

7주차(2/3)

순방향 신경망 예제

파이썬으로 배우는 기계학습

한동대학교
김영섭 교수

순방향 신경망 예제

- 학습 목표
 - 예제를 통해 순방향 신경망을 깊이 있게 이해한다
- 학습 내용
 - **MNIST** 자료 이해 하기
 - 다층 신경망 설계 하기
 - 순방향 신경망 신호처리
 - 순방향 신경망 예제 구현 하기

1. 다층 신경망: 입력 자료

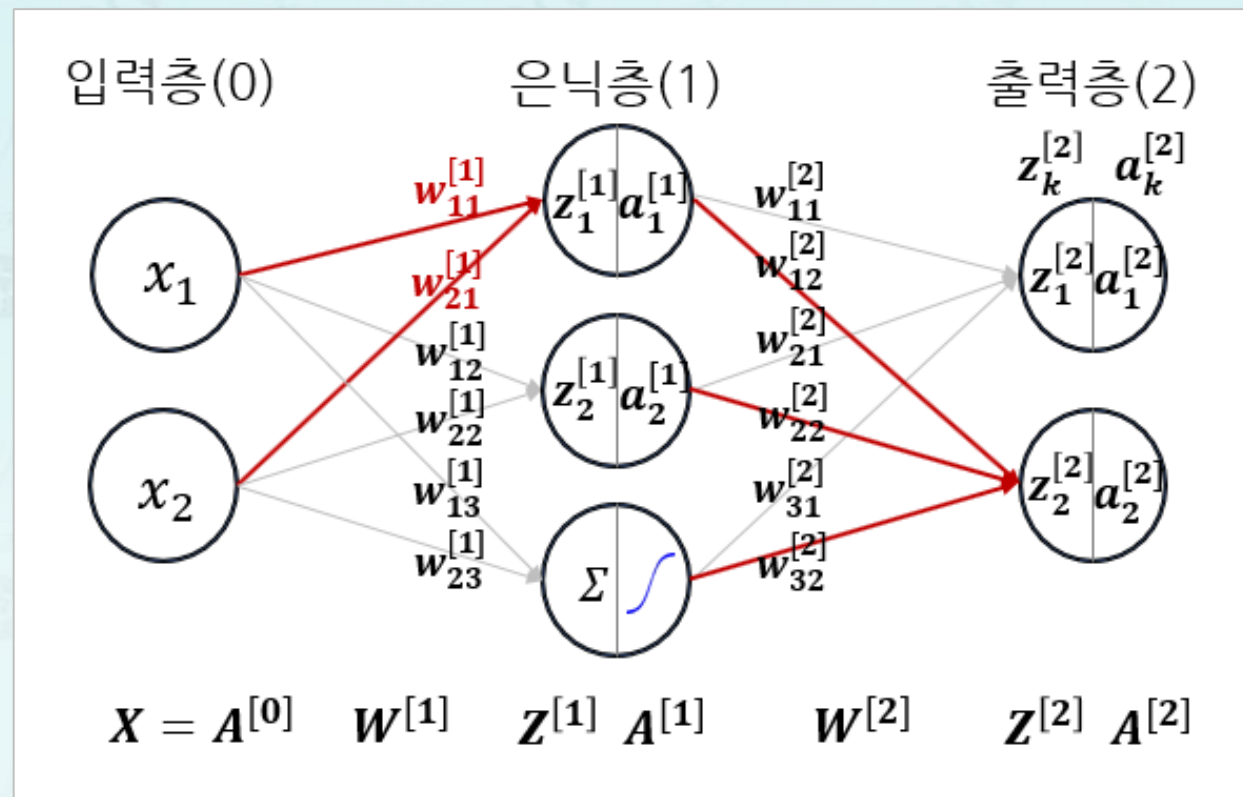
1. 다층 신경망: 입력 자료

- MNIST



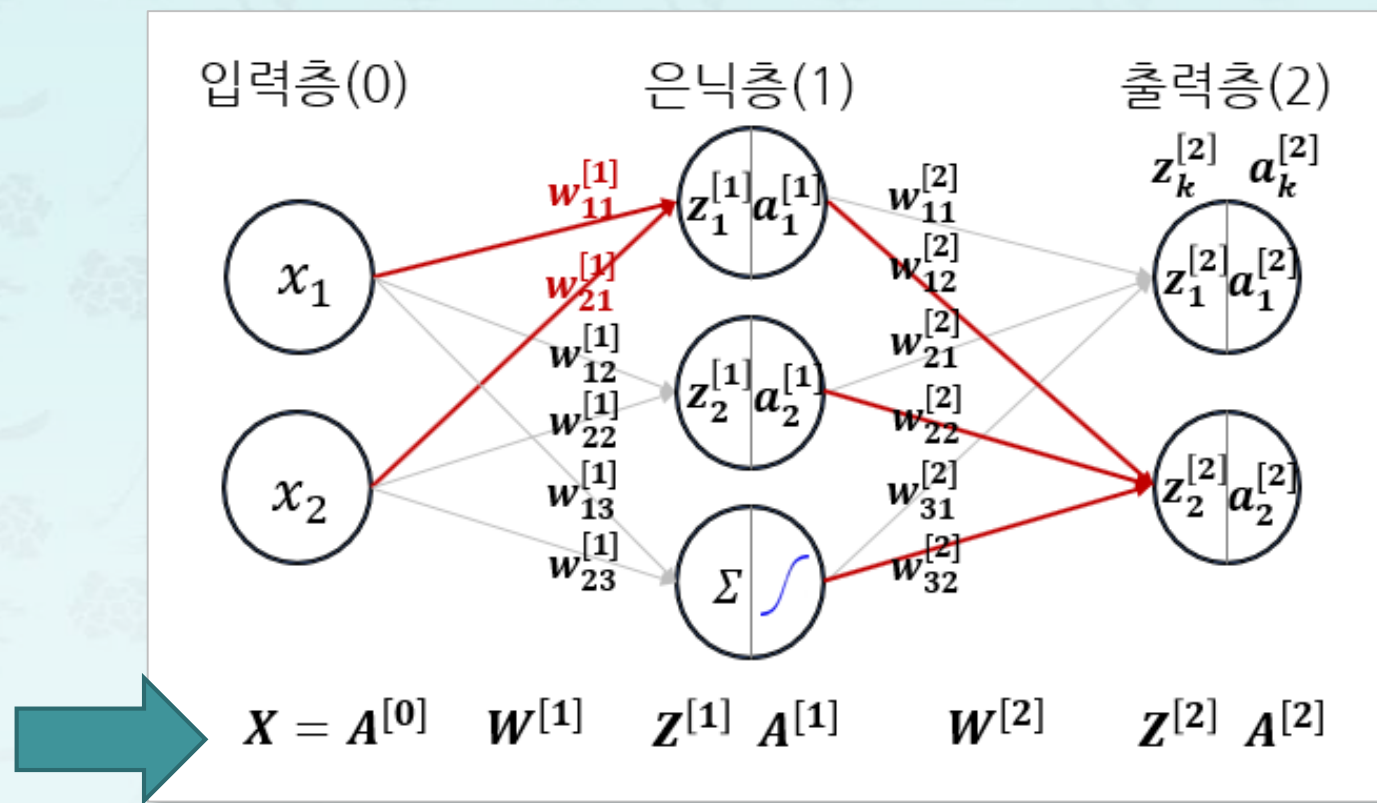
1. 다층 신경망: 신호 표기(복습)

- 다층 신경망



1. 다층 신경망: 신호 표기(복습)

- 다층 신경망

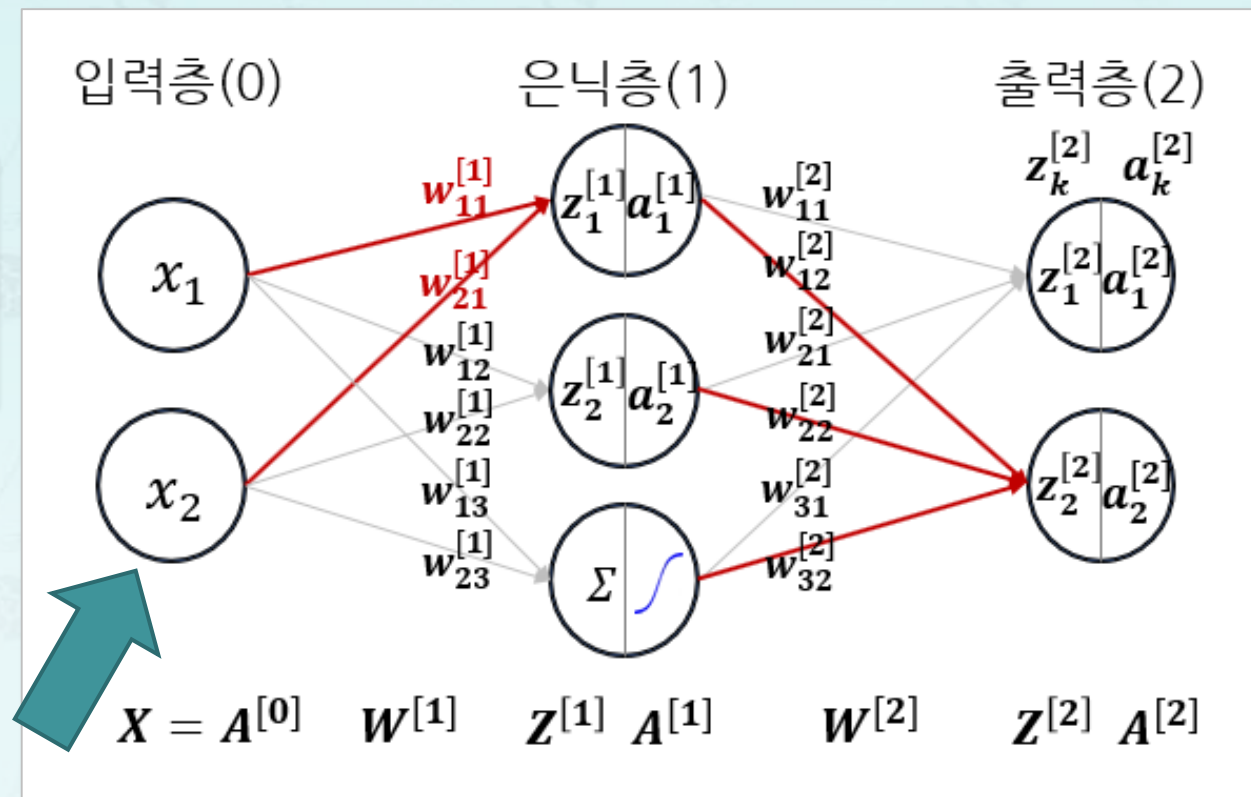


1. 다층 신경망: 입력 자료

- MNIST



- 다층 신경망

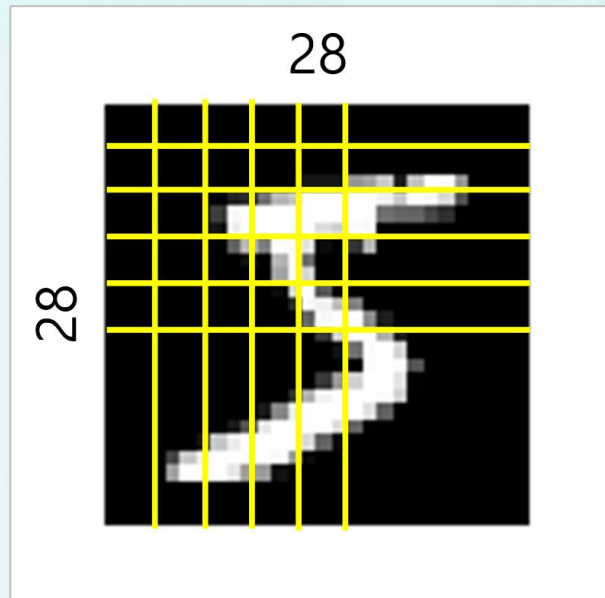
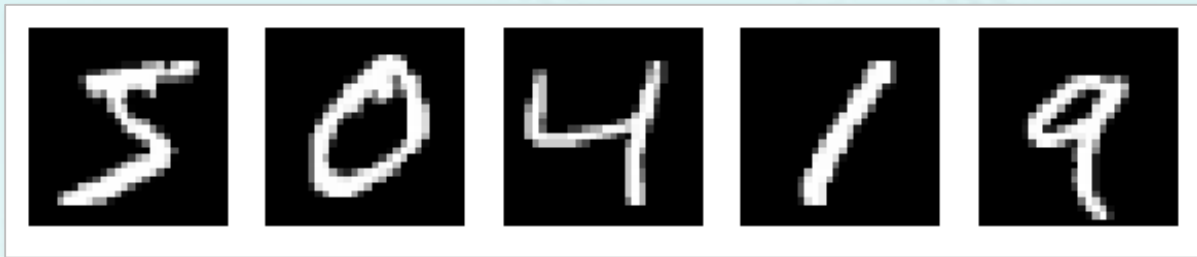


1. 다층 신경망: 입력 자료

- MNIST

- 28x28

- 다층 신경망

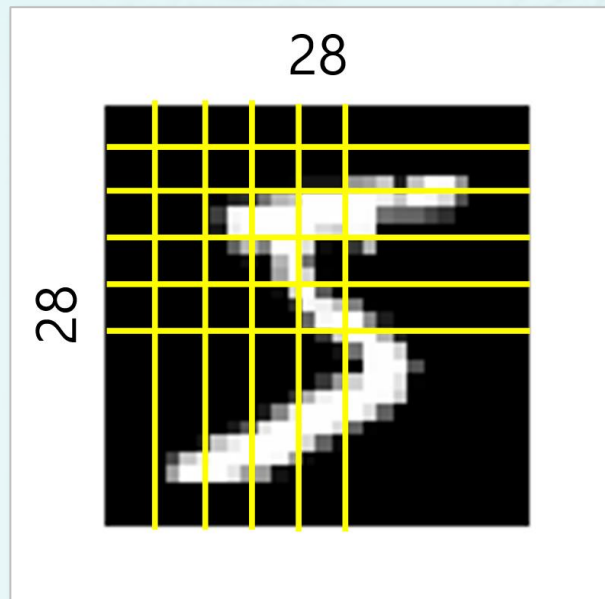
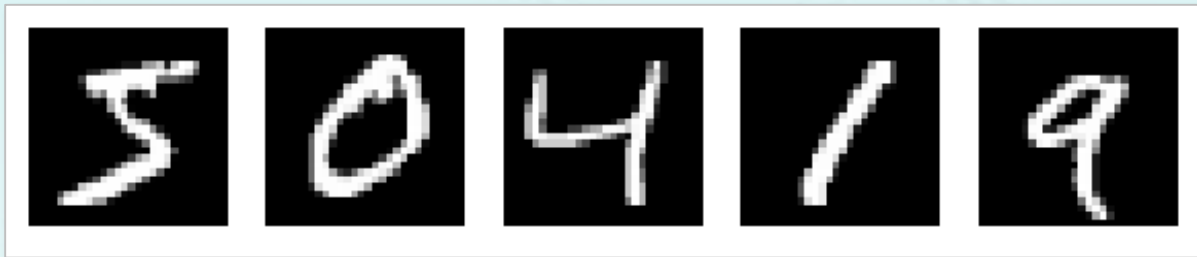


1. 다층 신경망: 신경망 구조

- MNIST

- 28x28

- 입력층 : 784(28x28)

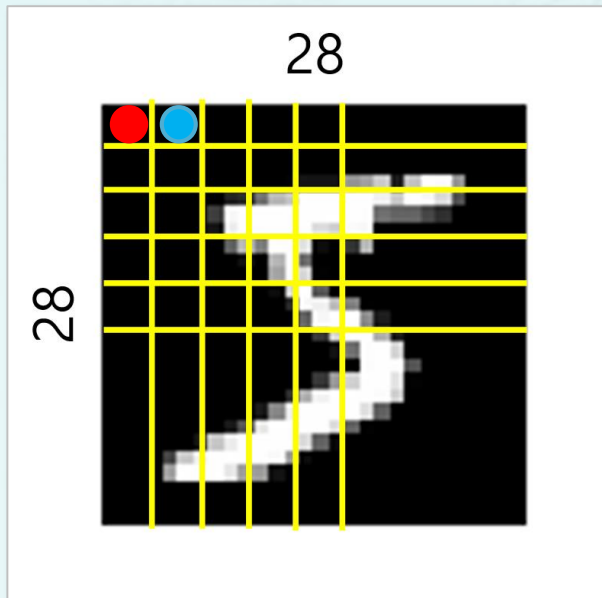


1. 다층 신경망: 신경망 구조

- MNIST

- 28x28

- 입력층 : 784(28x28)



$$\mathbf{X} \in \mathbb{R}^{n \times m}$$

$$\mathbf{X} = \begin{pmatrix} x_1^{(0)} \\ x_2^{(0)} \\ \vdots \\ x_{783}^{(0)} \\ x_{784}^{(0)} \end{pmatrix}$$

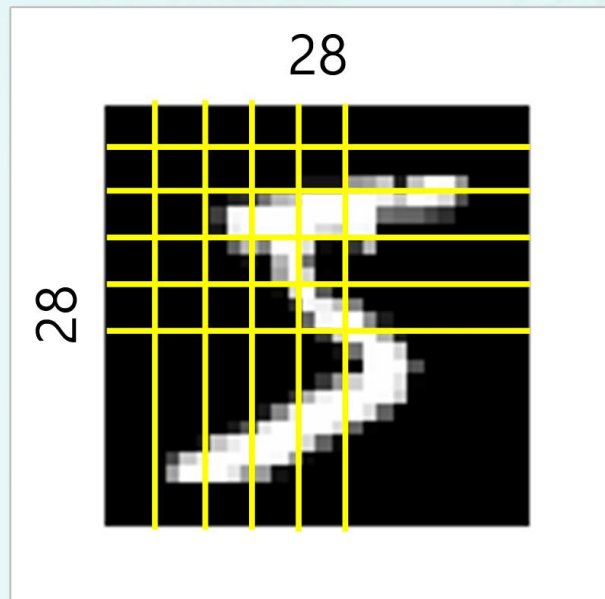
1. 다층 신경망: 신경망 구조

- MNIST

- 28x28

- 입력층 : 784(28x28)

- 은닉층 : 100



1. 다층 신경망: 신경망 구조

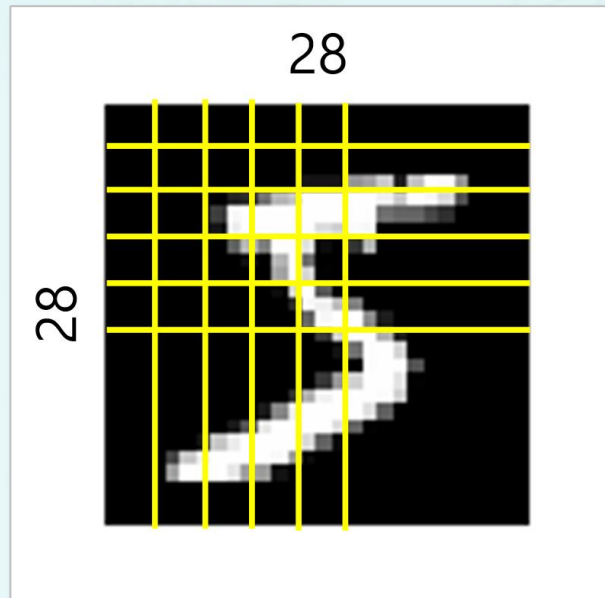
- MNIST

- 28x28

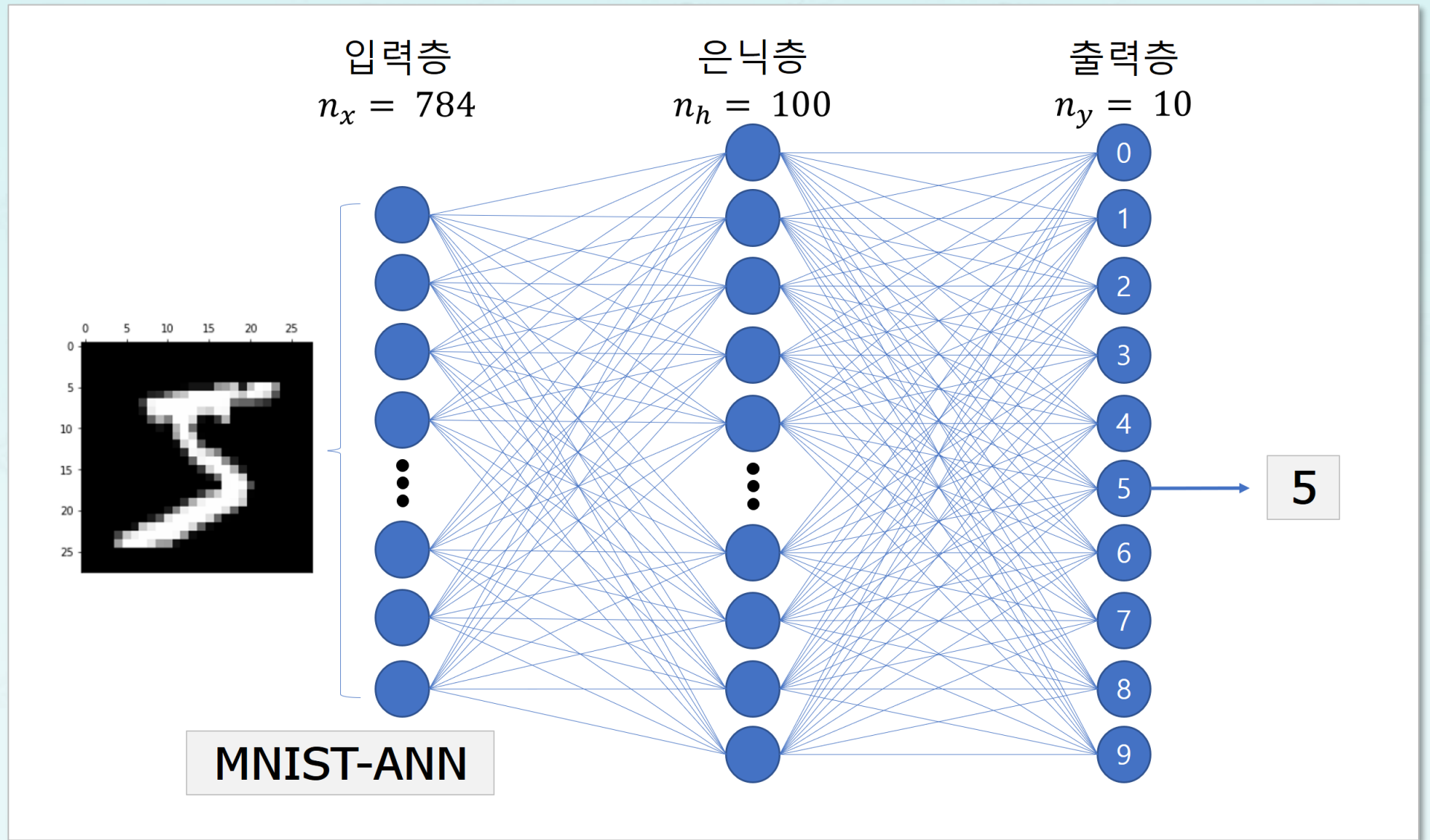
- 입력층 : 784(28x28)

- 은닉층 : 100

- 출력층 : 10

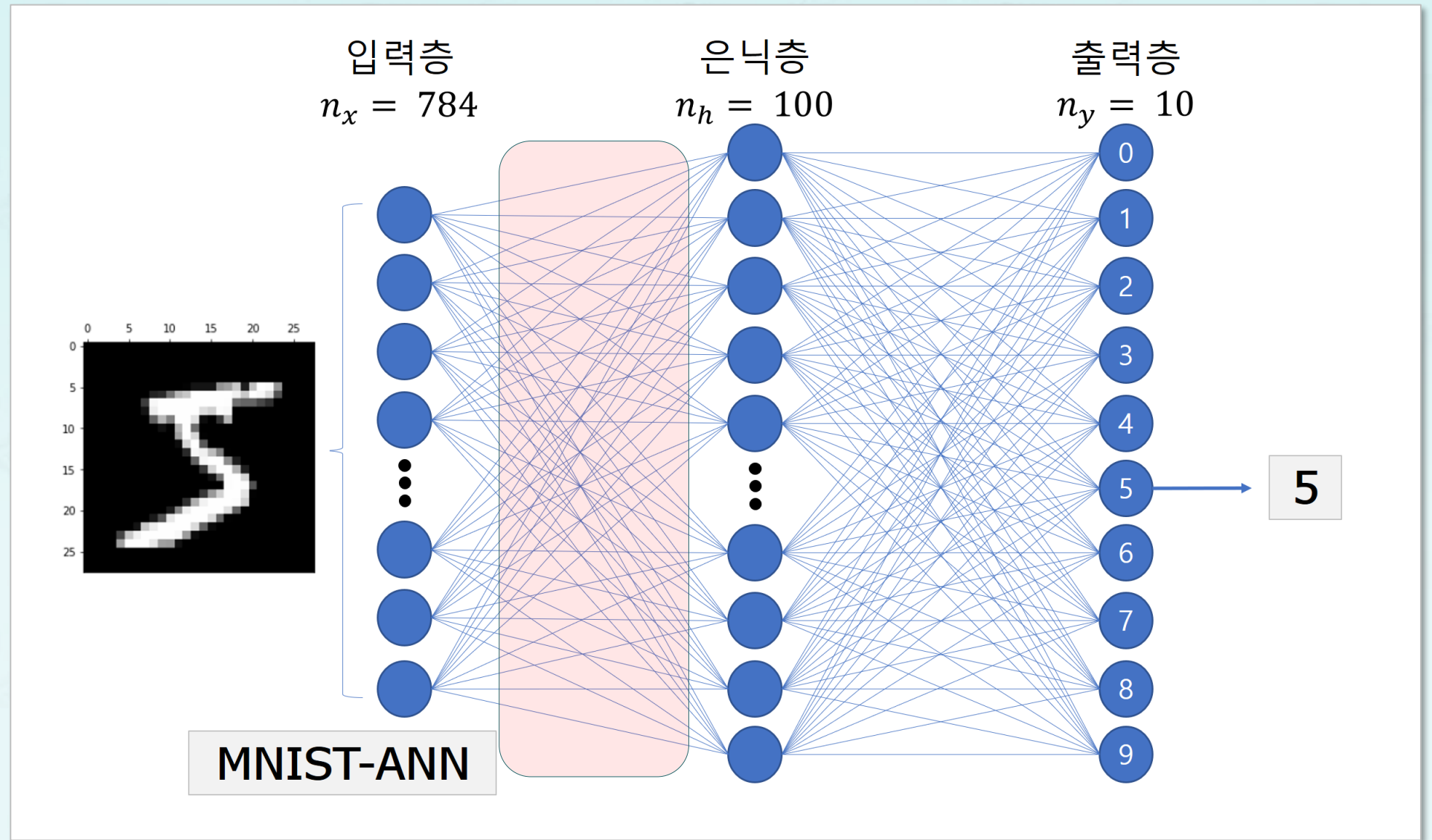


1. 다층 신경망: 신경망 구조



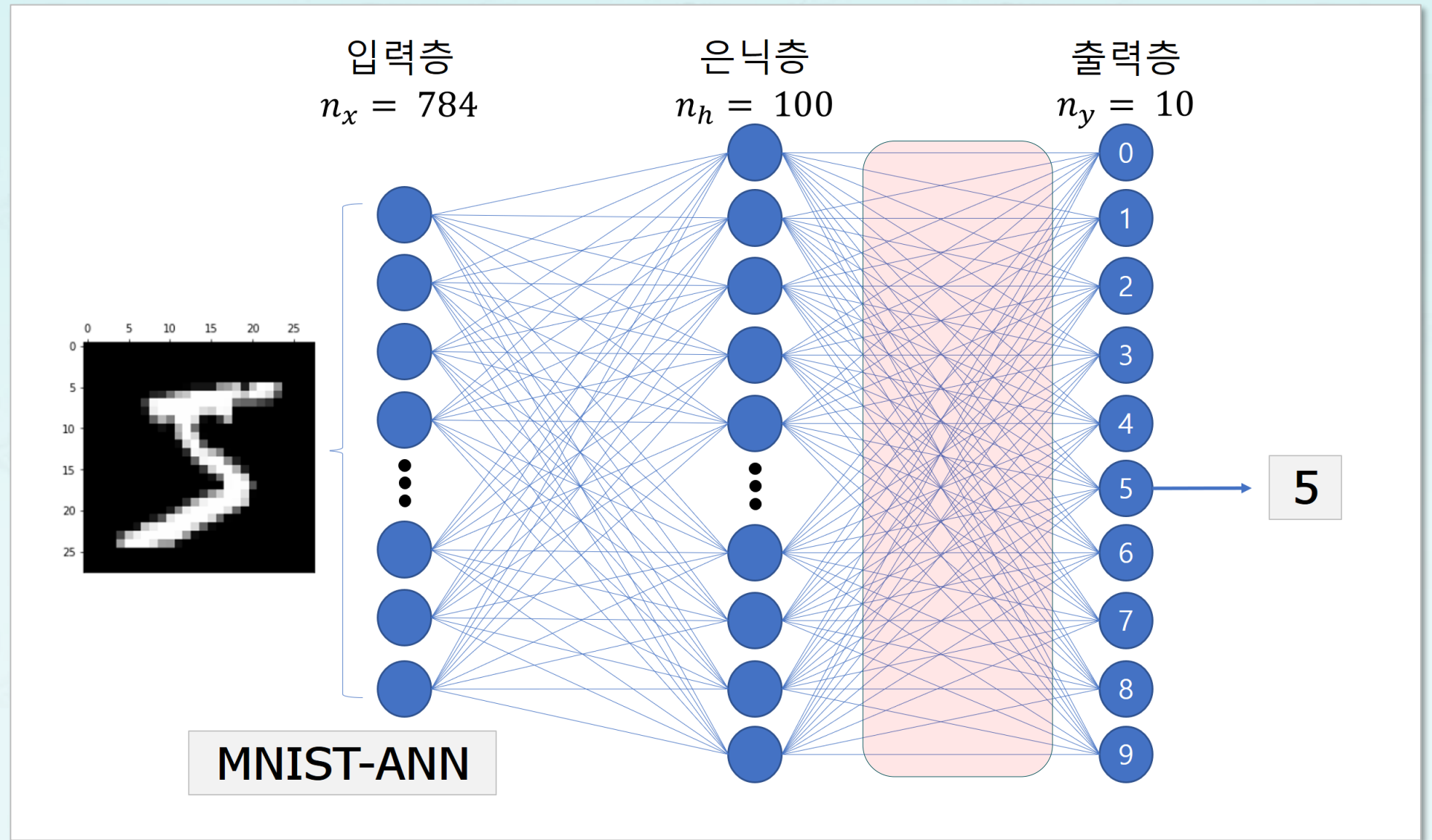
1. 다층 신경망: 신경망 구조

- 가중치
 - 784x100



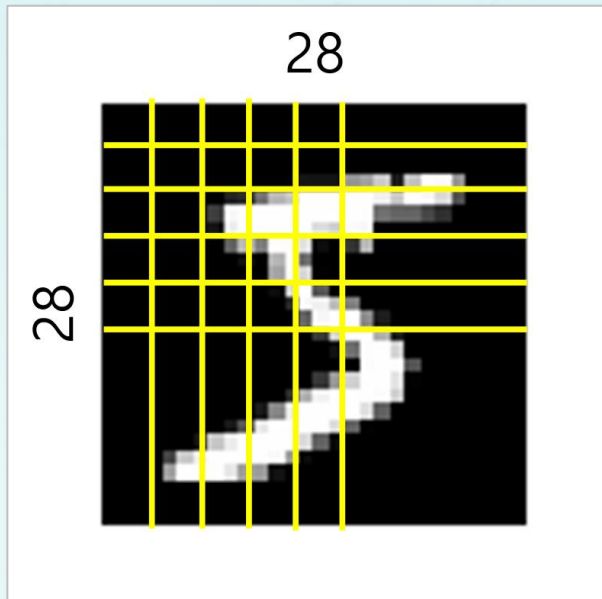
1. 다층 신경망: 신경망 구조

- 가중치
 - 784x100
 - 100x10



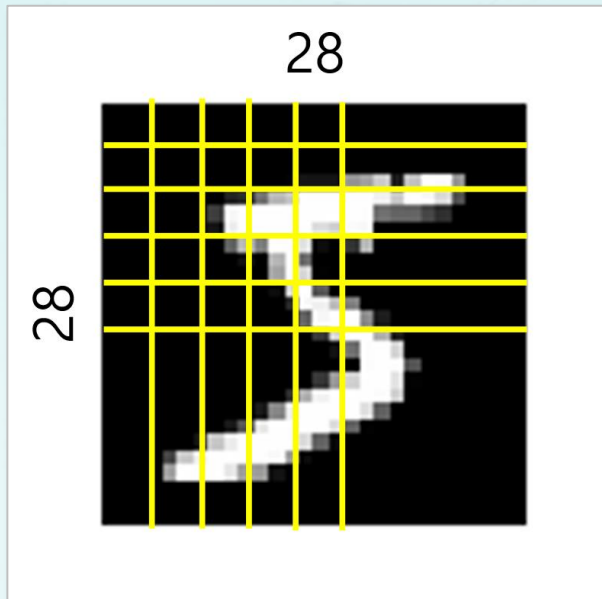
2. 숫자 인식 인공 신경망 구현: 입력 자료

- $X^{n \times m}$
- $n = 784, m = 1$



2. 숫자 인식 인공 신경망 구현: 입력 자료

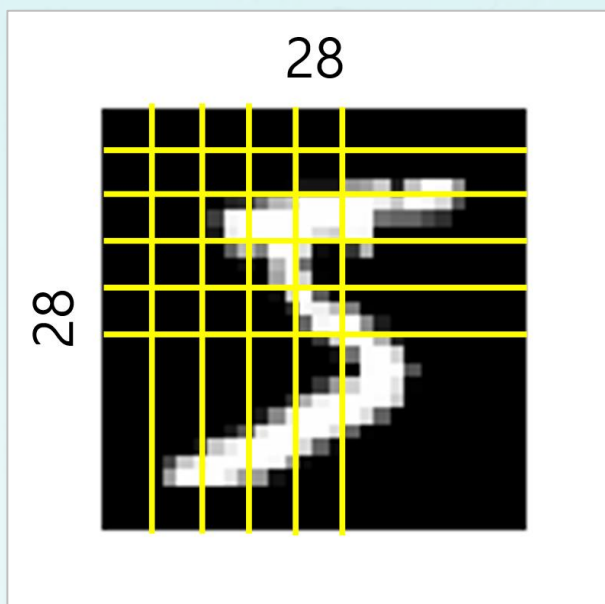
- $X^{n \times m}$
- $n = 784, m = 1$



$$\mathbf{X} \in \mathbb{R}^{n \times m}$$
$$\mathbf{X} = \begin{pmatrix} x_1^{(0)} \\ x_2^{(0)} \\ \vdots \\ x_{783}^{(0)} \\ x_{784}^{(0)} \end{pmatrix}$$

2. 숫자 인식 인공 신경망 구현: 입력 자료

- $X^{n \times m}$
- $n = 784, m = 1$



$$\mathbf{X} \in \mathbb{R}^{n \times m}$$

$$\mathbf{X} = \begin{pmatrix} x_1^{(0)} \\ x_2^{(0)} \\ \vdots \\ x_{783}^{(0)} \\ x_{784}^{(0)} \end{pmatrix}$$

2. 숫자 인식 인공 신경망 구현: 가중치 불러오기

- 사전에 학습된 가중치
- **96%** 성능

2. 숫자 인식 인공 신경망 구현: 가중치 불러오기

- $W^{[1]} = 100 \times 784$

$$W^{[1]} = \begin{pmatrix} w_1^{(1)} & w_1^{(2)} & \cdots & w_1^{(783)} & w_1^{(784)} \\ w_2^{(1)} & w_2^{(2)} & \cdots & w_2^{(783)} & w_2^{(784)} \\ \vdots & \vdots & \vdots & \vdots & \vdots \\ w_{100}^{(1)} & w_{100}^{(2)} & \cdots & w_{100}^{(783)} & w_{100}^{(784)} \end{pmatrix}$$

2. 숫자 인식 인공 신경망 구현: 가중치 불러오기

- $W^{[1]} = 100 \times 784$

$$W^{[1]} = \begin{pmatrix} w_1^{(1)} & w_1^{(2)} & \dots & w_1^{(783)} & w_1^{(784)} \\ w_2^{(1)} & w_2^{(2)} & \dots & w_2^{(783)} & w_2^{(784)} \\ \vdots & \vdots & \vdots & \vdots & \vdots \\ w_{100}^{(1)} & w_{100}^{(2)} & \dots & w_{100}^{(783)} & w_{100}^{(784)} \end{pmatrix}$$

```
In [1]: !cat weights/w_xh.txt
```

```
[-1.65955991e-01  4.40648987e-01  
-7.06488218e-01 -8.15322810e-01  
-2.06465052e-01  7.76334680e-02  
-5.92088802e-01  7.54060585e-01  
-1.65390395e-01  1.17379657e-01
```

2. 숫자 인식 인공 신경망 구현: 은닉층 순입력 계산

2. 숫자 인식 인공 신경망 구현: 은닉층 순입력 계산

$$\begin{aligned}\mathbf{Z}^{[1]} &= \mathbf{W}^{[1]} \mathbf{A}^{[0]} \\ &= \begin{pmatrix} w_{11}^{(1)} & w_{21}^{(1)} \\ w_{12}^{(1)} & w_{22}^{(1)} \\ w_{13}^{(1)} & w_{23}^{(1)} \end{pmatrix} \begin{pmatrix} x_1 \\ x_2 \end{pmatrix} \\ &= \begin{pmatrix} z_1^{(1)} \\ z_2^{(1)} \\ z_3^{(1)} \end{pmatrix}\end{aligned}$$

2. 숫자 인식 인공 신경망 구현: 은닉층 순입력 계산

$$\begin{aligned}\mathbf{Z}^{[1]} &= \mathbf{W}^{[1]} \mathbf{A}^{[0]} \\ &= \begin{pmatrix} w_{11}^{(1)} & w_{21}^{(1)} \\ w_{12}^{(1)} & w_{22}^{(1)} \\ w_{13}^{(1)} & w_{23}^{(1)} \end{pmatrix} \begin{pmatrix} x_1 \\ x_2 \end{pmatrix} \\ &= \begin{pmatrix} z_1^{(1)} \\ z_2^{(1)} \\ z_3^{(1)} \end{pmatrix}\end{aligned}$$

$$\mathbf{W}^{[1]} \mathbf{X}^{[0]} = \begin{pmatrix} w_1^{(1)} & w_1^{(2)} & \cdots & w_1^{(783)} & w_1^{(784)} \\ w_2^{(1)} & w_2^{(2)} & \cdots & w_2^{(783)} & w_2^{(784)} \\ \vdots & \vdots & \vdots & \vdots & \vdots \\ w_{100}^{(1)} & w_{100}^{(2)} & \cdots & w_{100}^{(783)} & w_{100}^{(784)} \end{pmatrix} \begin{pmatrix} x_1^{(0)} \\ x_2^{(0)} \\ \vdots \\ x_{783}^{(0)} \\ x_{784}^{(0)} \end{pmatrix}$$

$$\mathbf{A}^{[1]} = \text{sigmoid}(\mathbf{Z}^{[1]})$$

2. 숫자 인식 인공 신경망 구현: 출력층 계산

- $A^{[1]} : 100 \times 1$

$$W^{[2]} A^{[1]} = \begin{pmatrix} w_1^{(1)} & w_1^{(2)} & \dots & w_1^{(99)} & w_1^{(100)} \\ w_2^{(1)} & w_2^{(2)} & \dots & w_2^{(99)} & w_2^{(100)} \\ \vdots & \vdots & \vdots & \vdots & \\ w_{10}^{(1)} & w_{10}^{(2)} & \dots & w_{10}^{(99)} & w_{10}^{(100)} \end{pmatrix} \begin{pmatrix} a_1^{(1)} \\ a_2^{(1)} \\ \vdots \\ a_{99}^{(1)} \\ a_{100}^{(1)} \end{pmatrix}$$

2. 숫자 인식 인공 신경망 구현: 출력층 계산

- $A^{[1]} : 100 \times 1$
- $W^{[2]} : 10 \times 100$

$$W^{[2]} A^{[1]} = \begin{pmatrix} w_1^{(1)} & w_1^{(2)} & \cdots & w_1^{(99)} & w_1^{(100)} \\ w_2^{(1)} & w_2^{(2)} & \cdots & w_2^{(99)} & w_2^{(100)} \\ \vdots & \vdots & \vdots & \vdots & \\ w_{10}^{(1)} & w_{10}^{(2)} & \cdots & w_{10}^{(99)} & w_{10}^{(100)} \end{pmatrix} \begin{pmatrix} a_1^{(1)} \\ a_2^{(1)} \\ \vdots \\ a_{99}^{(1)} \\ a_{100}^{(1)} \end{pmatrix}$$

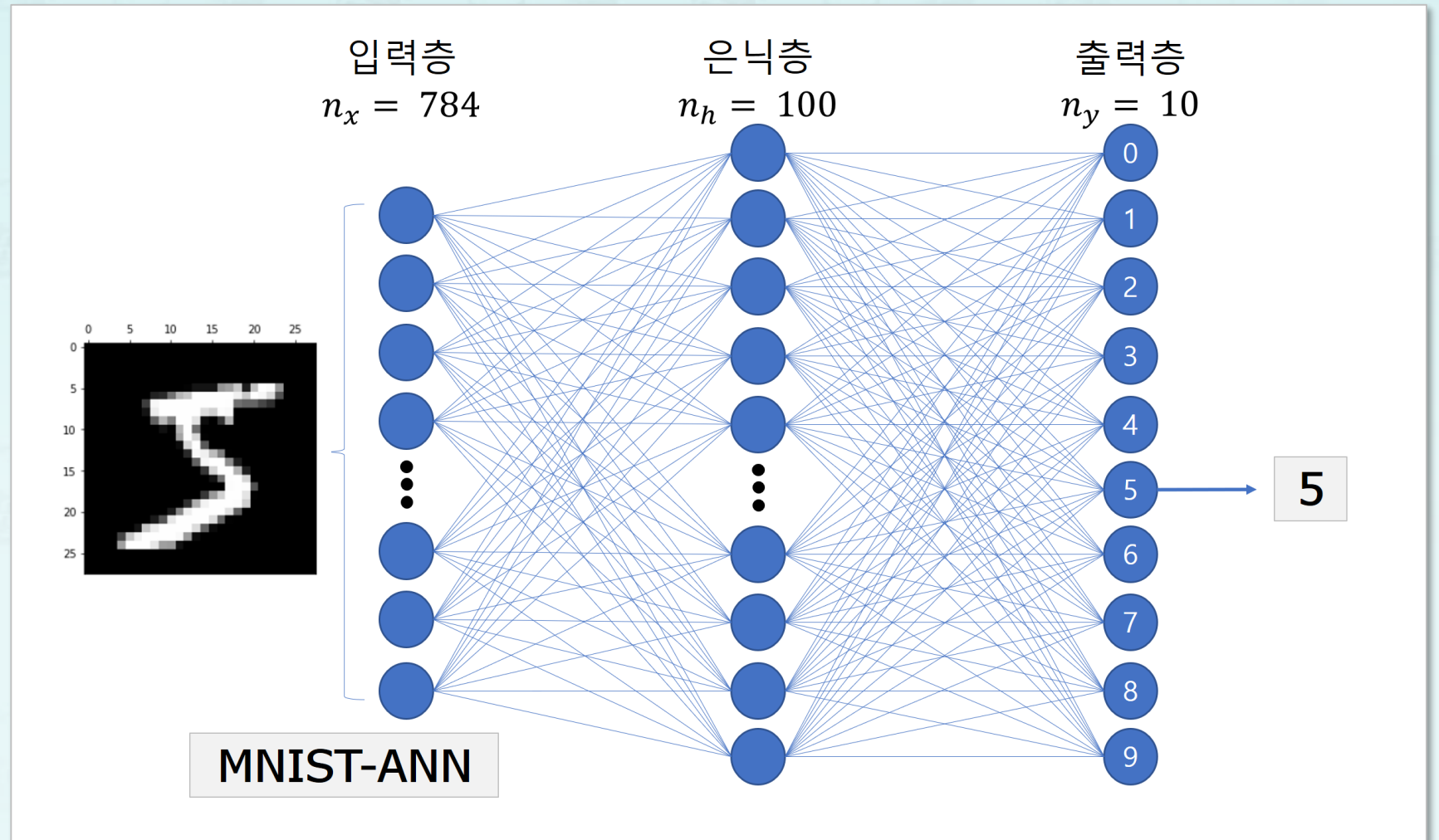
2. 숫자 인식 인공 신경망 구현: 출력층 계산

- $A^{[1]} : 100 \times 1$
- $W^{[2]} : 10 \times 100$

$$W^{[2]}A^{[1]} = \begin{pmatrix} w_1^{(1)} & w_1^{(2)} & \dots & w_1^{(99)} & w_1^{(100)} \\ w_2^{(1)} & w_2^{(2)} & \dots & w_2^{(99)} & w_2^{(100)} \\ \vdots & \vdots & \vdots & \vdots & \vdots \\ w_{10}^{(1)} & w_{10}^{(2)} & \dots & w_{10}^{(99)} & w_{10}^{(100)} \end{pmatrix} \begin{pmatrix} a_1^{(1)} \\ a_2^{(1)} \\ \vdots \\ a_{99}^{(1)} \\ a_{100}^{(1)} \end{pmatrix}$$

$$\mathbf{A}^{[2]} = \text{sigmoid}(\mathbf{Z}^{[2]})$$

2. 숫자 인식 인공 신경망 구현: 인공 신경망 구조



2. 숫자 인식 인공 신경망 구현


```
import joy
import numpy as np
g = lambda x : 1 / (1 + np.exp(-x))
```

```
(X, y) = joy.load_mnist_num(7)
W1 = joy.load_mnist_weight('data/w_xh.weights')
Z1 = np.dot(W1, X)
A1 = g(Z1)

W2 = joy.load_mnist_weight('data/w_hy.weights')
Z2 = np.dot(W2, A1)
yhat = g(Z2)

print('image:', y)
print('predict:', np.round_(yhat, 3))
```

2. 숫자 인식 인공 신경망 구현: 라이브러리 추가




```
import joy
import numpy as np
g = lambda x : 1 / (1 + np.exp(-x))
```

```
(X, y) = joy.load_mnist_num(7)
W1 = joy.load_mnist_weight('data/w_xh.weights')
Z1 = np.dot(W1, X)
A1 = g(Z1)

W2 = joy.load_mnist_weight('data/w_hy.weights')
Z2 = np.dot(W2, A1)
yhat = g(Z2)

print('image:', y)
print('predict:', np.round_(yhat, 3))
```

2. 숫자 인식 인공 신경망 구현: 라이브러리 추가




```
import joy
import numpy as np
g = lambda x : 1 / (1 + np.exp(-x))
```

```
(X, y) = joy.load_mnist_num(7)
W1 = joy.load_mnist_weight('data/w_xh.weights')
Z1 = np.dot(W1, X)
A1 = g(Z1)

W2 = joy.load_mnist_weight('data/w_hy.weights')
Z2 = np.dot(W2, A1)
yhat = g(Z2)

print('image:', y)
print('predict:', np.round_(yhat, 3))
```


2. 숫자 인식 인공 신경망 구현: 함수 생성



```
import joy
import numpy as np
g = lambda x : 1 / (1 + np.exp(-x))
```

```
(X, y) = joy.load_mnist_num(7)
W1 = joy.load_mnist_weight('data/w_xh.weights')
Z1 = np.dot(W1, X)
A1 = g(Z1)


W2 = joy.load_mnist_weight('data/w_hy.weights')
Z2 = np.dot(W2, A1)
yhat = g(Z2)

print('image:', y)
print('predict:', np.round_(yhat, 3))
```


2. 숫자 인식 인공 신경망 구현: 입력 자료 불러오기

■ 입력 자료 불러오기

```
import joy
import numpy as np
g = lambda x : 1 / (1 + np.exp(-x))
```



```
(X, y) = joy.load_mnist_num(7)
W1 = joy.load_mnist_weight('data/w_xh.weights')
Z1 = np.dot(W1, X)
A1 = g(Z1)


W2 = joy.load_mnist_weight('data/w_hy.weights')
Z2 = np.dot(W2, A1)
yhat = g(Z2)

print('image:', y)
print('predict:', np.round_(yhat, 3))
```

2. 숫자 인식 인공 신경망 구현: 입력 자료 불러오기

■ 입력 자료 불러오기

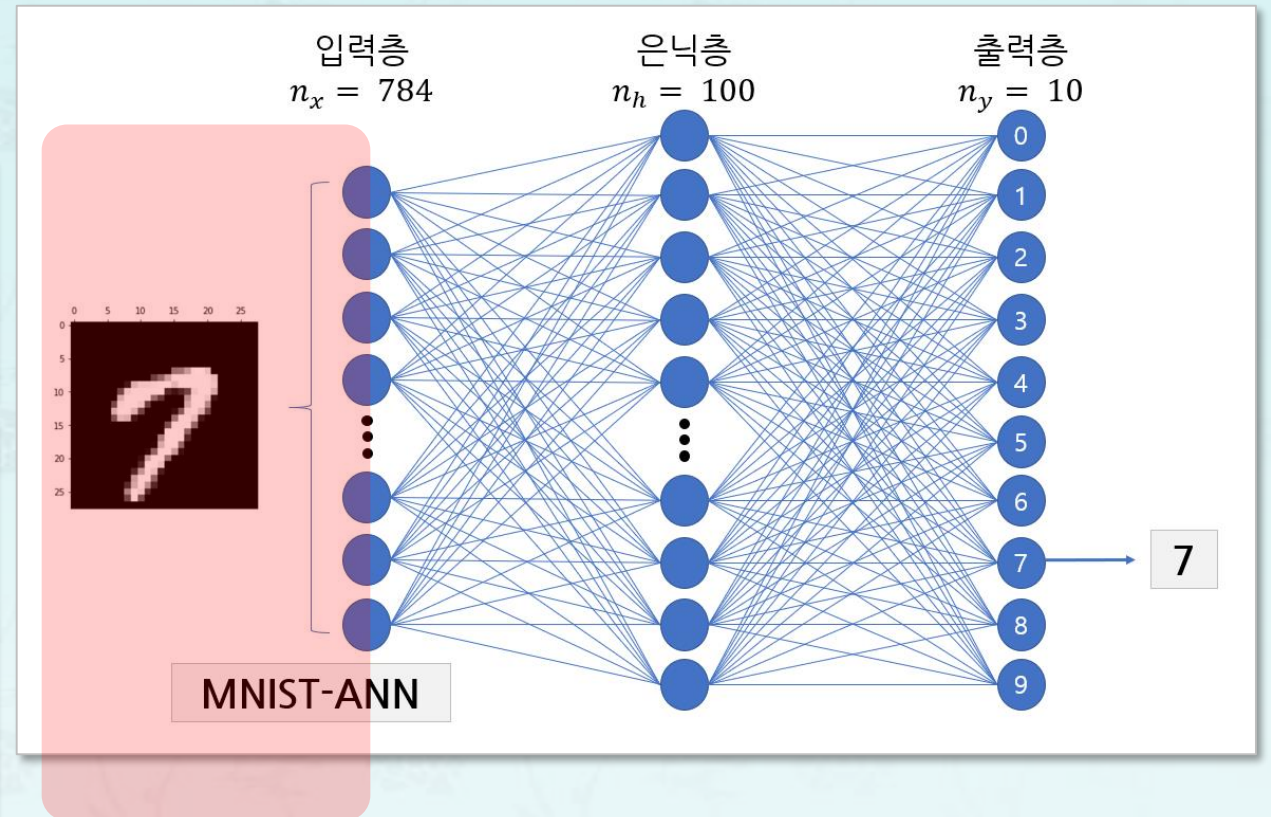
```
import joy
import numpy as np
g = lambda x : 1 / (1 + np.exp(-x))
```



```
(X, y) = joy.load_mnist_num(7)
W1 = joy.load_mnist_weight('data/w_xh.weights')
Z1 = np.dot(W1, X)
A1 = g(Z1)

W2 = joy.load_mnist_weight('data/w_hy.weights')
Z2 = np.dot(W2, A1)
yhat = g(Z2)


print('image:', y)
print('predict:', np.round_(yhat, 3))
```



2. 숫자 인식 인공 신경망 구현: 가중치 불러오기

■ 가중치 불러오기

```
import joy
import numpy as np
g = lambda x : 1 / (1 + np.exp(-x))
```



```
(X, y) = joy.load_mnist_num(7)
W1 = joy.load_mnist_weight('data/w_xh.weights')
Z1 = np.dot(W1, X)
A1 = g(Z1)

W2 = joy.load_mnist_weight('data/w_hy.weights')
Z2 = np.dot(W2, A1)
yhat = g(Z2)


print('image:', y)
print('predict:', np.round_(yhat, 3))
```

$$W^{[1]} = \begin{pmatrix} w_1^{(1)} & w_1^{(2)} & \dots & W_1^{(783)} & w_1^{(784)} \\ w_2^{(1)} & w_2^{(2)} & \dots & W_2^{(783)} & w_2^{(784)} \\ \vdots & \vdots & \vdots & \vdots & \vdots \\ w_{100}^{(1)} & w_{100}^{(2)} & \dots & w_{100}^{(783)} & w_{100}^{(784)} \end{pmatrix}$$

2. 숫자 인식 인공 신경망 구현: 가중치 불러오기

■ 가중치 불러오기

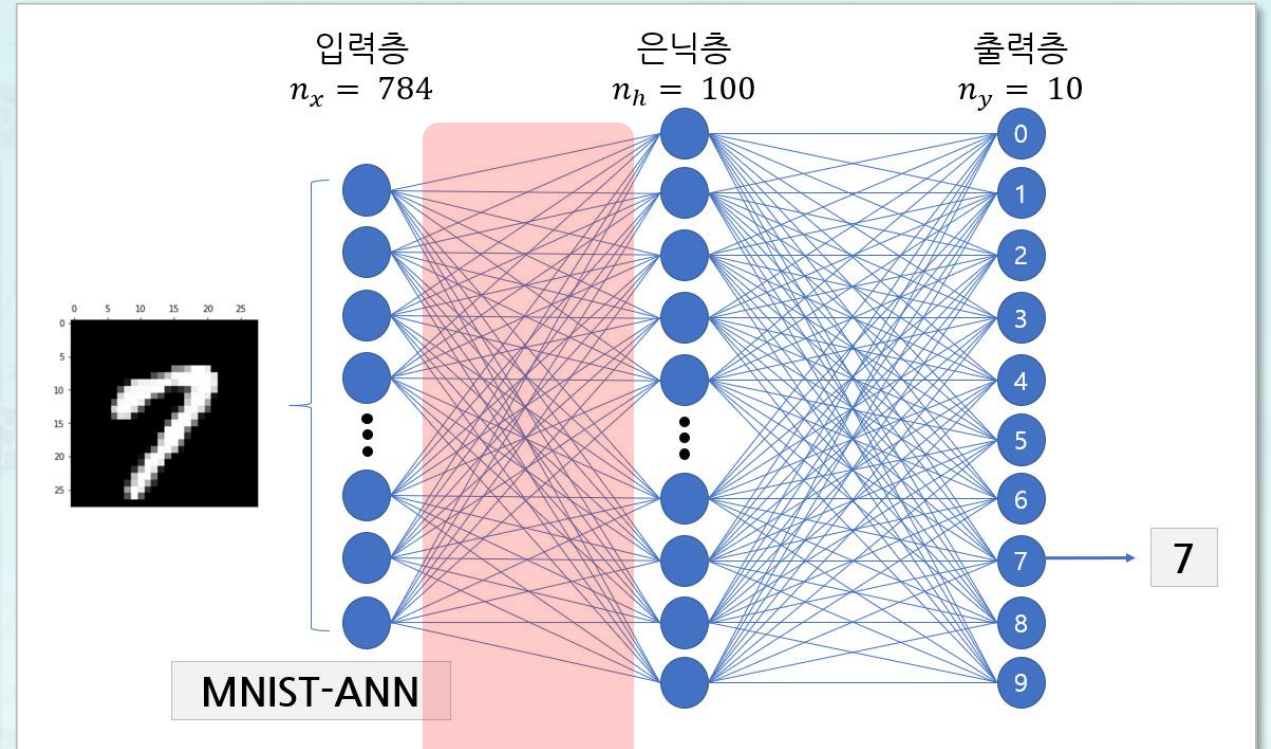
```
import joy
import numpy as np
g = lambda x : 1 / (1 + np.exp(-x))
```



```
(X, y) = joy.load_mnist_num(7)
W1 = joy.load_mnist_weight('data/w_xh.weights')
Z1 = np.dot(W1, X)
A1 = g(Z1)
```

```
W2 = joy.load_mnist_weight('data/w_hy.weights')
Z2 = np.dot(W2, A1)
yhat = g(Z2)
```


```
print('image:', y)
print('predict:', np.round_(yhat, 3))
```



2. 숫자 인식 인공 신경망 구현: 순입력 계산하기

■ 순입력 계산

```
import joy
import numpy as np
g = lambda x : 1 / (1 + np.exp(-x))
```



```
(X, y) = joy.load_mnist_num(7)
W1 = joy.load_mnist_weight('data/w_xh.weights')
Z1 = np.dot(W1, X)
A1 = g(Z1)

W2 = joy.load_mnist_weight('data/w_hy.weights')
Z2 = np.dot(W2, A1)
yhat = g(Z2)


print('image:', y)
print('predict:', np.round_(yhat, 3))
```

$$W^{[1]} = \begin{pmatrix} w_1^{(1)} & w_1^{(2)} & \dots & W_1^{(783)} & w_1^{(784)} \\ w_2^{(1)} & w_2^{(2)} & \dots & W_2^{(783)} & w_2^{(784)} \\ \vdots & \vdots & \vdots & \vdots & \vdots \\ w_{100}^{(1)} & w_{100}^{(2)} & \dots & w_{100}^{(783)} & w_{100}^{(784)} \end{pmatrix}$$

2. 숫자 인식 인공 신경망 구현: 순입력 계산하기

■ 순입력 계산

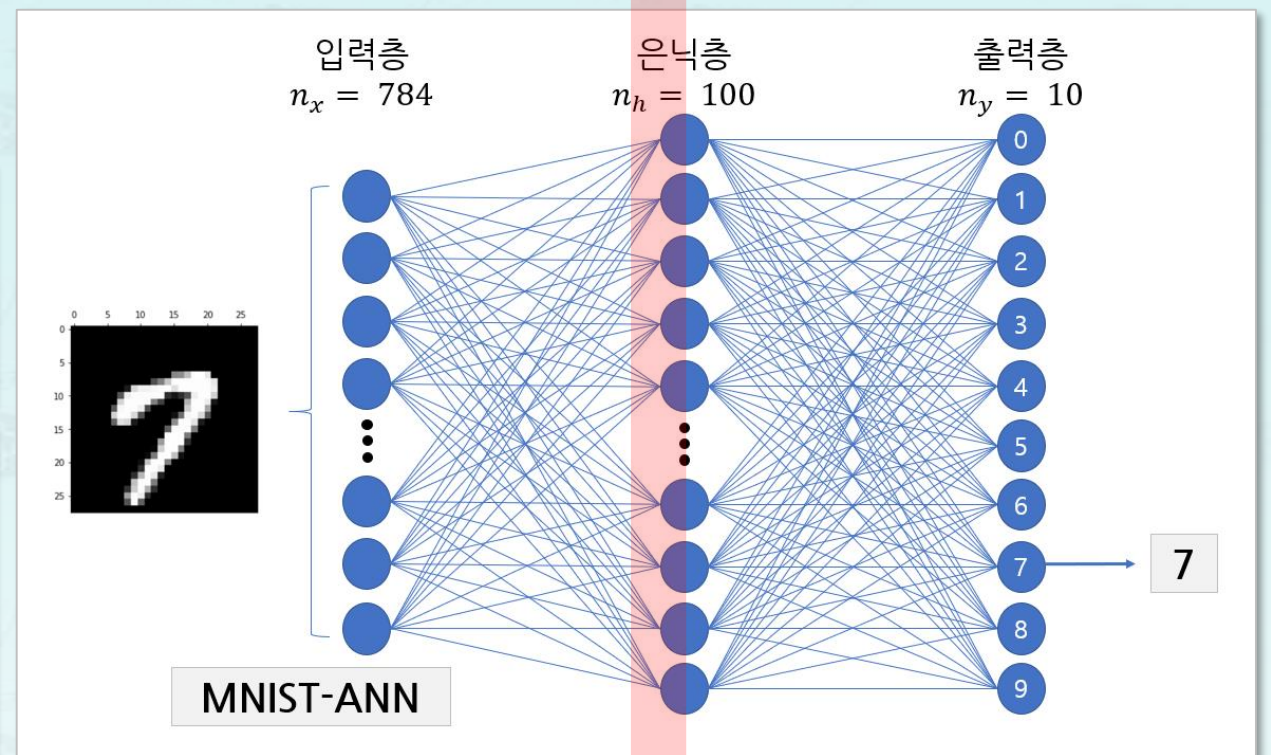
```
import joy
import numpy as np
g = lambda x : 1 / (1 + np.exp(-x))
```



```
(X, y) = joy.load_mnist_num(7)
W1 = joy.load_mnist_weight('data/w_xh.weights')
Z1 = np.dot(W1, X)
A1 = g(Z1)
```

```
W2 = joy.load_mnist_weight('data/w_hy.weights')
Z2 = np.dot(W2, A1)
yhat = g(Z2)
```


```
print('image:', y)
print('predict:', np.round_(yhat, 3))
```



2. 숫자 인식 인공 신경망 구현: 은닉층 계산하기

■ 은닉층 계산

```
import joy
import numpy as np
g = lambda x : 1 / (1 + np.exp(-x))
```



```
(X, y) = joy.load_mnist_num(7)
W1 = joy.load_mnist_weight('data/w_xh.weights')
Z1 = np.dot(W1, X)
A1 = g(Z1)

W2 = joy.load_mnist_weight('data/w_hy.weights')
Z2 = np.dot(W2, A1)
yhat = g(Z2)

print('image:', y)
print('predict:', np.round_(yhat, 3))
```

$$W^{[1]} = \begin{pmatrix} w_1^{(1)} & w_1^{(2)} & \dots & W_1^{(783)} & w_1^{(784)} \\ w_2^{(1)} & w_2^{(2)} & \dots & W_2^{(783)} & w_2^{(784)} \\ \vdots & \vdots & \vdots & \vdots & \vdots \\ w_{100}^{(1)} & w_{100}^{(2)} & \dots & w_{100}^{(783)} & w_{100}^{(784)} \end{pmatrix}$$

$$\mathbf{A}^{[1]} = \text{sigmoid}(\mathbf{Z}^{[1]})$$

2. 숫자 인식 인공 신경망 구현: 은닉층 계산하기

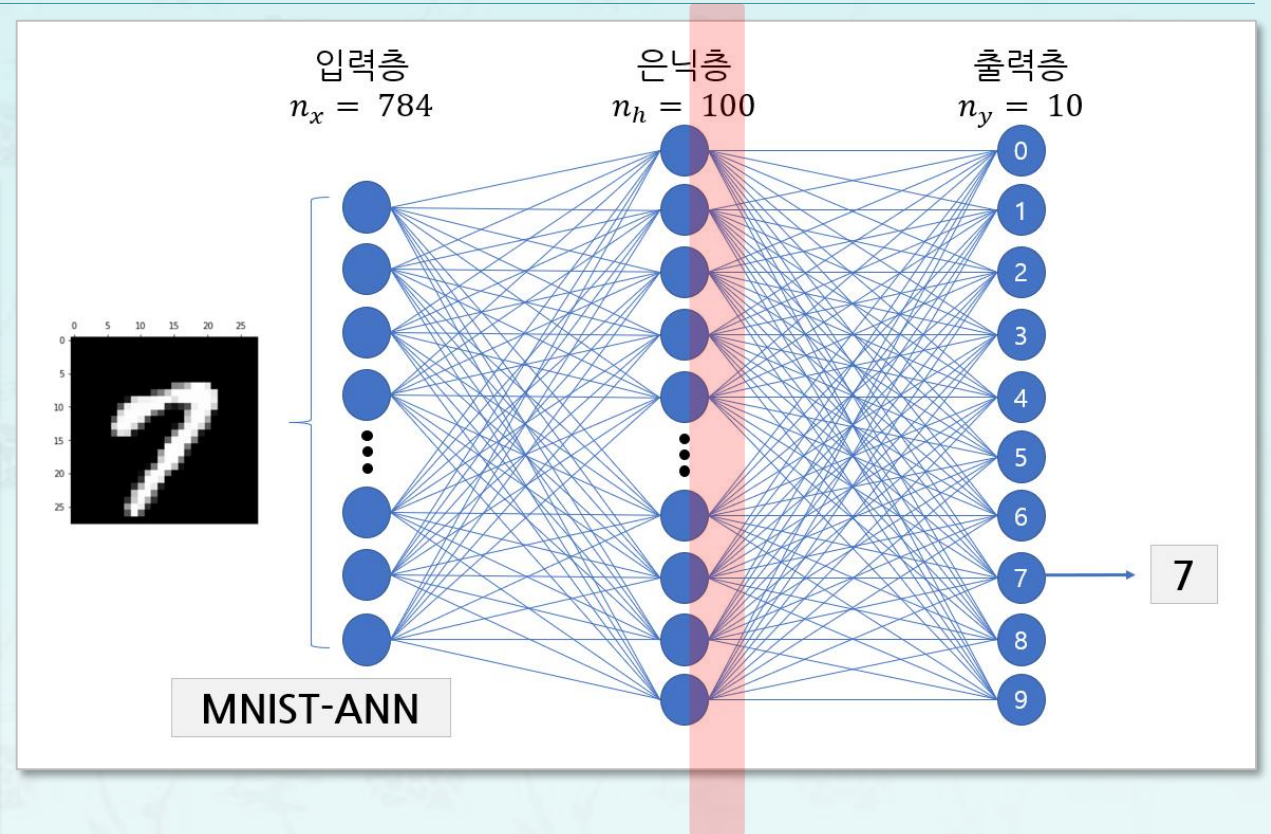
■ 은닉층 계산

```
import joy
import numpy as np
g = lambda x : 1 / (1 + np.exp(-x))
```

```
(X, y) = joy.load_mnist_num(7)
W1 = joy.load_mnist_weight('data/w_xh.weights')
Z1 = np.dot(W1, X)
A1 = g(Z1)
```

```
W2 = joy.load_mnist_weight('data/w_hy.weights')
Z2 = np.dot(W2, A1)
yhat = g(Z2)
```


```
print('image:', y)
print('predict:', np.round_(yhat, 3))
```



2. 숫자 인식 인공 신경망 구현: 가중치 불러오기

```
import joy
import numpy as np
g = lambda x : 1 / (1 + np.exp(-x))
```

```
(X, y) = joy.load_mnist_num(7)
W1 = joy.load_mnist_weight('data/w_xh.weights')
Z1 = np.dot(W1, X)
A1 = g(Z1)
```



```
W2 = joy.load_mnist_weight('data/w_hy.weights')
Z2 = np.dot(W2, A1)
yhat = g(Z2)
```

```
print('image:', y)
print('predict:', np.round_(yhat, 3))
```

$$W^{[2]} = \begin{pmatrix} w_1^{(1)} & w_1^{(2)} & \dots & W_1^{(99)} & w_1^{(100)} \\ w_2^{(1)} & w_2^{(2)} & \dots & W_2^{(99)} & w_2^{(100)} \\ \vdots & \vdots & \vdots & \vdots & \vdots \\ w_{100}^{(1)} & w_{100}^{(2)} & \dots & w_{100}^{(99)} & w_{100}^{(100)} \end{pmatrix}$$

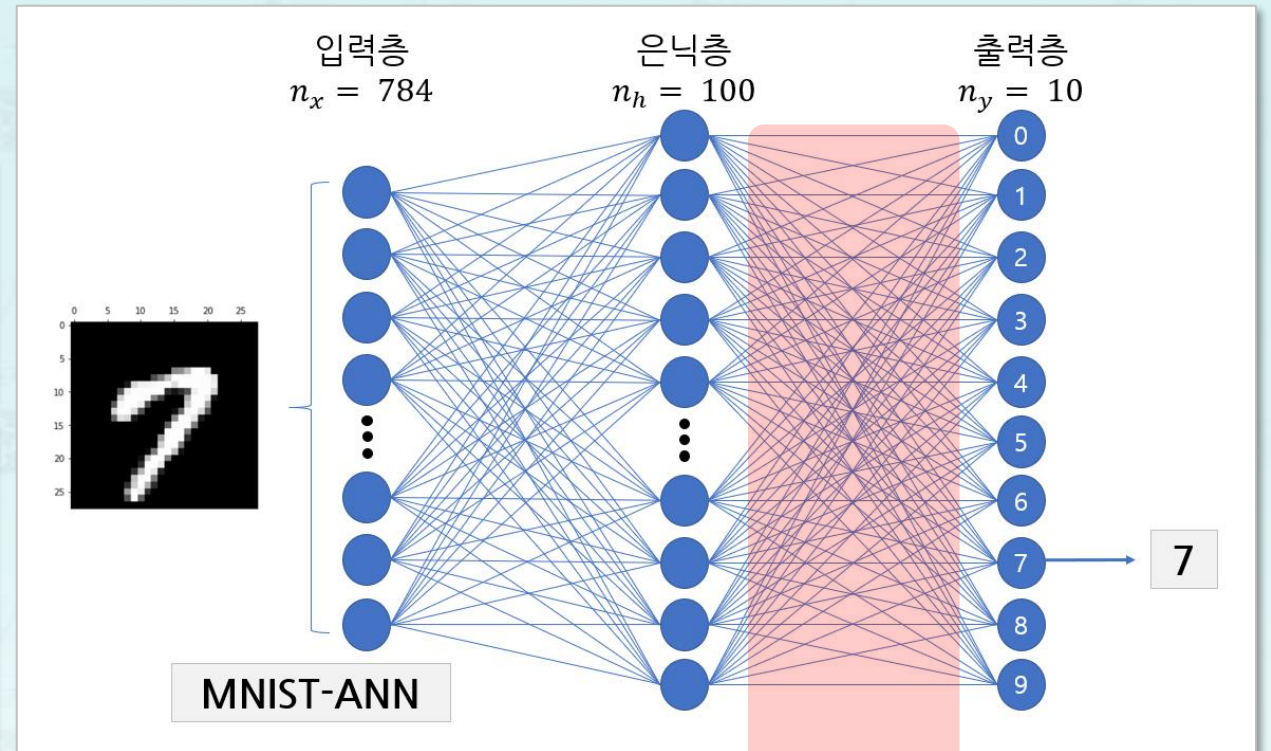
2. 숫자 인식 인공 신경망 구현

```
import joy
import numpy as np
g = lambda x : 1 / (1 + np.exp(-x))
```

```
(X, y) = joy.load_mnist_num(7)
W1 = joy.load_mnist_weight('data/w_xh.weights')
Z1 = np.dot(W1, X)
A1 = g(Z1)
```


```
W2 = joy.load_mnist_weight('data/w_hy.weights')
Z2 = np.dot(W2, A1)
yhat = g(Z2)
```

```
print('image:', y)
print('predict:', np.round_(yhat, 3))
```



2. 숫자 인식 인공 신경망 구현: 은닉층 계산

```
import joy
import numpy as np
g = lambda x : 1 / (1 + np.exp(-x))
```



```
(X, y) = joy.load_mnist_num(7)
W1 = joy.load_mnist_weight('data/w_xh.weights')
Z1 = np.dot(W1, X)
A1 = g(Z1)

W2 = joy.load_mnist_weight('data/w_hy.weights')
Z2 = np.dot(W2, A1)
yhat = g(Z2)

print('image:', y)
print('predict:', np.round_(yhat, 3))
```

$$W^{[2]}A^{[1]} = \begin{pmatrix} w_1^{(1)} & w_1^{(2)} & \cdots & w_1^{(99)} & w_1^{(100)} \\ w_2^{(1)} & w_2^{(2)} & \cdots & w_2^{(99)} & w_2^{(100)} \\ \vdots & \vdots & \vdots & \vdots & \vdots \\ w_{10}^{(1)} & w_{10}^{(2)} & \cdots & w_{10}^{(99)} & w_{10}^{(100)} \end{pmatrix} \begin{pmatrix} a_1^{(1)} \\ a_2^{(1)} \\ \vdots \\ a_{99}^{(1)} \\ a_{100}^{(1)} \end{pmatrix}$$

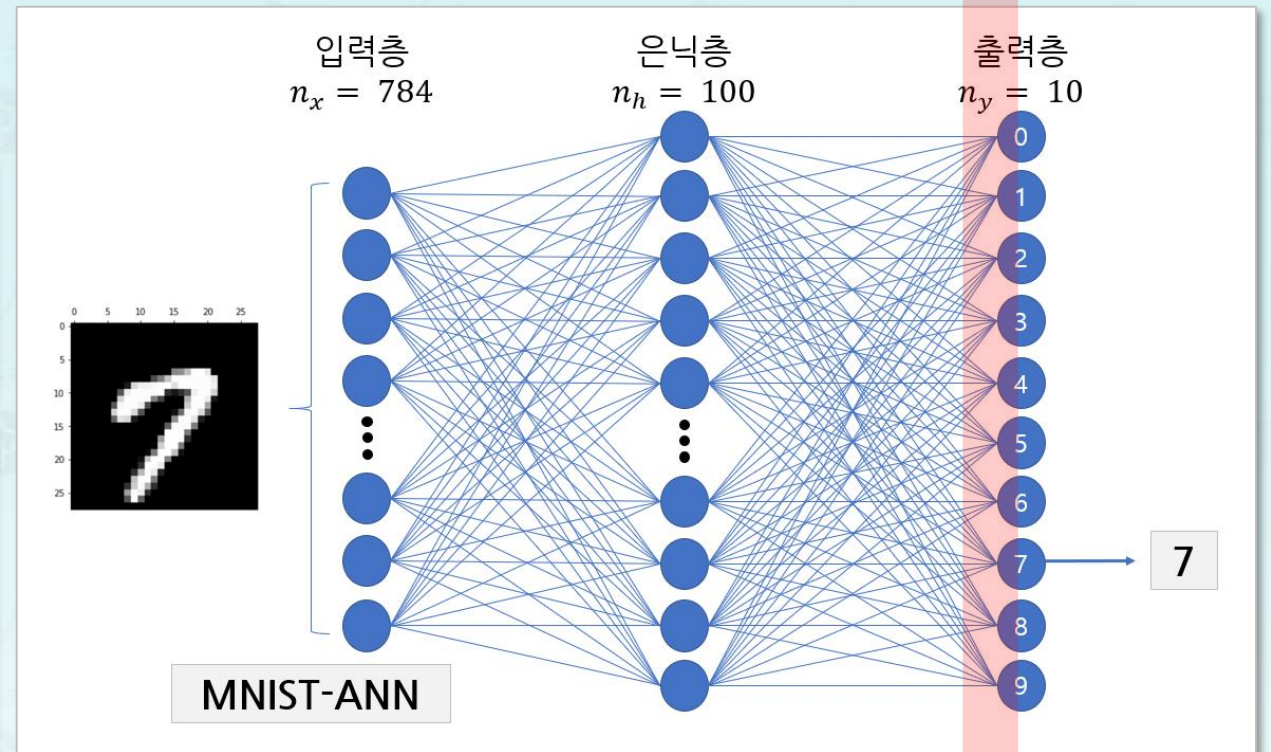
2. 숫자 인식 인공 신경망 구현: 은닉층 계산하기

```
import joy
import numpy as np
g = lambda x : 1 / (1 + np.exp(-x))
```

```
(X, y) = joy.load_mnist_num(7)
W1 = joy.load_mnist_weight('data/w_xh.weights')
Z1 = np.dot(W1, X)
A1 = g(Z1)
```

```
W2 = joy.load_mnist_weight('data/w_hy.weights')
Z2 = np.dot(W2, A1)
yhat = g(Z2)
```

```
print('image:', y)
print('predict:', np.round_(yhat, 3))
```



2. 숫자 인식 인공 신경망 구현: 출력층 계산하기

```
import joy
import numpy as np
g = lambda x : 1 / (1 + np.exp(-x))
```

```
(X, y) = joy.load_mnist_num(7)
W1 = joy.load_mnist_weight('data/w_xh.weights')
Z1 = np.dot(W1, X)
A1 = g(Z1)

W2 = joy.load_mnist_weight('data/w_hy.weights')
Z2 = np.dot(W2, A1)
yhat = g(Z2)

print('image:', y)
print('predict:', np.round_(yhat, 3))
```

$$W^{[2]}A^{[1]} = \begin{pmatrix} w_1^{(1)} & w_1^{(2)} & \cdots & w_1^{(99)} & w_1^{(100)} \\ w_2^{(1)} & w_2^{(2)} & \cdots & w_2^{(99)} & w_2^{(100)} \\ \vdots & \vdots & \vdots & \vdots & \vdots \\ w_{10}^{(1)} & w_{10}^{(2)} & \cdots & w_{10}^{(99)} & w_{10}^{(100)} \end{pmatrix} \begin{pmatrix} a_1^{(1)} \\ a_2^{(1)} \\ \vdots \\ a_{99}^{(1)} \\ a_{100}^{(1)} \end{pmatrix}$$

$$\mathbf{A}^{[2]} = \text{sigmoid}(\mathbf{Z}^{[2]})$$

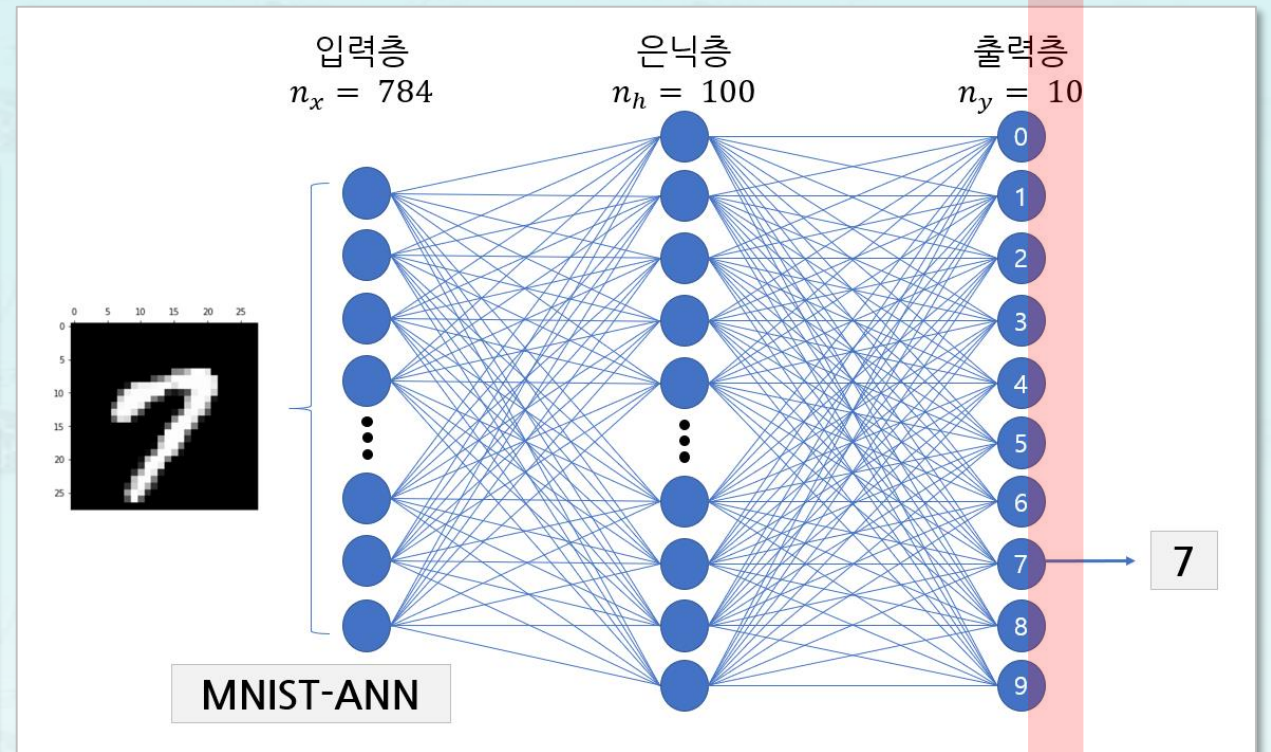
2. 숫자 인식 인공 신경망 구현: 출력층 계산하기

```
import joy
import numpy as np
g = lambda x : 1 / (1 + np.exp(-x))
```

```
(X, y) = joy.load_mnist_num(7)
W1 = joy.load_mnist_weight('data/w_xh.weights')
Z1 = np.dot(W1, X)
A1 = g(Z1)
```

```
W2 = joy.load_mnist_weight('data/w_hy.weights')
Z2 = np.dot(W2, A1)
yhat = g(Z2)
```

```
print('image:', y)
print('predict:', np.round_(yhat, 3))
```




2. 숫자 인식 인공 신경망 구현: 예측하기

```
import joy
import numpy as np
g = lambda x : 1 / (1 + np.exp(-x))
```

```
(X, y) = joy.load_mnist_num(7)
W1 = joy.load_mnist_weight('data/w_xh.weights')
Z1 = np.dot(W1, X)
A1 = g(Z1)

W2 = joy.load_mnist_weight('data/w_hy.weights')
Z2 = np.dot(W2, A1)
yhat = g(Z2)

print('image:', y)
print('predict:', np.round_(yhat, 3))
```



```
image: 7
predict: [0.
          0.002
          0.001
          0.
          0.
          0.001
          0.
          0.979
          0.006
          0.003]
```


2. 숫자 인식 인공 신경망 구현: 예측하기

```
import joy
import numpy as np
g = lambda x : 1 / (1 + np.exp(-x))
```

```
(X, y) = joy.load_mnist_num(7)
W1 = joy.load_mnist_weight('data/w_xh.weights')
Z1 = np.dot(W1, X)
A1 = g(Z1)

W2 = joy.load_mnist_weight('data/w_hy.weights')
Z2 = np.dot(W2, A1)
yhat = g(Z2)

print('image:', y)
print('predict:', np.round_(yhat, 3))
```

```
image: 7
predict: [0.
          0.002
          0.001
          0.
          0.
          0.001
          0.
          0.979
          0.006
          0.003]
```

2. 숫자 인식 인공 신경망 구현: 예측하기

```
import joy
import numpy as np
g = lambda x : 1 / (1 + np.exp(-x))
```

```
(X, y) = joy.load_mnist_num(7)
W1 = joy.load_mnist_weight('data/w_xh.weights')
Z1 = np.dot(W1, X)
A1 = g(Z1)

W2 = joy.load_mnist_weight('data/w_hy.weights')
Z2 = np.dot(W2, A1)
yhat = g(Z2)

print('image:', y)
print('predict:', np.round_(yhat, 3))
```

image: 7

predict: [0.
0.002
0.001
0.
0.
0.001
0.
0.979
0.006
0.003]

0
1
2
3
4
5
6
7
8
9

순방향 신경망 예제

- 학습 정리
 - 예제를 통해 순방향 신경망을 깊이 있게 이해하기
 - 자료의 특성 이해하기
 - 순방향 신경망 설계
 - 가중치 불러오기
- **7-3** 아달라인 경사하강법 소개

7주차(2/3)

순방향 신경망 예제

파이썬으로 배우는 기계학습

한동대학교
김영섭 교수

여러분 곁에 항상 열려 있는 K-MOOC 강의실에서 만나 뵙기를 바랍니다.