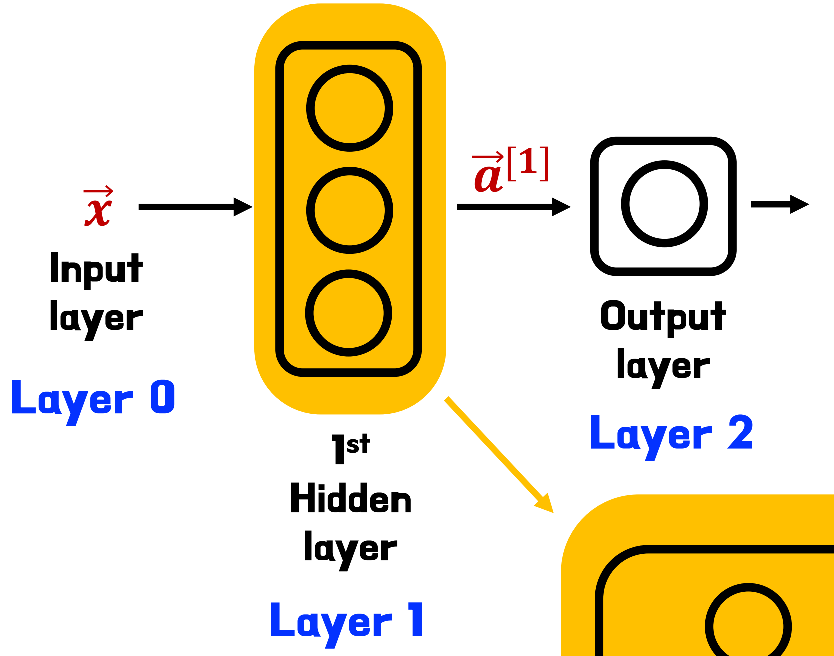
Machine Learning 06

Kihyun Shin
DMSE, HBNU

TheMAD

Neural network layer

Neural network layer



Notation for layer numbering

$$g(z) = \frac{1}{1 + e^{-(z)}}$$

$\vec{x}$

$$\begin{bmatrix} 197 \\ 184 \\ 136 \\ 214 \end{bmatrix}$$

$\vec{w}_1^{[1]}, b_1^{[1]}$

$\vec{w}_2^{[1]}, b_2^{[1]}$

$\vec{w}_3^{[1]}, b_3^{[1]}$

$$\vec{a}_1^{[1]} = g(\vec{w}_1^{[1]} \cdot \vec{x} + b_1^{[1]})$$

$$\vec{a}_2^{[1]} = g(\underbrace{\vec{w}_2^{[1]} \cdot \vec{x} + b_2^{[1]}}_{z})$$

$$\vec{a}_3^{[1]} = g(\vec{w}_3^{[1]} \cdot \vec{x} + b_3^{[1]})$$

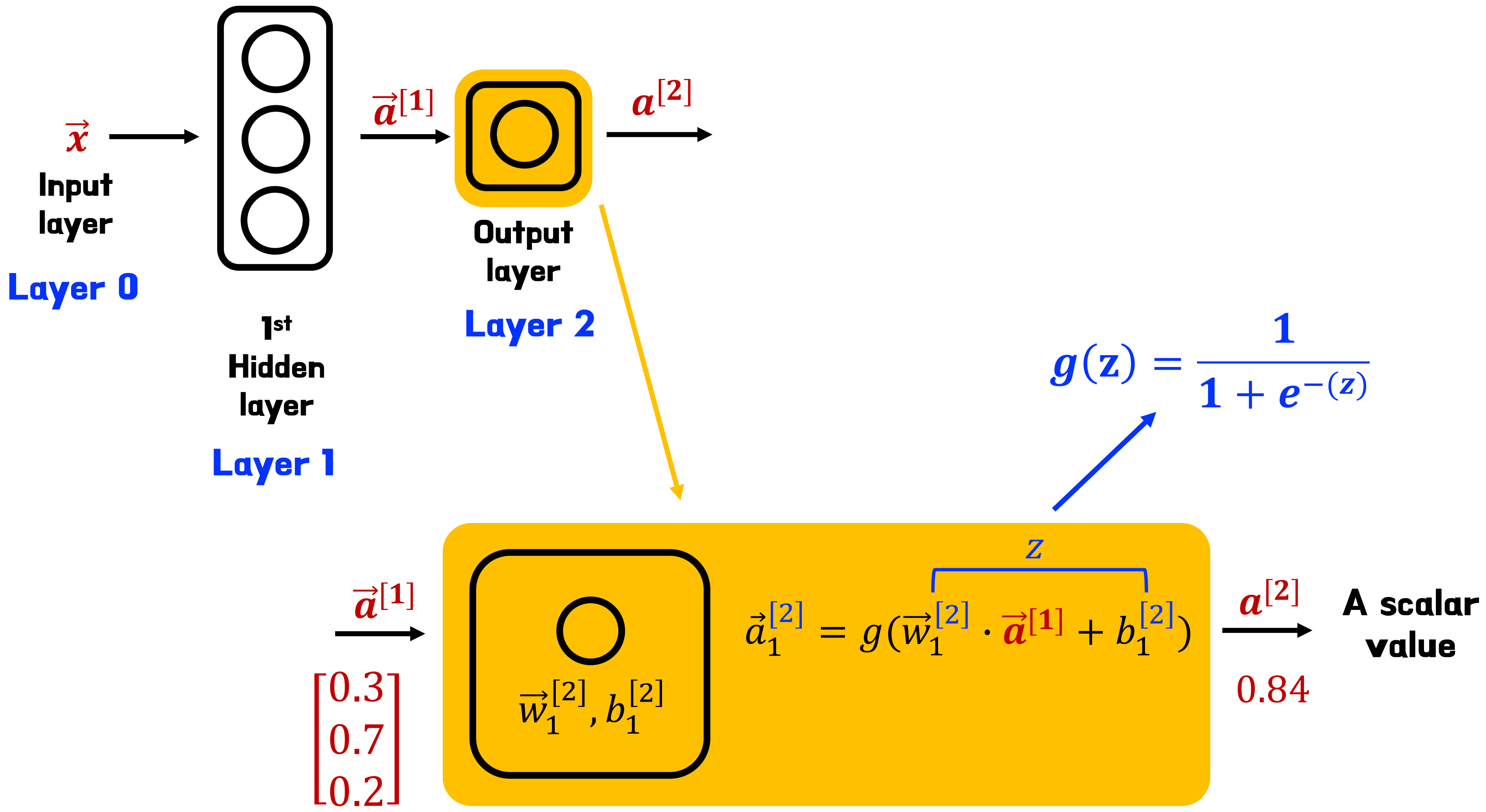
$\vec{a}^{[1]}$

$$\begin{bmatrix} 0.3 \\ 0.7 \\ 0.2 \end{bmatrix}$$

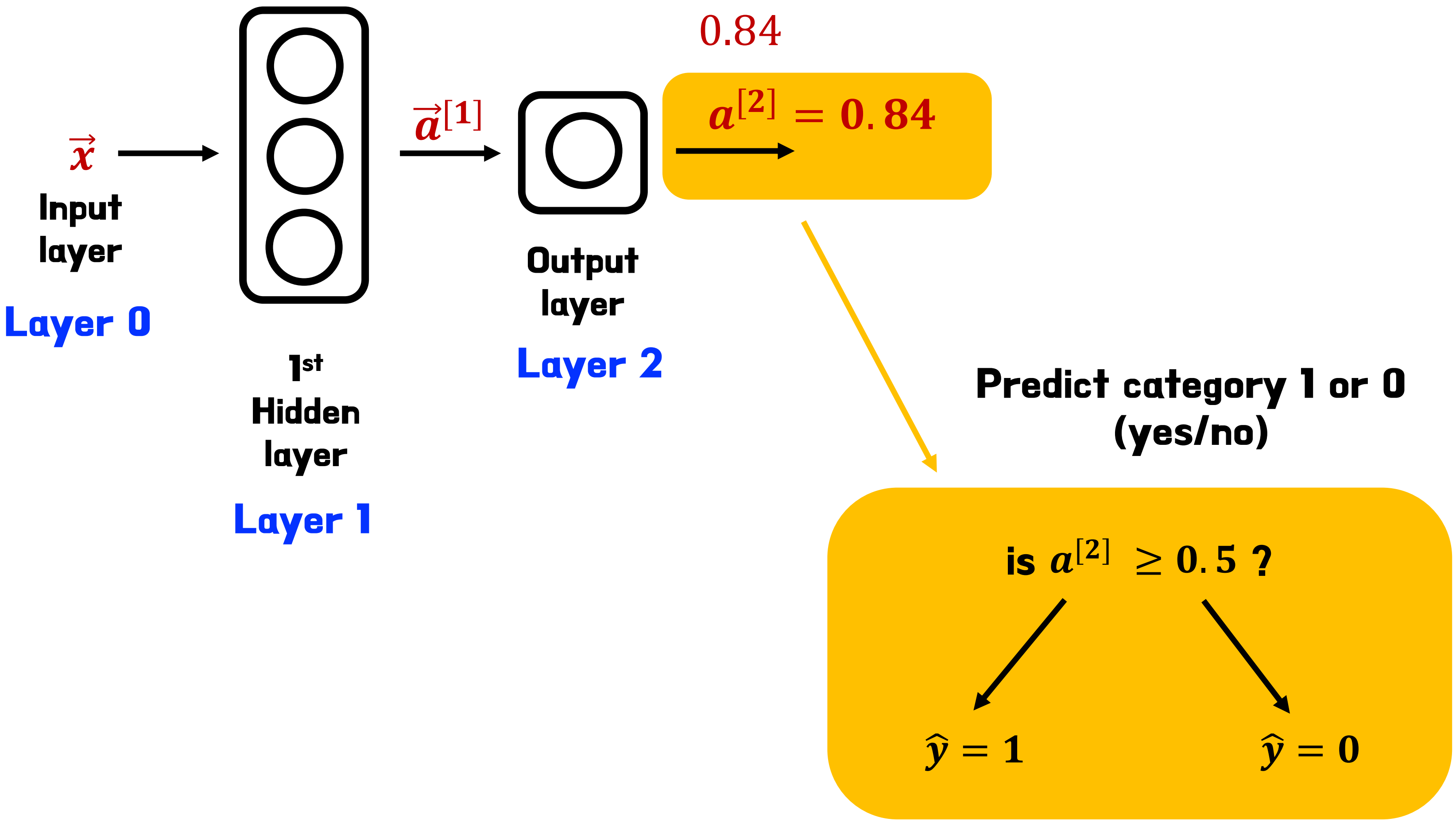
Vector of Activation values from layer [1]

TheMAD

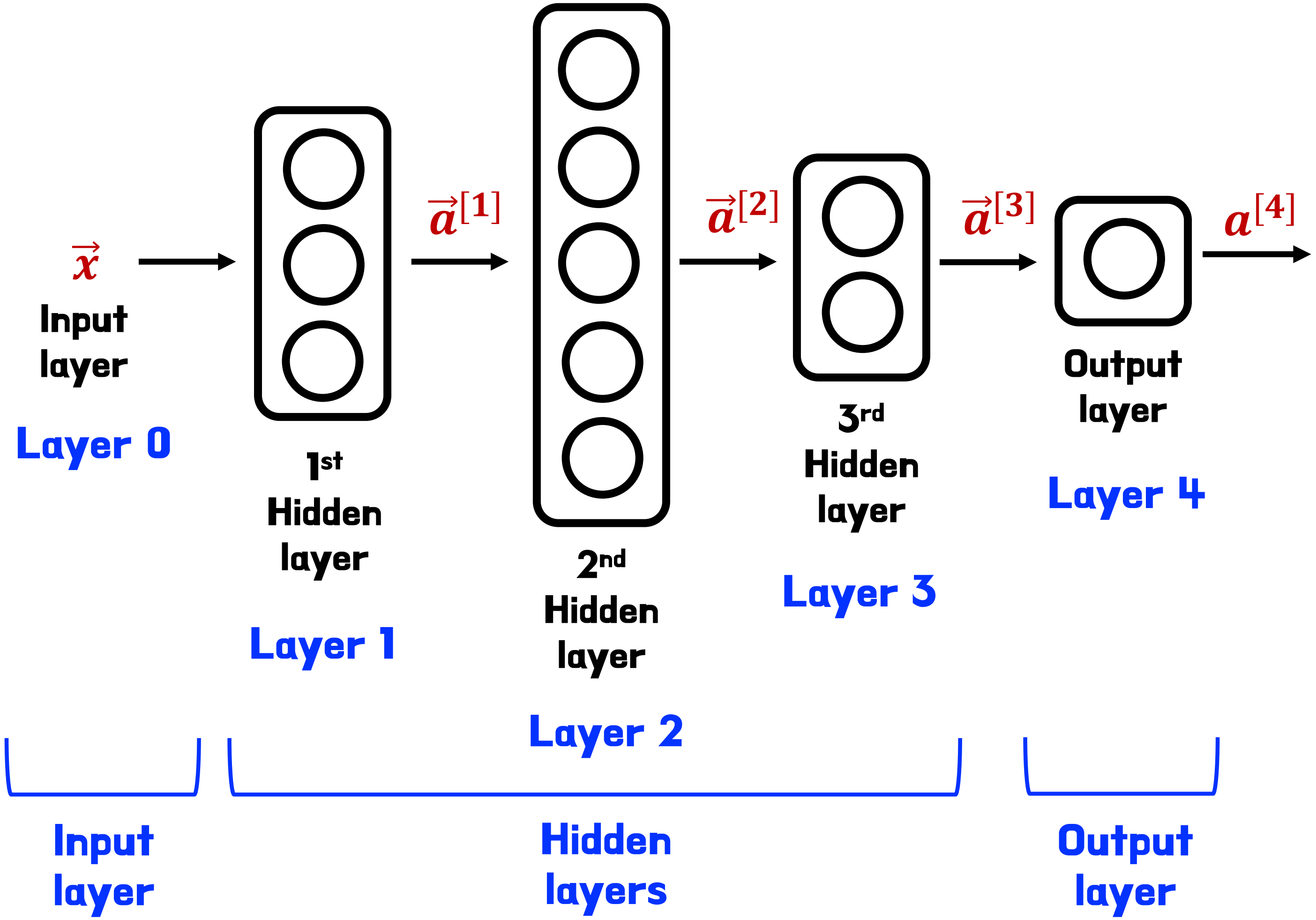
Neural network layer



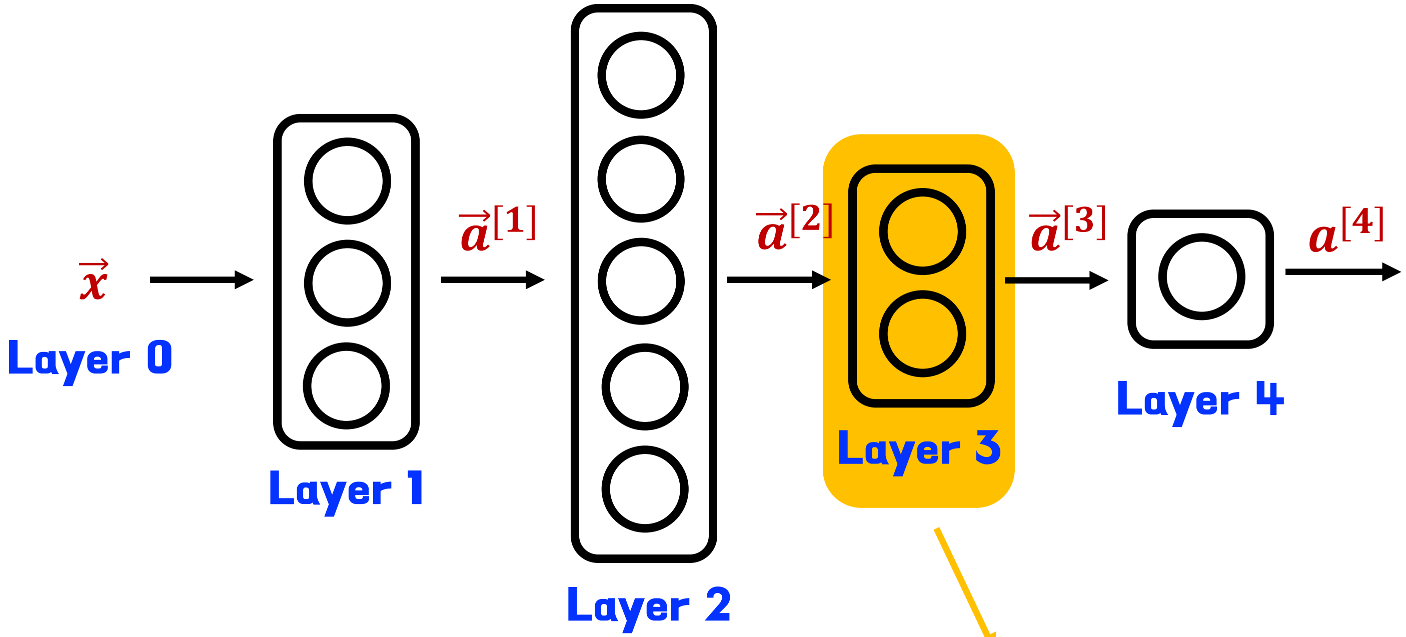
Neural network layer

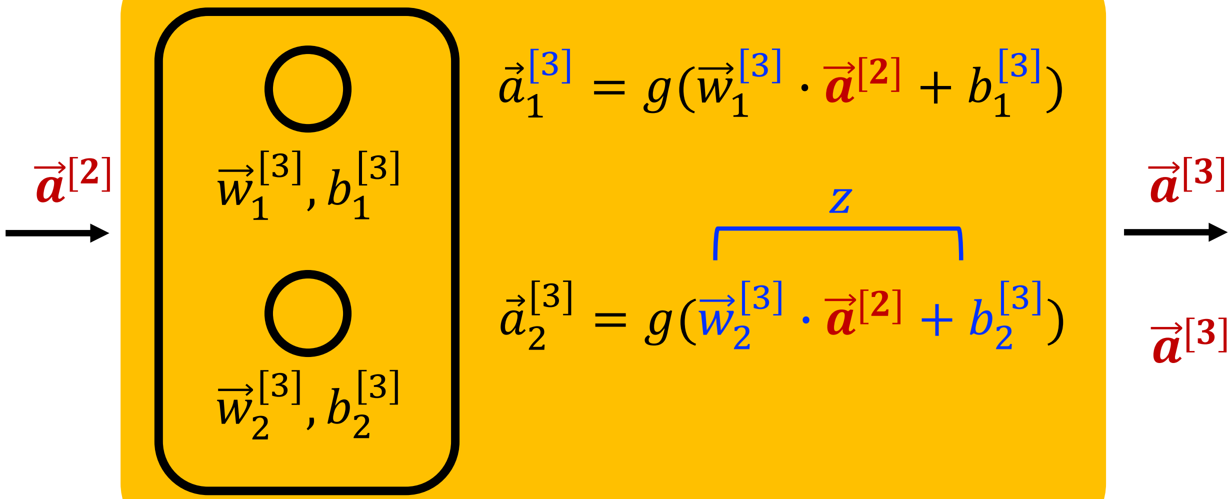


More complex neural network



More complex neural network





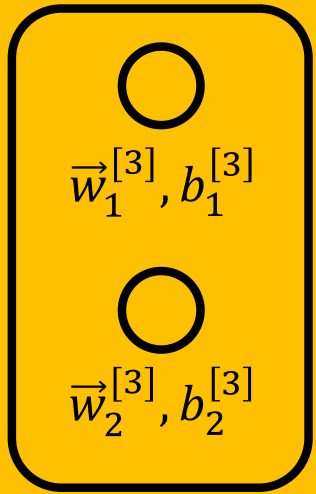
$$\vec{a}_1^{[3]} = g(\vec{w}_1^{[3]} \cdot \vec{a}^{[2]} + b_1^{[3]})$$

$$\vec{a}_2^{[3]} = g(\overbrace{\vec{w}_2^{[3]} \cdot \vec{a}^{[2]} + b_2^{[3]}}^{z})$$

$$\vec{a}^{[3]} = \begin{bmatrix} a_1 \\ a_2 \end{bmatrix}$$

More complex neural network

$\vec{a}^{[2]} \rightarrow$



$$\vec{a}_1^{[3]} = g(\vec{w}_1^{[3]} \cdot \vec{a}^{[2]} + b_1^{[3]})$$

$$\vec{a}_2^{[3]} = g(\overbrace{\vec{w}_2^{[3]} \cdot \vec{a}^{[2]} + b_2^{[3]}}^{z})$$

$\vec{a}^{[3]} \rightarrow$

$$\vec{a}^{[3]} = \begin{bmatrix} a_1 \\ a_2 \end{bmatrix}$$

Activation value of layer l, unit (neuron) j

Output of layer $l - 1$ (previous layer)

$$\vec{a}_j^{[l]} = g(\vec{w}_j^{[l]} \cdot \vec{a}^{[l-1]} + b_j^{[l]})$$

Sigmoid "activation function"

Parameters w & b of layer l, unit j

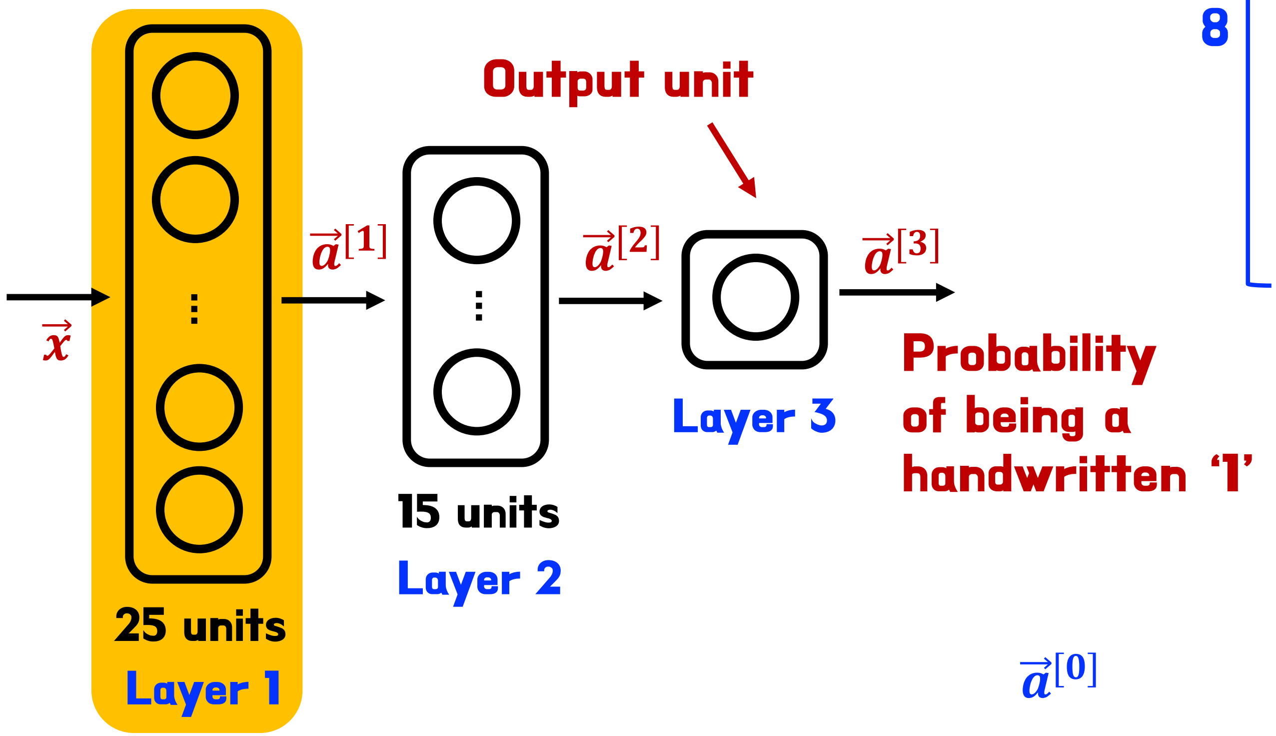
Inference: making predictions (forward propagation)

Handwritten digit recognition

	Digit images		
Digit images	0	1
Label	0	1

8 × 8

255	255	255	255	255	255	255	255
255	255	255	0	255	255	255	255
255	255	0	0	255	255	255	255
255	255	255	0	255	255	255	255
255	255	255	0	255	255	255	255
255	255	255	0	255	255	255	255
255	255	0	0	0	255	255	255
255	255	255	255	255	255	255	255



$\vec{a}^{[0]}$

Hidden units (neurons)

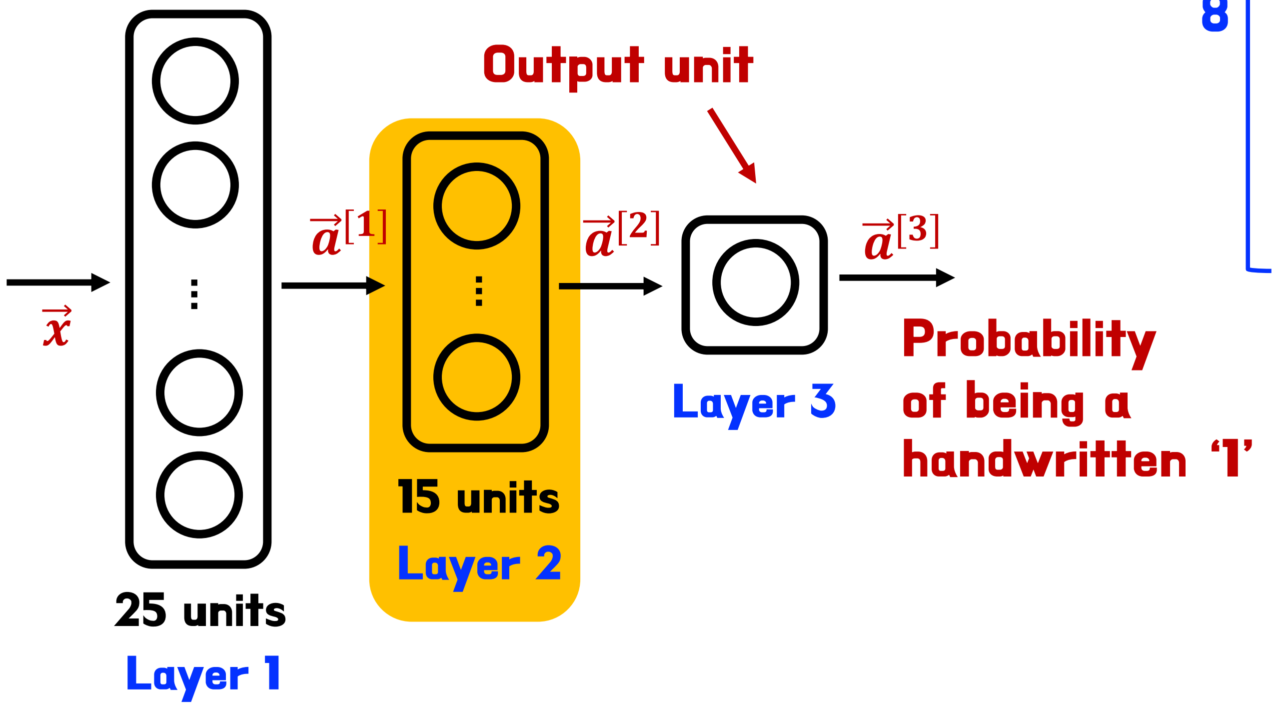
$$\vec{a}^{[1]} = \begin{bmatrix} g(\vec{w}_1^{[1]} \cdot \vec{x} + b_1^{[1]}) \\ \cdots \\ g(\vec{w}_{25}^{[1]} \cdot \vec{x} + b_{25}^{[1]}) \end{bmatrix}$$

Handwritten digit recognition

Digit images	0	1
Label	0	1

8 × 8

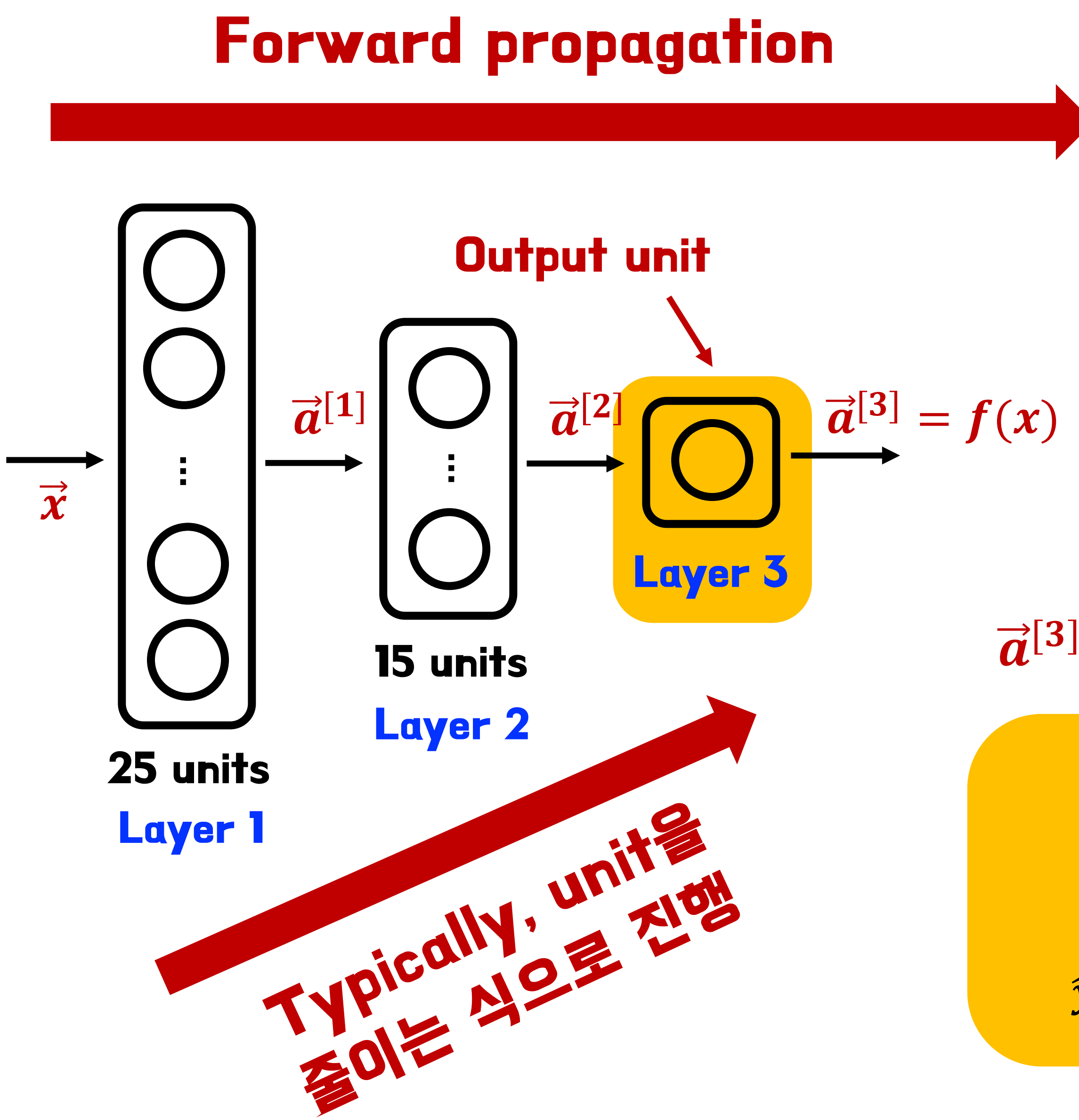
255	255	255	255	255	255	255	255
255	255	255	0	255	255	255	255
255	255	0	0	255	255	255	255
255	255	255	0	255	255	255	255
255	255	255	0	255	255	255	255
255	255	255	0	255	255	255	255
255	255	0	0	0	255	255	255
255	255	255	255	255	255	255	255



Hidden units (neurons)

$$\vec{a}^{[2]} = \begin{bmatrix} g(\vec{w}_1^{[2]} \cdot \vec{a}^{[1]} + b_1^{[2]}) \\ \cdots \\ g(\vec{w}_{15}^{[2]} \cdot \vec{a}^{[1]} + b_{15}^{[2]}) \end{bmatrix}$$

Handwritten digit recognition



Probability of being a handwritten '1'

$$\vec{a}^{[3]} = g(\vec{w}_1^{[3]} \cdot \vec{a}^{[2]} + b_1^{[3]})$$

is $a_1^{[3]} \geq 0.5$?

$\hat{y} = 1$ $\hat{y} = 0$

Speculation on artificial general intelligence (AGI)

AI

ANI

(artificial narrow intelligence)

e.g.) smart speaker, self-driving car, web search, AI in farming and factories

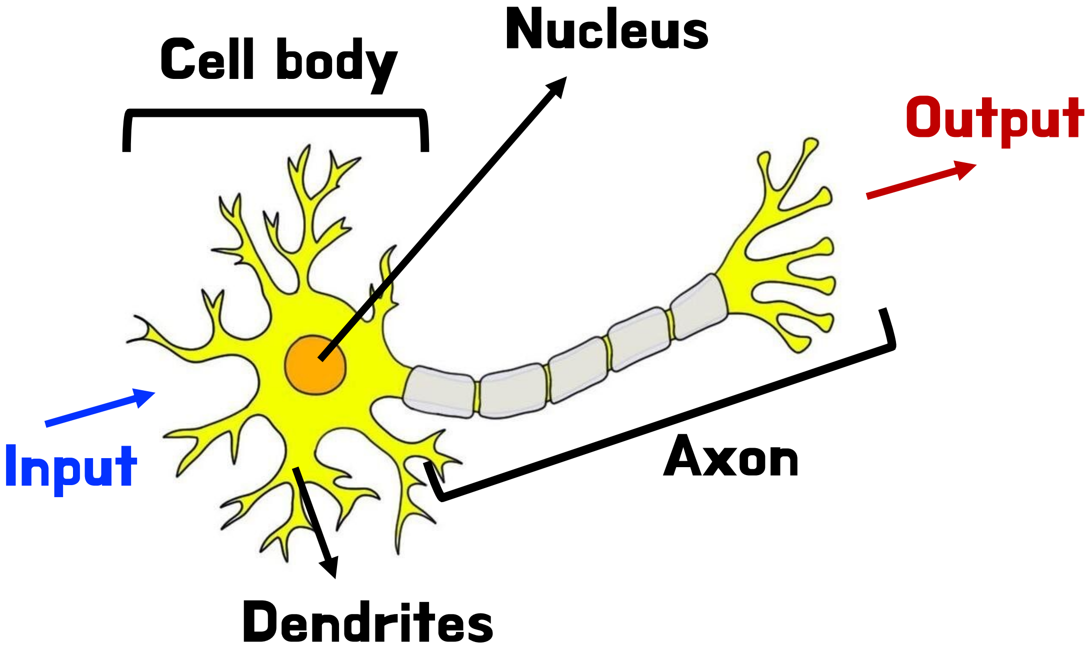
AGI

(artificial narrow intelligence)

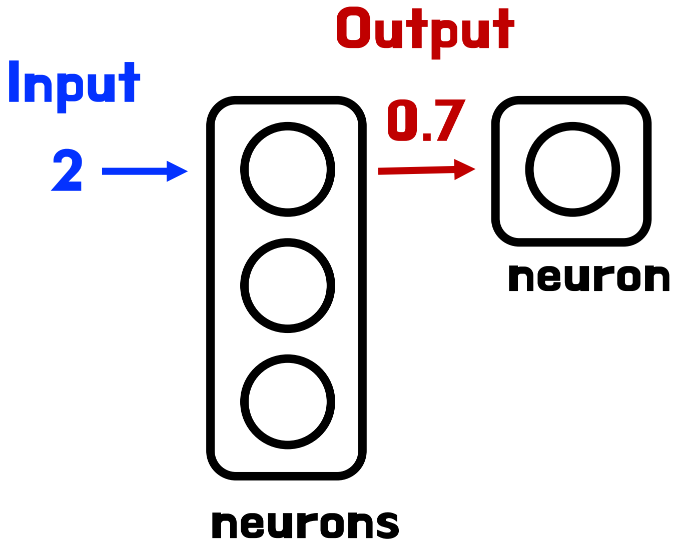
Do anything a human can do

Neural network and the brain

Biological neuron



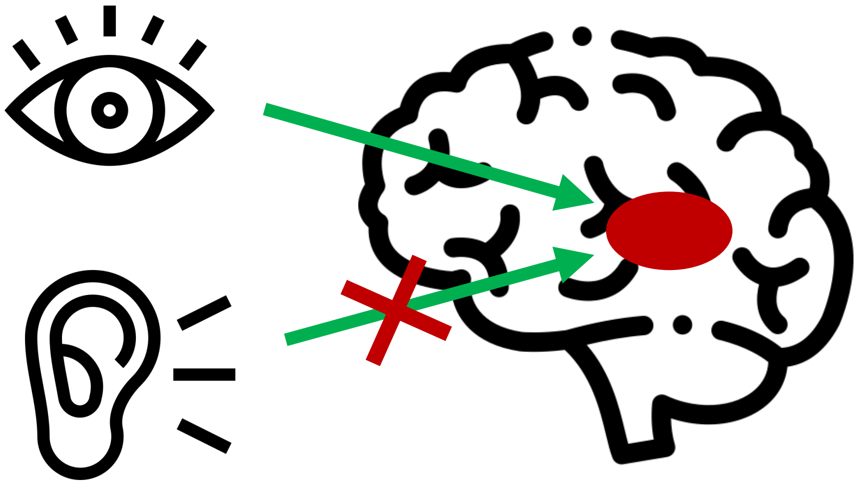
Simplified mathematical model of a neuron



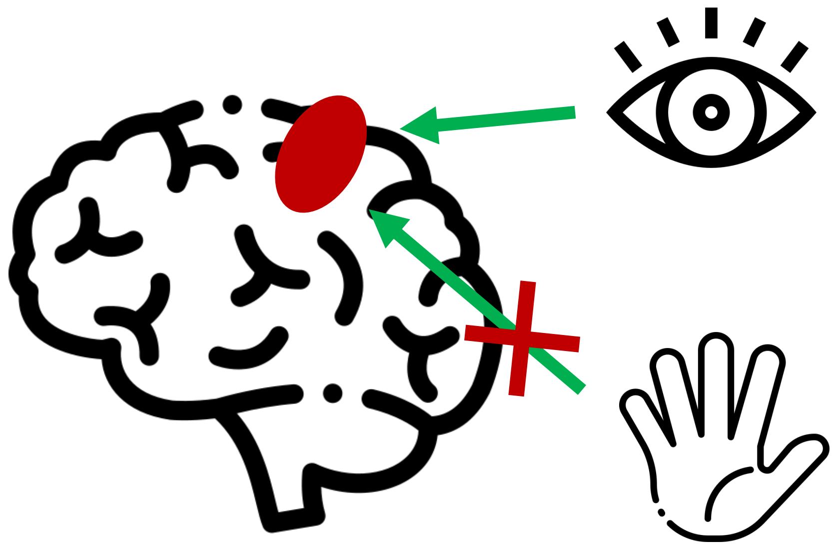
Can we mimic the human brain ?

1. Machine learning is too simple
2. We don't know how the brain workrs

The "one learning algorithm" hypothesis



Auditory cortex

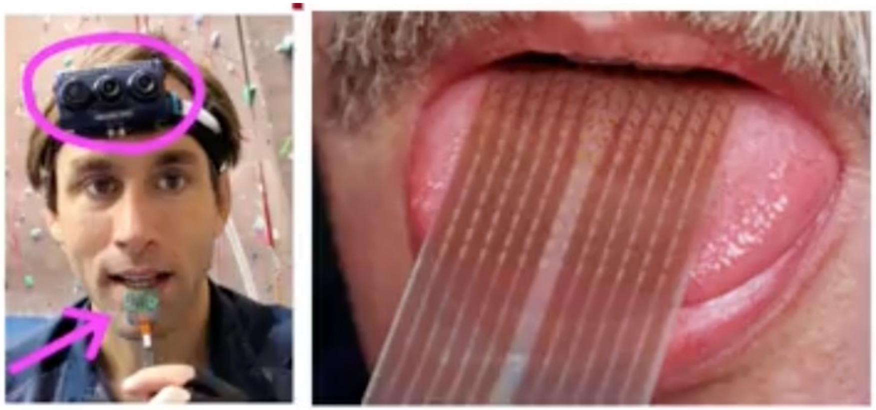


Somatosensory cortex

One brain cortex can learn other sense as well

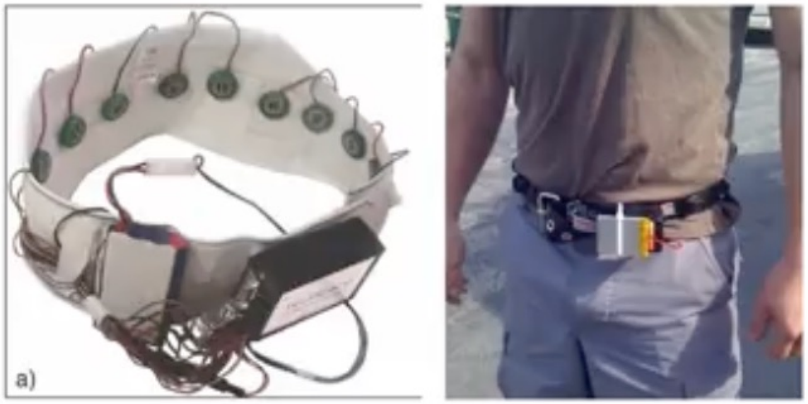
Sensor representations in the brain

We don't know if it is possible in machine learning



Seeing with your tongue

Human echolocation (sonar)

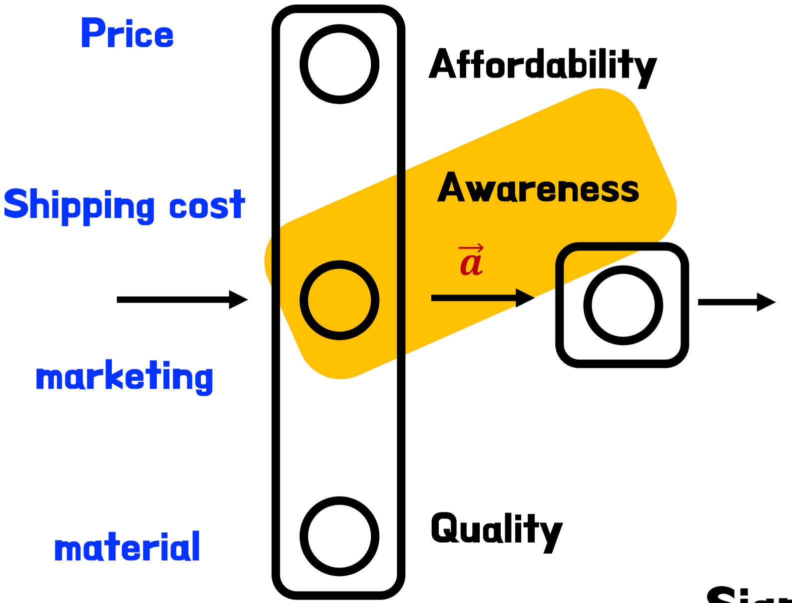


Haptic belt: Direction sense

Implanting a 3rd eye

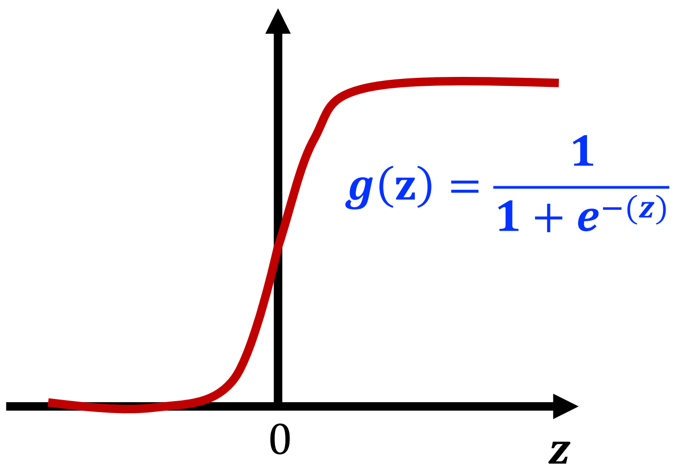
Activation function

Demand prediction example



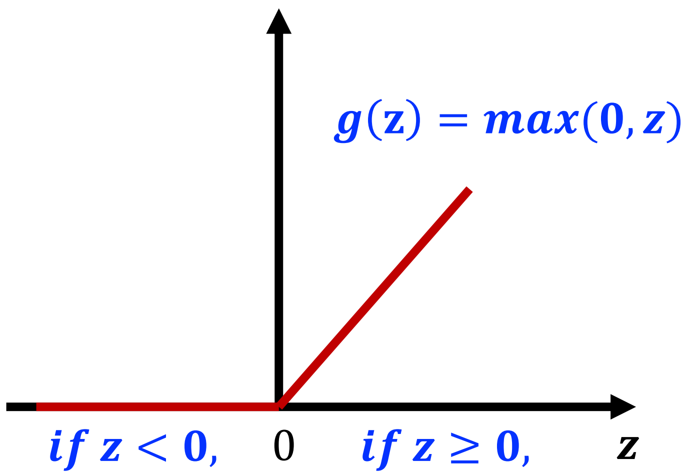
$$a_2^{[1]} = g(\vec{w}_2^{[1]} \cdot \vec{x} + b_2^{[1]})$$

Sigmoid



$0 < g(z) < 1$

ReLU (Rectified Linear Unit)



$if\ z < 0,$ $g(z) = 0$

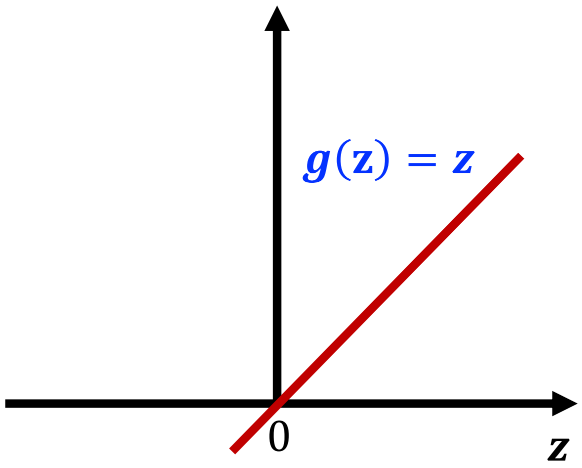
$if\ z \geq 0,$ $g(z) = z$

Examples of activation functions

$$a_2^{[1]} = g(\vec{w}_2^{[1]} \cdot \vec{x} + b_2^{[1]})$$

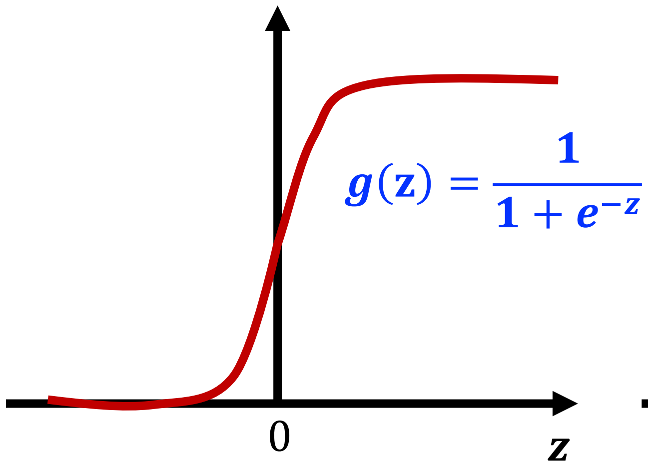
"No activation function"

Linear activation function



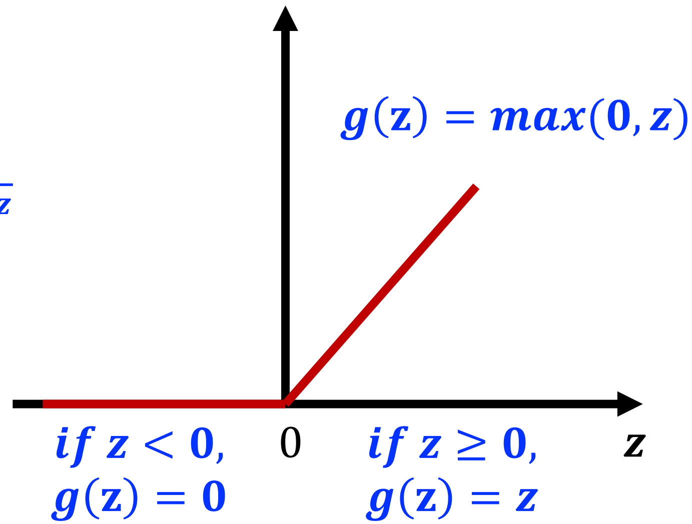
$a = g(z) = \vec{w} \cdot \vec{x} + b$

Sigmoid

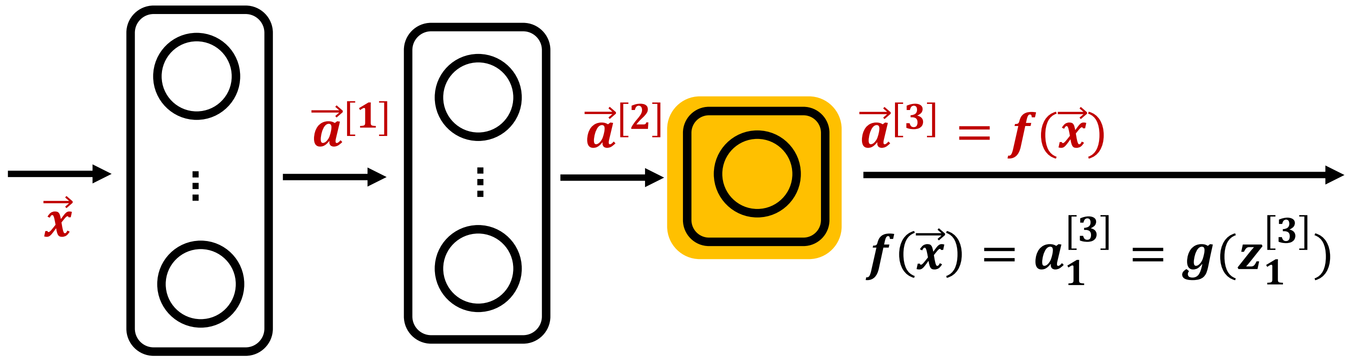


$0 < g(z) < 1$

ReLU
(Rectified Linear Unit)

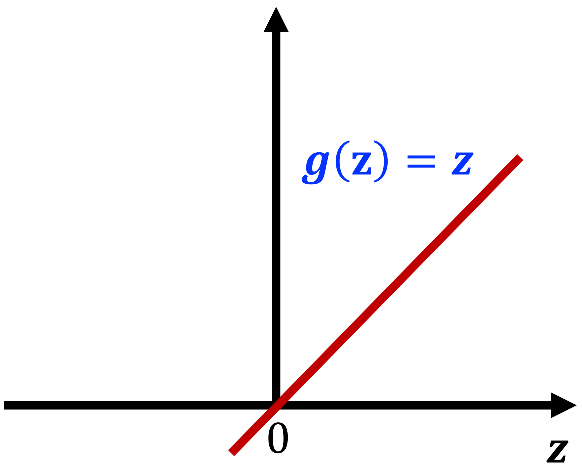


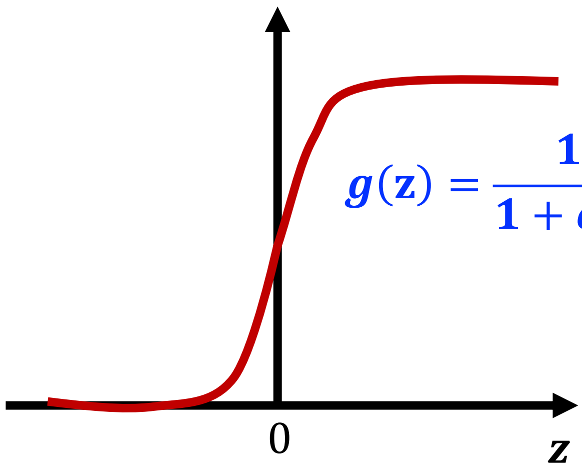
Output layer

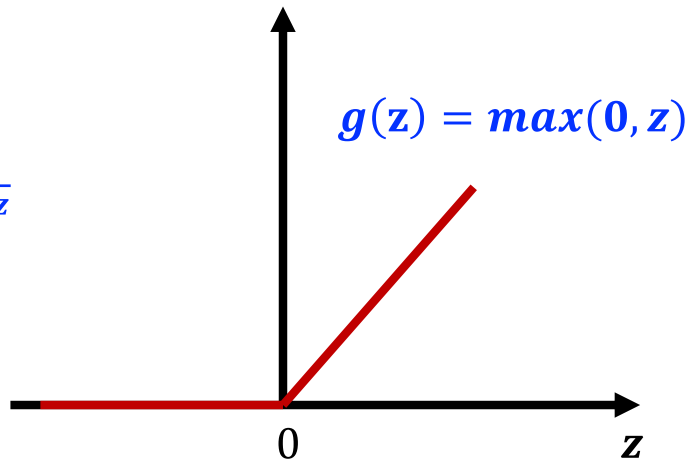


Choosing $g(z)$ for output layer ?

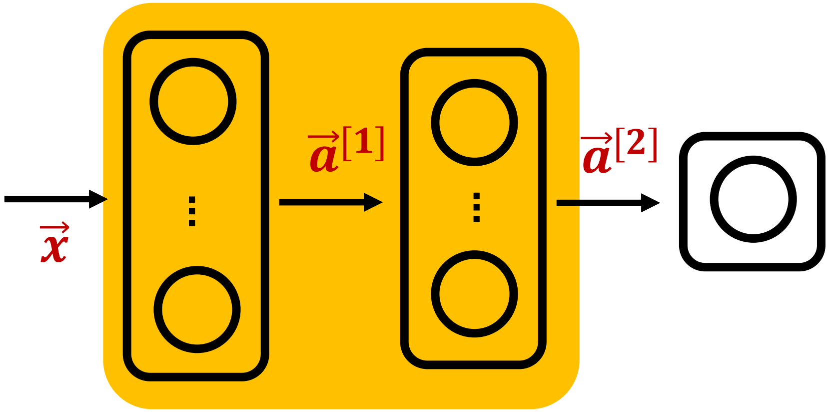
Regression	Binary classification	Regression
Linear activation function **y = +/-**	**Sigmoid** **y = 0/1**	**ReLU** **y = 0 or +**







Hidden layer



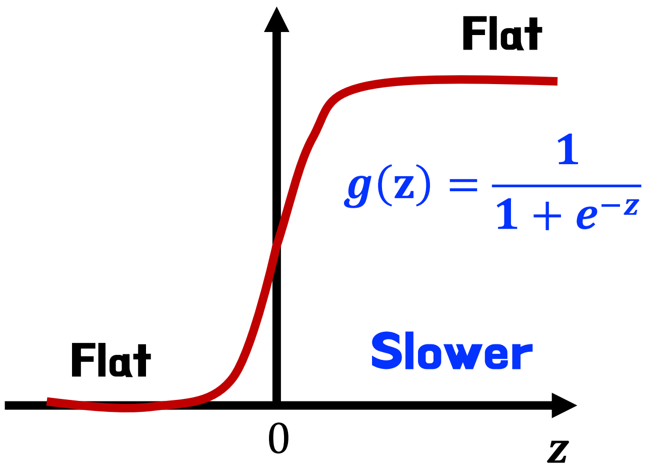
$\vec{a}^{[3]} = f(\vec{x})$

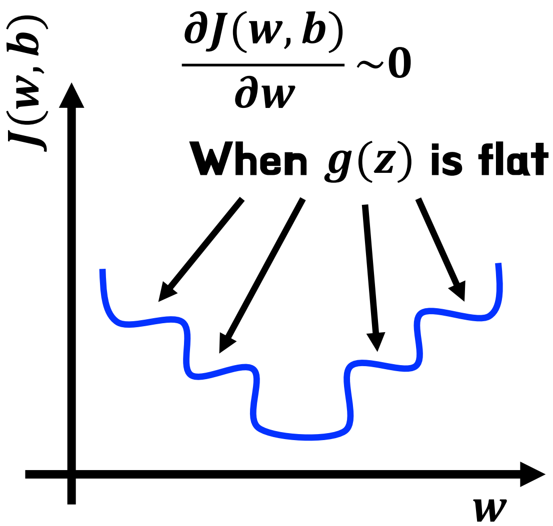
$f(\vec{x}) = a_1^{[3]} = g(z_1^{[3]})$

Choosing $g(z)$ for hidden layer ?

Classical

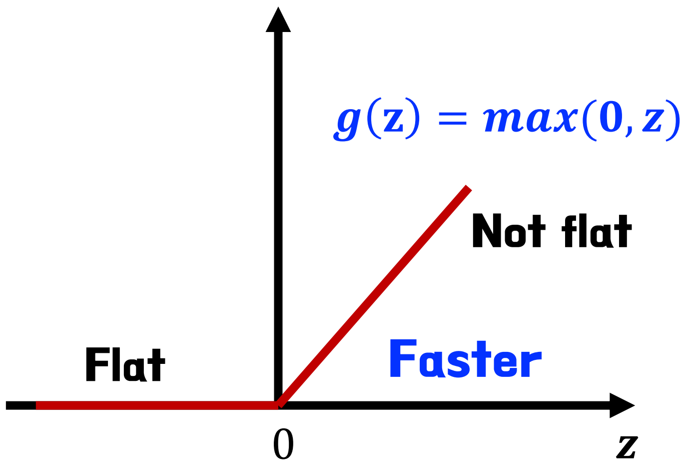
Sigmoid





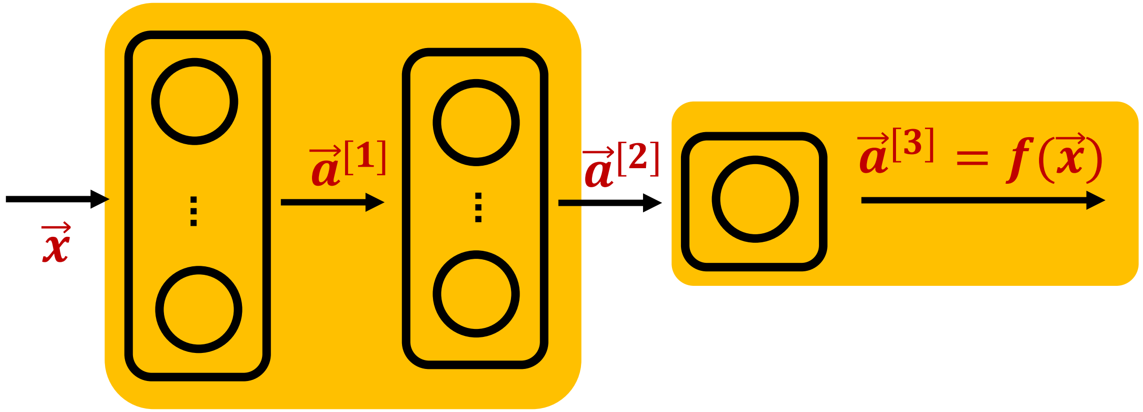
Most common

ReLU



TheMAD

Choosing activation summary



ReLU for hidden layers

Binary classification
Activation = **'sigmoid'**

Regression (y: -/+)
Activation = **'linear'**

Regression (y: 0/+)
Activation = **'relu'**

Typical coding

```
from tf.keras.layers import Dense
model = Sequential([
   Dense(units=25, activation='relu'),
   Dense(units=15, activation='relu'),
   Dense(units=1, activation='sigmoid'),
])
```

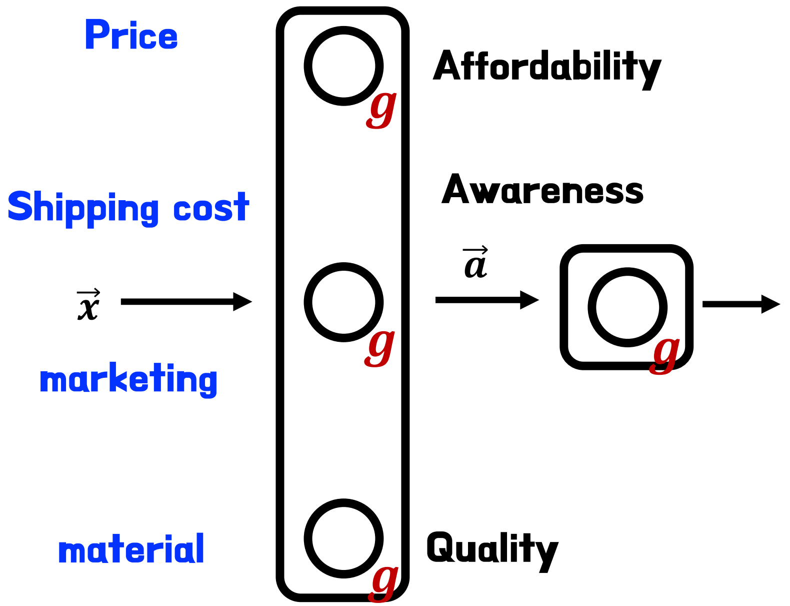
layer1

layer2

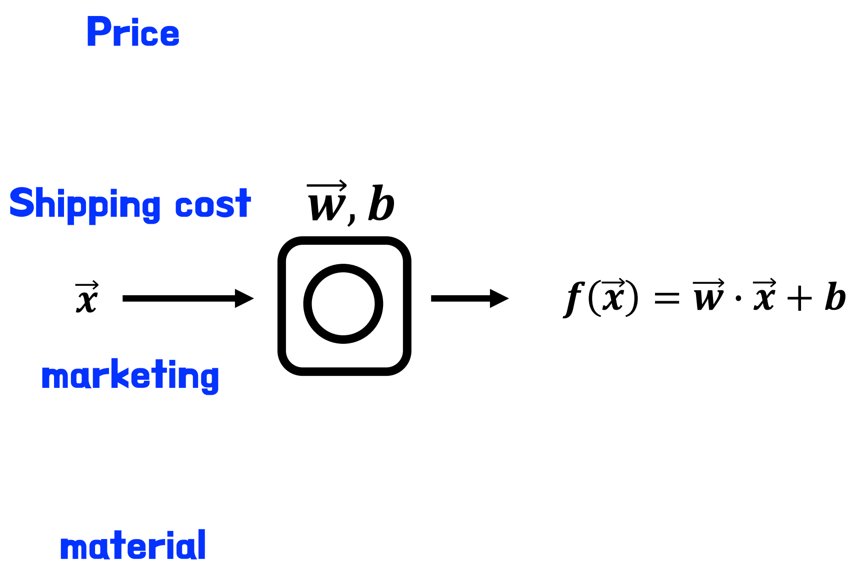
layer3 = output layer

There are another activation functions: 'Reacky ReLU', 'Tanh', 'Softmax'

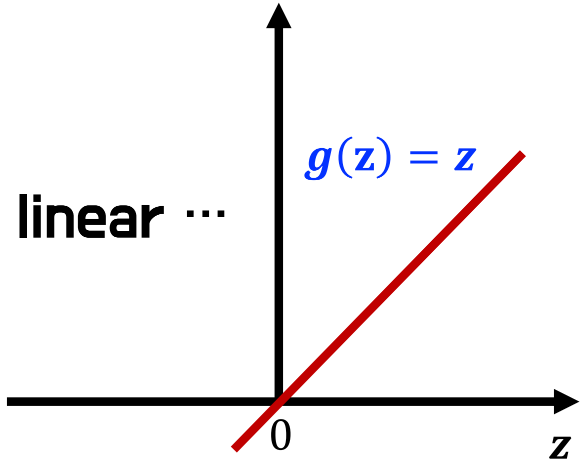
Why do we need activation functions ?



VS



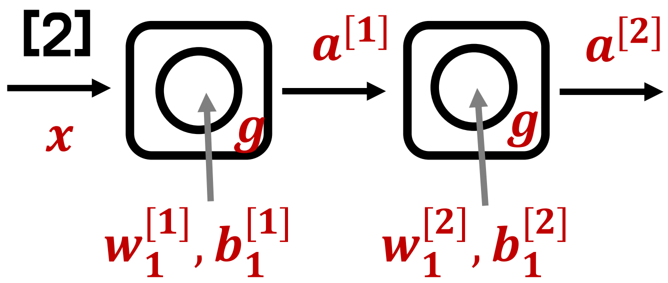
If all $g(z)$ is linear …



… no different than linear regression

We can't solve complicated cases

Linear example



$$g(z) = z$$

$$a^{[1]} = w_1^{[1]} x + b_1^{[1]}$$

$$a^{[2]} = w_1^{[2]} a^{[1]} + b_1^{[2]}$$

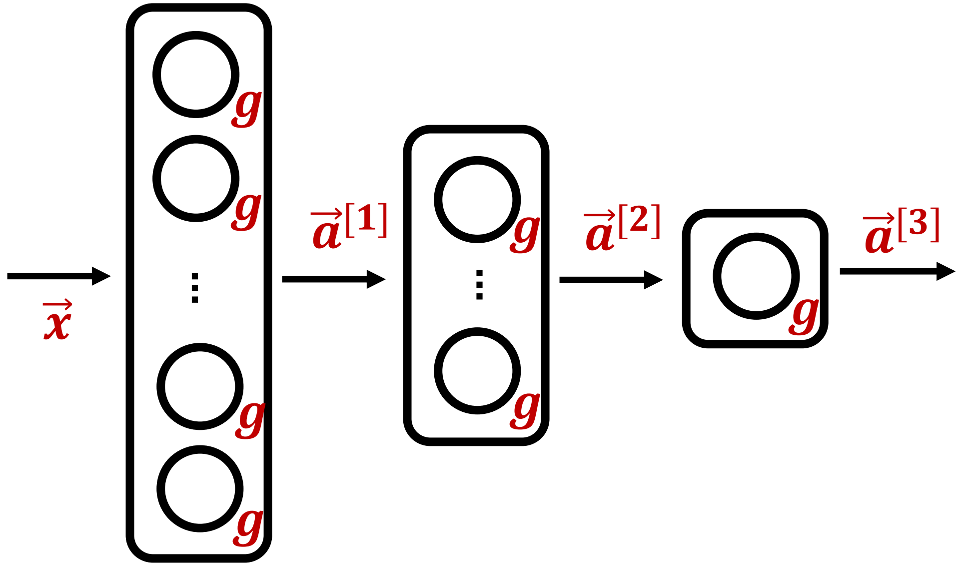
$$= w_1^{[2]} (w_1^{[1]} x + b_1^{[1]}) + b_1^{[2]}$$

$$= \underbrace{w_1^{[2]} w_1^{[1]}}_{w} x + \underbrace{(w_1^{[2]} b_1^{[1]} + b_1^{[2]})}_{b}$$

$$a^{[2]} = wx + b$$

Again, same linear regression

Example



$$g(z) = z$$

$$\vec{a}^{[3]} = \vec{w}_1^{[3]} \cdot \vec{a}^{[2]} + b_1^{[3]}$$

All linear (including output)
→ Equivalent to linear regression

$$g(z) = \frac{1}{1 + e^{-z}}$$

$$\vec{a}^{[3]} = \frac{1}{1 + e^{-(\vec{w}_1^{[3]} \cdot \vec{a}^{[2]} + b_1^{[3]})}}$$

Output : sigmoid (hidden : all linear)
→ Equivalent to logistic regression

Don't use linear activation in hidden layers (use ReLU)