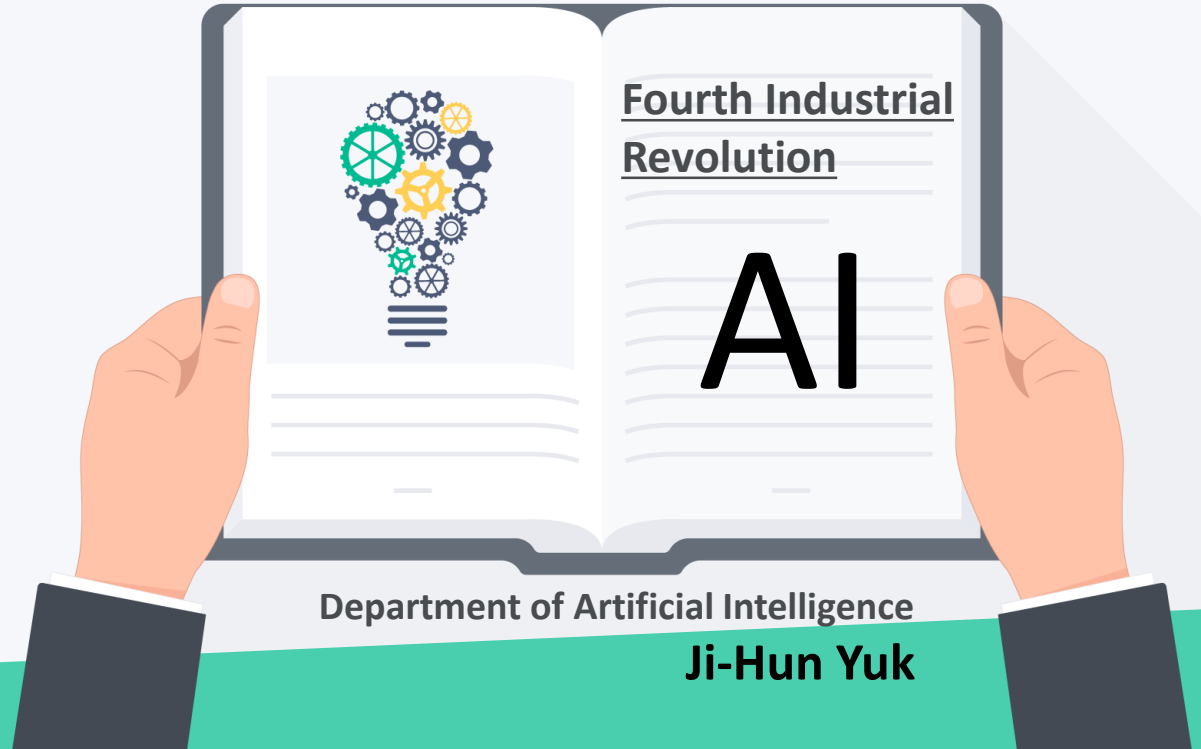
# What is Artificial Intelligence(AI)

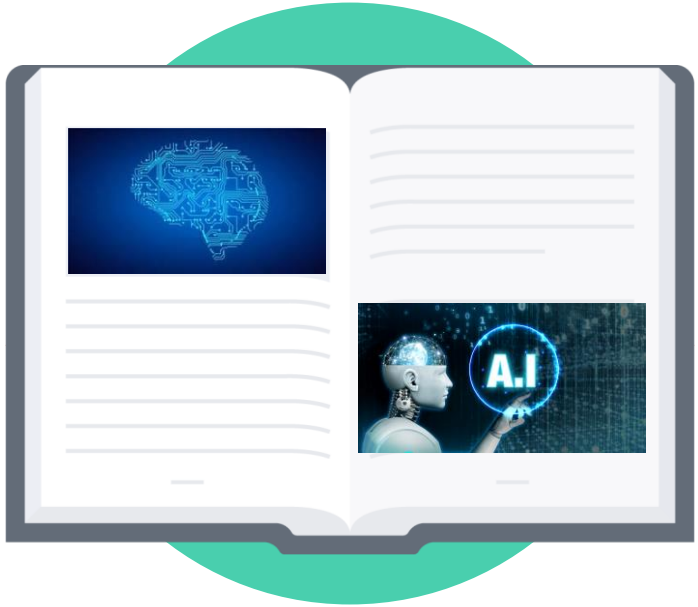
affective.AI Lab




Department of Artificial Intelligence
Ji-Hun Yuk

11 July 2023

# Table of contents



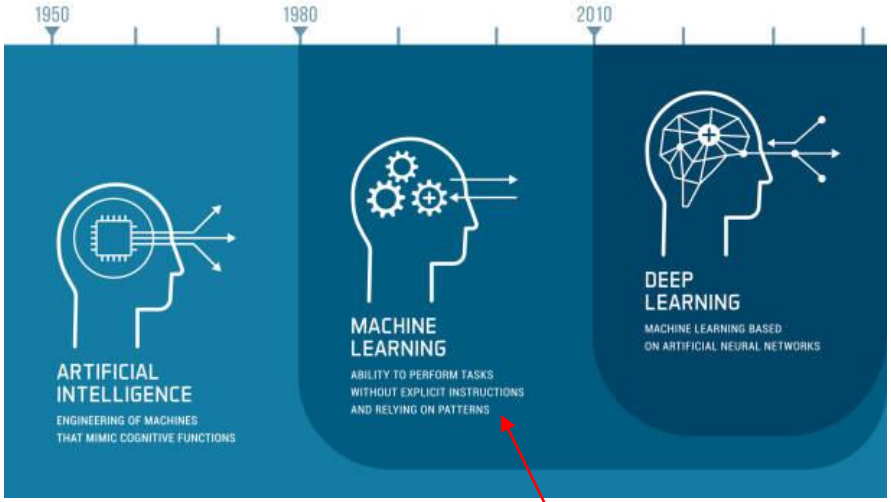

목차

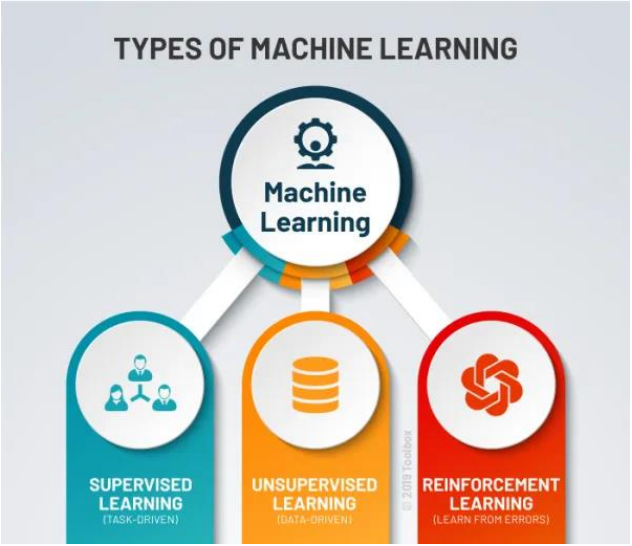
# 1. What is AI?

**Definition:**

**Artificial intelligence (AI) is intelligence—perceiving, synthesizing, and inferring information—demonstrated by machines, as opposed to intelligence displayed by humans or by other animals.**

# 2. What is Machine Learning?

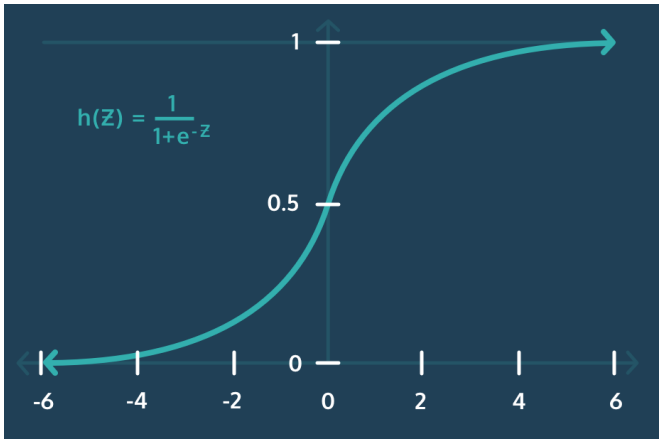


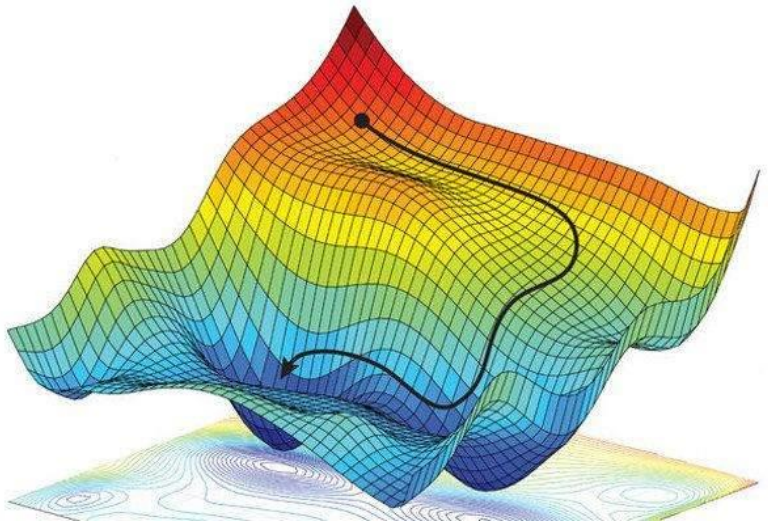



# 3-0. Weekly study

## Ch4. 다양한 분류 알고리즘





11 July 2023

# 3-1. 로지스틱 회귀

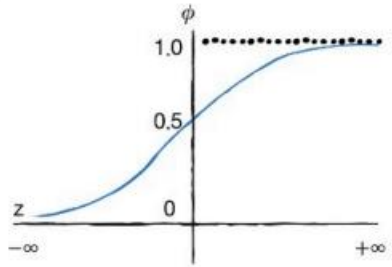
- 선형 방정식을 사용한 분류 알고리즘.
- 선형 회귀와 달리 시그모이드 함수나 소프트맥스 함수를 사용하여 클래스 확률을 출력할 수 있다.

**로지스틱 회귀**logistic regression는 이름은 회귀이지만 분류 모델입니다. 이 알고리즘은 선형 회귀와 동일하게 선형 방정식을 학습합니다. 예를 들면 다음과 같습니다.

$$z = a \times (Weight) + b \times (Length) + c \times (Diagonal) + d \times (Height) + e \times (Width) + f$$

$$\phi = \frac{1}{1+e^{-z}}$$

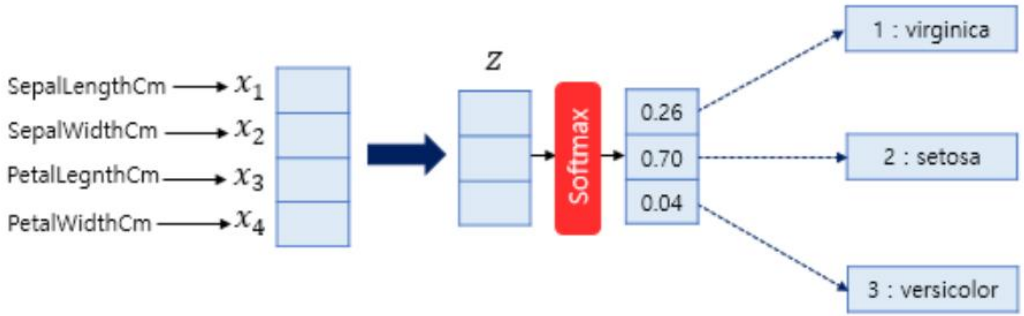
시그모이드 함수




시그모이드 그래프

# 3-1. 다중분류

- 타깃 클래스가 2개 이상인 분류 문제.
- 로지스틱 회귀는 다중 분류를 위해 소프트맥스 함수를 사용하여 클래스를 예측한다.



# 3-1. 시그모이드 & 소프트맥스 f

- **시그모이드 함수**는 선형 방정식의 출력을 0과 1 사이의 값으로 압축하며 이진 분류를 위해 사용합니다.
- **소프트맥스 함수**는 다중 분류에서 여러 선형 방정식의 출력 결과를 정규화하여 합이 1이 되도록 만듭니다.

## Sigmoid

2 classes

$$H(x) = sigmoid(Wx + b) = \frac{1}{1 + e^{-(Wx+b)}} = \sigma(Wx + b)$$

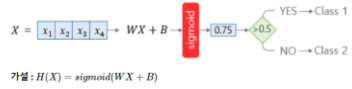
out = P(Y=class1|X)

## SoftMax

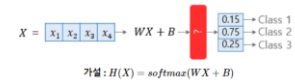
k>2 classes

$$\text{out} = \begin{bmatrix} P(Y=class1|X) \\ P(Y=class2|X) \\ P(Y=class3|X) \\ \vdots \\ P(Y=classk|X) \end{bmatrix}$$

- 로지스틱 회귀

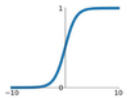



- 소프트맥스 회귀

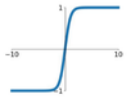



## Activation Functions

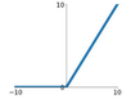
**Sigmoid**
$\sigma(x) = \frac{1}{1+e^{-x}}$



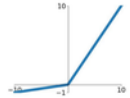
**tanh**
$\tanh(x)$



**ReLU**
$\max(0, x)$

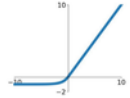


**Leaky ReLU**
$\max(0.1x, x)$



**Maxout**
$\max(w_1^T x + b_1, w_2^T x + b_2)$

**ELU**
$\begin{cases} x & x \geq 0 \\ \alpha(e^x - 1) & x < 0 \end{cases}$

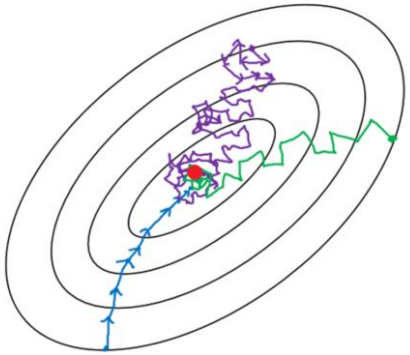


Output layer | Softmax activation function | Probabilities

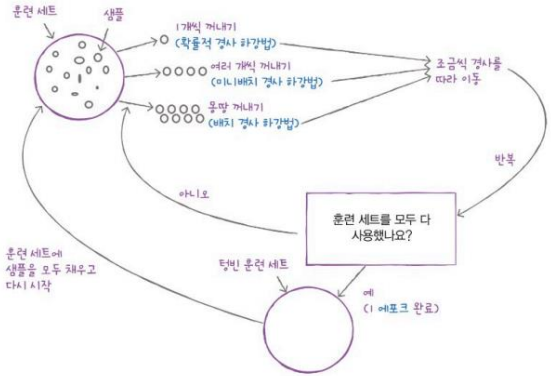
$$\begin{bmatrix} 1.3 \\ 5.1 \\ 2.2 \\ 0.7 \\ 1.1 \end{bmatrix} \rightarrow \frac{e^{z_i}}{\sum_{j=1}^{K} e^{z_j}} \rightarrow \begin{bmatrix} 0.02 \\ 0.90 \\ 0.05 \\ 0.01 \\ 0.02 \end{bmatrix}$$

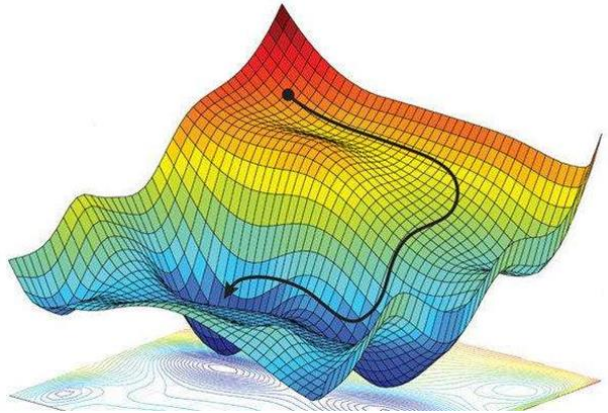
# 3-2. 확률적 경사 하강법

note 실제로 등산할 때는 등산로를 벗어나면 안 됩니다!

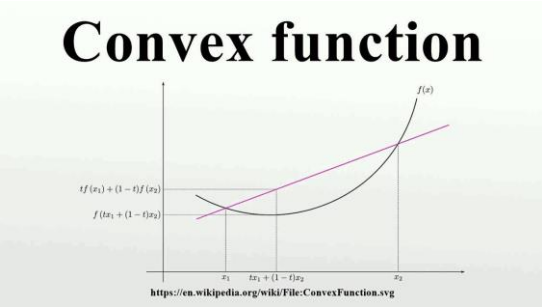
Batch gradient descent
Mini-batch gradient Descent
Stochastic gradient descent



# 3-2. 손실 함수

- 확률적 경사 하강법이 최적화할 대상.
- 대부분의 문제에 잘 맞는 손실 함수가 이미 정의되어 있다.
- 이진 분류에는 로지스틱 회귀(또는 이진 크로스엔트로피) 손실 함수를 사용.
- 다중 분류에는 크로스엔트로피 손실 함수를 사용한다.
- 회귀 문제에는 평균 제곱 오차 손실 함수를 사용한다.
- 기술적으로 손실 함수는 미분 가능해야 한다(연속이고 좌미분계수와 우미분계수가 같아야 한다).




```python
# 손실 함수 정의
class ComplexMSELoss(nn.Module):
    def __init__(self):
        super(ComplexMSELoss, self).__init__()

    def forward(self, input, target):
        real_loss = torch.mean(torch.square(input.real - target.real))
        imag_loss = torch.mean(torch.square(input.imag - target.imag))
        loss = torch.sqrt(real_loss * real_loss + imag_loss * imag_loss)
        return loss
```

# 3-2. 에포크

- 확률적 경사 하강법에서 전체 샘플을 모두 사용하는 한 번 반복을 의미.
- 일반적으로 경사 하강법 알고리즘은 수십에서 수백 번의 에포크를 반복.
- 일반적으로 batch_size에 해당하는 dimension은 표시되어 있지 않음.
- 실제 학습이 수행될 때는 미니 배치 경사하강법으로 수행하기 때문에 batch_size 만큼의 개수에 해당하는 데이터에 대해 일괄 연산을 수행하며, PyTorch에서는 일반적으로 dim 0이 batch_size에 해당됨


1 Epoch : 모든 데이터 셋을 한 번 학습
1 iteration : 1회 학습

minibatch : 데이터 셋을 batch size 크기로 쪼개서 학습

ex) 총 데이터가 100개, batch size가 10이면,
1 iteration = 10개 데이터에 대해서 학습
1 Epoch = 100/batch size = 10 iteration

```
# 학습
nb_epochs = 100
for epoch in range(nb_epochs + 1):
  for batch_idx, samples in enumerate(dataloader):
```

# 4. Summary

# 5. Q&A

# Thanks for listening