행렬의 곱

행렬끼리 곱셈을 하는 경우에는 앞 행렬의 열 개수와 뒤 행렬의 행 개수가 같아야 함.

$$\underset{3\times 1}{\begin{pmatrix} a \\ b \\ c \end{pmatrix}} \bullet \underset{1\times 5}{\begin{pmatrix} A & B & C & D & E \end{pmatrix}} = \underset{3\times 5}{\begin{pmatrix} aA & aB & aC & aD & aE \\ bA & bB & bC & bD & bE \\ cA & cB & cC & cD & cE \end{pmatrix}}$$

Shallow neural network1 - 예제 5-1, 5-2

1) 파라미터 설정

```
def get_parameters():

  beta_0 = np.zeros((3,1));    # formerly theta_x0
  omega_0 = np.zeros((3,1));   # formerly theta_x1
  beta_1 = np.zeros((1,1));    # formerly phi_0
  omega_1 = np.zeros((1,3));   # formerly phi_x

  beta_0[0,0] = 0.3; beta_0[1,0] = -1.0; beta_0[2,0] = -0.5

  omega_0[0,0] = -1.0; omega_0[1,0] = 1.8; omega_0[2,0] = 0.65

  beta_1[0,0] = 0.1

  omega_1[0,0] = -2.0; omega_1[0,1] = -1.0; omega_1[0,2] = 7.0

  return beta_0, omega_0, beta_1, omega_1
```

$$\underset{3\times 1}{\boldsymbol{\beta}_0} = \begin{pmatrix} 0.3 \\ -1.0 \\ -0.5 \end{pmatrix}, \quad \underset{3\times 1}{\boldsymbol{\Omega}_0} = \begin{pmatrix} -1.0 \\ 1.8 \\ 0.65 \end{pmatrix}, \quad \underset{1\times 1}{\boldsymbol{\beta}_1} = \begin{pmatrix} 0.1 \end{pmatrix}, \quad \underset{1\times 3}{\boldsymbol{\Omega}_1} = \begin{pmatrix} -2.0 & -1.0 & 7.0 \end{pmatrix}$$

2) shallow neural network 구축

```
def ReLU(preactivation):

  activation = preactivation.clip(0.0)

  return activation
```

```python
def shallow_nn(x, beta_0, omega_0, beta_1, omega_1):

  x = np.reshape(x,(1, x.size))

  h1 = ReLU(np.matmul(beta_0,np.ones((1, x.size))) + \
       np.matmul(omega_0, x))

  y = np.matmul(beta_1,np.ones((1, x.size))) + np.matmul(omega_1, h1)

  return y
```

- 입력 1

$x = (0.01, 0.02, \cdots, 0.99)^T$ -> reshape -> $\underset{1\times 100}{X} = (0.01\ \ 0.02\ \ \cdots\ \ 0.99)$

- 은닉 유닛 3

$h_1 = a[\beta_0 + \Omega_0 X]$

$$\underset{3\times 1}{\begin{pmatrix} h_1 \\ h_2 \\ h_3 \end{pmatrix}} = a\left[\underset{3\times 100}{\begin{pmatrix} 0.3 & 0.3 & \cdots & 0.3 \\ -1.0 & -1.0 & \cdots & -1.0 \\ -0.5 & -0.5 & \cdots & -0.5 \end{pmatrix}} + \underset{3\times 100}{\begin{pmatrix} -1.0\times 0.01 & -1.0\times 0.02 & \cdots & -1.0\times 0.99 \\ 1.8\times 0.01 & 1.8\times 0.02 & \cdots & 1.8\times 0.99 \\ 0.65\times 0.01 & 0.65\times 0.02 & \cdots & 0.65\times 0.99 \end{pmatrix}}\right]$$

$$\begin{pmatrix} h_1 \\ h_2 \\ h_3 \end{pmatrix} = a\left[\underset{3\times 100}{\begin{pmatrix} 0.3-1.0\times 0.01 & 0.3-1.0\times 0.02 & \cdots & 0.3-1.0\times 0.99 \\ -1.0+1.8\times 0.01 & -1.0+1.8\times 0.02 & \cdots & -1.0+1.8\times 0.99 \\ -0.5+0.65\times 0.01 & -0.5+0.65\times 0.02 & \cdots & -0.5+0.65\times 0.99 \end{pmatrix}}\right]$$

$$\downarrow$$

$$\begin{aligned} h_1 &= a[\theta_{10} + \theta_{11}x] \\ h_2 &= a[\theta_{20} + \theta_{21}x] \\ h_3 &= a[\theta_{30} + \theta_{31}x] \end{aligned}$$

- 출력 1

$y_1 = \beta_1 + \Omega_1 h_1$

$$y = \underset{1\times 100}{(1.0\ \ 1.0 \cdots 1.0)} + \underset{1\times 100}{(-2.0\times 0.29 - 1.0\times(-0.82) + 7.0\times(-0.44)\ \ \cdots)}$$

Shallow neural network2 - 예제 5-3

1) 파라미터 설정

```
def get_parameters():

  beta_0 = np.zeros((3,1));  # formerly theta_x0
  omega_0 = np.zeros((3,1)); # formerly theta_x1
  beta_1 = np.zeros((3,1));  # formerly phi_0
  omega_1 = np.zeros((3,3)); # formerly phi_1

  beta_0[0,0] = 0.3; beta_0[1,0] = -1.0; beta_0[2,0] = -0.5

  omega_0[0,0] = -1.0; omega_0[1,0] = 1.8; omega_0[2,0] = 0.65

  beta_1[0,0] = 2.0; beta_1[1,0] = -2; beta_1[2,0] = 0.0

  omega_1[0,0] = -24.0; omega_1[0,1] = -8.0; omega_1[0,2] = 50.0
  omega_1[1,0] = -2.0; omega_1[1,1] = 8.0; omega_1[1,2] = -30.0
  omega_1[2,0] = 16.0; omega_1[2,1] = -8.0; omega_1[2,2] =-8

  return beta_0, omega_0, beta_1, omega_1
```

$$\underset{3\times 1}{\boldsymbol{\beta}_0} = \begin{pmatrix} 0.3 \\ -1.0 \\ -0.5 \end{pmatrix} \qquad \underset{3\times 1}{\boldsymbol{\Omega}_0} = \begin{pmatrix} -1.0 \\ 1.8 \\ 0.65 \end{pmatrix} \qquad \underset{3\times 1}{\boldsymbol{\beta}_1} = \begin{pmatrix} 2.0 \\ -2.0 \\ 0.0 \end{pmatrix} \qquad \underset{3\times 3}{\boldsymbol{\Omega}_1} = \begin{pmatrix} -24 & -8 & 50 \\ -2 & 8 & -30 \\ 16 & -8 & -8 \end{pmatrix}$$

2) shallow neural network 구축

```
def ReLU(preactivation):

  activation = preactivation.clip(0.0)

  return activation
```

```python
def shallow_nn(x, beta_0, omega_0, beta_1, omega_1):

  n_data = x.size
  x = np.reshape(x,(1,n_data))

  h1 = ReLU(np.matmul(beta_0,np.ones((1,n_data))) \
            + np.matmul(omega_0,x))

  model_out = np.matmul(beta_1,np.ones((1,n_data))) \
              + np.matmul(omega_1,h1)

  return model_out
```

- 입력 1

$x = (0.01, 0.02, \cdots, 0.99)^T$ -> reshape -> $\underset{1\times 100}{X} = (0.01 \;\; 0.02 \;\; \cdots \;\; 0.99)$

- 은닉유닛 3

$h_1 = a[\beta_0 + \Omega_0 X]$

$$\begin{pmatrix} h_1 \\ h_2 \\ h_3 \end{pmatrix} = a\left[\underset{3\times 100}{\begin{pmatrix} 0.3 & 0.3 & \cdots & 0.3 \\ -1.0 & -1.0 & \cdots & -1.0 \\ 0.5 & 0.5 & \cdots & 0.5 \end{pmatrix}} + \underset{3\times 100}{\begin{pmatrix} -1.0\times 0.1 & -1.0\times 0.2 & \cdots & -1.0\times 0.99 \\ 1.8\times 0.1 & 1.8\times 0.2 & \cdots & 1.8\times 0.99 \\ 0.65\times 0.1 & 0.65\times 0.2 & \cdots & 0.65\times 0.99 \end{pmatrix}}\right]$$

$$\begin{pmatrix} h_1 \\ h_2 \\ h_3 \end{pmatrix} = a\left[\underset{3\times 100}{\begin{pmatrix} 0.3-1.0\times 0.1 & 0.3-1.0\times 0.2 & \cdots & 0.3-1.0\times 0.99 \\ -1.0+1.8\times 0.1 & -1.0+1.8\times 0.2 & \cdots & -1.0+1.8\times 0.99 \\ 0.5+0.65\times 0.1 & 0.5+0.65\times 0.2 & \cdots & 0.5+0.65\times 0.99 \end{pmatrix}}\right]$$

- 출력 3

$y_1 = \beta_1 + \Omega_1 h_1$

$$\begin{pmatrix} y_1 \\ y_2 \\ y_3 \end{pmatrix} = \begin{pmatrix} 2.0 \\ -2 \\ 0.0 \end{pmatrix}(1 \;\; 1 \cdots 1) + \begin{pmatrix} -24 & -8 & 50 \\ -2 & 8 & -30 \\ 16 & -8 & -8 \end{pmatrix}\begin{pmatrix} 0.2 & 0.1 & \cdots & -0.69 \\ -0.82 & -0.64 & \cdots & 0.78 \\ 0.57 & 0.63 & \cdots & 1.14 \end{pmatrix}$$

$$\begin{pmatrix} y_1 \\ y_2 \\ y_3 \end{pmatrix} = \underset{3\times 100}{\begin{pmatrix} 2.0 & 2.0 & \cdots & 2.0 \\ -2 & -2 & \cdots & -2 \\ 0.0 & 0.0 & \cdots & 0.0 \end{pmatrix}} + \underset{3\times 100}{\begin{pmatrix} -24\times 0.2-8\times(-0.82)+50\times 0.57 & \cdots \\ -2\times 0.2+8\times(-0.82)-30\times 0.57 & \cdots \\ 16\times 0.2-8\times(-0.82)-8\times 0.57 & \cdots \end{pmatrix}}$$

Shallow neural network2 - 입력층이 두 개인 경우

1) 파라미터 설정

```
def get_parameters1():

  beta_0 = np.zeros((3,1)) # formerly theta_x0
  omega_0 = np.zeros((3,2)) # formerly theta_x1
  beta_1 = np.zeros((1,1)) # formerly phi_0
  omega_1 = np.zeros((1,3)) # formerly phi_1

  beta_0[0,0] = 0.3; beta_0[1,0] = -1.0; beta_0[2,0] = -0.5

  omega_0[0,0] = -1.0; omega_0[1,0] = 1.8; omega_0[2,0] = 0.65
  omega_0[0,1] = 2.0; omega_0[1,1] = -0.2; omega_0[2,1] = 0.3

  beta_1[0,0] = 2.0

  omega_1[0,0] = -24.0; omega_1[0,1] = -8.0; omega_1[0,2] = 50.0

  return beta_0, omega_0, beta_1, omega_1
```

$$\underset{3\times 1}{\boldsymbol{\beta}_0} = \begin{pmatrix} 0.3 \\ -1.0 \\ -0.5 \end{pmatrix} \qquad \underset{3\times 2}{\boldsymbol{\Omega}_0} = \begin{pmatrix} -1.0 & 2.0 \\ 1.8 & -0.2 \\ 0.65 & 0.3 \end{pmatrix} \qquad \underset{1\times 1}{\boldsymbol{\beta}_1} = (2.0) \qquad \underset{3\times 3}{\boldsymbol{\Omega}_1} = \begin{pmatrix} -24 & -8 & 50 \end{pmatrix}$$

2) shallow neural network 구축

```
def ReLU(preactivation):

  activation = preactivation.clip(0.0)

  return activation
```

```python
def shallow_nn(x1, x2, beta_0, omega_0, beta_1, omega_1):

  x1 = np.reshape(x1, (1, x1.size))  #(1, 100)
  x2 = np.reshape(x2, (1, x2.size))  #(1, 100)
  x = np.vstack((x1, x2))  # 주어진 배열들을 수직(세로)   #(2,100)

  h1 = ReLU(np.matmul(beta_0,np.ones((1, x[0].size))) +\
       np.matmul(omega_0, x))

  y = np.matmul(beta_1,np.ones((1, x[0].size))) + np.matmul(omega_1, h1)

return y
```

- 입력 2

$$x_1 = (0.01,\ 0.02,\ \cdots\ ,\ 0.99)^T$$

$$x_2 = (1.01,\ 1.02,\ \cdots\ ,\ 1.99)^T$$

$$\underset{2\times 100}{X} = \begin{pmatrix} 0.01 & 0.02 & \cdots & 0.99 \\ 1.01 & 1.02 & \cdots & 1.99 \end{pmatrix}$$

- 은닉유닛 3

$$h_1 = a[\beta_0 + \Omega_0 X]$$

$$\begin{pmatrix} h_1 \\ h_2 \\ h_3 \end{pmatrix} = a\left[\underbrace{\begin{pmatrix} 0.3 & 0.3 & \cdots & 0.3 \\ -1.0 & -1.0 & \cdots & -1.0 \\ 0.5 & 0.5 & \cdots & 0.5 \end{pmatrix}}_{3\times 100} + \underbrace{\begin{pmatrix} -1.0 & 2.0 \\ 1{,}8 & -0.2 \\ 0.65 & 0.3 \end{pmatrix}}_{3\times 2} \underbrace{\begin{pmatrix} 0.01 & 0.02 \cdots 0.99 \\ 1.01 & 1.02 \cdots 1.99 \end{pmatrix}}_{2\times 100}\right]$$

$$\begin{pmatrix} h_1 \\ h_2 \\ h_3 \end{pmatrix} = a\left[\begin{pmatrix} 0.3 & 0.3 & \cdots & 0.3 \\ -1.0 & -1.0 & \cdots & -1.0 \\ 0.5 & 0.5 & \cdots & 0.5 \end{pmatrix} + \underbrace{\begin{pmatrix} -1.0\times 0.01+2.0\times 1.01 & -1.0\times 0.02+2.0\times 1.02 & \cdots \\ 1{,}8\times 0.01-0.2\times 1.01 & 1{,}8\times 0.02-0.2\times 1.02 & \cdots \\ 0.65\times 0.01+0.3\times 1.01 & 0.65\times 0.02+0.3\times 1.02 & \cdots \end{pmatrix}}_{3\times 100}\right]$$

- 출력 1

$$y = (2.0\ \ 2.0 \cdots 2.0) + (-24\ \ -8\ \ 50)\begin{pmatrix} h_1 \\ h_2 \\ h_3 \end{pmatrix}$$

$$y = \underbrace{(2.0\ \ 2.0 \cdots 2.0)}_{1\times 100} + \underbrace{(-24\ \ -8\ \ 50)}_{1\times 3}\underbrace{\left[\begin{pmatrix} 0.3-1.0\times 0.01+2.0\times 1.01 & \cdots \\ -1.0+1{,}8\times 0.01-0.2\times 1.01 & \cdots \\ 0.5+0.65\times 0.01+0.3\times 1.01 & \cdots \end{pmatrix}\right]}_{3\times 100}$$