

10주차(2/3)

XOR 신경망 모델링

파이썬으로 배우는 기계학습

한동대학교
김영섭 교수

XOR 신경망 모델링

- 학습 목표
 - **XOR** 신경망을 처리하기 위해 입력자료 행렬을 구성한다.
 - **XOR** 신경망을 처리하기 위해 가중치 행렬을 구성한다.
 - **XOR** 신경망 행렬의 형상을 일반화 한다.
 - **XOR** 신경망 객체를 코딩한다.
- 학습 내용
 - 입력과 출력
 - 각 층의 노드 수
 - 가중치
 - 일반화
 - **XOR** 객체 구현

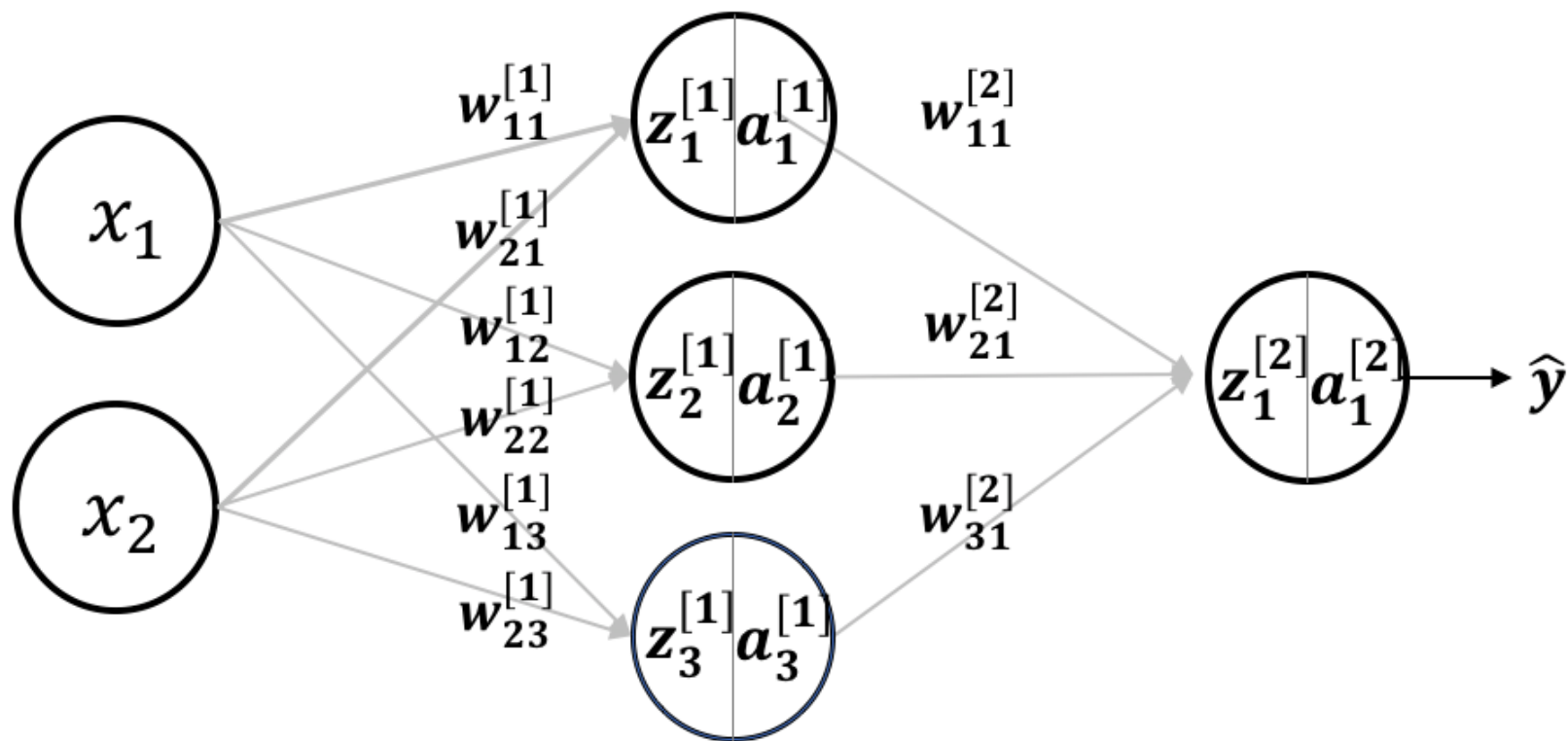
XOR 신경망 모델링

입력층(0)

은닉층(1)

출력층(2)

신경망 표기법, W_{ij}^T



$$W^{[1]} = \begin{pmatrix} w_{11}^{[1]} & w_{21}^{[1]} \\ w_{12}^{[1]} & w_{22}^{[1]} \\ w_{13}^{[1]} & w_{23}^{[1]} \end{pmatrix}$$

$$W^{[2]} = (w_{11}^{[2]} \quad w_{21}^{[2]} \quad w_{31}^{[2]})$$

$$X = A^{[0]} \quad W^{[1]} \quad Z^{[1]} \quad A^{[1]} \quad W^{[2]} \quad Z^{[2]} \quad A^{[2]}$$

$$Z^{[l]} = W^{[l]T} A^{[l-1]}$$

$$A^{[l]} = g(Z^{[l]})$$

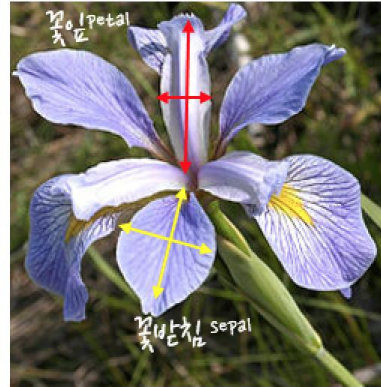
입력과 출력: 붓꽃 데이터



세토사 Setosa



버시컬라 versicolor



버지니카 Virginica

- 입력 자료 특성
 - 꽃잎의 길이
 - 꽃잎의 너비
 - 꽃받침의 길이
 - 꽃받침의 너비

입력과 출력: 붓꽃 데이터



- 입력 자료의 특성
 - 꽃잎의 길이
 - 꽃잎의 너비
 - 꽃받침의 길이
 - 꽃받침의 너비

$$X \in \mathbb{R}^{4 \times m}$$

$$X = \begin{pmatrix} x_1^{(1)} & x_1^{(2)} & x_1^{(3)} & \dots & x_1^{(m)} \\ x_2^{(1)} & x_2^{(2)} & x_2^{(3)} & \dots & x_2^{(m)} \\ x_3^{(1)} & x_3^{(2)} & x_3^{(3)} & \dots & x_3^{(m)} \\ x_4^{(1)} & x_4^{(2)} & x_4^{(3)} & \dots & x_4^{(m)} \end{pmatrix}$$

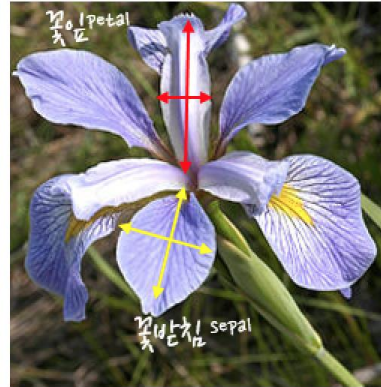
입력과 출력: 붓꽃 데이터



세타사 Setosa



버시컬라 Versicolor



버지니카 Virginica

■ 입력 자료의 특성

- 꽃잎의 길이
- 꽃잎의 너비
- 꽃받침의 길이
- 꽃받침의 너비

$$X \in \mathbb{R}^{4 \times m}$$

$$X = \begin{pmatrix} x_1^{(1)} & x_1^{(2)} & x_1^{(3)} & \cdots & x_1^{(m)} \\ x_2^{(1)} & x_2^{(2)} & x_2^{(3)} & \cdots & x_2^{(m)} \\ x_3^{(1)} & x_3^{(2)} & x_3^{(3)} & \cdots & x_3^{(m)} \\ x_4^{(1)} & x_4^{(2)} & x_4^{(3)} & \cdots & x_4^{(m)} \end{pmatrix}$$

$$X \in \mathbb{R}^{m \times 4}$$

$$X = \begin{pmatrix} x_1^{(1)} & x_2^{(1)} & x_3^{(1)} & x_4^{(1)} \\ x_1^{(2)} & x_2^{(2)} & x_3^{(2)} & x_4^{(2)} \\ x_1^{(3)} & x_2^{(3)} & x_3^{(3)} & x_4^{(3)} \\ \vdots & \vdots & \vdots & \vdots \\ x_1^{(m)} & x_2^{(m)} & x_3^{(m)} & x_4^{(m)} \end{pmatrix}$$

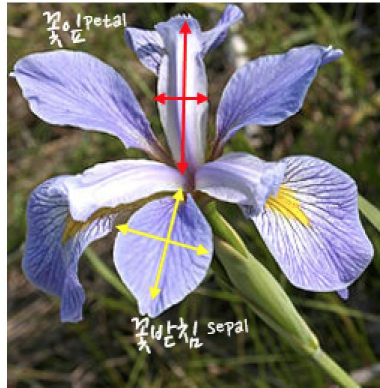
입력과 출력: 붓꽃 데이터



세토사 Setosa



버시칼라 versicolor



버지니카 Virginica

- 클래스 레이블
 - 세토사
 - 버시칼라
 - 버지니카

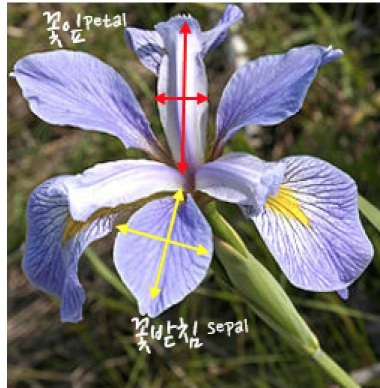
입력과 출력: 붓꽃 데이터



세토사 Setosa



버시칼라 Versicolor



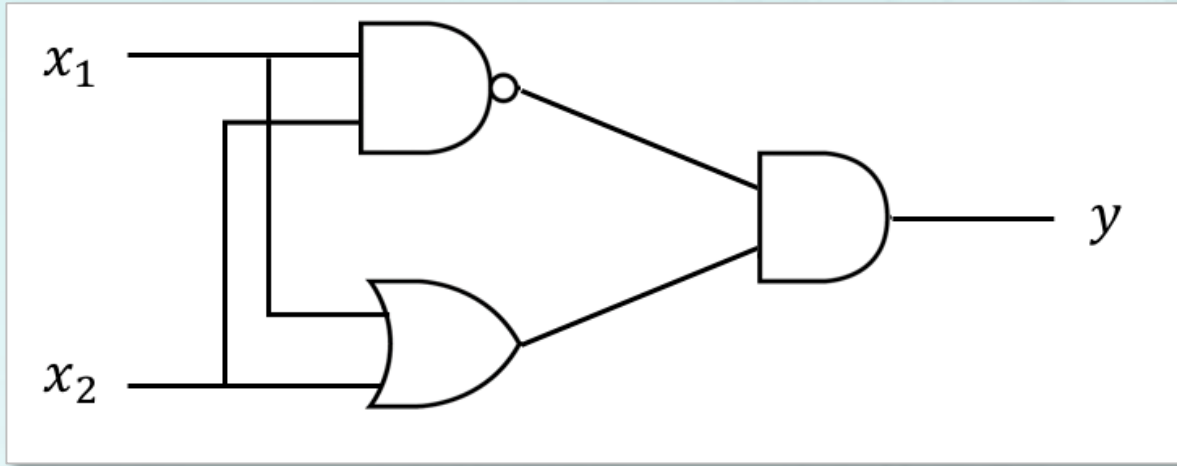
버지니카 Virginica

- 클래스 레이블
 - 세토사
 - 버시칼라
 - 버지니카

$$y \in \mathbb{R}^{1 \times m}$$

$$y = (y^{(1)} \quad y^{(2)} \quad \dots \quad y^{(m)})$$

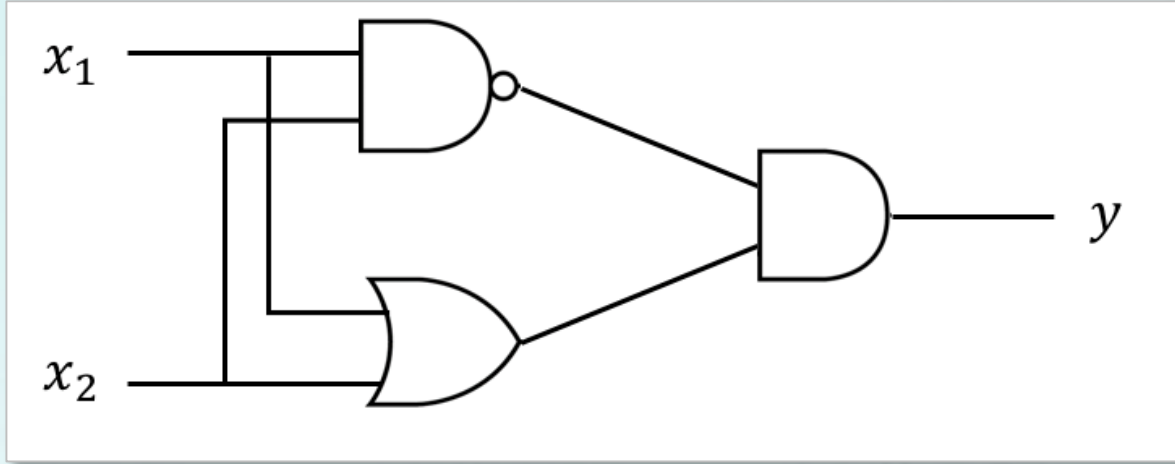
입력과 출력: XOR 데이터



■ 입력 자료의 특성

- x_1
- x_2

입력과 출력: XOR 데이터



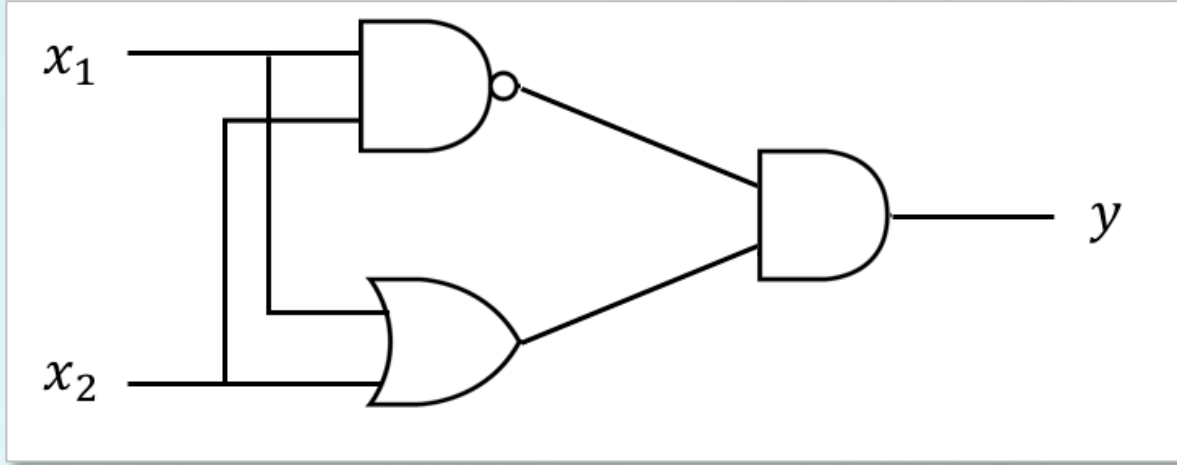
■ 입력 자료의 특성

- x_1
- x_2

$$X \in \mathbb{R}^{2 \times 4}$$

$$\begin{aligned} X &= \begin{pmatrix} x_1^{(1)} & x_1^{(2)} & x_1^{(3)} & x_1^{(4)} \\ x_2^{(1)} & x_2^{(2)} & x_2^{(3)} & x_2^{(4)} \end{pmatrix} \\ &= \begin{pmatrix} 0 & 0 & 1 & 1 \\ 0 & 1 & 0 & 1 \end{pmatrix} \end{aligned}$$

입력과 출력: XOR 데이터



- 클래스 레이블
 - True (1)
 - False (0)

$$y \in \mathbb{R}^{1 \times 4}$$

$$\begin{aligned} y &= (y^{(1)} \quad y^{(2)} \quad y^{(3)} \quad y^{(4)}) \\ &= (0 \quad 1 \quad 1 \quad 0) \end{aligned}$$

입력과 출력: XOR 데이터

■

$$X \in \mathbb{R}^{2 \times 4}$$

$$X = \begin{pmatrix} x_1^{(1)} & x_1^{(2)} & x_1^{(3)} & x_1^{(4)} \\ x_2^{(1)} & x_2^{(2)} & x_2^{(3)} & x_2^{(4)} \end{pmatrix}$$
$$= \begin{pmatrix} 0 & 0 & 1 & 1 \\ 0 & 1 & 0 & 1 \end{pmatrix}$$

```
1 import numpy as np
2 X = np.array([[0, 0, 1, 1], [0, 1, 0, 1]])
3 y = np.array([[0, 1, 1, 0]])
4 print('X.shape={}, y.shape={}'.format(X.shape, y.shape))
5 print(X)
6 print(y)
```

■

$$y \in \mathbb{R}^{1 \times 4}$$

$$y = (y^{(1)} \quad y^{(2)} \quad y^{(3)} \quad y^{(4)})$$
$$= (0 \quad 1 \quad 1 \quad 0)$$

입력과 출력: XOR 데이터

- $X \in \mathbb{R}^{2 \times 4}$

$$X = \begin{pmatrix} x_1^{(1)} & x_1^{(2)} & x_1^{(3)} & x_1^{(4)} \\ x_2^{(1)} & x_2^{(2)} & x_2^{(3)} & x_2^{(4)} \end{pmatrix}$$
$$= \begin{pmatrix} 0 & 0 & 1 & 1 \\ 0 & 1 & 0 & 1 \end{pmatrix}$$

- $y \in \mathbb{R}^{1 \times 4}$

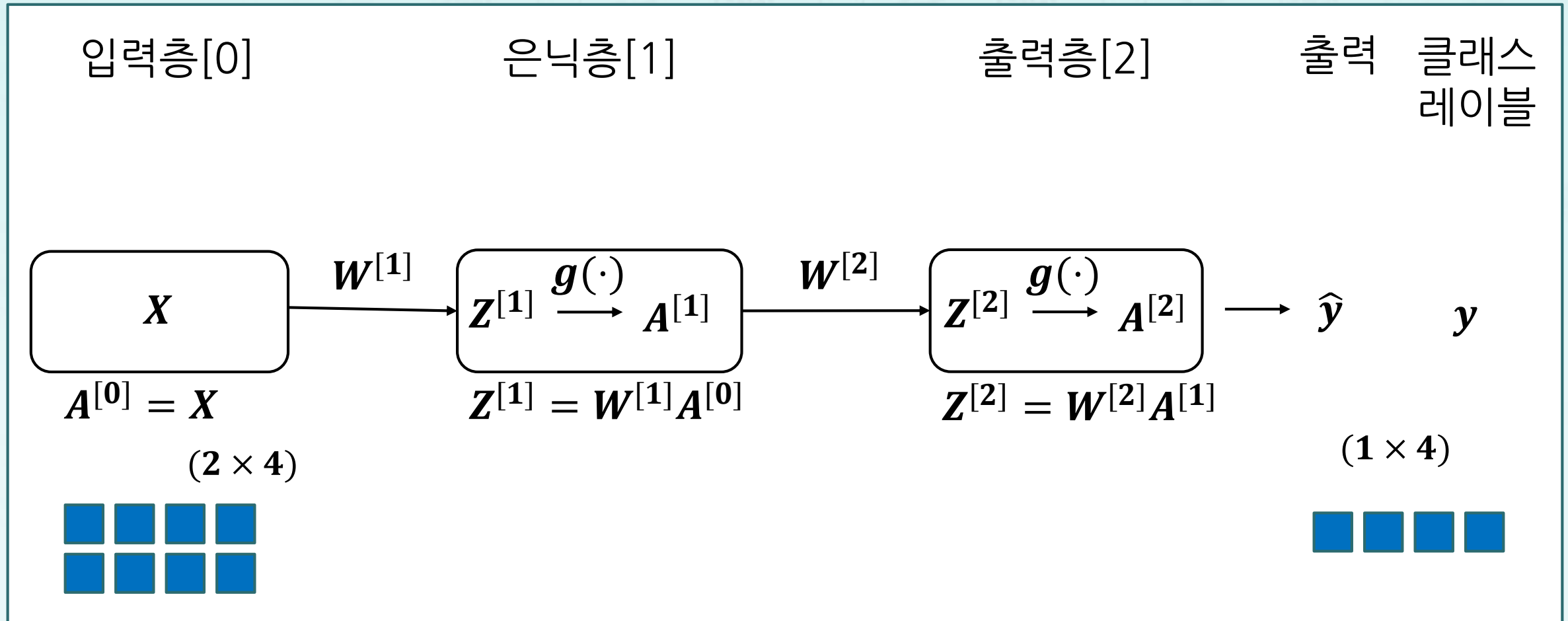
$$y = (y^{(1)} \quad y^{(2)} \quad y^{(3)} \quad y^{(4)})$$
$$= (0 \quad 1 \quad 1 \quad 0)$$

```
1 import numpy as np
2 X = np.array([[0, 0, 1, 1], [0, 1, 0, 1]])
3 y = np.array([[0, 1, 1, 0]])
4 print('X.shape={}, y.shape={}'.format(X.shape, y.shape))
5 print(X)
6 print(y)
```

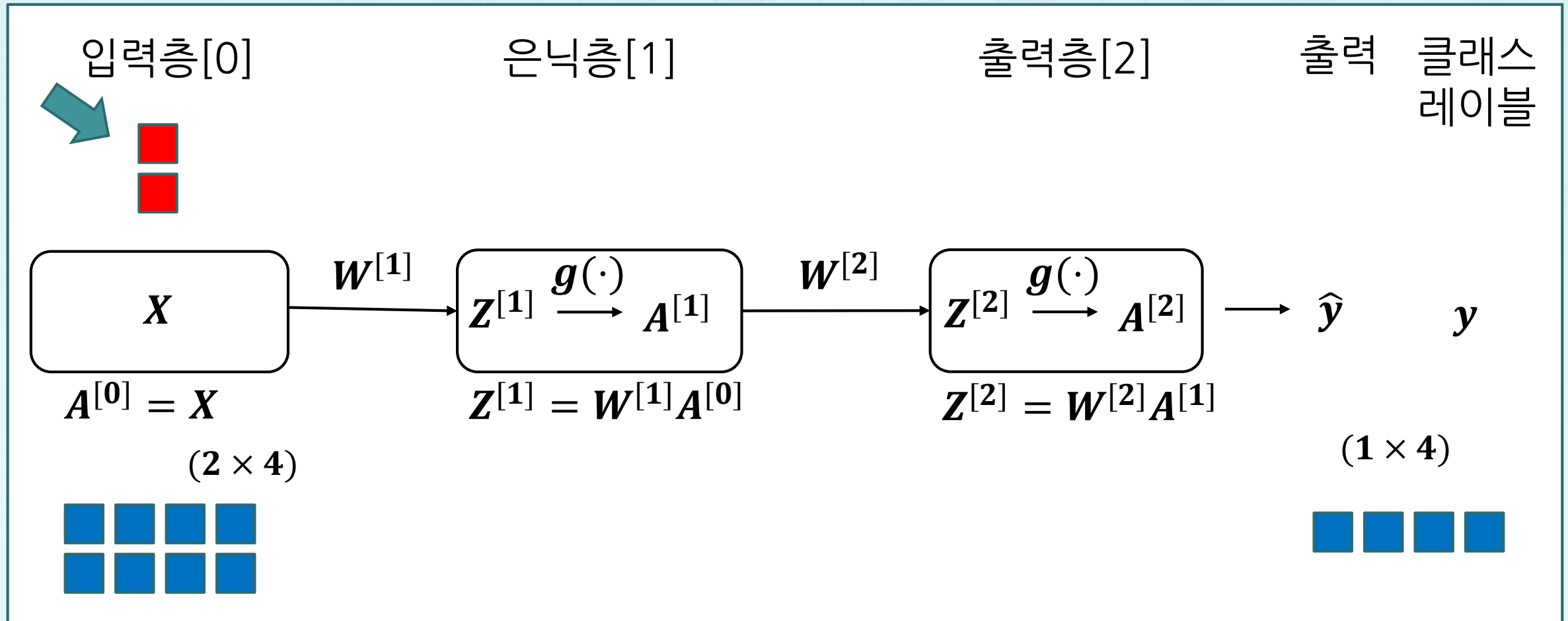


```
X.shape=(2, 4), y.shape=(1, 4)
[[0 0 1 1]
 [0 1 0 1]]
[[0 1 1 0]]
```

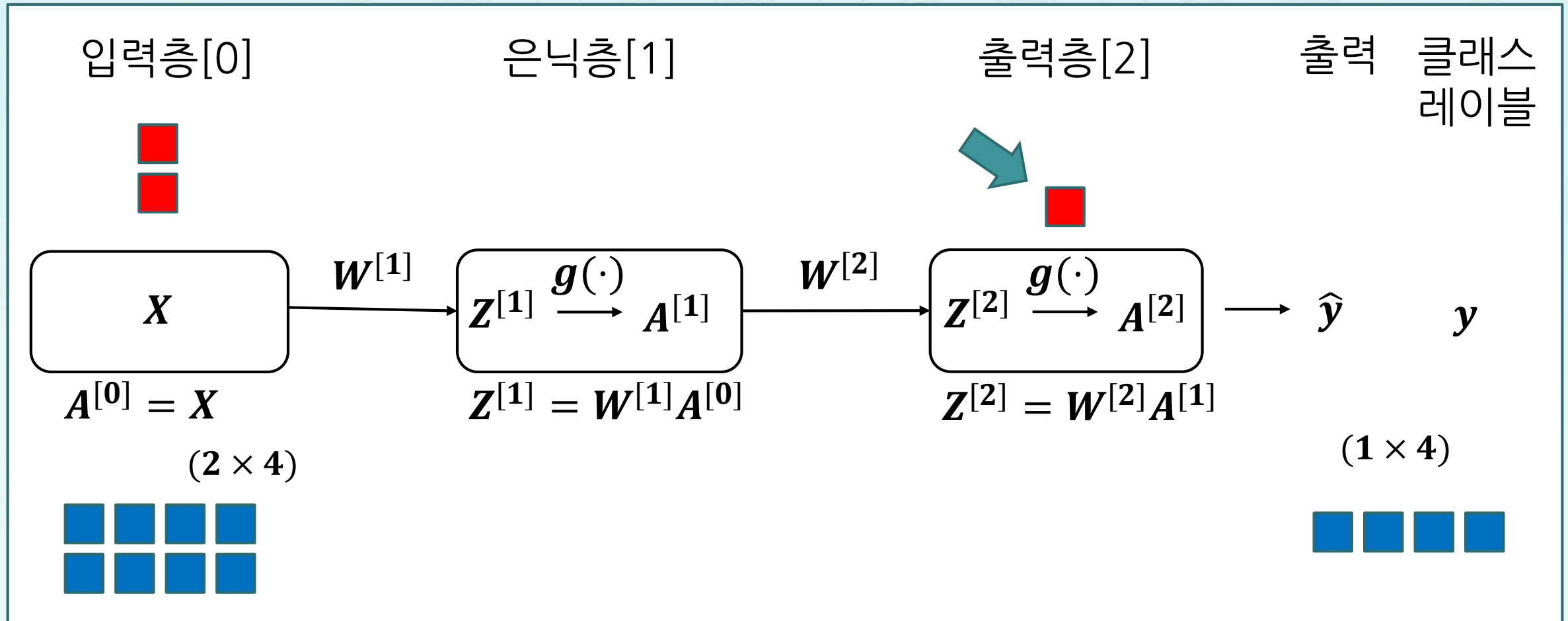
입력과 출력: XOR 데이터



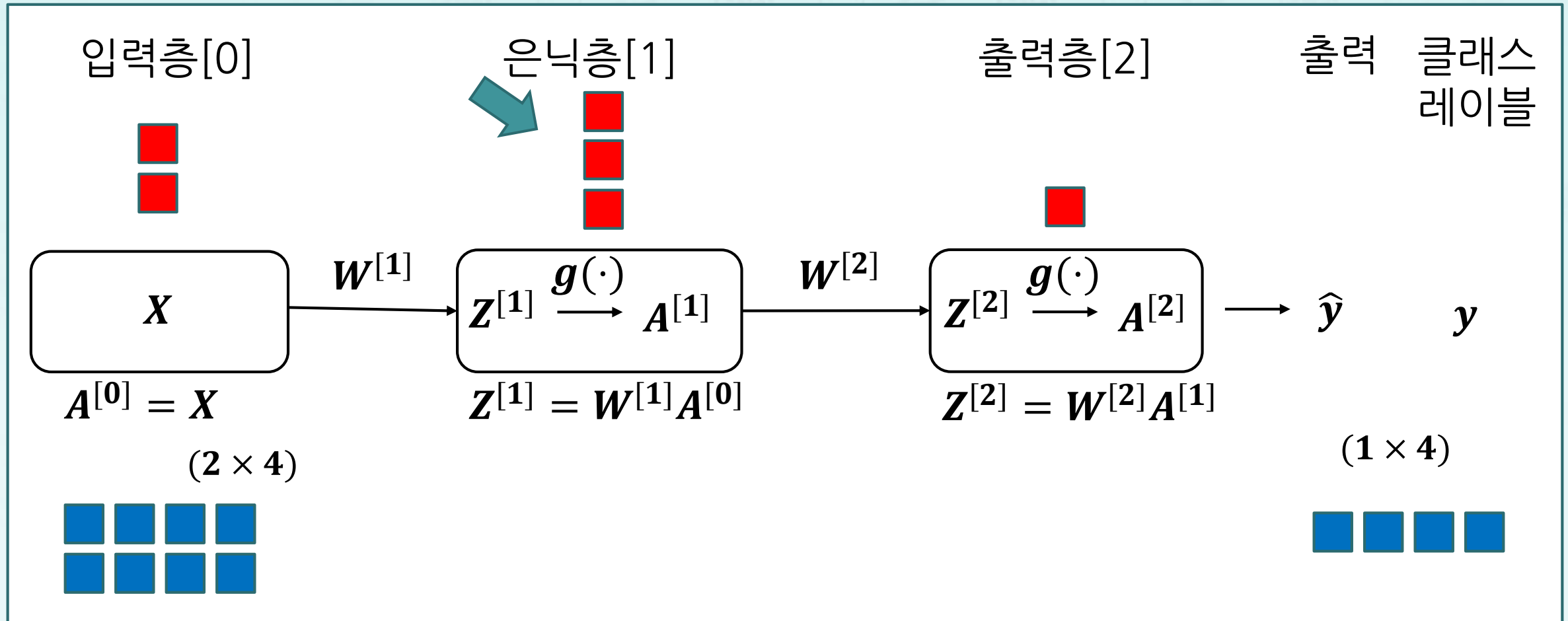
각 층의 노드 수: 입력층



각 층의 노드 수: 출력층



각 층의 노드 수: 은닉층

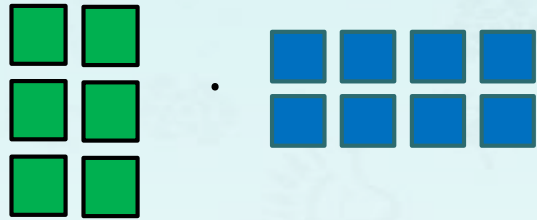


가중치: 행렬의 형상

- $W^{[1]}$: (은닉층 노드 수 \times 입력층 노드 수)

$$W^{[1]} \cdot A^{[0]}$$

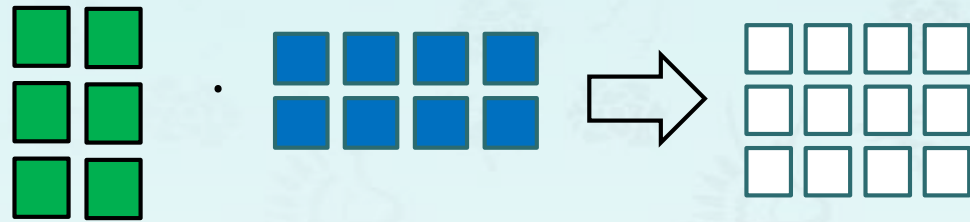
$$(? \times ?) \cdot (2 \times 4)$$



가중치: 행렬의 형상

- $W^{[1]}$: (은닉층 노드 수 \times 입력층 노드 수)

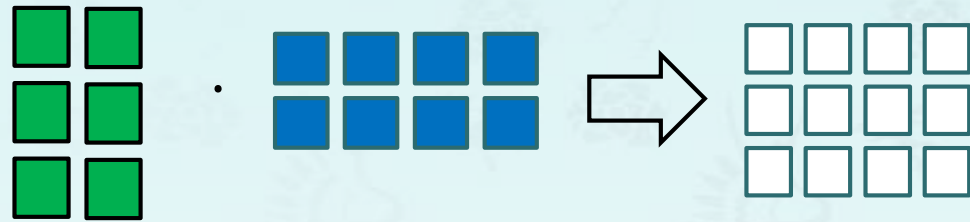
$$W^{[1]} \cdot A^{[0]} \Rightarrow Z^{[1]}$$
$$(3 \times 2) \cdot (2 \times 4) \Rightarrow (3 \times 4)$$



가중치: 행렬의 형상

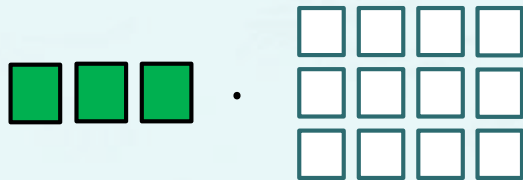
■

$$W^{[1]} \cdot A^{[0]} \Rightarrow Z^{[1]}$$
$$(3 \times 2) \cdot (2 \times 4) \quad (3 \times 4)$$



- $W^{[2]}$: (출력층 노드 수 x 은닉층 노드 수)

$$W^{[2]} \cdot A^{[1]}$$
$$(? \times ?) \cdot (3 \times 4)$$

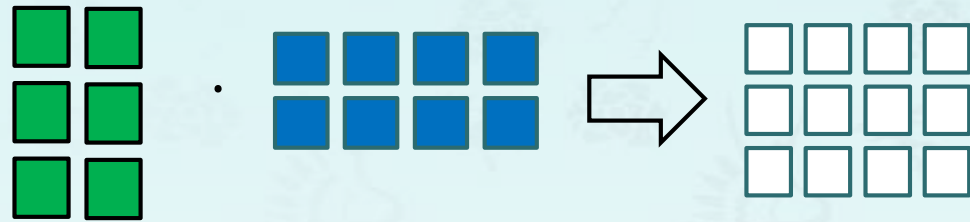


가중치: 행렬의 형상

■

$$W^{[1]} \cdot A^{[0]} \Rightarrow Z^{[1]}$$

$$(3 \times 2) \cdot (2 \times 4) \quad (3 \times 4)$$



■ $W^{[2]}$: (출력층 노드 수 x 은닉층 노드 수)

$$W^{[2]} \cdot A^{[1]} \Rightarrow Z^{[2]}$$

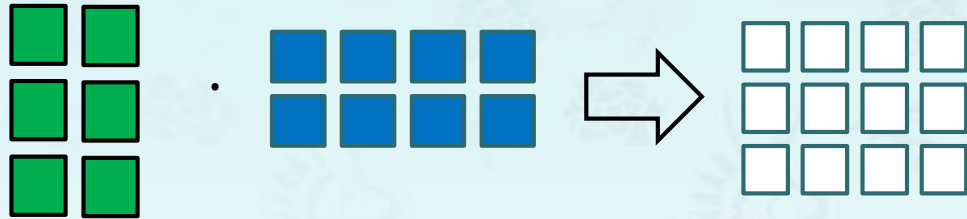
$$(1 \times 3) \cdot (3 \times 4) \quad (4 \times 1)$$



가중치: 행렬의 형상 구현


$$W^{[1]} \cdot A^{[0]} \Rightarrow Z^{[1]}$$

$(3 \times 2) \cdot (2 \times 4) \Rightarrow (3 \times 4)$



$$W^{[2]} \cdot A^{[1]} \Rightarrow Z^{[2]}$$

$(1 \times 3) \cdot (3 \times 4) \Rightarrow (4 \times 1)$

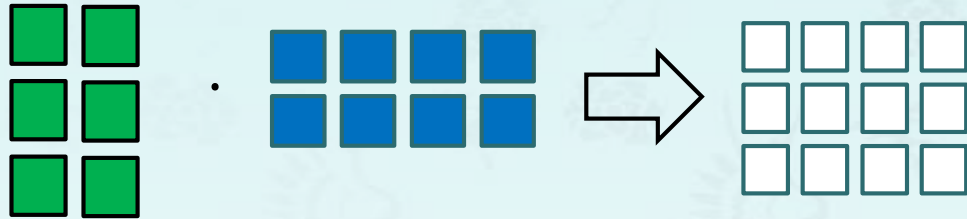


```
1 import numpy as np
2 n_x = X.shape[0]
3 n_y = Y.shape[0]
4 n_h = 3
5 np.random.seed(1)
6 W1 = 2*np.random.random((n_h, n_x)) - 1
7 W2 = 2*np.random.random((n_y, n_h)) - 1
```

가중치: 행렬의 형상 구현


$$W^{[1]} \cdot A^{[0]} \Rightarrow Z^{[1]}$$

$(3 \times 2) \cdot (2 \times 4) \Rightarrow (3 \times 4)$



$$W^{[2]} \cdot A^{[1]} \Rightarrow Z^{[2]}$$

$(3 \times 1) \cdot (3 \times 4) \Rightarrow (4 \times 1)$



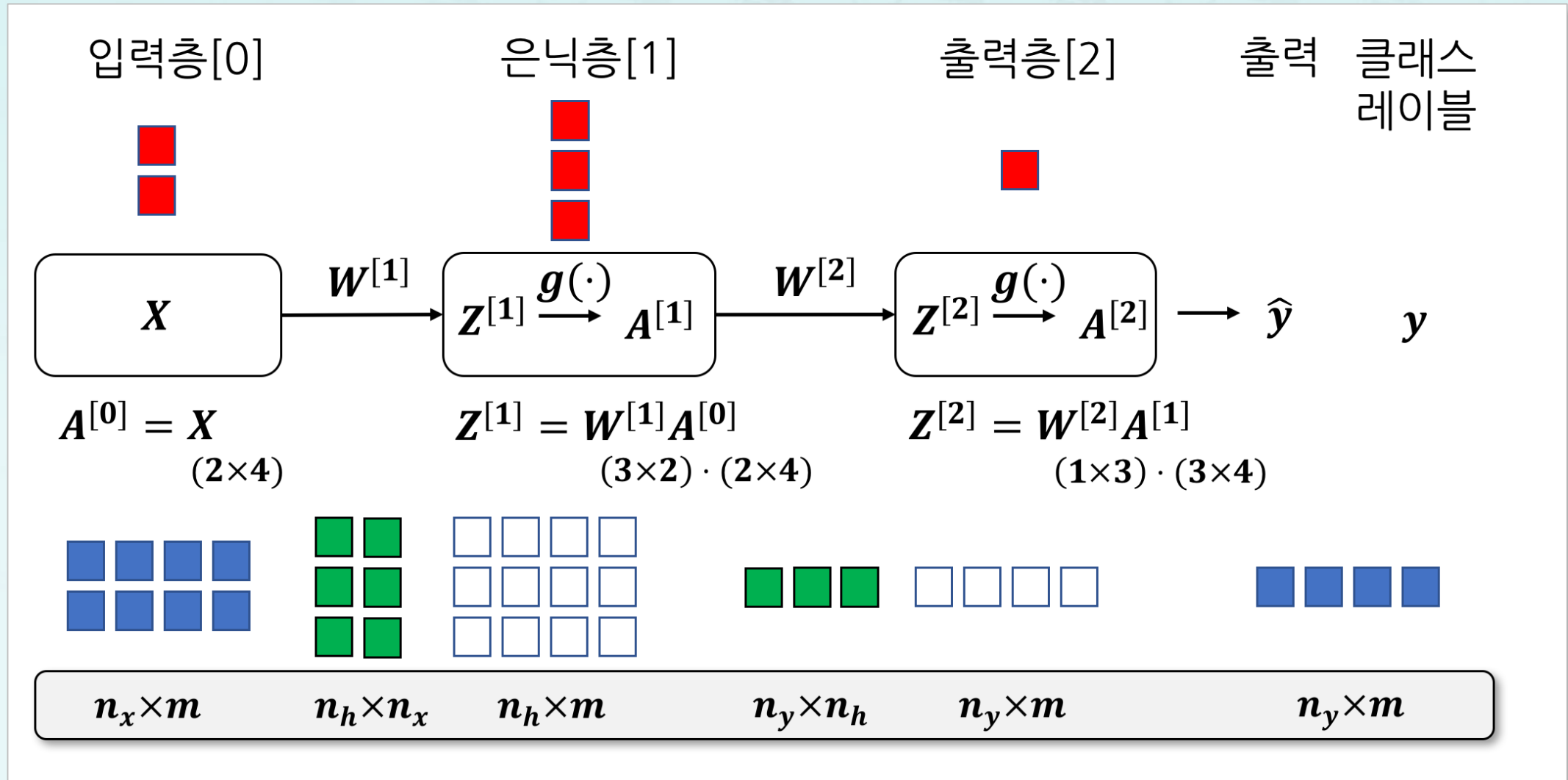
```
1 import numpy as np
2 n_x = X.shape[0]
3 n_y = Y.shape[0]
4 n_h = 3
5 np.random.seed(1)
6 W1 = 2*np.random.random((n_h, n_x)) - 1
7 W2 = 2*np.random.random((n_y, n_h)) - 1
```

```
print("W1: {}".format(W1))
print("W2: {}".format(W2))
```

W1: $\begin{bmatrix} -0.16595599 & 0.44064899 \\ -0.99977125 & -0.39533485 \\ -0.70648822 & -0.81532281 \end{bmatrix}$

W2: $\begin{bmatrix} -0.62747958 & -0.30887855 & -0.20646505 \end{bmatrix}$


신경망 행렬: 형상 일반화



XOR 객체 구현: 클래스 이름

XOR 객체 구현: 클래스 이름

- 클래스
 - 이름: **NeuralNetwork**



```
1 class NeuralNetwork():
2     """ This class implements a multi-perceptron
3         with backpropagation. This handles a simple logics
4         such as OR, AND, NAND, and NOR gates, including XOR.
5     """
6     def __init__(self, net_arch, eta=0.1, epochs=10000,
7                 random_seed=1):
8         self.layers = len(net_arch)
9         self.net_arch = net_arch
10        self.eta = eta
11        self.epochs = epochs
12        self.random_seed = random_seed
13
14    def g(self, x):
15        return 1/(1 + np.exp((-x)))
16
17    def g_prime(self, x):
18        return self.g(x) * (1 - self.g(x))
19
20    def fit(self, X, Y):
```

XOR 객체 구현: 생성자

- 클래스

- 이름: **NeuralNetwork**
- 생성자: **__init__()**
 - **net_arch** : 신경망 구조
 - Ex. [2, 3, 1] 혹은 [2, 3, 4, 1]
 - **eta** : 학습률
 - **epochs** : 반복 횟수
 - **random_seed** : 랜덤 시드

```
1 class NeuralNetwork():
2     """ This class implements a multi-perceptron
3         with backpropagation. This handles a simple logics
4         such as OR, AND, NAND, and NOR gates, including XOR.
5     """
6     def __init__(self, net_arch, eta=0.1, epochs=10000,
7                 random_seed=1):
8         self.layers = len(net_arch)
9         self.net_arch = net_arch
10        self.eta = eta
11        self.epochs = epochs
12        self.random_seed = random_seed
13
14        def g(self, x):
15            return 1/(1 + np.exp((-x)))
16
17        def g_prime(self, x):
18            return self.g(x) * (1 - self.g(x))
19
20        def fit(self, X, Y):
```

XOR 객체 구현: 활성화 함수

- 클래스
 - 이름: **NeuralNetwork**
 - 생성자: **__init__()**
 - 활성화 함수: **g()**

```
1 class NeuralNetwork():
2     """ This class implements a multi-perceptron
3         with backpropagation. This handles a simple logics
4         such as OR, AND, NAND, and NOR gates, including XOR.
5     """
6     def __init__(self, net_arch, eta=0.1, epochs=10000,
7                 random_seed=1):
8         self.layers = len(net_arch)
9         self.net_arch = net_arch
10        self.eta = eta
11        self.epochs = epochs
12        self.random_seed = random_seed
13
14    ➡ def g(self, x):
15        return 1/(1 + np.exp((-x)))
16
17    def g_prime(self, x):
18        return self.g(x) * (1 - self.g(x))
19
20    def fit(self, X, Y):
```

XOR 객체 구현: 활성화 함수 미분

■ 클래스

- 이름: **NeuralNetwork**
- 생성자: **__init__()**
- 활성화 함수: **g()**
- 활성화 함수 미분 : **g_prime()**

```
1 class NeuralNetwork():
2     """ This class implements a multi-perceptron
3         with backpropagation. This handles a simple logics
4         such as OR, AND, NAND, and NOR gates, including XOR.
5     """
6     def __init__(self, net_arch, eta=0.1, epochs=10000,
7                 random_seed=1):
8         self.layers = len(net_arch)
9         self.net_arch = net_arch
10        self.eta = eta
11        self.epochs = epochs
12        self.random_seed = random_seed
13
14        def g(self, x):
15            return 1/(1 + np.exp((-x)))
16
17        ➡ def g_prime(self, x):
18            return self.g(x) * (1 - self.g(x))
19
20        def fit(self, X, Y):
```

XOR 객체 구현: 활성화 함수 미분

■ 클래스

- 이름: **NeuralNetwork**
- 생성자: **__init__()**
- 활성화 함수: **g()**
- 활성화 함수 미분 : **g_prime()**

```
1 class NeuralNetwork():
2     """ This class implements a multi-perceptron
3         with backpropagation. This handles a simple logics
4         such as OR, AND, NAND, and NOR gates, including XOR.
5     """
6     def __init__(self, net_arch, eta=0.1, epochs=10000,
7                 random_seed=1):
8         self.layers = len(net_arch)
9         self.net_arch = net_arch
10        self.eta = eta
11        self.epochs = epochs
12        self.random_seed = random_seed
13
14    def g(self, x):
15        return 1/(1 + np.exp((-x)))
16
17    def g_prime(self, x):
18        return self.g(x) * (1 - self.g(x))
19
20    def fit(self, X, Y):
```


XOR 신경망 모델링

- 학습 정리
 - **XOR** 신경망의 구조를 설계하기
 - **XOR** 신경망 행렬의 형상을 일반화 하기
 - **XOR** 신경망 객체를 코딩하기
- **9-3 XOR** 신경망 구현

10주차(2/3)

XOR 신경망 모델링

파이썬으로 배우는 기계학습

한동대학교
김영섭 교수