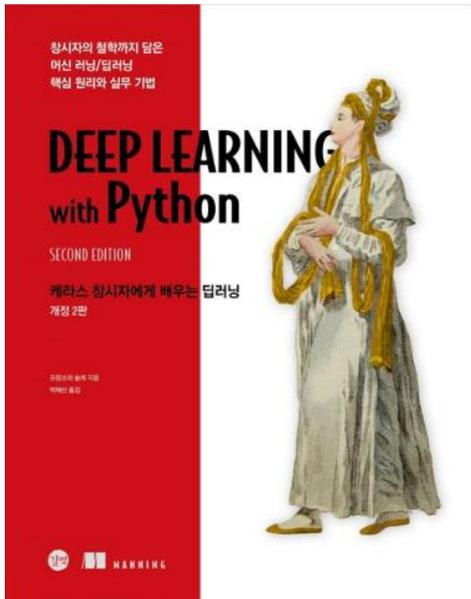


# 머신러닝과 딥러닝

연준모



1. 딥러닝과 머신러닝
2. 라이브러리와 최근 동향
3. 환경 구축하기
4. 기본 개념 알고 가기
5. MLP 수식으로 열어보기
6. Numpy로 MLP 구현하기

<https://wikidocs.net/book/2155>

<https://arxiv.org/>

<https://www.deeplearningbook.org/>

**Deep Learning**

An MIT Press book

# 1. 딥러닝과 머신러닝 머신러닝이란?

## 알고리즘과 머신러닝의 차이

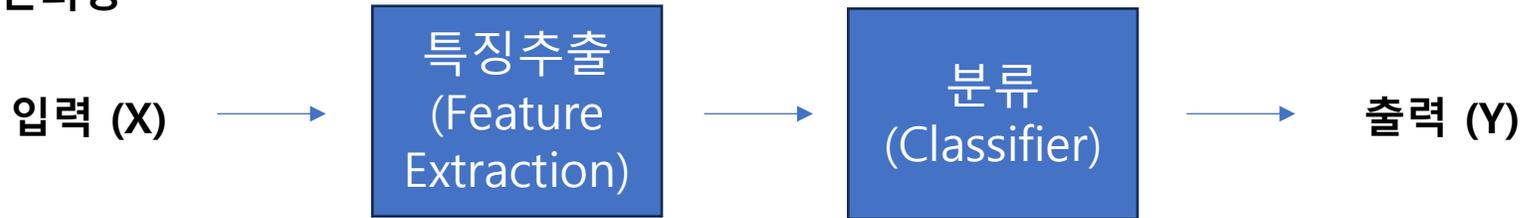


데이터와 답을 통한 훈련으로  
규칙을 찾아내는 것이  
머신러닝

# 1. 딥러닝과 머신러닝 머신러닝과 딥러닝의 차이

## 머신러닝과 딥러닝의 차이

### ✓ 머신러닝



특징추출 위한 알고리즘을  
사람이 제공  
해당 분야에 대한 많은 지식 필요

학습

- ✓ 딥러닝은 머신러닝 방법 중 하나
- ✓ 딥러닝은 특징추출이 필요 없다

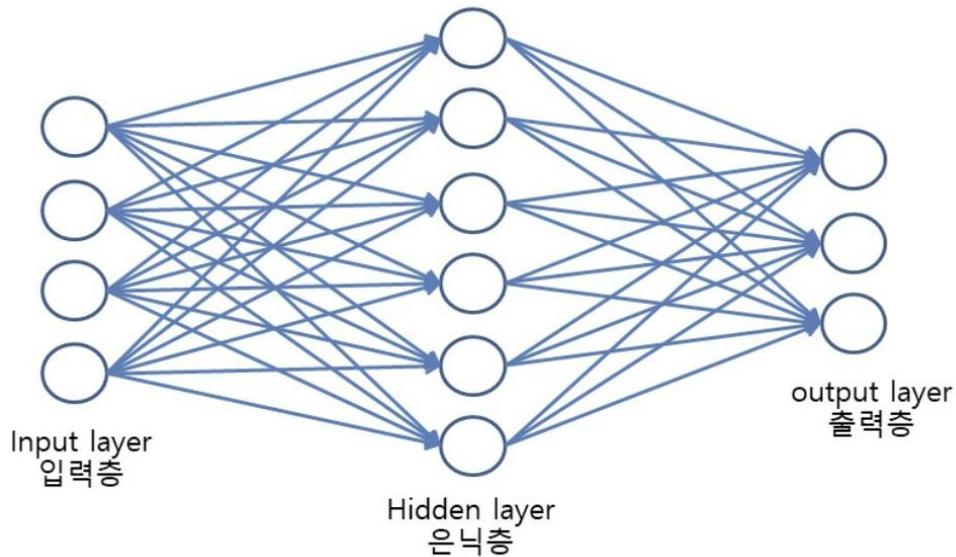
### ✓ 딥러닝



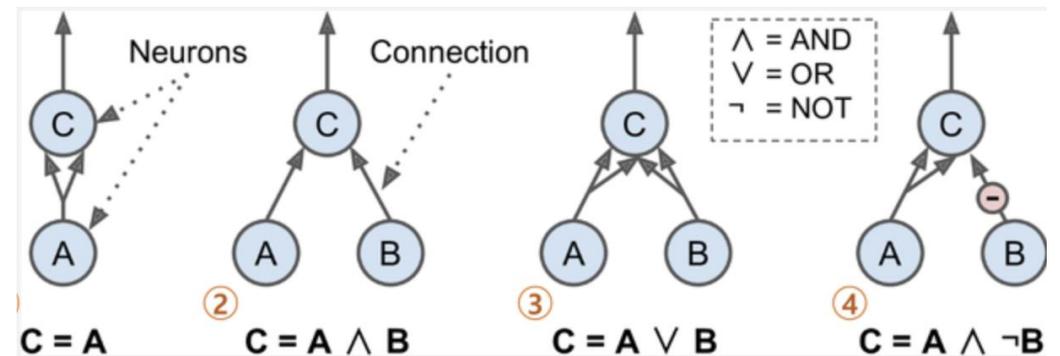
학습

# 1. 딥러닝과 머신러닝 어떻게 생겼을까?

## ✓ MLP 구조 (하나의 은닉층)



### 은닉층 노드에서 일어나는 연산



### 들어온 값을 하나로 만든다

올바른 가중치 값을 얻기 위해(규칙) 딥러닝 한다  
딥러닝이 잘 되면, 더 정답에 가까운 가중치(규칙)를 얻게 된다.

## 2. 라이브러리와 최근 동양 요즘은 어떨까?

ONECLICK AI

✓ 주로 사용되는 라이브러리

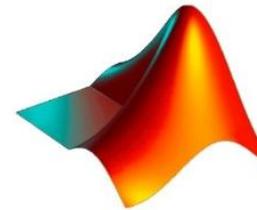
 TensorFlow

 PyTorch

딥러닝을 돌리기 위한 레이어들이  
구현되어 있는 라이브러리

✓ 많이 사용되는 언어

 python

 MATLAB®

라이브러리들을 실행하기 위한 언어

# 3. 환경 구축 파이썬 설치해보기

## 1. 먼저, 파이썬을 설치하자

<https://www.python.org/downloads/release/python-3124/>

- ✓ 꼭 환경변수 설정하기(이미 체크되어 있다 아마도)
- ✓ 설치되면 pip도 함께 깔린다.
- ✓ Cmd에 들어가서, 다음 명령어를 실행해 보자!

```
pip --version  
python --version
```

- ✓ 아래처럼 나오면 성공!

```
C:\Users\연준모>pip --version  
pip 25.1.1 from C:\Users\연준모\AppData\Local\Programs\Python\Python312\Lib\site-packages\pip (python 3.12)  
  
C:\Users\연준모>python --version  
Python 3.12.4
```

### 3. 환경 구축 tensorflow 설치하기

#### 2. Cmd에서 다음 명령어를 실행

```
| pip install tensorflow
```

✓ 아래처럼 나오면 성공!

✓ 이어서, txt 파일 생성, .py 로 저장, vs코드로 연 후,  
다음 코드를 넣고, 실행시켜 보자

```
import tensorflow as tf  
print(tf.__version__)
```

```
(myenv) C:\Users\윤준모\Desktop\AI 커리큘럼\test>python  
Python 3.12.4 (tags/v3.12.4:8e8a4ba, Jun 6 2024, 19:30:16) [MSC v.1940 64 bit (AMD64)] on win32  
Type "help", "copyright", "credits" or "license" for more information.  
>>> import tensorflow as tf  
2025-09-14 15:12:55.179374: I tensorflow/core/util/port.cc:153] oneDNN custom operations are on.  
fferent numerical results due to floating-point round-off errors from different computation order  
t the environment variable `TF_ENABLE_ONEDNN_OPTS=0`.  
2025-09-14 15:12:59.524003: I tensorflow/core/util/port.cc:153] oneDNN custom operations are on.  
fferent numerical results due to floating-point round-off errors from different computation order  
t the environment variable `TF_ENABLE_ONEDNN_OPTS=0`.  
>>> print(tf.__version__)  
2.20.0
```

만일, python 가상환경을 사용하고 싶다면?

[https://gihak111.github.io/pynb/2024/08/25/python\\_virtual\\_environment\\_upload.html](https://gihak111.github.io/pynb/2024/08/25/python_virtual_environment_upload.html)

vs 코드 설치해야 한다면?

<https://code.visualstudio.com/download>

# 3. 환경 구축 간단한 코드 실행해 보자

## 3. 다음 링크에서, 코드를 다운받자

[https://huggingface.co/OneclickAI/CNN\\_test\\_Model/tree/main](https://huggingface.co/OneclickAI/CNN_test_Model/tree/main)

Test.py 다운받기

✓ 이어서, vs코드로 열고, 실행을 위해 다음 명령어를 cmd에서 실행

```
pip install huggingface_hub
```

✓ 설치가 다 된 후, 아무 이미지나 넣어서 테스트 해 보자

```
Please enter the path to your image (or type 'exit' to quit): C:\Users\연준모\Desktop\1234.jpg
1/1 _____ 0s 76ms/step

--- Prediction Result ---
Predicted Digit: 8
Confidence: 59.83%
-----
```

✓ 다음과 같이 나오면, 모델 사용 성공!

### 3. 환경 구축 간단한 코드 실행해 보자

- ✓ 많은 연산이 들어가는 딥러닝은 많은 코어가 필요하다!
- ✓ 코어수가 압도적으로 많은 GPU가 CPU보다 더 유리!
- ✓ 이런 GPU에서 딥러닝 하기 위해 CUDA를 설치해야 한다

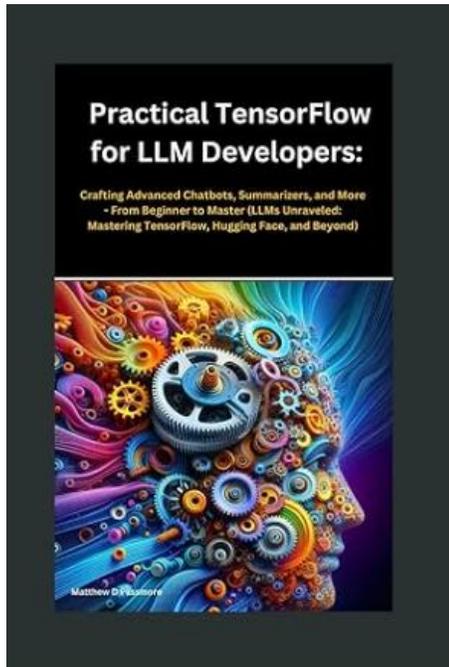


### 4. 다음 링크를 따라 해보자

[https://gihak111.github.io/pynb/2024/11/08/cuda\\_upload.html](https://gihak111.github.io/pynb/2024/11/08/cuda_upload.html)

## 4. 기본 개념 알고가기 들어가기 앞서 책 추천?

ONECLICK AI



원서라 읽기 귀찮음

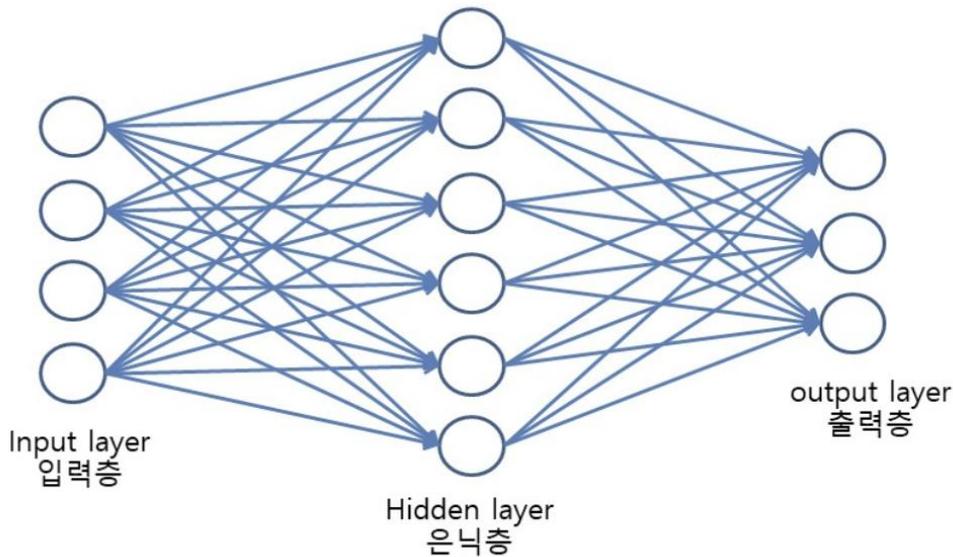
25000원 이라는 비싼 가격

그래도 입문용으로 좋지 않을까 해서 추천해 봅니다

딥러닝에 사용되는 단어, 개념을 쉽고 짧게 정리해논 책  
하지만, 너무 짧고 얇은 설명이라 조금만 알아도 쓸모가 없다

# 4. 기본 개념 알고가기 입출력과 가중치?

## ✓ 입력과 출력? 은닉층?



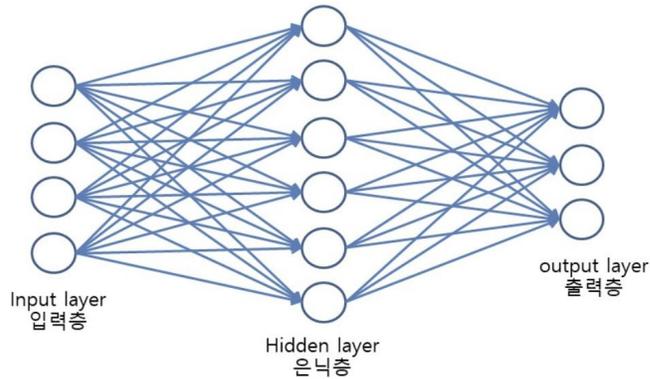
- Input\_size : 입력층 노드 수
- Hidden\_size : 은닉층 노드 수
- Output\_size : 출력층 노드 수
  
- W : 가중치 : 노드 간 곱해지는 값
- b : 편향 : 노드 간 더해지는 값

$$y = wx + b$$

- Epoch : 전체 학습 라인을 한 번 반복하여 학습하는 단위

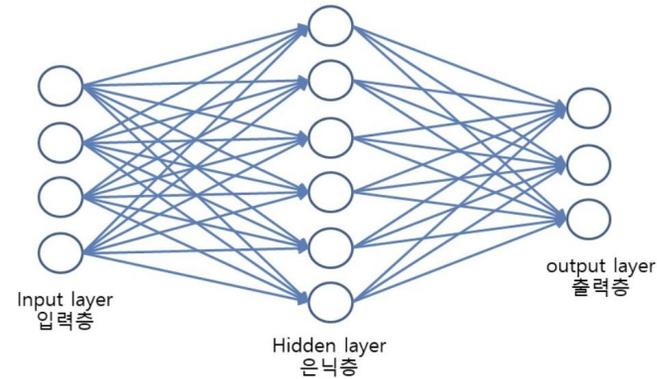
# 4. 기본 개념 알고가기 순전파와 역전파

## 순전파



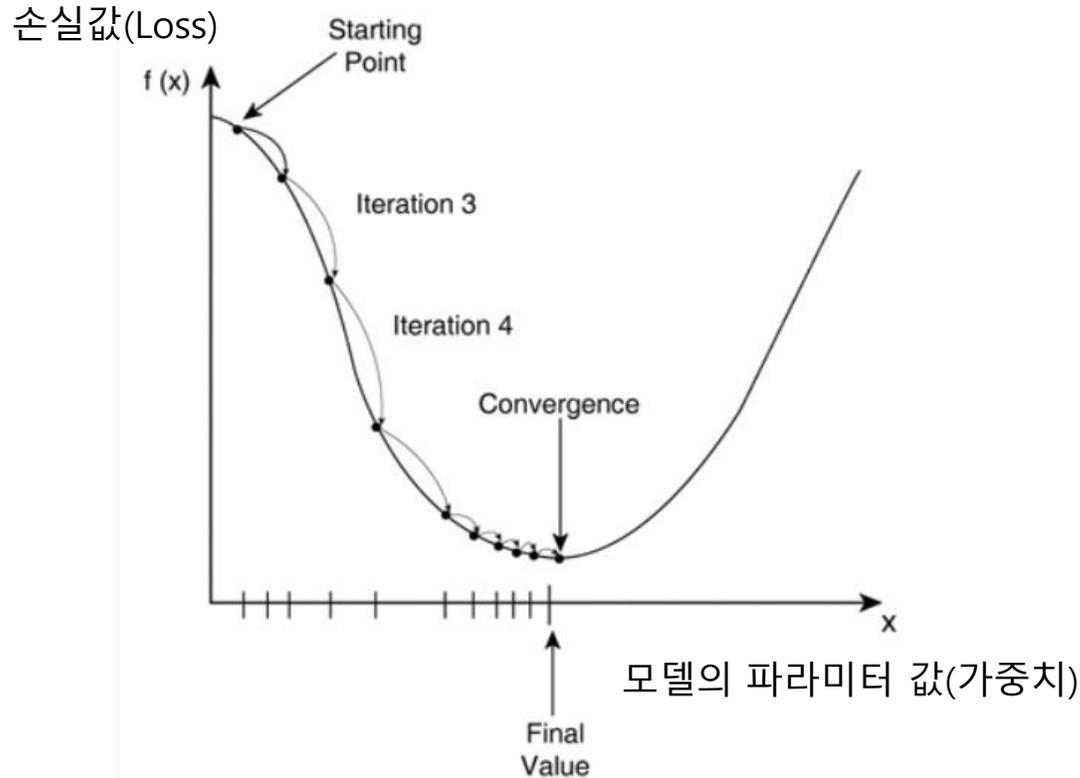
- 입력 데이터가 신경망의 층 들을 순서대로 통과
- 예측값을 계산하는 과정

## 역전파



- 예측값과 실제값의 오차를 계산
- 오차가 각 층에 기여한 정도를 역순으로 전달, 가중치 업데이트 한다

# 4. 기본 개념 알고가기 경사하강법



오차를 최소화 하는 방향으로  
가중치, 편향을 업데이트 한다

$$w_{\text{new}} = w_{\text{old}} - \eta \nabla J(w)$$

새로운 위치 = 현재 위치 + 움직인 방향과 거리

$w_{\text{new}}$  : 업데이트 된 가중치. 다음 학습에서 사용된다.

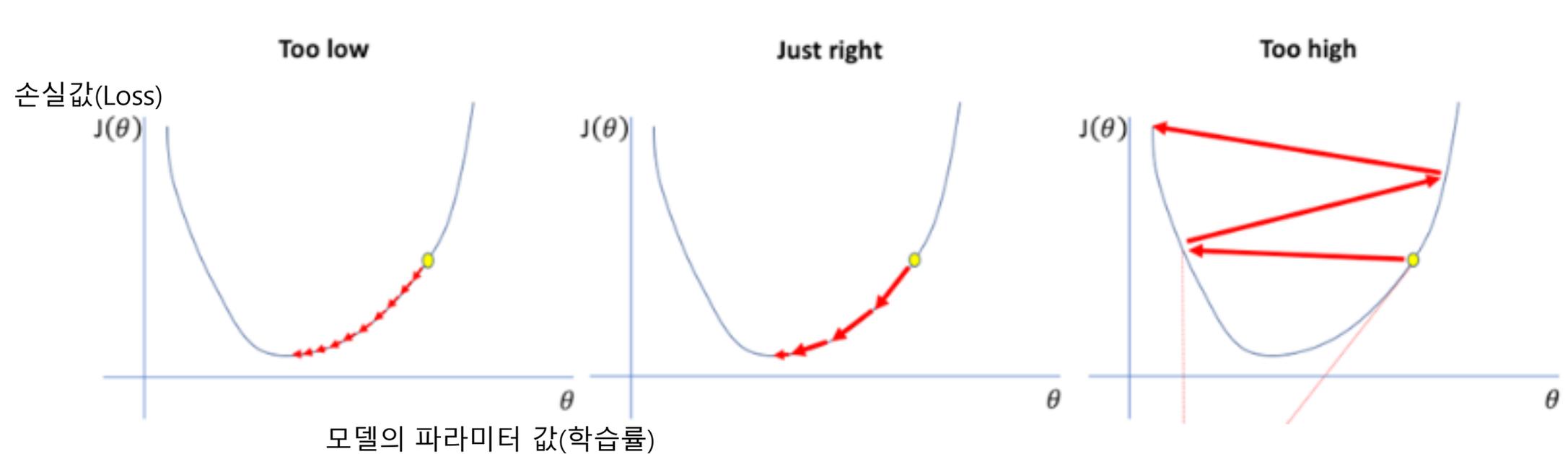
$w_{\text{old}}$  : 현재 가중치. 출발하는 값.

$\eta$  : 학습률. 경사를 얼마나 많이 이동할 지 정한다.

$\nabla J(w)$  : 현재 위치에서 기울기가 가장 가파른 방향.

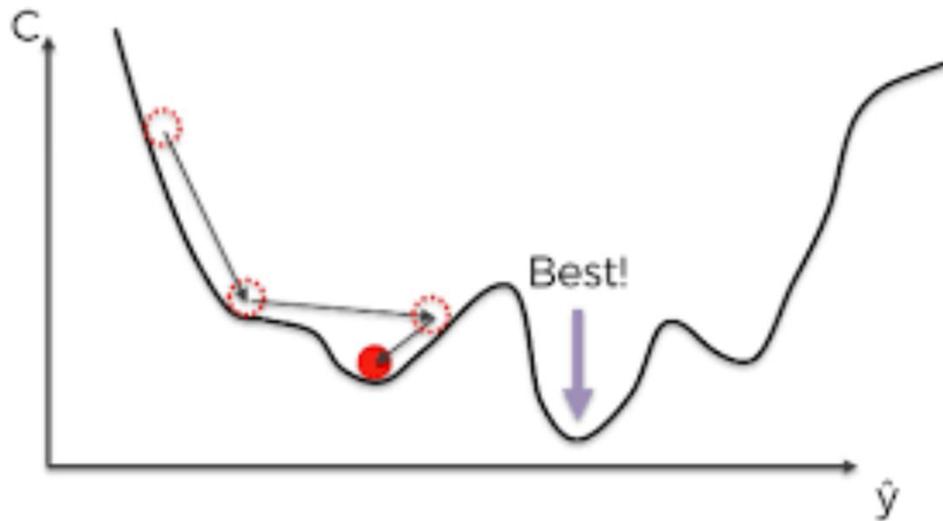
여기에 사용되는 기울기를 역전파가 알려준다

# 4. 기본 개념 알고가기 학습률



- 경사를 따라 얼마나 크게 이동할지 정하는 하이퍼파라미터 값.
- 적절한 값을 찾지 않으면 딥러닝이 망한다.

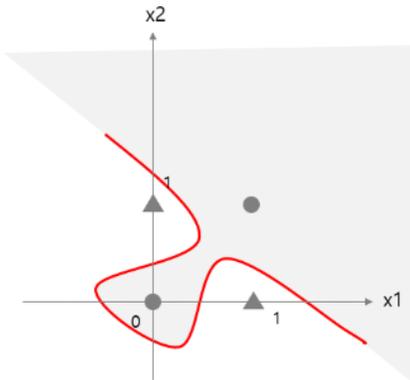
## 4. 기본 개념 알고가기 학습률과 경사하강법



- ✓ 학습률이 너무 작으면 지역 최솟점으로 수렴해 빠져 나오지 못한다
- ✓ 전역 최솟점으로 가기 위해 적절한 값을 설정해야 한다

## ✓ 선형과 비선형?

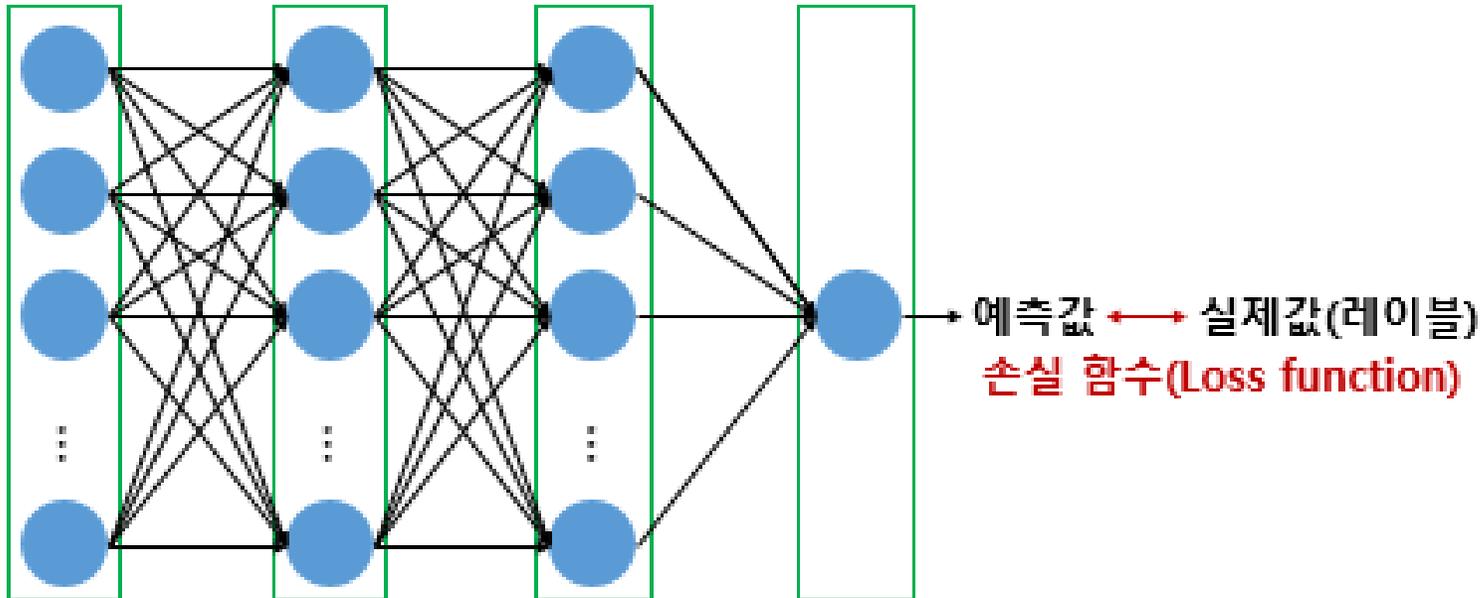
구분	선형(Linear)	비선형(Nonlinear)
관계	직선적인 비례 관계	곡선 또는 복잡한 관계
원리	중첩의 원리 만족	중첩의 원리 만족 안함
방정식	$y = ax$ 형태	$y = ax^2, y = \sin(x)$ 등
분석	비교적 간단	매우 복잡, 예측 어려움
예시	축의 법칙, 저항	혼돈 이론, 카메라 렌즈 왜곡



- ✓ 각 층의 노드 출력에 비선형성을 추가하는 함수
- ✓ 대표적으로, Sigmoid, Relu 등이 있다.
- ✓ 선형적인 딥러닝은 활성화 함수 없이 가능하다
- ✓ 비선형 적이면 활성화 함수가 들어야 한다
- ✓ 예시로, 활성화 함수 없으면 xor 문제를 학습할 수 없다.

# 4. 기본 개념 알고가기 손실함수

✓ 오차를 최소화하는 방향으로 모델을 학습시키기 위함 이다

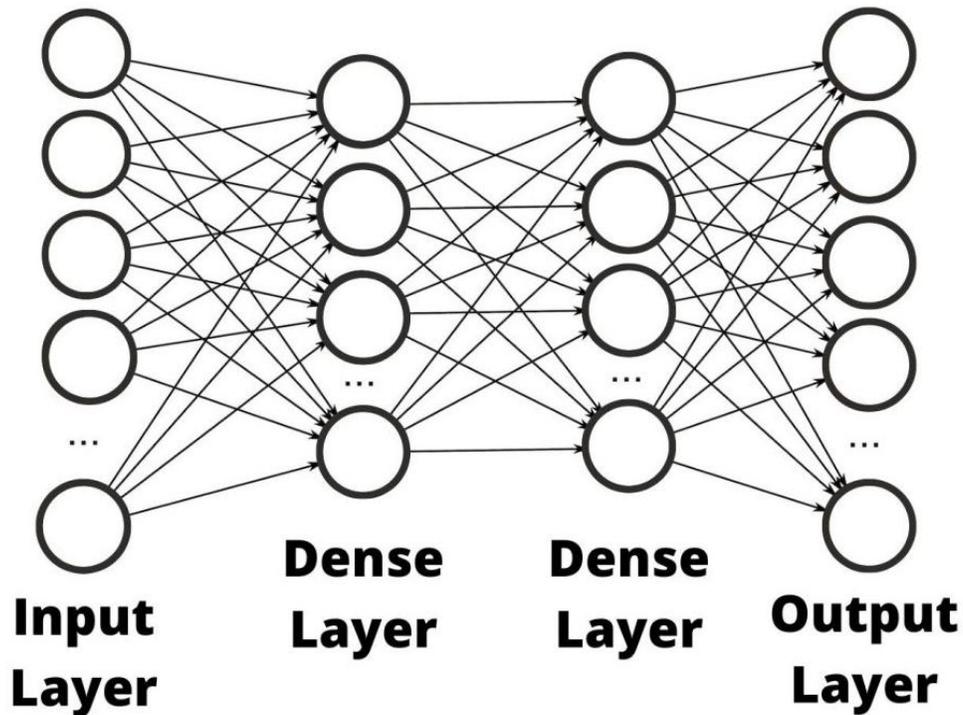


- ✓ 모델의 예측값과 실제값 간의 차이(오차)를 측정하는 함수
- ✓ MSE, MAE 등이 있다.

✓ 자세한 건 나중에 알아보자

# 4. 기본 개념 알고가기 Dense Layer

✓ 완전연결 레이어인 Dense Layer



✓ 모든 노드가 다음 노드에 전부 다 연결된 형태

✓ MLP에서는 입력, 은닉, 출력층 모두 완전연결 이므로, 전부 Dense Layer 이다

✓ 이것 말고도 엄청나게 많은 레이어

✓ 일단, 이것만 알고 넘어가자

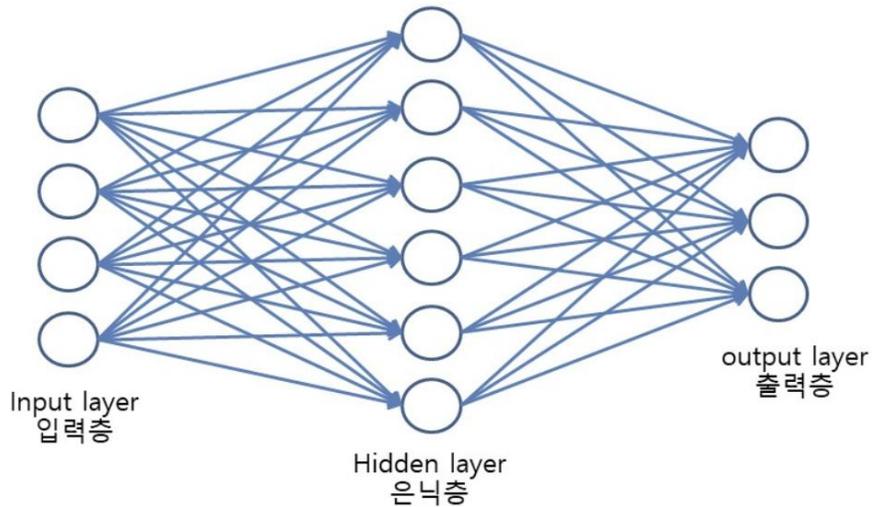
# 5. MLP 수식으로 열어보기

딥러닝이 진행되는 단계?

ONECLICK AI



1. 순전파 : 예측값 계산



3. 가중치 및 편향 업데이트



2. 역전파 : 기울기 계산

# 5. MLP 수식으로 알아보기 1. 순전파 : 예측값 계산?

✓ 입력 데이터가 신경망의 각 층을 통과하며 예측값을 계산하는 과정

순전파 1단계. 은닉층 계산

$$Z_1 = XW_1 + b_1$$

$$A_1 = \sigma(Z_1)$$

$X$       입력 데이터 행렬

$\sigma$     활성화 함수(시그모이드)

$W_1$     입력층과 은닉층 사이의 가중치 행렬

$A_1$     은닉층의 활성화 값 행렬

$b_1$     은닉층의 편향 벡터

$Z_1$     은닉층의 가중 입력 행렬

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

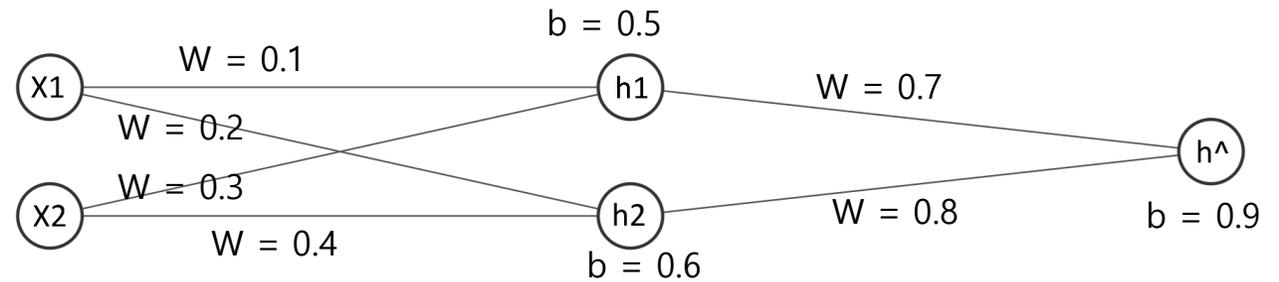
# 5. MLP 수식으로 열어보기 1. 순전파 : 예측값 계산?

## ✓ 숫자를 통해 따라가보자

$X = [1, 0]$ , 실제 정답  $y = [1]$

입력층 2개, 은닉층 2개, 출력층 1개

학습률 = 0.1



$W_1$

- (입력층  $\rightarrow$  은닉층):  $[[0.1, 0.2], [0.3, 0.4]]$ 
  - 가중치 행렬의 크기 = (이전 층의 노드 수, 다음 층의 노드 수)

$b_1$

- (은닉층 편향):  $[0.5, 0.6]$

$W_2$

- (은닉층  $\rightarrow$  출력층):  $[[0.7], [0.8]]$

$b_2$

- (출력층 편향):  $[0.9]$

# 5. MLP 수식으로 열어보기 1. 순전파 : 예측값 계산?

✓ 숫자를 통해 따라가보자

순전파 1단계. 은닉층 계산

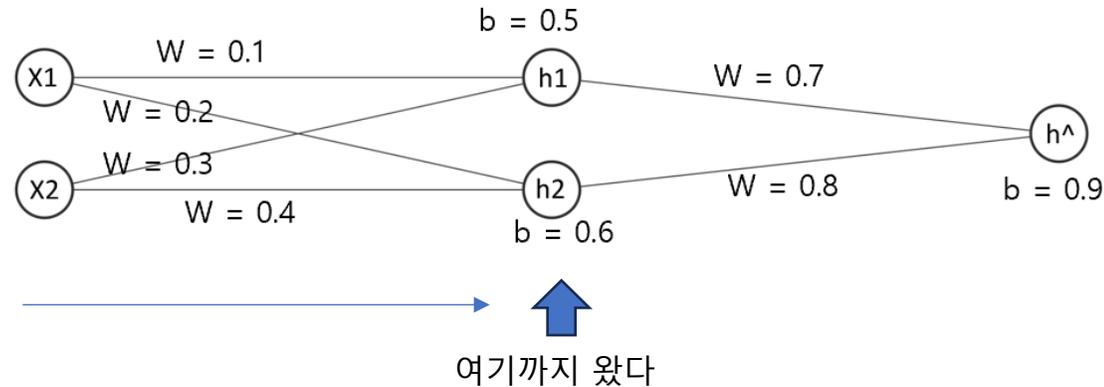
$$Z_1 = XW_1 + b_1$$

$$Z_1 = [1, 0] @ [[0.1, 0.2], [0.3, 0.4]] + [0.5, 0.6]$$

$$Z_1 = [0.1, 0.2] + [0.5, 0.6] = [0.6, 0.8]$$

$$A_1 = \sigma(Z_1)$$

$$A_1 = \text{sigmoid}([0.6, 0.8]) = [0.646, 0.690]$$



# 5. MLP 수식으로 알아보기 1. 순전파 : 예측값 계산?

## 순전파 2단계. 출력층 계산

$$Z_2 = A_1 W_2 + b_2$$

$$\hat{y} = \sigma(Z_2)$$

$A_1$  은닉층 활성화 행렬

$\hat{y}$  최종 예측값 행렬

$W_2$  은닉층과 출력층 사이의 가중치 행렬

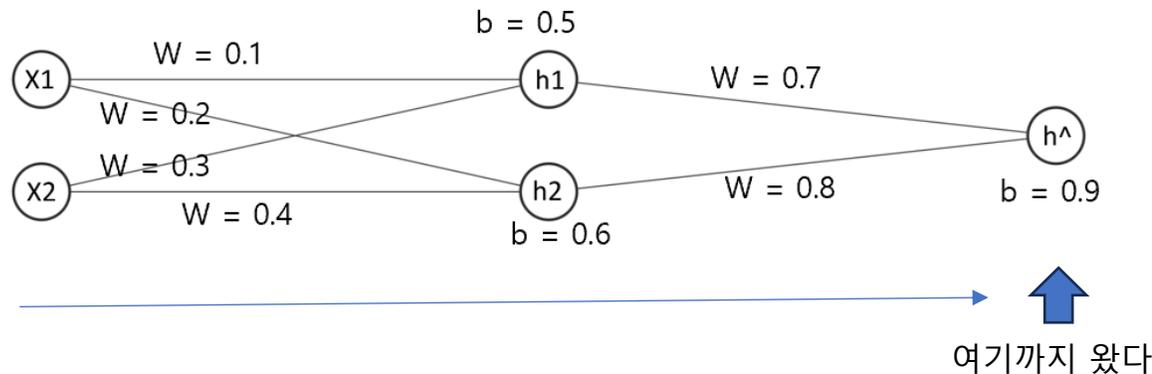
$b_2$  출력층의 편향 벡터

$Z_2$  출력층의 가중 입력 행렬

# 5. MLP 수식으로 열어보기 1. 순전파 : 예측값 계산?

✓ 숫자를 통해 따라가보자

순전파 2단계. 출력층 계산



$$Z_2 = A_1 W_2 + b_2$$

$$Z_2 = [0.646, 0.690] @ [[0.7], [0.8]] + [0.9]$$

$$Z_2 = [0.4522 + 0.552] + [0.9] = [1.0042] + [0.9] = [1.9042]$$

$$\hat{y} = \sigma(Z_2)$$

$$\hat{y} = \text{sigmoid}(1.9042) = [0.870]$$

즉, 0.130 의 오차가 발생

# 5. MLP 수식으로 알아보기 2. 역전파 : 기울기 계산?

✓ 예측값과 실제값의 차이를 바탕으로, 각 가중치와 편향을 얼마나 변경해야 할 지 나타내는 기울기를 계산하는 과정      기울기 : Gradient

역전파 1단계. 출력층의 오차 델타 계산

$$E_2 = y - \hat{y}$$

$y$       실제 정답 레이블 행렬

$\hat{y}$       예측값 행렬

$E_2$       출력층의 오차 행렬

$$\delta_2 = E_2 \odot \sigma'(\hat{y})$$

$\odot$       요소별 곱셈

$\sigma'(\hat{y})$       시그모이드 함수의 미분 값

$\delta_2$       출력층의 델타 행렬

W2와 b2에 대한 기울기를 계산하는 데 사용

실제론 손실함수 사용한다. 이번엔 간단한 방법을 사용하겠다.

미분을 통해서 기울기 값 얻는다. 손실값이 얼마나, 어느 방향으로 변하는지 알 수 있다.

# 5. MLP 수식으로 열어보기 2. 역전파 : 기울기 계산?

✓ 숫자를 통해 따라가보자

역전파 1단계. 출력층의 오차 델타 계산

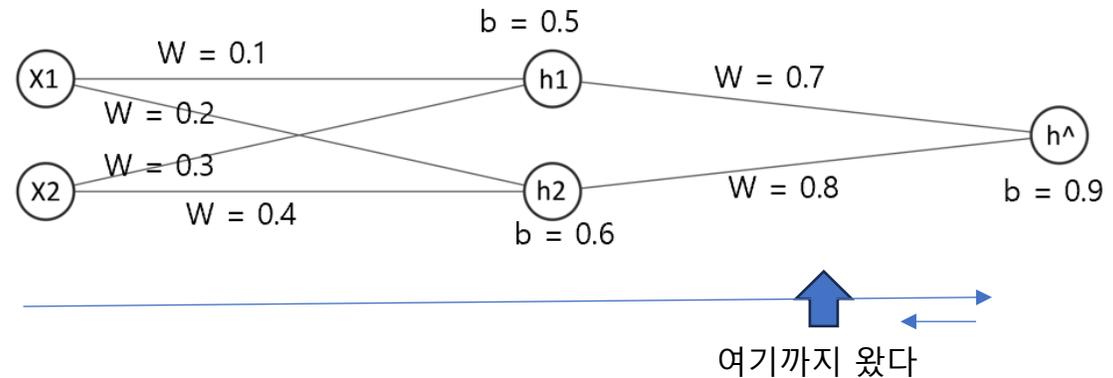
$$E_2 = y - \hat{y}$$

$$E_2 = [1] - [0.870] = [0.130]$$

$$\delta_2 = E_2 \odot \sigma'(\hat{y})$$

$\sigma'(x) = \sigma(x) * (1 - \sigma(x))$  이므로,  $\sigma'(0.870) = 0.870 * (1-0.870) = 0.113$

$$\delta_2 = [0.130] * [0.113] = [0.0147]$$



# 5. MLP 수식으로 알아보기 2. 역전파 : 기울기 계산?

⊙ 하드마르 곱이 더 궁금하다면? [https://gihak111.github.io/ai/2025/09/12/Hadamard\\_product\\_upload.html](https://gihak111.github.io/ai/2025/09/12/Hadamard_product_upload.html)

## 역전파 2단계. 은닉층의 오차 델타 계산

$$E_1 = \delta_2 W_2^T$$

$W_2^T$  실제 정답 레이블 행렬

$E_1$  예측값 행렬

$$\delta_1 = E_1 \odot \sigma'(A_1)$$

$\sigma'(A_1)$  은닉층 활성화 값  
A1의 시그모이드 미분 값

$\delta_1$  은닉층의 델타 행렬  
W1과 b1에 대한 기울기를  
계산하는 데 사용

# 5. MLP 수식으로 열어보기 2. 역전파 : 기울기 계산?

✓ 숫자를 통해 따라가보자

역전파 2단계. 은닉층의 오차 델타 계산

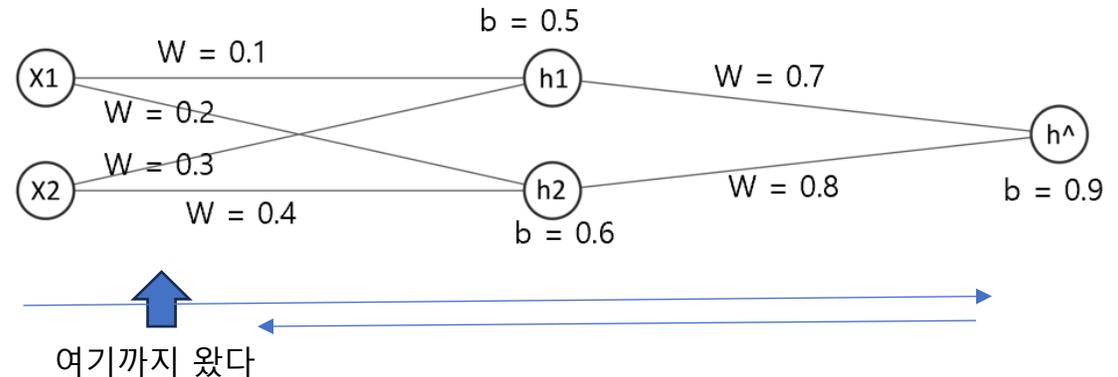
$$E_1 = \delta_2 W_2^T$$

$$E_1 = [0.0147] @ [[0.7, 0.8]] = [0.0103, 0.0118]$$

$$\delta_1 = E_1 \odot \sigma'(A_1)$$

$$\sigma'(A_1) = [0.646*(1-0.646), 0.690*(1-0.690)] = [0.229, 0.214]$$

$$\delta_1 = [0.0103, 0.0118] * [0.229, 0.214] = [0.0024, 0.0025]$$



# 5. MLP 수식으로 알아보기 3. 경사하강법 : 업데이트? ONECLICK AI

✓ 역전파로 계산된 기울기를 통해 가중치, 편향 업데이트

출력층 -> 은닉층 가중치 업데이트

$$W_{2,new} = W_{2,old} + \eta(A_1^T \delta_2)$$

$$b_{2,new} = b_{2,old} + \eta \sum \delta_2$$

$W_{2,new}$       업데이트 된 출력층 가중치

$\sum \delta_2$       행렬의 모든 행을 더한 값

$W_{2,old}$       현재 출력층 가중치

$A_1^T$       은닉층 활성화 값 A1의 전치

$\eta$       학습률

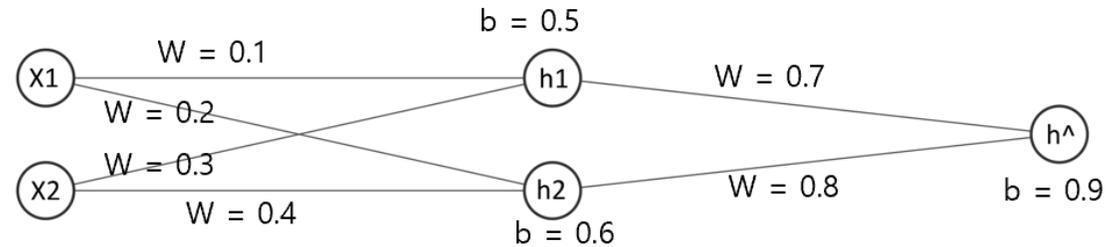
전치란? 행렬의 행과 열을 서로 맞바꾸는 연산

$$A = \begin{pmatrix} 1 & 2 \\ 3 & 4 \\ 5 & 6 \end{pmatrix} \quad \rightarrow \quad A^T = \begin{pmatrix} 1 & 3 & 5 \\ 2 & 4 & 6 \end{pmatrix}$$

# 5. MLP 수식으로 열어보기 3. 경사하강법 : 업데이트? ONECLICK AI

✓ 숫자를 통해 따라가보자

출력층 -> 은닉층 가중치 업데이트



$$W_{2,new} = W_{2,old} + \eta(A_1^T \delta_2)$$

업데이트 양 =  $0.1 * [[0.646], [0.690]] @ [0.0147] = [[0.00095], [0.00101]]$  @ = 행렬곱셈

$W_{2,new} = [[0.7], [0.8]] + [[0.00095], [0.00101]] = [[0.70095], [0.80101]]$

$$b_{2,new} = b_{2,old} + \eta \sum \delta_2$$

업데이트 양 =  $0.1 * [0.0147] = [0.00147]$

$b_{2,new} = [0.9] + [0.00147] = [0.90147]$

# 5. MLP 수식으로 알아보기 3. 경사하강법 : 업데이트? ONECLICK AI

은닉층 -> 입력층 가중치 업데이트

$$W_{1,new} = W_{1,old} + \eta(X^T \delta_1)$$

$W_{1,new}$  업데이트 된 은닉층 가중치

$W_{1,old}$  현재 은닉층 가중치

$\delta_1$  은닉층의 델타 행렬

$X^T$  입력 데이터에서 행렬 X의 전치

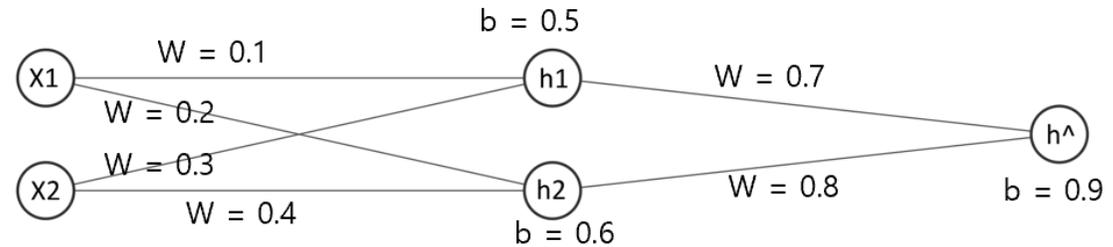
$$b_{1,new} = b_{1,old} + \eta \sum \delta_1$$

$\sum \delta_1$  행렬의 모든 행을 더한 값

# 5. MLP 수식으로 열어보기 3. 경사하강법 : 업데이트? ONECLICK AI

✓ 숫자를 통해 따라가보자

은닉층 -> 입력층 가중치 업데이트



$$W_{1,new} = W_{1,old} + \eta(X^T \delta_1)$$

업데이트 양 =  $0.1 * [[1], [0]] @ [0.0024, 0.0025] = [[0.00024, 0.00025], [0, 0]]$

$W1\_new = [[0.1, 0.2], [0.3, 0.4]] + [[0.00024, 0.00025], [0, 0]] = [[0.10024, 0.20025], [0.3, 0.4]]$

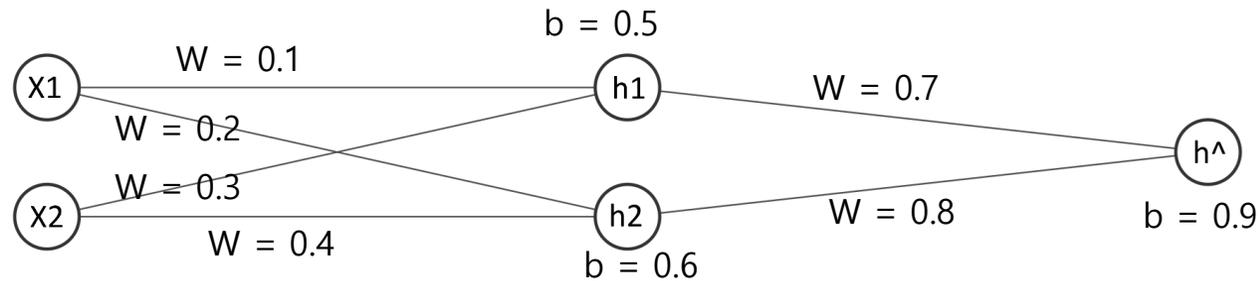
$$b_{1,new} = b_{1,old} + \eta \sum \delta_1$$

업데이트 양 =  $0.1 * [0.0024, 0.0025] = [0.00024, 0.00025]$

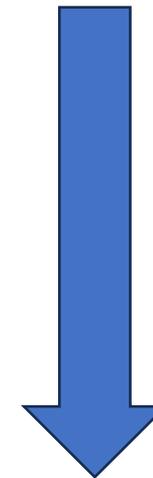
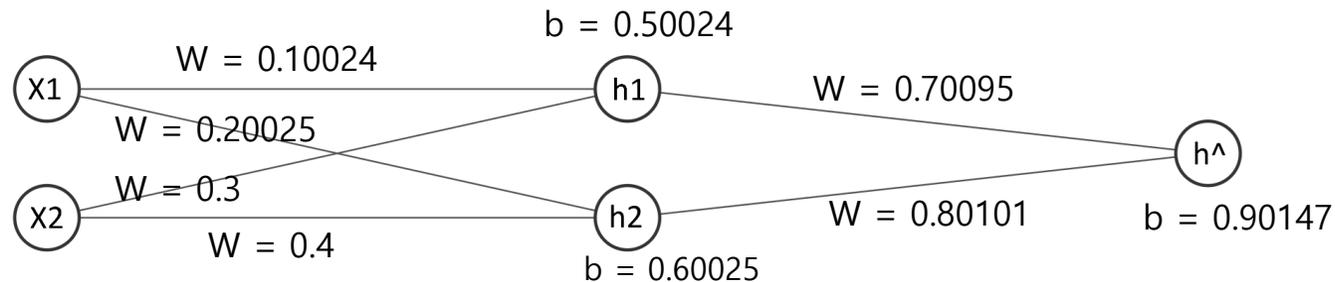
$b1\_new = [0.5, 0.6] + [0.00024, 0.00025] = [0.50024, 0.60025]$

# 5. MLP 수식으로 열어보기 4. 업데이트 전 후 비교

✓ 처음 파라미터 값



✓ 업데이트 후 파라미터 값



- ✓ 이게 Epoch 한번 한 것
- ✓ 이걸 계속 반복해서 가중치와 편향 값을 업데이트 하면
- ✓ 점점 손실값이 줄어들며 딥러닝이 잘 진행된다

## 6. Numpy로 MLP 구현하기 Numpy란?

### ✓ 수학적 계산을 위한 라이브러리 Numpy

```
pip install numpy
```

위 코드를 통해서 설치. 코드와 앞선 개념을 연결해 보자.

[https://huggingface.co/gihakkk/MLP\\_test](https://huggingface.co/gihakkk/MLP_test)

MLP.py 의 내용을 토대로 진행하겠다.

## 6. Numpy로 MLP 구현하기

```
# 하이퍼파라미터 설정
input_size = 2      # 입력층 노드 수
hidden_size = 3     # 은닉층 노드 수
output_size = 1     # 출력층 노드 수
learning_rate = 0.5 # 학습률
epochs = 10000      # 학습 반복 횟수

# 1. 데이터셋 정의 (XOR 문제)
# 입력 데이터: [0,0], [0,1], [1,0], [1,1]
X = np.array([[0, 0], [0, 1], [1, 0], [1, 1]])
# 정답 레이블: [0], [1], [1], [0]
y = np.array([[0], [1], [1], [0]])
```

- ✓ 입력, 출력 등 필요한 파라미터 정의
- ✓ 비선형 XOR 테스트를 위한 데이터셋 정의

## 6. Numpy로 MLP 구현하기

```
# 2. 가중치(w)와 편향(b) 초기화
W1 = np.random.randn(input_size, hidden_size)
b1 = np.random.randn(hidden_size)
W2 = np.random.randn(hidden_size, output_size)
b2 = np.random.randn(output_size)

# 3. 활성화 함수와 그 미분 함수 정의
def sigmoid(x):
    return 1 / (1 + np.exp(-x))

def sigmoid_derivative(x):
    return x * (1 - x)
```

- ✓ 가중치, 편향을 랜덤으로 생성
- ✓ 시그모이드 함수 정의, 미분 함수 정의

# 6. Numpy로 MLP 구현하기

```
# 4. 학습 시작
for epoch in range(epochs):
    # 순전파 (Forward Propagation)
    # 은닉층
    hidden_output = np.dot(X, W1) + b1
    hidden_activation = sigmoid(hidden_output)

    # 출력층
    output_output = np.dot(hidden_activation, W2) + b2
    predicted_output = sigmoid(output_output)

    # 역전파 (Backpropagation)
    # 1단계: 출력층의 오차와 기울기 계산
    error_output = y - predicted_output
    delta_output = error_output * sigmoid_derivative(predicted_output)
```

- ✓ 은닉층, 출력층 순서대로 통과
- ✓ 역전파 1단계로 출력층 오차, 기울기 계산

# 6. Numpy로 MLP 구현하기

```
# 2단계: 은닉층의 오차와 기울기 계산
error_hidden = np.dot(delta_output, W2.T)
delta_hidden = error_hidden * sigmoid_derivative(hidden_activation)

# 3단계: 가중치와 편향 업데이트
W2 += np.dot(hidden_activation.T, delta_output) * learning_rate
b2 += np.sum(delta_output, axis=0) * learning_rate
W1 += np.dot(X.T, delta_hidden) * learning_rate
b1 += np.sum(delta_hidden, axis=0) * learning_rate

# 매 1000번째 epoch마다 오차 출력
if epoch % 1000 == 0:
    loss = np.mean(np.abs(error_output))
    print(f"Epoch: {epoch}, Loss: {loss:.4f}")
```

- ✓ 2단계로 은닉층 오차, 기울기 계산
- ✓ 가중치, 편향 업데이트
- ✓ Epoch 만큼 반복

## 6. Numpy로 MLP 구현하기

```
print("\n--- 학습 완료 ---")

# 5. 학습된 모델로 예측 (결과 확인)
hidden_output_final = np.dot(X, W1) + b1
hidden_activation_final = sigmoid(hidden_output_final)
predicted_final = sigmoid(np.dot(hidden_activation_final, W2) + b2)

print("입력 데이터:\n", X)
print("예측 결과:\n", predicted_final.round())
print("정답 레이블:\n", y)
```

✓ 학습 완료 및 결과 확인

- 딥러닝과 머신러닝의 차이
- 딥러닝에 필요한 개념 정리
- MLP 수식으로 증명
- MLP를 코드로 정의

감사합니다