

## 1-2. 데이터를 준비하자!

### MNIST 숫자 손글씨 Dataset 불러들이기

오늘은 텐서플로우(TensorFlow)의 표준 API인 `tf.keras`의 Sequential API를 이용하여 숫자 손글씨 인식기를 만들 거예요. 구글(Google)에서 오픈소스로 제공하는 텐서플로우는 가장 널리 사용되고 있는 머신러닝 라이브러리 중 하나입니다. 앞으로 대부분의 딥러닝 구현실습은 Tensorflow 버전 2.0(혹은 그 이상)에서 진행될 예정입니다.

자, 그럼 TF 2.0이 설치된 환경에서 먼저 다음의 코드를 실행해 봅시다. 앞으로 보게 될 코드의 구체적인 의미와 메커니즘은 이후에 더 자세하게 배우게 될 테니, 지금은 완벽하게 이해하지 못하더라도 마음 편하게 실행해 보세요.

#### [Input]

```
import tensorflow as tf
from tensorflow import keras

import numpy as np
import matplotlib.pyplot as plt

print(tf.__version__) # Tensorflow의 버전을 출력

mnist = keras.datasets.mnist

# MNIST 데이터를 로드. 다운로드하지 않았다면 다운로드까지 자동으로 진행됩니다.
(x_train, y_train), (x_test, y_test) = mnist.load_data()

print(len(x_train)) # x_train 배열의 크기를 출력
```



실행 ▶

#### [Output]

위 코드를 실행하면 숫자 손글씨 데이터베이스인 MNIST 데이터셋을 읽을 수 있습니다. MNIST 데이터셋은 Yann Lecun 교수님이 공개한 데이터로써, 아래 페이지에 방문하면 자세한 내용을 확인할 수 있습니다.

**참고문헌 : MNIST Dataset** <http://yann.lecun.com/exdb/mnist/>

**Q2. 숫자 손글씨 이미지의 크기는 무엇일까요? "nxn"(n은 정수)의 형태로 나타내 보세요.**

28x28

제출

**예시답안**

28x28

**Q3. MNIST dataset에는 총 몇 장의 손글씨 이미지가 있을까요?**

60000

제출

#### 예시답안

70000, 60000장의 training set과 10000장의 test set으로 구성

Q4. 학습용 데이터(training set)은 대략 몇명의 손글씨가 들어있을까요?

250

제출

#### 예시답안

250

자, 한번 불러들인 숫자 손글씨 이미지 하나를 출력해볼까요?

MNIST 데이터셋의 X항목(위 코드에서는 x\_train, x\_test)은 이미지 데이터를 담은 행렬(matrix)입니다.

[Input]

```
plt.imshow(x_train[0], cmap=plt.cm.binary)
plt.show()
```



실행 ▶

[Output]

숫자 0 이미지가 나왔나요? 주의할 것은, x\_train[1]에 담긴 이미지는 x\_train 행렬의 1번째가 아니라 2번째 이미지라는 점입니다. 1번째 이미지는 x\_train[0]에 담겨 있습니다.

그렇다면 Y항목에는 어떤 값이 들어 있을까요? y\_train 행렬의 2번째 값을 확인해 봅시다.

[Input]

```
print(y_train[0])
```

📄 ↺

실행 ▶

[Output]

네, Y항목(위 코드의 y\_train, y\_test)에는 X항목에 들어있는 이미지에 대응하는 실제 숫자 값이 담겨 있는 것을 확인하실 수 있습니다.

그럼 이번에는 또 다른 이미지를 출력해볼까요?

[Input]

```
# index에 0에서 59999 사이 숫자를 지정해 보세요.  
index=1  
plt.imshow(x_train[index],cmap=plt.cm.binary)  
plt.imshow(x_train[index], 'jet')  
plt.show()  
print( (index+1), '번째 이미지의 숫자는 바로 ', y_train[index], '입니다.')
```

📄 ↺

[Output]

## 참고: Matplotlib 이란?

파이썬에서 제공하는 시각화(Visualization) 패키지인 Matplotlib은 차트(chart), 플롯(plot) 등 다양한 형태로 데이터를 시각화할 수 있는 강력한 기능을 제공합니다.

어떤 유용한 기능이 제공되는지 Matplotlib 공식홈페이지에서 제공하는 다양한 활용 예제들을 통해 직접 확인해 보세요.

[Matplotlib 활용 사례 보기](#) 

## 학습용 데이터와 시험용 데이터

---

위 코드를 다시 살펴봅시다.

```
(x_train, y_train), (x_test, y_test) = mnist.load_data()
```


`mnist.load( )` 함수를 통해 학습용 데이터 (`x_train, y_train`) 와 시험용 데이터 (`x_test, y_test`) 를 나누어서 받아들이는 것을 볼 수 있는데요.

우리가 만든 숫자 손글씨 분류기는 학습용 데이터 (`x_train, y_train`) 만을 가지고 학습시킵니다. 학습이 끝난 후에는 이 손글씨 분류기가 얼마나 좋은 성능을 보이는지 확인해보고 싶을 텐데요, 이 때 시험용 데이터 (`x_test, y_test`) 로 테스트를 할 수 있습니다.

MNIST 데이터셋은 약 500명 사용자가 작성한 숫자 이미지를 가지고 있습니다. 그 중 250여명의 데이터가 학습용 데이터로, 다른 250여명의 데이터가 시험용 데이터로 이용됩니다.

## 학습용 데이터

입력(x\_train) , 정답(y\_train)

(  , 0 )  
(  , 5 )  
(  , 7 )  
(  , 5 )

## 시험용 데이터

입력(x\_test) , 정답(y\_test)

(  , 7 )  
(  , 3 )  
...

[학습용 데이터(training set)와 시험용 데이터(test set)의 예]

자 그러면 우리가 불러들인 학습용 데이터는 과연 몇 장일까요? 아래 코드를 실행시켜 봅시다.

[Input]

```
print(x_train.shape)
```



실행 ▶

## [Output]

아마도 (60000,28,28) 이라는 값을 보실 수 있을 겁니다. 이것은 28x28 크기의 숫자 이미지가 60,000장이 있다는 뜻인데요. 마찬가지로 시험용 데이터의 개수를 확인하고 싶다면 아래 코드를 실행하면 됩니다.

## [Input]

```
print(x_test.shape)
```



실행 ▶

## [Output]

10,000장의 시험용 데이터가 저장되어 있음을 알 수 있습니다. 아래 참고문헌을 읽어보시면 학습용 데이터, 검증용 데이터, 그리고 시험용 데이터의 의미와 그 차이점을 보다 자세히 파악할 수 있습니다.

참고문헌 : 데이터셋 이야기 [https://tykimos.github.io/2017/03/25/Dataset\\_and\\_Fit\\_Talk/](https://tykimos.github.io/2017/03/25/Dataset_and_Fit_Talk/)

## Q5. 언제 검증용 데이터(validation set)를 사용하나요?

반복학습을 어느정도로 해야 좋을지를 위해,

제출

## 예시답안

머신러닝 학습 과정이 정상적으로 진행되고 있는지, 오버피팅이 발생하고 있지 않은지, 학습을 중단해도 되는지 등을 확인하고 싶을 때

### Q6. 교차 검증(cross validation) 기법을 옆 친구와 서로 토론하며 이해해봅시다.

사진에서 특정한 영역을 정하고 거기에 내가 찾는 물체를 찾는방법인데 그 특정한영역을 넓게 잡으면 안되고 물체의 크기와 비슷하게 영역을 잡아줘야 하는데 그렇게 이미지 전체에다가 특정한 영역을 여러개 설정하고 찾으려하면 성능을 너무 소비하기때문

제출

## 예시답안

예시답안이 없는 퀴즈입니다. 동료들과 함께 토의해보세요 :)

## 데이터 전처리 하기

숫자 손글씨 이미지의 실제 픽셀 값은 0~255 사이의 값을 가집니다. 한번 확인해 볼까요?

[Input]

```
print('최소값:',np.min(x_train), ' 최대값:',np.max(x_train))
```





실행 ▶

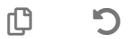
[Output]

인공지능 모델을 훈련시키고 사용할 때, 일반적으로 입력은 0~1 사이의 값으로 정규화 시켜주는 것이 좋습니다. MNIST 데이터는 각 픽셀의 값이 0~255 사이 범위에 있으므로 데이터들을 255.0 으로 나누어주면 됩니다.

최소값이 0, 최대값이 1에 근접하도록 나오는지 확인해 봅시다.

[Input]

```
x_train_norm, x_test_norm = x_train / 255.0, x_test / 255.0  
print('최소값:', np.min(x_train_norm), ' 최대값:', np.max(x_train_norm))  
  
print(x_train.shape)
```



실행 ▶

[Output]