

기계학습프로그래밍

6강 – Neural Network

한국폴리텍대학 대구캠퍼스
SI엔지니어링학과 강현우

기계학습프로그래밍 - 6강 Neural Network

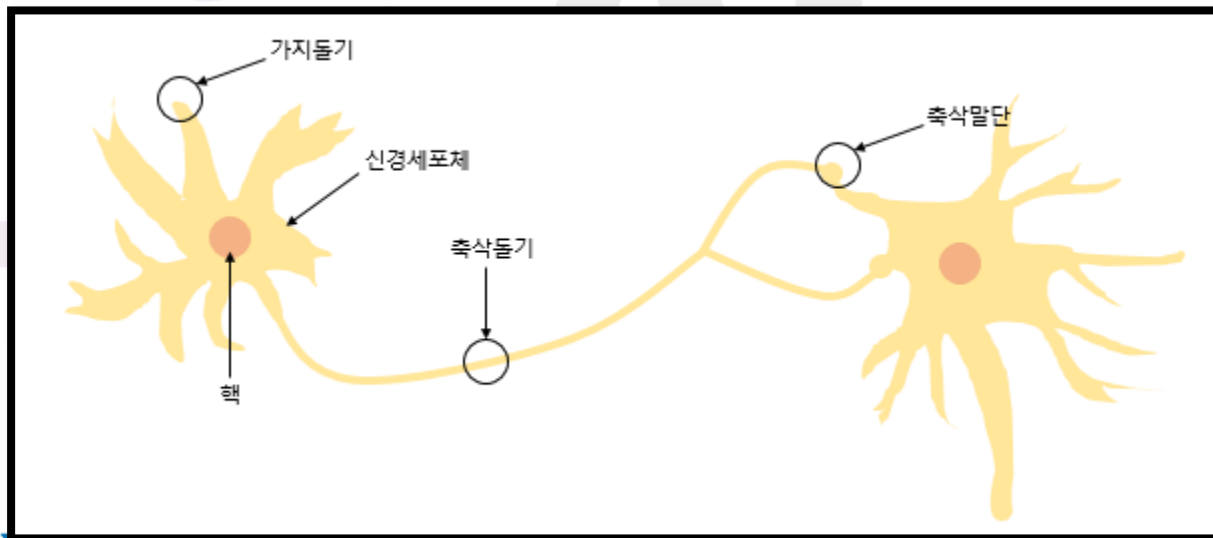
-신경망의 학습 원리를 이해하고,
신경망 모델을 프로그래밍 할 수 있다.

-선수과목: 시기초프로그래밍

Perceptron

◆ 신경망

- 인간이 뇌를 통해 문제를 처리하는 방법과 비슷한 방법으로 컴퓨터에서 문제를 해결하려는 모델
- 뉴런
 - ✓ 가지돌기에서 신호를 받음
 - ✓ 신호가 일정치 이상이면 축삭 돌기로 신호를 전달

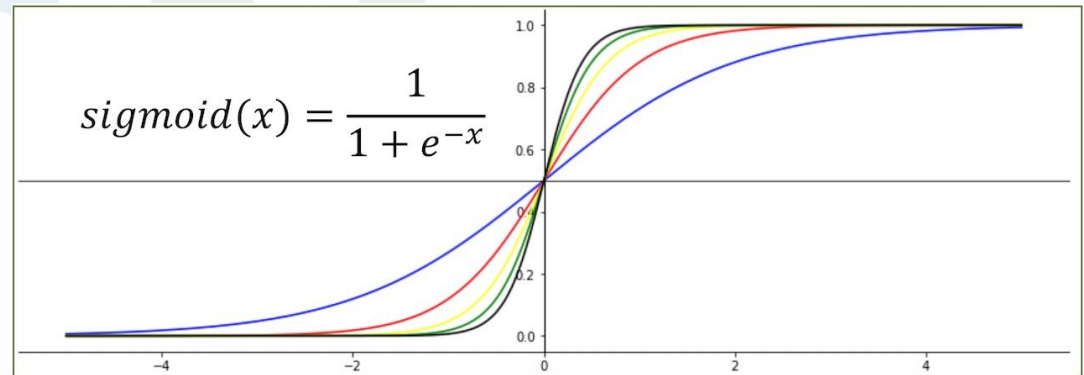
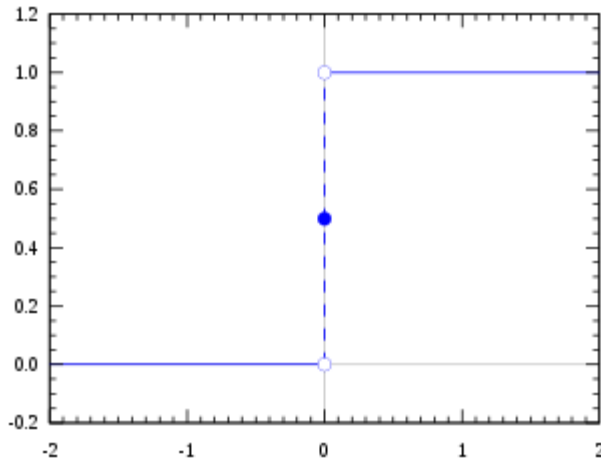


출처: <https://wikidocs.net/24958>

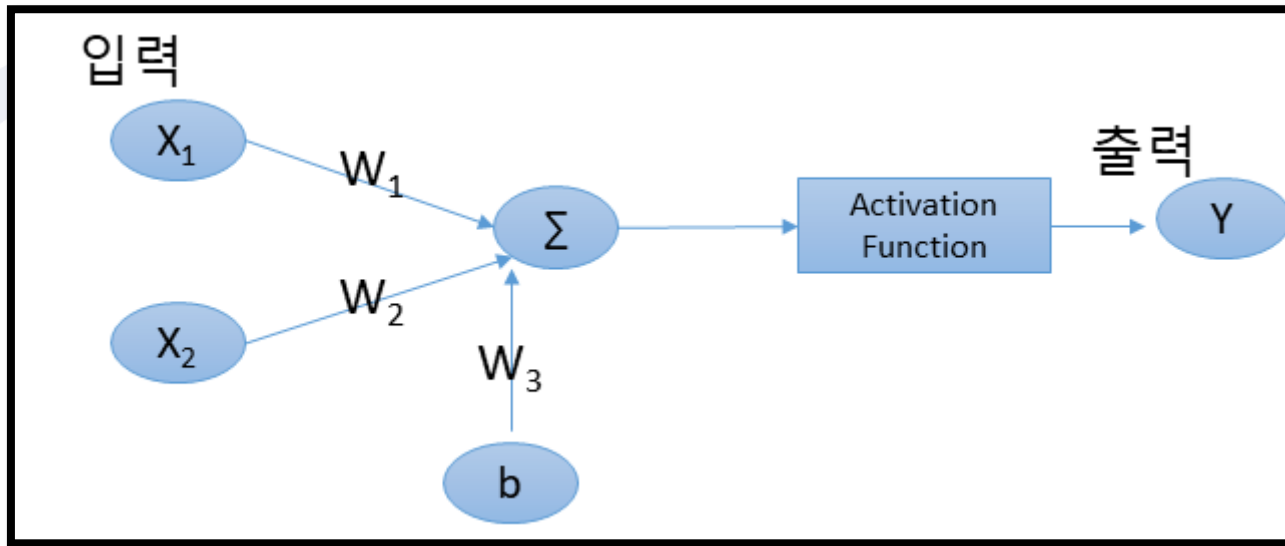
Perceptron

◆ Activation Function (활성화 함수)

- 뉴런에 가해지는 신호가 특정 임계치 까지는
- 아무런 신호를 보내지 않다가 임계치를 초과하면
- 신호를 내보내는 것을 모방



Perceptron Model



활성화 함수 $y = f(\sum W_i X_i)$

$$\text{가중합} = W_0 X_0 + W_1 X_1 + W_3 b$$

Neural Network

◆ 신경망

- 뉴런들을 여러 개 연결한 네트워크

◆ 신경망 학습

- 데이터를 통해 W 의 값을 찾아내겠다

◆ 모델

- W 값들을 다 저장해 둔게 모델이지 뭐.

Weight 학습

◆ 어떤 입력에 대하여

- 원하는 출력값(목표값) d 와
- 모델의 출력 y 사이에 차이가 발생
- 현재의 weight에 차이를 더해서 weight를 갱신

◆ weight를 갱신할 때 차이를 얼마나 줘?

- 그게 바로 **학습률**
- 학습률이 너무 높으면 최적의 결과를 얻지 못함
- 너무 낮으면 학습 시간이 오래 걸림

$$W_{n+1} = W_n + \eta(d - y)$$

And 게이트 학습

학습데이터

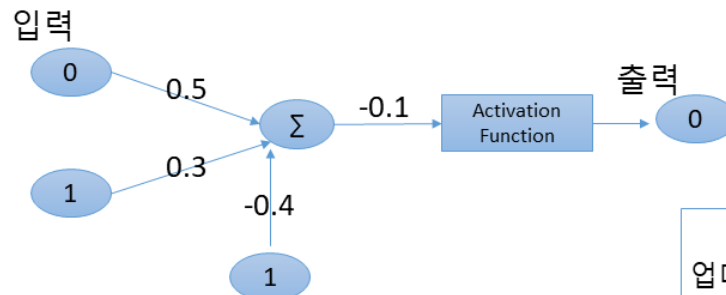
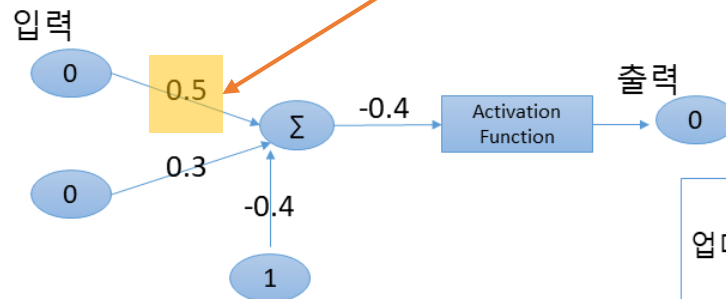
x1	x2	d
0	0	0
0	1	0
1	0	0
1	1	1

Activation Function

$$f(x) = \begin{cases} 1, & x \geq 0 \\ 0, & x < 0 \end{cases}$$

학습율 5% (0.05)

Weight의 초기값?
-1 ~ 1 사이 랜덤

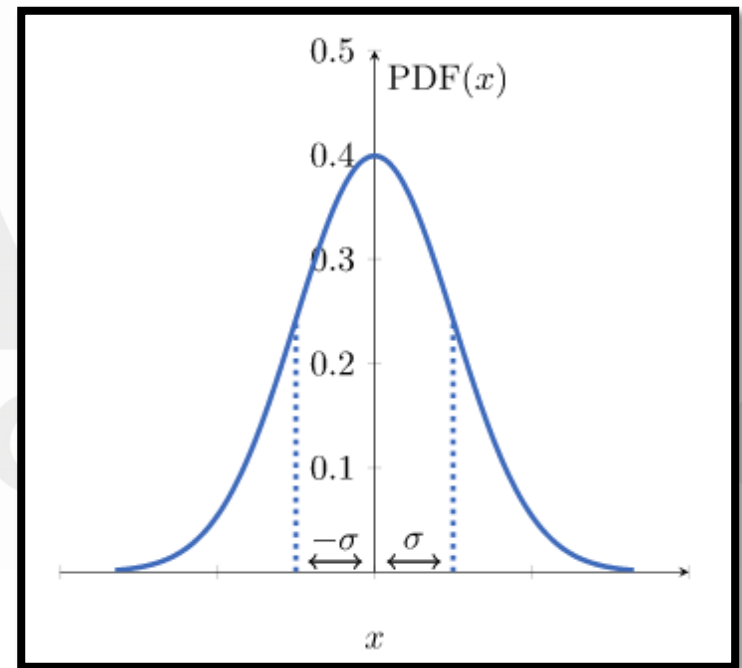
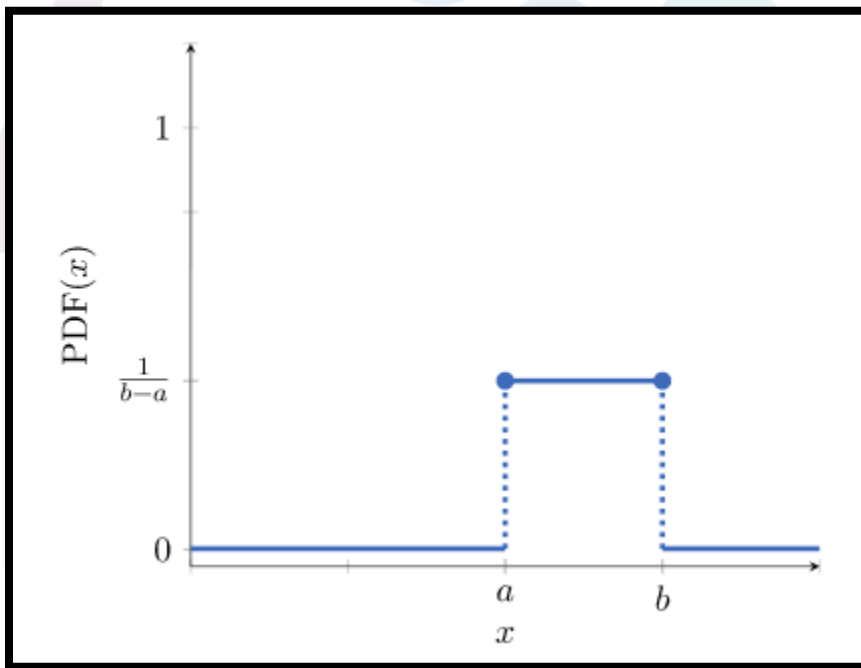


1라운드

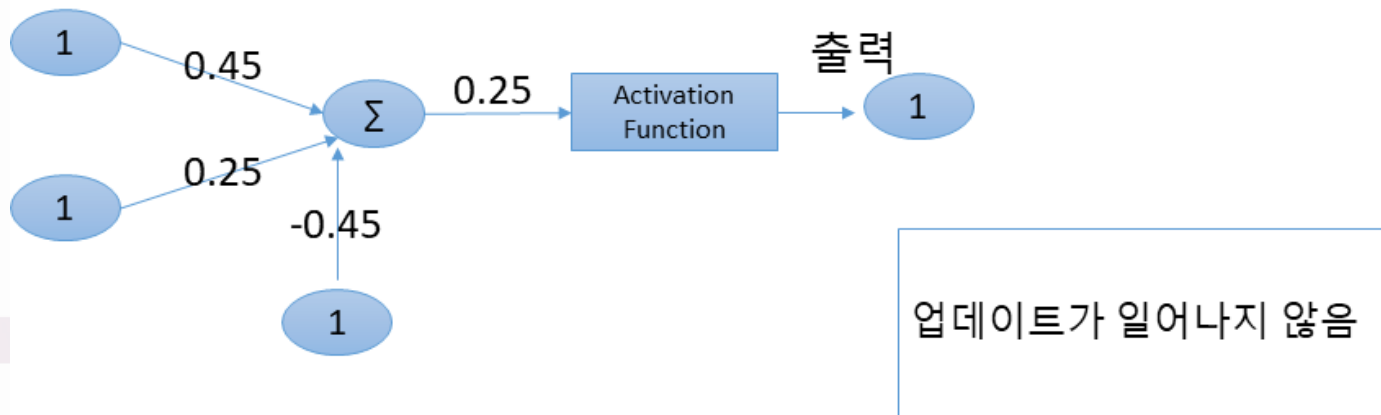
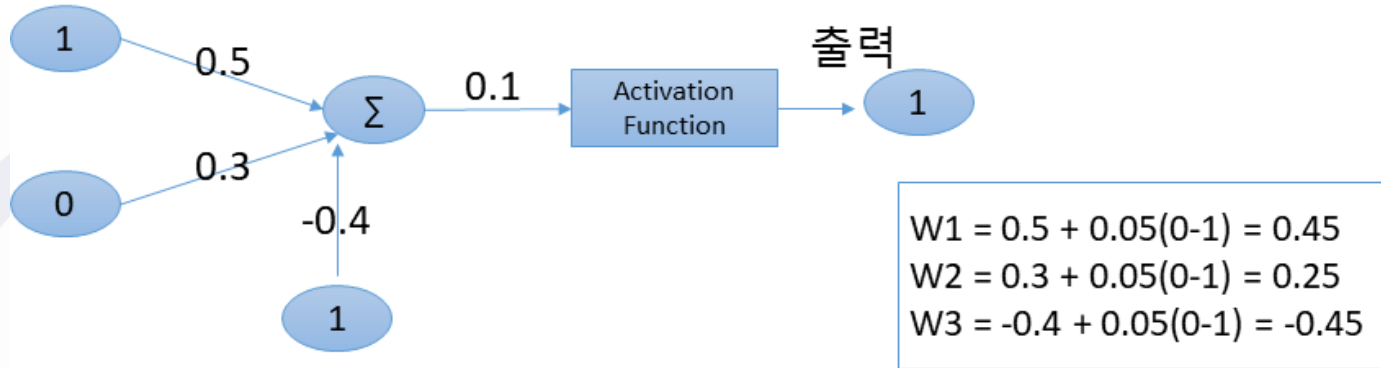
Random

◆ Uniform Distribution / Normal Distribution

➤ 확률 밀도 함수 (Probability Density Function)



And 게이트 학습



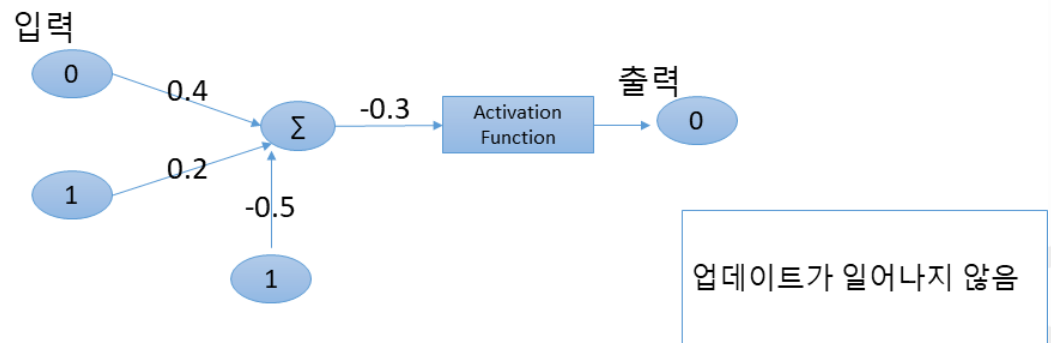
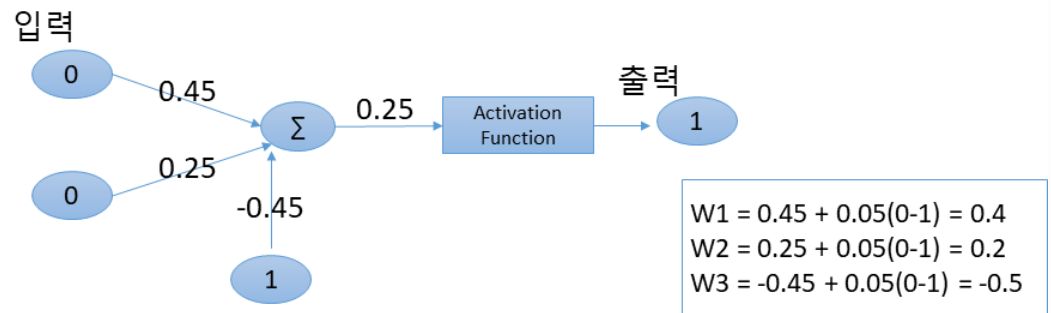
1라운드

라운드? = 모든 학습데이터를 1회 학습하면 1라운드
Epoch 라는 표현으로 많이 사용함

And 게이트 학습

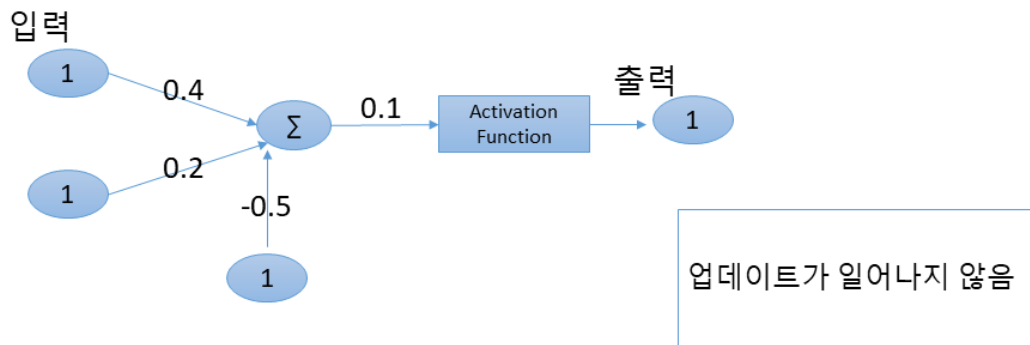
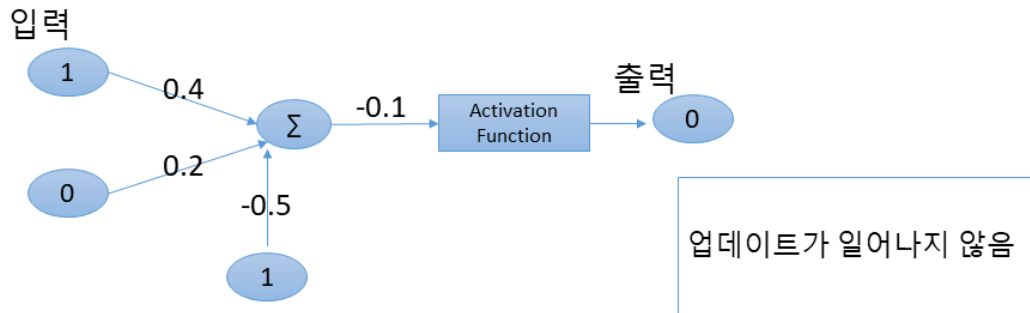
x1	x2	d	y
0	0	0	0
0	1	0	0
1	0	0	1
1	1	1	1

1라운드 종료 후 출력값
정답률 = 75%



2라운드

And 게이트 학습

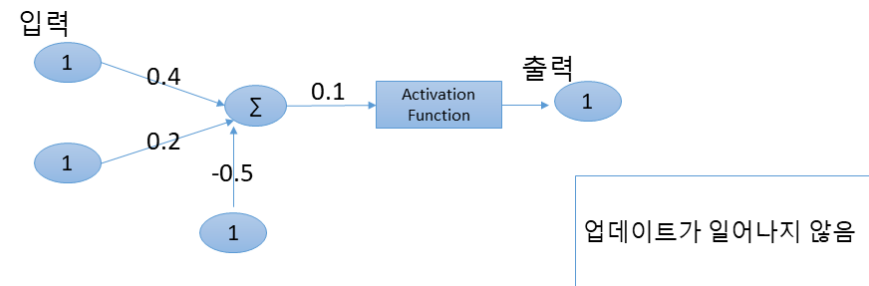
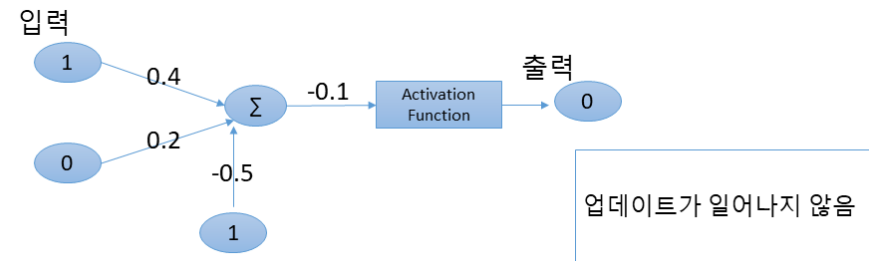
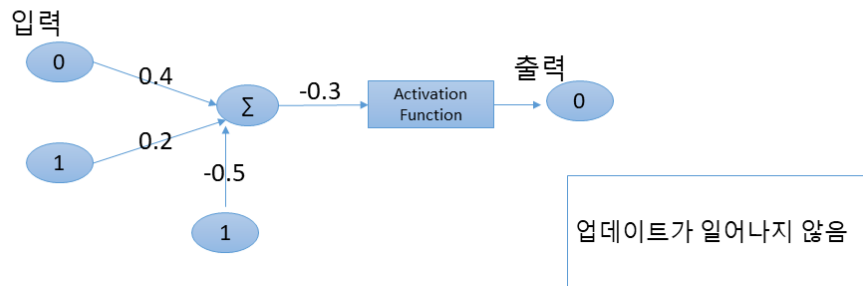
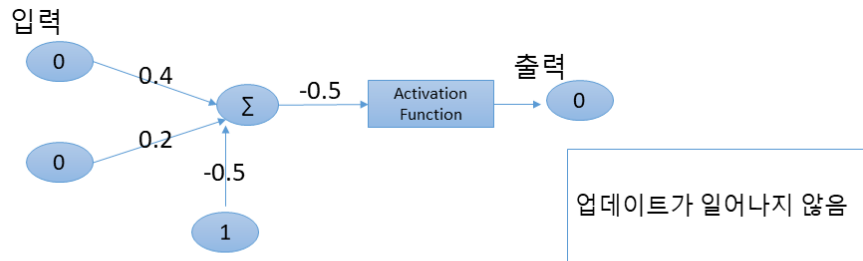


2라운드

x1	x2	d	y
0	0	0	1
0	1	0	0
1	0	0	0
1	1	1	1

2라운드 결과

And 게이트 학습



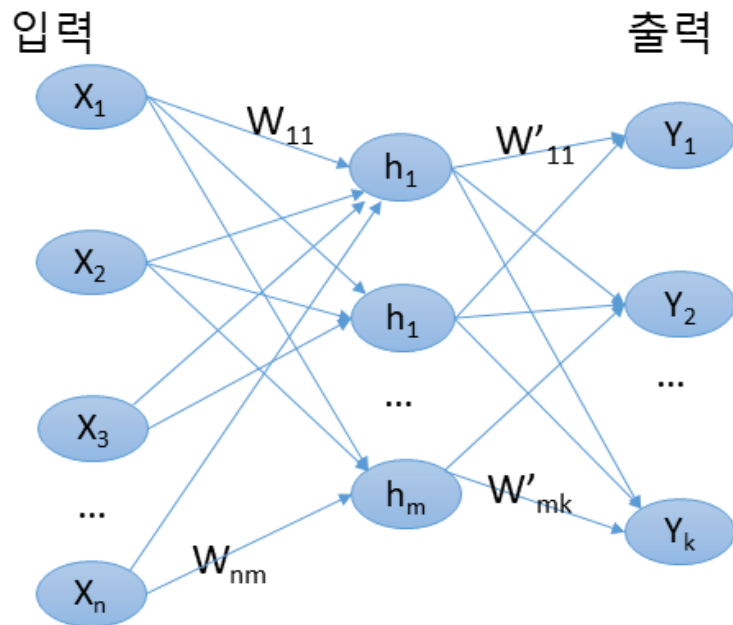
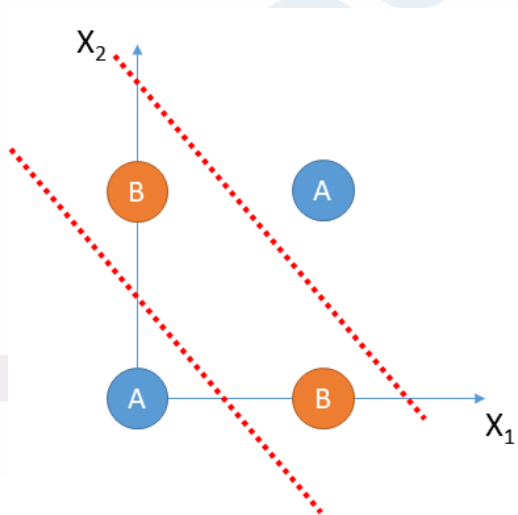
3라운드

학습된 모델 사용해 보기

```
def AND_gate(x1, x2):  
    w1 = 0.4  
    w2 = 0.2  
    # w3은 b로 표시.  
    # b는 1이므로  $1 * w3 = w3$  와 같다. 그러나! 다른 웨이트와 구별하기 위해  
    b = -0.5  
  
    w_sum = w1 * x1 + w2 * x2 + b  
  
    # activation function  
    # 0이하 0, 그 외 1  
    if w_sum <= 0:  
        return 0  
    else:  
        return 1  
  
if __name__ == "__main__":  
    print("(0, 0)", AND_gate(0, 0))  
    print("(0, 1)", AND_gate(0, 1))  
    print("(1, 0)", AND_gate(1, 0))  
    print("(1, 1)", AND_gate(1, 1))
```

XOR 문제

- ◆ 단층 신경망으로는 XOR 문제를 해결 못함
- ◆ 다층 신경망으로 발전 (10년 걸림...)
 - Multi Layer Perceptron; MLP



딥러닝 환경 구축시 고려 사항

◆ IDE

- Jupyter Lab / notebook
- Visual Studio / VS code
- PyCharm
- Eclipse



Visual Studio Code



딥러닝 프레임워크

◆ TensorFlow (텐서 플로우)

- 구글, 2015년, 가장 널리 사용
- 개방성, 다양한 언어 지원, 많은 사용자
- Python, C/C++ 등 지원
- 초기에 리눅스와 맥OS 만 지원, 현재 윈도우도 지원
- <https://www.tensorflow.org/?hl=ko>

◆ Keras (케라스)

keras in TensorFlow

- 구글, 2015년, (2017년부터 TensorFlow에서 지원)
- 래퍼 라이브러리(MXNet, DL4J, TensorFlow, CNTK, Theano), Python 기반
- 사용성 우수(모듈화, 최소주의, 확장성)
- <https://keras.io/>, <https://keras.io/ko/>

딥러닝 프레임워크

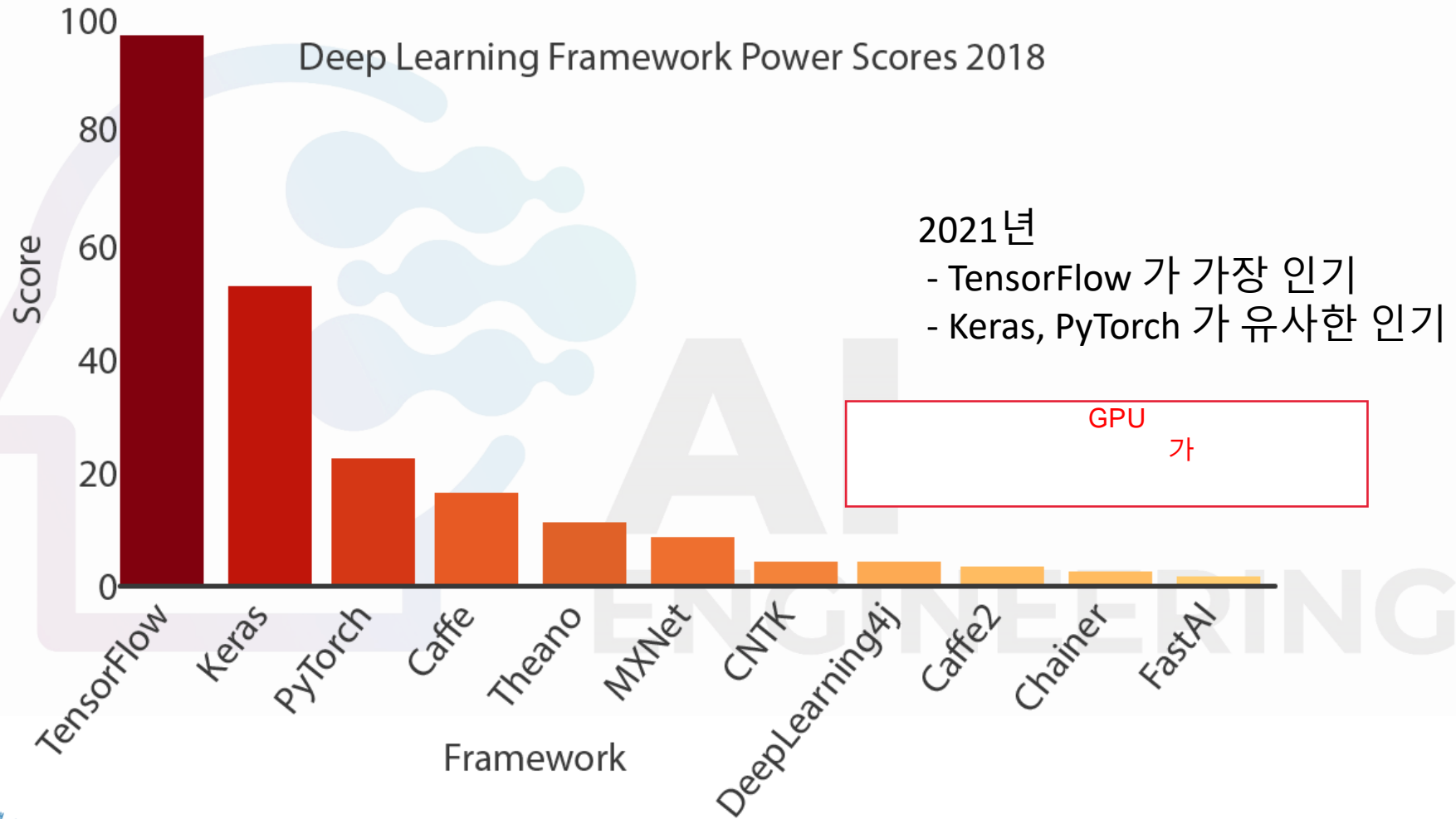
◆ PyTorch (파이토치)

- 페이스북, 2016년
- 사용성 우수(코드 구현 장벽 낮음, 사용법 쉬움)
- 연구 기관 / 대학에서 주로 사용
- <https://pytorch.org/>

◆ Caffe (카페)

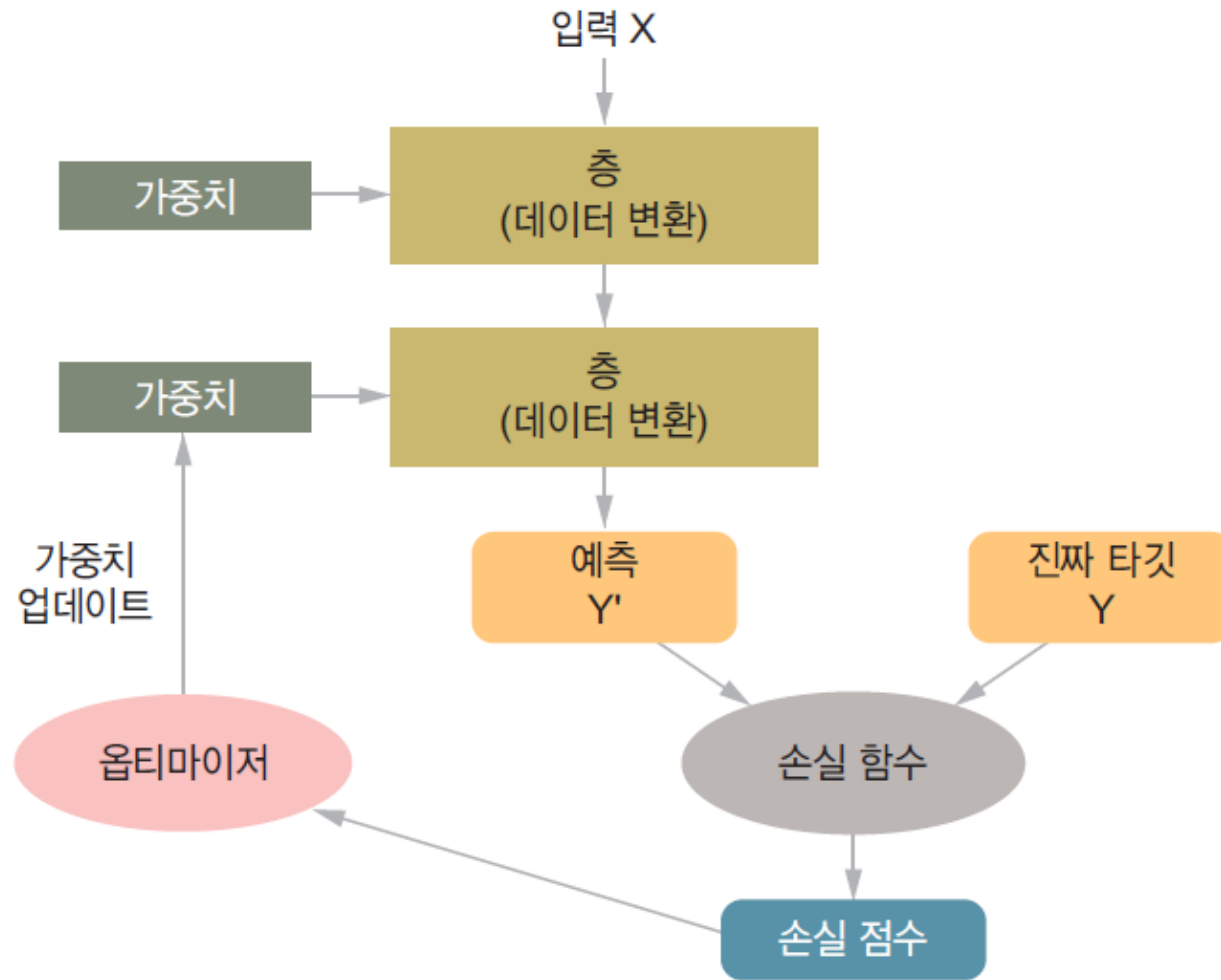
- 버클리 대학, 2013년
- C/C++ 기반, Python 인터페이스
- 컴퓨터비전/CNN/음성 특화, 범용성 낮음
- 모델 공유 네트워크 (커뮤니티, Model Zoo)
- <https://caffe.berkeleyvision.org/>

딥러닝 프레임워크



[출처] <https://towardsdatascience.com/top-5-deep-learning-frameworks-to-watch-in-2021-and-why-tensorflow-98d8d6667351>

신경망 학습 과정



[출처] 모두의 딥러닝

신경망의 학습

◆ Loss Optimization

- 손실을 최소화 하는 가중치를 찾는 것

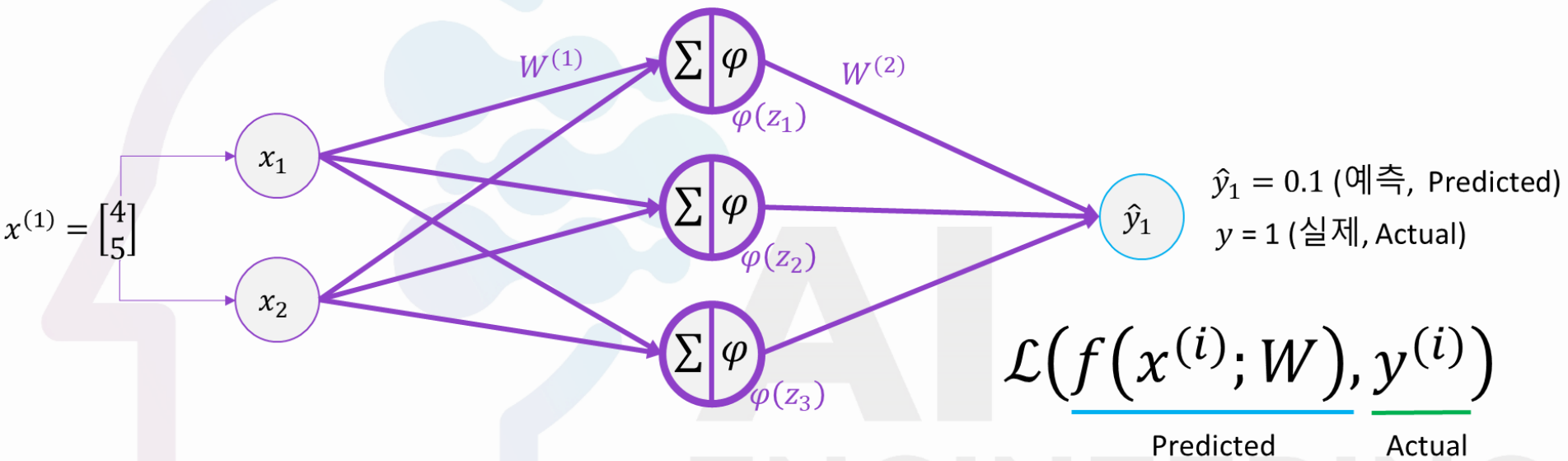
$$W^* = \operatorname{argmin}_W \frac{1}{N} \sum_{i=1}^N \mathcal{L}(f(x^{(i)}; W), y^{(i)})$$

$$W^* = \operatorname{argmin}_W J(W)$$

↑

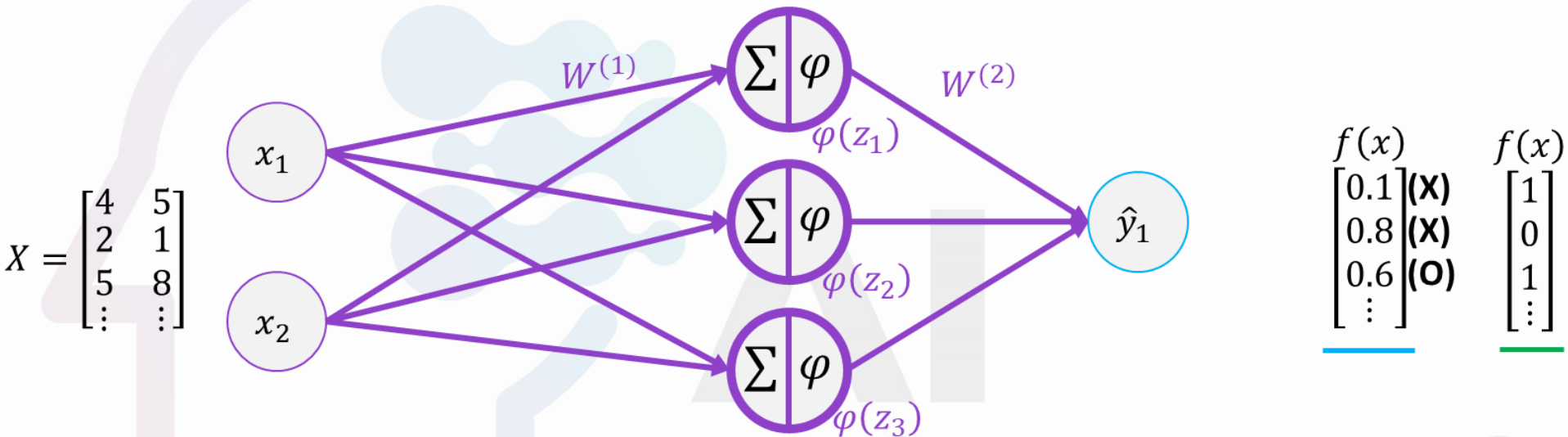
$$W = \{W^{(1)}, W^{(2)}, \dots\}$$

Loss [손실]



Empirical Loss

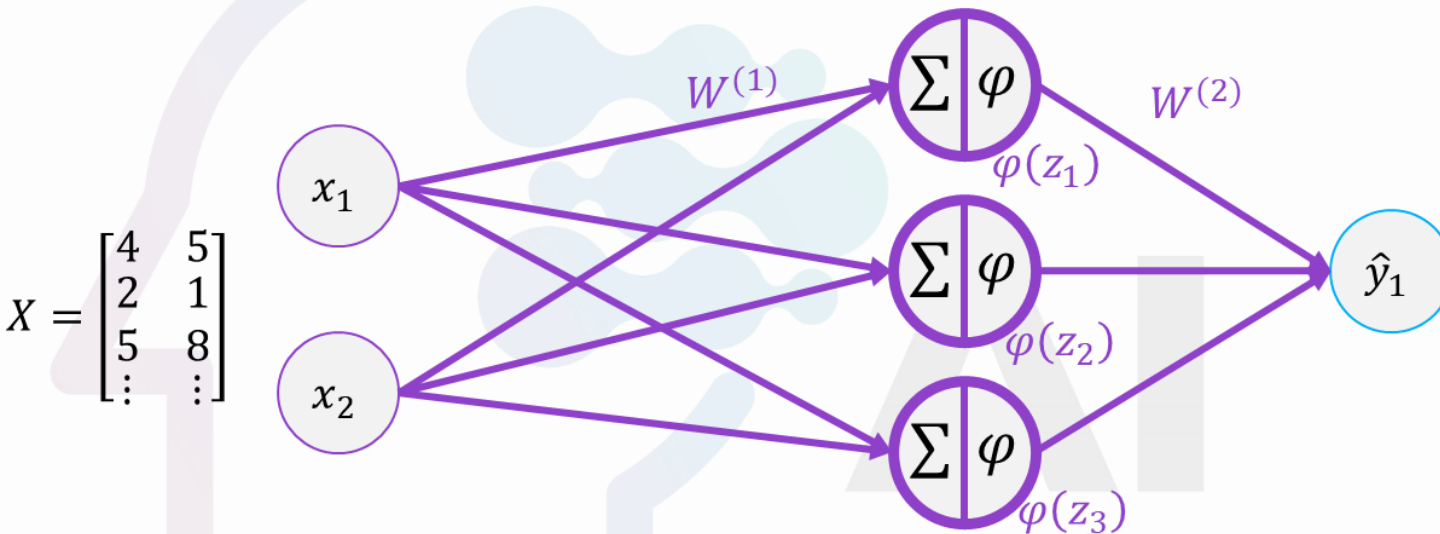
◆ 전체 데이터셋에 대한 총 손실을 측정



$$J(W) = \frac{1}{N} \sum_{i=1}^N \mathcal{L}(f(x^{(i)}; W), y^{(i)})$$

Mean Squared Error Loss

◆ Regression 문제에서 많이 사용

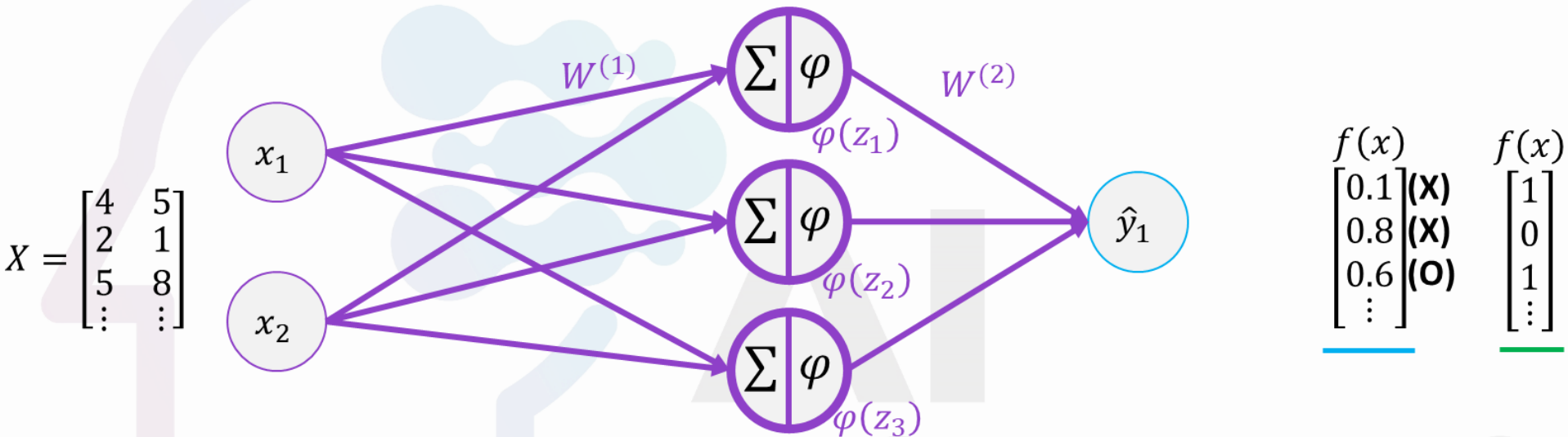


$$\begin{array}{c} f(x) \\ \begin{bmatrix} 30 \\ 80 \\ 85 \\ \vdots \end{bmatrix} \end{array} \begin{array}{c} (\mathbf{x}) \\ (\mathbf{x}) \\ (\mathbf{o}) \\ \end{array} \quad \begin{array}{c} f(x) \\ \begin{bmatrix} 90 \\ 20 \\ 95 \\ \vdots \end{bmatrix} \end{array}$$

$$J(W) = -\frac{1}{N} \sum_{i=1}^N \left(y^{(i)} - f(x^{(i)}; W) \right)^2$$

Binary Cross Entropy Loss

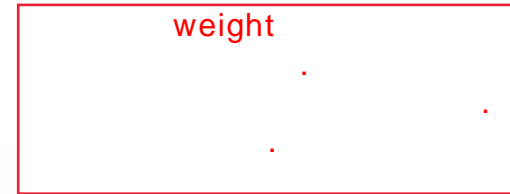
◆ 출력이 확률(0~1) 형태일 때 사용



$$J(W) = -\frac{1}{N} \sum_{i=1}^N y^{(i)} \log(f(x^{(i)}; W)) + (1 - y^{(i)}) \log(1 - f(x^{(i)}; W))$$

Categorical Cross Entropy Loss ()
가 :

Back Propagation



◆ 오차 역전파

- MLP 에서 최적화 과정
- 모델의 예측값 과 원하는 값 사이의 오차를 계산
- 경사 하강법을 이용해 오차가 작아지는 방향으로
웨이트를 업데이트

◆ 오차가 작아지는 것?

- 미분 값 (기울기)가 0이 되는 방향으로 나아간다

새 가중치는 현 가중치에서 '가중치에 대한 기울기'를 뺀 값

$$W(t+1) = W_t - \frac{\partial \text{오차}}{\partial W}$$

Back Propagation

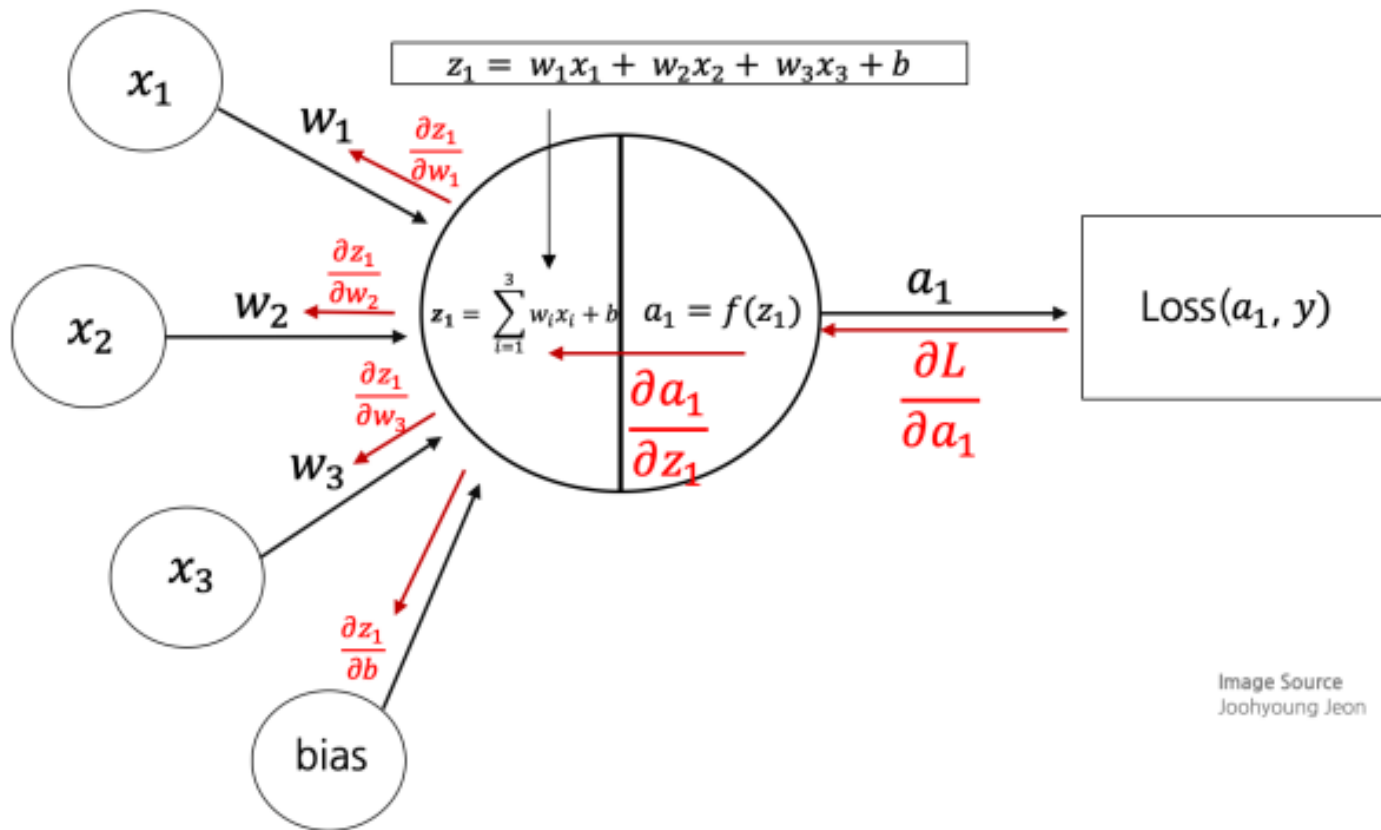


Image Source
Joohyoung Jeon

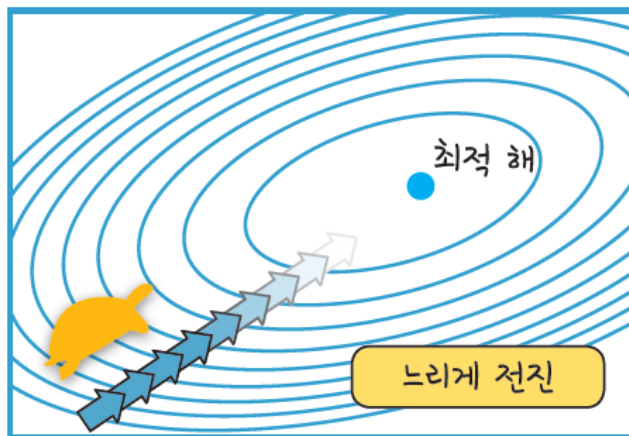
NG

[출처] <https://ds-academy.net/chapter-1-introduction-to-dl/>

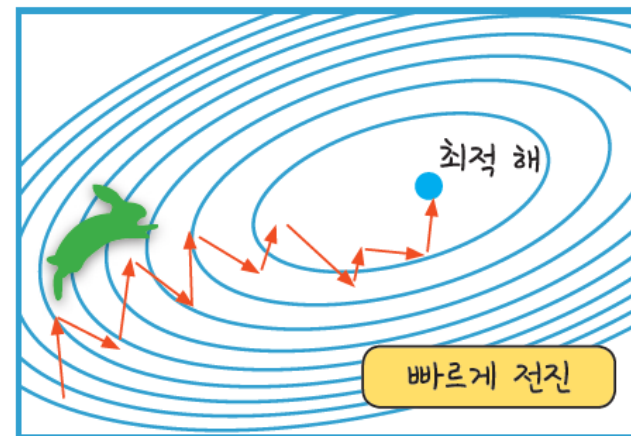
Loss Optimization

◆ Gradient Descent

- 1차 근사값 발견용 최적화 알고리즘
- 함수의 기울기를 구하고 기울기의 절대값이 낮은 쪽으로 계속 이동시켜 극값에 이를때 까지 반복



경사 하강법



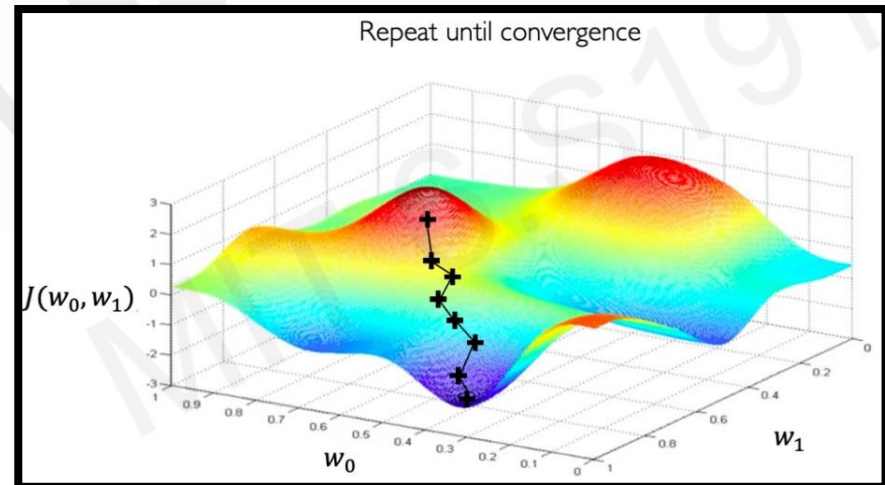
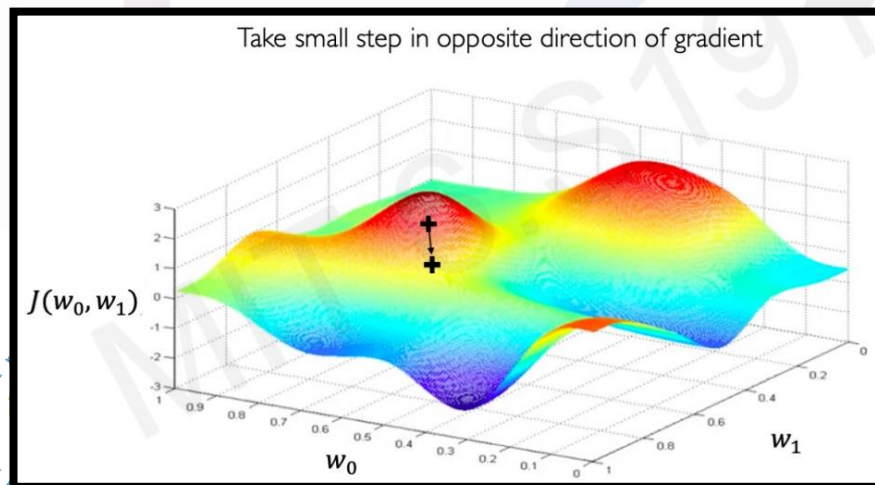
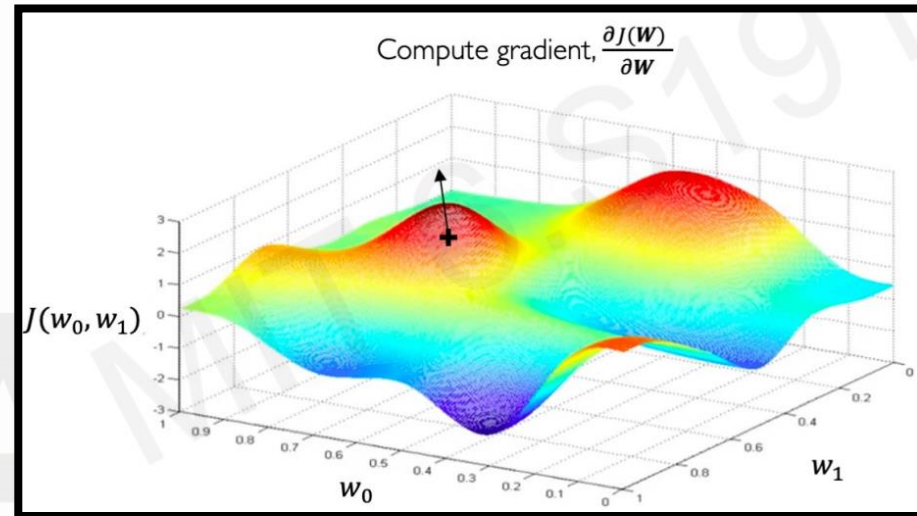
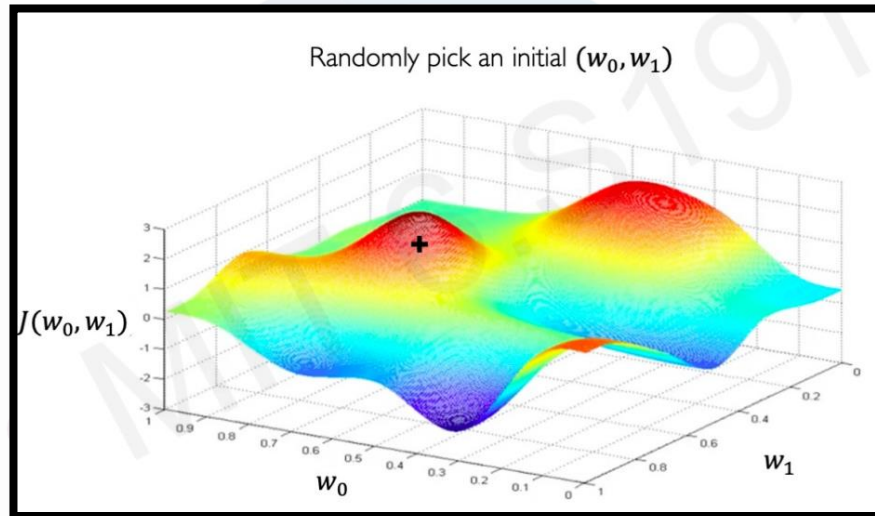
확률적 경사 하강법

$$W \leftarrow W - \eta \frac{\partial J(W)}{\partial W}$$

[출처] 모두의 딥러닝

Loss Optimization

◆ 가중치에 따른 손실 분포



Optimization

고급 경사 하강법	개요	효과	케라스 사용법
확률적 경사 하강법 (SGD)	랜덤하게 추출한 일부 데이터를 사용해 더 빨리, 자주 업데이트를 하게 하는 것	속도 개선	<code>keras.optimizers.SGD(lr = 0.1)</code> 케라스 최적화 함수를 이용합니다.
모멘텀 (Momentum)	관성의 방향을 고려해 진동과 폭을 줄이는 효과	정확도 개선	<code>keras.optimizers.SGD(lr = 0.1, momentum = 0.9)</code> 모멘텀 계수를 추가합니다.
네스테로프 모멘텀 (NAG)	모멘텀이 이동시킬 방향으로 미리 이동해서 그레디언트를 계산. 불필요한 이동을 줄이는 효과	정확도 개선	<code>keras.optimizers.SGD(lr = 0.1, momentum = 0.9, nesterov = True)</code> 네스테로프 옵션을 추가합니다.
아다그라드 (Adagrad)	변수의 업데이트가 잦으면 학습률을 적게 하여 이동 보폭을 조절하는 방법	보폭 크기 개선	<code>keras.optimizers.Adagrad(lr = 0.01, epsilon = 1e - 6)</code> 아다그라드 함수를 사용합니다. ※ 참고: 여기서 epsilon, rho, decay 같은 파라미터는 바꾸지 않고 그대로 사용하기를 권장하고 있습니다. 따라서 lr, 즉 learning rate(학습률) 값만 적절히 조절하면 됩니다.

[출처] 모두의 딥러닝

Optimization

고급 경사 하강법	개요	효과	케라스 사용법
알엠에스프롭 (RMSProp)	아다그라드의 보폭 민감도를 보완한 방법	보폭 크기 개선	<code>keras.optimizers.RMSprop(lr = 0.001, rho = 0.9, epsilon = 1e - 08, decay = 0.0)</code> 알엠에스프롭 함수를 사용합니다.
아담(Adam)	모멘텀과 알엠에스프롭 방법을 합친 방법	정확도와 보폭 크기 개선	<code>keras.optimizers.Adam(lr = 0.001, beta_1 = 0.9, beta_2 = 0.999, epsilon = 1e - 08, decay = 0.0)</code> 아담 함수를 사용합니다.

[출처] 모두의 딥러닝

잘 모르겠으면,
그냥 Adam 을 사용한다!

TensorFlow로 And 게이트 학습해보기

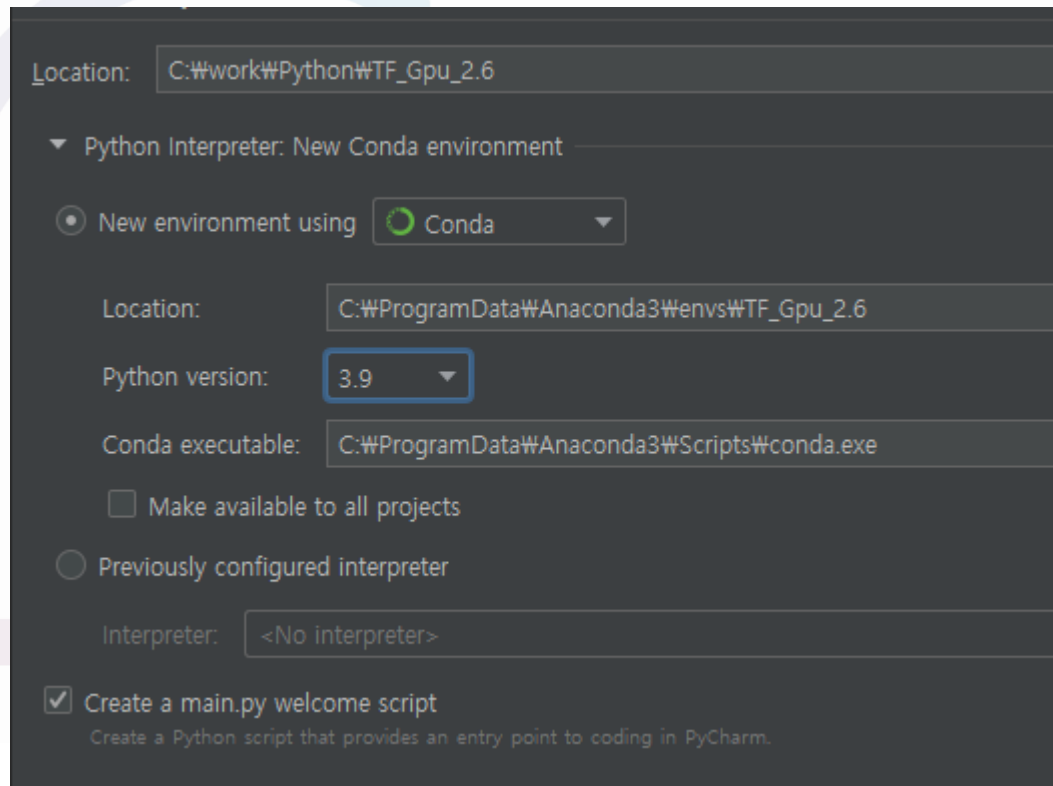
◆ TensorFlow 환경 꾸미기

- 공식 홈페이지를 항상 참고로 한다!
- https://www.tensorflow.org/install/pip.html?hl=ko#windows_1

Windows	
Python 3.6 GPU 지원	https://storage.googleapis.com/tensorflow/windows/gpu/tensorflow_gpu-2.6.0-cp36-cp36m-win_amd64.whl
Python 3.6 CPU만	https://storage.googleapis.com/tensorflow/windows/cpu/tensorflow_cpu-2.6.0-cp36-cp36m-win_amd64.whl
Python 3.7 GPU 지원	https://storage.googleapis.com/tensorflow/windows/gpu/tensorflow_gpu-2.6.0-cp37-cp37m-win_amd64.whl
Python 3.7 CPU만	https://storage.googleapis.com/tensorflow/windows/cpu/tensorflow_cpu-2.6.0-cp37-cp37m-win_amd64.whl
Python 3.8 GPU 지원	https://storage.googleapis.com/tensorflow/windows/gpu/tensorflow_gpu-2.6.0-cp38-cp38-win_amd64.whl
Python 3.8 CPU만	https://storage.googleapis.com/tensorflow/windows/cpu/tensorflow_cpu-2.6.0-cp38-cp38-win_amd64.whl
Python 3.9 GPU 지원	https://storage.googleapis.com/tensorflow/windows/gpu/tensorflow_gpu-2.6.0-cp39-cp39-win_amd64.whl
Python 3.9 CPU만	https://storage.googleapis.com/tensorflow/windows/cpu/tensorflow_cpu-2.6.0-cp39-cp39-win_amd64.whl

새 가상환경 만들기

◆ Python Version을 3.9로 한다.



The image shows the 'New Environment' dialog box in PyCharm. The 'Location' field is set to 'C:\work\Python\TF_Gpu_2.6'. Under the 'Python Interpreter' section, 'New environment using' is selected, and 'Conda' is chosen from the dropdown. The 'Location' field is set to 'C:\ProgramData\Anaconda3\envs\TF_Gpu_2.6'. The 'Python version' dropdown is set to '3.9' and is highlighted with a blue border. The 'Conda executable' field is set to 'C:\ProgramData\Anaconda3\Scripts\conda.exe'. The 'Make available to all projects' checkbox is unchecked. The 'Previously configured interpreter' radio button is selected. The 'Interpreter' field is set to '<No interpreter>'. The 'Create a main.py welcome script' checkbox is checked, with a note below it: 'Create a Python script that provides an entry point to coding in PyCharm.'

Location: C:\work\Python\TF_Gpu_2.6

Python Interpreter: New Conda environment

☒ New environment using Conda

Location: C:\ProgramData\Anaconda3\envs\TF_Gpu_2.6

Python version: 3.9

Conda executable: C:\ProgramData\Anaconda3\Scripts\conda.exe

☐ Make available to all projects

☒ Previously configured interpreter

Interpreter: <No interpreter>

☒ Create a main.py welcome script
Create a Python script that provides an entry point to coding in PyCharm.

EERING

가상환경 활성화

◆ 가상환경 확인

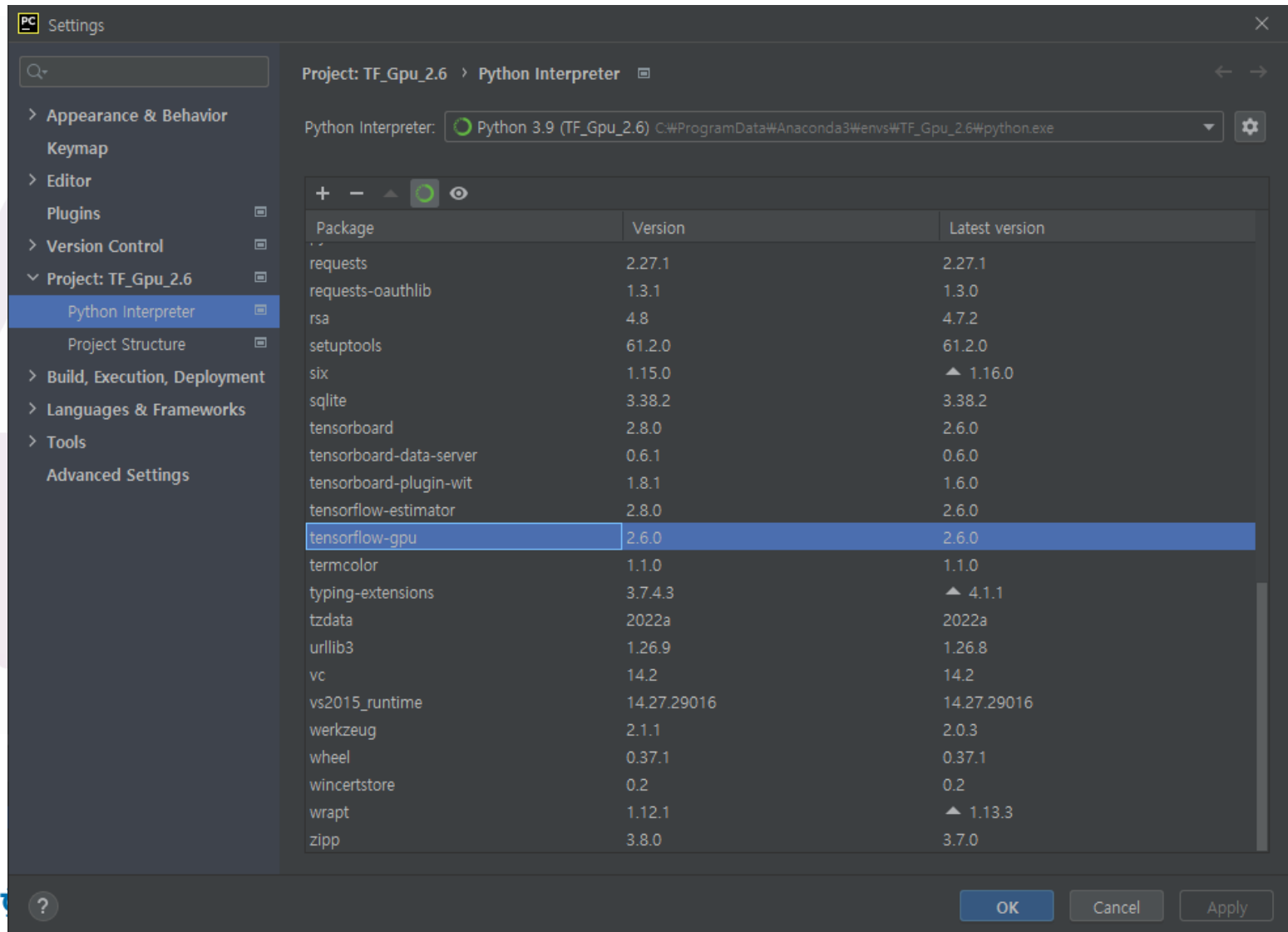
- PyCharm → Terminal
- `conda info -envs`
- `conda activate TF_Gpu_2.6`

◆ Python whl 파일 설치 방법

- 설치하고자 하는 whl 파일을 다운로드 받는다.
- `python -m pip install whl파일명`

ENGINEERING

TensorFlow Gpu 설치 확인



Project: TF_Gpu_2.6 > Python Interpreter

Python Interpreter: Python 3.9 (TF_Gpu_2.6) C:\ProgramData\Anaconda3\envs\TF_Gpu_2.6\python.exe

Package	Version	Latest version
requests	2.27.1	2.27.1
requests-oauthlib	1.3.1	1.3.0
rsa	4.8	4.7.2
setuptools	61.2.0	61.2.0
six	1.15.0	▲ 1.16.0
sqlite	3.38.2	3.38.2
tensorboard	2.8.0	2.6.0
tensorboard-data-server	0.6.1	0.6.0
tensorboard-plugin-wit	1.8.1	1.6.0
tensorflow-estimator	2.8.0	2.6.0
tensorflow-gpu	2.6.0	2.6.0
termcolor	1.1.0	1.1.0
typing-extensions	3.7.4.3	▲ 4.1.1
tzdata	2022a	2022a
urllib3	1.26.9	1.26.8
vc	14.2	14.2
vs2015_runtime	14.27.29016	14.27.29016
werkzeug	2.1.1	2.0.3
wheel	0.37.1	0.37.1
wincertstore	0.2	0.2
wrapt	1.12.1	▲ 1.13.3
zipp	3.8.0	3.7.0

OK Cancel Apply

IG

실행 해보자.

```
from tensorflow.keras.models import Sequential

if __name__ == '__main__':
    model = Sequential()
```

◆ 예러가 날 것이다.

- CUDA 와 CuDnn 이 없기 때문

◆ Nvidia GPU를 사용하는 경우

- 설치된 GPU 카드에 맞고, TF 버전과도 맞는
- CUDA 툴킷을 설치해야 한다.

CUDA 버전 확인

◆ https://www.tensorflow.org/install/source_windows#tested_build_configurations

GPU

버전	Python 버전	컴파일러	빌드 도구	cuDNN	CUDA
tensorflow_gpu-2.7.0	3.7~3.9	MSVC 2019	Bazel 3.7.2	8.1	11.2
tensorflow_gpu-2.6.0	3.6~3.9	MSVC 2019	Bazel 3.7.2	8.1	11.2
tensorflow_gpu-2.5.0	3.6~3.9	MSVC 2019	Bazel 3.7.2	8.1	11.2
tensorflow_gpu-2.4.0	3.6-3.8	MSVC 2019	Bazel 3.1.0	8.0	11.0

➤ 우리는 cuDNN 8.1 과 CUDA toolkit 11.2를 설치

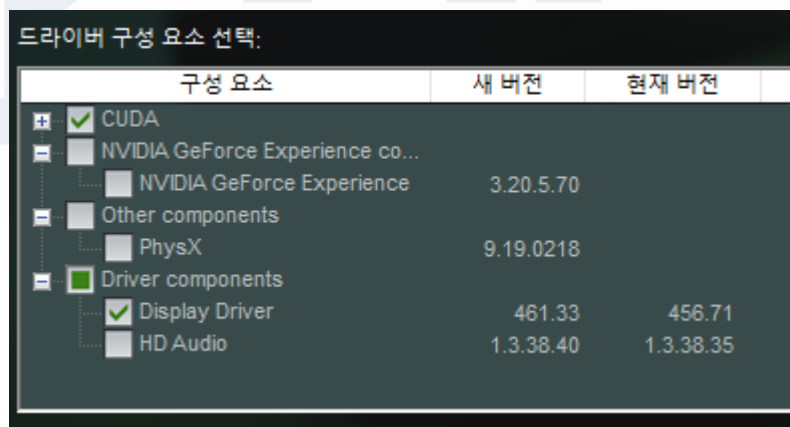
CUDA Toolkit 다운로드

◆ Nvidia 공식 사이트

➤ <https://developer.nvidia.com/cuda-downloads>

◆ CUDA 설치

- 11.2 버전을 다운받아 설치한다.
- 최신 버전이라고 좋은 게 아니다.
- TensorFlow 사이트에서 테스트한 버전을 쓴다.



CUDA 설치 확인

◆ cmd 창에서

➤ **nvcc --version**

```
C:\WINDOWS\system32\cmd.exe
Microsoft Windows [Version 10.0.19044.1645]
(c) Microsoft Corporation. All rights reserved.

C:\Users\Kang>nvcc --version
nvcc: NVIDIA (R) Cuda compiler driver
Copyright (c) 2005-2021 NVIDIA Corporation
Built on Sun_Feb_14_22:08:44_Pacific_Standard_Time_2021
Cuda compilation tools, release 11.2, V11.2.152
Build cuda_11.2.r11.2/compiler.29618528_0

C:\Users\Kang>.
```

cuDNN 다운로드

- ◆ <https://developer.nvidia.com/rdp/cudnn-archive>
- ◆ cuDNN 다운로드를 위해서는 Nvidia 로그인이 필요하다.
- ◆ 압축을 풀고
 - CUDA가 설치된 폴더에 덮어 쓴다.
 - C:\Program Files\NVIDIA GPU Computing Toolkit\CUDA\v11.2

Test

Cannot register 2 metrics with the same name: /tensorflow/api/keras/optimizers

◆ 이런 에러가 뜬다면

➤ TensorFlow 버전을 2.5로 다운그레이드한다.

GPU

버전	Python 버전	컴파일러	빌드 도구	cuDNN	CUDA
tensorflow_gpu-2.7.0	3.7~3.9	MSVC 2019	Bazel 3.7.2	8.1	11.2
tensorflow_gpu-2.6.0	3.6~3.9	MSVC 2019	Bazel 3.7.2	8.1	11.2
tensorflow_gpu-2.5.0	3.6~3.9	MSVC 2019	Bazel 3.7.2	8.1	11.2
tensorflow_gpu-2.4.0	3.6~3.8	MSVC 2019	Bazel 3.1.0	8.0	11.0

➤ `pip install tensorflow-gpu==2.5.0`

TensorFlow로 And 게이트 학습해보기

◆ 모델의 입력 (X)

- 2개의 정수 값
- [[0 0], [0 1], [1, 0] [1, 1]]

◆ 모델의 출력 (Y)

- 0 or 1
- 0과 1을 각각 다른 클래스로 간주할 것이다.
- $0 \rightarrow [1, 0]$
- $1 \rightarrow [0, 1]$



```
0      1,0
1      0,1      .      .
      ( ) a,b,c 3
      . 0.5, 0.3, 0.2 ( 1.0 . )
- > 0.5가 가      0.5 a
      가      . a      .
```

One Hot Encoding

◆ Target 은 1로 나머지는 0으로

◆ MNIST 예

➤ 1 → [0, 1, 0, 0, 0, 0, 0, 0, 0, 0]

➤ 5 → [0, 0, 0, 0, 0, 1, 0, 0, 0, 0]

➤ 9 → [0, 0, 0, 0, 0, 0, 0, 0, 0, 1]

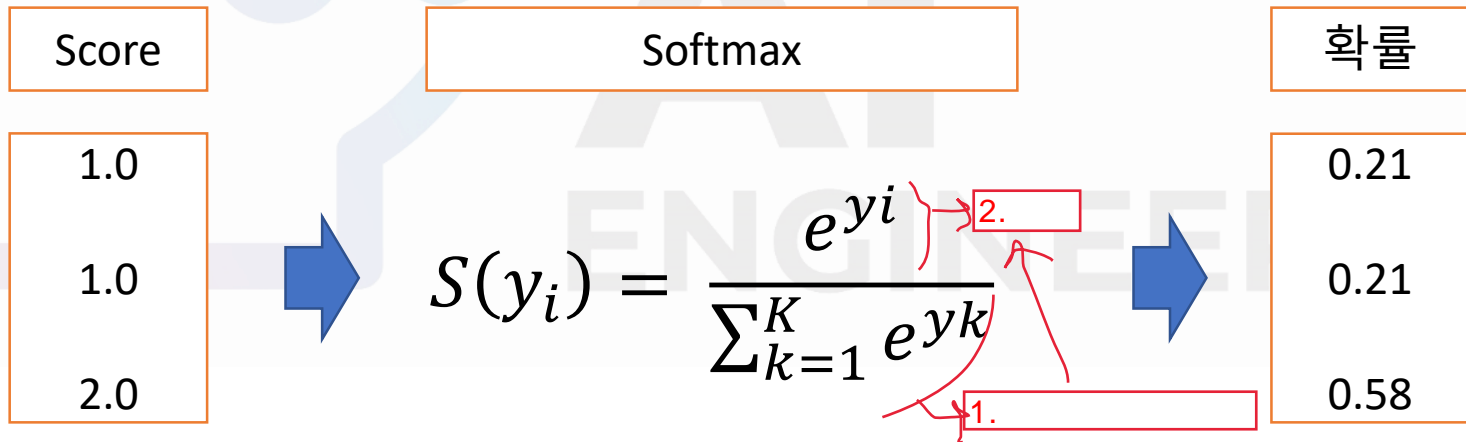
◆ 우리 문제

➤ 0 → [1, 0]

➤ 1 → [0, 1]

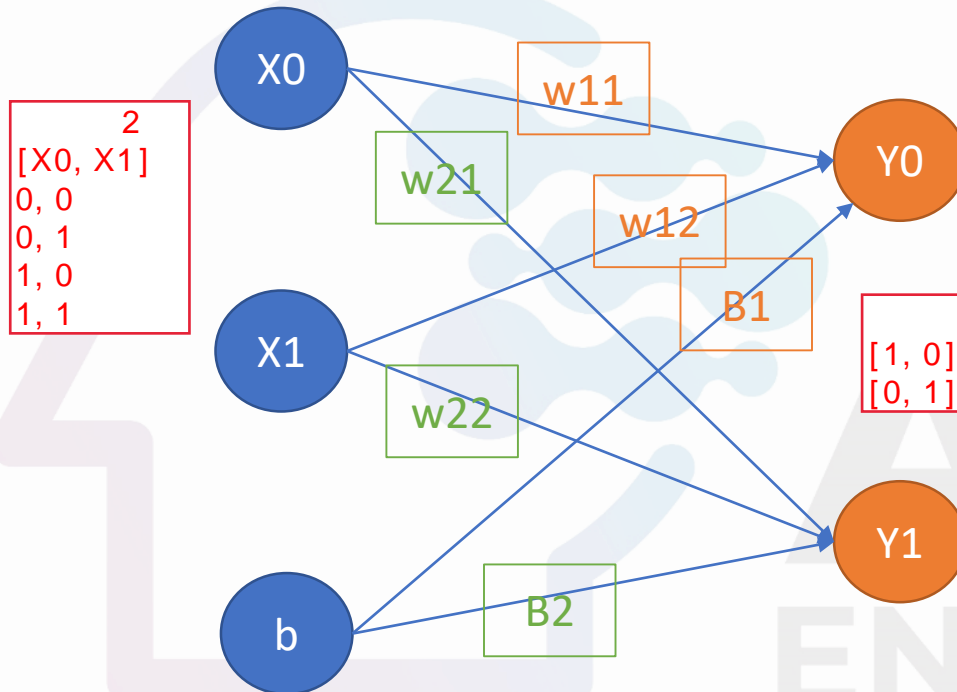
출력 정규화 - Softmax

- ◆ 출력값들의 합이 1이 되도록 정규화
- ◆ 확률과 동일한 개념
- ◆ multi class 분류 문제에서 많이 사용



모델 설계

$$\begin{matrix} 2 & + & b \\ \text{가 } 3 & . & \\ & 2 & . \\ & & \text{가} \\ & & 6 & . \\ 3 & * & 2 & = & 6 \end{matrix}$$



$$\begin{matrix} 2 \\ [X_0, X_1] \\ 0, 0 \\ 0, 1 \\ 1, 0 \\ 1, 1 \end{matrix}$$

$$\begin{bmatrix} W_{11} & W_{12} \\ W_{21} & W_{22} \end{bmatrix} \begin{bmatrix} x_0 \\ x_1 \end{bmatrix} + \begin{bmatrix} B_1 \\ B_2 \end{bmatrix} = \begin{bmatrix} Y_0 \\ Y_1 \end{bmatrix}$$

$$\begin{matrix} 2 \\ [1, 0] \\ [0, 1] \end{matrix}$$

$$\begin{bmatrix} W_{11} & W_{12} & B_1 \\ W_{21} & W_{22} & B_2 \end{bmatrix} \begin{bmatrix} x_0 \\ x_1 \\ 1 \end{bmatrix} = \begin{bmatrix} Y_0 \\ Y_1 \end{bmatrix}$$

이렇게 모든 노드가 연결된 Layer를 Dense Layer 혹은 Fully Connected Layer라고 한다.

TensorFlow에서 구현

```
from tensorflow.keras.models import Sequential
from tensorflow.keras.layers import Dense

if __name__ == '__main__':
    model = Sequential()
    model.add(Dense(2, input_dim=2,
                    kernel_initializer="normal", activation="softmax"))
    model.summary()
```

Layer (type)	Output Shape	Param #
=====	=====	=====
dense (Dense)	(None, 2)	6
=====	=====	=====

Total params: 6

Trainable params: 6

Non-trainable params: 0

Model compile

```
model.compile(optimizer="adam",  
loss="categorical_crossentropy",  
metrics=['accuracy'])
```

◆ Optimizer

- loss를 최적화 하기 위한 알고리즘
- 잘 모르면 뭐라고??

◆ loss

- 손실 계산 함수
- categorical_crossentropy 로 한다.

◆ metric

- 성능 측정 방법
- 정확도를 보면 된다. Accuracy

모델 학습 및 테스트

```
x = [[0, 0], [0, 1], [1, 0], [1, 1]]  
y = [[1, 0], [1, 0], [1, 0], [0, 1]]  
  
model.fit(x, y, epochs=1000)  
  
result = model.predict(x)  
print(result)
```

Epoch 1000/1000

1/1 [=====] - 0s 0s/step - loss: 0.3597 - accuracy: 1.0000

```
[[0.6848706 0.3151294 ]  
 [0.2230476 0.77695245]  
 [0.380831  0.619169  ]  
 [0.7218615 0.27813852]]
```

출력의 형태가 각 class의 확률 값이다.
0일 확률 1일 확률

Summary

◆ 신경망에 대하여 알아보았다.

➤ 신경망?

- ✓ 인간의 신경을 모방
- ✓ 입력에 대한 가중치 합과 활성화 함수로 뉴런을 표현
- ✓ 뉴런들을 여러 개 연결하여 신경망을 구성

➤ 활성화 함수

- ✓ Step 함수 → 미분 가능하지 않다.
- ✓ Sigmoid 함수 → 고전적으로 많이 사용
- ✓ ReLu, Leaky ReLu 등은 나중에...

◆ Keras

- 신경망, 딥러닝 구현을 위한 라이브러리
- 잘 쓰면 된다!