

AI and Deep Learning

Being familiar with ML programming

Jeju National University

Yung-Cheol Byun

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

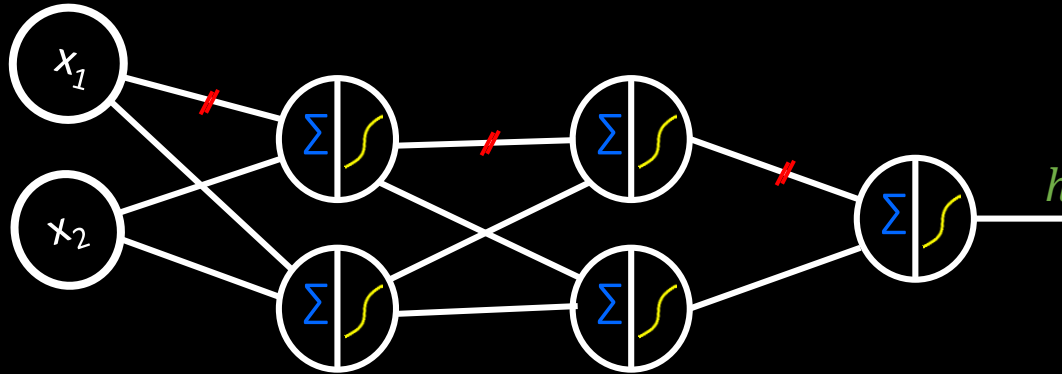
```
w = tf.Variable(tf.random_normal([2,1]))
```

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

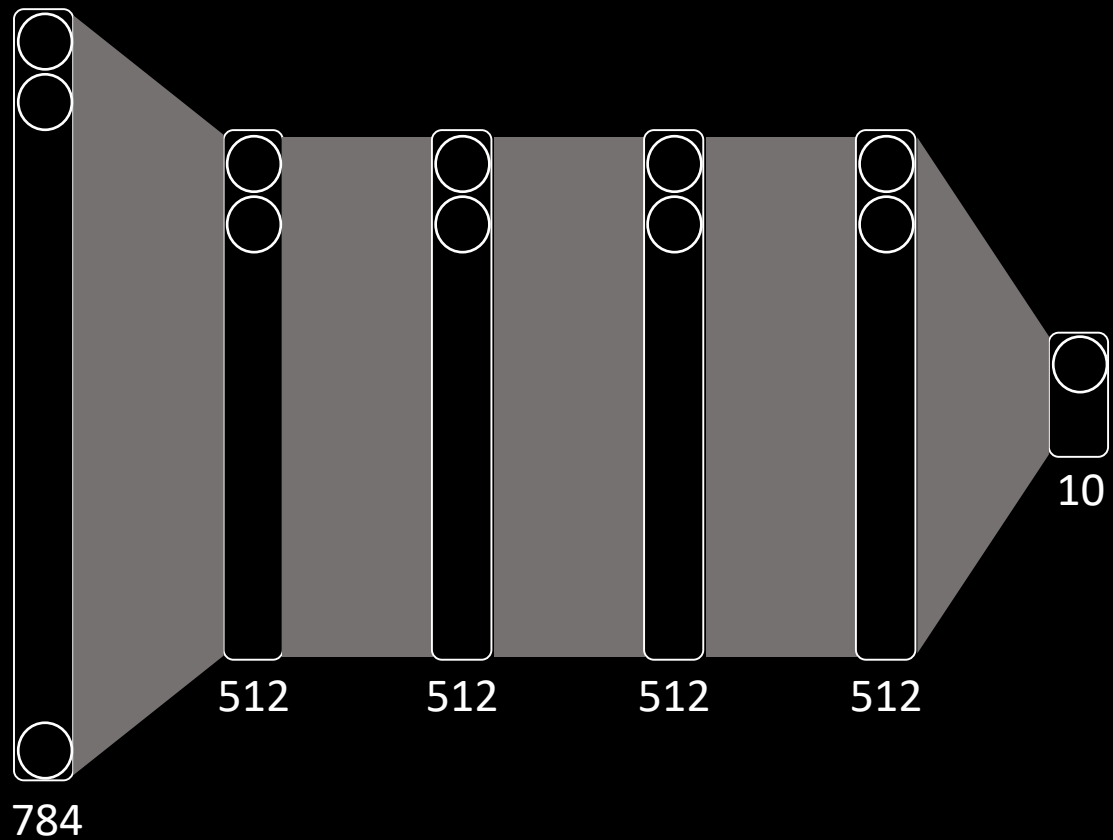
A neuron having 1 input

A neuron having 2 inputs

Four neurons where each one has 2 inputs



```
w = tf.Variable(tf.random_normal([?, ?]))
```

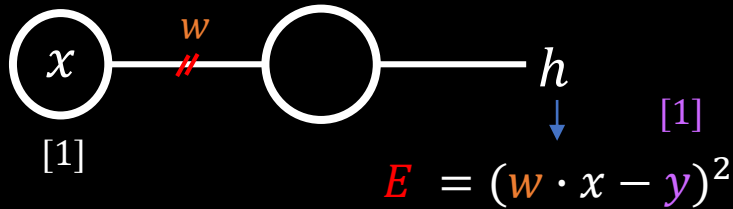


```
w = tf.Variable(tf.random_normal([?, ?]))
```

Matrix Notification

- Input data
- Weights (in synapses)
- Answer(s) by a neuron (hypothesis)
- Error/Loss *E*

(1) 1-1/L



(1) (w)

(1 · w)

(1 · w - 1)² → $E(cost)$ (1 value)

#---- training data

```
x_data = [[1]]
```

```
y_data = [[1]]
```

#---- a neuron

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

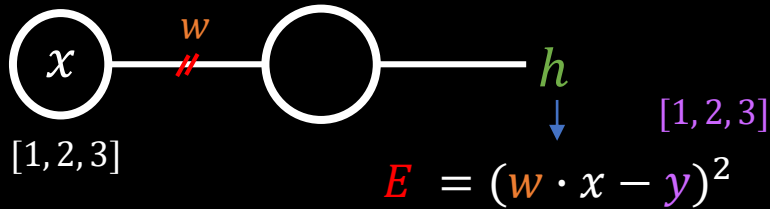
```
hypo = w * x_data
```

#---- learning

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

1 input data → 1 answer by the neuron

(3) 1-1/L



→ (1) (w)

(1 · w)

(1 · w - 1)²

```
#---- training data
```

```
x_data = [[1], [2], [3]]
```

```
y_data = [[1], [2], [3]]
```

```
#---- a neuron
```

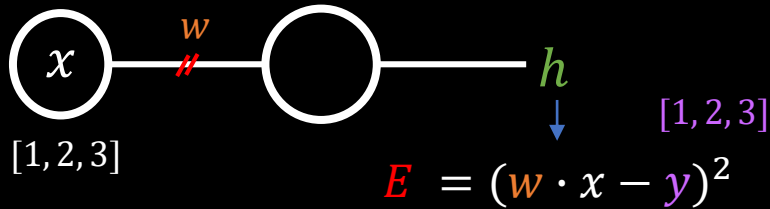
```
w = tf.Variable(tf.random_normal([1,1]))
```

```
hypo = w * x_data
```

```
#---- learning
```

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

(3) 1-1/L



→ (1) (w)
 → (2) (w)

($1 \cdot w$)
 ($2 \cdot w$)

($1 \cdot w - 1$)² +
 ($2 \cdot w - 2$)²

#----- training data

$x_data = [[1], [2], [3]]$

$y_data = [[1], [2], [3]]$

#----- a neuron

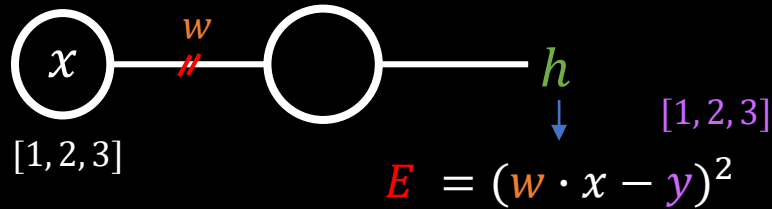
$w = \text{tf.Variable}(\text{tf.random_normal}([1, 1]))$

$\text{hypo} = w * x_data$

#----- learning

$\text{cost} = \text{tf.reduce_mean}((\text{hypo} - y_data) ** 2)$

(3) 1-1/L



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

#----- training data

$x_data = [[1], [2], [3]]$

$y_data = [[1], [2], [3]]$

#----- a neuron

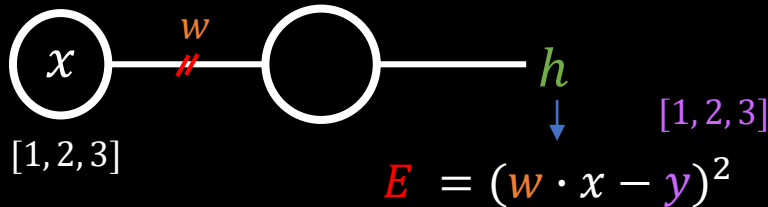
$w = \text{tf.Variable}(\text{tf.random_normal}([1, 1]))$

$\text{hypo} = w * x_data$

#----- learning

$\text{cost} = \text{tf.reduce_mean}((\text{hypo} - y_data) ** 2)$

(3)1-1/L



- (1) (w)
- (2) (w)
- (3) (w)

- (1 · w)
- (2 · w)
- (3 · w)

- (1 · w - 1)² +
- (2 · w - 2)² +
- (3 · w - 3)²

3 input data → 3 answers by the neuron (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
```

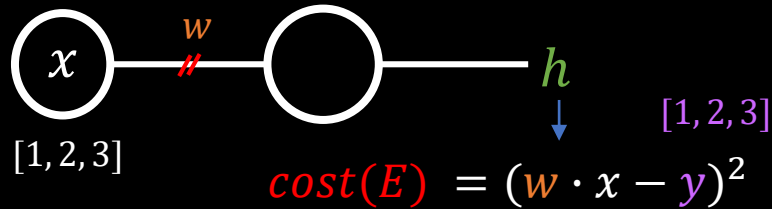
```
#---- learning
```

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

Number of input data

Number of answer/hypothesis
of a neuron(network)

(3) 1-1/L



→ (1) (w)
 → (2) (w)
 → (3) (w)

(1 · w)
 (2 · w)
 (3 · w)

$(1 \cdot w - 1)^2 +$
 $(2 \cdot w - 2)^2 +$
 $(3 \cdot w - 3)^2$ } $\frac{1}{3} \Sigma \rightarrow E$ (1 value)

#----- training data

$x_data = [[1], [2], [3]]$

$y_data = [[1], [2], [3]]$

#----- a neuron

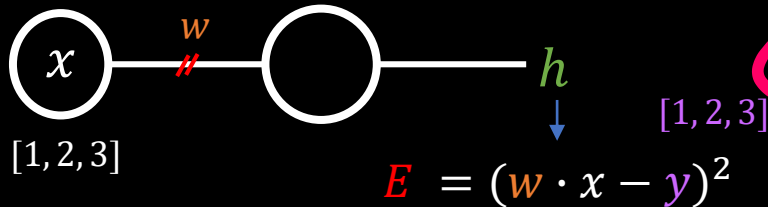
$w = \text{tf.Variable}(\text{tf.random_normal}([1, 1]))$

$\text{hypo} = w * x_data$

#----- learning

$\text{cost} = \text{tf.reduce_mean}((\text{hypo} - y_data) ** 2)$

(3) 1-1/L



#---- training data

$x_data = [[1], [2], [3]]$

$y_data = [[1], [2], [3]]$

#---- a neuron

$w = \text{tf.Variable}(\text{tf.random_normal}([1, 1]))$

$\text{hypo} = w * x_data$

#---- learning

$\text{cost} = \text{tf.reduce_mean}((\text{hypo} - y_data) ** 2)$

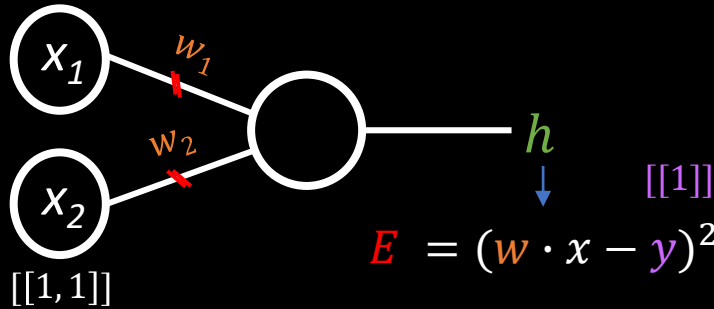
$$\begin{array}{l} \rightarrow (1) (w) \\ \rightarrow (2) (w) \\ \rightarrow (3) (w) \end{array} \quad \begin{array}{l} (1 \cdot w) \\ (2 \cdot w) \\ (3 \cdot w) \end{array} \quad \left. \begin{array}{l} (1 \cdot w - 1)^2 + \\ (2 \cdot w - 2)^2 + \\ (3 \cdot w - 3)^2 \end{array} \right\} \frac{1}{3} \Sigma \rightarrow E \text{ (1 value)}$$

$$\downarrow \begin{pmatrix} 1 \\ 2 \\ 3 \end{pmatrix} (w)$$

$$\downarrow \begin{pmatrix} 1 \cdot w \\ 2 \cdot w \\ 3 \cdot w \end{pmatrix} h$$

more data, increases downward
the same number of hypothesis
but E is a single value.

(1)2-1/L



$(1, 1) (w_1, w_2)$

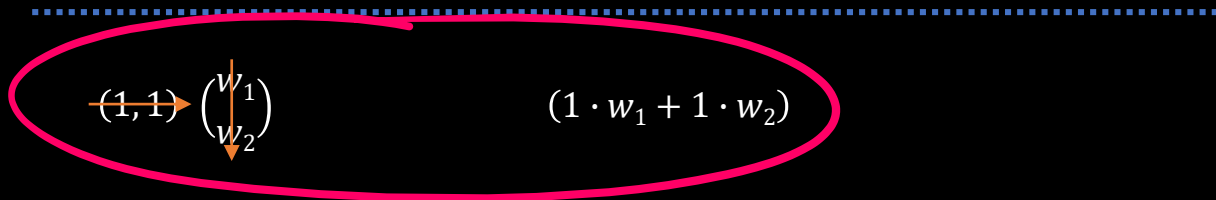
$(1 \cdot w_1 + 1 \cdot w_2)$

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

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

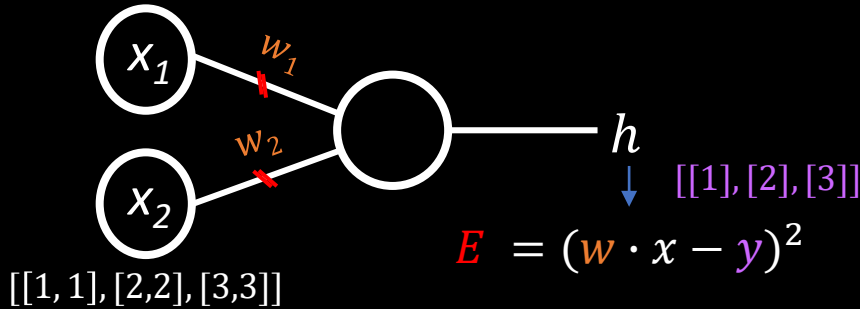
#---- a neuron
w = tf.Variable(tf.random_normal([2, 1]))
hypo = tf.matmul(x_data, w)

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



if we add more input, weight will also increase.

(3)2-1/L



#---- 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)

$$\begin{array}{l}
 \rightarrow (1, 1) (w_1, w_2) \\
 \rightarrow (2, 2) (w_1, w_2) \\
 \rightarrow (3, 3) (w_1, w_2)
 \end{array}
 \begin{array}{l}
 (1 \cdot w_1 + 1 \cdot w_2) \\
 (2 \cdot w_1 + 2 \cdot w_2) \\
 (3 \cdot w_1 + 3 \cdot w_2)
 \end{array}
 \begin{array}{l}
 ((1 \cdot w_1 + 1 \cdot w_2) - 1)^2 \\
 ((2 \cdot w_1 + 2 \cdot w_2) - 2)^2 \\
 ((3 \cdot w_1 + 3 \cdot w_2) - 3)^2
 \end{array}
 \left. \vphantom{\begin{array}{l} \rightarrow (1, 1) (w_1, w_2) \\ \rightarrow (2, 2) (w_1, w_2) \\ \rightarrow (3, 3) (w_1, w_2) \end{array}} \right\} \frac{1}{3} \Sigma \rightarrow E$$

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

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

More input data, more row in the matrix,
the same number of hypothesis
however, E is a single value.

Draw a neuron for the below operation.

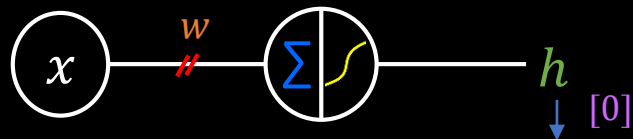
$$\begin{pmatrix} 1, 1 \\ 2, 2 \\ 3, 3 \end{pmatrix} \begin{pmatrix} w_1 \\ w_2 \end{pmatrix}$$

Add 1 more data.

Add 1 more input.

Add 1 more neuron.

(1)1-1/C



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

$(-2) (w)$

$$\text{sigmoid}(-2 \cdot w) \rightarrow h \quad E$$

#---- training data

`x_data = [[-2.]]`

`y_data = [[0.]]`

#----- a neuron

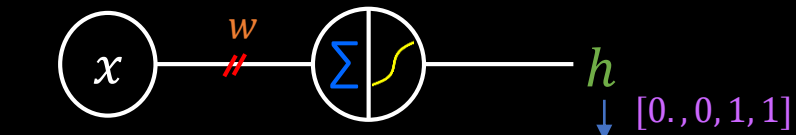
`w = tf.Variable(tf.random_normal([1,1]))`

`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)))`

(4)1-1/C



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

→ (-2) (w)

$\text{sigmoid}(-2 \cdot w) \rightarrow h : E_1$

#---- training data

x_data = [[-2.], [-1], [1], [2]]

y_data = [[0.], [0], [1], [1]]

#----- a neuron

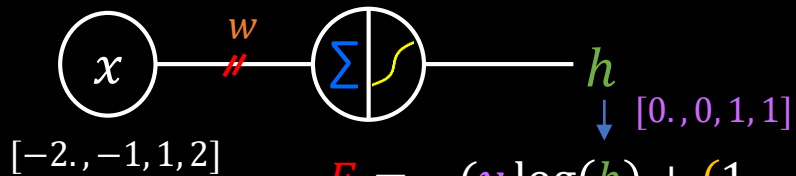
w = tf.Variable(tf.random_normal([1,1]))

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



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

→ $(-2) (w)$

$\text{sigmoid}(-2 \cdot w) \rightarrow h : E_1$

→ $(-1) (w)$

$\text{sigmoid}(-1 \cdot w) \rightarrow h : E_2$

#---- training data

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

$y_data = [[0.], [0], [1], [1]]$

#----- a neuron

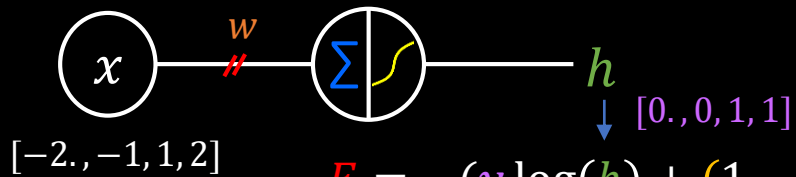
$w = \text{tf.Variable}(\text{tf.random_normal}([1, 1]))$

$\text{hypo} = \text{tf.sigmoid}(x_data * w)$

#---- learning

$\text{cost} = -\text{tf.reduce_mean}(y_data * \text{tf.log}(\text{hypo}) + \text{tf.subtract}(1., y_data) * \text{tf.log}(\text{tf.subtract}(1., \text{hypo})))$

(4)1-1/C



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

→ $(-2) (w)$

→ $(-1) (w)$

→ $(1) (w)$

$\text{sigmoid}(-2 \cdot w) \rightarrow h$: E_1

$\text{sigmoid}(-1 \cdot w) \rightarrow h$: E_2

$\text{sigmoid}(1 \cdot w) \rightarrow h$: E_3

#---- training data

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

$y_data = [[0.], [0], [1], [1]]$

#----- a neuron

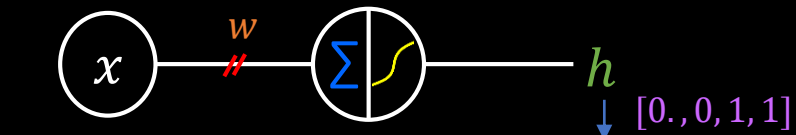
$w = \text{tf.Variable}(\text{tf.random_normal}([1, 1]))$

$\text{hypo} = \text{tf.sigmoid}(x_data * w)$

#---- learning

$\text{cost} = -\text{tf.reduce_mean}(y_data * \text{tf.log}(\text{hypo}) + \text{tf.subtract}(1., y_data) * \text{tf.log}(\text{tf.subtract}(1., \text{hypo})))$

(4)1-1/C



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

- | | | |
|------------|--|---------|
| → (-2) (w) | $\text{sigmoid}(-2 \cdot w) \rightarrow h$ | : E_1 |
| → (-1) (w) | $\text{sigmoid}(-1 \cdot w) \rightarrow h$ | : E_2 |
| → (1) (w) | $\text{sigmoid}(1 \cdot w) \rightarrow h$ | : E_3 |
| → (2) (w) | $\text{sigmoid}(2 \cdot w) \rightarrow h$ | : E_4 |

#---- training data

x_data = [[-2.], [-1], [1], [2]]

y_data = [[0.], [0], [1], [1]]

#----- a neuron

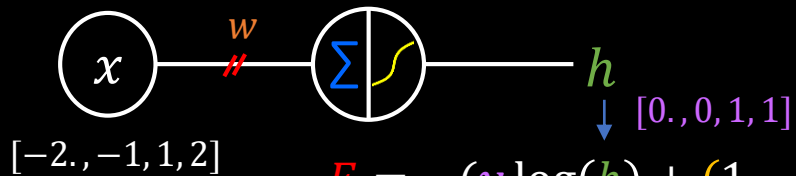
w = tf.Variable(tf.random_normal([1,1]))

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



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

- $(-2) (w)$
- $(-1) (w)$
- $(1) (w)$
- $(2) (w)$

$$\left. \begin{array}{l} \text{sigmoid}(-2 \cdot w) \rightarrow h \\ \text{sigmoid}(-1 \cdot w) \rightarrow h \\ \text{sigmoid}(1 \cdot w) \rightarrow h \\ \text{sigmoid}(2 \cdot w) \rightarrow h \end{array} \right\} \begin{array}{l} : E_1 \\ : E_2 \\ : E_3 \\ : E_4 \end{array} \left. \vphantom{\begin{array}{l} \text{sigmoid}(-2 \cdot w) \\ \text{sigmoid}(-1 \cdot w) \\ \text{sigmoid}(1 \cdot w) \\ \text{sigmoid}(2 \cdot w) \end{array}} \right\} \frac{1}{4} \sum \rightarrow E \text{ (1 value)}$$

#---- training data

x_data = [[-2.], [-1], [1], [2]]

y_data = [[0.], [0], [1], [1]]

#----- a neuron

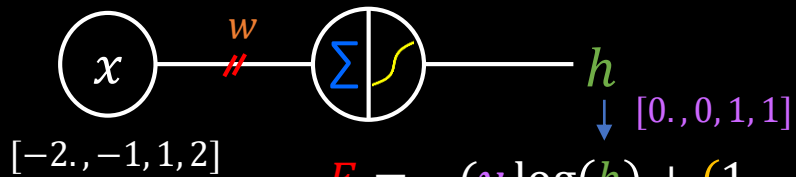
w = tf.Variable(tf.random_normal([1,1]))

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



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

- $(-2) (w)$
- $(-1) (w)$
- $(1) (w)$
- $(2) (w)$

$$\left. \begin{array}{l} \text{sigmoid}(-2 \cdot w) \rightarrow h : E_1 \\ \text{sigmoid}(-1 \cdot w) \rightarrow h : E_2 \\ \text{sigmoid}(1 \cdot w) \rightarrow h : E_3 \\ \text{sigmoid}(2 \cdot w) \rightarrow h : E_4 \end{array} \right\} \frac{1}{4} \sum \rightarrow E \text{ (1 value)}$$

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

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

#---- training data

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

$y_data = [[0.], [0], [1], [1]]$

#----- a neuron

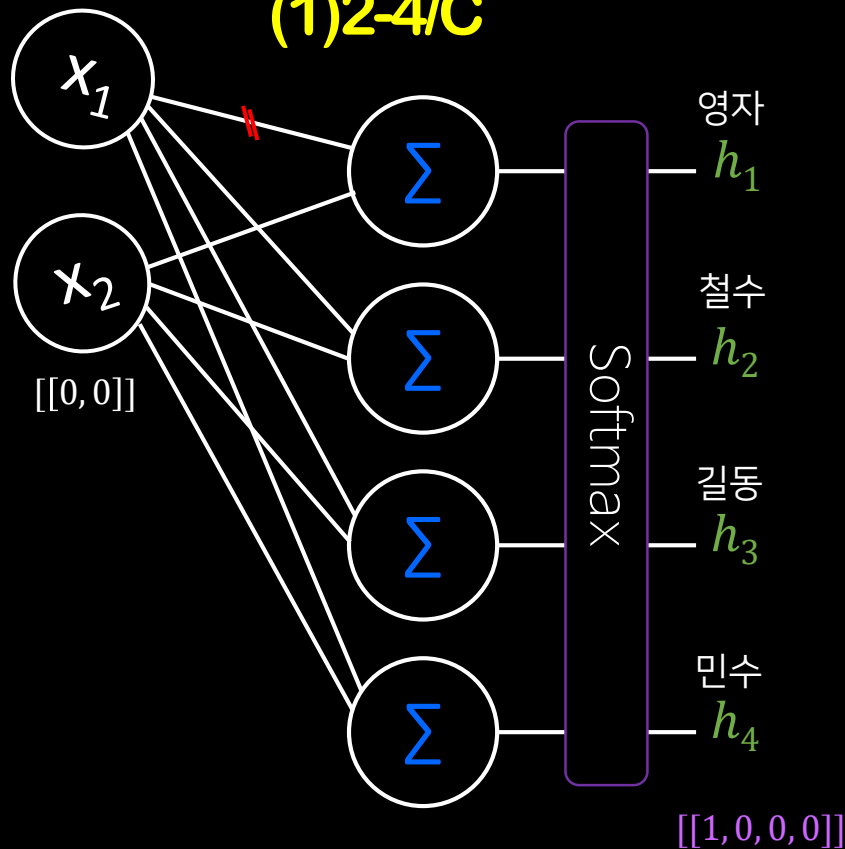
$w = \text{tf.Variable}(\text{tf.random_normal}([1, 1]))$

$\text{hypo} = \text{tf.sigmoid}(x_data * w)$

#---- learning

$\text{cost} = -\text{tf.reduce_mean}(y_data * \text{tf.log}(\text{hypo}) + \text{tf.subtract}(1., y_data) * \text{tf.log}(\text{tf.subtract}(1., \text{hypo})))$

(1)2-4/C



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

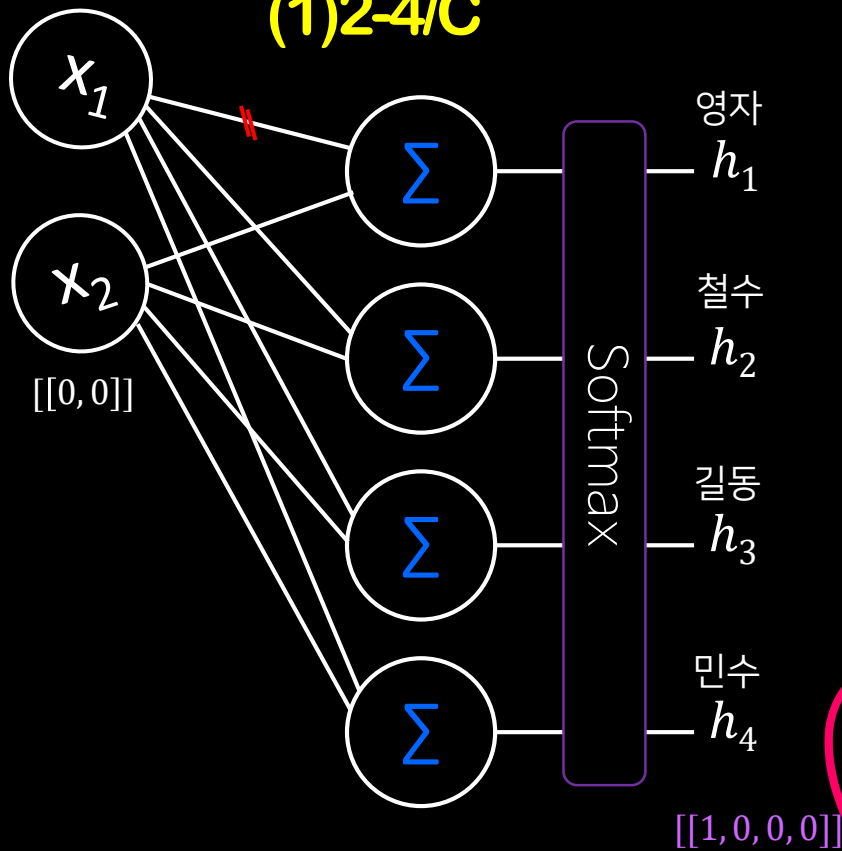
#----- 2 inputs 4 neurons
W = tf.Variable(tf.random_normal([2, 4]))
output = tf.matmul(x_data, W) # logit (?, 4)

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

$$(0, 0) \begin{pmatrix} w_{11} & w_{21} & w_{31} & w_{41} \\ w_{12} & w_{22} & w_{32} & w_{42} \end{pmatrix}$$

if we add 1 more neuron,
then, 1 more column in weight matrix.

(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]]

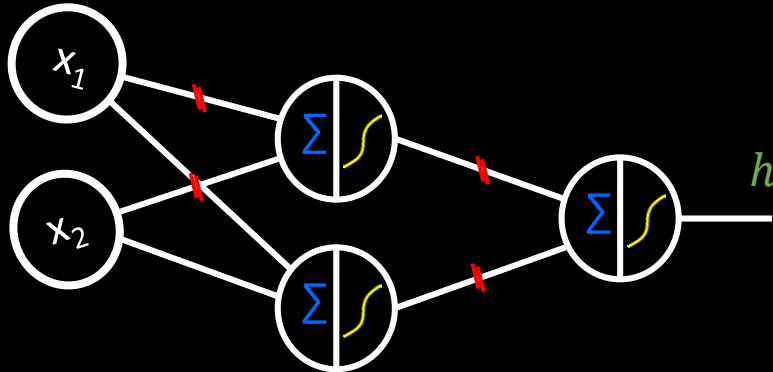
#----- 2 inputs 4 neurons
W = tf.Variable(tf.random_normal([2, 4]))
output = tf.matmul(x_data, W) # logit (?, 4)

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

$$\begin{pmatrix} 0, 0 \\ 0, 1 \\ 1, 0 \\ 1, 1 \end{pmatrix} \begin{pmatrix} w_{11}, w_{21}, w_{31}, w_{41} \\ w_{12}, w_{22}, w_{32}, w_{42} \end{pmatrix}$$

Add 3 more data instances.

(4)2-2-1/C



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

(4×2) (2×2) (2×1) (4×1)

#---- training data

$x_data = [[0., 0], [0, 1], [1, 0], [1, 1]]$

$y_data = [[0], [1], [1], [0]]$

#----- 2 neurons + 1 neuron

$W1 = \text{tf.Variable}(\text{tf.random_normal}([2, 2]))$

$b1 = \text{tf.Variable}(\text{tf.random_normal}([2]))$

$\text{output1} = \text{tf.sigmoid}(\text{tf.matmul}(x_data, W1) + b1)$

$W2 = \text{tf.Variable}(\text{tf.random_normal}([2, 1]))$

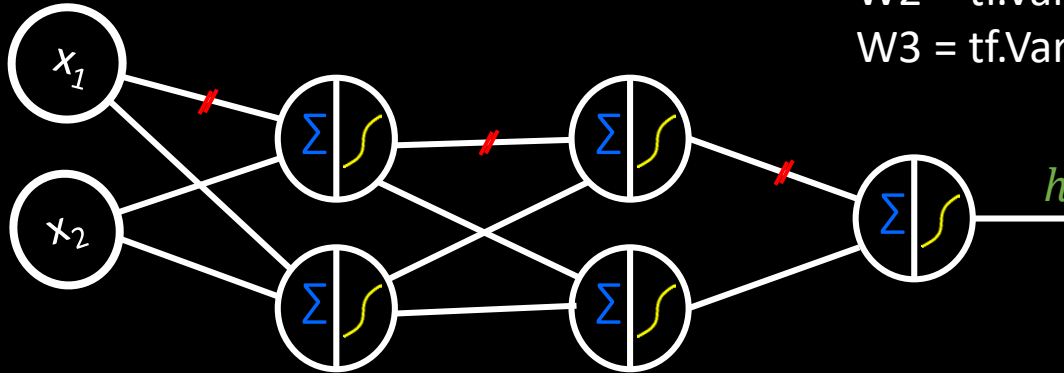
$b2 = \text{tf.Variable}(\text{tf.random_normal}([1]))$

$\text{hypo} = \text{tf.sigmoid}(\text{tf.matmul}(\text{output1}, W2) + b2)$

#---- learning

$\text{cost} = -\text{tf.reduce_mean}(y_data * \text{tf.log}(\text{hypo}) + \text{tf.subtract}(1., y_data) * \text{tf.log}(\text{tf.subtract}(1., \text{hypo})))$

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