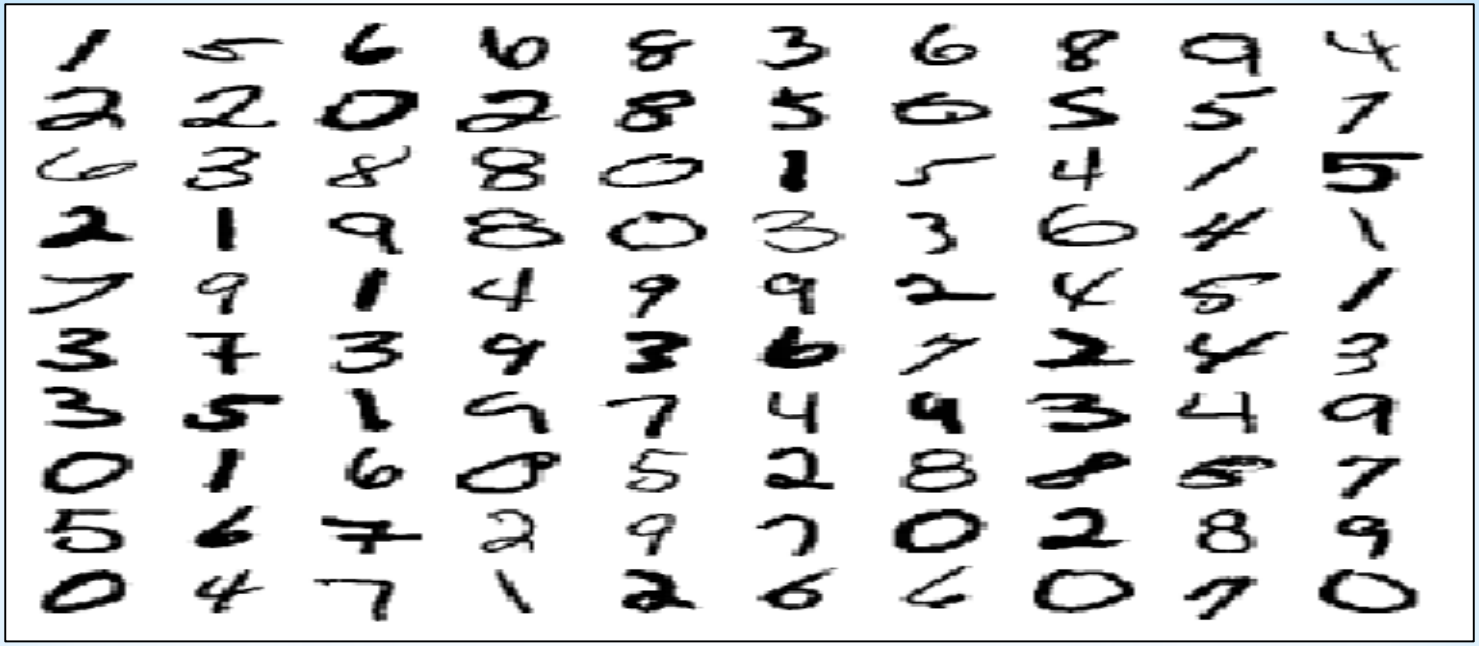
파이썬(Python)으로 구현하는

MNIST
(필기체숫자)

MNIST – 개요



➢ MNIST

- MNIST (Modified National Institute of Standards and Technology)는 손으로 직접 쓴 숫자(필기체 숫자)들로 이루어진 데이터 셋 (Data Set) 이며,
- 우리가 새로운 프로그래밍 언어를 배울 때 'Hello, World' 를 출력하는 것처럼, MNIST는 딥러닝을 배울 때 반드시 거쳐야 하는 'Hello, World' 같은 존재임.
- MNIST는 0부터 9까지의 숫자 이미지로 구성되며, 60,000개의 트레이닝 데이터와 10,000개의 테스트 데이터로 이루어져 있음

MNIST - 다운로드

- from internet
  - Training Data (csv format) - http://www.pjreddie.com/media/files/mnist_train.csv
  - Test Data (csv format) - http://www.pjreddie.com/media/files/mnist_test.csv

입력, 정답
동시 존재!!

- from deep learning framework (Keras, TensorFlow,…)
  - MNIST는 Keras, TensorFlow 등의 딥러닝 프레임웍을 통해서도 가져올 수 있음.

```
(C:₩Program Files₩Anaconda3) C:₩Users₩SungHoPark>pip  install  keras
Requirement already satisfied: keras in c:₩program files₩anaconda3₩lib₩site-packages (2.2.4)
Requirement already satisfied: scipy>=0.14 in c:₩program files₩anaconda3₩lib₩site-packages (from keras) (0.18.1)
Requirement already satisfied: numpy>=1.9.1 in c:₩program files₩anaconda3₩lib₩site-packages (from keras) (1.14.0)
Requirement already satisfied: keras-preprocessing>=1.0.5 in c:₩program files₩anaconda3₩lib₩site-packages (from keras) (1.0.5)
Requirement already satisfied: keras-applications>=1.0.6 in c:₩program files₩anaconda3₩lib₩site-packages (from keras) (1.0.6)
Requirement already satisfied: pyyaml in c:₩program files₩anaconda3₩lib₩site-packages (from keras) (3.12)
Requirement already satisfied: h5py in c:₩program files₩anaconda3₩lib₩site-packages (from keras) (2.6.0)
Requirement already satisfied: six>=1.9.0 in c:₩program files₩anaconda3₩lib₩site-packages (from keras) (1.11.0)
```

  - Keras 를 설치 한 후에, 다음과 같이 mnist.load_data() 를 통하여 MNIST 를 가져올 수 있음.

입력, 정답
개별 존재!!

```
from keras.datasets import mnist

(x_train_data, t_train_data), (x_test_data, t_test_data) = mnist.load_data()
```

MNIST – 구조 (I)

➢ mnist_train.csv

⇒ mnist_train.csv 파일에는 학습에 이용될 수 있도록 정답(label)이 있는 총 60,000 개의 데이터 존재함. 1개의 데이터는 785개의 숫자가 콤마(,)로 분리되어 있는데, 정답을 나타내는 1개의 숫자와 실제 필기체 숫자 이미지를 나타내는 784 개의 숫자로 구성되어 있음

[예] 다음과 같은 데이터라면, 정답은 첫 번째 나오는 숫자 5 이며, 콤마로 분리되어 있는 나머지 784개의 숫자는 정답인 5라는 필기체 숫자의 이미지를 나타내고 있는 숫자들의 조합임

```
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,
190,253,70,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,35,241,225,160,108,1,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0
,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,
0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,16,93,252,253,187,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,249,253,249,64,0,0,
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
```

➢ mnist_test.csv

⇒ mnist_test.csv 파일에는 총 10,000개의 데이터가 있으며, 학습을 마친 후에 구현된 딥러닝 아키텍처가 얼마나 잘 동작하는지 테스트 하기 위해 사용됨. 테스트 데이터 또한 정답(label)이 포함된 785 개의 숫자로 되어 있음

```
7,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,
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,
0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,
0,0,0,0,0,0,0,0,0,0,0,84,185,159,151,60,36,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,222,254,254,254,254,241,198,198,198,198,1
98,198,198,198,170,52,0,0,0,0,0,0,0,0,0,0,0,0,67,114,72,114,163,227,254,225,254,254,254,250,229,254,254,140,0,0,0,0,0,0,0,0,0,0,
0,0,0,0,0,0,0,17,66,14,67,67,67,59,21,236,254,106,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,83,253,209,18,0,0,0,0,0,0,0,0,
0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,22,233,255,83,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,129,254,238,44,0,0,0,0,0,0,0,0,0,0,0
,0,0,0,0,0,0,0,0,0,0,0,0,59,249,254,62,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,133,254,187,5,0,0,0,0,0,0,0,0,0,0,0,0,0,0
,0,0,0,0,0,0,0,0,0,9,205,248,58,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,126,254,182,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,
0,0,0,0,0,0,75,251,240,57,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,19,221,254,166,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0
,0,0,3,203,254,219,35,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,38,254,254,77,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,
31,224,254,115,1,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,133,254,254,52,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,61,2
42,254,254,52,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,121,254,254,219,40,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,121
,254,207,18,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0
```

MNIST – 구조 (II)

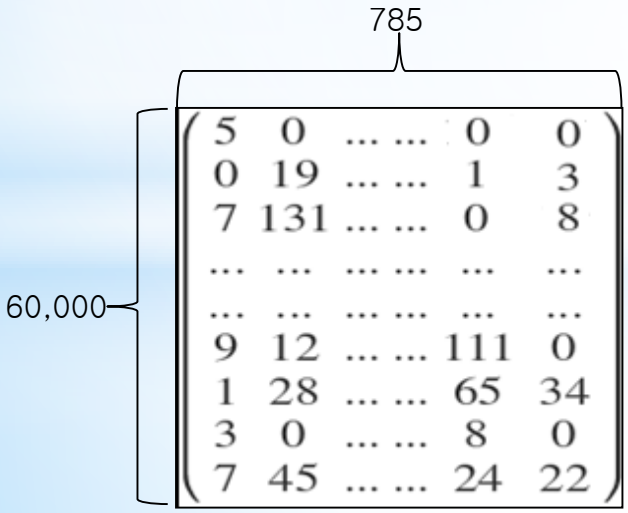
➢ MNIST 가져오기 (numpy.loadtxt 활용)

```
import numpy as np

training_data = np.loadtxt('./mnist_train.csv', delimiter=',', dtype=np.float32)

test_data = np.loadtxt('./mnist_test.csv', delimiter=',', dtype=np.float32)


print("training_data.shape = ", training_data.shape, " ,  test_data.shape = ", test_data.shape)
```

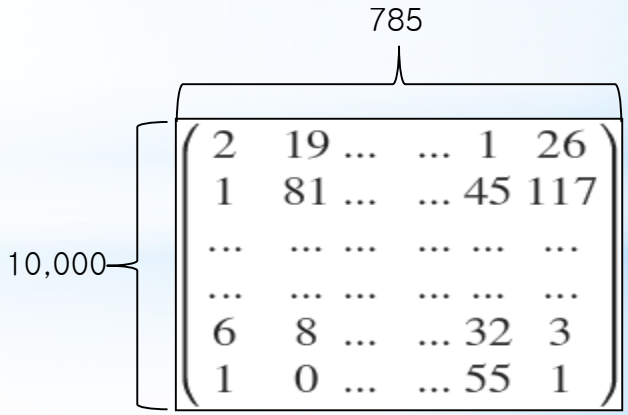
```
training_data.shape =  (60000, 785)  ,  test_data.shape =  (10000, 785)
```

- loadtxt(…) 를 이용하여 mnist_train.csv 로 부터 60,000개의 training data 와 mnist_test.csv 에서 10,000 개의 test data를 2차원 행렬 (matrix) 데이터 타입으로 가져옴

training_data (60,000 X 785)



test_data (10,000 X 785)



MNIST – 구조 (III)

- training_data 행렬

5	0	...	...	0	0
0	19	...	...	1	3
7	131	...	...	0	8
...	...	...	...	...	...
...	...	...	...	...	...
9	12	...	...	111	0
1	28	...	...	65	34
3	0	...	...	8	0
7	45	...	...	24	22

- 레코드 (1개의 행, row)

① 1개의 레코드는 785개의 열(column)로 구성

② 1 열(column)에는 정답이 있음

③ 2 열 (column)부터 마지막 열(column) 까지는 정답을 나타내는 이미지의 색(color)을 나타내는 숫자 값들이 784개 연속으로 있음.

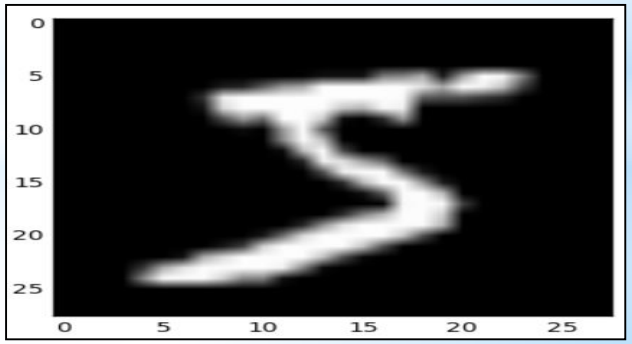
※ 흑백 이미지 표현 할때 숫자 0에 가까울 수록 검은색으로, 255에 가까울수록 하얀색으로 나타내는데 2열부터 마지막 열까지 나열된 숫자가 바로 이미지 색을 나타내는 정보임

- training_data 이미지 표현

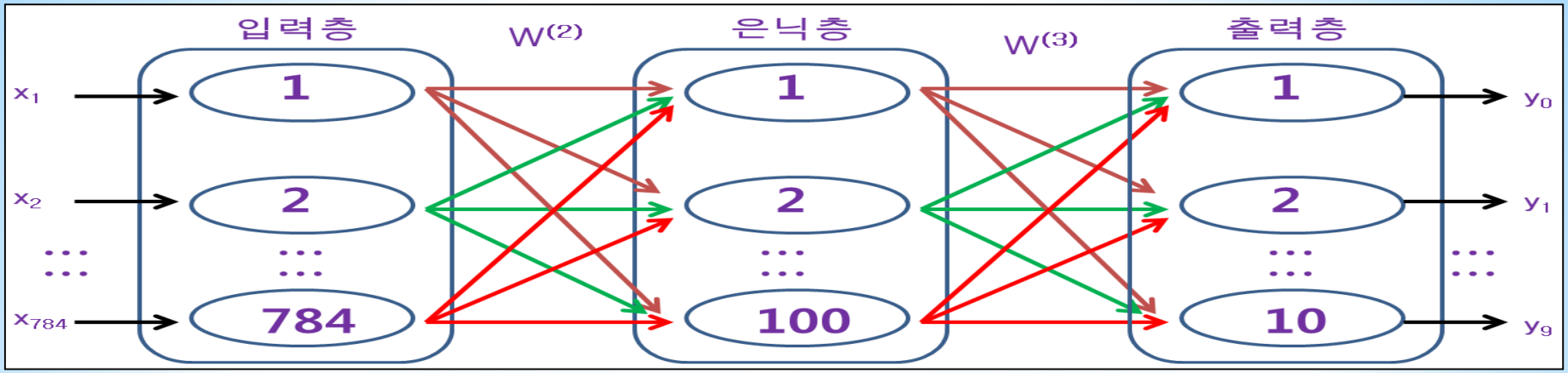
image 로 나타내기 위해서 2열부터 마지막 열까지의 784 개 데이터를 28 X 28 행렬로 reshape 시켜줌

```
import matplotlib.pyplot as plt
%matplotlib inline

img = training_data[0, 1:].reshape(28,28)

plt.imshow(img, cmap='gray')
plt.show()
```

첫번째 레코드 (1st row)



딥러닝 아키텍처 (one-hot encoding)



$W^{(2)}$ = (784 X 100)

$W^{(3)}$ = (100 X 10)

입력층 노드(node)를 입력 데이터 개수와 일치하도록 784개 설정.

즉, training_data 행렬에서 1개의 레코드(1개의 행, row)는 785개의 숫자를 포함하지만, 정답을 나타내는 1열을 제외하면 2열부터 785 열까지 총 784개의 데이터가 숫자 이미지를 나타내므로 노드 개수도 입력데이터와 일치하도록 784개 설정

은닉층 노드(node)를 몇 개로 설정할 것인가는 정해진 규칙이 없으므로 임의로 100개 설정.

출력층 노드(node)는 10개 설정.

즉, 정답은 0~9 中 하나의 숫자이므로 10개의 원소를 갖는 리스트를 만들고, 리스트에서 가장 큰 값을 가지는 인덱스(index)를 정답으로 판단할 수 있도록 출력 노드를 10개로 설정함 (one-hot encoding)

[one-hot encoding 예제] 10 개 출력 노드 값이 다음과 같은 경우, 인덱스가 5인 5번째 노드(y_5) 출력 값이 0.99로 가장 크기 때문에 인덱스 값 5 를 정답 5 로 판단함

정답

입력데이터 784개

5	0	... ...	0	0
0	19	... ...	1	3
7	131	... ...	0	8
...	...	... ...	...	...
...	...	... ...	...	...
9	12	... ...	111	0
1	28	... ...	65	34
3	0	... ...	8	0
7	45	... ...	24	22

정답을 5로 판단함

인덱스	0	1	2	3	4	5	6	7	8	9
값	0.01	0.01	0.01	0.01	0.01	0.99	0.01	0.01	0.01	0.01
노드번호	y_0	y_1	y_2	y_3	y_4	y_5	y_6	y_7	y_8	y_9

[예제] 수치미분을 이용한 MNIST_Test class 구현 및 검증

[은닉층 노드 1개인 MNIST_Test 객체 생성 및 학습]

obj = MNIST_Test(i_nodes, h1_nodes, h2_nodes, o_nodes, learning_rate)

정규화 수행 후 입력데이타 / 정답데이터 분리 후, 반복횟수를 설정한 후

obj.train(input_data, target_data)

[정확도 검증 및 다음과 같은 손실함수 추세 확인 (matplotlib를 이용)]

obj.accuracy(test_input_data, test_target_data)