AI and Deep Learning

ML 프로그래밍 익숙해지기

Jeju National University
Yung-Cheol Byun

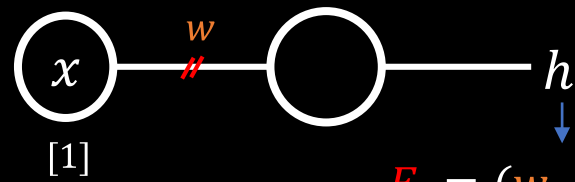
```
w = tf.Variable(tf.random_normal([1,1]))
w = tf.Variable(tf.random_normal([2,1]))
w = tf.Variable(tf.random_normal([2,4]))
```

입력이 1개인 뉴런이 1개 있다.

입력이 2개인 뉴런이 1개 있다.

입력이 2개인 뉴런이 4개 있다.

(1)1-1/L



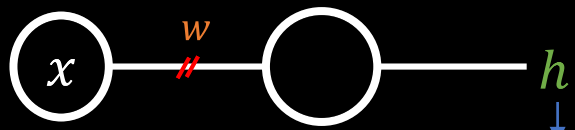
[1]

$$E = (w \cdot x - y)^2$$

```
#----- training data
x_data = [[1]]
y_data = [[1]]

#----- a neuron
w = tf.Variable(tf.random_normal([1,1]))
hypo = w * x_data

#----- learning
cost = (hypo - y_data) ** 2
```

$(1)\ (w)$ $\quad$ $\underbrace{(1 \cdot w)}_{h}$ $\quad$ $(1 \cdot w - 1)^2 \longrightarrow E$ (값 1개)

(3)1-1/L



$[1, 2, 3]$

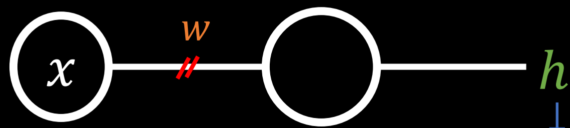
$[1, 2, 3]$

$$E = (w \cdot x - y)^2$$

```
#----- training data
x_data = [[1], [2], [3]]
y_data = [[1], [2], [3]]

#----- a neuron
w = tf.Variable(tf.random_normal([1,1]))
hypo = w * x_data

#----- learning
cost = tf.reduce_mean((hypo - y_data) ** 2)
```

$\rightarrow$ (1) (w) $(1 \cdot w)$ $(1 \cdot w - 1)^2$

(3)1-1/L



$[1, 2, 3]$

$[1, 2, 3]$

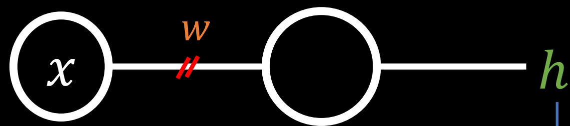
$$E = (w \cdot x - y)^2$$

```
#----- training data
x_data = [[1], [2], [3]]
y_data = [[1], [2], [3]]

#----- a neuron
w = tf.Variable(tf.random_normal([1,1]))
hypo = w * x_data

#----- learning
cost = tf.reduce_mean((hypo - y_data) ** 2)
```

$\rightarrow$ $(1)\ (w)$ $\quad (1 \cdot w) \quad (1 \cdot w - 1)^2 +$

$\rightarrow$ $(2)\ (w)$ $\quad (2 \cdot w) \quad (2 \cdot w - 2)^2$

(3)1-1/L



$[1, 2, 3]$

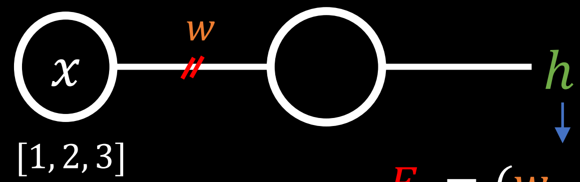
$[1, 2, 3]$

$$E = (w \cdot x - y)^2$$

```
#----- training data
x_data = [[1], [2], [3]]
y_data = [[1], [2], [3]]

#----- a neuron
w = tf.Variable(tf.random_normal([1,1]))
hypo = w * x_data

#----- learning
cost = tf.reduce_mean((hypo - y_data) ** 2)
```

$\rightarrow$	$(1)\ (w)$	$(1 \cdot w)$	$(1 \cdot w - 1)^2 +$
$\rightarrow$	$(2)\ (w)$	$(2 \cdot w)$	$(2 \cdot w - 2)^2 +$
$\rightarrow$	$(3)\ (w)$	$(3 \cdot w)$	$(3 \cdot w - 3)^2$

(3)1-1/L



$$\downarrow \quad [1,2,3]$$

$$E = (w \cdot x - y)^2$$

```
#----- training data
x_data = [[1], [2], [3]]
y_data = [[1], [2], [3]]

#----- a neuron
w = tf.Variable(tf.random_normal([1,1]))
hypo = w * x_data

#----- learning
cost = tf.reduce_mean((hypo - y_data) ** 2)
```

$\rightarrow$ $(1)\ (w)$ $\quad (1 \cdot w) \quad (1 \cdot w - 1)^2 +$

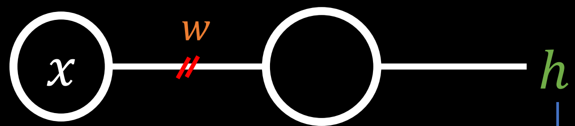
$\rightarrow$ $(2)\ (w)$ $\quad (2 \cdot w) \quad (2 \cdot w - 2)^2 +$

$\rightarrow$ $(3)\ (w)$ $\quad (3 \cdot w) \quad (3 \cdot w - 3)^2$

$$\left.\begin{matrix}(1 \cdot w - 1)^2 + \\ (2 \cdot w - 2)^2 + \\ (3 \cdot w - 3)^2\end{matrix}\right\} \frac{1}{3}\sum \longrightarrow E \text{ (값 1개)}$$

h

(3)1-1/L



[1, 2, 3]

$$E = (w \cdot x - y)^2$$

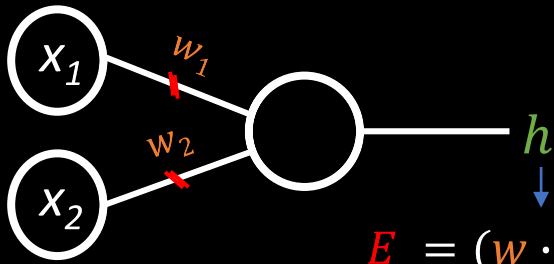
$$\left.\begin{array}{lll} \rightarrow\ (1)\ (w) & (1 \cdot w) & (1 \cdot w - 1)^2 + \\ \rightarrow\ (2)\ (w) & (2 \cdot w) & (2 \cdot w - 2)^2 + \\ \rightarrow\ (3)\ (w) & (3 \cdot w) & (3 \cdot w - 3)^2 \end{array}\right\} \frac{1}{3}\sum \longrightarrow E \text{ (값 1개)}$$

$$\downarrow \begin{pmatrix} 1 \\ 2 \\ 3 \end{pmatrix} (w) \qquad \downarrow \underbrace{\begin{pmatrix} 1 \cdot w \\ 2 \cdot w \\ 3 \cdot w \end{pmatrix}}_{h}$$

```
#----- training data
x_data = [[1], [2], [3]]
y_data = [[1], [2], [3]]

#----- a neuron
w = tf.Variable(tf.random_normal([1,1]))
hypo = w * x_data

#----- learning
cost = tf.reduce_mean((hypo - y_data) ** 2)
```

데이터가 추가되면 아래로 늘어난다.
데이터 수만큼 h도 증가
최종적으로 구해진 E 값은 1개

(1)2-1/L



$[[1,1]]$

$[[1]]$

$$E = (w \cdot x - y)^2$$

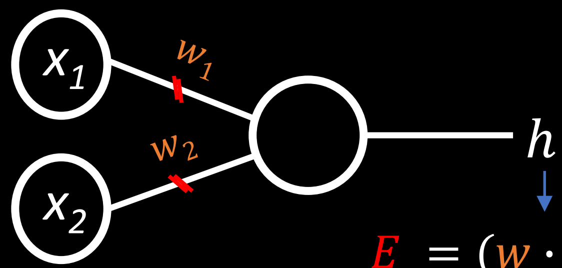
$(1,1)\ (w_1, w_2)$ $\quad (1 \cdot w_1 + 1 \cdot w_2)$ $\quad ((1 \cdot w_1 + 1 \cdot w_2) - 1)^2$

```
#----- training data
x_data = [[1., 1]]
y_data = [[1.]]

#----- a neuron
w = tf.Variable(tf.random_normal([2, 1]))
hypo = tf.matmul(x_data, w)

#----- learning
cost = tf.reduce_mean((hypo - y_data) ** 2)
```

$\cancel{(1,1)} \begin{pmatrix} w_1 \\ w_2 \end{pmatrix}$ $\quad (1 \cdot w_1 + 1 \cdot w_2)$

입력이 늘어나면 입력→, 가중치↓로 늘어난다.

(3)2-1/L



[[1], [2], [3]]

$$E = (w \cdot x - y)^2$$

[[1, 1], [2,2], [3,3]]

```
#----- training data
x_data = [[1., 1], [2, 2], [3, 3]]
y_data = [[1.], [2], [3]]

#----- a neuron
w = tf.Variable(tf.random_normal([2, 1]))
hypo = tf.matmul(x_data, w)

#----- learning
cost = tf.reduce_mean((hypo - y_data) ** 2)
```

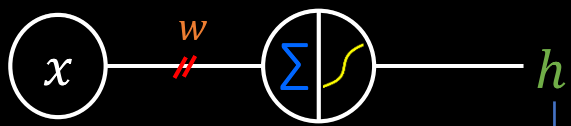
$$\left.\begin{array}{llll} \rightarrow (1,1)\ (w_1, w_2) & (1 \cdot w_1 + 1 \cdot w_2) & ((1 \cdot w_1 + 1 \cdot w_2) - 1)^2 \\ \rightarrow (2,2)\ (w_1, w_2) & (2 \cdot w_1 + 2 \cdot w_2) & ((2 \cdot w_1 + 2 \cdot w_2) - 2)^2 \\ \rightarrow (3,3)\ (w_1, w_2) & (3 \cdot w_1 + 3 \cdot w_2) & ((3 \cdot w_1 + 3 \cdot w_2) - 3)^2 \end{array}\right\} \frac{1}{3}\sum \longrightarrow E$$

$$\begin{pmatrix} 1, 1 \\ 2, 2 \\ 3, 3 \end{pmatrix} \begin{pmatrix} w_1 \\ w_2 \end{pmatrix} \qquad \begin{pmatrix} 1 \cdot w_1 + 1 \cdot w_2 \\ 2 \cdot w_1 + 2 \cdot w_2 \\ 3 \cdot w_1 + 3 \cdot w_2 \end{pmatrix}$$

데이터가 추가되면 아래로 늘어난다.

```
#----- training data
x_data = [[-2.]]
y_data = [[0.]]

#------- a neuron
w = tf.Variable(tf.random_normal([1,1]))
hypo = tf.sigmoid(x_data * w)

#----- learning
cost = -tf.reduce_mean(y_data * tf.log(hypo) +
      tf.subtract(1., y_data) * tf.log(tf.subtract(1., hypo)))
```

(1)1-1/C



$[0]$

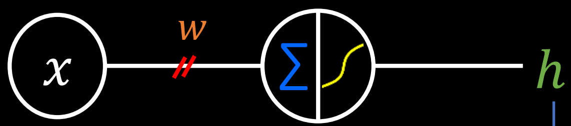
$[-2]$

$$E = -(y\log(h) + (1-y)\log(1-h))$$

$(-2)\ (w)$ $\qquad$ $(-2 \cdot w)$ $\qquad$ $sigmoid(-2 \cdot w) \rightarrow h$

(4)1-1/C

```
#----- training data
x_data = [[-2.], [-1], [1], [2]]
y_data = [[0.], [0], [1], [1]]

#------- a neuron
w = tf.Variable(tf.random_normal([1,1]))
hypo = tf.sigmoid(x_data * w)

#----- learning
cost = -tf.reduce_mean(y_data * tf.log(hypo) +
      tf.subtract(1., y_data) * tf.log(tf.subtract(1., hypo)))
```



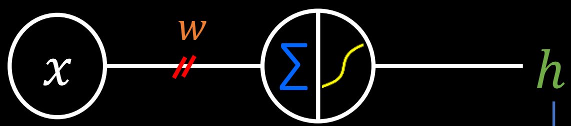
$[0., 0, 1, 1]$

$[-2., -1, 1, 2]$

$$E = -(y\log(h) + (1-y)\log(1-h))$$

$\rightarrow\ (-2)\ (w)$ $\quad (-2\cdot w)$ $\quad sigmoid(-2\cdot w) \rightarrow h$ $\quad E_1$

(4)1-1/C



$[-2., -1, 1, 2]$

$[0., 0, 1, 1]$

$$E = -(y\log(h) + (1-y)\log(1-h))$$

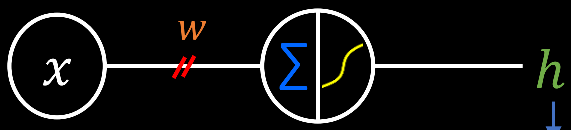
```
#----- training data
x_data = [[-2.], [-1], [1], [2]]
y_data = [[0.], [0], [1], [1]]

#------- a neuron
w = tf.Variable(tf.random_normal([1,1]))
hypo = tf.sigmoid(x_data * w)

#----- learning
cost = -tf.reduce_mean(y_data * tf.log(hypo) +
      tf.subtract(1., y_data) * tf.log(tf.subtract(1., hypo)))
```

$\rightarrow\ (-2)\ (w)$ $\quad (-2\cdot w) \quad sigmoid(-2\cdot w) \rightarrow h \quad E_1$

$\rightarrow\ (-1)\ (w)$ $\quad (-1\cdot w) \quad sigmoid(-1\cdot w) \rightarrow h \quad E_2$

```
#----- training data
x_data = [[-2.], [-1], [1], [2]]
y_data = [[0.], [0], [1], [1]]

#------- a neuron
w = tf.Variable(tf.random_normal([1,1]))
hypo = tf.sigmoid(x_data * w)

#----- learning
cost = -tf.reduce_mean(y_data * tf.log(hypo) +
      tf.subtract(1., y_data) * tf.log(tf.subtract(1., hypo)))
```

(4)1-1/C



$[-2., -1, 1, 2]$

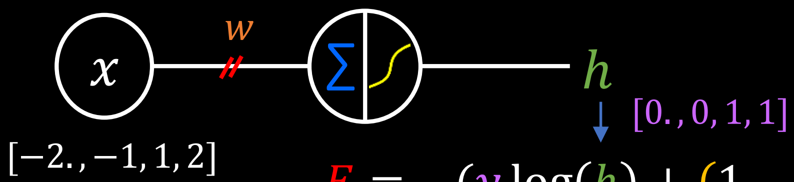
$[0., 0, 1, 1]$

$$E = -(y \log(h) + (1-y)\log(1-h))$$

→ $(-2)\ (w)$	$(-2 \cdot w)$	$sigmoid(-2 \cdot w) \rightarrow h$	E_1
→ $(-1)\ (w)$	$(-1 \cdot w)$	$sigmoid(-1 \cdot w) \rightarrow h$	E_2
→ $(1)\ (w)$	$(1 \cdot w)$	$sigmoid(1 \cdot w) \rightarrow h$	E_3

(4)1-1/C

```
#----- training data
x_data = [[-2.], [-1], [1], [2]]
y_data = [[0.], [0], [1], [1]]

#------- a neuron
w = tf.Variable(tf.random_normal([1,1]))
hypo = tf.sigmoid(x_data * w)

#----- learning
cost = -tf.reduce_mean(y_data * tf.log(hypo) +
       tf.subtract(1., y_data) * tf.log(tf.subtract(1., hypo)))
```



$$E = -(y\log(h) + (1-y)\log(1-h))$$

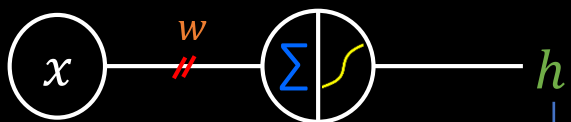
$\rightarrow (-2)\,(w) \qquad (-2\cdot w) \qquad sigmoid(-2\cdot w) \rightarrow h \quad : E_1$

$\rightarrow (-1)\,(w) \qquad (-1\cdot w) \qquad sigmoid(-1\cdot w) \rightarrow h \quad : E_2$

$\rightarrow (1)\,(w) \qquad (1\cdot w) \qquad sigmoid(1\cdot w) \rightarrow h \quad : E_3$

$\rightarrow (2)\,(w) \qquad (2\cdot w) \qquad sigmoid(2\cdot w) \rightarrow h \quad : E_4$

(4)1-1/C



$[-2., -1, 1, 2]$

$\downarrow$ $[0., 0, 1, 1]$

$$E = -(y\log(h) + (1-y)\log(1-h))$$

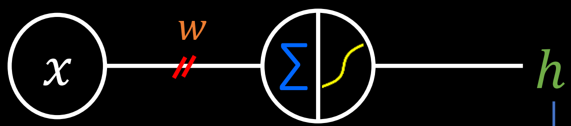
```
#----- training data
x_data = [[-2.], [-1], [1], [2]]
y_data = [[0.], [0], [1], [1]]

#------- a neuron
w = tf.Variable(tf.random_normal([1,1]))
hypo = tf.sigmoid(x_data * w)

#----- learning
cost = -tf.reduce_mean(y_data * tf.log(hypo) +
      tf.subtract(1., y_data) * tf.log(tf.subtract(1., hypo)))
```

$\rightarrow (-2)\,(w)$	$(-2\cdot w)$	$sigmoid(-2\cdot w) \rightarrow h$	$: E_1$
$\rightarrow (-1)\,(w)$	$(-1\cdot w)$	$sigmoid(-1\cdot w) \rightarrow h$	$: E_2$
$\rightarrow (1)\,(w)$	$(1\cdot w)$	$sigmoid(1\cdot w) \rightarrow h$	$: E_3$
$\rightarrow (2)\,(w)$	$(2\cdot w)$	$sigmoid(2\cdot w) \rightarrow h$	$: E_4$

$$\left.\begin{matrix} E_1 \\ E_2 \\ E_3 \\ E_4 \end{matrix}\right\} \frac{1}{4}\sum \rightarrow E$$ (값 1개)

(4)1-1/C

```
#----- training data
x_data = [[-2.], [-1], [1], [2]]
y_data = [[0.], [0], [1], [1]]

#------- a neuron
w = tf.Variable(tf.random_normal([1,1]))
hypo = tf.sigmoid(x_data * w)

#----- learning
cost = -tf.reduce_mean(y_data * tf.log(hypo) +
       tf.subtract(1., y_data) * tf.log(tf.subtract(1., hypo)))
```



$[-2., -1, 1, 2]$

$[0., 0, 1, 1]$

$$E = -(y\log(h) + (1-y)\log(1-h))$$

$\rightarrow\ (-2)\,(w)$ $\quad (-2\cdot w)$ $\quad sigmoid(-2\cdot w) \rightarrow h$ $\quad : E_1$

$\rightarrow\ (-1)\,(w)$ $\quad (-1\cdot w)$ $\quad sigmoid(-1\cdot w) \rightarrow h$ $\quad : E_2$

$\rightarrow\ (1)\,(w)$ $\quad (1\cdot w)$ $\quad sigmoid(1\cdot w) \rightarrow h$ $\quad : E_3$

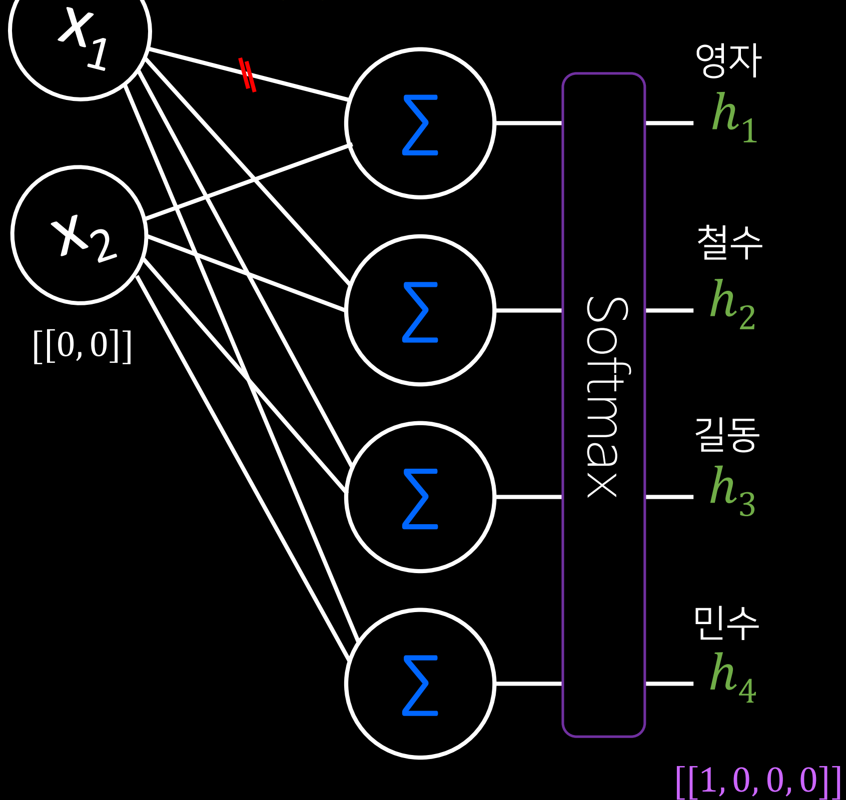
$\rightarrow\ (2)\,(w)$ $\quad (2\cdot w)$ $\quad sigmoid(2\cdot w) \rightarrow h$ $\quad : E_4$

$\left.\begin{matrix} E_1 \\ E_2 \\ E_3 \\ E_4 \end{matrix}\right\} \frac{1}{4}\sum \rightarrow E$ (값 1개)

$$\begin{pmatrix} -2 \\ -1 \\ 1 \\ 2 \end{pmatrix}(w) \qquad \begin{pmatrix} -2\cdot w \\ -1\cdot w \\ 1\cdot w \\ 2\cdot w \end{pmatrix} \qquad \begin{pmatrix} sigmoid(-2\cdot w) \\ sigmoid(-1\cdot w) \\ sigmoid(1\cdot w) \\ sigmoid(2\cdot w) \end{pmatrix}$$

데이터가 추가되면 아래로 늘어난다.

(1)2-4/C

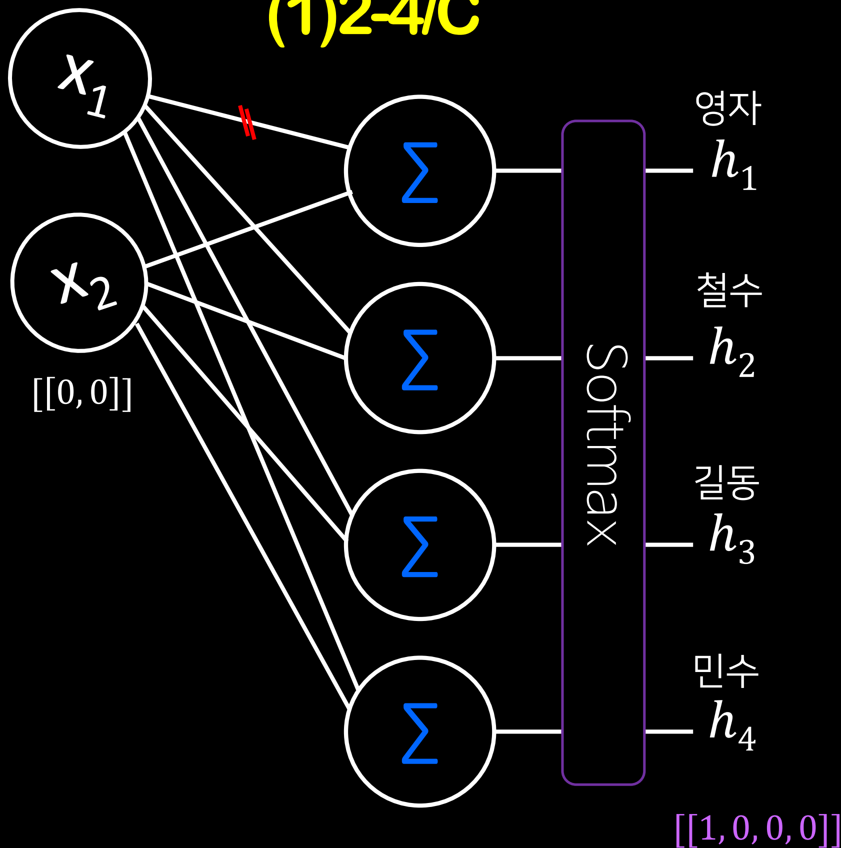



```
#----- training data
x_data = [[0., 0]]
y_data = [[1,0,0,0]]

#------- 2 inputs 4 neurons
W = tf.Variable(tf.random_normal([2, 4]))
b = tf.Variable(tf.random_normal([4]))
output = tf.matmul(x_data, W) + b  # logit (?, 4)

#----- learning
cost = tf.reduce_mean(tf.nn.softmax_cross_
entropy_with_logits(logits=logits, labels=y_data))
```

$$(0, 0) \begin{pmatrix} w_{11}, w_{21}, w_{31}, w_{41} \\ w_{12}, w_{22}, w_{31}, w_{41} \end{pmatrix}$$

신경세포가 추가되면 가중치가 오른쪽으로 늘어난다.

(1)2-4/C

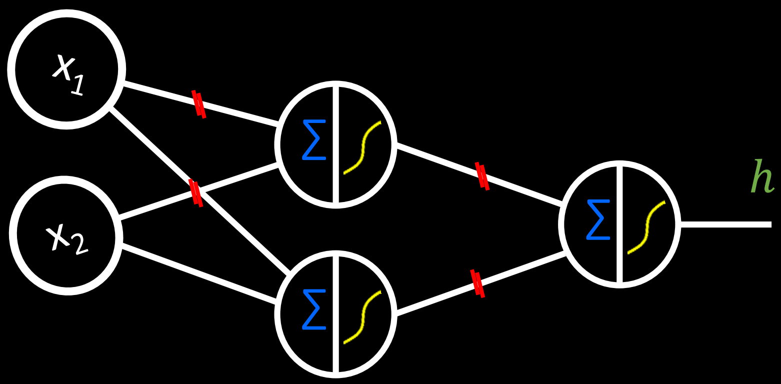



```
#----- training data
x_data = [[0., 0], [0, 1], [1, 0], [1, 1]]
y_data = [[1,0,0,0], [0,1,0,0], [0,0,1,0], [0,0,0,1]]

#------- 2 inputs 4 neurons
W = tf.Variable(tf.random_normal([2, 4]))
b = tf.Variable(tf.random_normal([4]))
output = tf.matmul(x_data, W) + b  # logit (?, 4)

#----- learning
cost = tf.reduce_mean(tf.nn.softmax_cross_
entropy_with_logits(logits=logits, labels=y_data))
```

$$\begin{pmatrix} 0, 0 \\ 0, 1 \\ 1, 0 \\ 1, 1 \end{pmatrix} \begin{pmatrix} w_{11}, w_{21}, w_{31}, w_{41} \\ w_{12}, w_{22}, w_{31}, w_{41} \end{pmatrix}$$

데이터가 추가되면 아래로 늘어난다.

(4)2-2-1/C



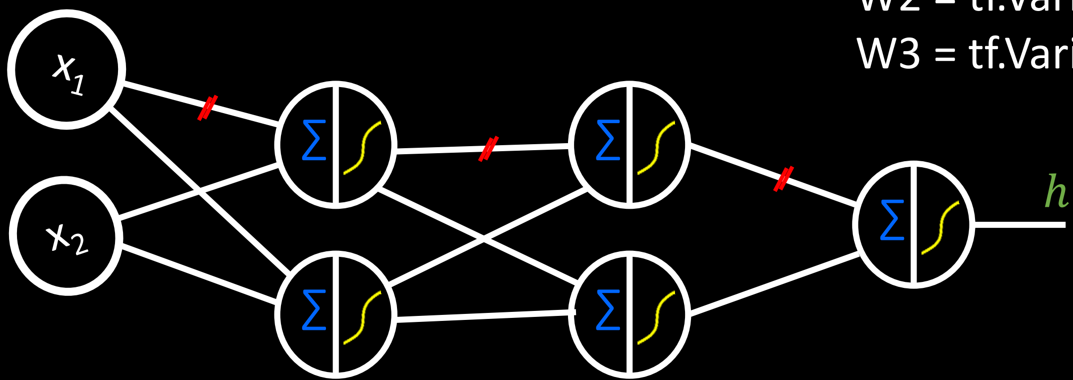
$$\begin{pmatrix} 0,0 \\ 0,1 \\ 1,0 \\ 1,1 \end{pmatrix} \quad \begin{pmatrix} w_{11}^1, w_{21}^1 \\ w_{12}^1, w_{22}^1 \end{pmatrix} \quad \begin{pmatrix} w_1^2 \\ w_1^2 \end{pmatrix} \longrightarrow h$$

$$(4 \times 2) \quad (2 \times 2) \quad (2 \times 1) \quad (4 \times 1)$$

```
#----- training data
x_data = [[0., 0], [0, 1], [1, 0], [1, 1]]
y_data = [[0], [1], [1], [0]]

#------- 2 neurons + 1 neuron
W1 = tf.Variable(tf.random_normal([2, 2]))
b1 = tf.Variable(tf.random_normal([2]))
output1 = tf.sigmoid(tf.matmul(x_data, W1) + b1)

W2 = tf.Variable(tf.random_normal([2, 1]))
b2 = tf.Variable(tf.random_normal([1]))
hypo = tf.sigmoid(tf.matmul(output1, W2) + b2)

#----- learning
cost = -tf.reduce_mean(y_data * tf.log(hypo) +
        tf.subtract(1., y_data) * tf.log(tf.subtract(1., hypo)))
```

(4)2-2-2-1/C

```
x_data = [[0., 0], [0, 1], [1, 0], [1, 1]]
y_data = [[0], [1], [1], [0]]
W1 = tf.Variable(tf.random_normal([2, 2]))
W2 = tf.Variable(tf.random_normal([2, 2]))
W3 = tf.Variable(tf.random_normal([2, 1]))
```



$$\begin{pmatrix} 0,0 \\ 0,1 \\ 1,0 \\ 1,1 \end{pmatrix} \quad \begin{pmatrix} w_{11}^1, w_{21}^1 \\ w_{12}^1, w_{22}^1 \end{pmatrix} \quad \begin{pmatrix} w_{11}^2, w_{21}^2 \\ w_{12}^2, w_{22}^2 \end{pmatrix} \quad \begin{pmatrix} w_1^3 \\ w_1^3 \end{pmatrix} \quad \longrightarrow h$$

$$(4 \times 2) \quad (2 \times 2) \quad (2 \times 2) \quad (2 \times 1) \quad (4 \times 1)$$

(n)784-10/C

(n)784-256-256-10/C

(n)784-512-512-512-512-10/C