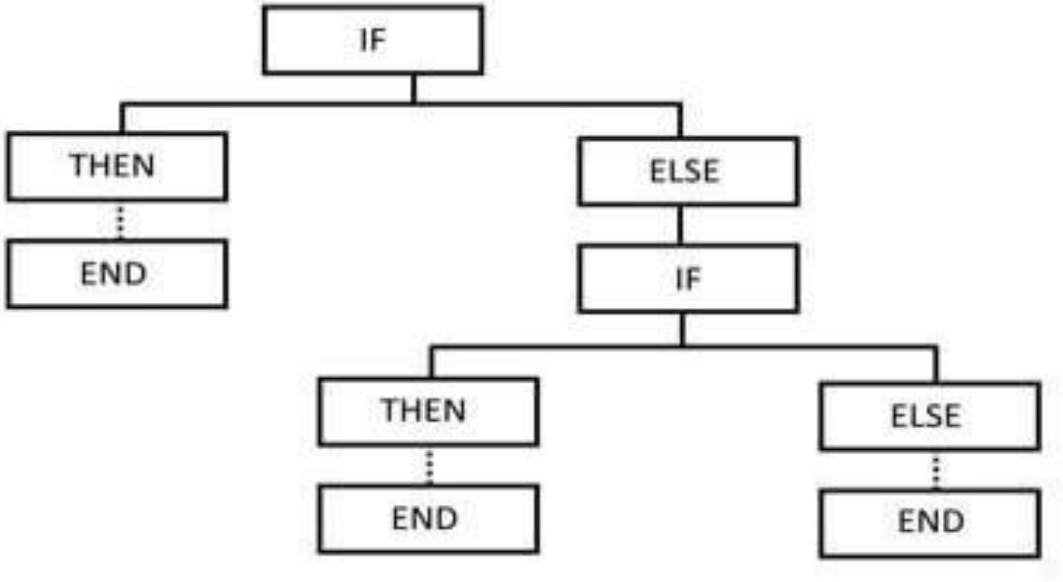
머신러닝

머신러닝 *Machine Learning*

인공지능의 하위집합으로 컴퓨터가 **데이터를 통해 스스로 학습**하는 것



머신러닝

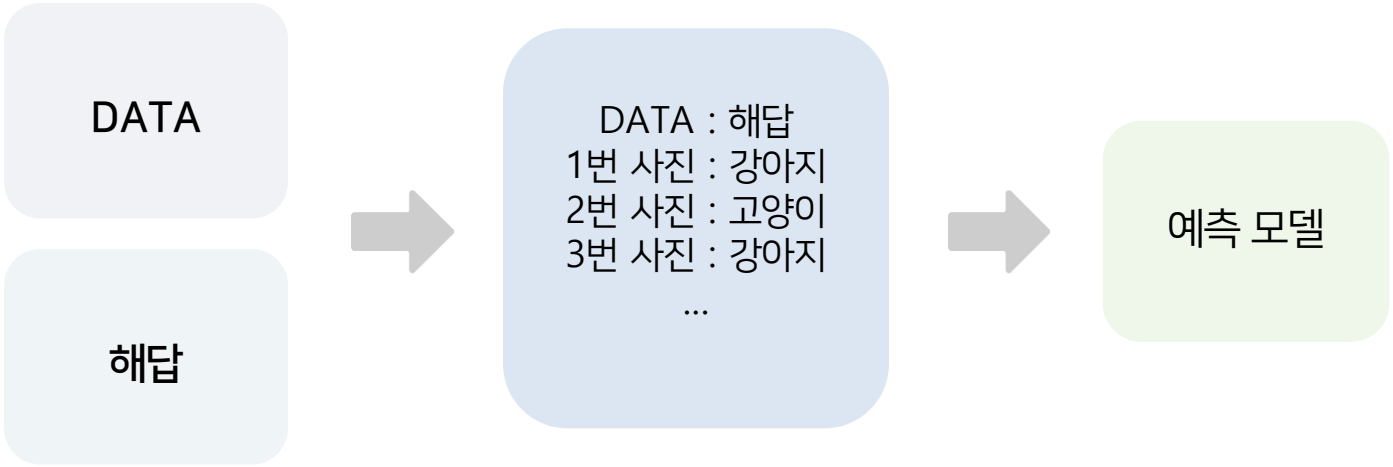


규칙 기반 시스템의 경우 if-else를 연속적으로 나열한 것이 대표적임.

그러나 **데이터의 다양성**이 증가한다면 구현이 불가능해질 정도로 복잡해짐.

1 머신러닝과 딥러닝

머신러닝

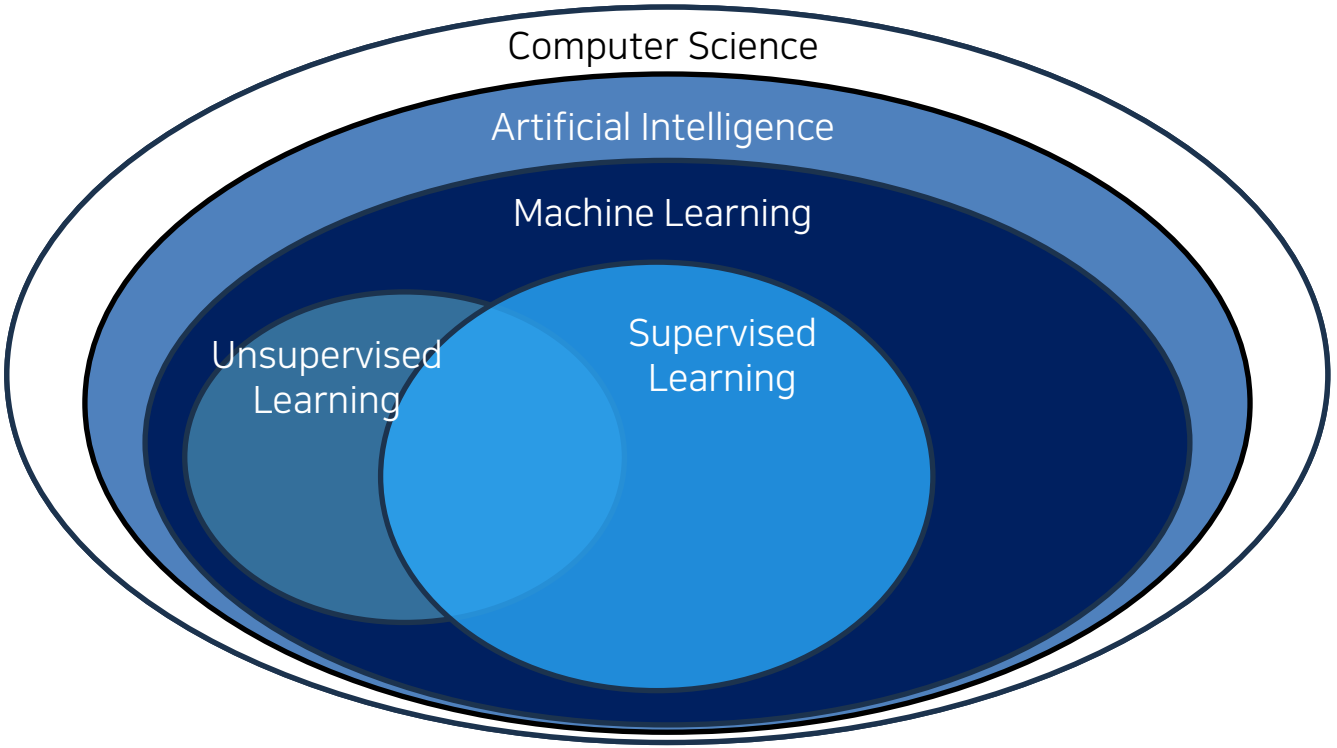


하지만, 머신러닝은 학습 과정에서 **데이터를 통해 스스로 규칙을 찾고**
새로운 값이 들어오면 자신이 찾은 규칙을 통해 결과를 예측함.

지도학습과 비지도학습

지도학습 *Supervised Learning*

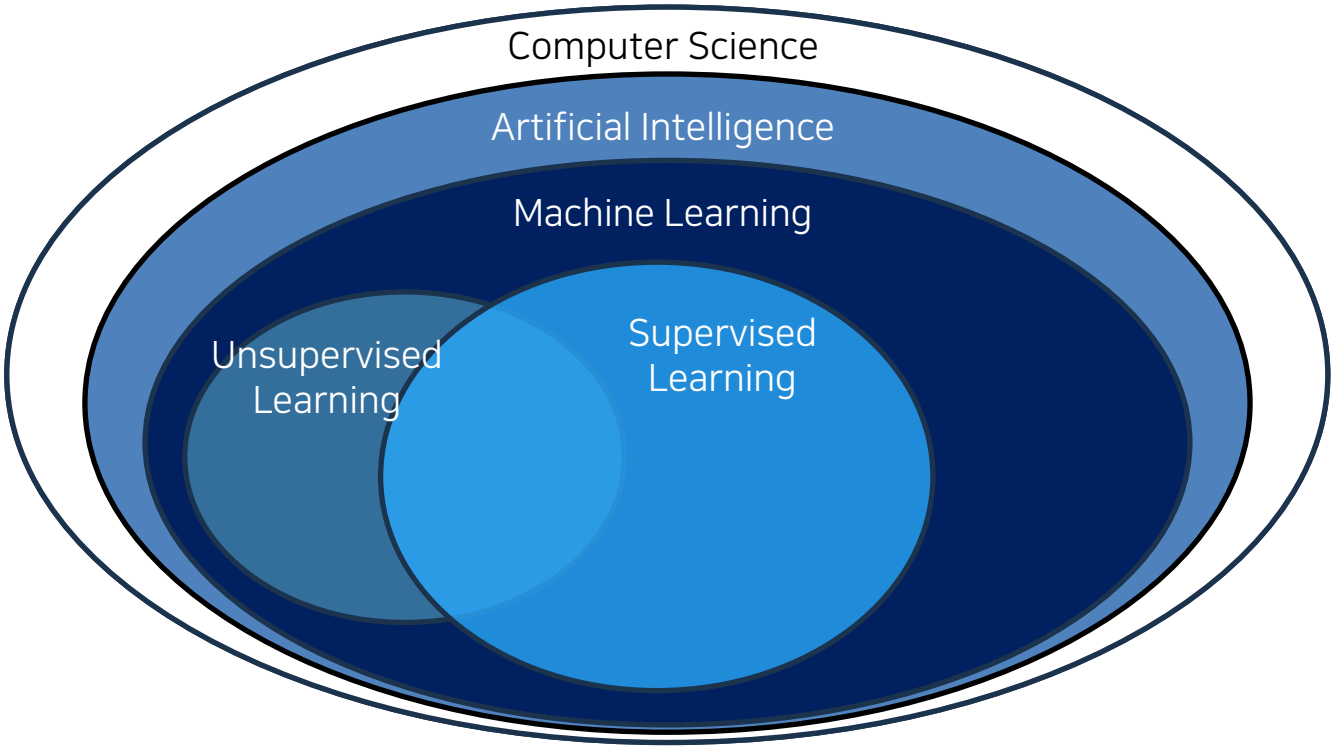
데이터와 레이블이 함께 주어졌을 때 **데이터와 레이블의 관계**를 학습하는 방법



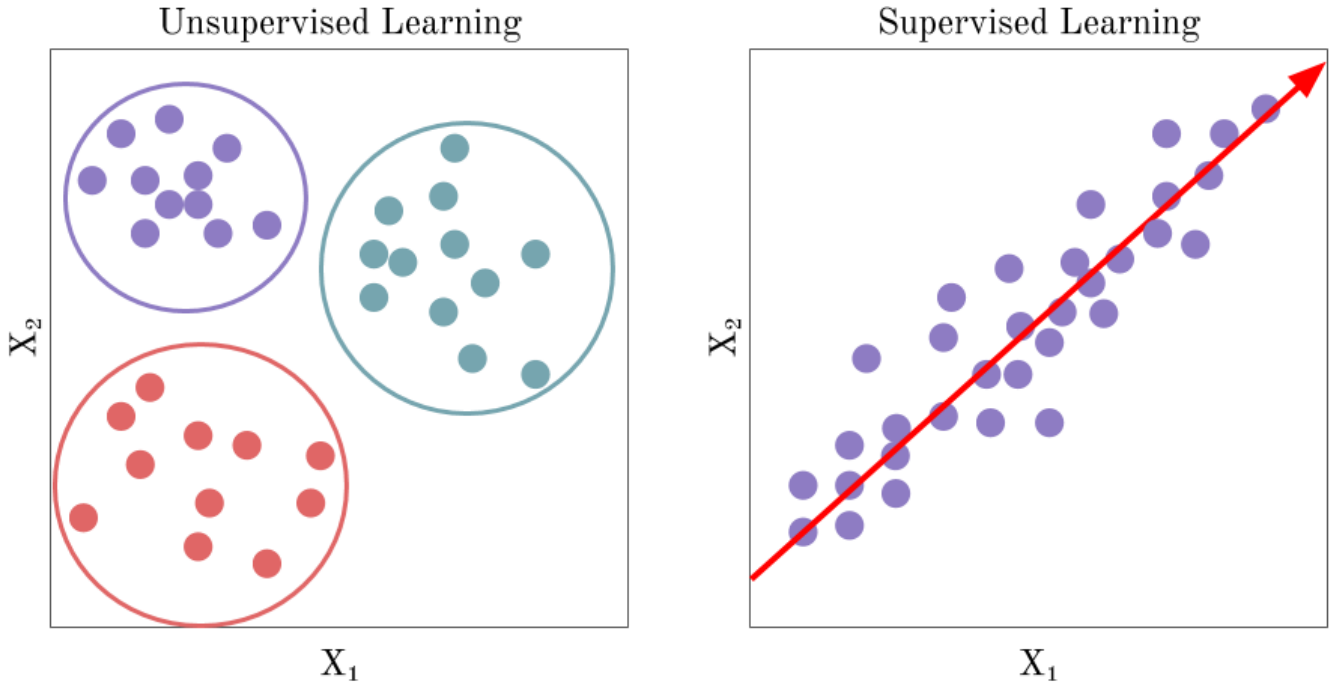
지도학습과 비지도학습

비지도학습 *Unsupervised Learning*

레이블 없이 **데이터만 주어진 경우**, 데이터의 특징, 패턴, 구조 등을 파악하는 방식



지도학습과 비지도학습

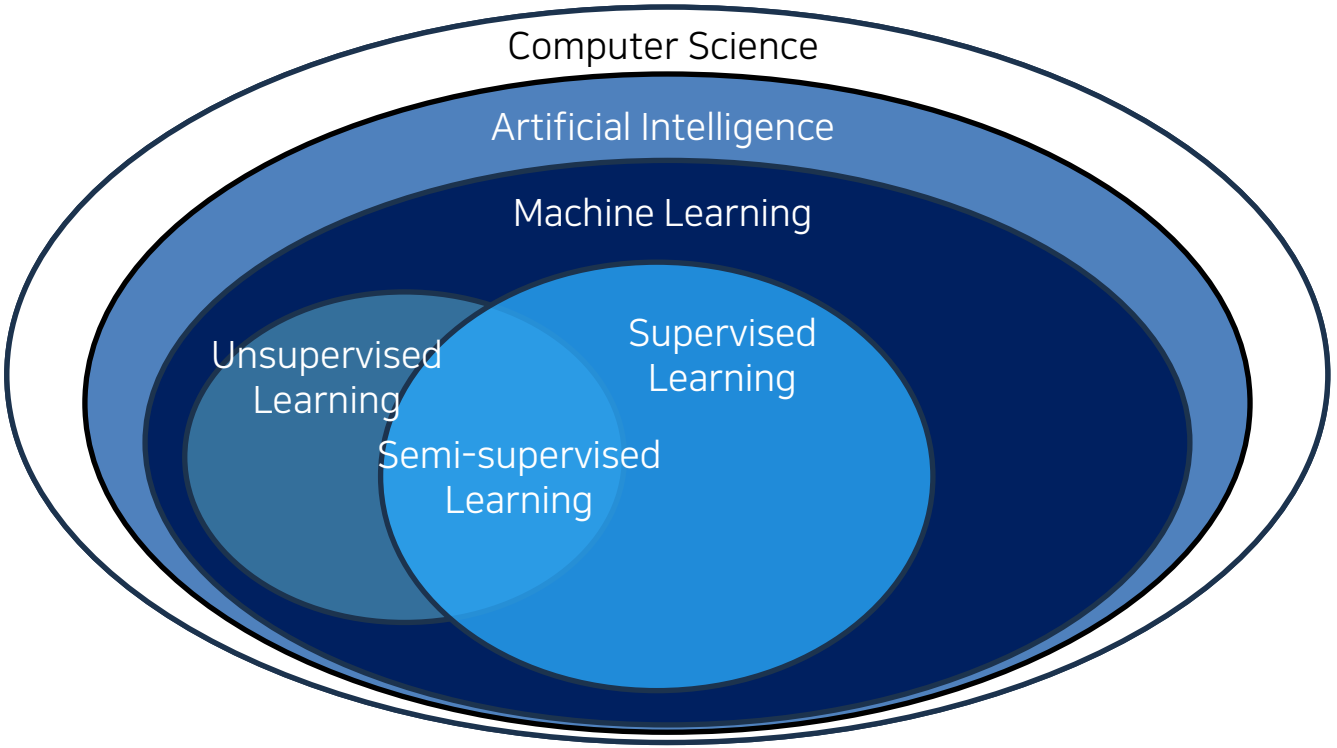


주로 **분류와 회귀** 등은 지도학습, **군집화와 이상치 탐지** 등은 비지도학습을 사용함

지도학습과 비지도학습

준지도학습 *Semi-supervised Learning*

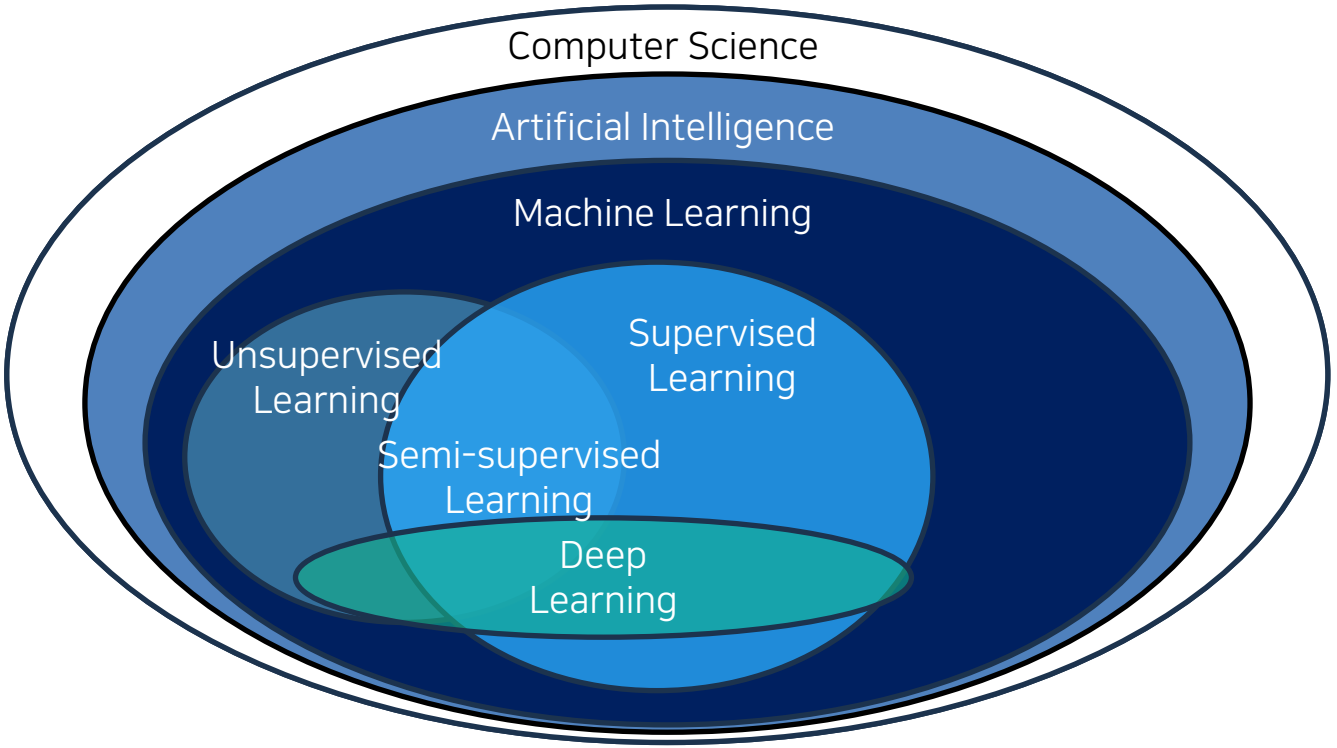
라벨링된 데이터와 그렇지 않은 데이터를 **모두 활용하여 학습**하는 방식



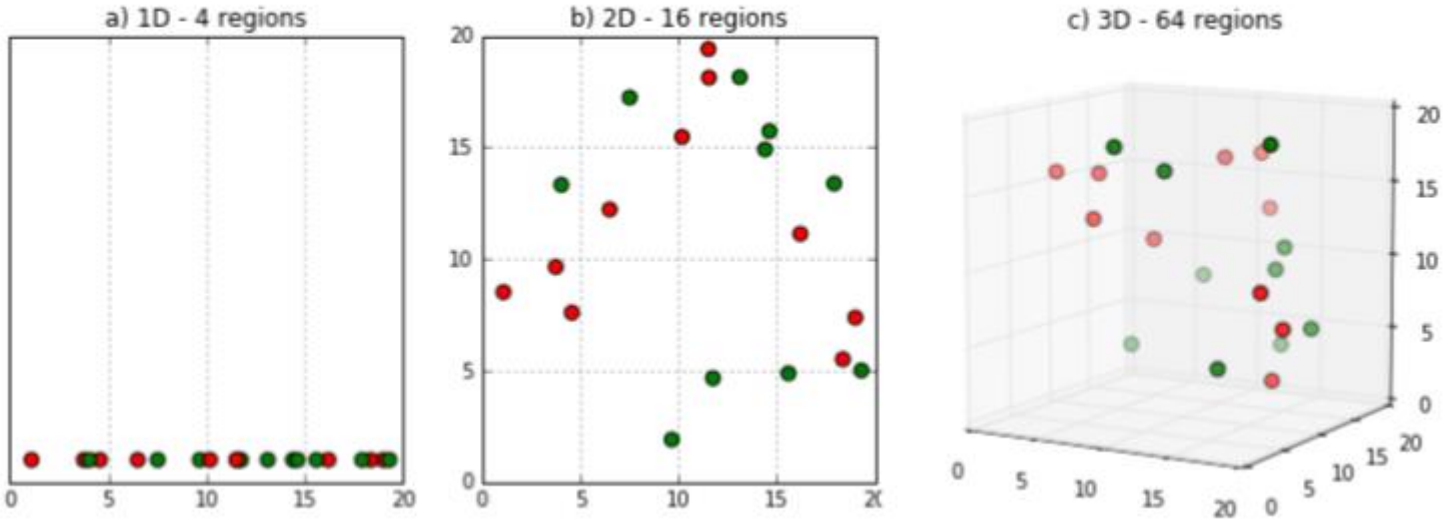
딥러닝

딥러닝 *Deep Learning*

심층 인공신경망(Artificial Neural Network)을 사용하는 머신러닝 기법.



딥러닝의 필요성 ① – 차원의 저주



데이터의 차원이 커지면 특징의 수가 많아지고 **데이터의 밀도가 낮아짐**

밀집한 데이터가 없어 **패턴을 찾기 어려워지고** 전체 공간을 표현할 수 없음

딥러닝의 필요성 ① – 차원의 저주

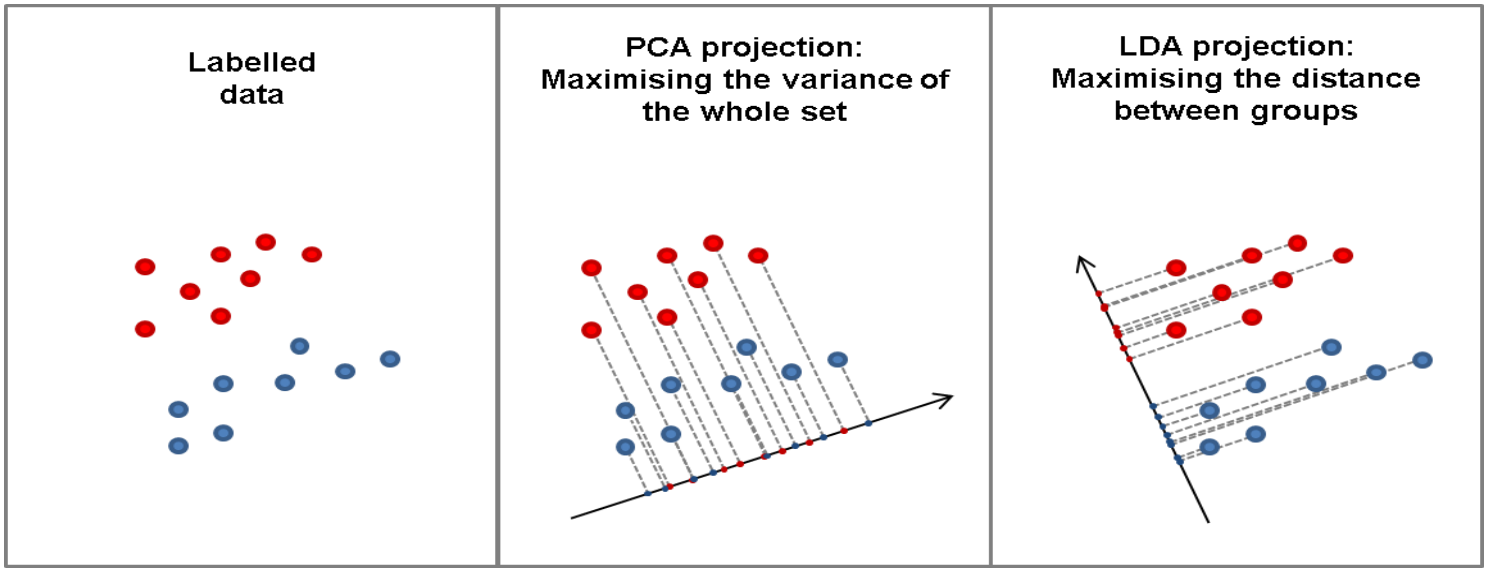
결국 모델은 불필요한 특징들을 과하게 학습하여 **과대적합**으로 이어질 수 있음

데이터의 차원이 커지면 [illegible] 데이터의 밀도가 낮아짐

밀집한 데이터가 없어 패턴을 [illegible] 전체 공간을 표현할 수 없음

딥러닝의 필요성 ① – 차원의 저주

Ex) LDA



LDA를 예시로 보면, 축을 줄였을 때 설명력이 높아지는 현상을 발견할 수 있음

딥러닝의 필요성 ① – 차원의 저주

차원의 저주를 해결하기 위해 머신러닝은 Feature Selection 과정이 필요

↓

자동 특징 추출 *Automatic Feature Extraction*

사람이 특별히 지도하지 않아도 훈련에 사용된 데이터에서
자율적으로 판별에 사용될 **적합한 특징**을 찾아내는 것

차원의 저주 문제 해결!

딥러닝의 필요성 ① – 차원의 저주

차원의 저주를 해결하기 위해 머신러닝은 **Feature Selection 과정이 필요**

↓

자동 특징 추출 ***Automatic Feature Extraction***

사람이 특별히 지도하지 않아도 훈련에 사용된 데이터에서

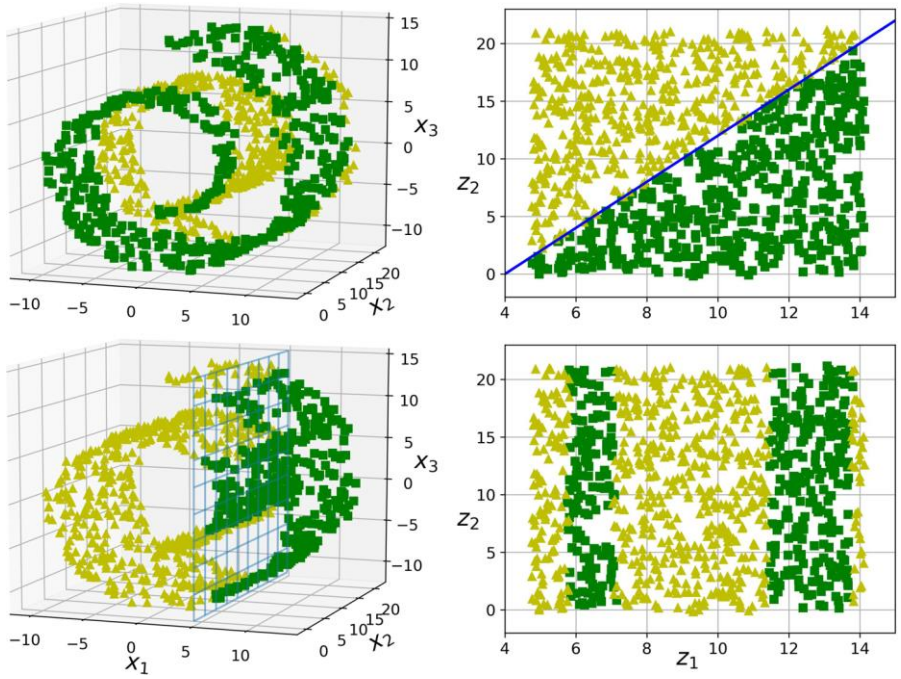
자율적으로 판별에 사용될 **적합한 특징**을 찾아내는 것

차원의 저주 문제 해결!

딥러닝의 필요성 ② – 다양체 가설

다양체 가설 *Manifold Hypothesis*

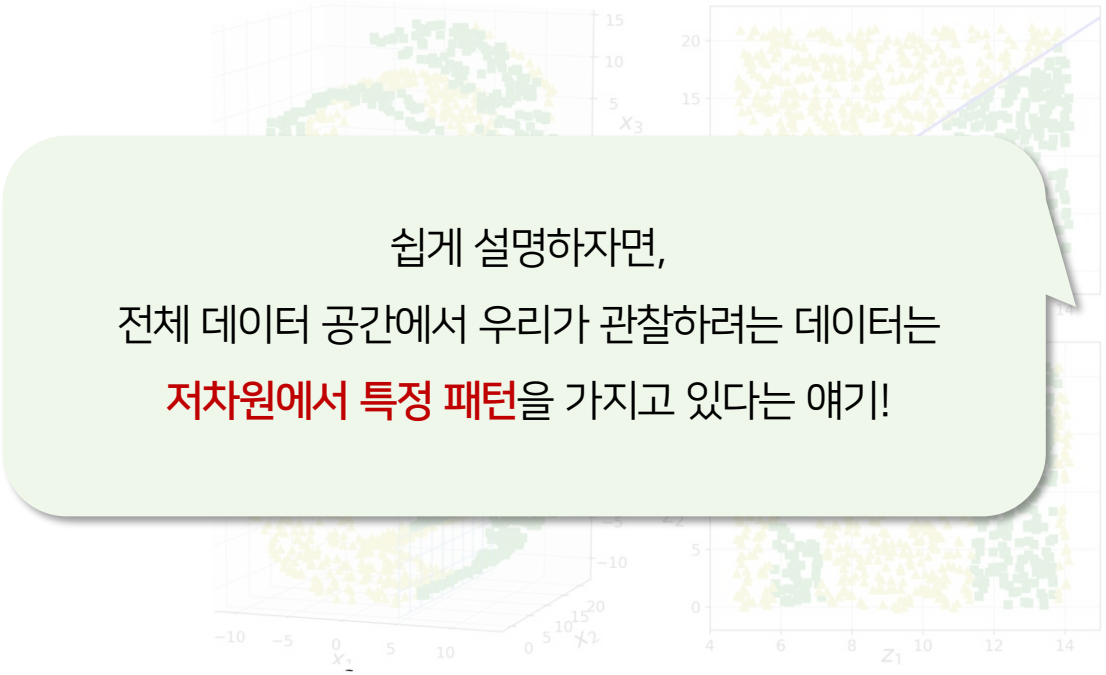
실세계의 고차원 데이터가 고차원에 놓여있는 **저차원의 잠재 다양체**들로 구성된다는 가설



딥러닝의 필요성 ② – 다양체 가설

다양체 가설 *Manifold Hypothesis*

실세계의 고차원 데이터가 고차원에 놓여있는 **저차원의 잠재 다양체**들로 구성된다는 가설



쉽게 설명하자면,
전체 데이터 공간에서 우리가 관찰하려는 데이터는
저차원에서 특정 패턴을 가지고 있다는 얘기!

딥러닝의 필요성 ② – 다양체 가설

Ex) 이미지 데이터

RGB 픽셀 값을 랜덤하게 뽑아오면 노이즈 이미지가 나옴

딥러닝의 필요성 ② – 다양체 가설

Ex) 이미지 데이터

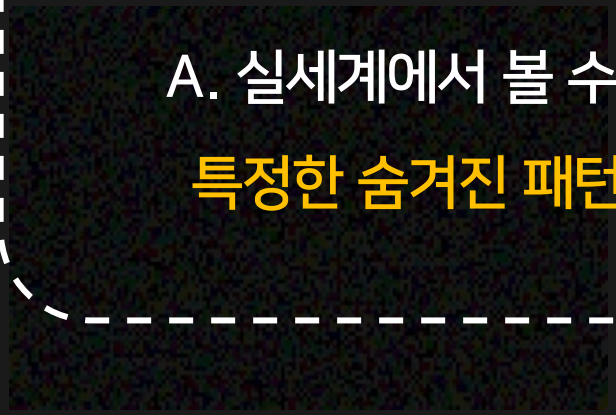
오른쪽 고양이 이미지가 다양체 가설을 만족함

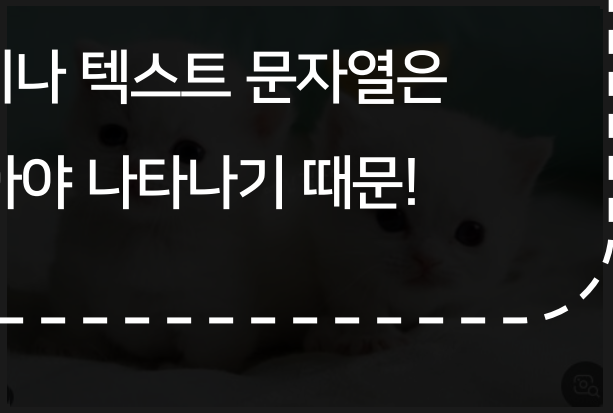
1 머신러닝과 딥러닝

딥러닝의 필요성 ② – 다양체 가설

Q. 왜 고양이 이미지가 다양체 가설을 만족할까?

A. 실세계에서 볼 수 있는 이미지나 텍스트 문자열은
특정한 숨겨진 패턴을 따라 뽑아야 나타나기 때문!





그냥 랜덤하게 뽑아오면 전혀 의미 없는 데이터가 나옴

Ex) 자연어의 문장 구조를 고려하지 않은 경우

기차 학교 돌아간다 그리고 사람 → 의미 없는 문장

딥러닝의 필요성 ② – 다양체 가설

Ex) 이미지 데이터

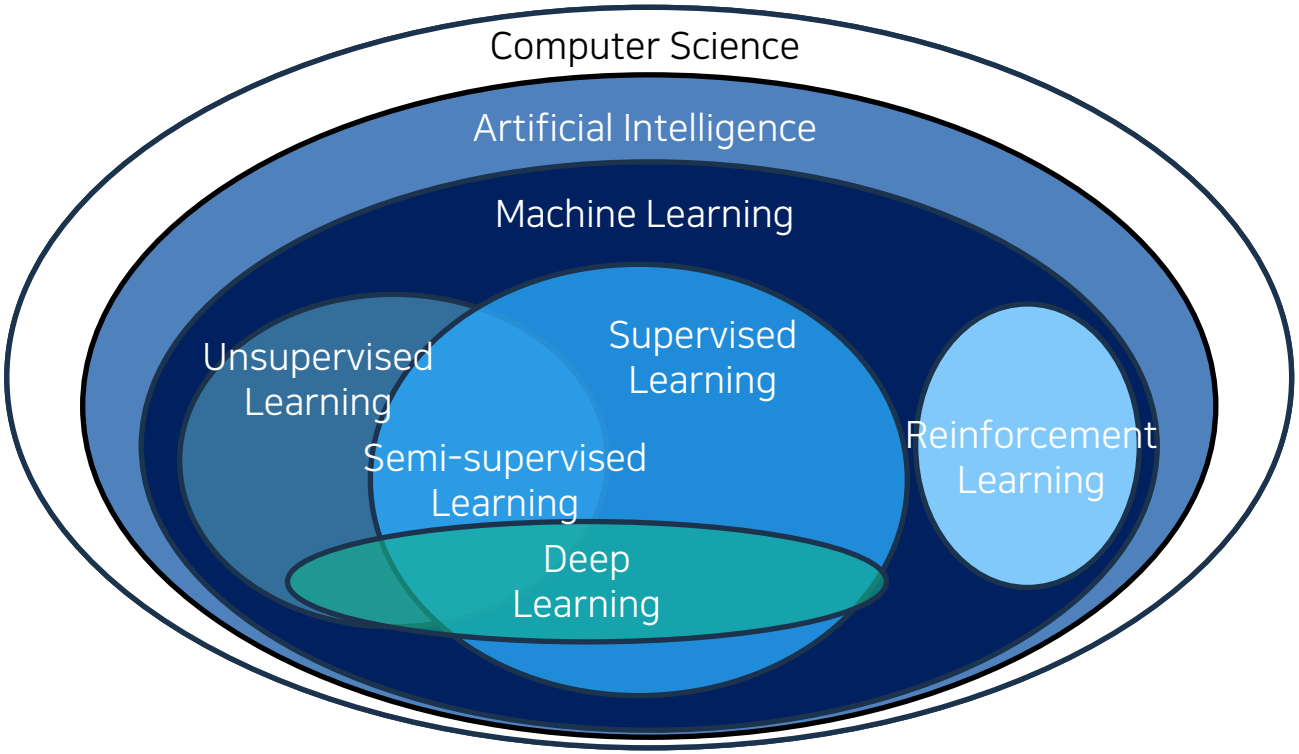
기존 머신러닝은 이런 다양체를 잘 잡아내지 못하지만
딥러닝은 **효과적으로 패턴을 잡아냄**!

오른쪽 고양이 이미지가 다양체 가설을 만족함

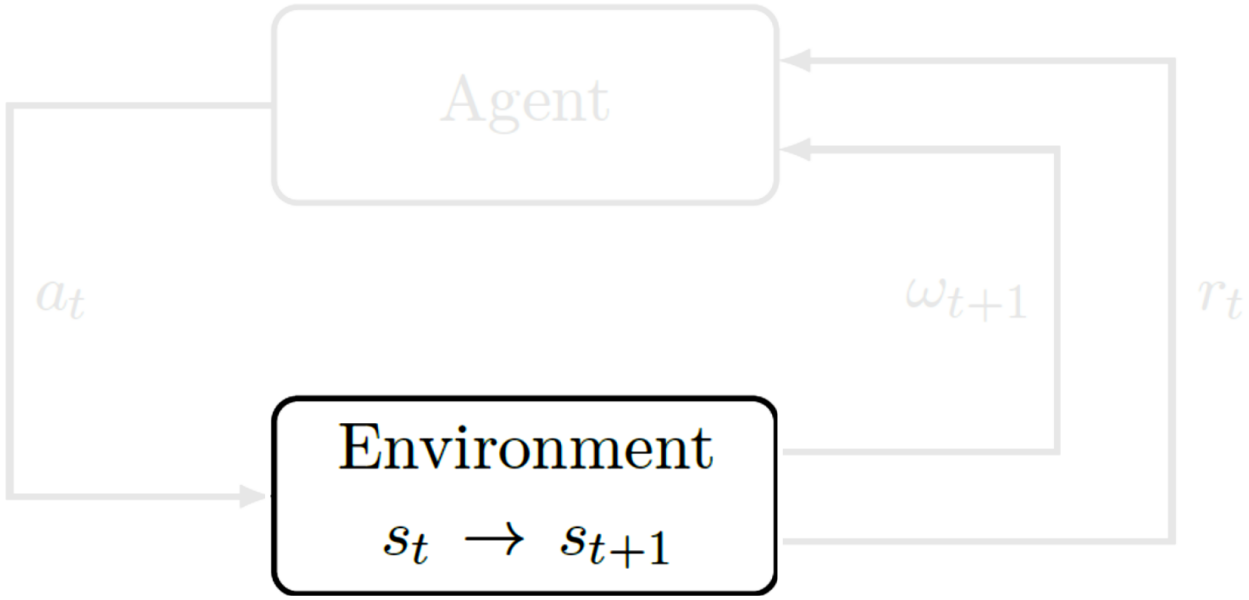
강화학습

강화학습 *Reinforcement Learning*

어떤 **환경** 안에서 정의된 **Agent**가 현재의 **상태**를 인식하여,
선택 가능한 **행동**들 중 **보상**을 최대화하는 **정책**을 선택하는 방법

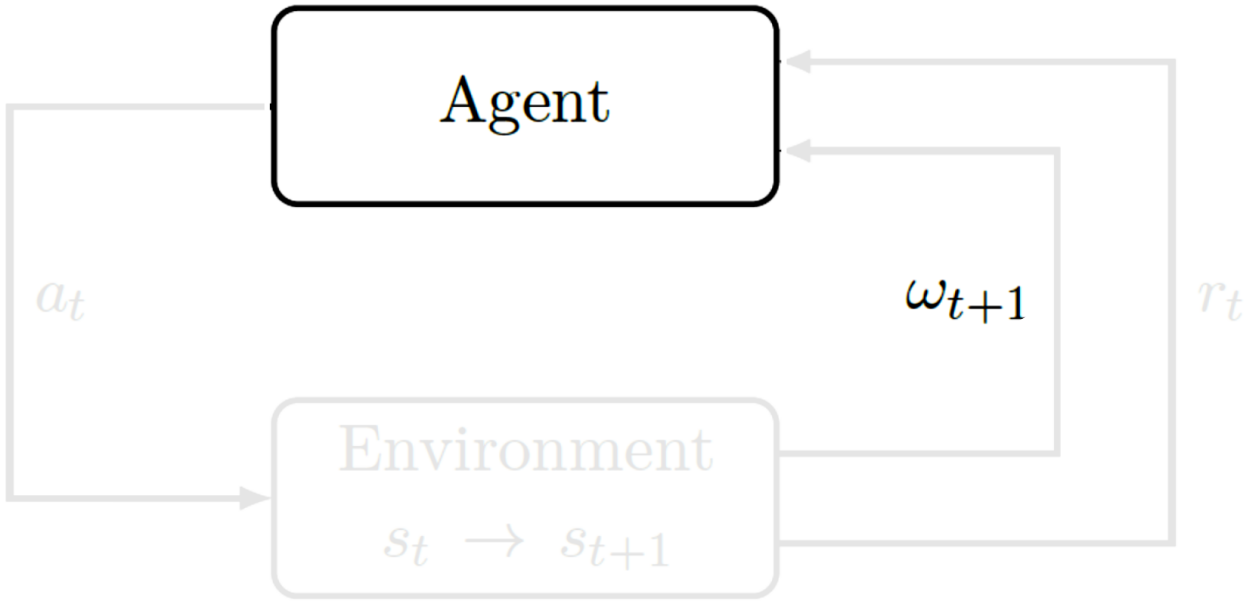


강화학습 - step ①



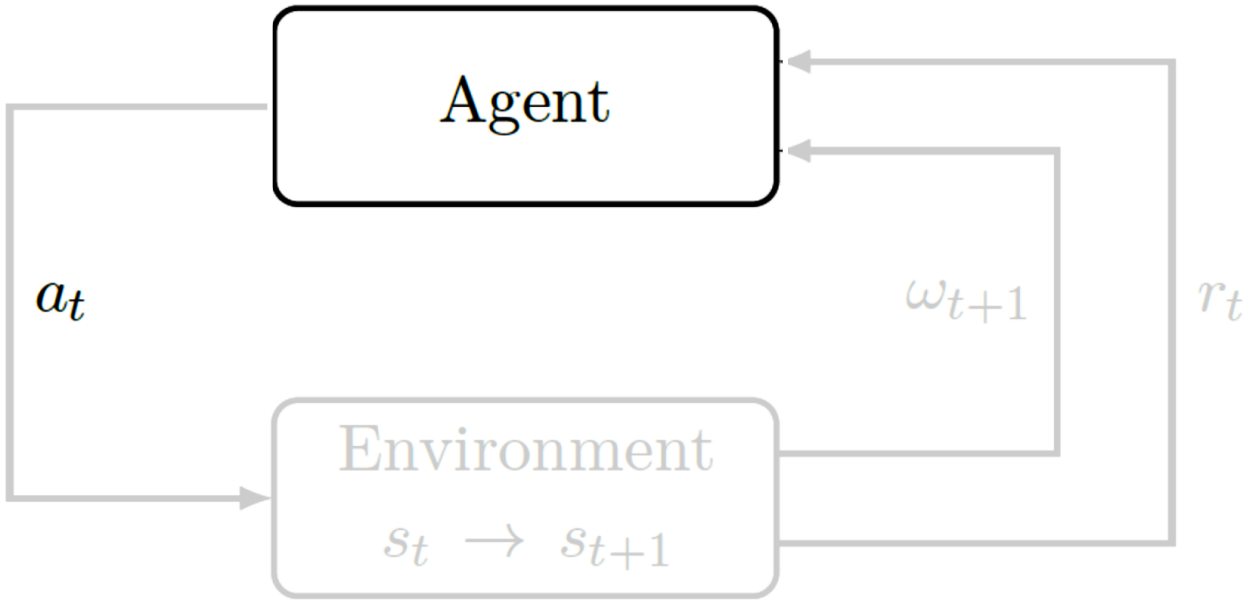
현재 Environment는 s_t

강화학습 - step ②



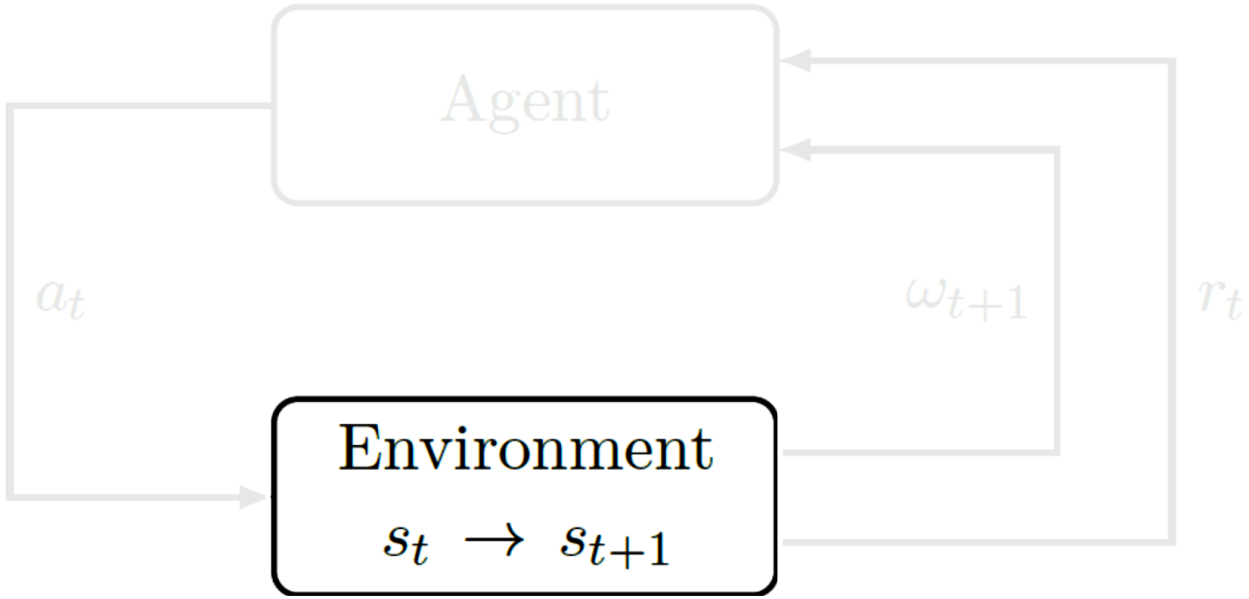
s_t 의 부분적 정보인 Observation ω_{t+1} 가 Agent에게 전달됨

강화학습 - step ③



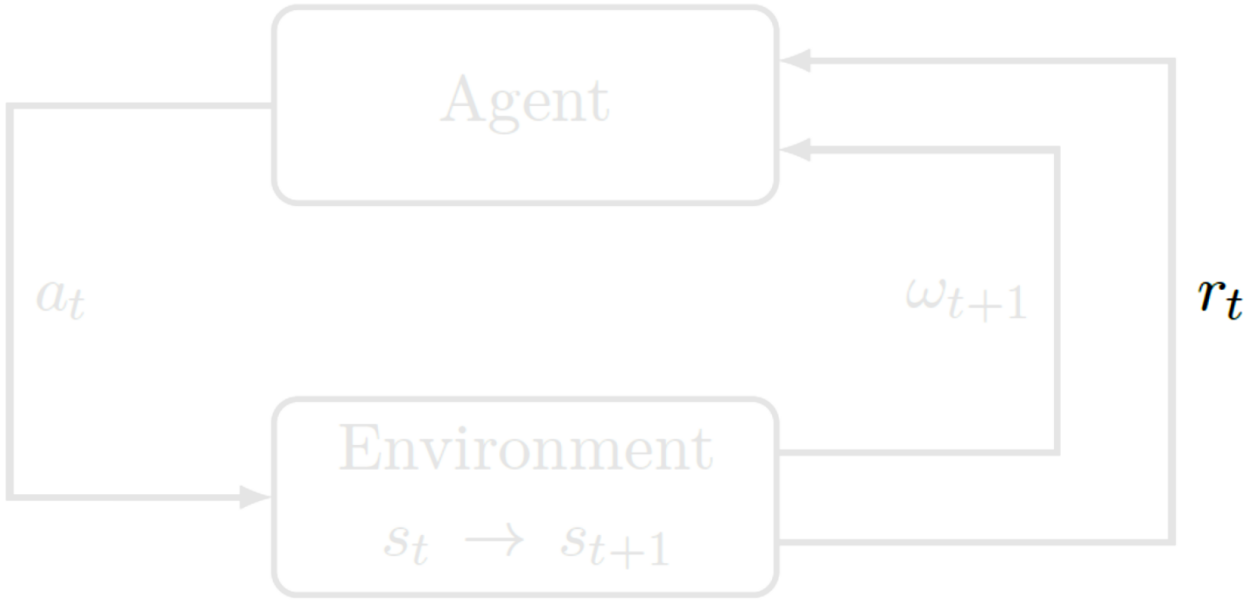
Agent가 action a_t 수행

강화학습 - step ④



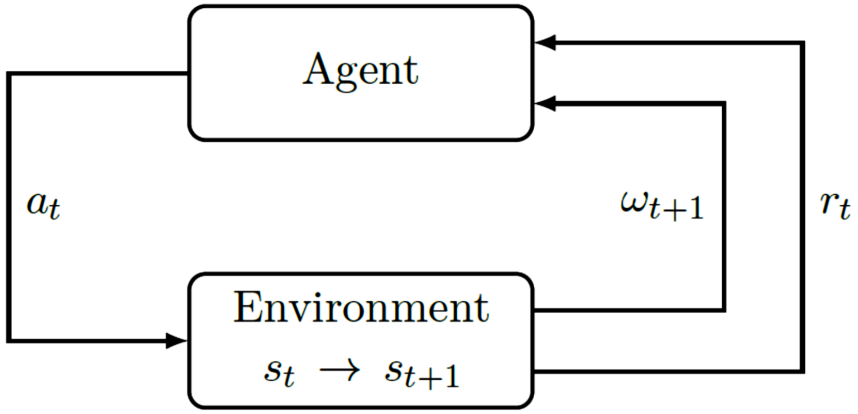
action a_t 에 따라 Environment의 state가 s_{t+1}로 변화

강화학습 - step ⑤



Environment에 따라 보상 r_t가 Agent에게 주어짐

강화학습



단계들을 반복 시행하며 **보상을 최대화**하도록 agent의 **행동을 조절**

데이터가 주어지지 않더라도 특정 환경 내에서

Agent, 행동, 상태, 보상이 잘 정의 된다면 학습이 가능

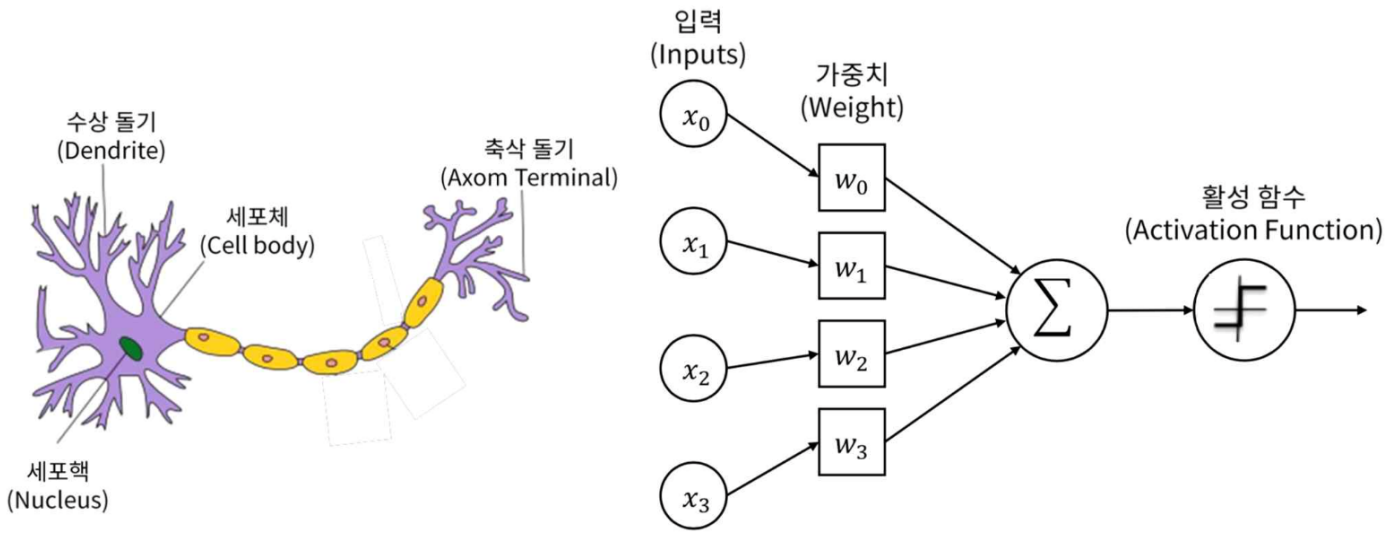
2

퍼셉트론

퍼셉트론

퍼셉트론 *Perceptron*

신경 조직의 기본 단위인 뉴런을 수학적으로 모델링한 것



2 퍼셉트론

퍼셉트론

퍼셉트론 *Perceptron*

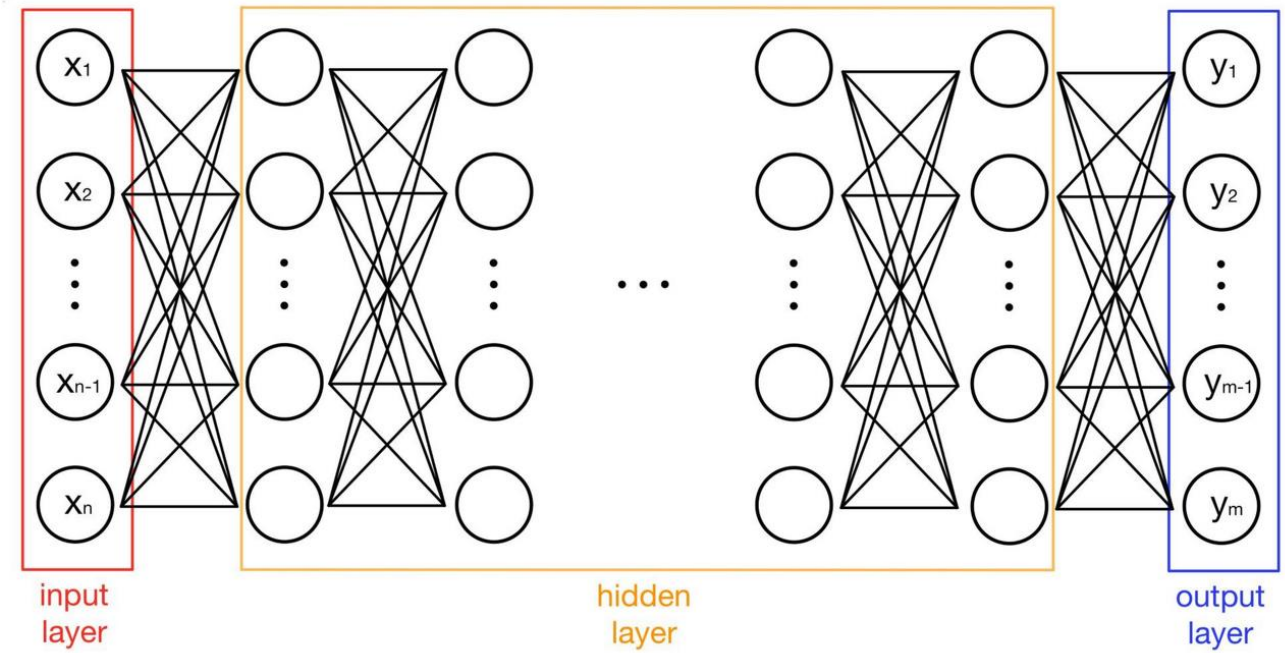
신경 조직의 기본 단위인 뉴런을 수학적으로 모델링한 것

자세한 내용은 리드오프에서 만났으니 생략!

다층 퍼셉트론

다층 퍼셉트론 *Multi Layer Perceptron*

퍼셉트론을 **여러 층**으로 쌓은 순방향의 인공 신경망



2 퍼셉트론

다층 퍼셉트론

다층 퍼셉트론 *Multi Layer Perceptron*

퍼셉트론을 **여러 층**으로 쌓은 순방향의 인공 신경망

Q. 왜 층을 여러 개 쌓아야 할까?

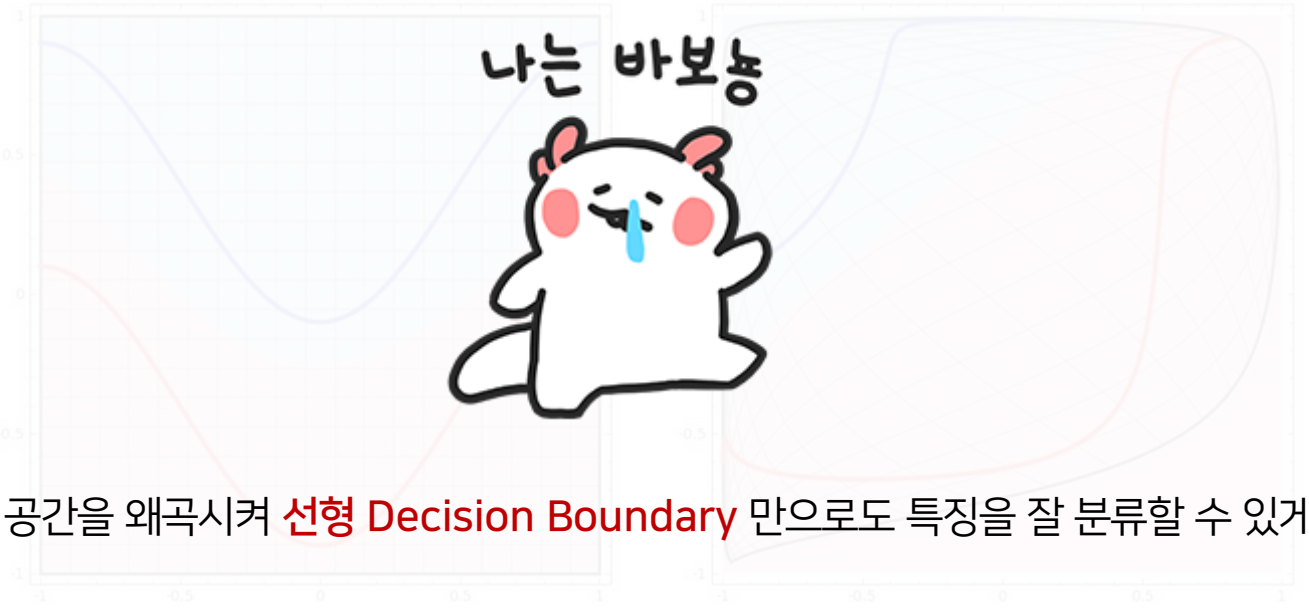
A. **활성화함수**로 **비선형성**을 추가하기 위해서!

2 퍼셉트론

활성화함수

활성화함수 *Activation Function*

퍼셉트론의 출력 값을 결정하는 **비선형(non-linear) 함수**

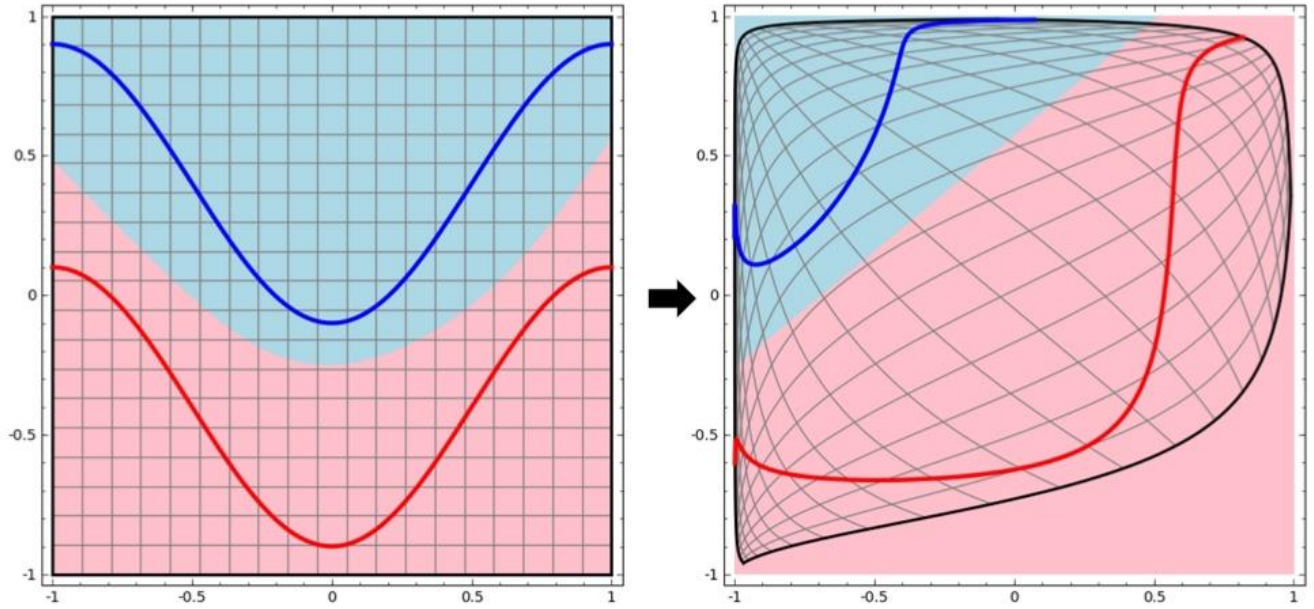


Input 공간을 왜곡시켜 **선형 Decision Boundary** 만으로도 특징을 잘 분류할 수 있게 함

활성화함수

활성화함수 *Activation Function*

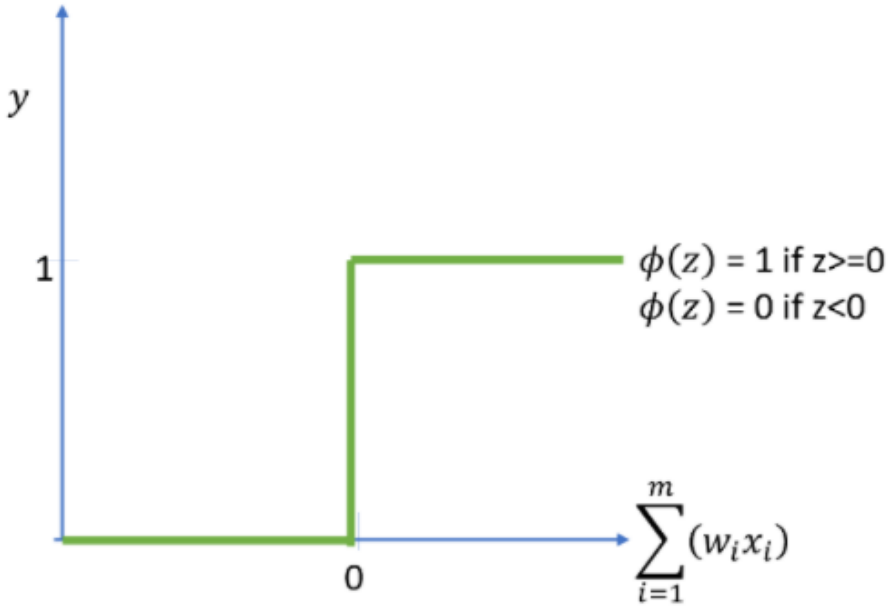
퍼셉트론의 출력 값을 결정하는 **비선형(non-linear) 함수**



활성화함수의 종류

Sign 함수

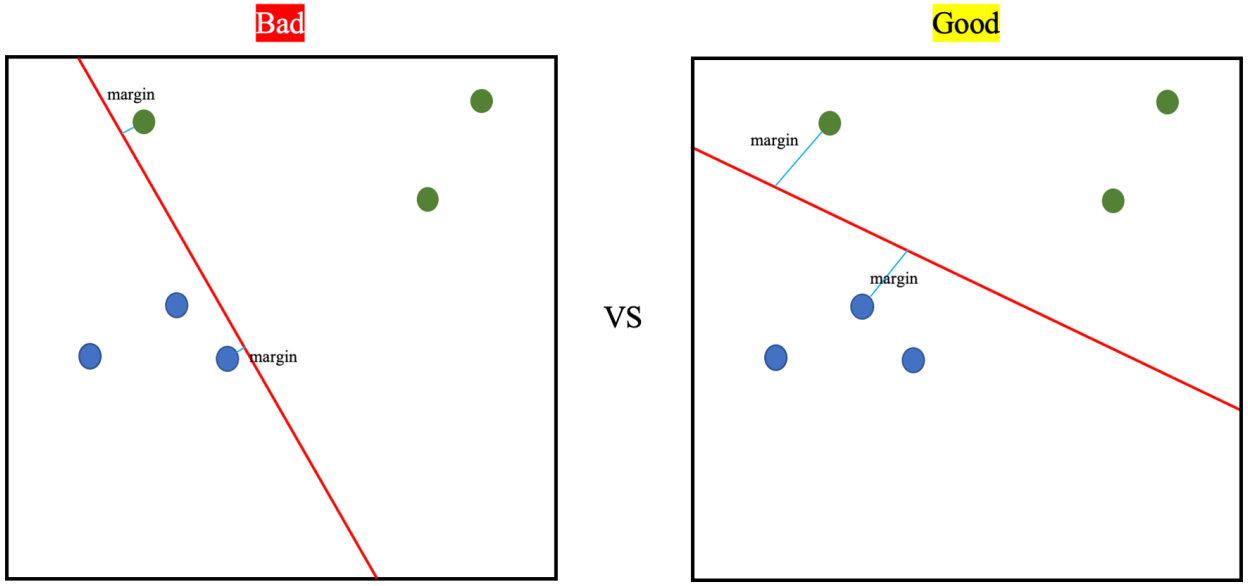
단순히 **부호를 나타내는 함수**로 고전적인 퍼셉트론에서 사용



활성화함수의 종류

Sign 함수의 문제점 ①

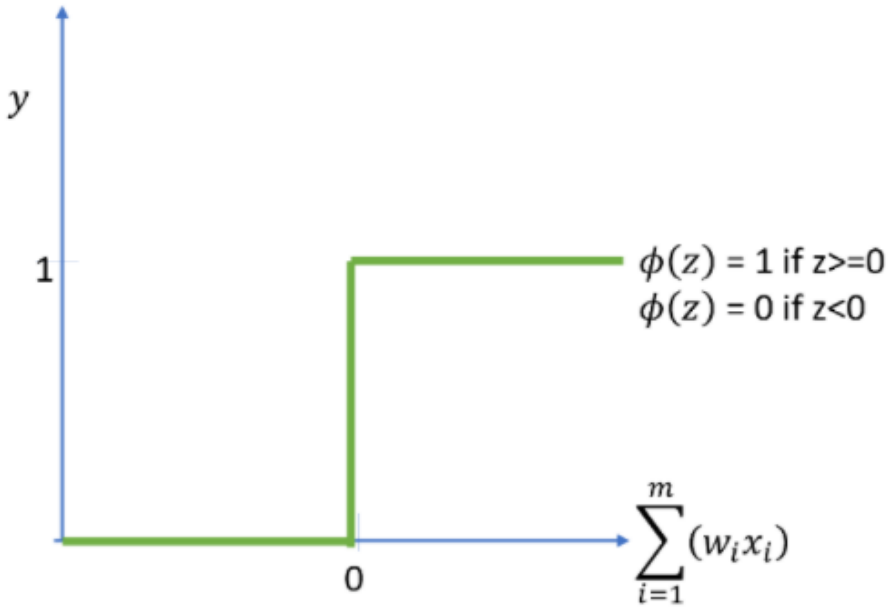
데이터와 Decision Boundary 간 **거리 정보를 고려하지 못함**



활성화함수의 종류

Sign 함수의 문제점 ②

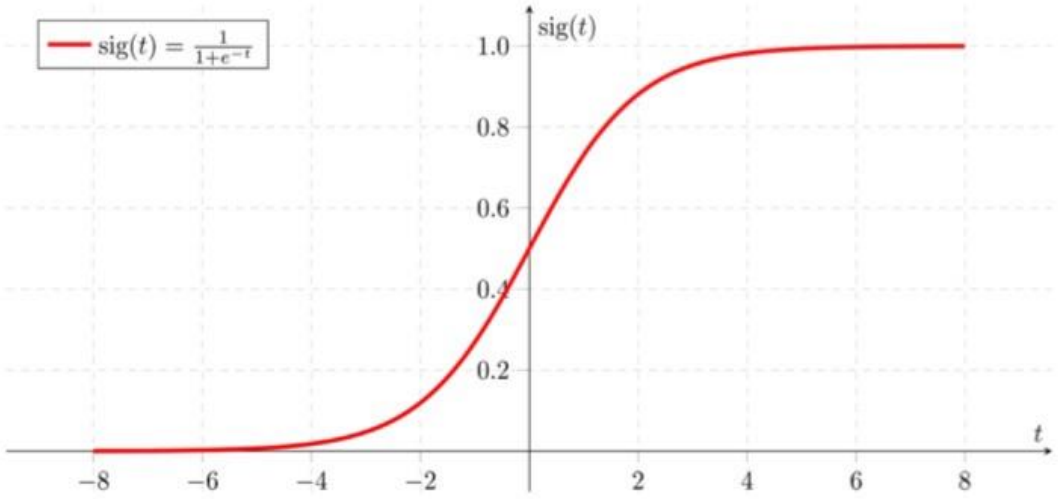
미분이 불가능하거나 값이 0이라서 Gradient Descent를 적용할 수 없음



활성화함수의 종류

Sigmoid 함수

입력이 작으면 **0에 가까운 값**, 크면 **1에 가까운 값** 출력

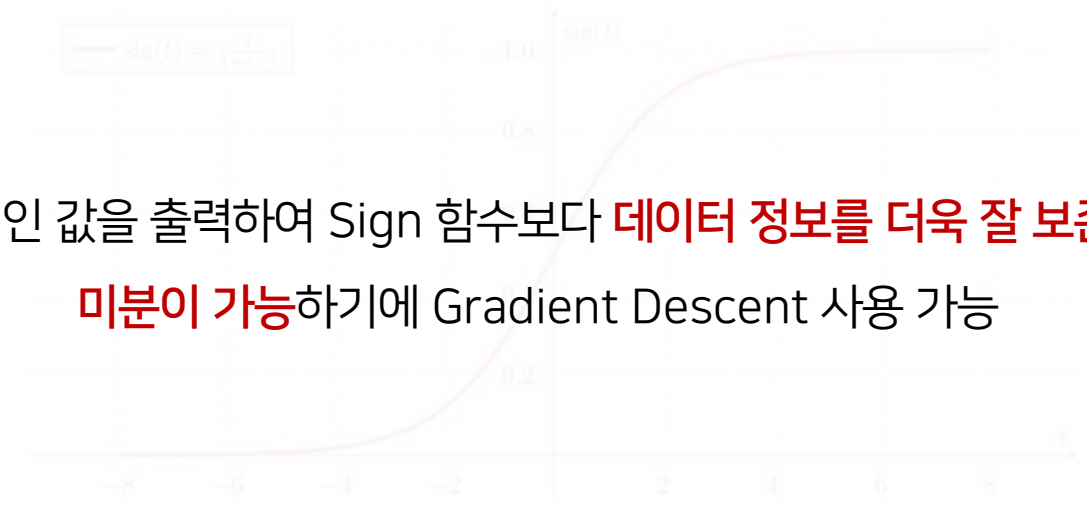


$$\text{sigmoid}(x) = \frac{1}{1+e^{-x}}$$

활성화함수의 종류

Sigmoid 함수

입력이 작으면 **0에 가까운 값**, 크면 **1에 가까운 값** 출력



연속적인 값을 출력하여 Sign 함수보다 **데이터 정보를 더욱 잘 보존**하며

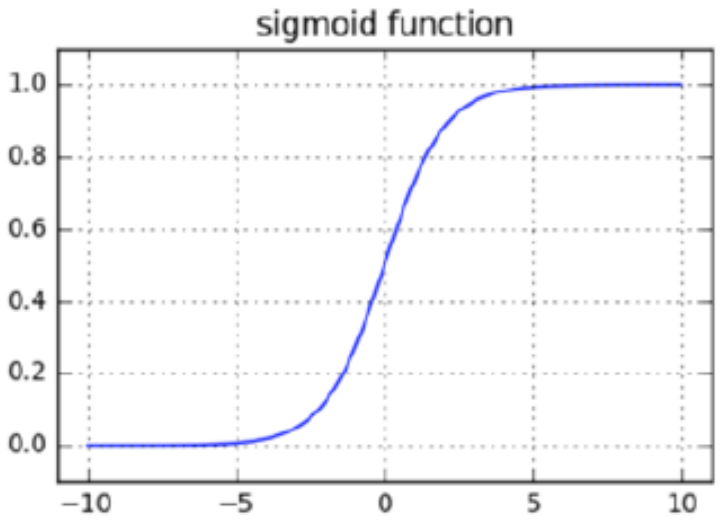
미분이 가능하기에 Gradient Descent 사용 가능

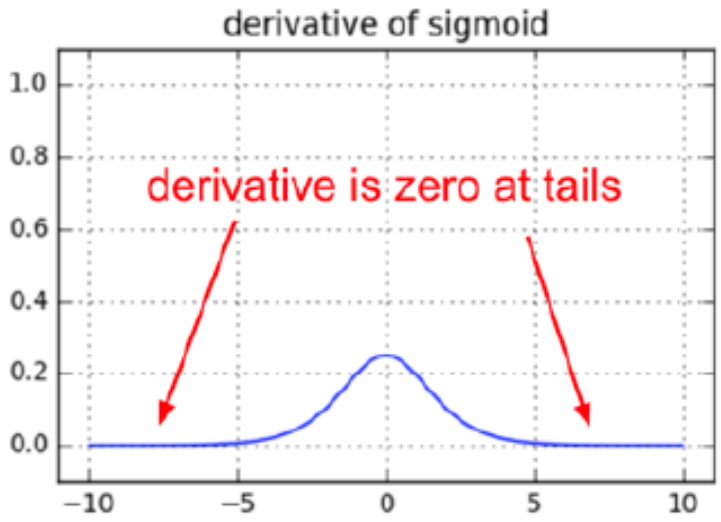
$$\text{sigmoid}(x) = \frac{1}{1+e^{-x}}$$

활성화함수의 종류

Sigmoid 함수의 문제점 ①

Gradient Vanishing 현상이 발생하여 추가적인 학습 불가



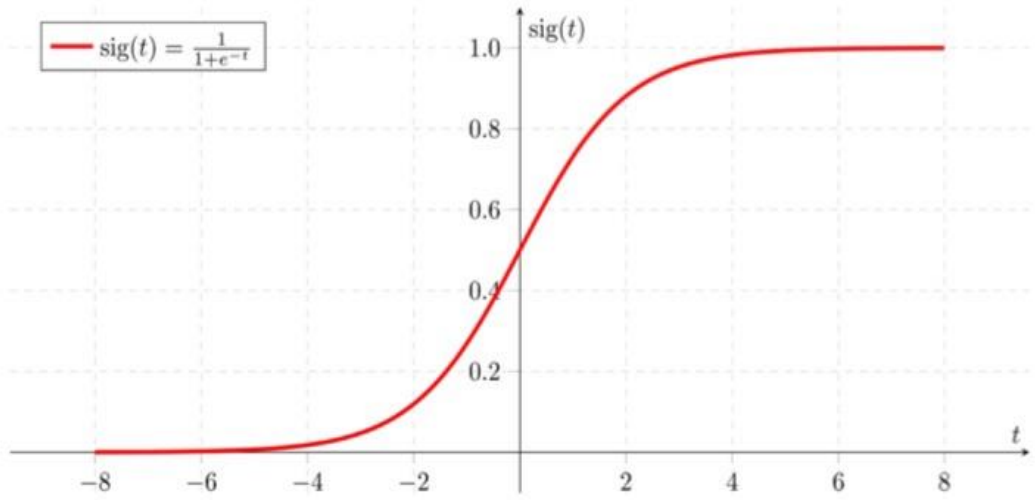


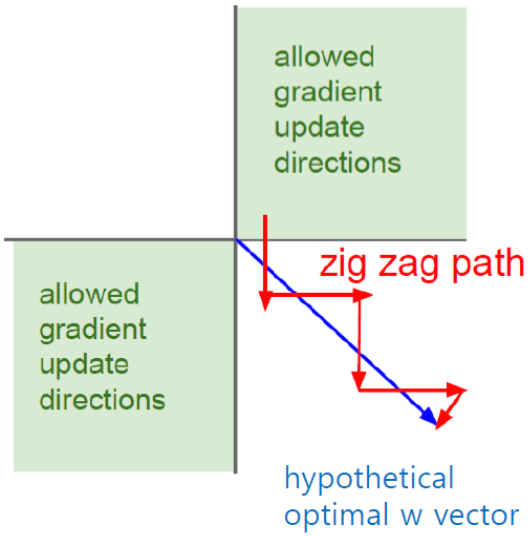
미분 값이 양 극단에서 0으로 수렴하는 모습

활성화함수의 종류

Sigmoid 함수의 문제점 ②

중앙의 값이 0이 아니기 때문에 학습의 방향이 치우쳐지는 zig zag problem



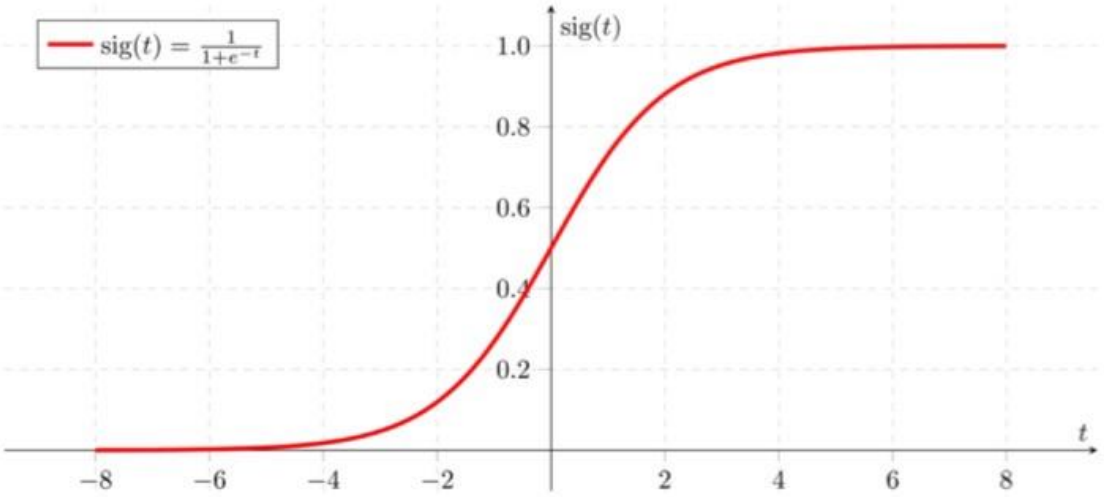


활성화함수의 종류

Output layer에서의 Sigmoid 함수

0과 1사이의 값을 출력하기에

마지막 Layer에서 output으로 **0~1사이의 유사확률**로 표현 가능



활성화함수의 종류

Output layer에서의 Sigmoid 함수

0과 1사이의 값을 출력하기에

마지막 Layer에서 output으로 **0~1사이의 유사확률**로 표현 가능

이진 분류에서 Output Layer의 활성화함수!

이거 어떤가요 괜찮으면 계속 씀 ㅎㅎ

활성화함수의 종류

Softmax 함수

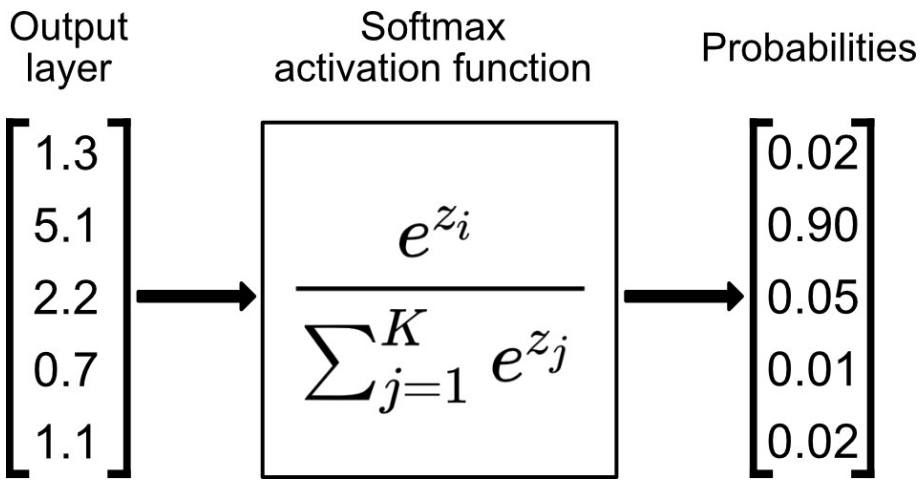
Sigmoid 함수와 유사하게

Multi-class classification을 할 때, Output layer에서 사용

$$\text{softmax}(x)_i = \frac{e^{x_i}}{\sum_j e^{x_j}}$$

전체 합이 1이 되도록 만들어진 다중 분류 함수

활성화함수의 종류



Output layer의 활성화함수로 사용하여 다중 분류 시행

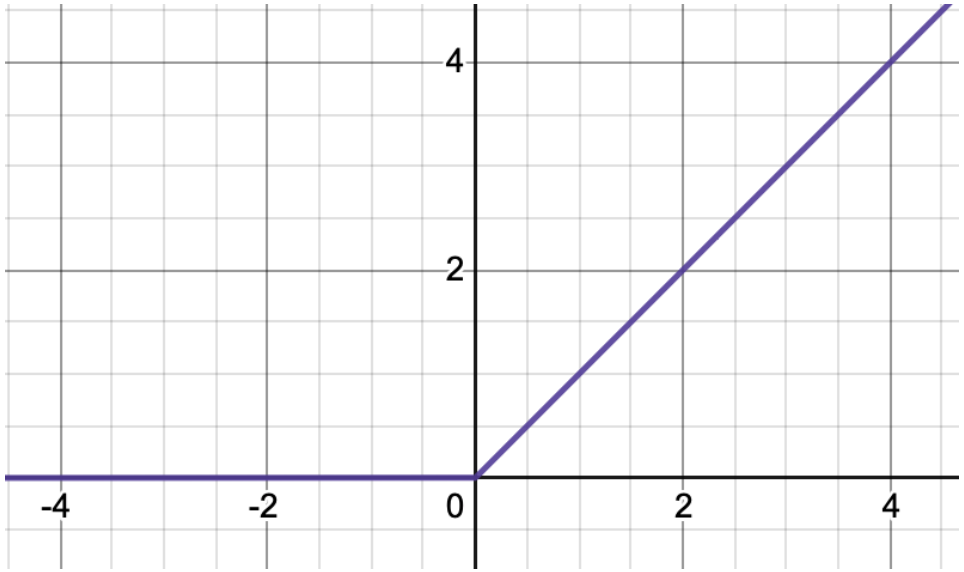
↓

확률이 가장 높은 항목의 클래스로 예측 진행

활성화함수의 종류

ReLU 함수

양수 영역에서 $y=x$, 음수 영역에서 0 값을 가지는 함수



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

활성화함수의 종류

ReLU 함수의 장단점

☑ 장점

- Gradient Vanishing 문제 해결
- 구현의 단순함
- 연산속도의 빠름

☑ 단점

음수 영역의 경우
출력 값과 미분 값이 0이 됨.

↓

Dying ReLU

활성화함수의 종류

ReLU 함수의 장단점

☑ 장점

- Gradient Vanishing 문제 해결
- 구현의 단순함
- 연산속도의 빠름

☑ 단점

음수 영역의 경우

출력 값과 미분 값이 0이 됨.

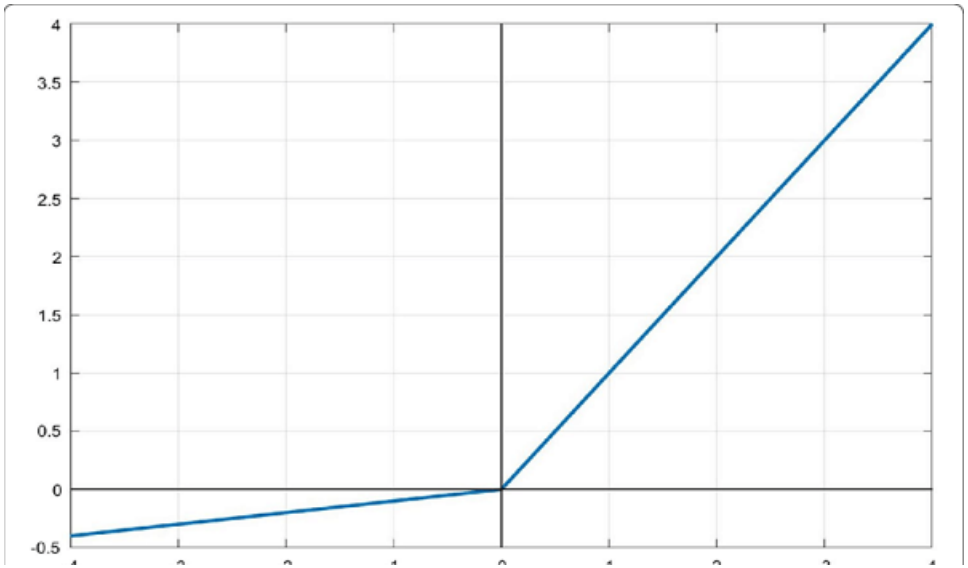
↓

Dying ReLU

활성화함수의 종류

Leaky ReLU 함수

음수 영역에 조금의 기울기를 주어 Dying ReLU 현상을 해결 가능

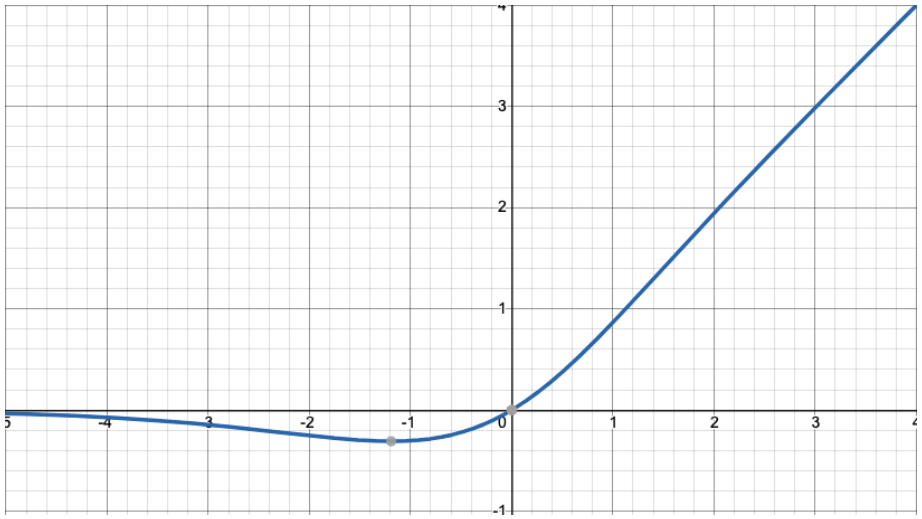


$$\text{Leaky ReLU}(x) = \max(0.01x, x)$$

활성화함수의 종류

Mish 함수

YOLO 등의 이미지 분야에서 매우 잘 활용되는 최신의 활성화함수



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

3

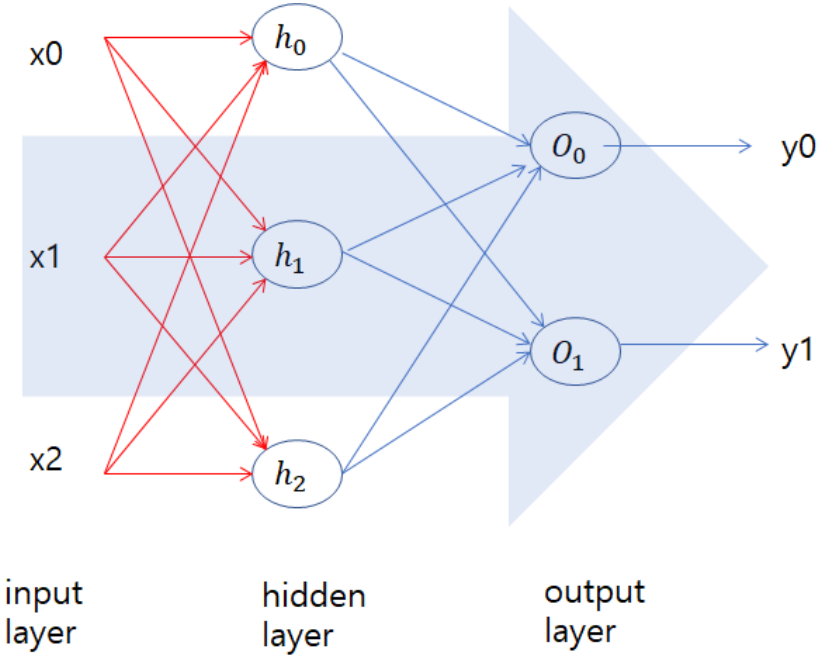
신경망 학습

3 신경망 학습

순전파

순전파 *Forward Propagation*

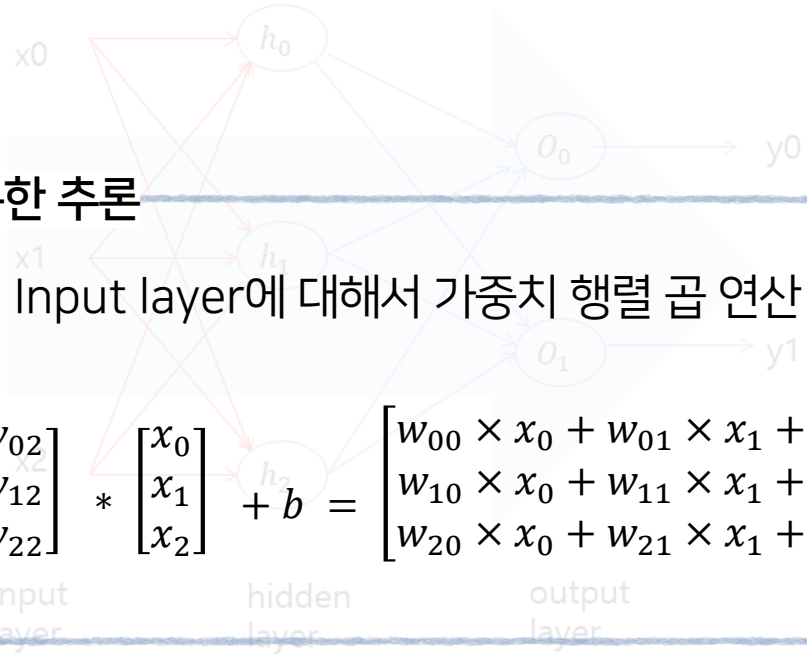
입력 데이터를 기반으로 신경망을 따라 입력층부터 출력층까지
차례대로 변수를 계산하여 결과를 도출하는 **추론의 과정**



순전파

순전파 *Forward Propagation*

입력 데이터를 기반으로 신경망을 따라 입력층부터 출력층까지
차례대로 변수를 계산하여 결과를 도출하는 **추론의 과정**



다중 퍼셉트론을 통한 추론

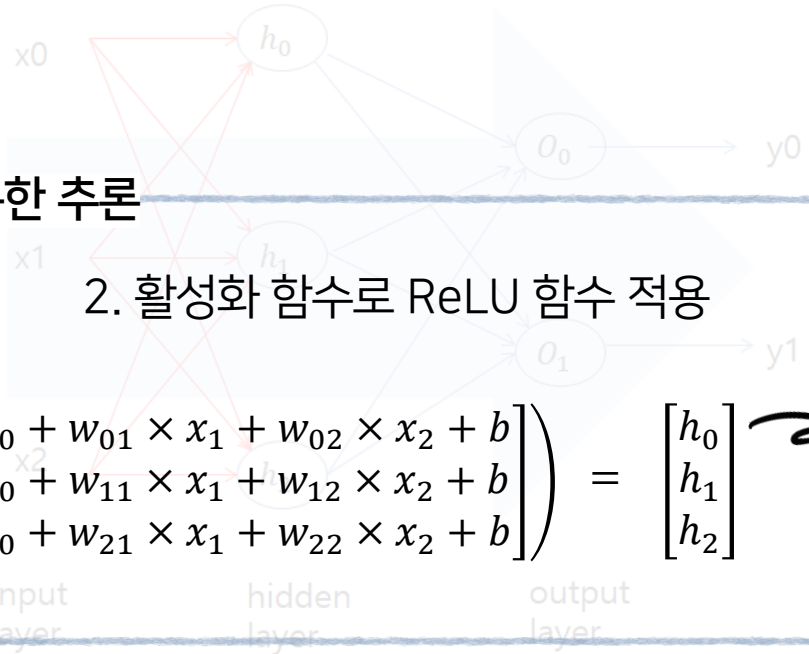
1. Input layer에 대해서 가중치 행렬 곱 연산

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

순전파

순전파 *Forward Propagation*

입력 데이터를 기반으로 신경망을 따라 입력층부터 출력층까지
차례대로 변수를 계산하여 결과를 도출하는 **추론의 과정**



다중 퍼셉트론을 통한 추론

2. 활성화 함수로 ReLU 함수 적용

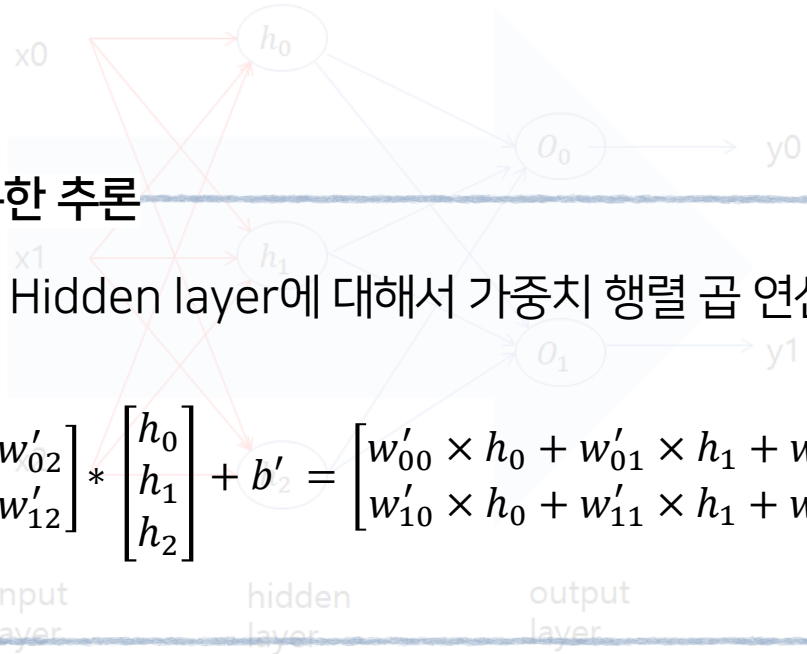
$$ReLU\left(\begin{bmatrix} w_{00} \times x_0 + w_{01} \times x_1 + w_{02} \times x_2 + b \\ w_{10} \times x_0 + w_{11} \times x_1 + w_{12} \times x_2 + b \\ w_{20} \times x_0 + w_{21} \times x_1 + w_{22} \times x_2 + b \end{bmatrix}\right) = \begin{bmatrix} h_0 \\ h_1 \\ h_2 \end{bmatrix}$$

Hidden layer

순전파

순전파 *Forward Propagation*

입력 데이터를 기반으로 신경망을 따라 입력층부터 출력층까지
차례대로 변수를 계산하여 결과를 도출하는 **추론의 과정**



다중 퍼셉트론을 통한 추론

3. Hidden layer에 대해서 가중치 행렬 곱 연산

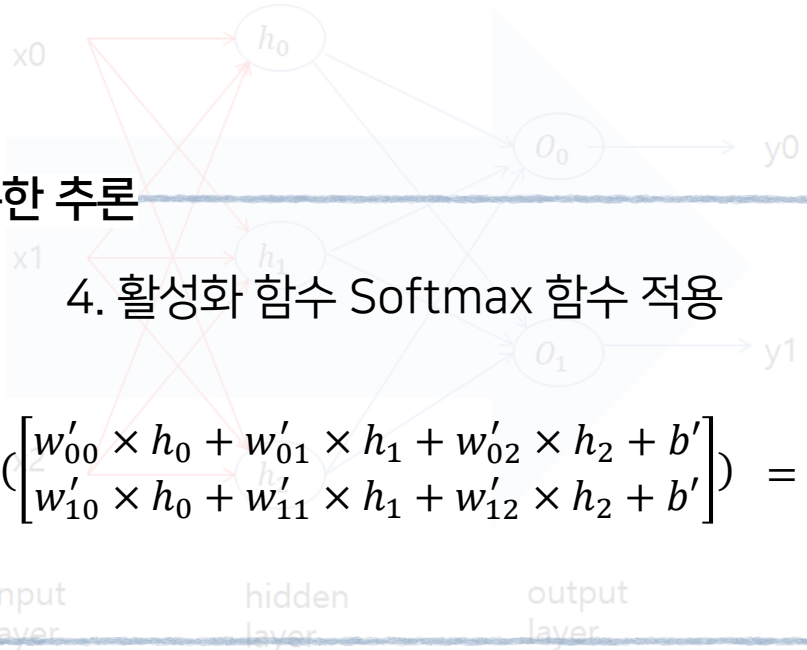
$$\begin{bmatrix} w'_{00} & w'_{01} & w'_{02} \\ w'_{10} & w'_{11} & w'_{12} \end{bmatrix} * \begin{bmatrix} h_0 \\ h_1 \\ h_2 \end{bmatrix} + b' = \begin{bmatrix} w'_{00} \times h_0 + w'_{01} \times h_1 + w'_{02} \times h_2 + b' \\ w'_{10} \times h_0 + w'_{11} \times h_1 + w'_{12} \times h_2 + b' \end{bmatrix}$$

3 신경망 학습

순전파

순전파 *Forward Propagation*

입력 데이터를 기반으로 신경망을 따라 입력층부터 출력층까지
차례대로 변수를 계산하여 결과를 도출하는 **추론의 과정**



다중 퍼셉트론을 통한 추론

4. 활성화 함수 Softmax 함수 적용

$$Softmax\left(\begin{bmatrix} w'_{00} \times h_0 + w'_{01} \times h_1 + w'_{02} \times h_2 + b' \\ w'_{10} \times h_0 + w'_{11} \times h_1 + w'_{12} \times h_2 + b' \end{bmatrix}\right) = \begin{bmatrix} \hat{y}_0 \\ \hat{y}_1 \end{bmatrix}$$

손실함수 계산

Mean Squared Error

회귀 문제에서 적용

$$\mathrm{MSE} = \frac{1}{2}((\mathrm{y}_0 - \hat{\mathrm{y}}_0)^2 + (\mathrm{y}_1 - \hat{\mathrm{y}}_1)^2)$$

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

Cross Entropy

분류 문제에서 적용

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

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

→ 적절한 **손실함수의 선택**이 성능에 큰 영향을 미침

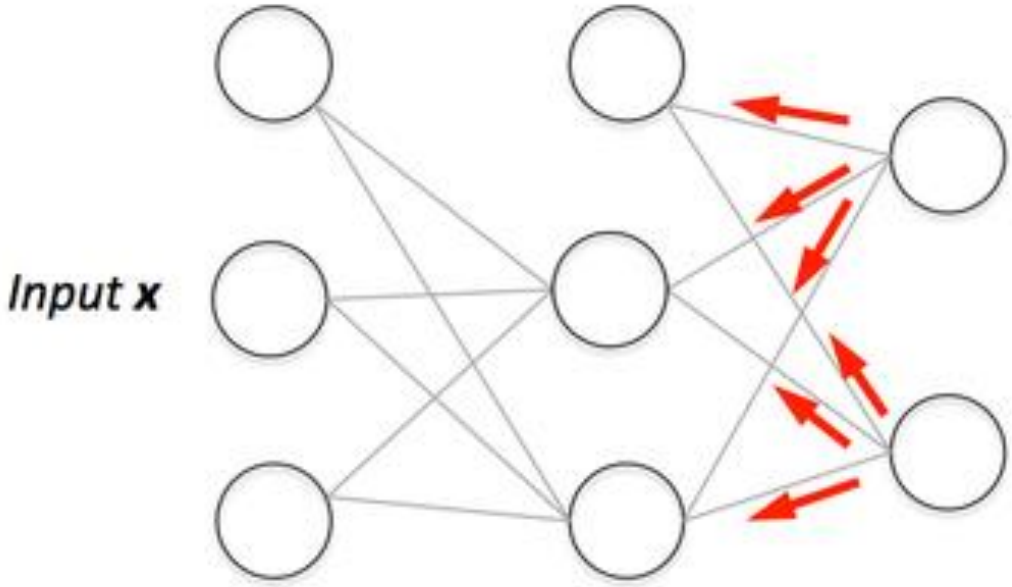
3 신경망 학습

역전파

역전파 *Backward Propagation*

모델이 예측을 위한 **최적의 파라미터를 학습**해나가는 과정

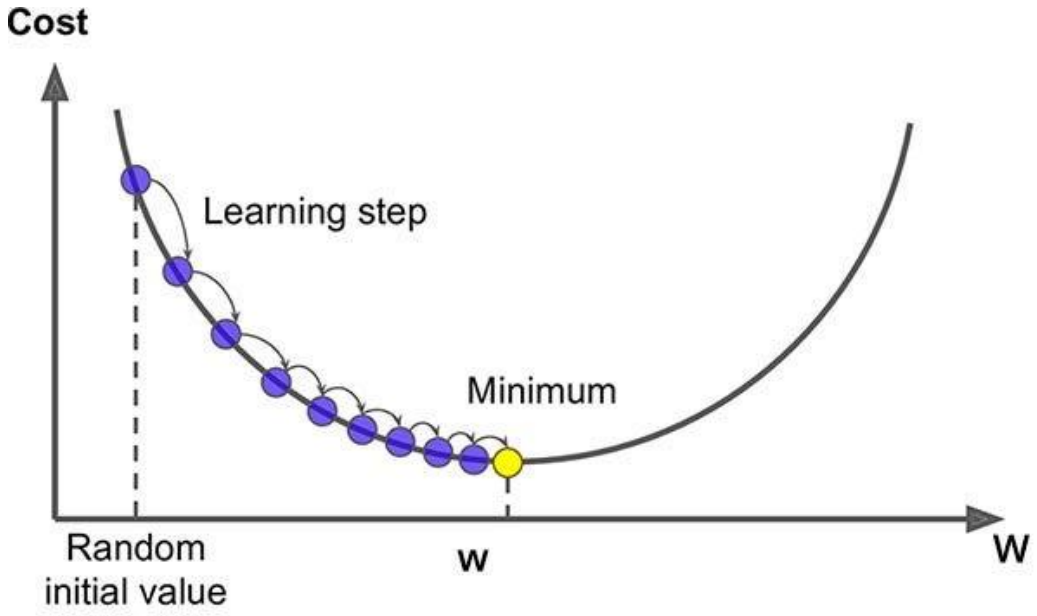
손실함수로 구한 예측 값과 정답의 차이를 최소화 시키는 것이 목적



역전파

경사하강법 *Gradient Descent*

1차 미분계수를 이용하여 파라미터의 최적 값을 찾아가는 최적화 방법

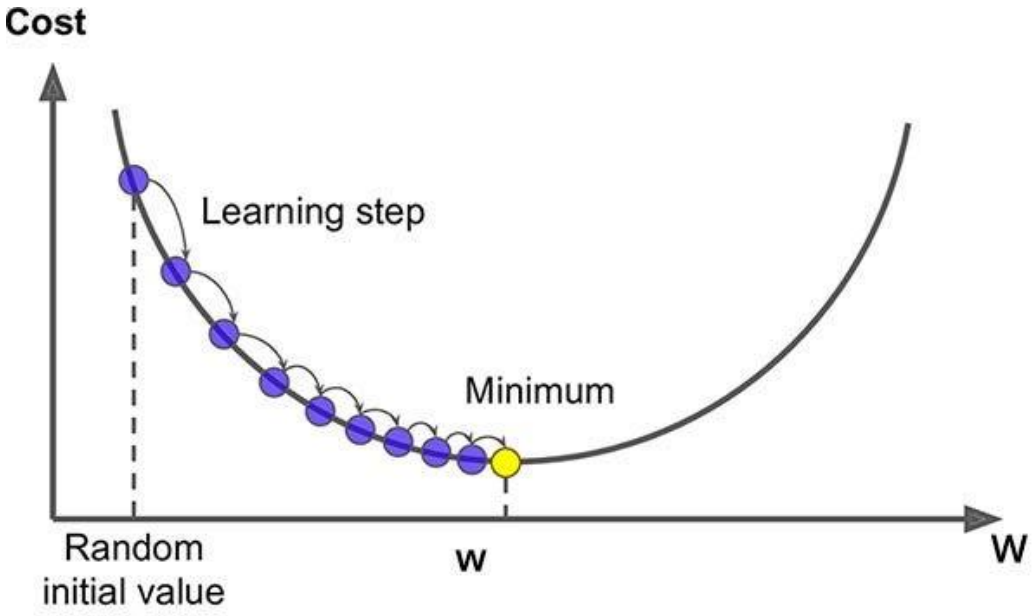


$$w := w - \eta \frac{\partial C}{\partial w}$$

역전파

경사하강법 *Gradient Descent*

1차 미분계수를 이용하여 파라미터의 최적 값을 찾아가는 최적화 방법



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

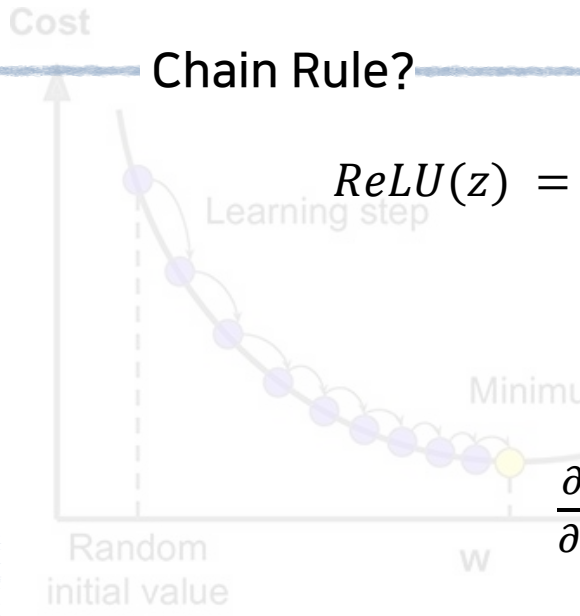
Chain Rule을 통해

가중치와 편향을 업데이트!!

역전파

경사하강법 *Gradient Descent*

1차 미분계수를 이용하여 파라미터의 최적 값을 찾아가는 최적화 방법



Chain Rule?

$ReLU(z) = h,\ Sigmoid(z') = y$이라 했을 때,

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

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

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

Chain Rule을 통해
가중치와 편향을 업데이트!!

4

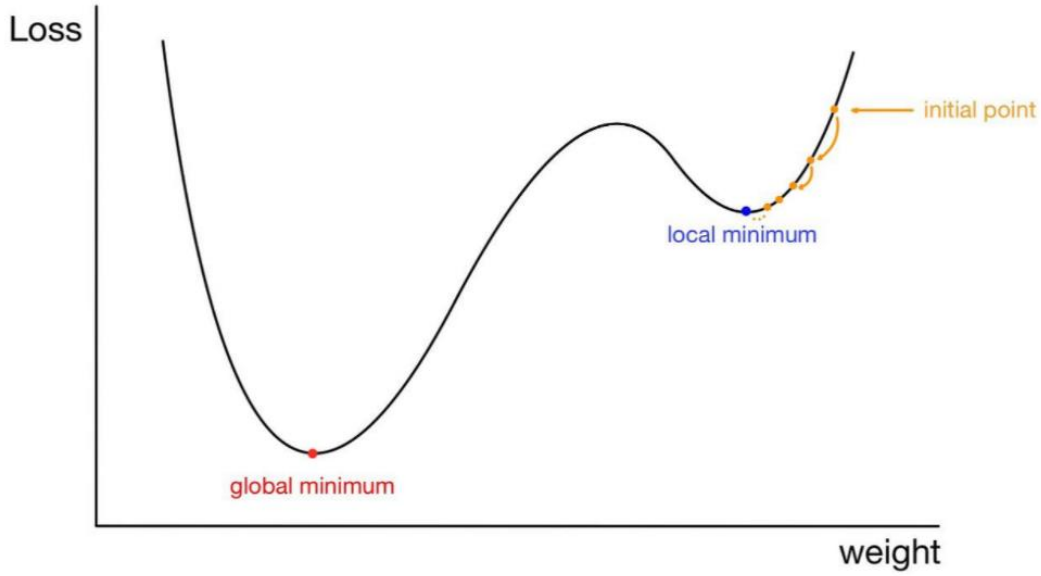
기존 Gradient Descent의 문제점과 해결방안

4 기존 Gradient Descent의 문제점과 해결방안

Gradient Descent의 문제

Local Minimum

국소 최적해로 수렴하여 전역 최적해를 놓치는 문제

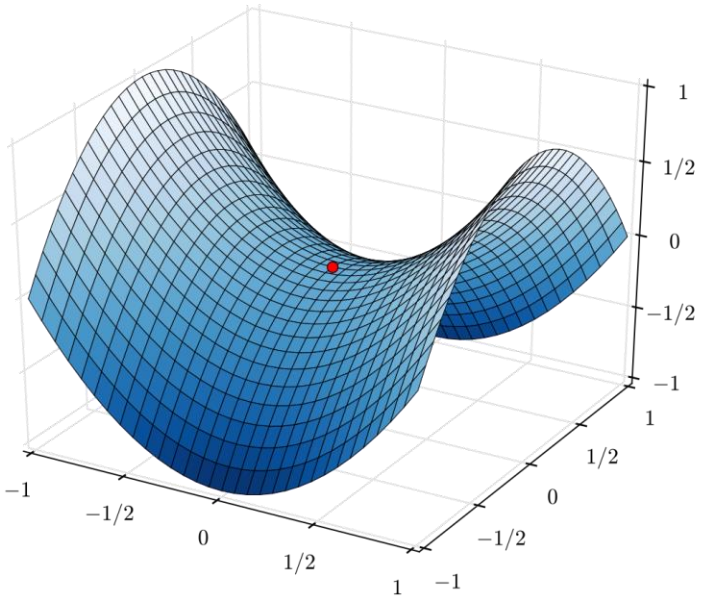


4 기존 Gradient Descent의 문제점과 해결방안

Gradient Descent의 문제

Saddle Point

극값이 아닌 **안장점**에 수렴하여 학습이 종료되는 문제



4 기존 Gradient Descent의 문제점과 해결방안

Gradient Descent의 문제

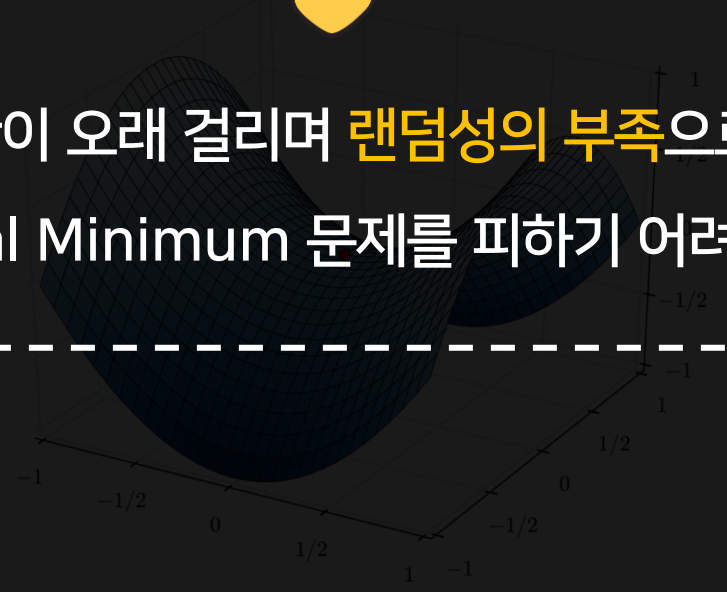
Saddle Point

기존의 Gradient Descent는

전체 데이터(Batch)를 사용

시간이 오래 걸리며 **랜덤성의 부족**으로

Local Minimum 문제를 피하기 어려움



4 기존 Gradient Descent의 문제점과 해결방안

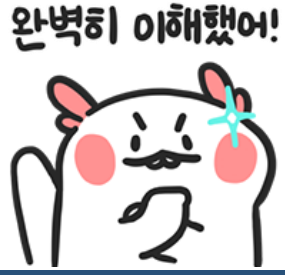
Batch 크기의 문제

Stochastic Gradient Descent

데이터 전체가 아닌

데이터의 일부(Mini Batch)를 사용하는 Gradient Descent 방법

SGD	MGD
Mini batch size	Mini batch size
N = 1	1 < N < M

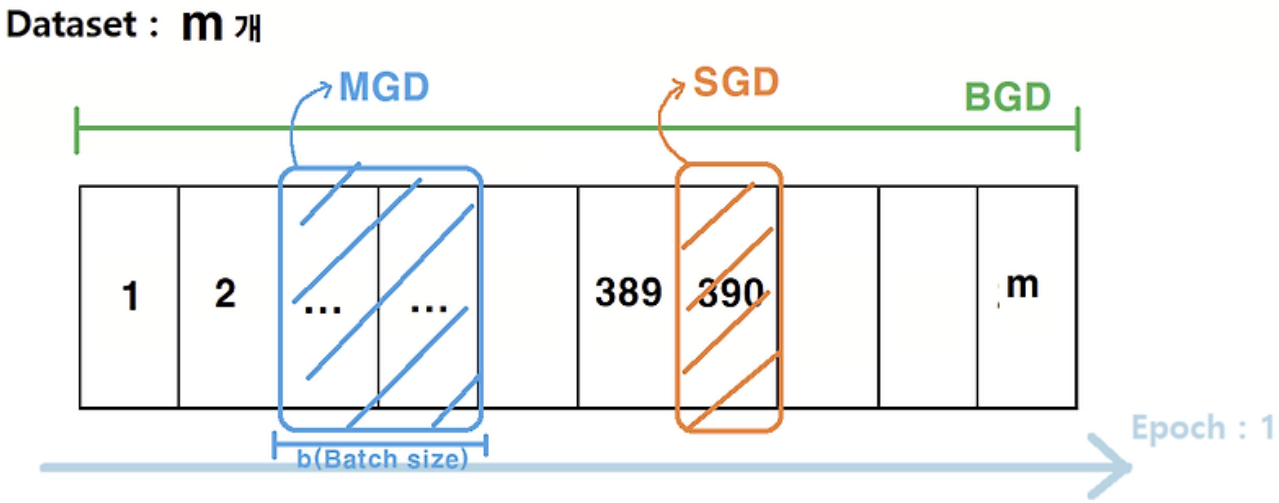


4 기존 Gradient Descent의 문제점과 해결방안

Batch 크기의 문제

Stochastic Gradient Descent

한 번에 적은 데이터를 사용하기 때문에 **속도와 메모리 측면에서의 이점**

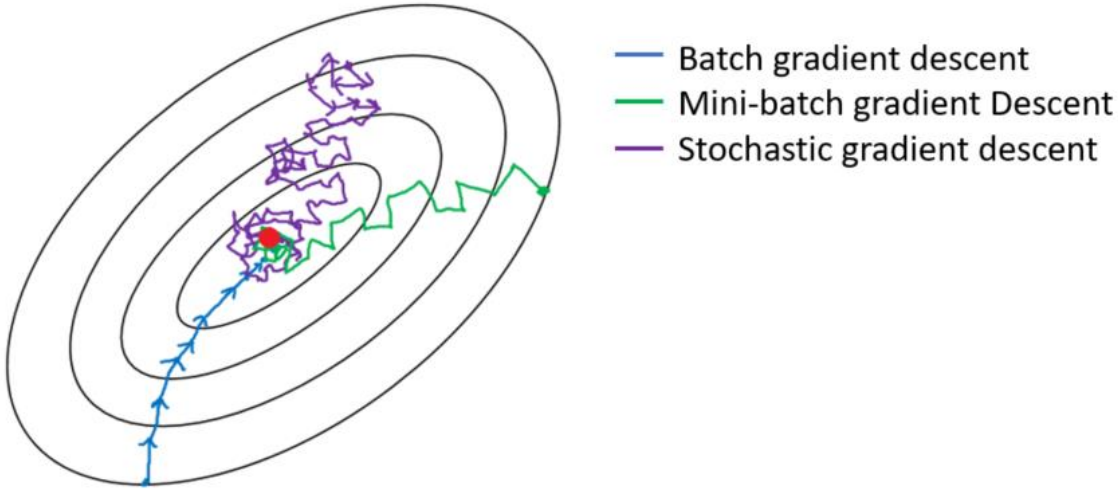


4 기존 Gradient Descent의 문제점과 해결방안

Batch 크기의 문제

Stochastic Gradient Descent

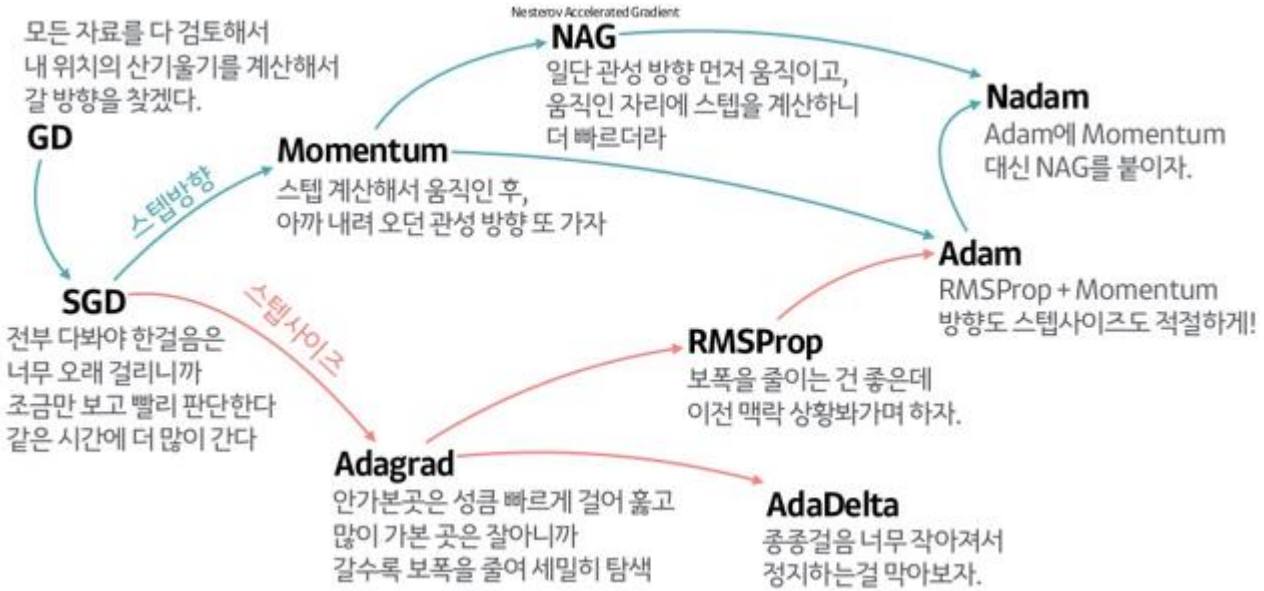
노이즈가 발생하여 Local Minimum에서 탈출할 가능성이 있음



4 기존 Gradient Descent의 문제점과 해결방안

Special Optimizers

SGD를 변형한 여러 Optimizer들이 존재하며, 장단점 또한 다양함

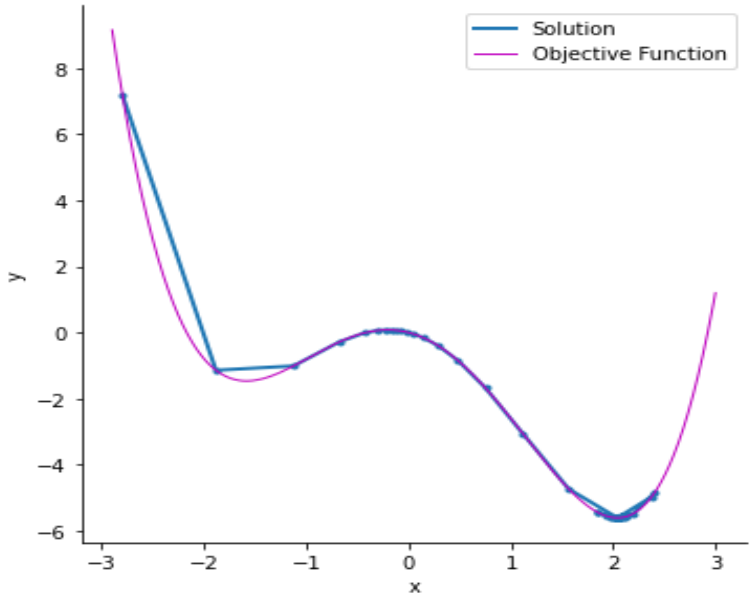


4 기존 Gradient Descent의 문제점과 해결방안

Special Optimizers

Momentum

기울기가 클 때 **강한 관성을 부여**하여 Local minimum을 넘어가는데 도움



Momentum term

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

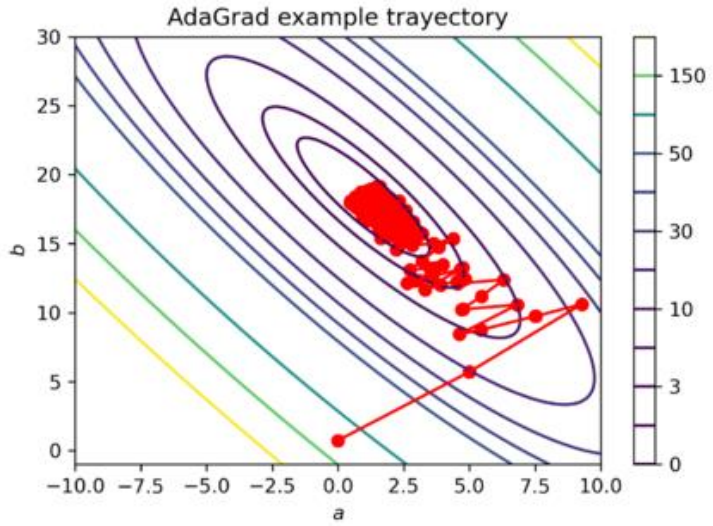
$$x_t = x_{t-1} - v_t$$

4 기존 Gradient Descent의 문제점과 해결방안

Special Optimizers

Adagrad

Feature 별로 **서로 다른 학습률**을 적용



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

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

Model parameter

4 기존 Gradient Descent의 문제점과 해결방안

Special Optimizers

Adagrad

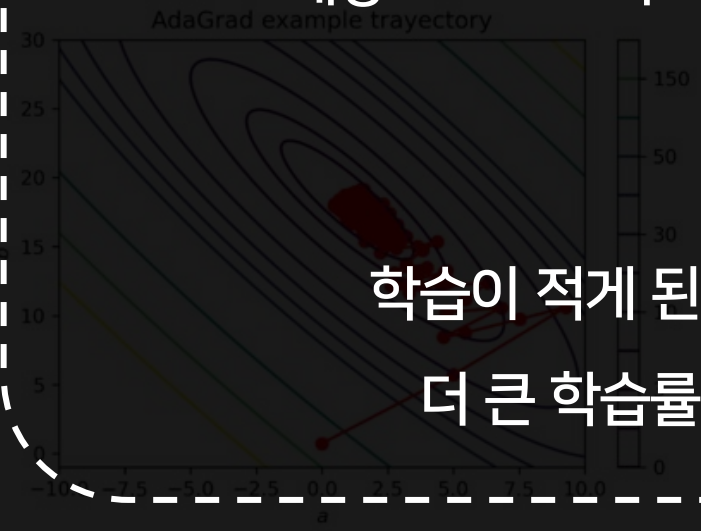
벡터 g_t의 역할

Feature별 학습률을 유연하게 조절

그러나 학습률이 매우 작아질 위험 존재

g_t 는 누적 기울기 크기 벡터이며 특정 성분 값이 클수록

해당 feature의 학습률의 크기가 더 감소



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

학습이 적게 된 변수가 상대적으로

더 큰 학습률을 가지도록 유도

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

4 기존 Gradient Descent의 문제점과 해결방안

Special Optimizers

Adagrad

Feature 별로 **서로 다른 학습률**을 적용

g_t가 점점 커져서 **학습률이 0으로 가는 현상**이 발생함

4 기존 Gradient Descent의 문제점과 해결방안

Special Optimizers

RMSProp

Adagrad와 다르게 **지수이동평균**으로 기울기 누적 크기를 업데이트

최근 time step의 기울기를 많이 반영하고 먼 과거의 time step에서의 기울기는 적게 반영

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

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

지수이동평균을 통해 g_t 가 무한히 커지는 것을 방지하여

더욱 오래 학습할 수 있게 됨

4 기존 Gradient Descent의 문제점과 해결방안

Special Optimizers

Adam

Momentum과 RMSProp을 **결합**하여 좋은 성능을 이끌어냄

추가로, 학습 초기 m_t와 g_t의 소실 문제를 방지하는 **편향 보정**을 적용

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

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

$$\hat{m}_t = \frac{m_t}{1-\beta_1^t},\ \hat{g}_t = \frac{g_t}{1-\beta_2^t}$$ 편향보정

$$x_t = x_{t-1} - \frac{\alpha}{\sqrt{g_t+\epsilon}} \cdot \hat{m}_t$$

다음주 예고

1. 이미지 데이터
2. CNN
3. 발전된 CNN 모델
4. Object Detection

THANK YOU

NLP 이제 나와~~