7주차(2/3)

순방향 신경망 예제

파이썬으로 배우는 기계학습

한동대학교
김영섭 교수

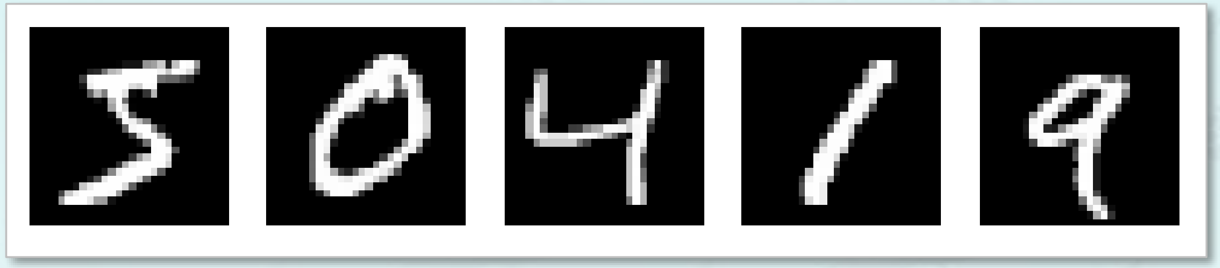
순방향 신경망 예제

- 학습 목표
  - 예제를 통해 순방향 신경망을 깊이 있게 이해한다.

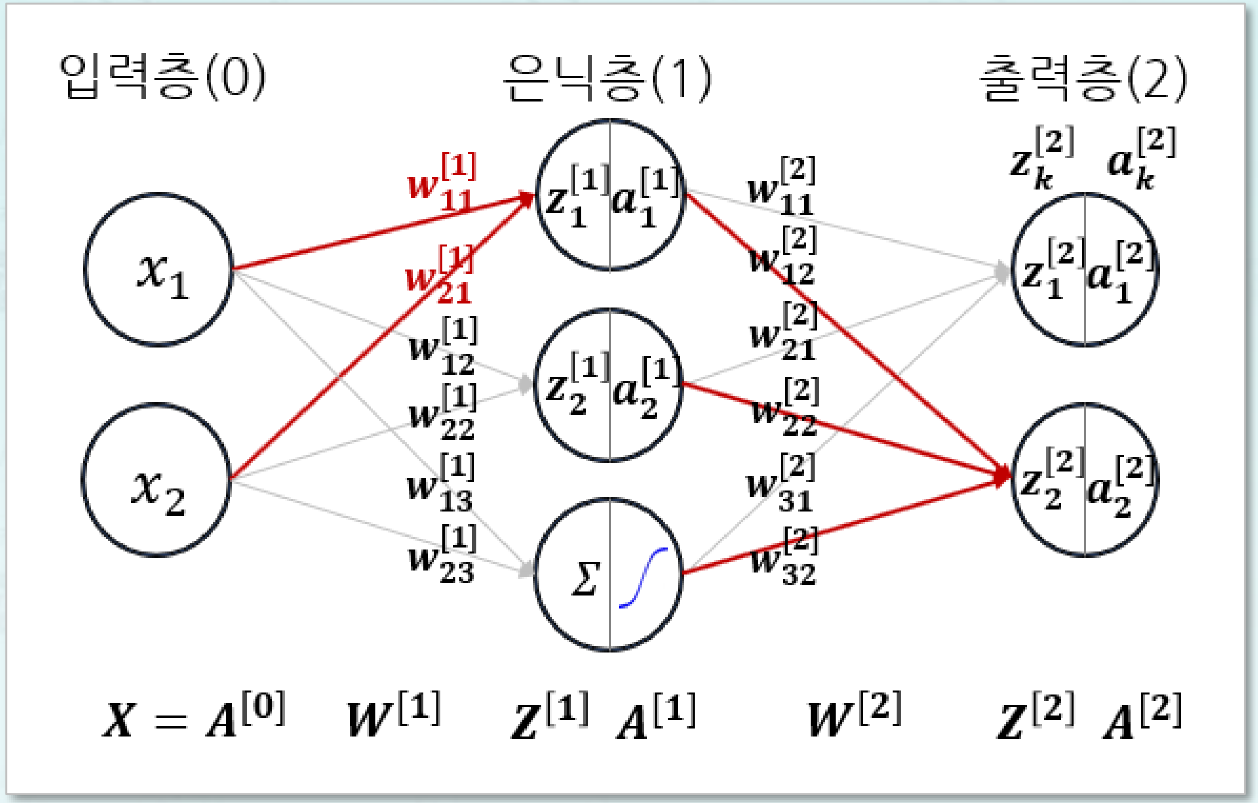
- 학습 내용
  - **MNIST** 자료 이해
  - 다층 신경망 설계
  - 순방향 신경망 신호처리
  - 순방향 신경망 예제 구현

1. 인공 신경망: 입력자료

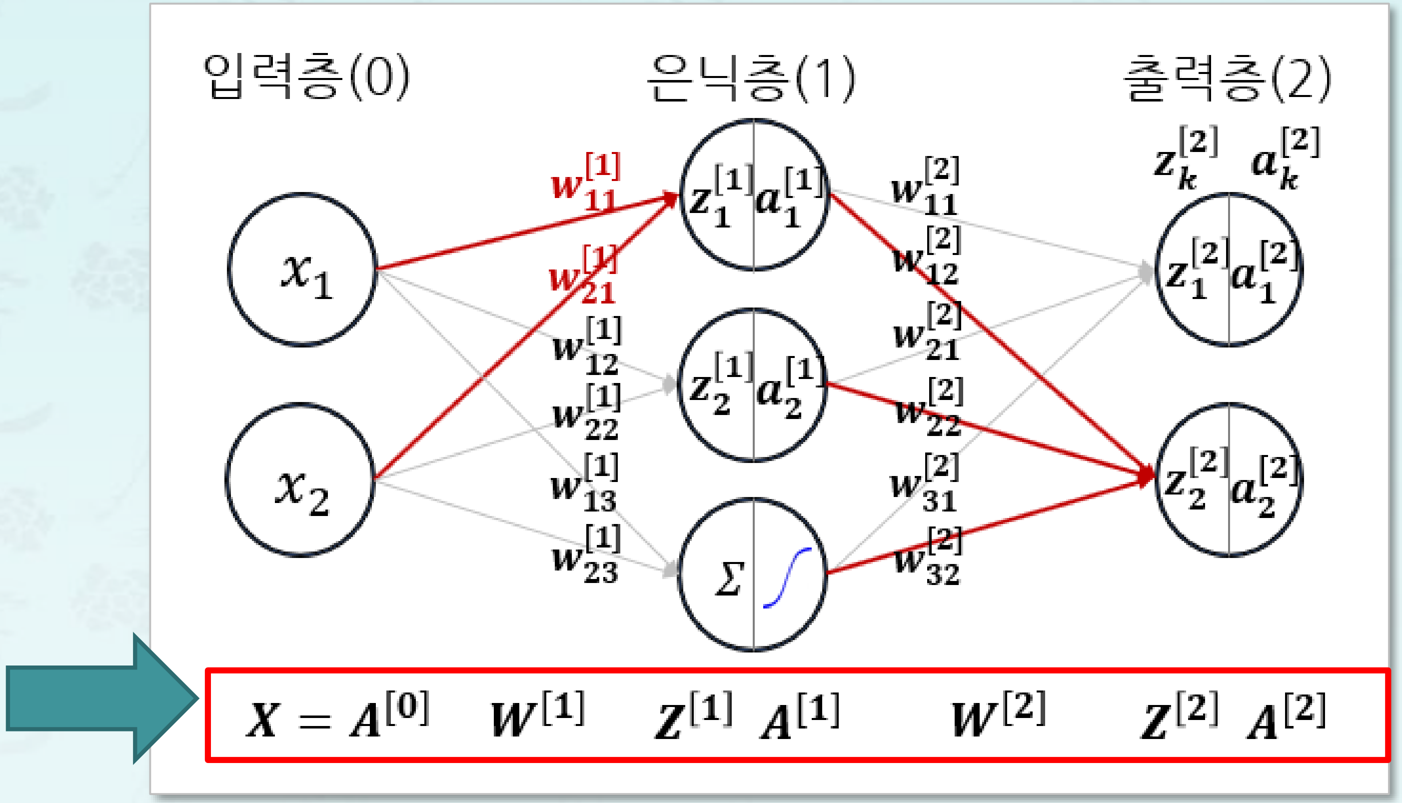
- **MNIST**



1. 인공 신경망: 간단한 신경망(복습)

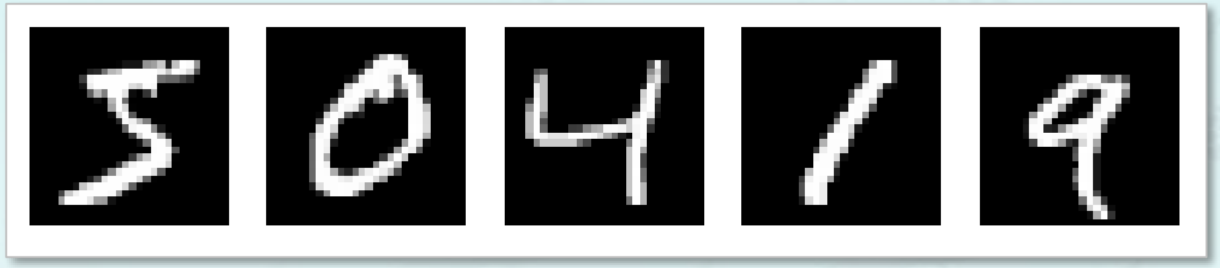


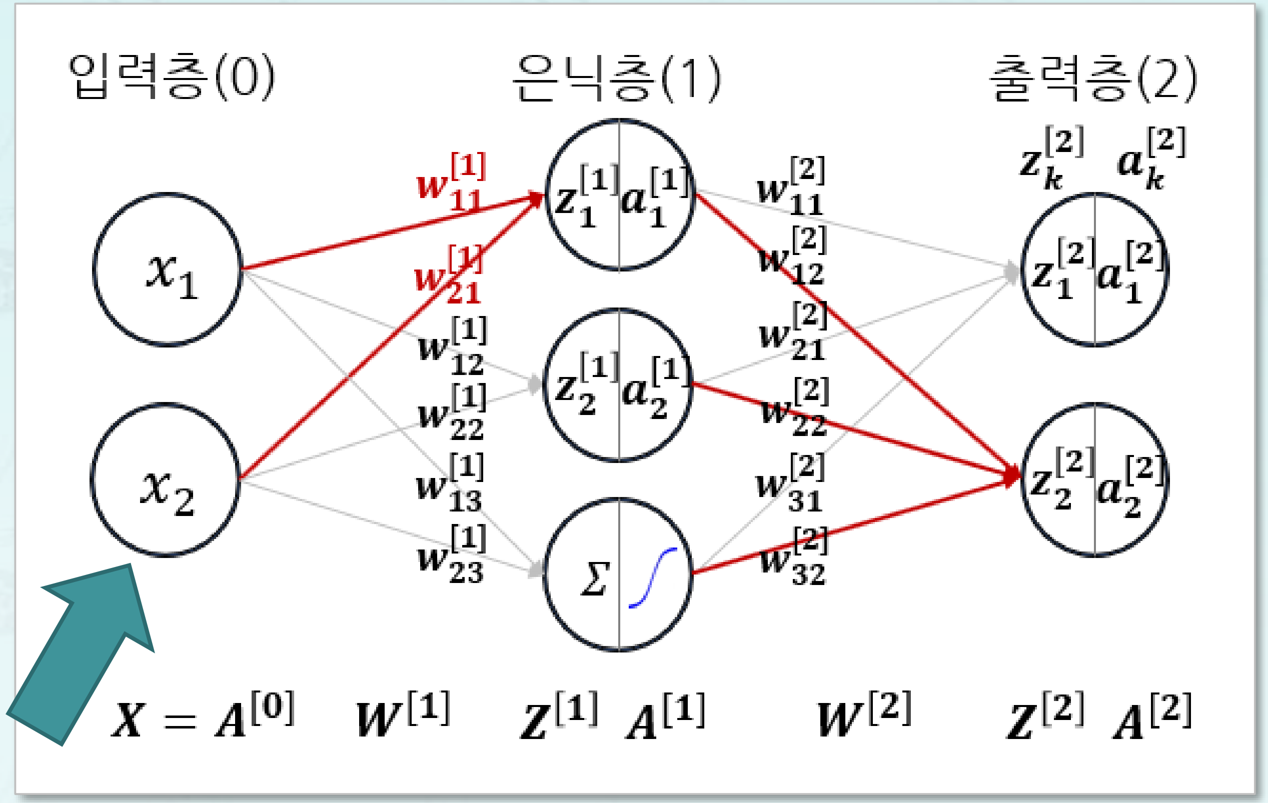
1. 인공 신경망: 간단한 신경망(복습)



1. 인공 신경망: 입력자료

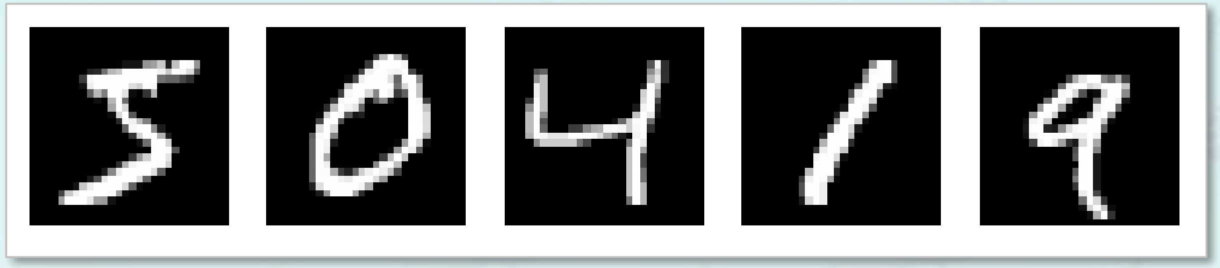
- **MNIST**

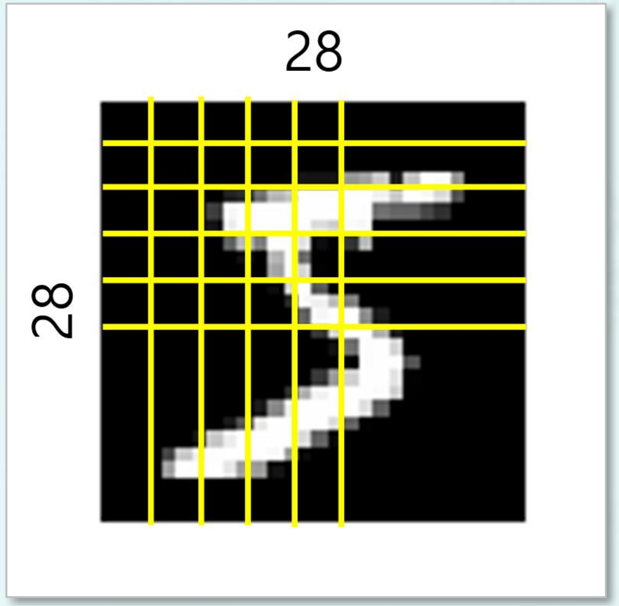




1. 인공 신경망: 입력자료

- **MNIST**
  - **28x28**





1. 인공 신경망: 신경망 구조

- **입력층 : 784 (28x28)**

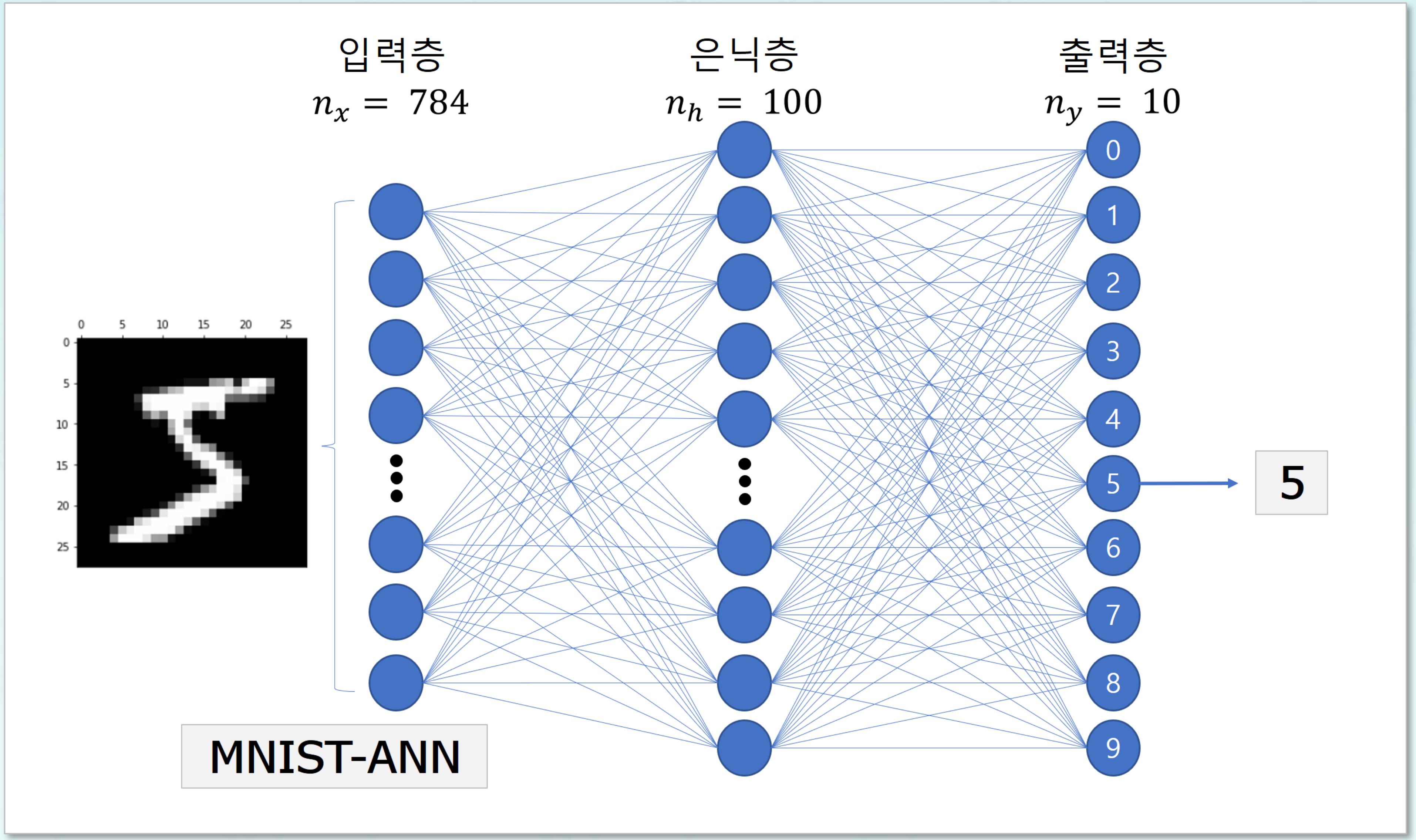
$$\mathbf{X} \in \mathbb{R}^{nxm}$$

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

1. 인공 신경망: 신경망 구조

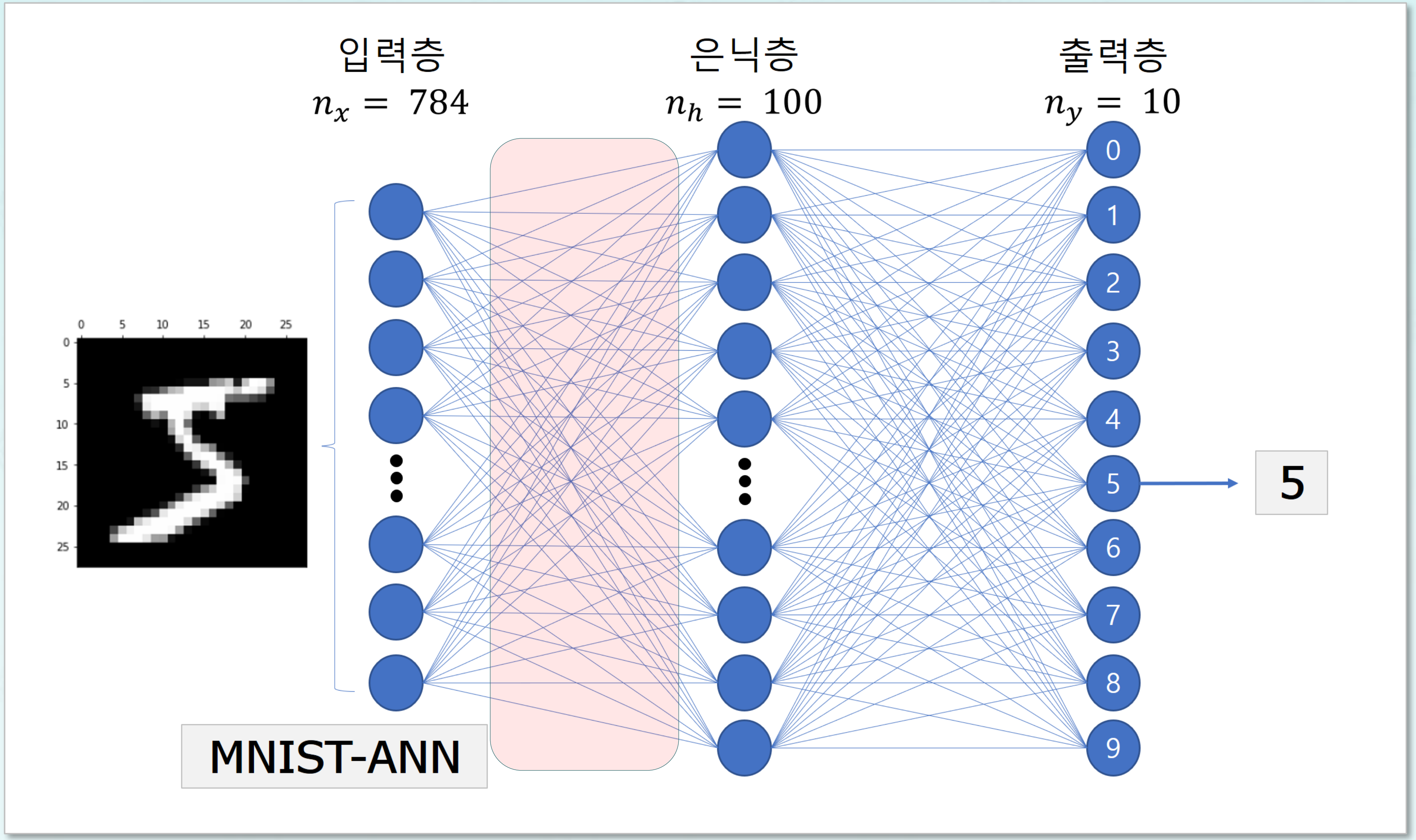
- 입력층 : **784(28x28)**
- 은닉층 : **100**
- **출력층 : 10**

2. 인공 신경망 구조 : 전체 그림



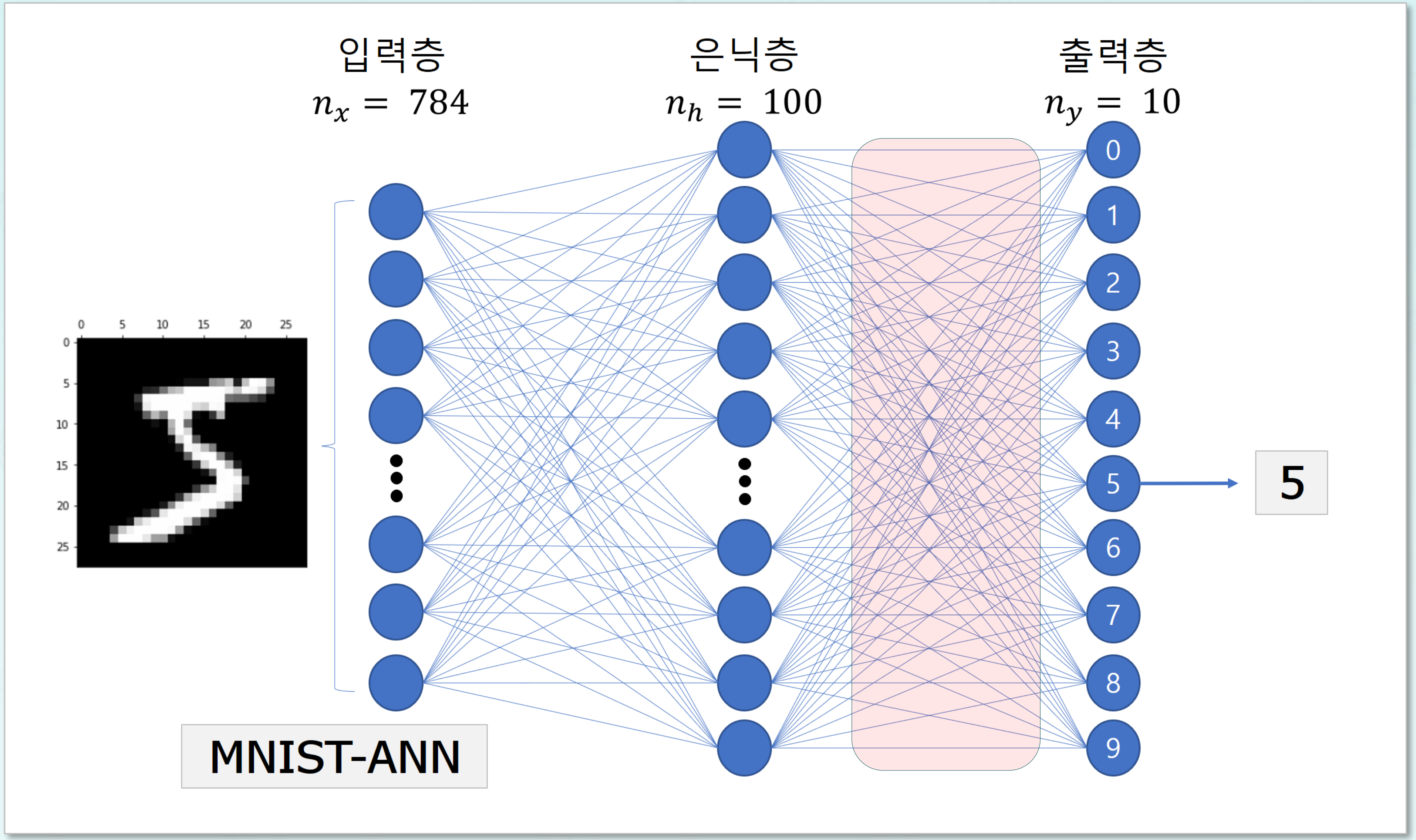
2. 인공 신경망 구조 : 가중치

- 가중치 계산
  - **784x100**



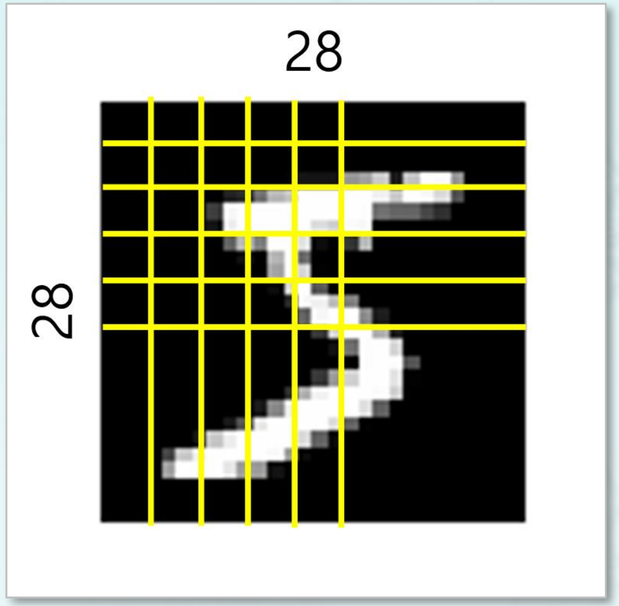
2. 인공 신경망 구조 : 가중치

- 가중치 계산
  - **784x100**
  - **100x10**



3. 인공 신경망 구현: 입력자료

- **X^{nxm}**
- **n = 784, m = 1**

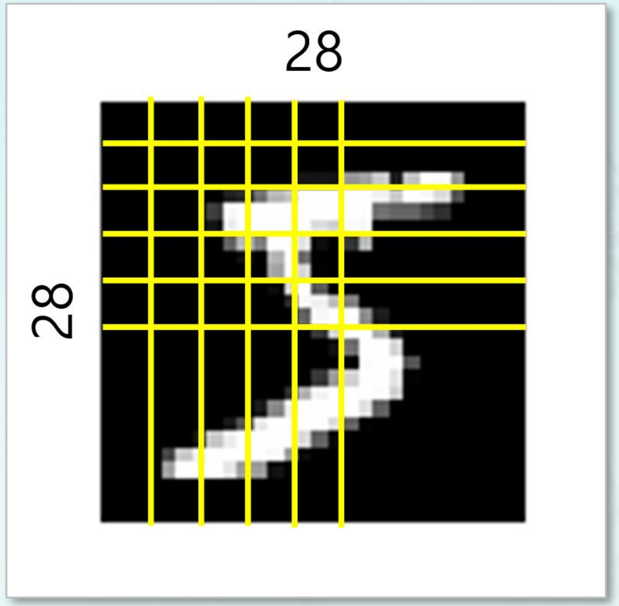


$$\mathbf{X} \in \mathbb{R}^{nxm}$$

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

3. 인공 신경망 구현: 입력자료

- X^{nxm}
- n = 784, m = 1



$$\mathbf{X} \in \mathbb{R}^{nxm}$$

$$\mathbf{X} = \begin{pmatrix} x_{\boxed{1}}^{(0)} \\ x_{\boxed{2}}^{(0)} \\ \vdots \\ x_{\boxed{783}}^{(0)} \\ x_{\boxed{784}}^{(0)} \end{pmatrix}$$

3. 인공 신경망 구현: 가중치 불러오기

- 사전에 학습된 가중치 사용
- **96%** 성능

3. 인공 신경망 구현: 가중치 불러오기

- **$W^{[1]}$ = 100x784**

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

3. 인공 신경망 구현: 가중치 불러오기

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

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

```
!type data/w_xh.txt
```

```
!cat data/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
```

3. 인공 신경망 구현: 은닉층 순입력 계산

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

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

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

3. 인공 신경망 구현: 출력층 계산

- **$A^{[1]}$: 100x1**

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

3. 인공 신경망 구현: 출력층 계산

- $A^{[1]}$: 100x1
- $W^{[2]}$: 10x100

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

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

4. 인공 신경망 구현 코드 : 전체 코드

```
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))
```

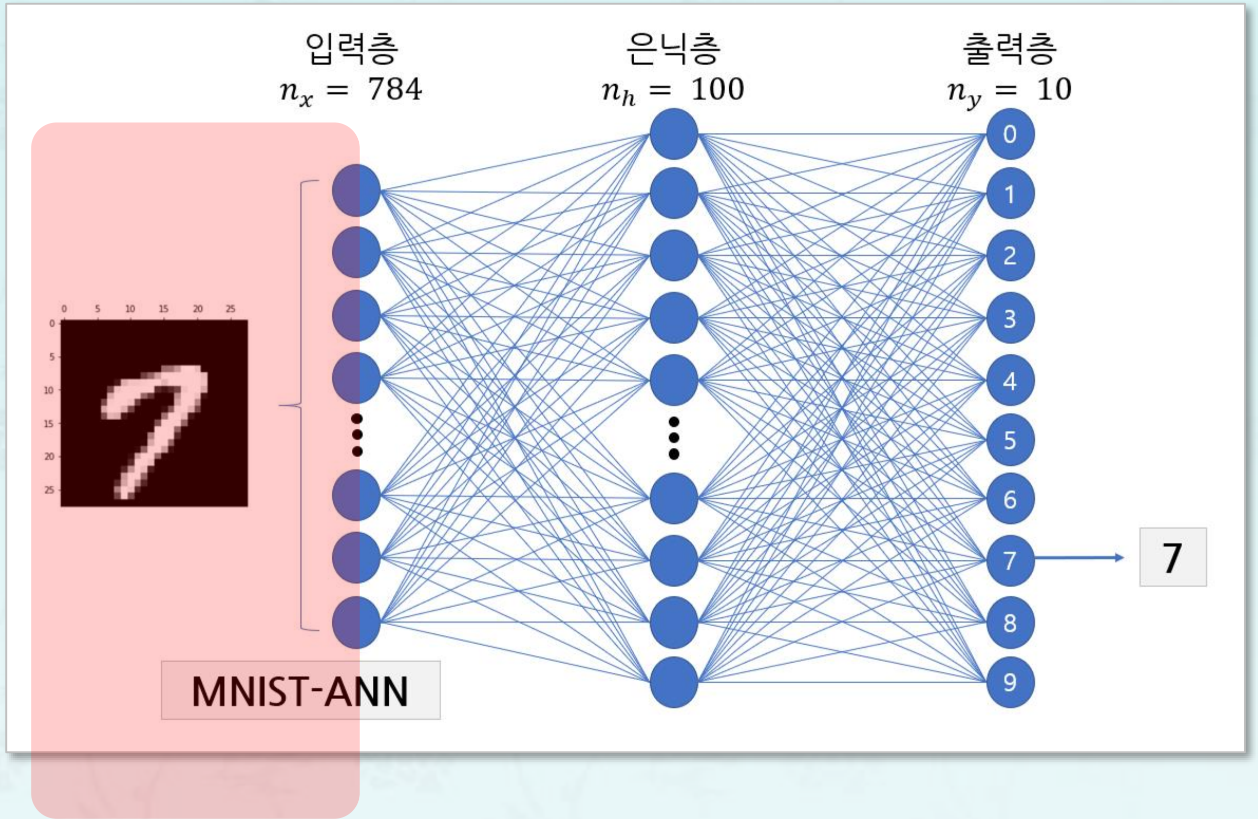
4. 인공 신경망 구현 코드 : 입력자료 불러오기

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

```python
(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))
```



4. 인공 신경망 구현 코드 : 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))
```

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

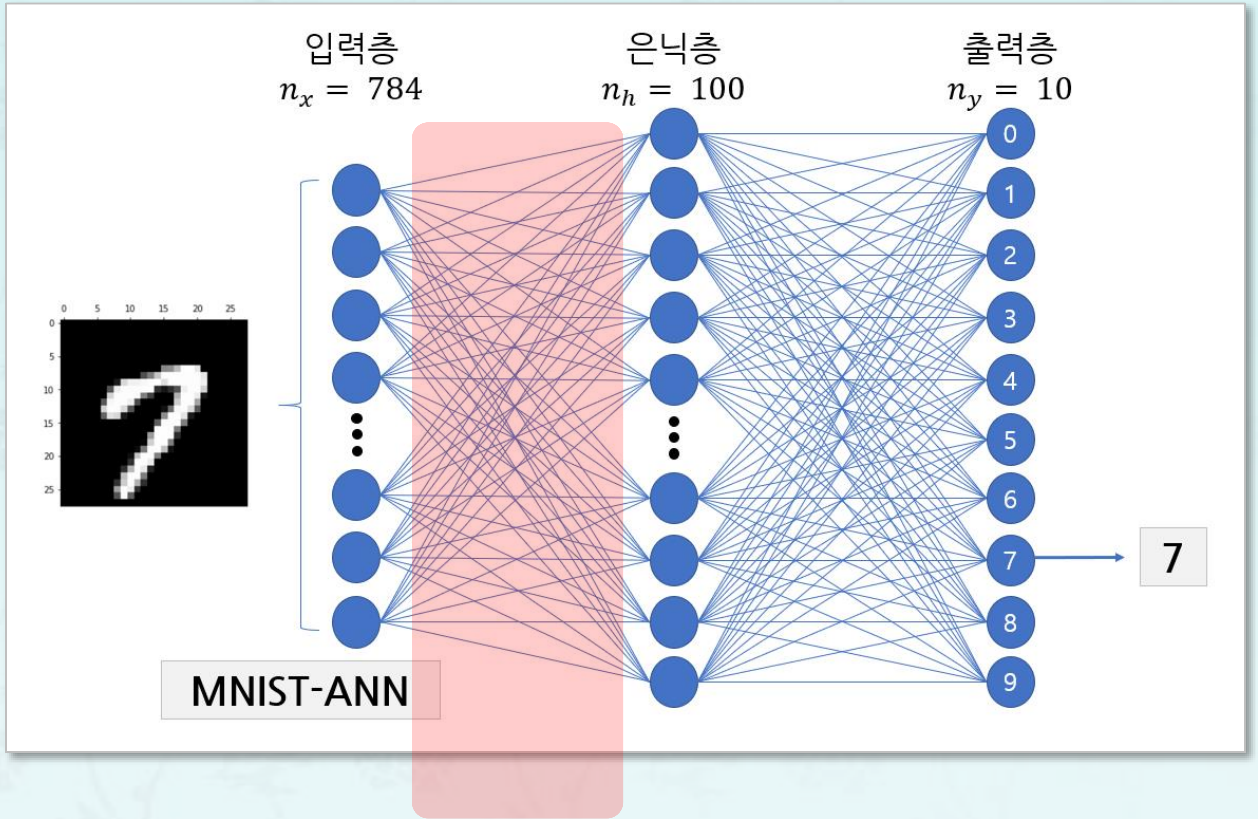
4. 인공 신경망 구현 코드 : 가중치 불러오기

```
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))
```



4. 인공 신경망 구현 코드 : 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))
```

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

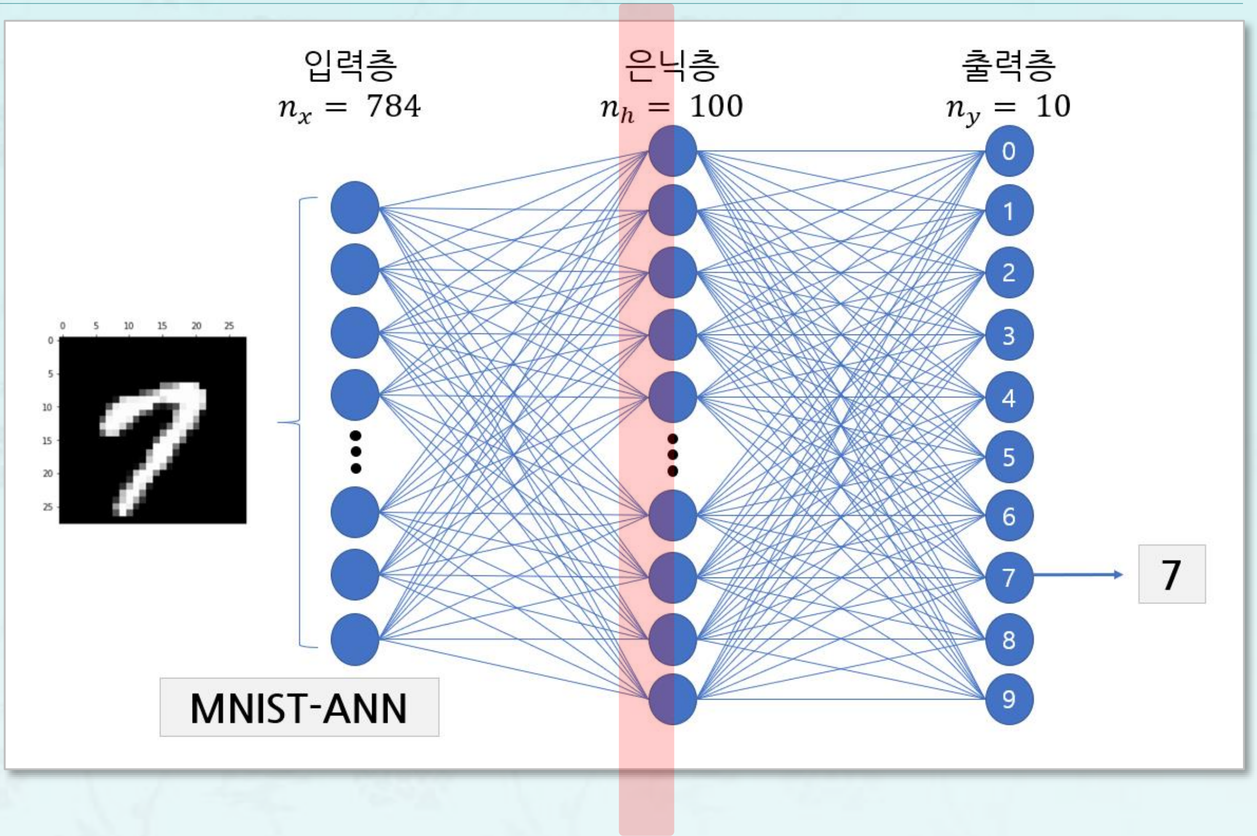
4. 인공 신경망 구현 코드 : 순입력 계산

```
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))
```



4. 인공 신경망 구현 코드 : 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))
```

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

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

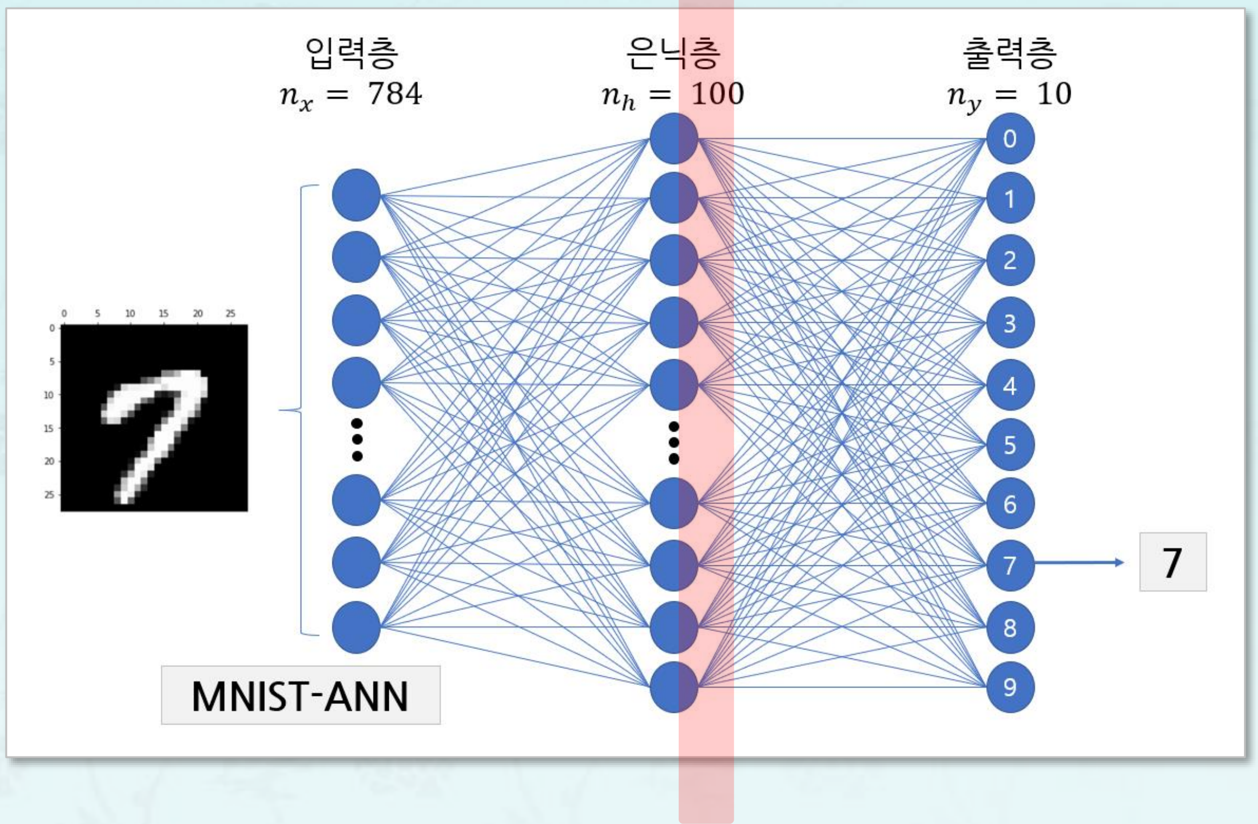
4. 인공 신경망 구현 코드 : 은닉층 계산

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

```python
(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))
```



4. 인공 신경망 구현 코드 : 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))
```

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

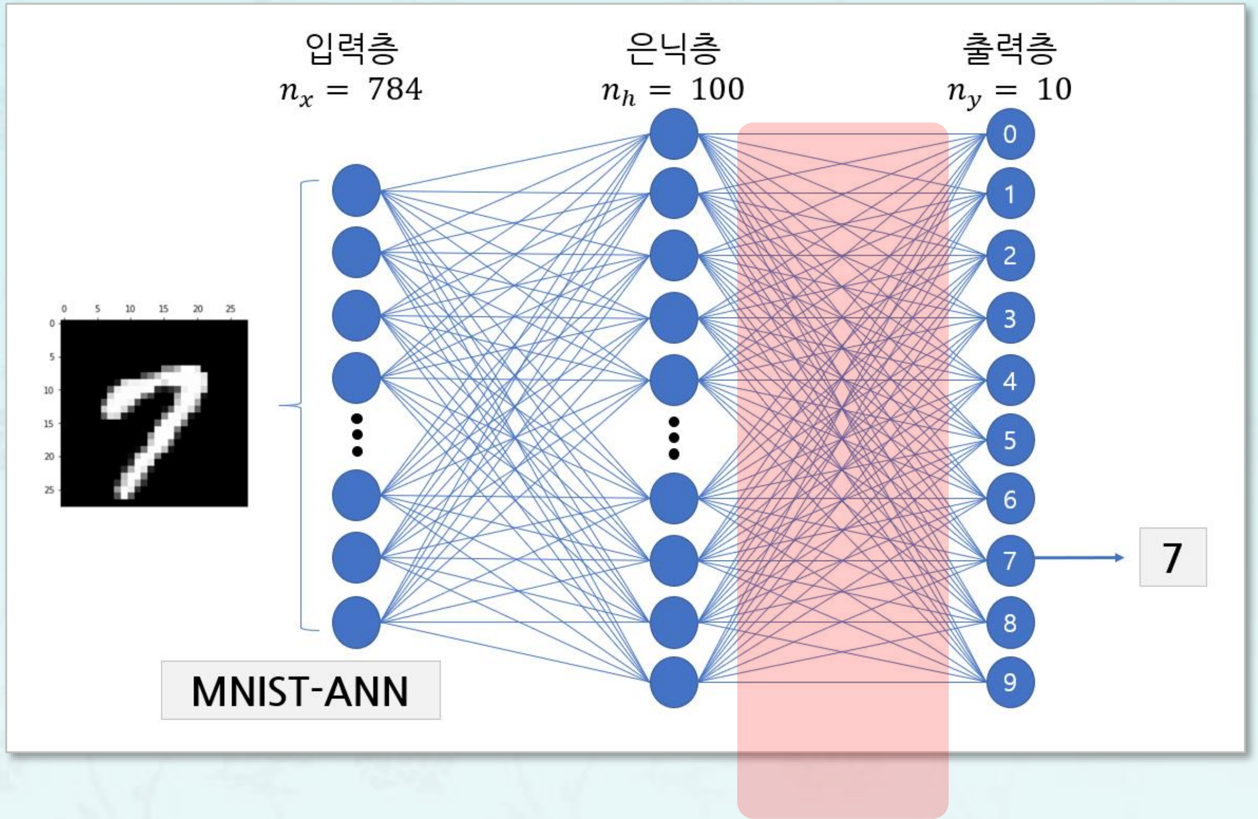
4. 인공 신경망 구현 코드 : 가중치 불러오기

```
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))
```



4. 인공 신경망 구현 코드 : 출력층 순입력 계산

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

```python
(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))
```

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

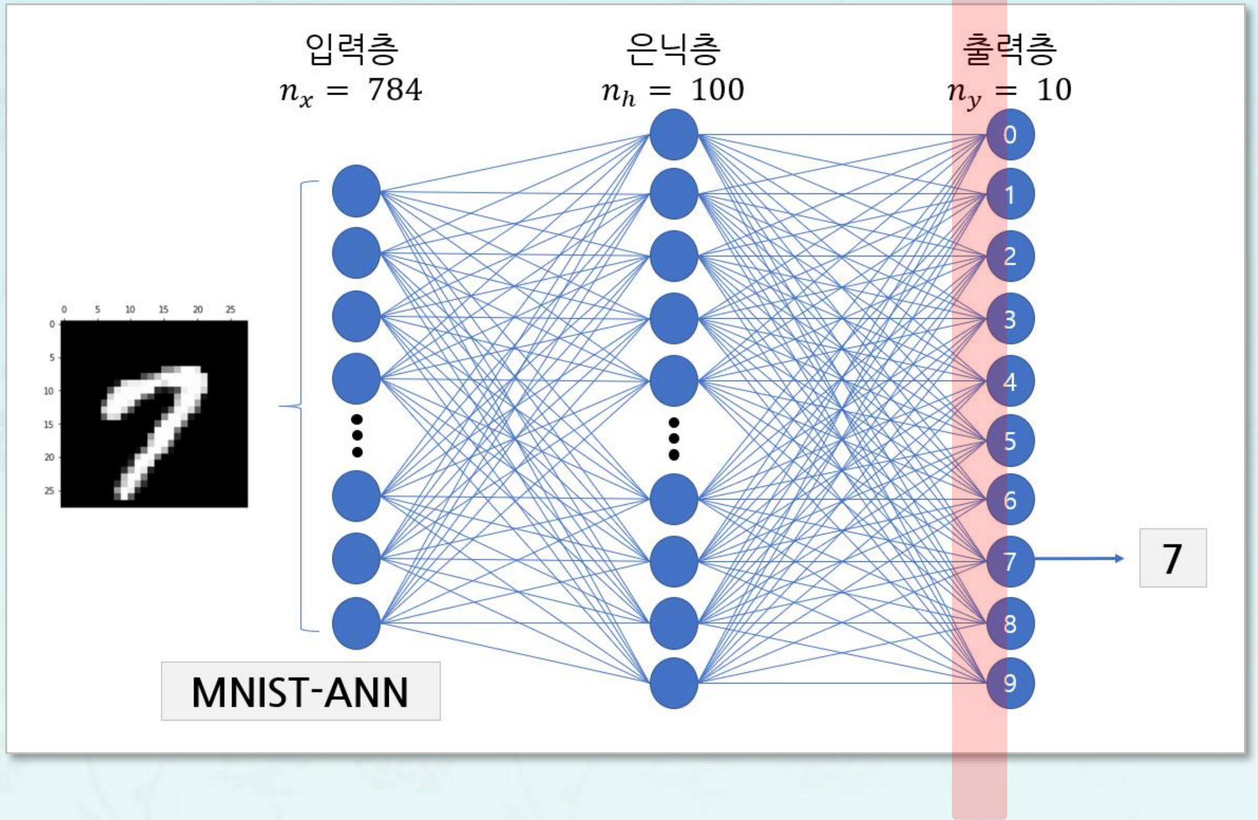
4. 인공 신경망 구현 코드 : 출력층 순입력 계산

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

```python
(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))
```



4. 인공 신경망 구현 코드 : 출력층 출력 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))
```

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

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

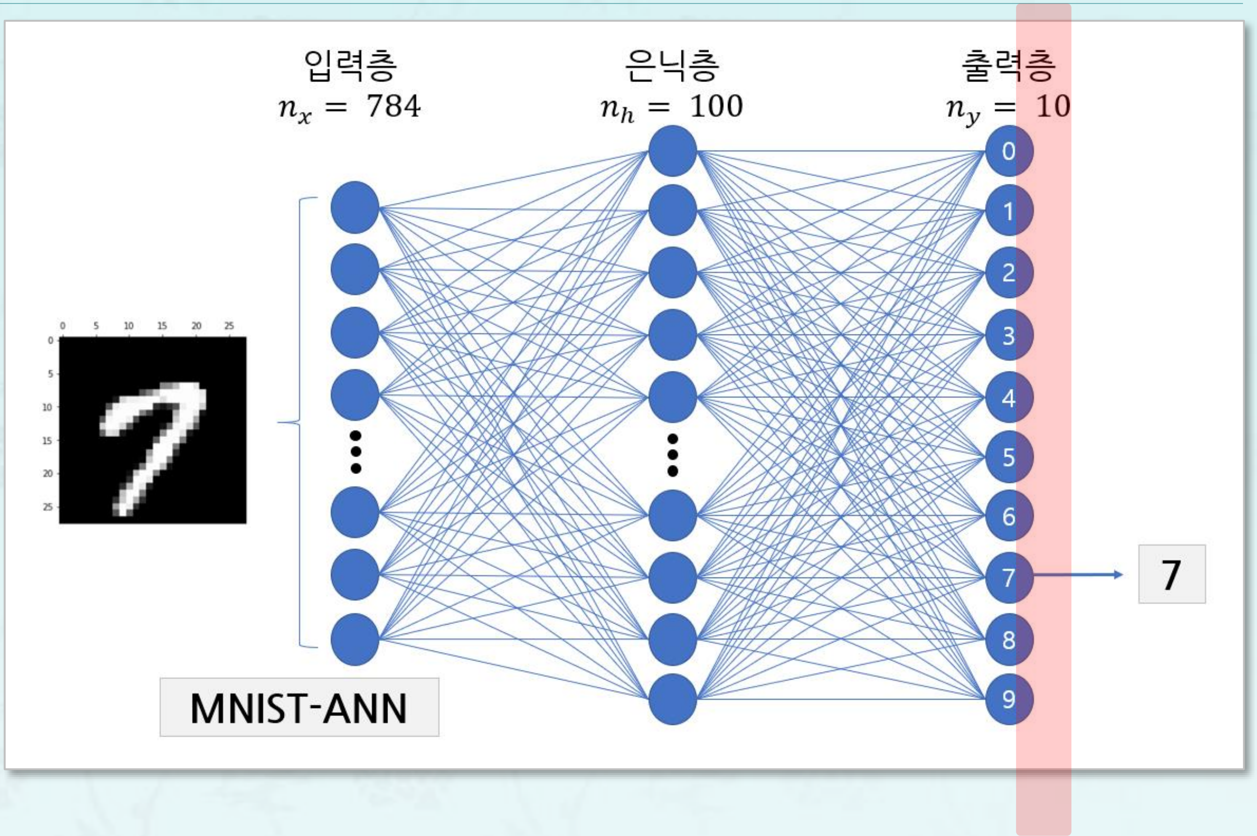
4. 인공 신경망 구현 코드 : 출력층 출력 계산

```
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))
```



4. 인공 신경망 구현 코드 : 결과 출력

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

```python
(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]
```

4. 인공 신경망 구현 코드 : 결과 출력

```
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 통한 순방향 신경망의 이해
  - 자료의 특성 이해
  - 순방향 신경망 설계
  - 가중치 불러오기