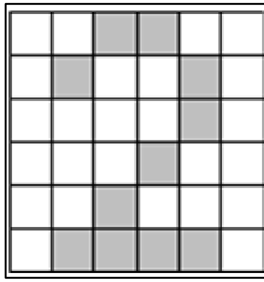
머신러닝 / 딥러닝 리뷰

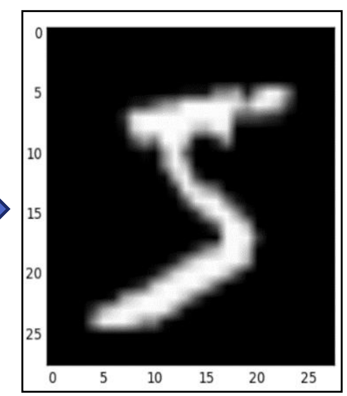
박성호 (neowizard2018@gmail.com)

행렬(matrix) 필요성 ⇒ Text Data + Image Data

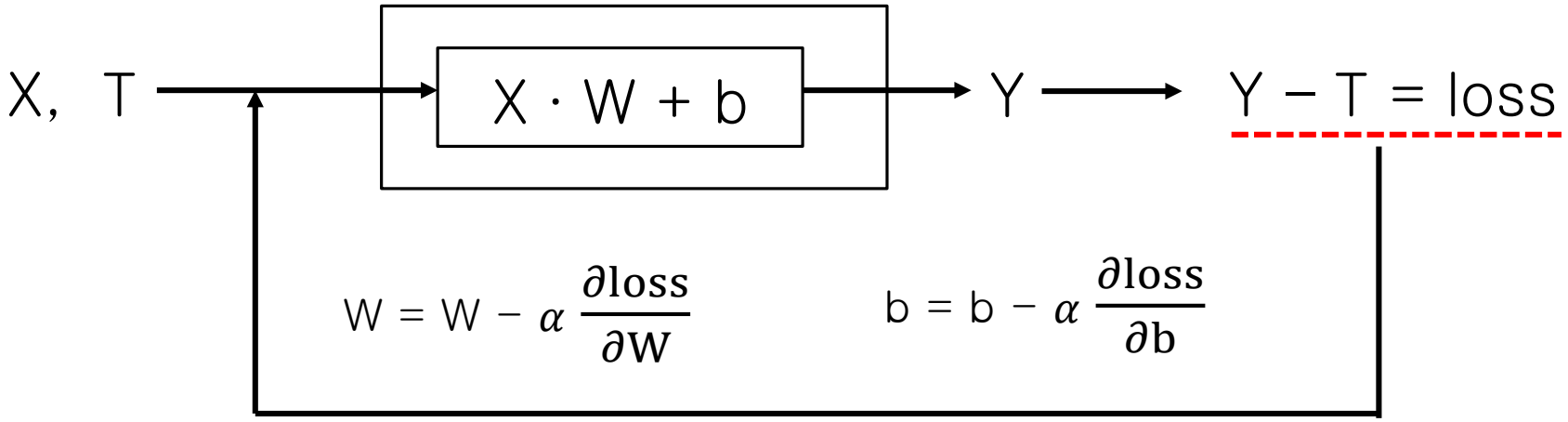


0	0	1	1	0	0
0	1	0	0	1	0
0	0	0	0	1	0
0	0	0	1	0	0
0	0	1	0	0	0
0	1	1	1	1	0

5,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0
,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,
0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,3,18,18,18,126,136,175,26,166,255,247,127,0,0
,0,0,0,0,0,0,0,0,0,0,30,36,94,154,170,253,253,253,253,253,225,172,253,242,195,64,0,0,0,0,0,0,0,0,0,0,0,49,238,253,253
,253,253,253,253,253,253,251,93,82,82,56,39,0,0,0,0,0,0,0,0,0,0,0,0,18,219,253,253,253,253,253,198,182,247,241,0,0,0,
0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,80,156,107,253,253,205,11,0,43,154,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,14,1,154,253,9
0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,139,253,190,2,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,11,
190,253,70,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,35,241,225,160,108,1,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0
,0,0,0,0,0,81,240,253,253,119,25,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,45,186,253,253,150,27,0,0,0,0,0,0,0,0,
0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,16,93,252,253,187,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,249,253,249,64,0,0,
0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,46,130,183,253,253,207,2,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,39,148,229,253,2
53,253,250,182,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,24,114,221,253,253,253,253,201,78,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,
0,23,66,213,253,253,253,253,198,81,2,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,18,171,219,253,253,253,253,195,80,9,0,0,0,0,0,0,
0,0,0,0,0,0,0,0,0,0,55,172,226,253,253,253,253,244,133,11,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,136,253,253,253,212,135
,132,16,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0
,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0



머신러닝 / 딥러닝 기본 아키텍처



학습 (learning): 계산 값 Y 와 정답 T 와의 차이를 나타내는 손실 값 (또는 손실함수) loss 가 최소가 될 때까지 가중치 W 와 바이어스 b 를 최적화시키는 과정.

즉 주어진 학습데이터에 대해 손실 함수가 최소가 되는 가중치와 바이어스를 찾는것이 머신러닝/딥러닝 궁극적 목적이라 할 수 있음

데이터관점에서 곱하기 / 더하기 / 행렬곱

곱하기

$3 \times 1 = 3$

$3 \times 0 = 0$

$3 \times 2 = 6$

더하기

$1 + 2 + 3 = 6$

행렬곱

$$A = \begin{pmatrix} 0 & 1 \\ 2 & 3 \end{pmatrix} \qquad B = \begin{pmatrix} 4 & 5 \\ 6 & 7 \end{pmatrix}$$

$$A \cdot B = \begin{pmatrix} 0 \times 4 + 1 \times 6 & 0 \times 5 + 1 \times 7 \\ 2 \times 4 + 3 \times 6 & 2 \times 5 + 3 \times 7 \end{pmatrix}$$

$$= \begin{pmatrix} 6 & 7 \\ 26 & 31 \end{pmatrix}$$

Review – 손실함수 (loss function)

모든 손실 함수를 알아야 한다 ? 손실 함수 목적 ?

모든 손실 함수에서 나타나는 공통점, 즉 손실 함수 공식을 통해 알 수 있는 Insight ?

- 일반 딥러닝 손실함수 종류

$$MAE = \frac{1}{N}\sum_{i=1}^{n}\left|\hat{y}_i - y_i\right|$$

$$MSE = \frac{1}{N}\sum_{i=1}^{n}(\hat{y}_i - y_i)^2$$

$$RMSE = \sqrt{\frac{1}{N}\sum_{i=1}^{n}(\hat{y}_i - y_i)^2}$$

$$BCE = -\frac{1}{N}\sum_{i=0}^{N} y_i \cdot \log(\hat{y}_i) + (1-y_i) \cdot \log(1-\hat{y}_i)$$

$$CCE = -\frac{1}{N}\sum_{i=0}^{N}\sum_{j=0}^{J} y_j \cdot \log(\hat{y}_j) + (1-y_j) \cdot \log(1-\hat{y}_j)$$

- GAN 손실함수

$$\min_{\theta_g}\max_{\theta_d}\left[\mathbb{E}_{x\sim p_{data}}\log D_{\theta_d}(x) + \mathbb{E}_{z\sim p(z)}\log(1 - D_{\theta_d}(G_{\theta_g}(z)))\right]$$

Review – 손실함수 (loss function)

- YOLO 손실함수

$$\lambda_{\text{coord}} \sum_{i=0}^{S^2} \sum_{j=0}^{B} \mathbb{1}_{ij}^{\text{obj}} \left[(x_i - \hat{x}_i)^2 + (y_i - \hat{y}_i)^2 \right]$$

$$+ \lambda_{\text{coord}} \sum_{i=0}^{S^2} \sum_{j=0}^{B} \mathbb{1}_{ij}^{\text{obj}} \left[\left(\sqrt{w_i} - \sqrt{\hat{w}_i} \right)^2 + \left(\sqrt{h_i} - \sqrt{\hat{h}_i} \right)^2 \right]$$

$$+ \sum_{i=0}^{S^2} \sum_{j=0}^{B} \mathbb{1}_{ij}^{\text{obj}} \left(C_i - \hat{C}_i \right)^2$$

$$+ \lambda_{\text{noobj}} \sum_{i=0}^{S^2} \sum_{j=0}^{B} \mathbb{1}_{ij}^{\text{noobj}} \left(C_i - \hat{C}_i \right)^2$$

$$+ \sum_{i=0}^{S^2} \mathbb{1}_{i}^{\text{obj}} \sum_{c \in \text{classes}} \left(p_i(c) - \hat{p}_i(c) \right)^2$$

미분 (Derivative)

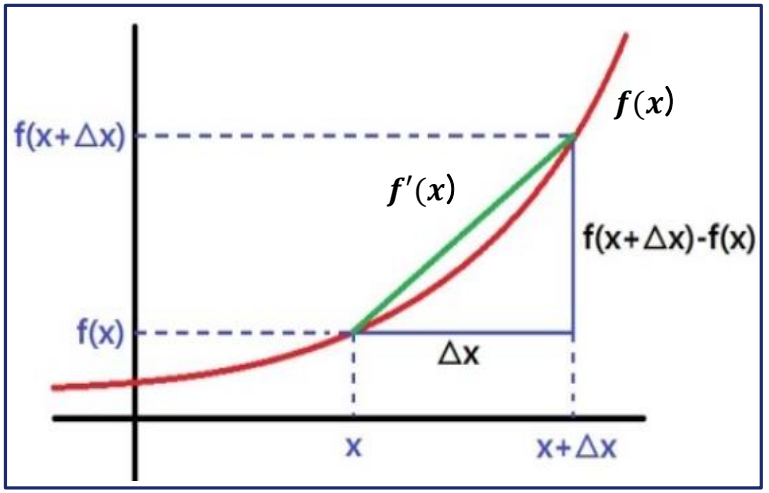
미분을 왜 하는가 ?

미분으로 얻을 수 있는 인사이트 ?

$$f'(x) = \frac{df(x)}{dx} = \lim_{\Delta x \to 0} \frac{f(x+\Delta x) - f(x)}{\Delta x}$$

f(x) 를 미분하라

접선의 기울기 ? 순간 변화율 ?



참고 동영상 강의: https://youtu.be/PnQe3-6812k

미분 – derivative

- 머신러닝 / 딥러닝에서 자주 사용되는 함수의 미분

$f(x) = \text{상수} \Rightarrow f'(x) = 0$

$f(x) = ax^n \Rightarrow f'(x) = nax^{n-1}$

$f(x) = e^x \Rightarrow f'(x) = e^x$

$f(x) = \ln x \Rightarrow f'(x) = \frac{1}{x}$

$f(x) = e^{-x} \Rightarrow f'(x) = -e^{-x}$

[예 1] $f(x) = 3x^2 + e^x + 7 \Rightarrow f'(x) = 6x + e^x$

[예 2] $f(x) = \ln x + \frac{1}{x} \Rightarrow f'(x) = \frac{1}{x} - \frac{1}{x^2}$

참고

$\frac{1}{x} = x^{-1}$

편미분 – partial derivative

- 편미분은 입력변수가 하나 이상인 다변수 함수에서, 미분하고자 하는 변수 하나를 제외한 나머지 변수들은 상수로 취급하고, 해당 변수를 미분하는 것

예를들어 $f(x,y)$ 를 변수 x 에 대해 편미분 하는 경우 다음과 같이 나타냄 → $\frac{\partial f(x,y)}{\partial x}$

[예1] $f(x,y) = 2x + 3xy + y^3$, 변수 x 에 대하여 편미분

$$\frac{\partial f(x,y)}{\partial x} = \frac{\partial(2x+3xy+y^3)}{\partial x} = 2 + 3y$$

[예2] $f(x,y) = 2x + 3xy + y^3$, 변수 y 에 대하여 편미분

$$\frac{\partial f(x,y)}{\partial y} = \frac{\partial(2x+3xy+y^3)}{\partial y} = 3x + 3y^2$$

[예3] 체중 함수가 '체중(야식, 운동)' 처럼 야식/운동에 영향을 받는 2변수 함수라고 가정할 경우, 편미분을 이용하면 각 변수 변화에 따른 체중 변화량을 구할 수 있음

현재 먹는 야식의 양에서 조금 변화를 줄 경우 체중은 얼마나 변하는가 ? → $\frac{\partial 체중}{\partial 야식}$

현재 하고 있는 운동량에 조금 변화를 줄 경우 체중은 얼마나 변하는가 ? → $\frac{\partial 체중}{\partial 운동}$

수치미분 – numerical derivative (딥러닝 프레임웍 내부에 구현)

```python
import numpy as np

def numerical_derivative(f, x):
    delta_x = 1e-4
    grad = np.zeros_like(x)

    it = np.nditer(x, flags=['multi_index'], op_flags=['readwrite'])

    while not it.finished:
        idx = it.multi_index

        tmp_val = x[idx]
        x[idx] = float(tmp_val) + delta_x
        fx1 = f(x)   # f(x+delta_x)

        x[idx] = float(tmp_val) - delta_x
        fx2 = f(x)   # f(x-delta_x)
        grad[idx] = (fx1 - fx2) / (2*delta_x)

        x[idx] = tmp_val
        it.iternext()

    return grad
```

참고 동영상 강의: https://youtu.be/3ELMlbGTIDs

[예제] Numerical Derivative

[예제 1] $f(x) = x^2$ 함수에 대해, x=1 에서의 미분 값 f'(3.0) 구하고 결과를 해석 하시오

[예제 2] $f(x, y) = 2x + 3xy + y^3$ 함수에 대해, 미분 값 f'(1.0, 2.0) 구하고 결과를 해석 하시오

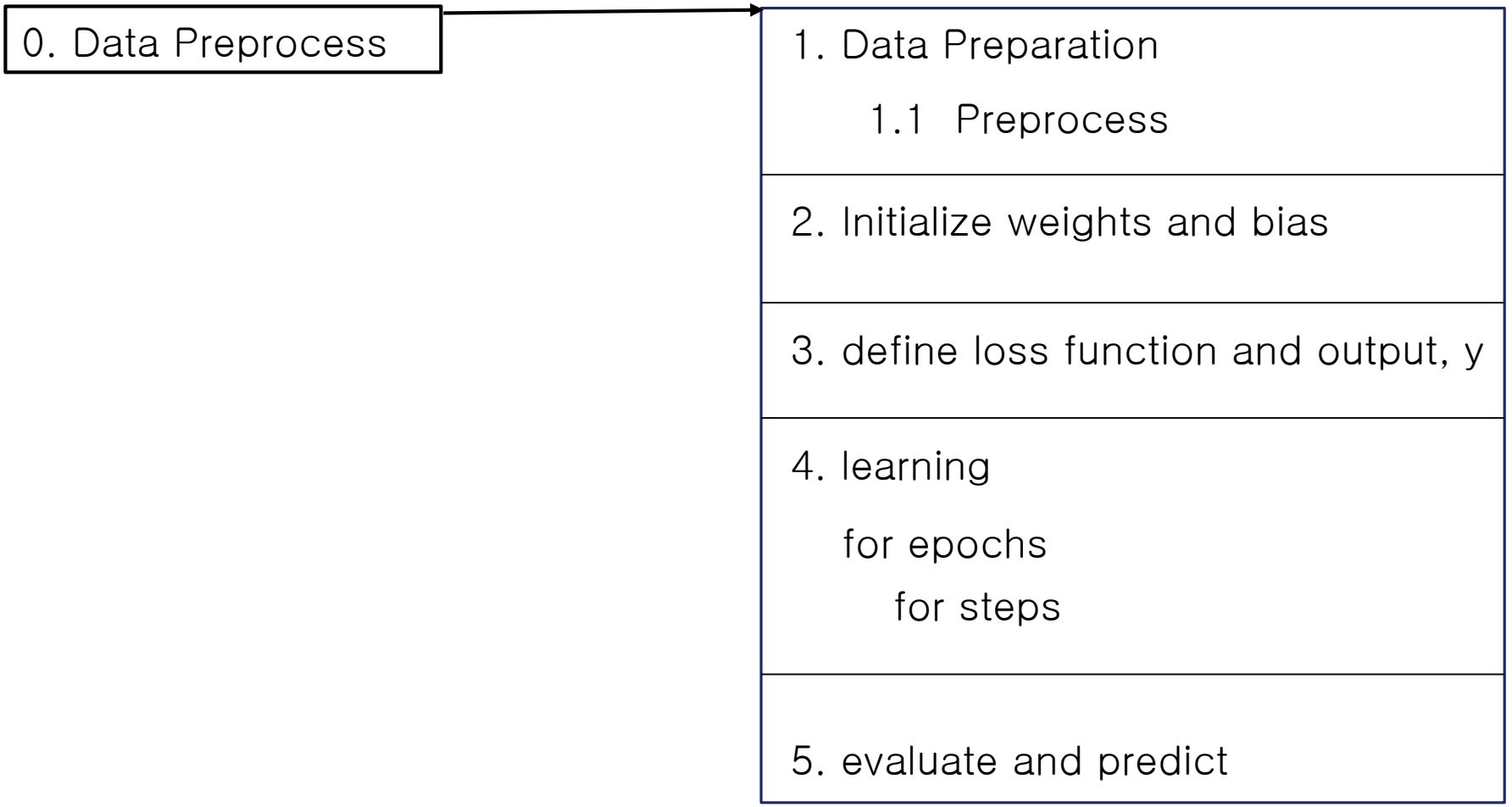
[예제 3] 다음과 같은 행렬을 입력으로 받는 4변수 함수 f(w, x, y, z) 에서 주어진 값에서의 미분 값을 구하시오

$$\begin{pmatrix} w & x \\ y & z \end{pmatrix} = \begin{pmatrix} 1.0 & 2.0 \\ 3.0 & 4.0 \end{pmatrix}$$

$$f(w,x,y,z) = wx + xyz + 3w + zy^2$$

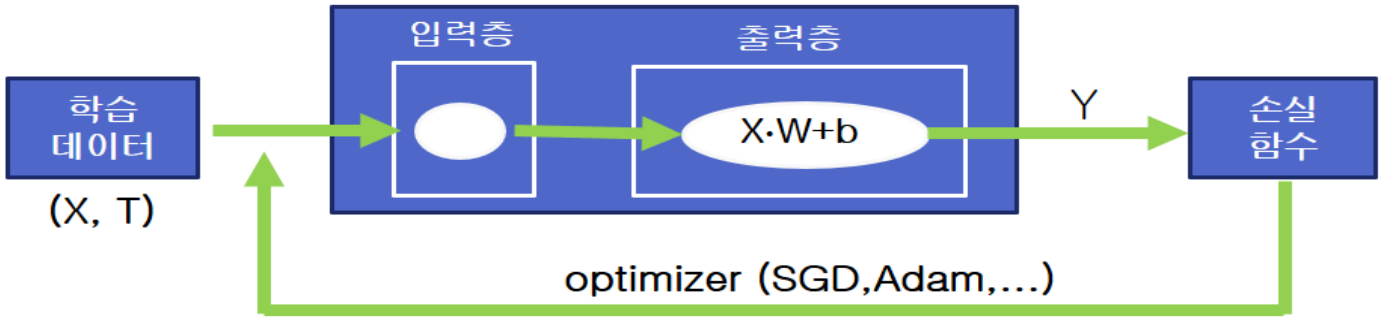
참고 동영상 강의: https://youtu.be/g4YmXwVttVg

머신러닝 / 딥러닝 일반적인 개발 프로세스

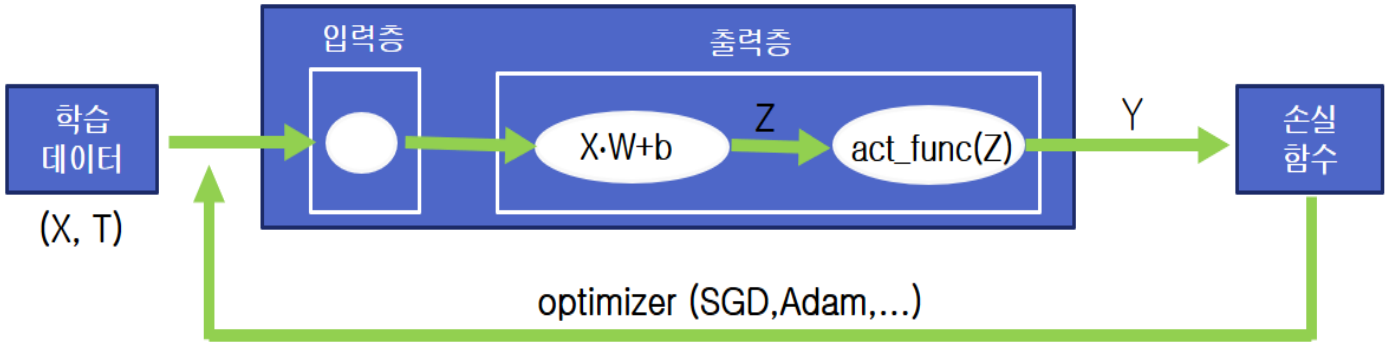


아키텍처 비교

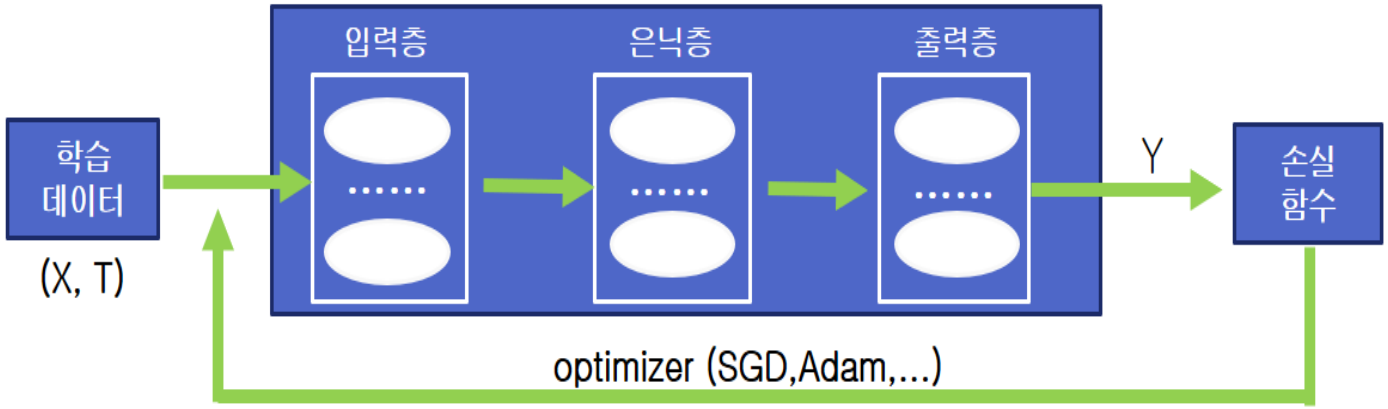
Linear Regression



Logistic Regression



Deep Learning



[예제] Linear Regression

주어진 sps.csv 파일은 1열은 정답데이터, 2열부터 마지막열까지 입력데이터이다.

[Q1] training data 는 어떤 특징(패턴)이 있는지 파악하시오

[Q2] 특징(패턴)을 파악하였다면, 머신러닝의 Linear Regression 을 이용하여 학습할 경우 가중치와 바이어스는 각각 몇 개인지 확인하시오

[Q3] 수치미분을 이용하여 손실함수가 최소가 되도록 가중치와 바이어스를 업데이트하시오

[Q4] 다음과 같은 4개의 값을 주어 데이터 패턴을 제대로 학습했는지 확인하시오

(4, 4, 4, 4) , (-3, 0, 9, -1) , (-7, -9, -2, 8) , (1, -2, 3, -2) ,
(19, -12, 0, -76) , (2001, -1, 109, 31) , (-1, 102, -200 , 1000)

[예제] Logistic Regression

공부시간 (x)	Fail/Pass (t)
2	0
4	0
6	0
8	0
10	0
12	0
14	1
16	1
18	1
20	1

training data

다음의 training data를 바탕으로 수치 미분을 이용하여 Logistic Regression 시스템을 구현한 후,

테스트 데이터 13.0, 11.0, 31.0, 3.0, 3.5 에 대하여 미래 값을 예측하시오

Review – feed forward · one hot encoding

- feed forward
  - 신경망의 입력 층(input layer)으로 데이터가 입력되고, 1개 이상으로 구성되는 은닉층(hidden layer)을 거쳐서 출력 층(output layer)으로 출력 값을 내보내는 과정

- one hot encoding
  - 정답을 표현하는 방식으로서, 정답에 해당하는 리스트의 인덱스 값에는 1 을 넣고, 나머지 인덱스에는 모두 0을 넣어 정답을 표현하는 기법

[예] 학습 데이터 정답이 tdata = np.array([0, 1, 2]) 를 가지는 경우,
이러한 정답을 one-hot encoding 방식으로 나타내면 다음과 같음

정답 0 => [1, 0 ,0]
정답 1 => [0, 1, 0]
정답 2 => [0, 0, 1]

이러한 one-hot encoding 은 tensorflow.keras.utils.to_categorical(label, num_classes=...) 함수를 통해서 쉽게 실행할 수 있음

[예제] 딥러닝 아키텍처 설계 예시

① Training Data 가 다음과 같을 때, 딥러닝 아키텍처를 설계하시오

```
input_data = [ [1, 1], [4, 3] ]
target_data = [ 0, 1 ]
```

② Training Data 가 다음과 같을 때, 딥러닝 아키텍처를 설계하시오

```
input_data = np.array([ [1, 2, 3, 4], [0, 1, 2, 0], [0, 0, 1, 1] ])
target_data = np.array([ 0, 1, 0 ])
```

[예제] Feed Forward

입력데이터 input_data = np.array([1, 2])

정답데이터 target_data = np.array([1])

학습율 learning_rate = 1e-1

3개의 노드로 구성된 1개의 은닉층 (hidden layer)을 가지는 신경망 (Neural Network)에서 피드포워드 (feed forward) 1회 진행할 경우,

① 은닉층과 출력층 각 노드에서의 선형회귀 값 (Z), 시그모이드 계산 값 (A) 확인

② loss function 은 MSE 라고 할 경우, 신경망 출력값 Y, 오차 loss_val 계산

참고 동영상 강의: https://youtu.be/WstuEsvJJLA