

12주차(2/3)

MNIST Dataset

파이썬으로 배우는 기계학습

한동대학교
김영섭 교수

MNIST Dataset

- 학습 목표
 - **MNIST Dataset** 이 무엇인지 이해한다.
 - **MNIST Dataset** 을 사용하는 방법을 익힌다.
 - **MNIST Dataset** 을 학습하는 신경망을 설계해본다.
- 학습 내용
 - **MNIST** 자료 개요
 - **MNIST** 자료 읽기
 - **MNSIT** 자료 전처리
 - **MNIST** 신경망 설계
 - 원-핫-인코딩(**one-hot encoding**)

1. MNIST 자료 개요: MNIST란?

- MNIST란?

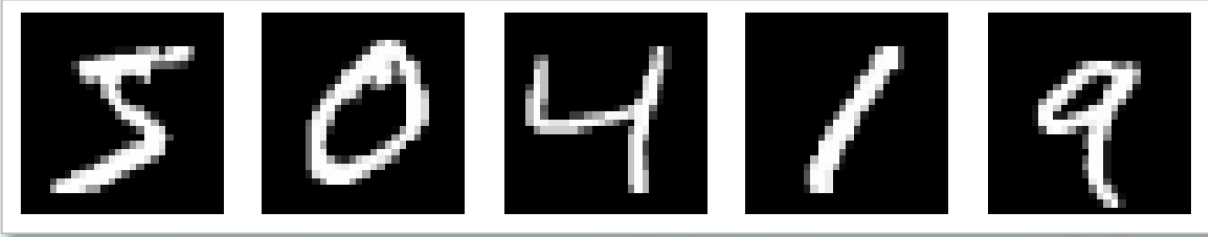
```
print('Hello World!')
```



Hello World!

1. MNIST 자료 개요: MNIST란?

- MNIST란?



```
print('Hello World!')
```



Hello World!

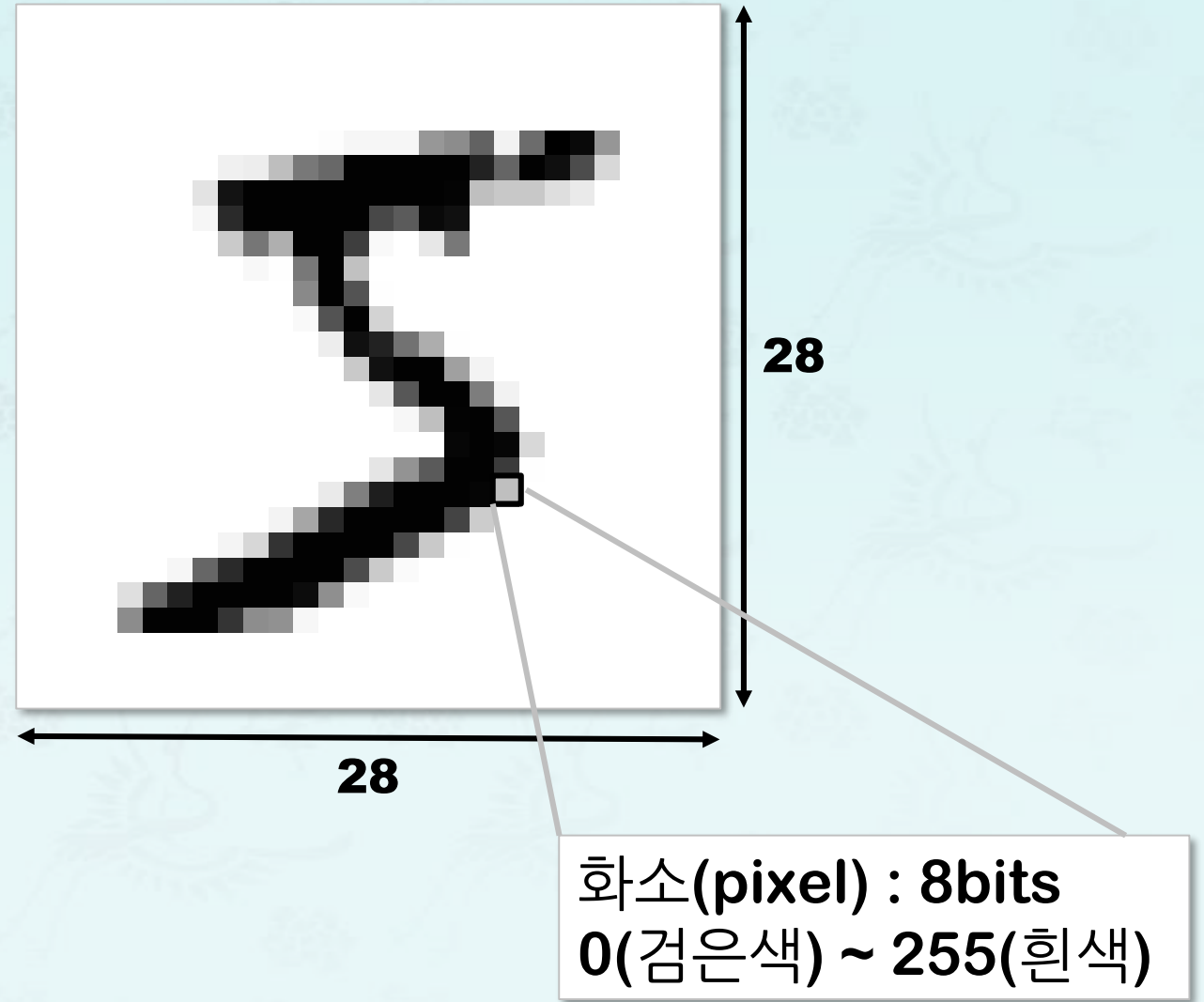
- NIST: National Institute of Standards and Technology
- 미국 국립 표준 기술 연구소
- MNIST: Modified NIST
- 손으로 쓴 숫자들의 이미지

1. MNIST 자료 개요: 자료 형태

- 자료 형태
 - 훈련용 이미지: **60,000**장
 - 테스트 이미지: **10,000**장

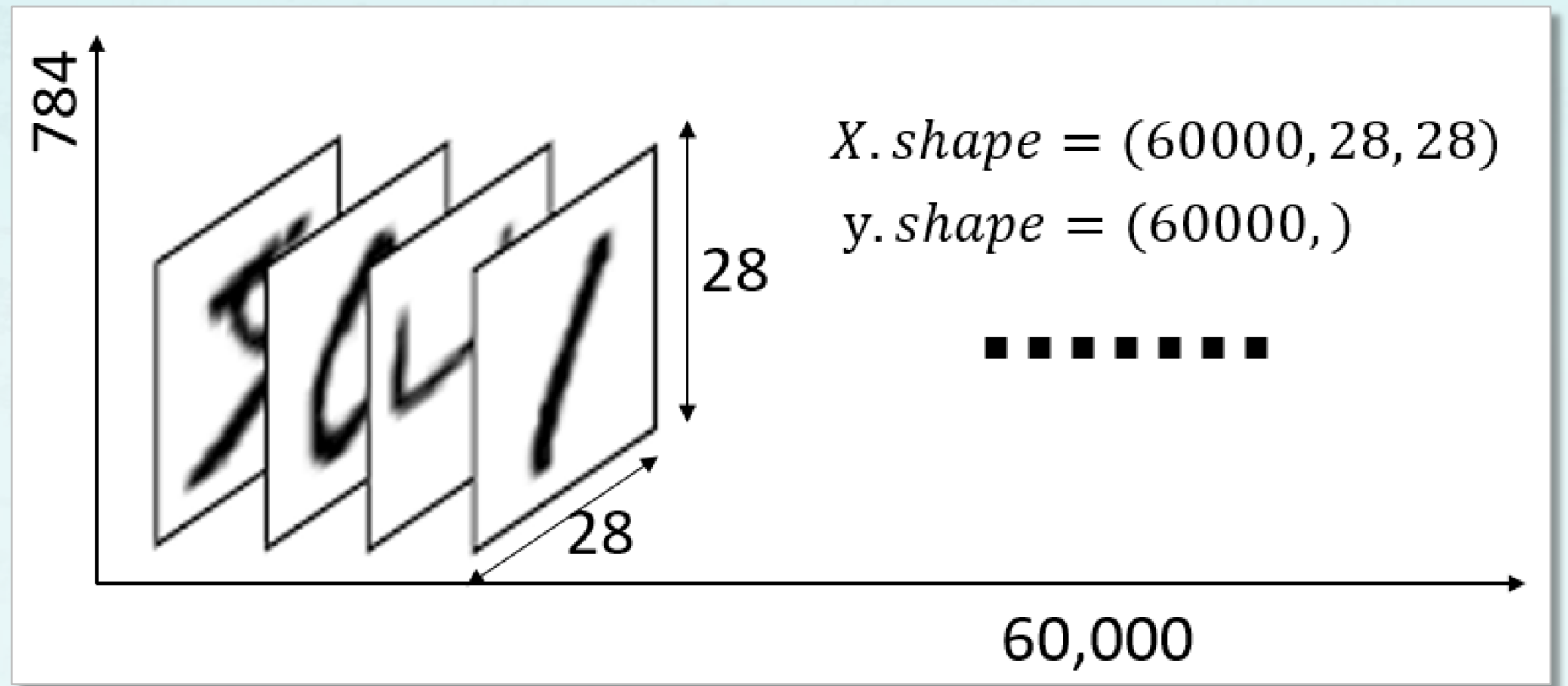
1. MNIST 자료 개요: 자료 형태

- 자료 형태
 - 훈련용 이미지: **60,000**장
 - 테스트 이미지: **10,000**장



1. MNIST 자료 개요: 자료 형태

- 자료 형태
 - 훈련용 이미지: **60,000**장
 - 테스트 이미지: **10,000**장
 - 훈련자료 형상



1. MNIST 자료 개요: 자료 형태

- 자료 형태
 - 훈련용 이미지: **60,000**장
 - 테스트 이미지: **10,000**장
 - 훈련자료 형상
- 저장소
 - <http://yann.lecun.com/exdb/mnist/>

2. MNIST 자료 읽기: `load_mnist()`

2. MNIST 자료 읽기: load_mnist()

```
def load_mnist(normalize=True, flatten=True):  
    ...
```

2. MNIST 자료 읽기: load_mnist()

```
def load_mnist(normalize=True, flatten=True):  
    ...
```

- 매개변수
 - **normalize**: 정규화
 - **flatten = True**: 1차원 배열 – 학습용으로 사용
 - **flatten = False**: 2차원 배열 – 이미지 출력용

2. MNIST 자료 읽기: load_mnist()

```
def load_mnist(normalize=True, flatten=True):  
    ...
```

- 매개변수
 - **normalize**: 정규화
 - **flatten = True**: 1차원 배열 – 학습용으로 사용
 - **flatten = False**: 2차원 배열 – 이미지 출력용
- 반환값
 - (훈련 이미지, 훈련 레이블), (시험 이미지, 시험 레이블)

2. MNIST 자료 읽기: load_mnist()

```
def load_mnist(normalize=True, flatten=True):  
    ...
```

- 매개변수
 - **normalize**: 정규화
 - **flatten = True**: 1차원 배열 – 학습용으로 사용
 - **flatten = False**: 2차원 배열 – 이미지 출력용
- 반환값
 - (훈련 이미지, 훈련 레이블), (시험 이미지, 시험 레이블)
- 사용예제

```
(X, y), (Xtest, ytest) = load_mnist()  
(X, y), (Xtest, ytest) = load_mnsit(normalize=False,  
                                     flatten=False)
```

2. MNIST 자료 읽기: pickle

- 자료 크기에 따른 성능 문제
 - 해결책:

2. MNIST 자료 읽기: pickle

- 자료 크기에 따른 성능 문제
 - 해결책:

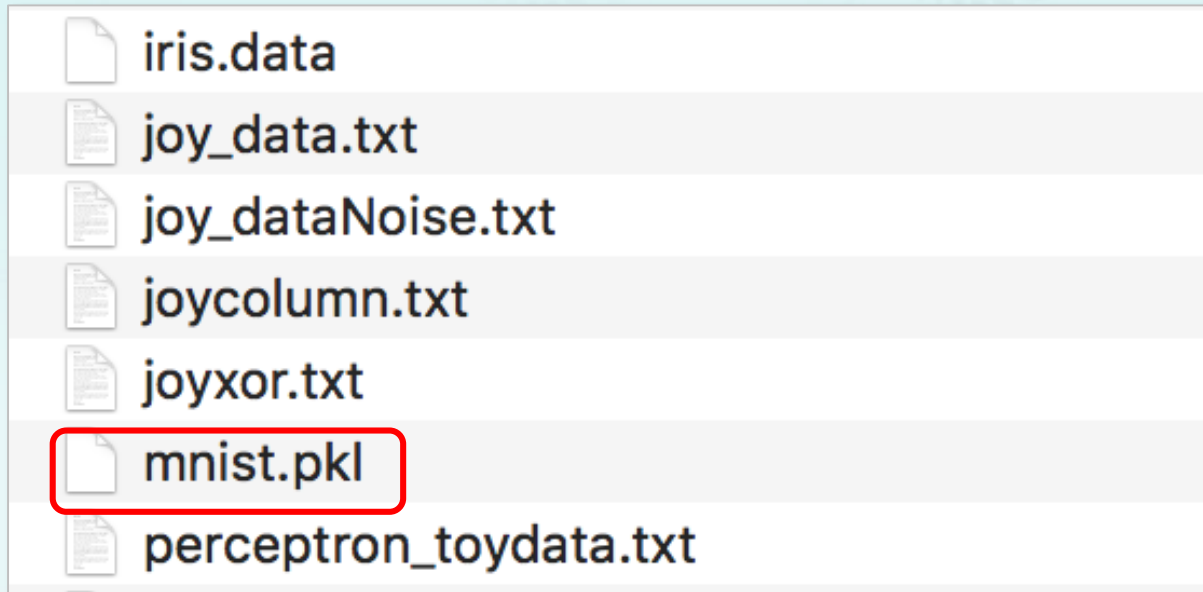
```
def iris_data(standardized=False, shuffled=False):  
    datafile = 'data/iris.data'  
    if os.path.isfile(datafile):  
        df = pd.read_csv(datafile)  
    else :  
        datafile = 'https://archive.ics.uci.edu/ml/'  
        'machine-learning-databases/iris/iris.data'  
        df = pd.read_csv(datafile, header=None)  
    ...
```

```
def joy_data(standardized=False, shuffled=False):  
    return getXy('data/joy_data.txt',  
                 standardized, shuffled)
```

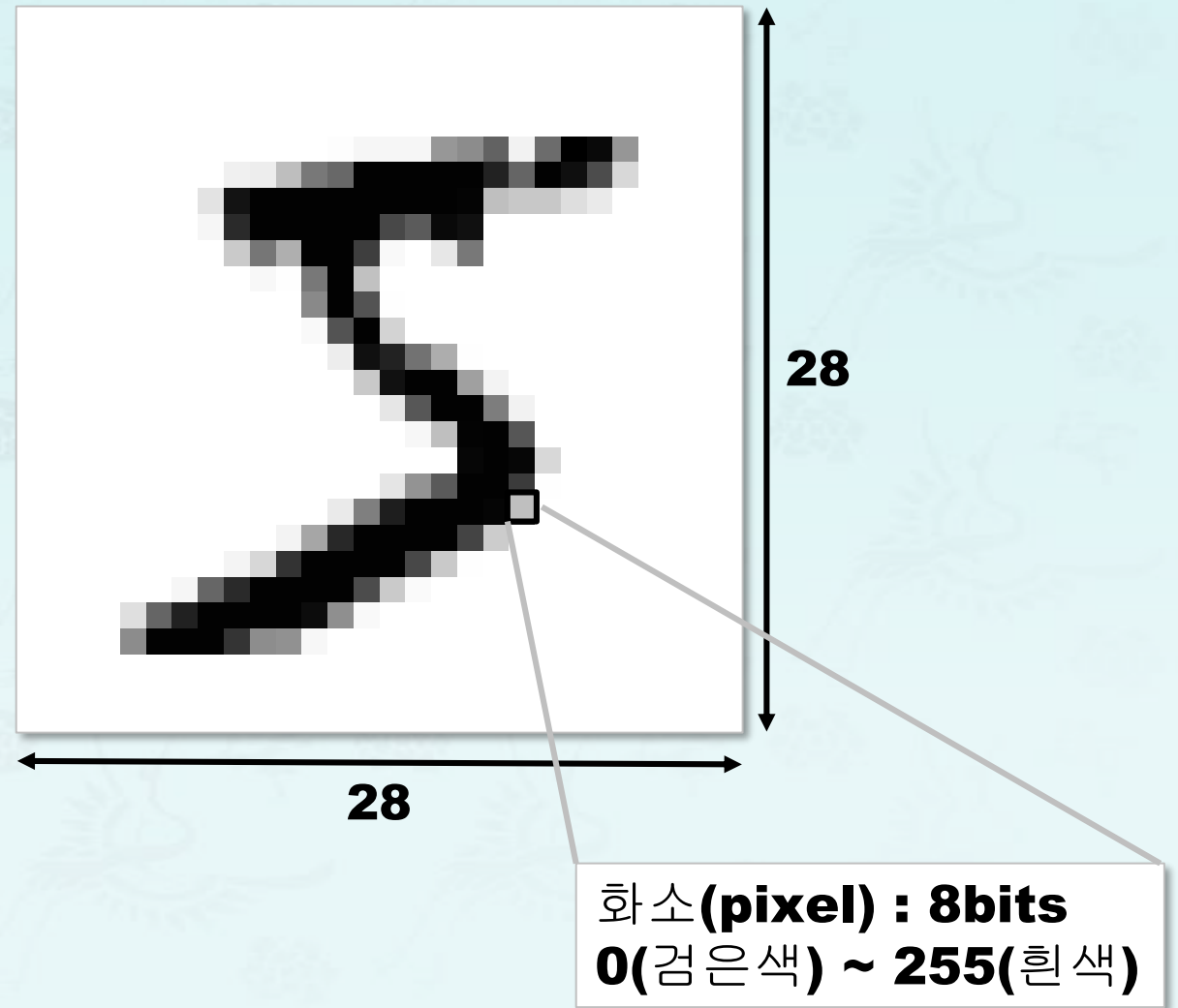
```
def toy_data(standardized=False, shuffled=False):  
    return getXy('data/toy_data.txt',  
                 standardized, shuffled)
```

2. MNIST 자료 읽기: pickle

- 자료 크기에 따른 성능 문제
 - 해결책: **pickle**



3. MNIST 자료 전처리: 정규화



3. MNIST 자료 전처리: 정규화

```
1 import joy
2 import numpy as np
3
4 (X, y), (Xt, yt) = joy.load_mnist()
5 X_std = X[:]
6 X_std = np.asfarray(X)/255.0 * 0.99 + 0.01
7 print(X_std[0])
```



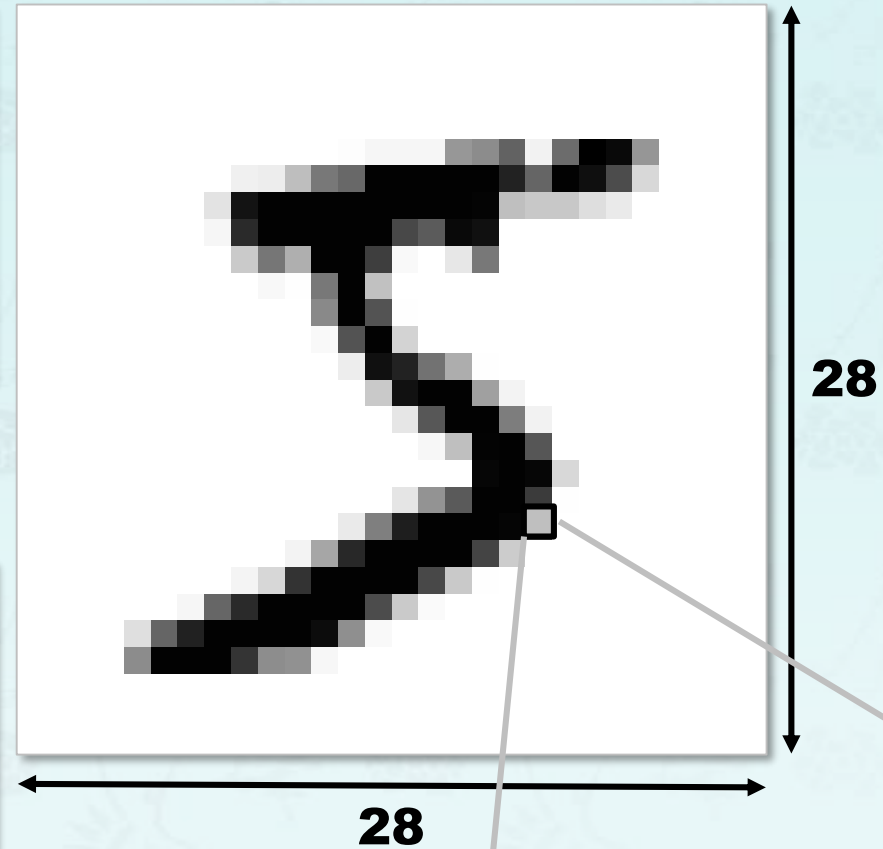
화소(pixel) : 8bits
0.01(검은색) ~ 1.00(흰색)

3. MNIST 자료 전처리: 정규화

```
1 import joy
2 import numpy as np
3
4 (X, y), (Xt, yt) = joy.load_mnist()
5 X_std = X[:]
6 X_std = np.asfarray(X)/255.0 * 0.99 + 0.01
7 print(X_std[0])
```



36	0.01027405	0.01027405	0.01027405	0.01191834	0.01207059	0.012664
0.01	0.01039585	0.01252734	0.01388235	0.01376055	0.01193356	0.01
0.01	0.01	0.01	0.01	0.01	0.01	0.01
0.01	0.01	0.01	0.01	0.01045675	0.0105481	0.011431
14	0.01234464	0.01258824	0.0138519	0.0138519	0.0138519	0.013851
9	0.0138519	0.01342561	0.01261869	0.0138519	0.01368443	0.012968
86	0.01097439	0.01	0.01	0.01	0.01	0.01
0.01	0.01	0.01	0.01	0.01	0.010746	
02	0.01362353	0.0138519	0.0138519	0.0138519	0.0138519	0.013851



화소(pixel) : 8bits
0.01(검은색) ~ 1.00(흰색)

3. MNIST 자료 전처리: 이미지 시각화

3. MNIST 자료 전처리: 이미지 시각화

```
1 import joy
2 (X, y), (Xt, yt) = joy.load_mnist(
3     normalize = False, flatten = False)
4 joy.save_mnist_csv(X[:3], y[:3], 'train3.csv')
```

3. MNIST 자료 전처리: 이미지 시각화

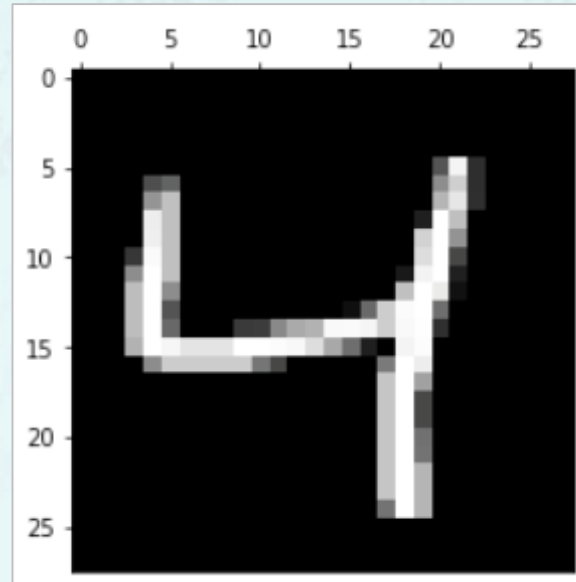
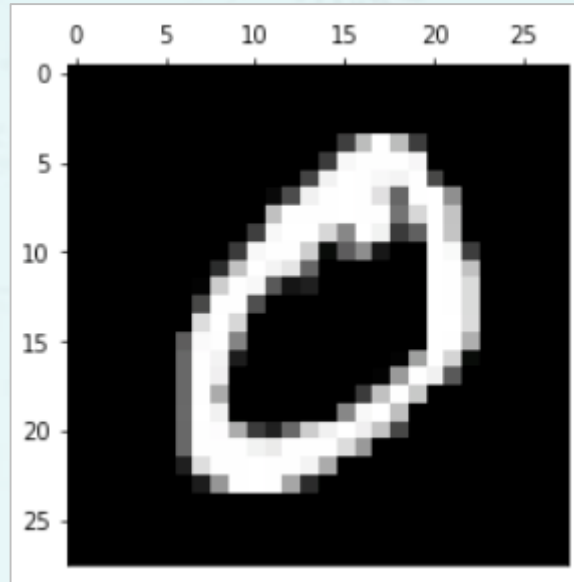
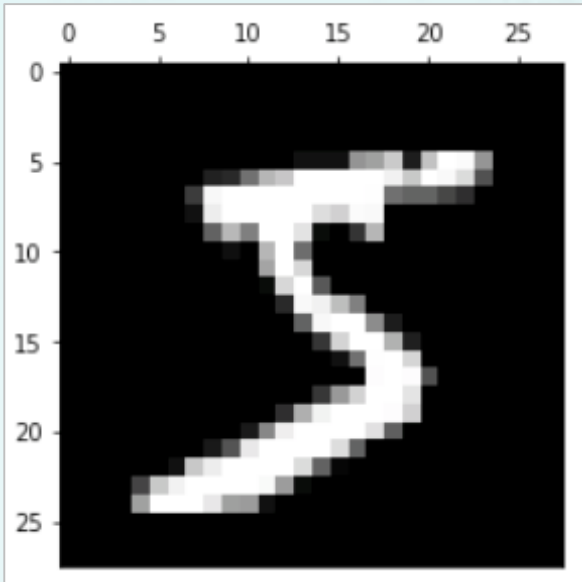
```
1 import joy
2 (X, y), (Xt, yt) = joy.load_mnist(
3     normalize = False, flatten = False)
4 joy.save_mnist_csv(X[:3], y[:3], 'train3.csv')
```

```
1 import joy
2 X, y = joy.read_mnist_csv('train3.csv')
3 for i, yi in enumerate(y):
4     joy.show_mnist(X[i])
```

3. MNIST 자료 전처리: 이미지 시각화

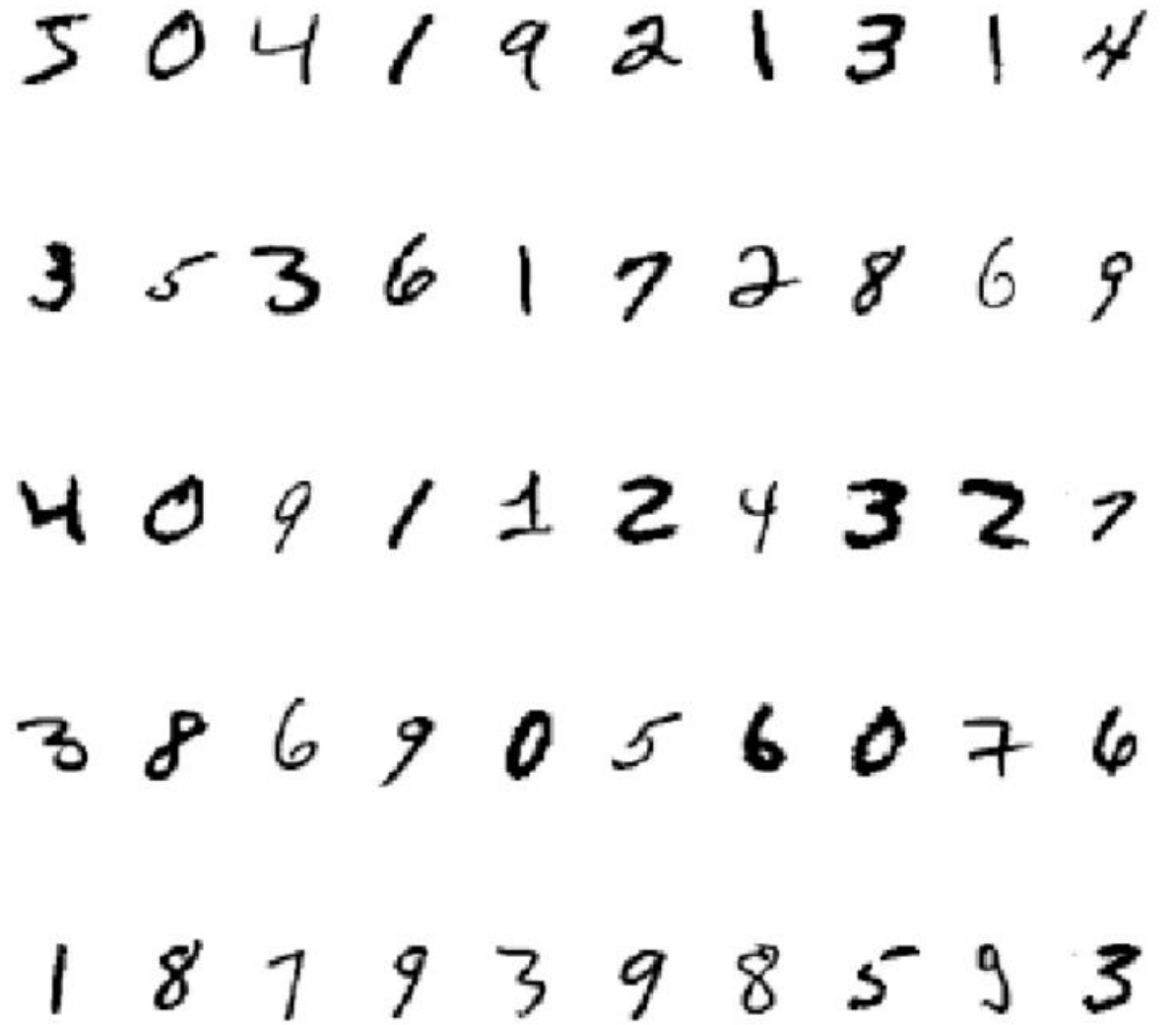
```
1 import joy
2 (X, y), (Xt, yt) = joy.load_mnist(
3     normalize = False, flatten = False)
4 joy.save_mnist_csv(X[:3], y[:3], 'train3.csv')
```

```
1 import joy
2 X, y = joy.read_mnist_csv('train3.csv')
3 for i, yi in enumerate(y):
4     joy.show_mnist(X[i])
```

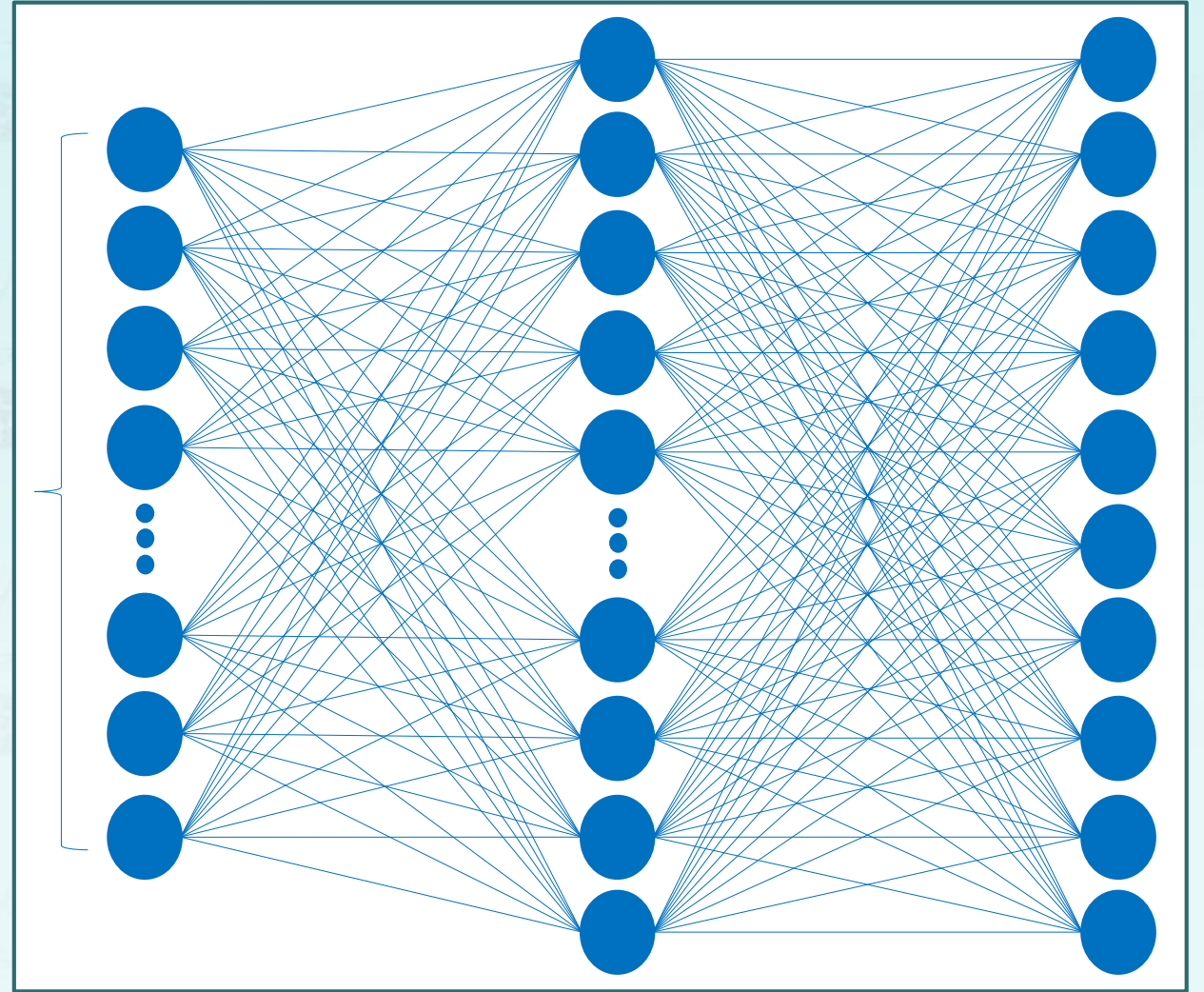


3. MNIST 자료 전처리: 이미지 시각화

```
1 import joy
2 (X, y), (Xt, yt) = joy.load_mnist(
3     flatten=False)
4 joy.show_mnist_grid(X[:50])
```

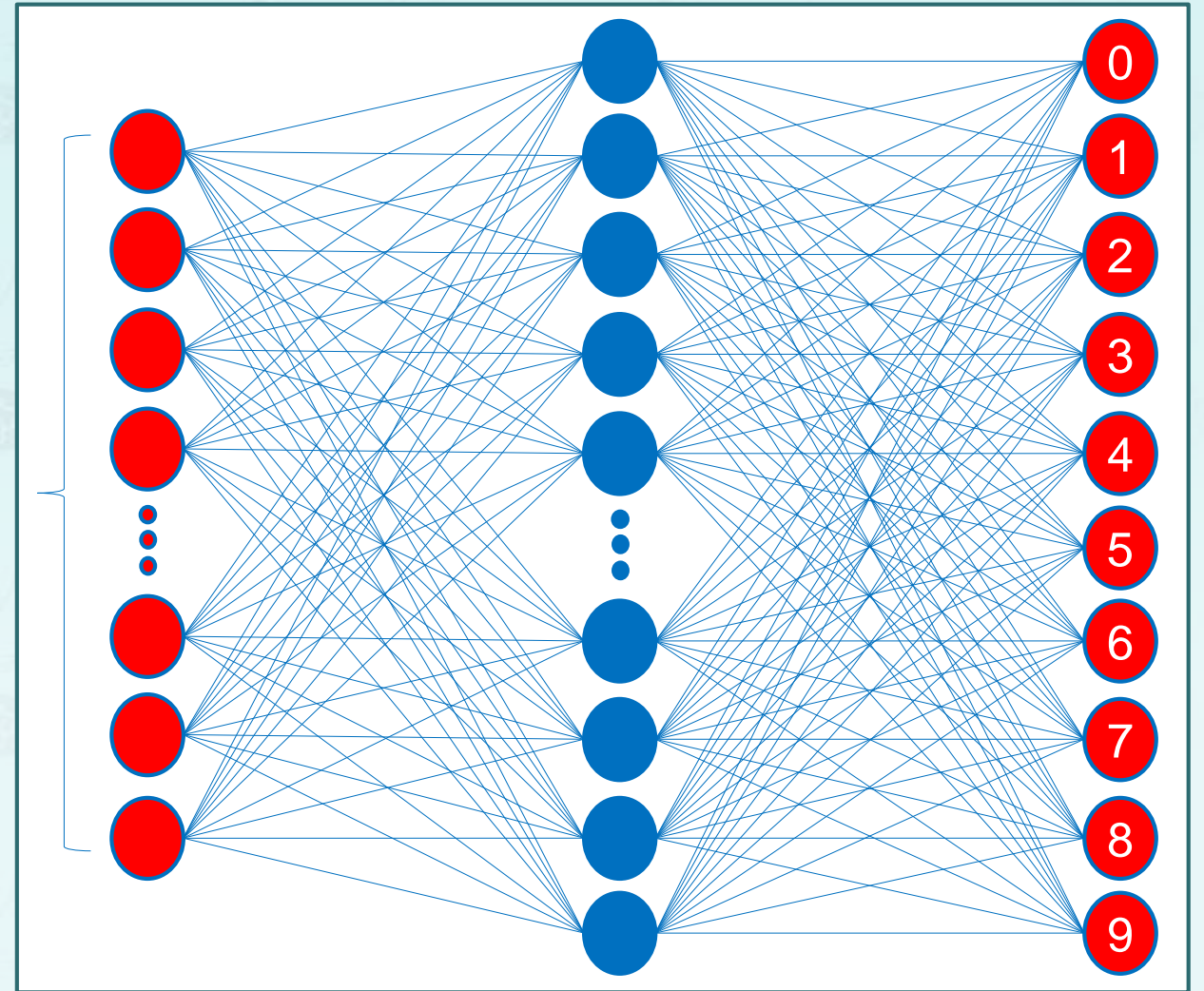


4. MNIST 신경망 설계: 각 층별 노드 수



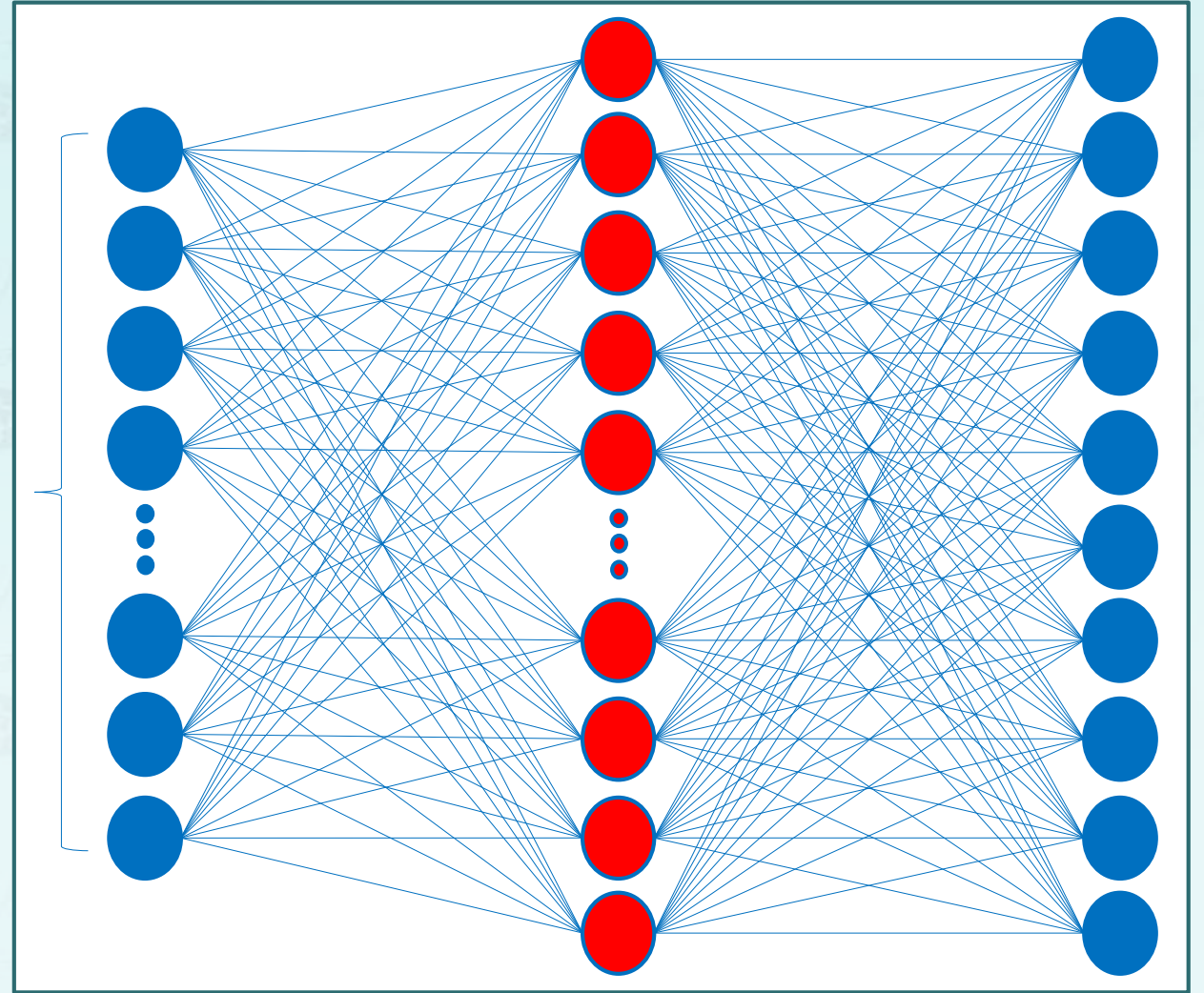
4. MNIST 신경망 설계: 출력층

- 입력층 노드 수: $28 \times 28 = 784$
- 출력층 노드 수: 10

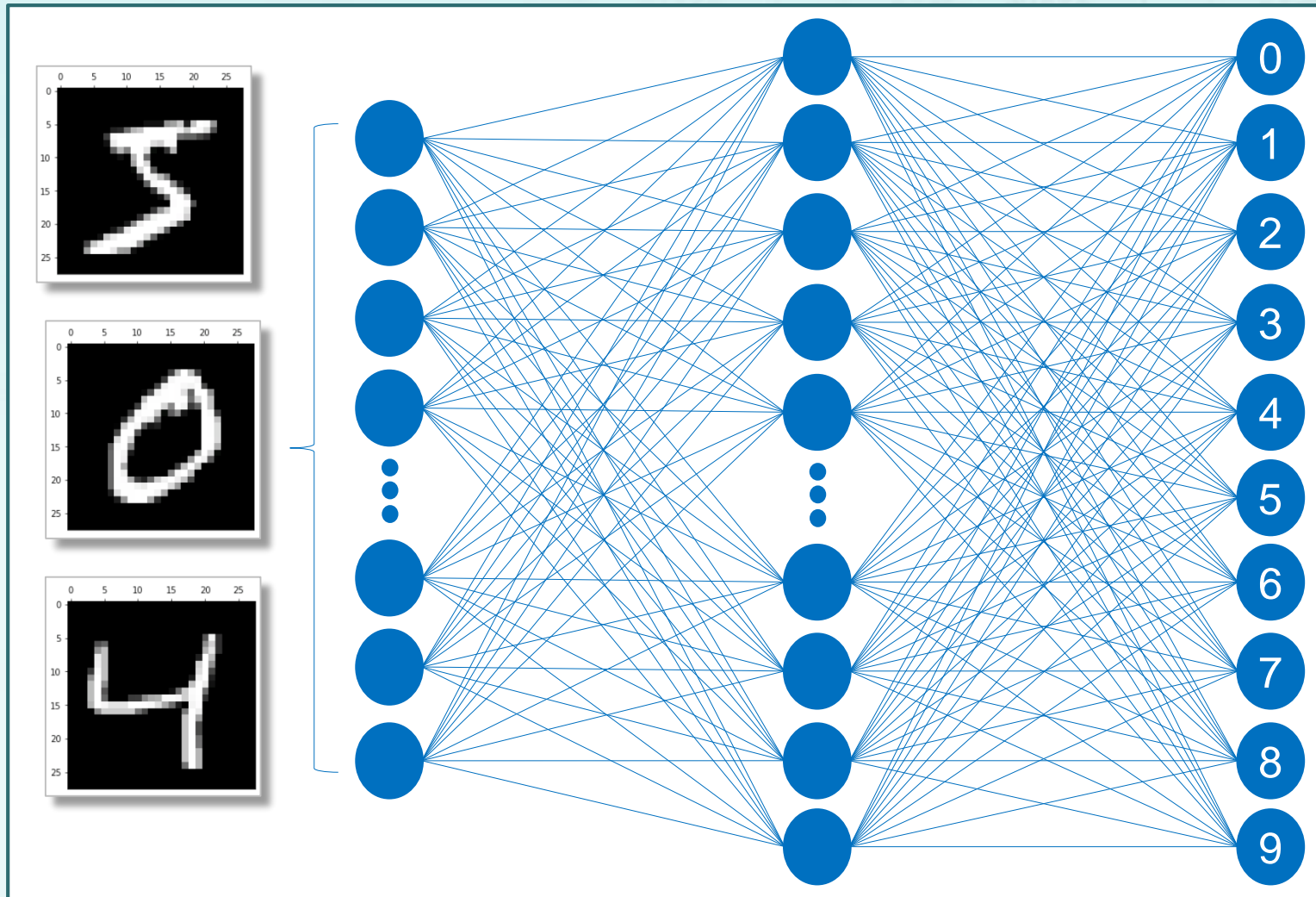


4. MNIST 신경망 설계: 은닉층

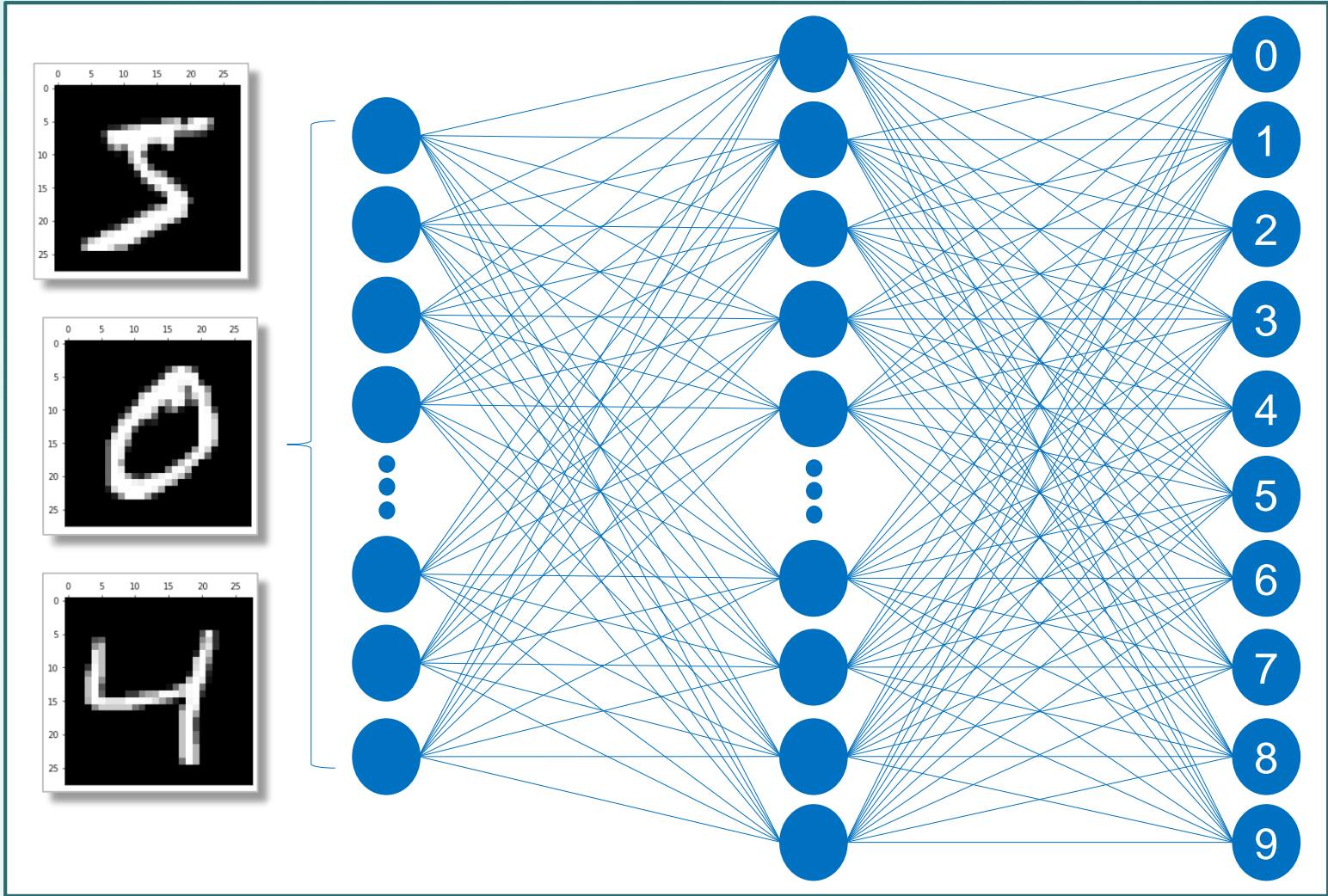
- 입력층 노드 수: $28 \times 28 = 784$
- 출력층 노드 수: 10



4. MNIST 신경망 설계: 출력층의 결과

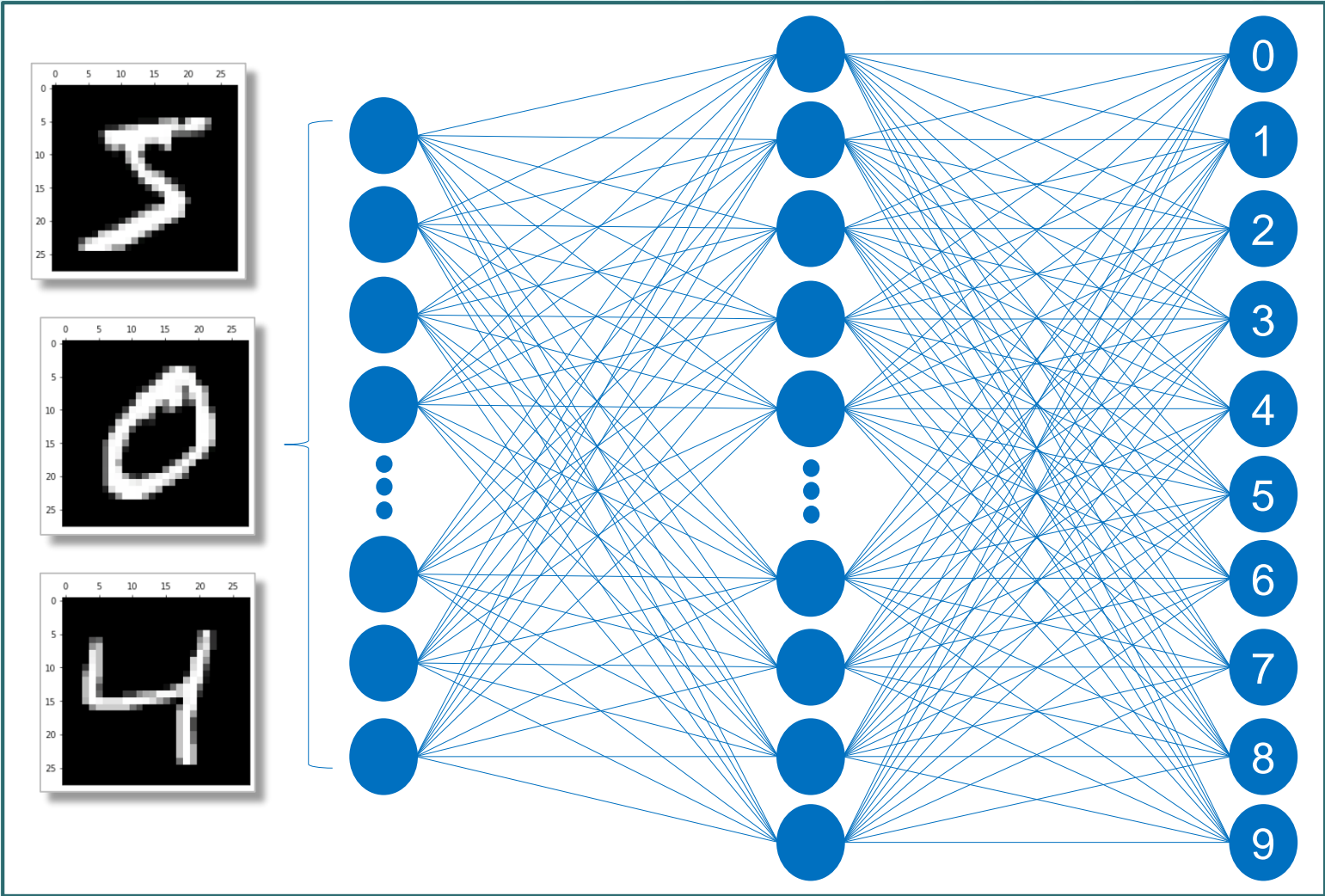


4. MNIST 신경망 설계: 출력층의 결과



$x^0 = 5$	$x^1 = 0$	$x^2 = 4$	클래스 레이블		
\hat{y}^0	\hat{y}^1	\hat{y}^2	y^0	y^1	y^2
0.00	0.84	0.02	0	1	0
0.00	0.00	0.00	0	0	0
0.01	0.01	0.01	0	0	0
0.00	0.01	0.01	0	0	0
0.00	0.02	0.60	0	0	1
0.96	0.00	0.01	1	0	0
0.00	0.00	0.01	0	0	0
0.00	0.10	0.00	0	0	0
0.02	0.00	0.01	0	0	0
0.01	0.02	0.33	0	0	0

4. MNIST 신경망 설계: 출력층의 결과

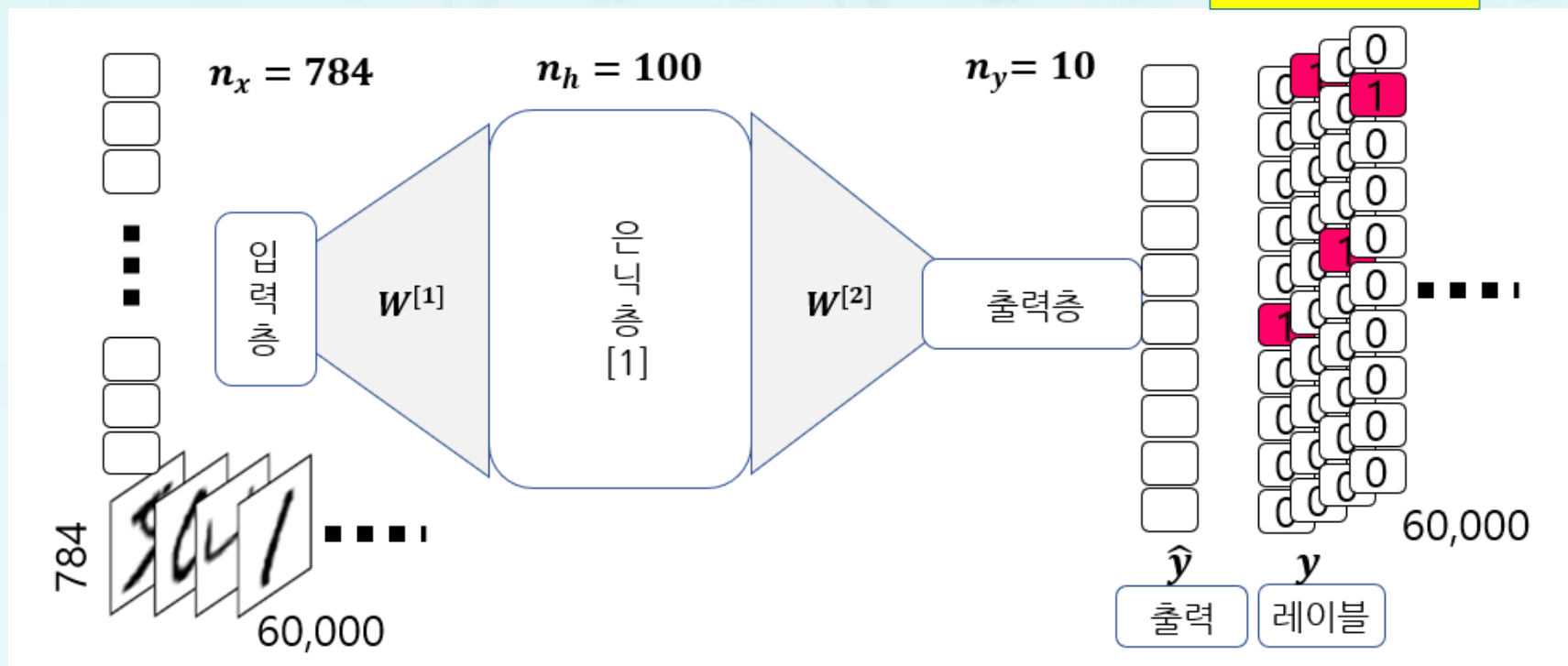


$x^0 = 5$	$x^1 = 0$	$x^2 = 4$	클래스 레이블		
\hat{y}^0	\hat{y}^1	\hat{y}^2	y^0	y^1	y^2
0.00	0.84	0.02	0	1	0
0.00	0.00	0.00	0	0	0
0.01	0.01	0.01	0	0	0
0.00	0.01	0.01	0	0	0
0.00	0.02	0.60	0	0	1
0.96	0.00	0.01	1	0	0
0.00	0.00	0.01	0	0	0
0.00	0.10	0.00	0	0	0
0.02	0.00	0.01	0	0	0
0.01	0.02	0.33	0	0	0

5. 원-핫-인코딩: one-hot encoding

$x^0 = 5$	$x^1 = 0$	$x^2 = 4$	one-hot encoding		
\hat{y}^0	\hat{y}^1	\hat{y}^2	y^0	y^1	y^2
0.00	0.84	0.02	0	1	0
0.00	0.00	0.00	0	0	0
0.01	0.01	0.01	0	0	0
0.00	0.01	0.01	0	0	0
0.00	0.02	0.60	0	0	1
0.96	0.00	0.01	1	0	0
0.00	0.00	0.01	0	0	0
0.00	0.10	0.00	0	0	0
0.02	0.00	0.01	0	0	0
0.01	0.02	0.33	0	0	0

5. 원-핫-인코딩: one-hot encoding



$x^0 = 5$	$x^1 = 0$	$x^2 = 4$	one-hot encoding		
\hat{y}^0	\hat{y}^1	\hat{y}^2	y^0	y^1	y^2
0.00	0.84	0.02	0	1	0
0.00	0.00	0.00	0	0	0
0.01	0.01	0.01	0	0	0
0.00	0.01	0.01	0	0	0
0.00	0.02	0.60	0	0	1
0.96	0.00	0.01	1	0	0
0.00	0.00	0.01	0	0	0
0.00	0.10	0.00	0	0	0
0.02	0.00	0.01	0	0	0
0.01	0.02	0.33	0	0	0

5. 원-핫-인코딩: one-hot encoding

```
1 def one_hot_encoding(y, n_y, modified=True):
2     yhot = np.eye(n_y)[
3         np.array(y, dtype='int32').flatten()
4     ]
5     if modified:
6         yhot[yhot == 0] = 0.01
7         yhot[yhot == 1] = 0.99
8     return yhot
```

5. 원-핫-인코딩: one-hot encoding

- `np.eye(n_y)`:
 - $(n_y \times n_y)$ 형상의 단위행렬

```
1 def one_hot_encoding(y, n_y, modified=True):
2     yhot = np.eye(n_y)[
3         np.array(y, dtype='int32').flatten()
4     ]
5     if modified:
6         yhot[yhot == 0] = 0.01
7         yhot[yhot == 1] = 0.99
8     return yhot
```

5. 원-핫-인코딩: one-hot encoding

- `np.eye(n_y)`:
 - $(n_y \times n_y)$ 형상의 단위행렬
- `[]`:
 - 원-핫-인코딩 행렬 제작

```
1 def one_hot_encoding(y, n_y, modified=True):
2     yhot = np.eye(n_y)[
3         np.array(y, dtype='int32').flatten()
4     ]
5     if modified:
6         yhot[yhot == 0] = 0.01
7         yhot[yhot == 1] = 0.99
8     return yhot
```

5. 원-핫-인코딩: one-hot encoding

- `np.eye(n_y)`:
 - $(n_y \times n_y)$ 형상의 단위행렬
- `[]`:
 - 원-핫-인코딩 행렬 제작

```
1 def one_hot_encoding(y, n_y, modified=True):
2     yhot = np.eye(n_y)[
3         np.array(y, dtype='int32').flatten()
4     ]
5     if modified:
6         yhot[yhot == 0] = 0.01
7         yhot[yhot == 1] = 0.99
8     return yhot
```

$$\text{np.eye}(n_y)[y] = \begin{pmatrix} 1 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 \\ \vdots & \vdots & \vdots & \vdots & \vdots \\ 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \end{pmatrix} \dots \begin{pmatrix} 0 \\ 0 \\ 0 \\ 0 \\ \vdots \\ 0 \\ 0 \\ 1 \end{pmatrix} \begin{pmatrix} 5 \\ 0 \\ 4 \\ 1 \\ 9 \\ \vdots \\ \vdots \\ \vdots \\ 3 \end{pmatrix}$$

5. 원-핫-인코딩: one-hot encoding

- `np.eye(n_y)`:
 - $(n_y \times n_y)$ 형상의 단위행렬
- `[]`:
 - 원-핫-인코딩 행렬 제작

```
1 def one_hot_encoding(y, n_y, modified=True):
2     yhot = np.eye(n_y)[
3         np.array(y, dtype='int32').flatten()
4     ]
5     if modified:
6         yhot[yhot == 0] = 0.01
7         yhot[yhot == 1] = 0.99
8     return yhot
```

$$\text{np.eye}(n_y)[y] = \begin{pmatrix} 1 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 \\ \vdots & \vdots & \vdots & \vdots & \vdots \\ 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \end{pmatrix} \dots \begin{pmatrix} 0 \\ 0 \\ 0 \\ 0 \\ \vdots \\ 0 \\ 0 \\ 1 \end{pmatrix} \begin{pmatrix} 5 \\ 0 \\ 4 \\ 1 \\ 9 \\ \vdots \\ \vdots \\ 3 \end{pmatrix} = \begin{pmatrix} 0 & 0 & 0 & 0 & 1 \\ 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 \\ 0 & 1 & 0 & 0 & 0 \\ \vdots & \vdots & \vdots & \vdots & \vdots \\ 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 \end{pmatrix} \dots \begin{pmatrix} 0 \\ 0 \\ 0 \\ 0 \\ \vdots \\ 0 \\ 0 \\ 0 \end{pmatrix}$$

MNIST Dataset

- 학습 정리
 - **MNIST Dataset** 이 무엇인지 이해한다.
 - **MNIST Dataset** 을 사용하는 방법을 익힌다.
 - **MNIST Dataset** 을 학습하는 신경망을 설계해본다.
- 차시 예고
 - **11-3 MNIST 경사하강법 1**

12주차(2/3)

MNIST Dataset

파이썬으로 배우는 기계학습

한동대학교
김영섭 교수

