Introduction to Deep Neural Networks

주재걸 교수
KAIST 김재철AI대학원

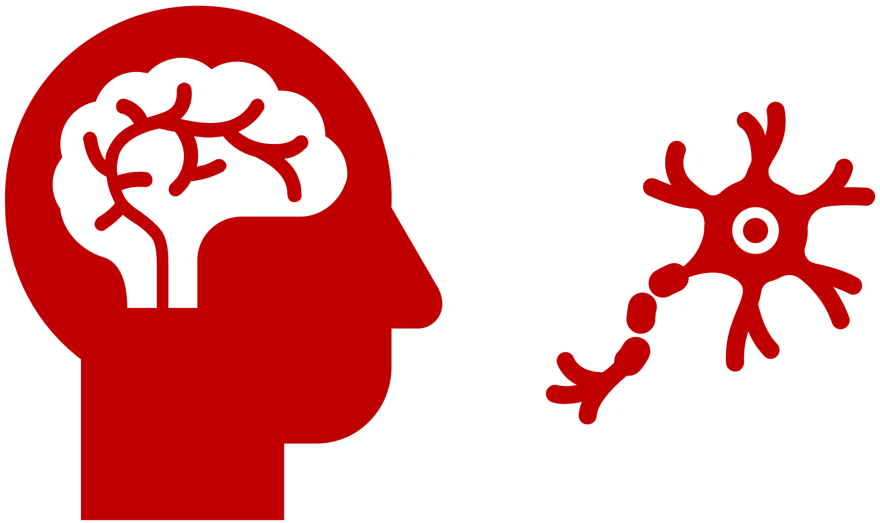
DAVIAN
Data and Visual Analytics Lab

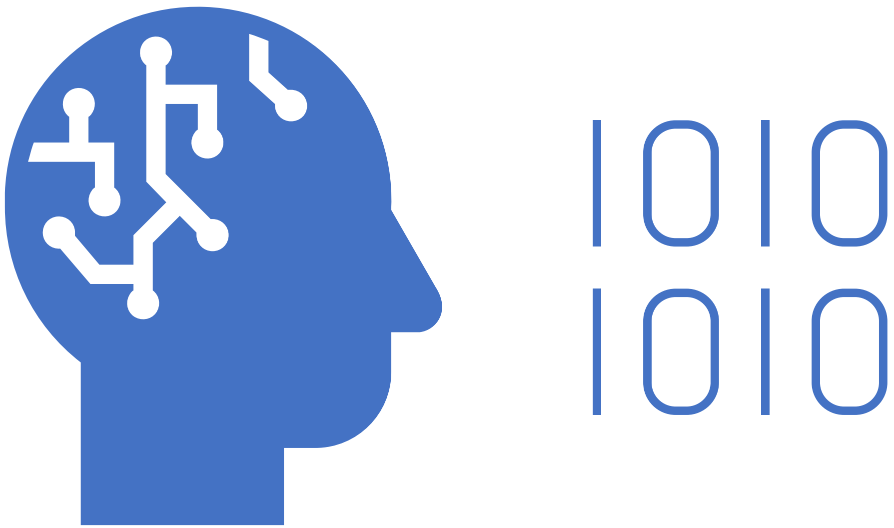
심층신경망의 기본 동작 과정

Deep Learning

Deep Learning

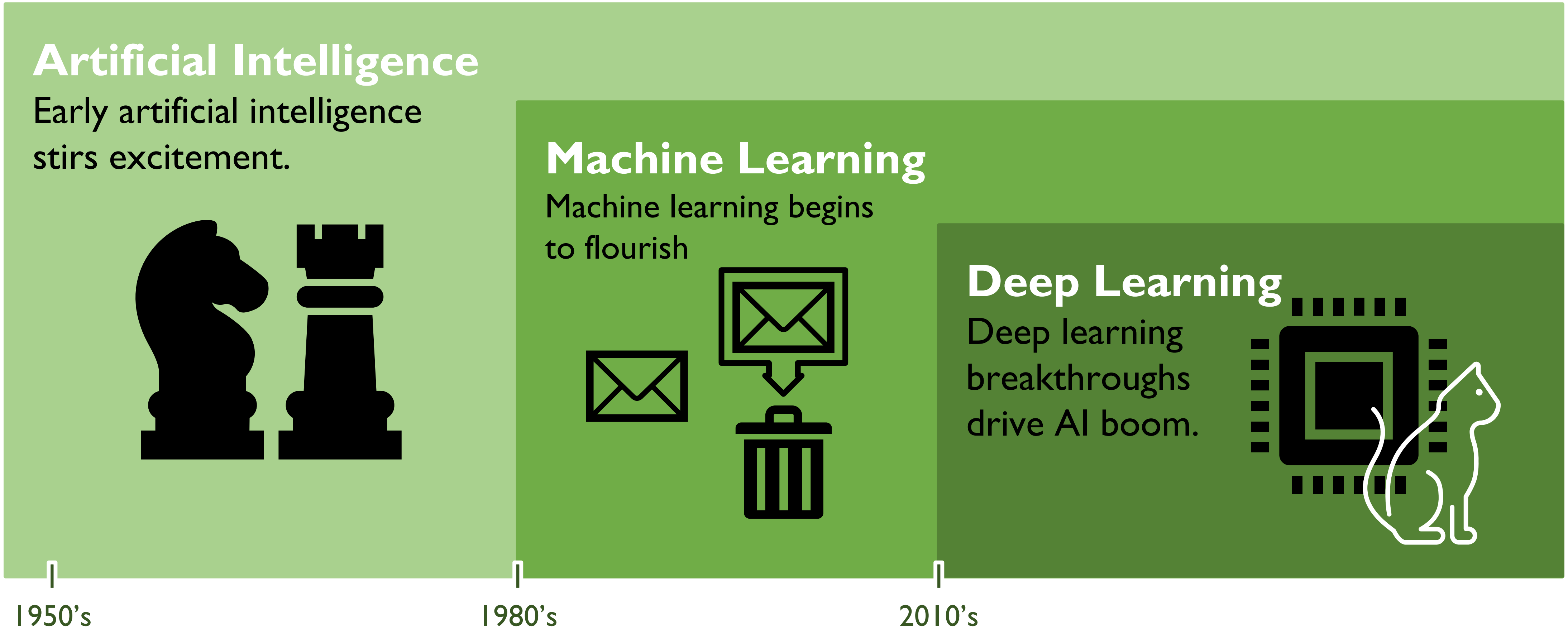
- Deep learning refers to artificial neural networks that are composed of many layers.





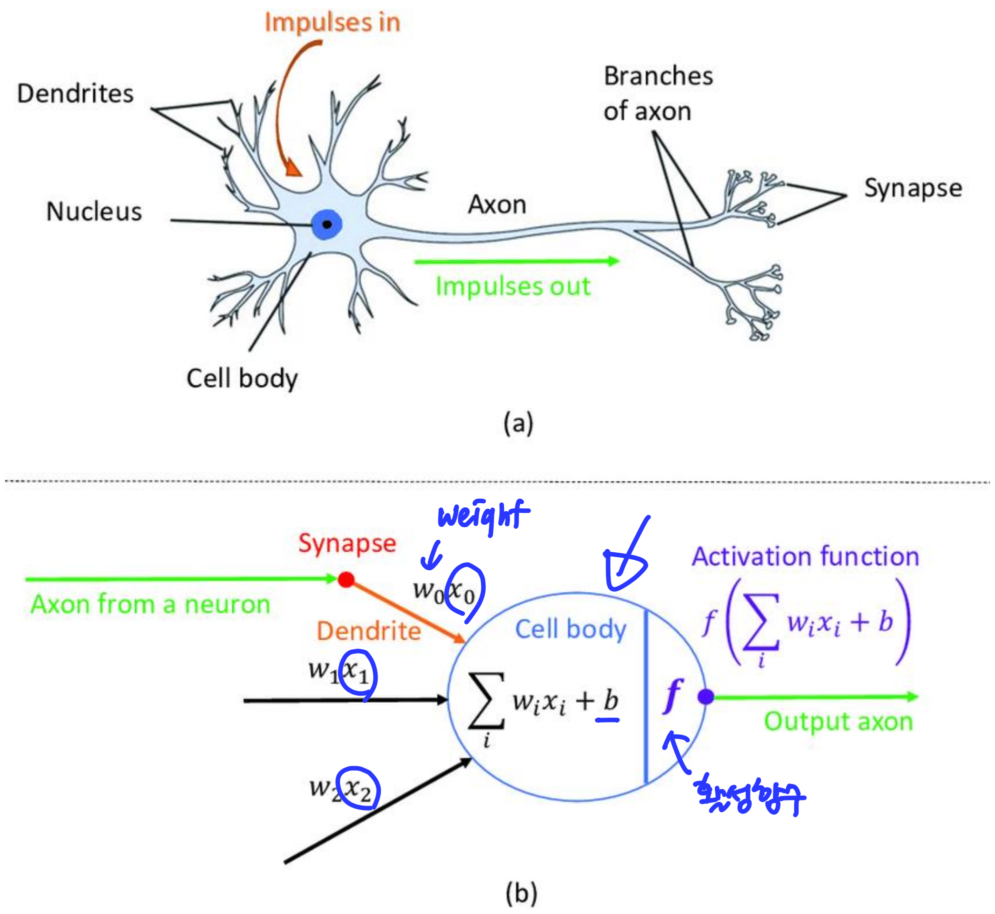
2

Artificial Intelligence vs. Machine Learning vs. Deep Learning

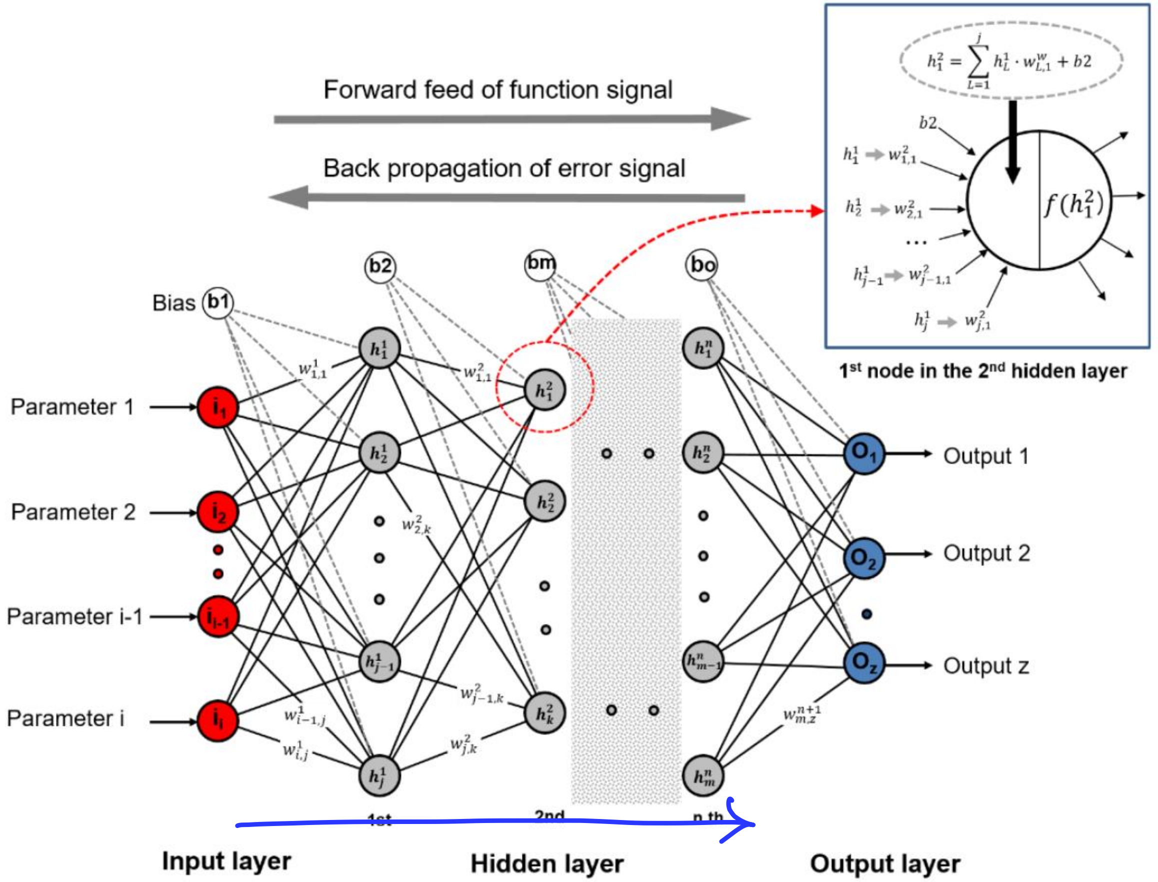


Artificial Neural Networks

- A technology that imitates neurons existing in the human brain



https://arxiv.org/abs/1706.05933

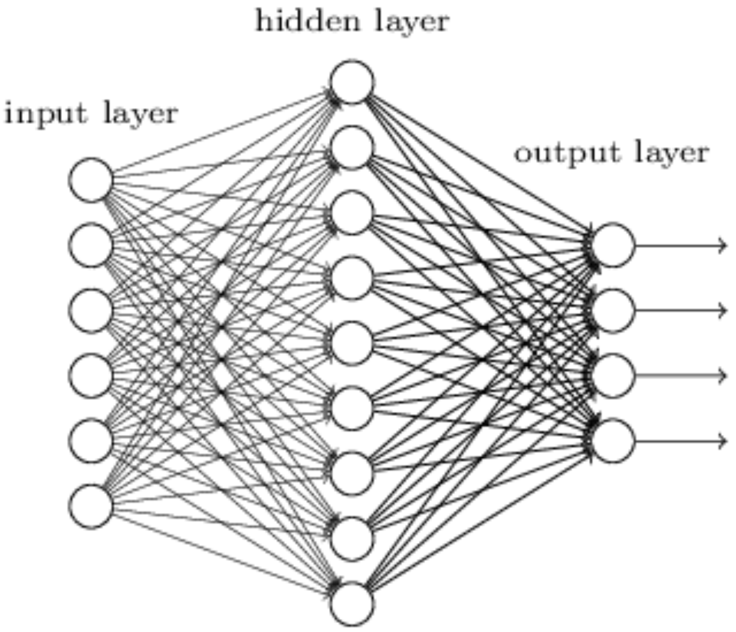


https://www.mdpi.com/2076-3417/10/4/1302

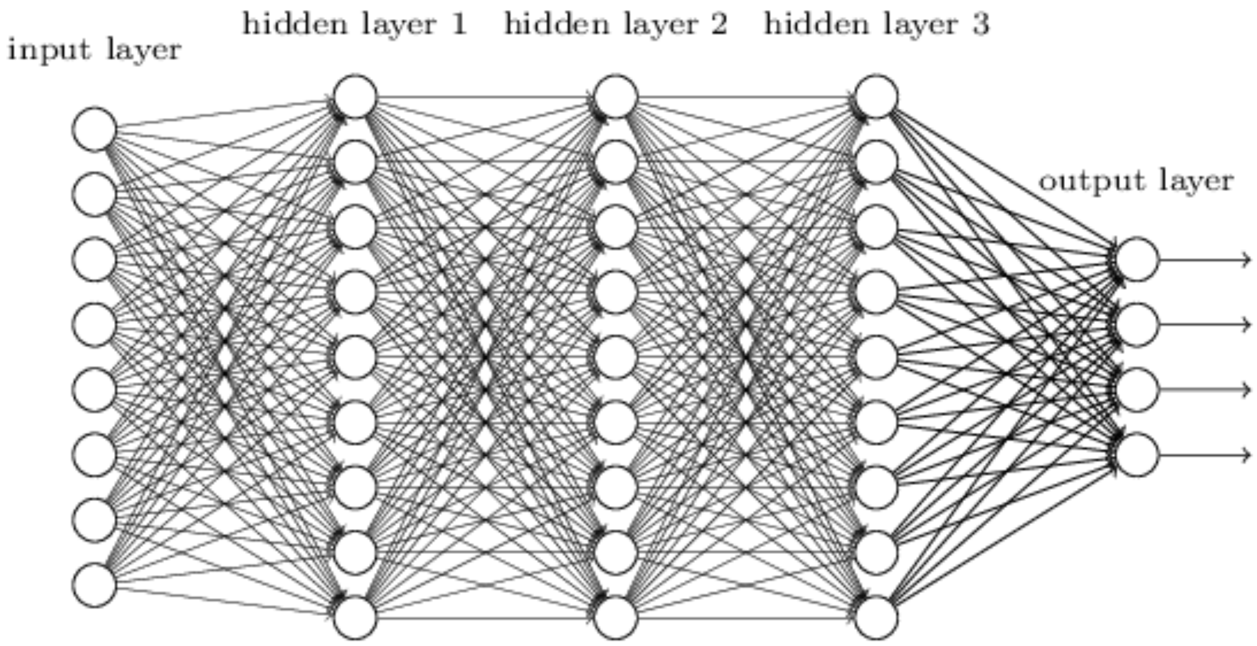
Deep Neural Network (DNN)

- DNN improves accuracy of AI technology by stacking neural network layers

"Non-deep" feedforward neural network



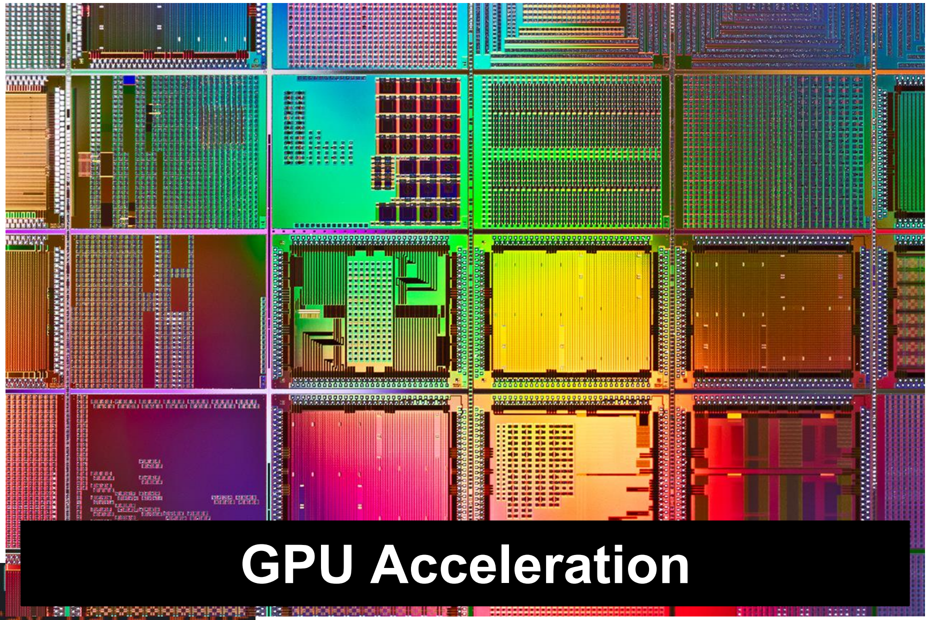
Deep neural network



http://neuralnetworksanddeeplearning.com/chap5.html

Reason Why Deep Learning has been Successful

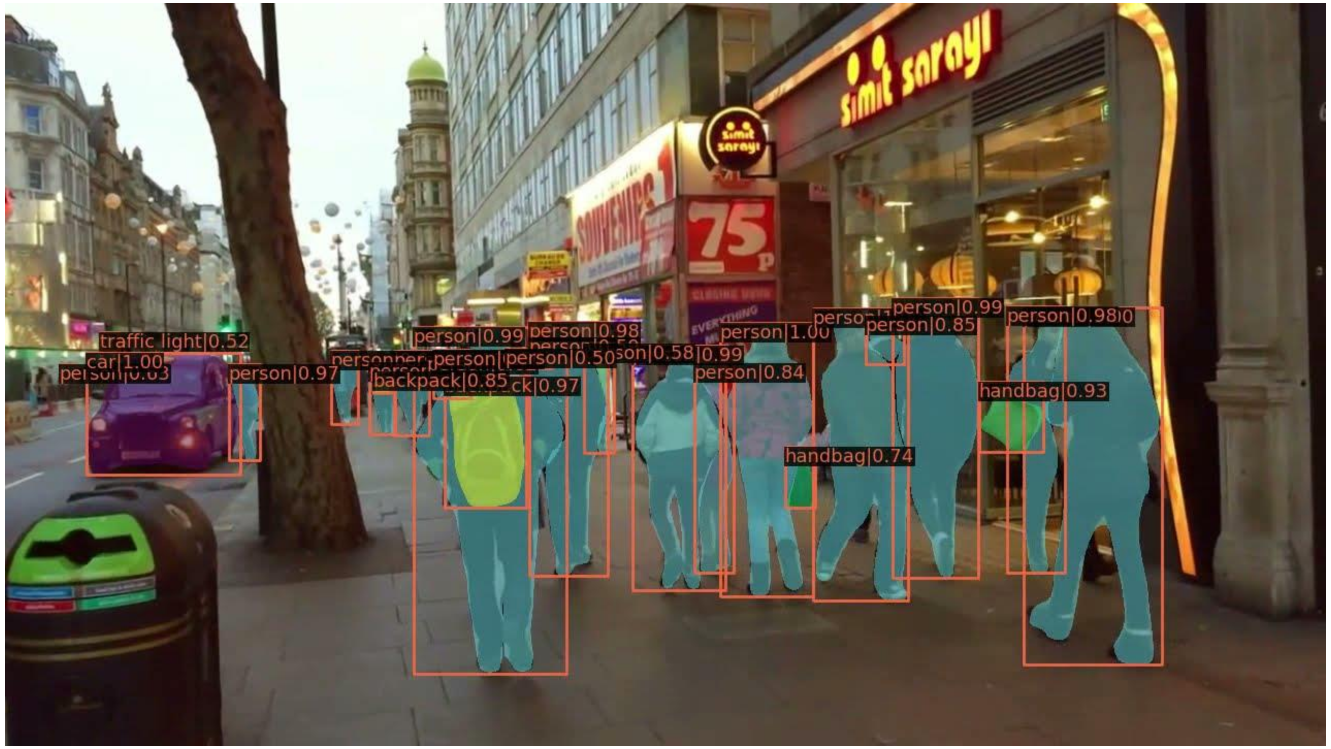
Big Data



GPU Acceleration

Algorithm Improvements

Computer Vision



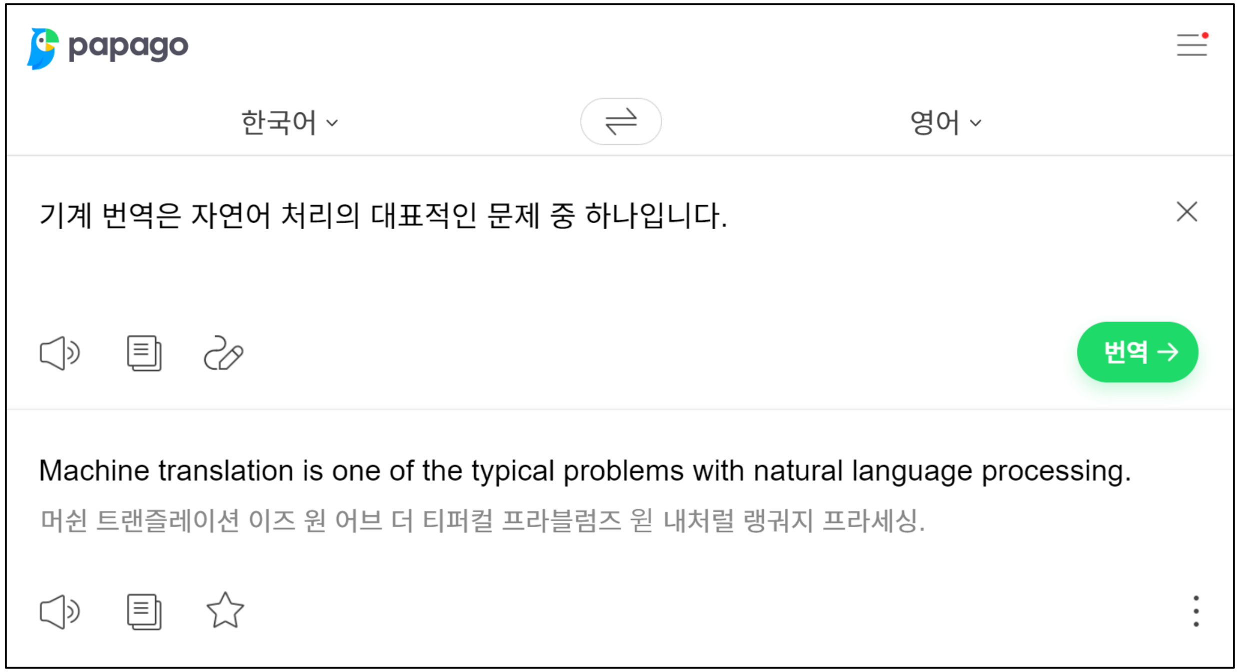
Object Detection

Image Synthesis

Liu, Ze et al. "Swin Transformer: Hierarchical Vision Transformer Using Shifted Windows.“
Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV). 2021.

Karras, Tero et al. "Analyzing and Improving the Image Quality of StyleGAN.“
Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR). 2020.

Natural Language Processing



Machine Translation

Mail Classification

Reinforcement Learning

Go

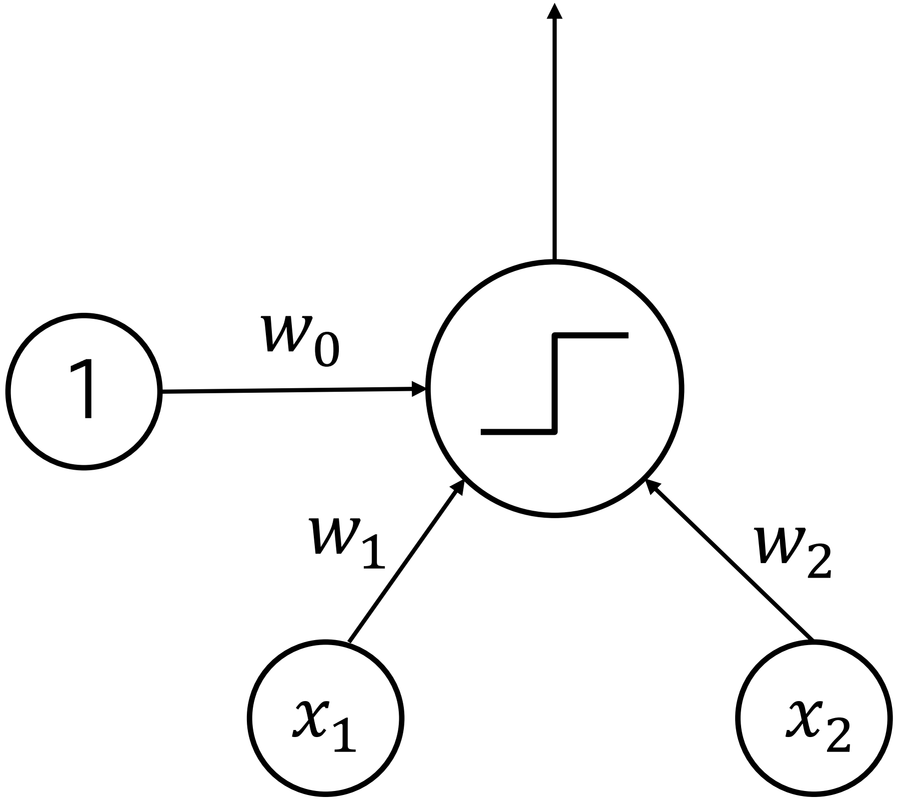
Atari Gane

https://www.deepmind.com/research/highlighted-research/alphago
https://www.deepmind.com/publications/playing-atari-with-deep-reinforcement-learning

Perceptron and Neural Networks

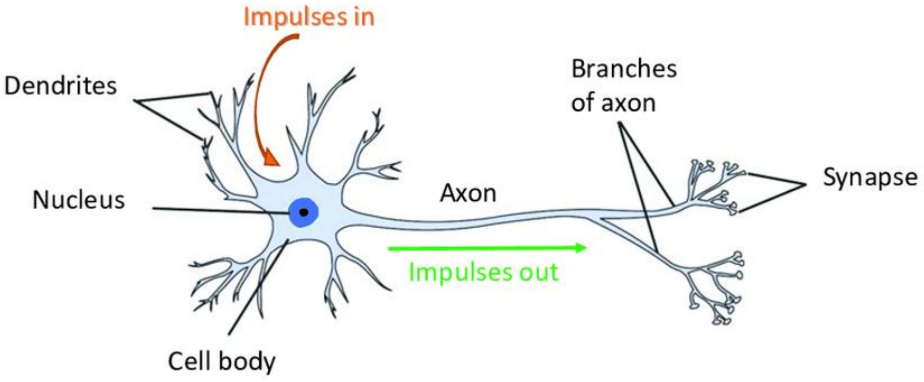
What is Perceptron?

$$y = f(w_0 + w_1 x_1 + w_2 x_2)$$



Perceptron

- One kind of neural networks
- Frank Rosenblatt devised in 1957
- Linear classifier

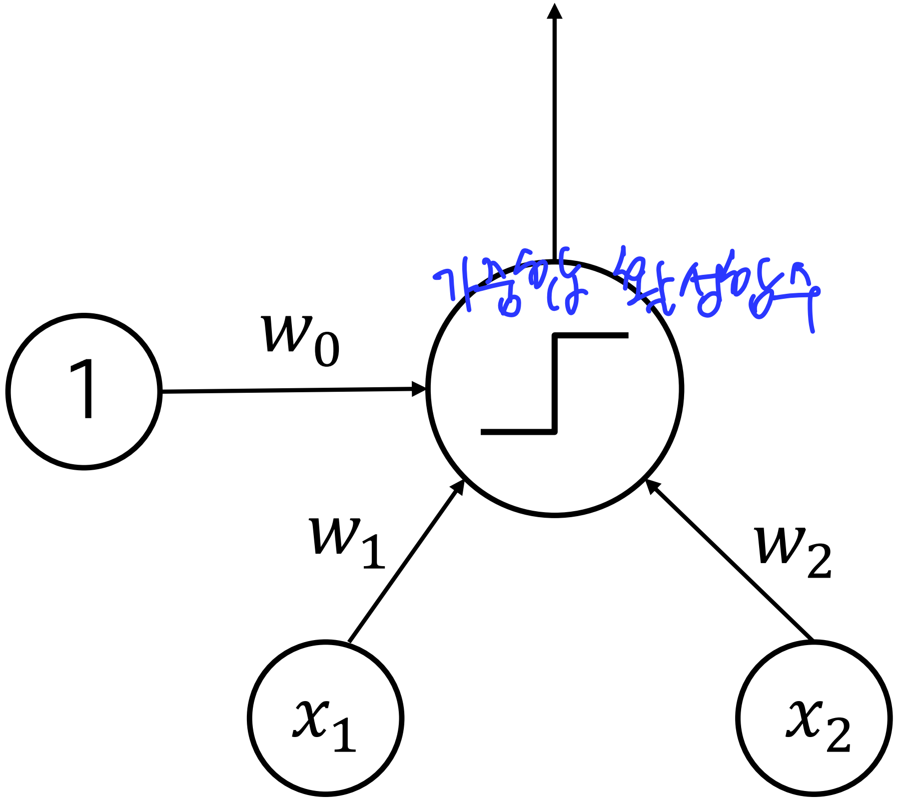


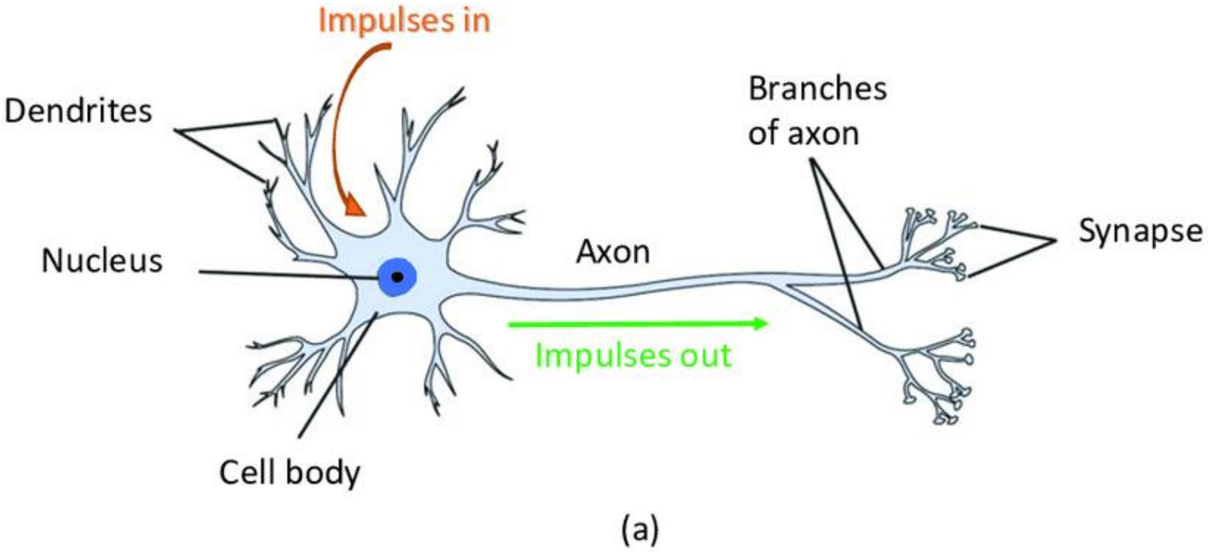
(a)

- Similar with structure of a neuron

What is Perceptron?

$$y = f(w_0 + w_1 x_1 + w_2 x_2)$$



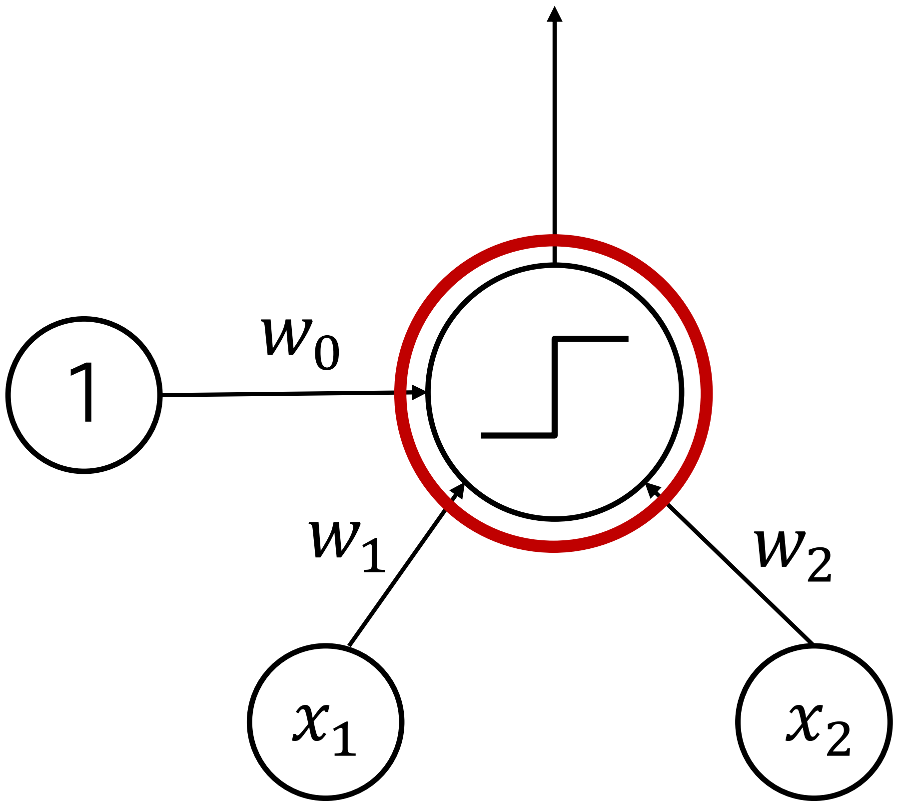


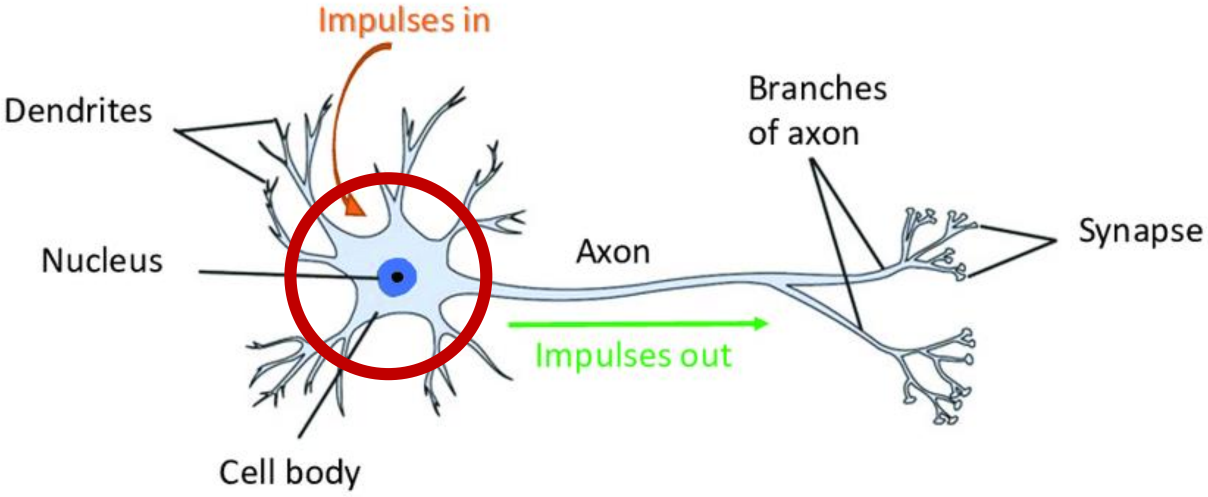
Input: x_1, x_2 Output: y

Weights: w_0, w_1, w_2

What is Perceptron?

$$y = f(w_0 + w_1 x_1 + w_2 x_2)$$





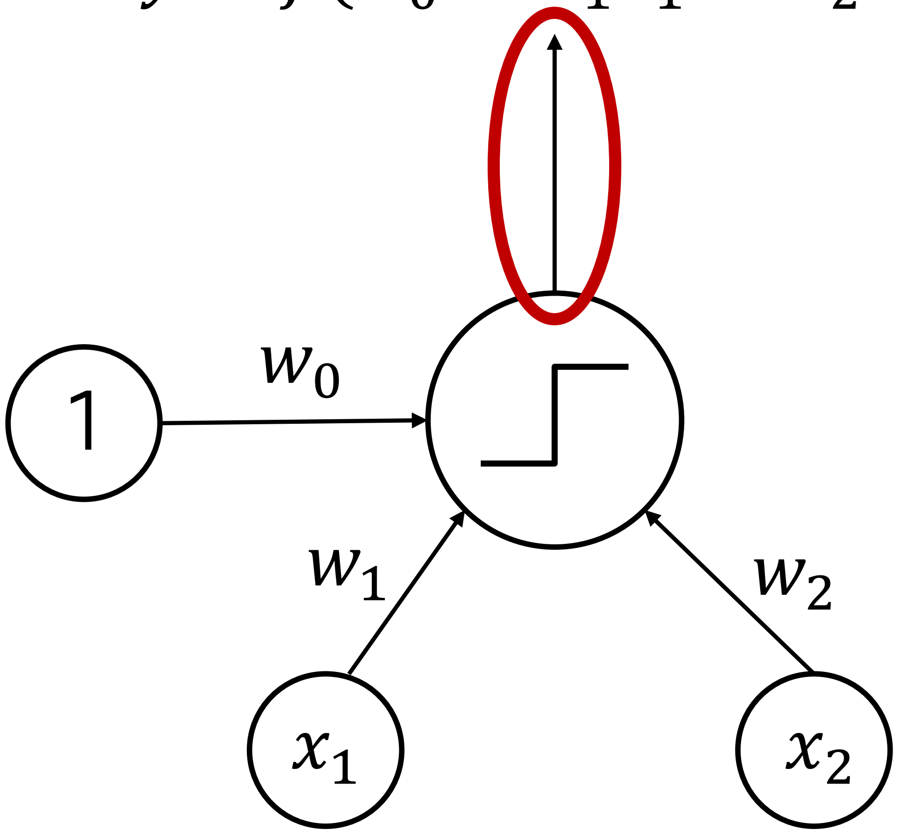
(a)

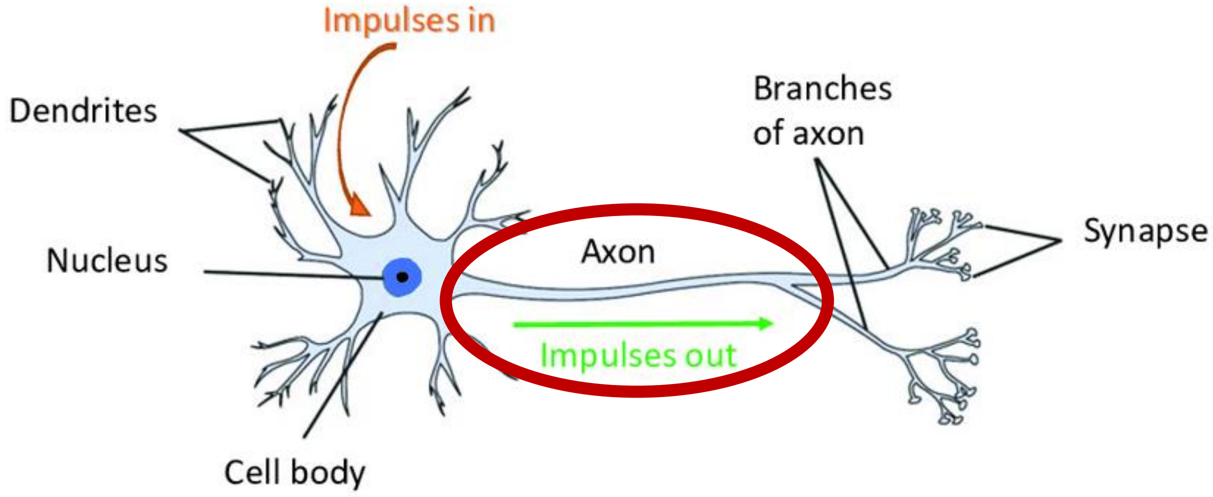
Input: x_1, x_2 Output: y

Weights: w_0, w_1, w_2

What is Perceptron?

$$y = f(w_0 + w_1 x_1 + w_2 x_2)$$





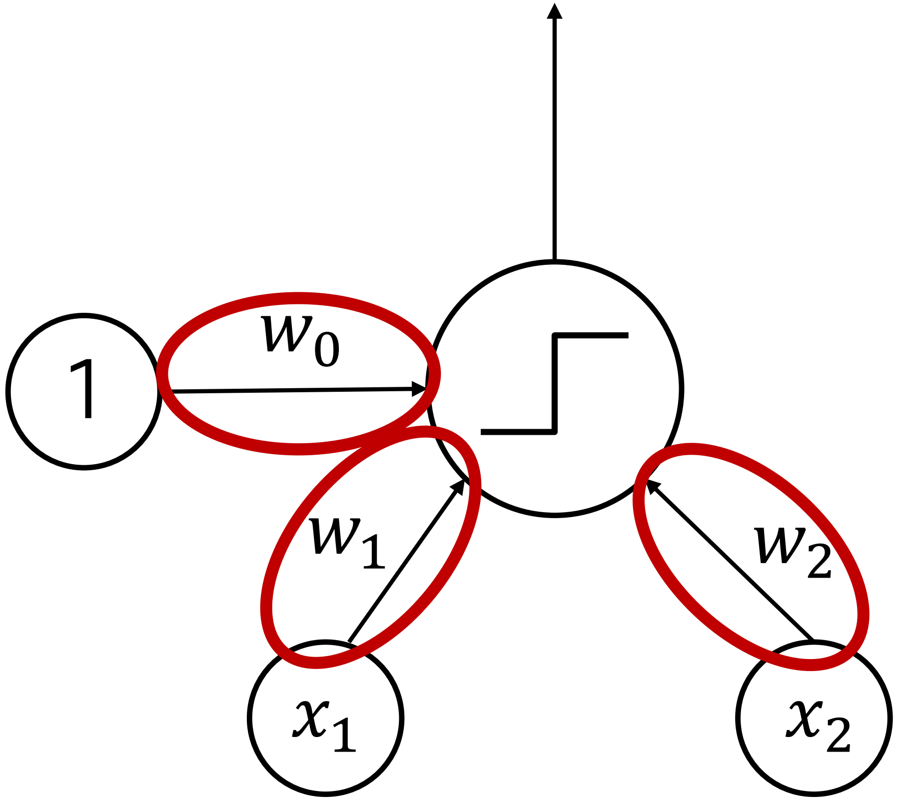
(a)

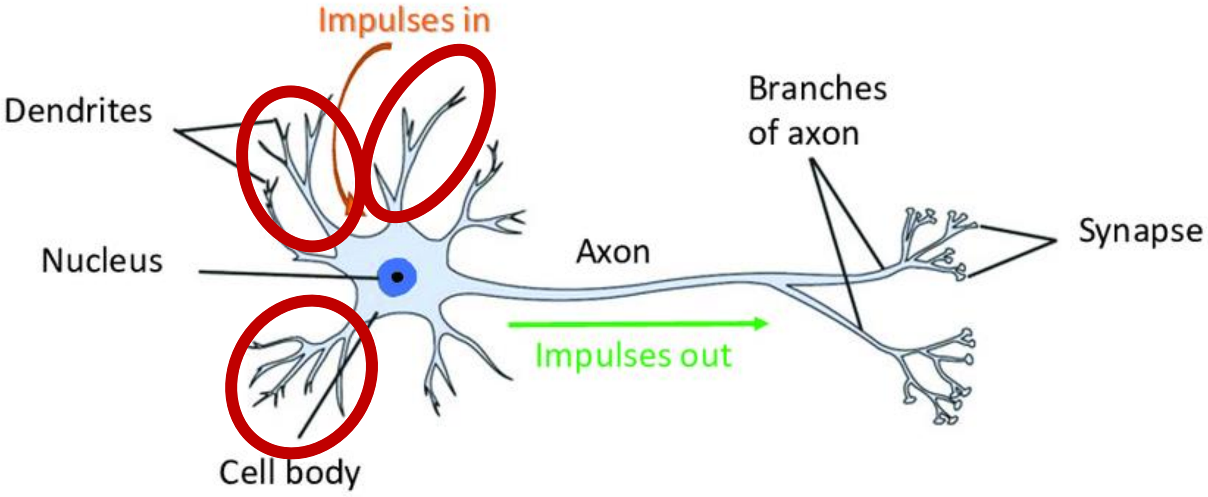
Input: x_1, x_2 Output: y

Weights: w_0, w_1, w_2

What is Perceptron?

$$y = f(w_0 + w_1 x_1 + w_2 x_2)$$





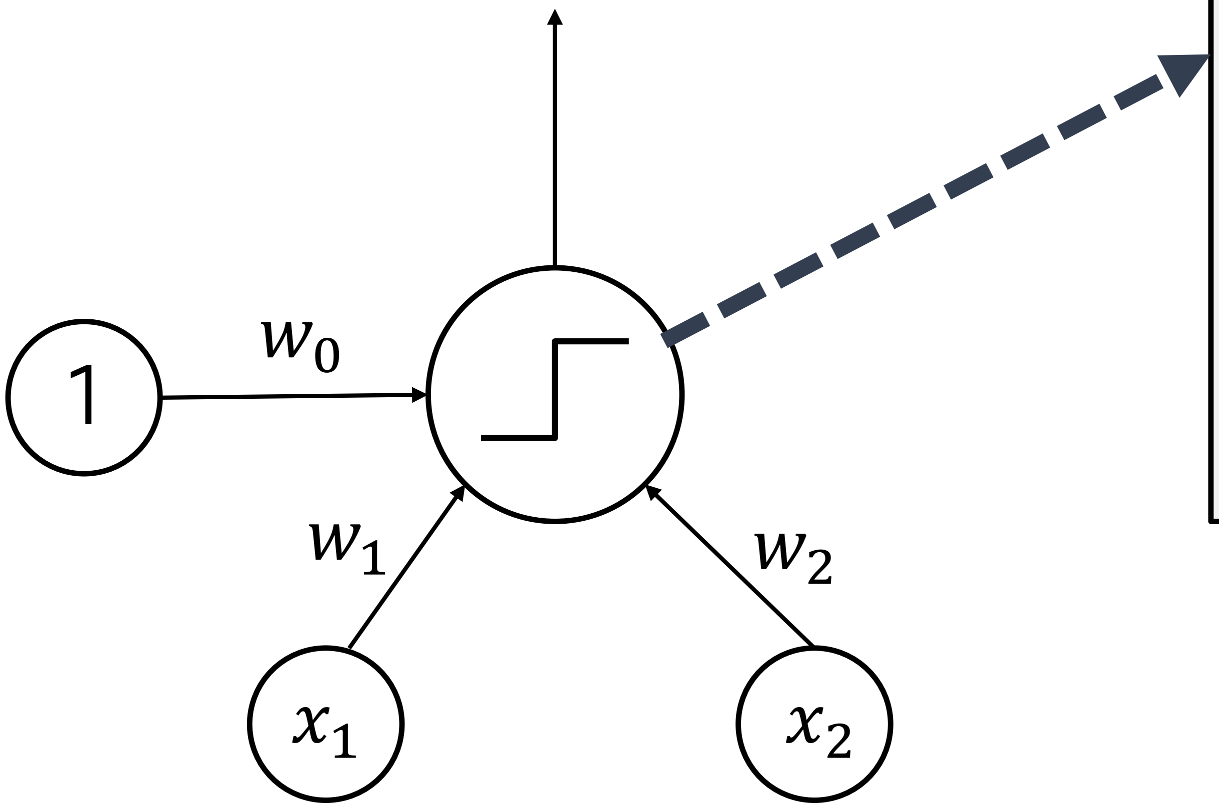
(a)

Input: x_1, x_2 Output: y

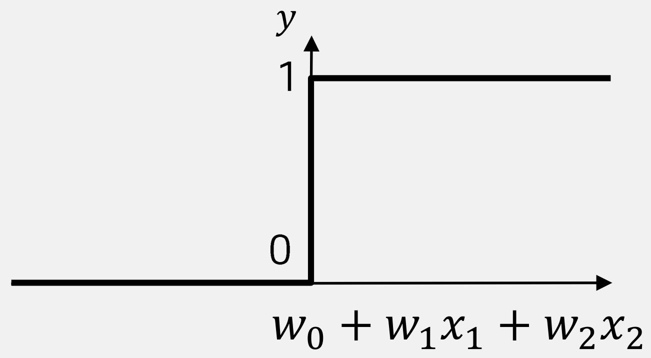
Weights: w_0, w_1, w_2

Single Layer Perceptron

$$y = f(w_0 + w_1 x_1 + w_2 x_2)$$



Hard thresholding function



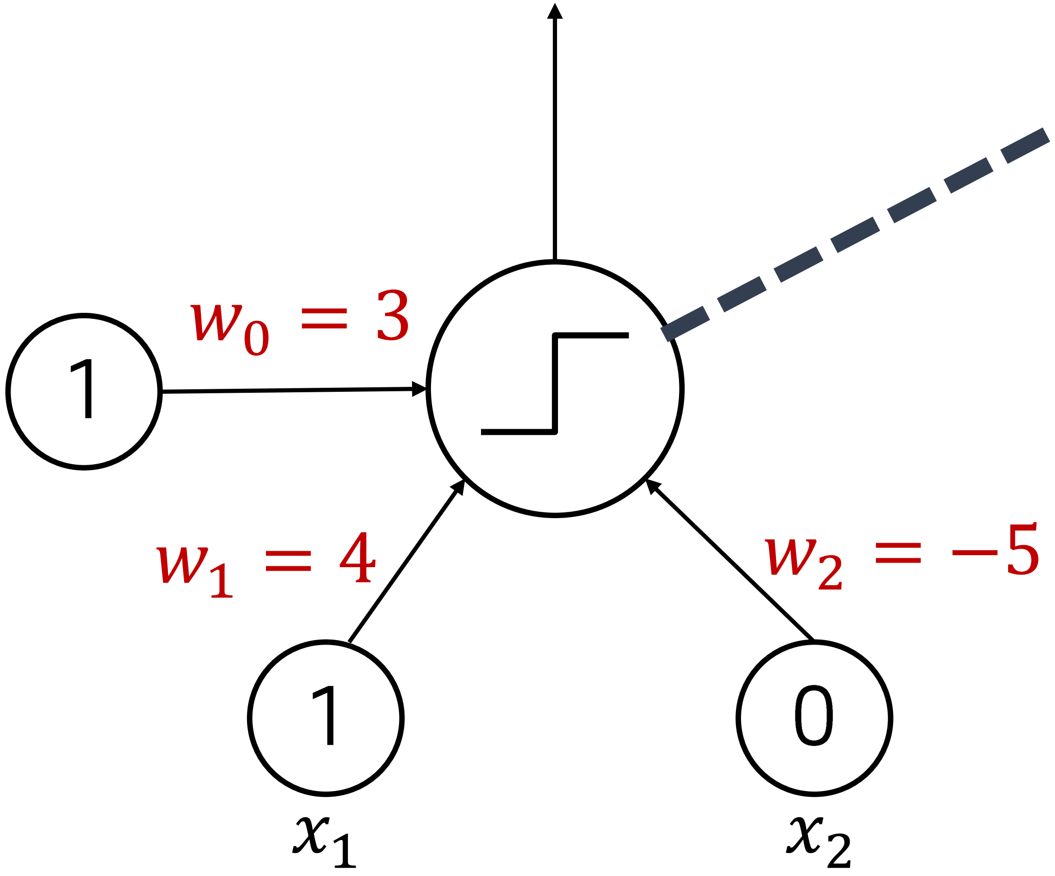
$$y = \begin{cases} 1 & w_0 + w_1 x_1 + w_2 x_2 \geq 0 \\ 0 & \text{otherwise} \end{cases}$$

Input: x_1, x_2 Output: y

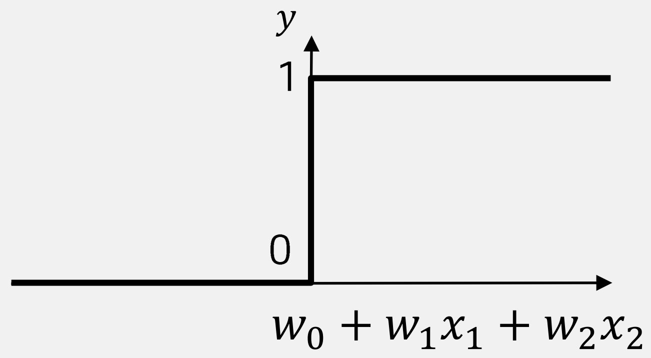
Weights: w_0, w_1, w_2

Single Layer Perceptron

$$y = f(\overset{w_0 \times 1}{3} + 4 \times 1 - 5 \times 0) = 1$$



Hard thresholding function

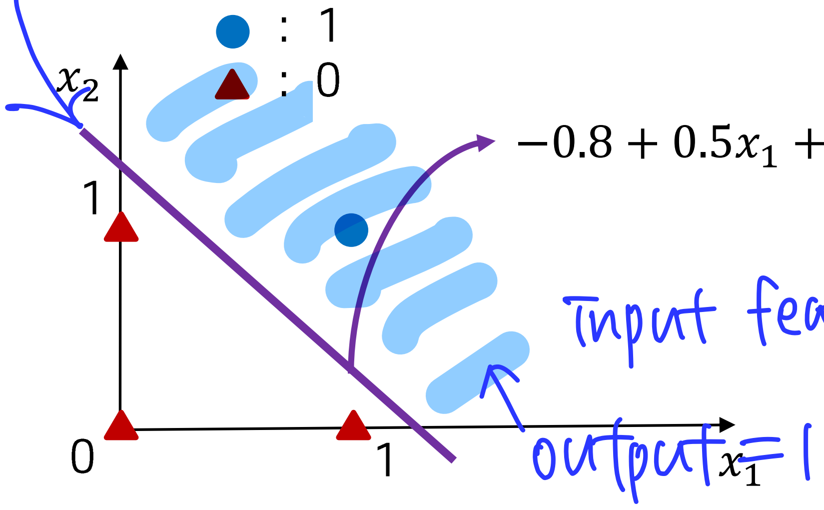


$$y = \begin{cases} 1 & w_0 + w_1x_1 + w_2x_2 \geq 0 \\ 0 & \text{otherwise} \end{cases}$$

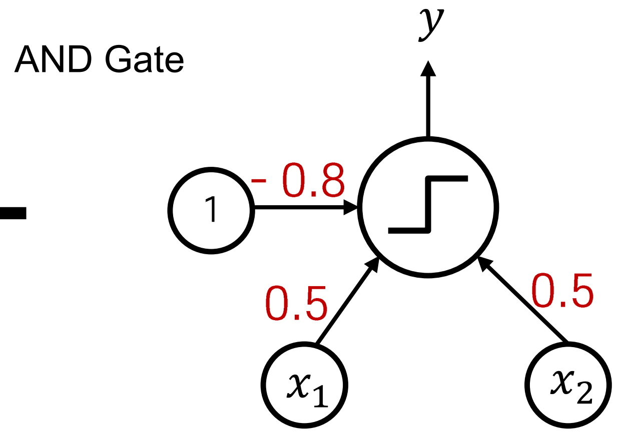
Input: x_1, x_2 Output: y

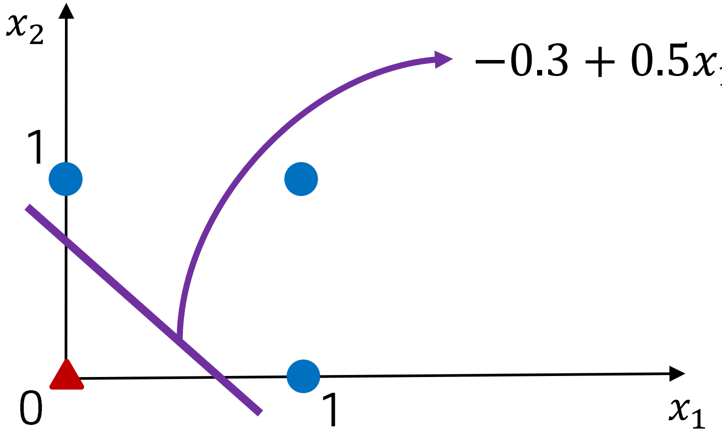
Weights: w_0, w_1, w_2

Decision Boundary in Perceptron

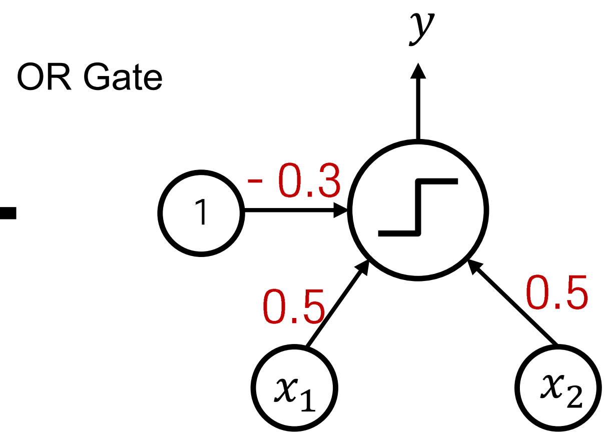


$$-0.8 + 0.5x_1 + 0.5x_2 = 0$$





$$-0.3 + 0.5x_1 + 0.5x_2 = 0$$

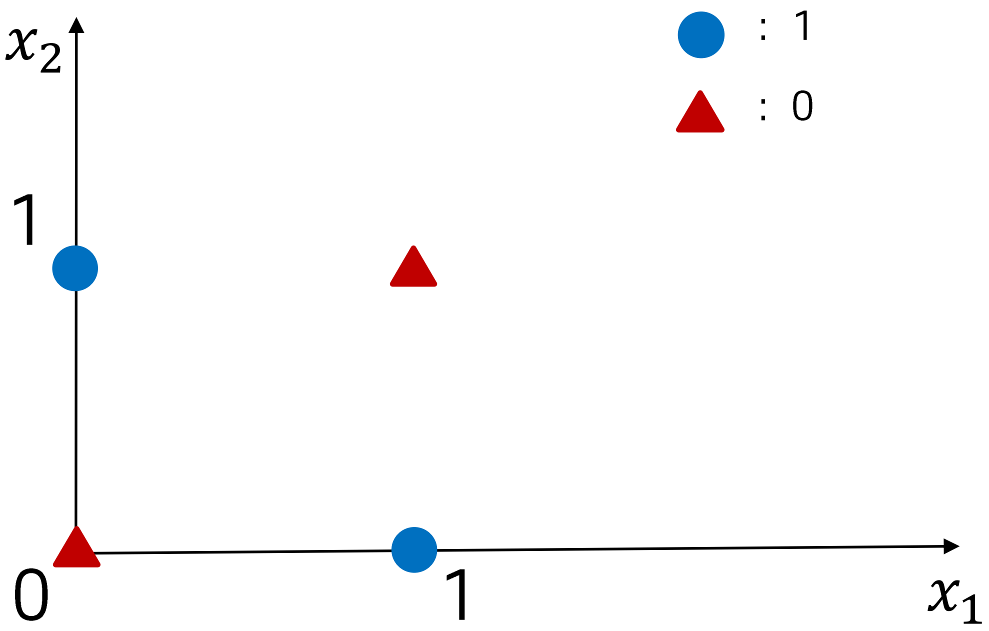


Multi-Layer Perceptron for XOR Gate

Is it possible to solve a XOR problem using a single layer perceptron?

→ No. Single layer perceptron can only solve linear problem. XOR problem is non-linear

$a + bx_1 + cx_2 = 0$

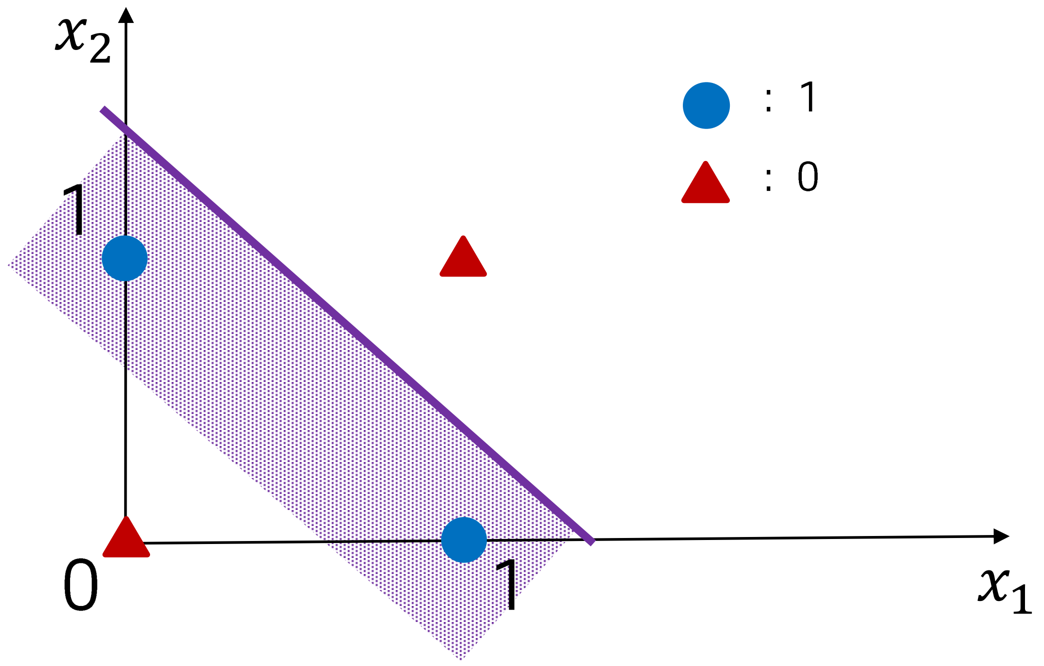


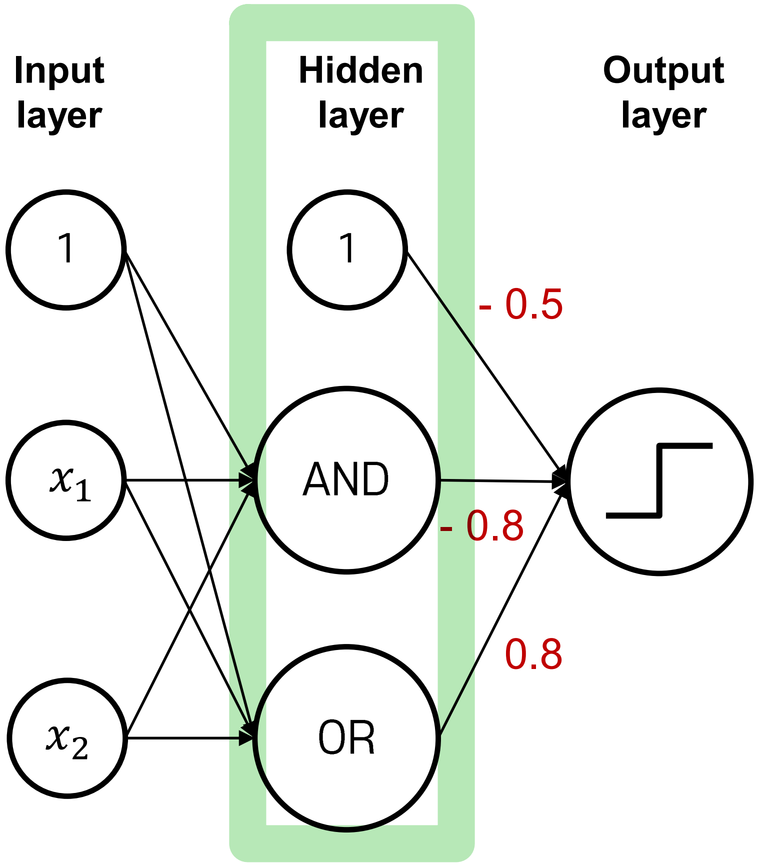
XOR Gate

x_1	x_2	y
0	0	0
0	1	1
1	0	1
1	1	0

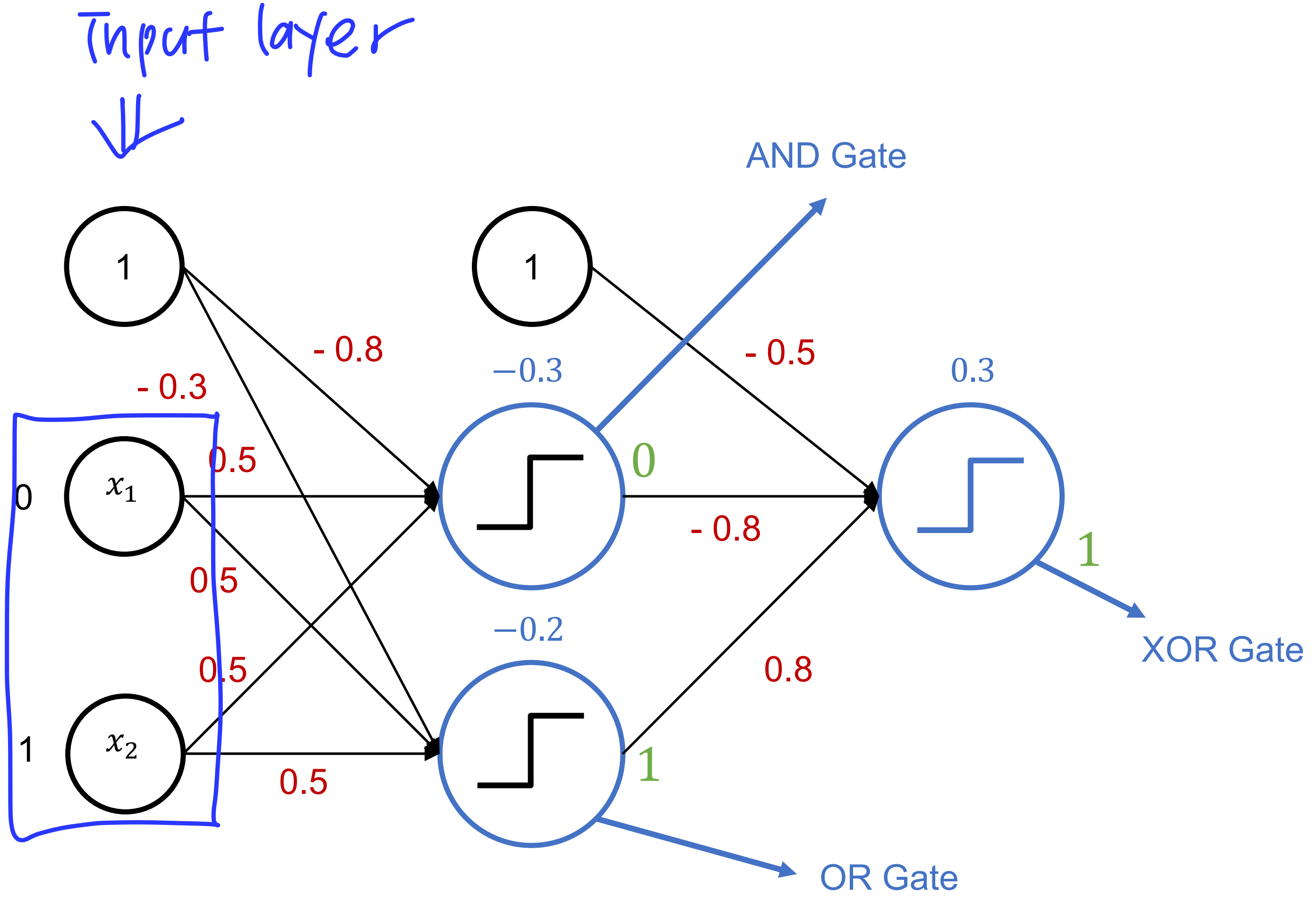
Multi-Layer Perceptron

But if we use two-layer perceptron, we can solve XOR problem
→ This model is called multi-layer perceptron





Multi-Layer Perceptron



AND Gate

x_1	x_2	y
0	0	0
0	1	0
1	0	0
1	1	1

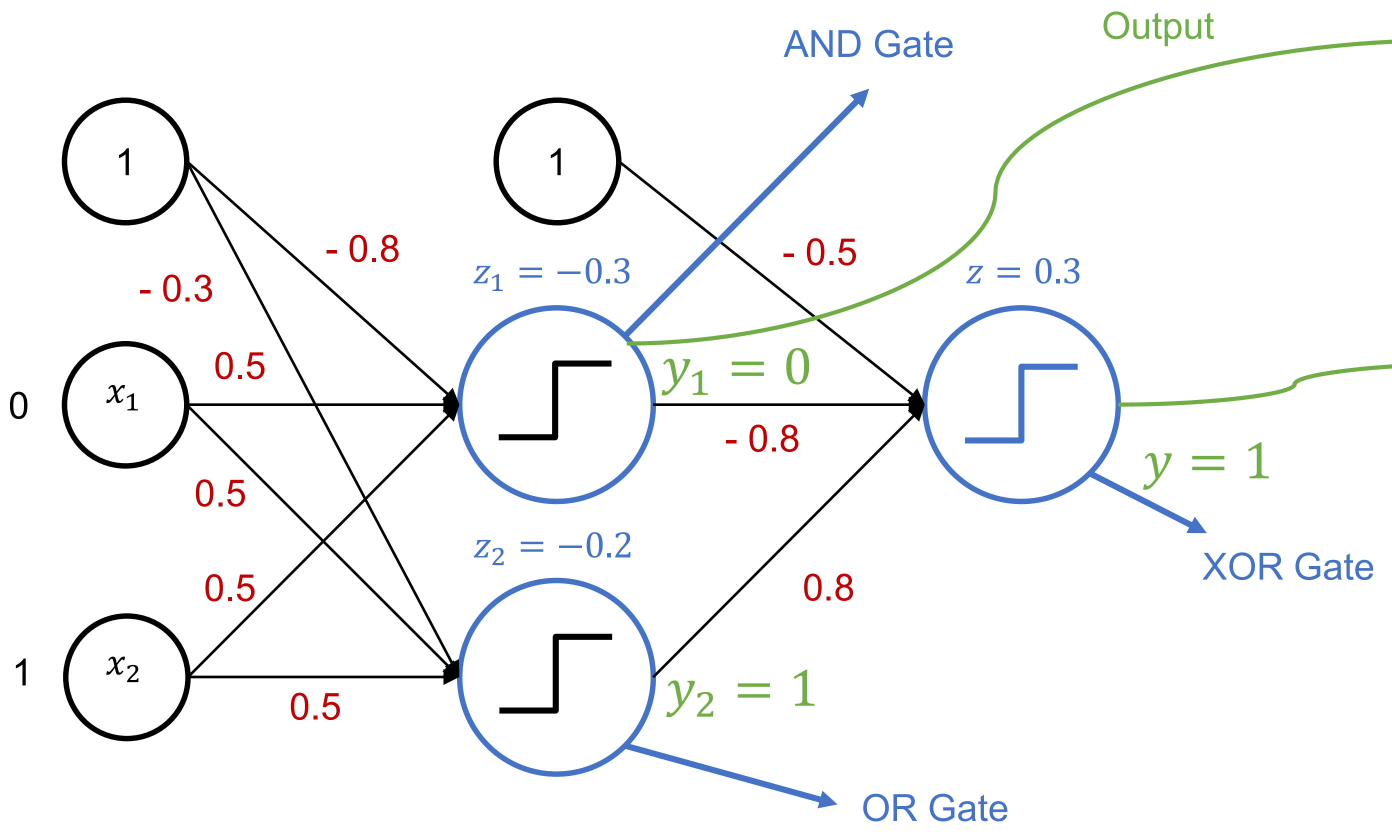
OR Gate

x_1	x_2	y
0	0	0
0	1	1
1	0	1
1	1	1

XOR Gate

x_1	x_2	y
0	0	0
0	1	1
1	0	1
1	1	0

Multi-Layer Perceptron



$$z_1 = -0.8 + 0.5x_1 + 0.5x_2$$
$$z_2 = -0.3 + 0.5x_1 + 0.5x_2$$

Mapping

$$y_i = \begin{cases} 1 & z_i \geq 0 \\ 0 & z_i < 0 \end{cases}$$

$$z = -0.5 - 0.8y_1 + 0.8y_2$$

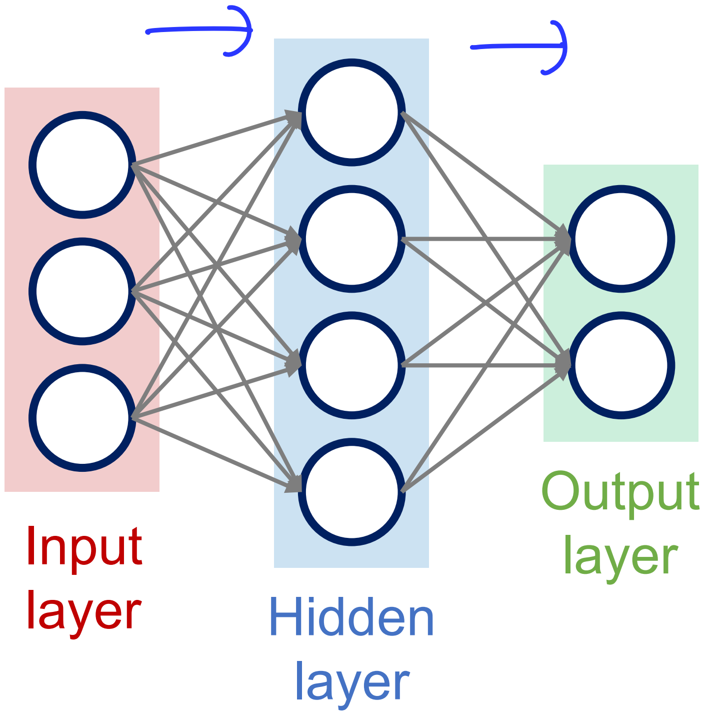
Mapping

$$y = \begin{cases} 1 & z \geq 0 \\ 0 & z < 0 \end{cases}$$

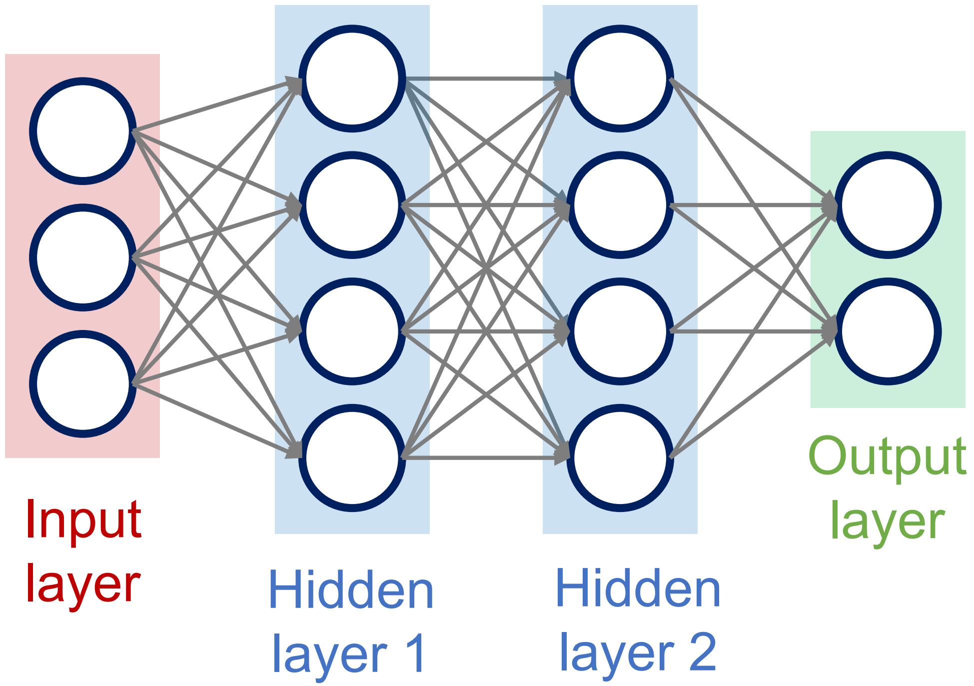
$$y$$

Hidden Layer

2-layer Neural Network
or
1-hidden–layer Neural Network

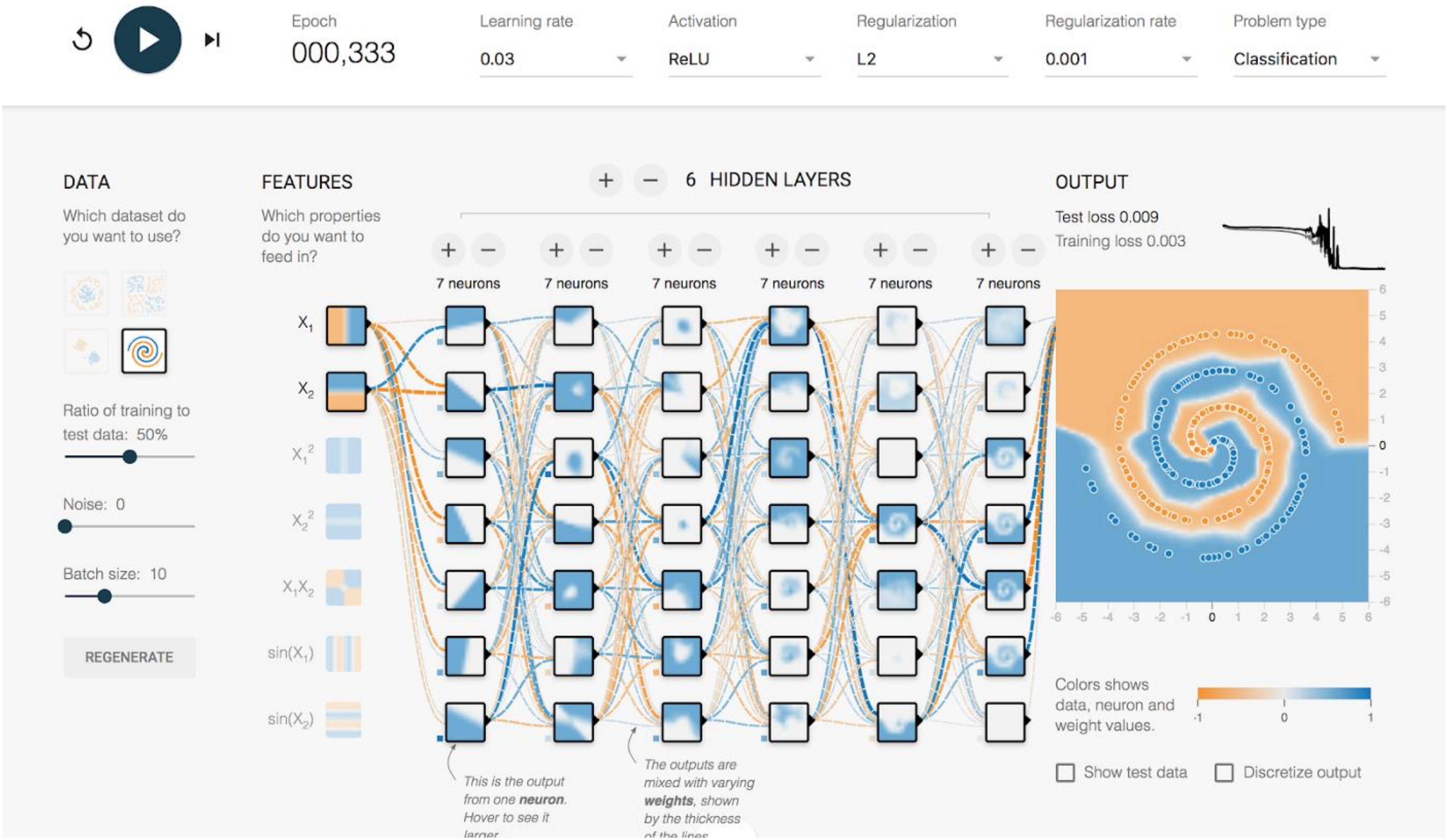


3-layer Neural Network
or
2–hidden–layer Neural Network



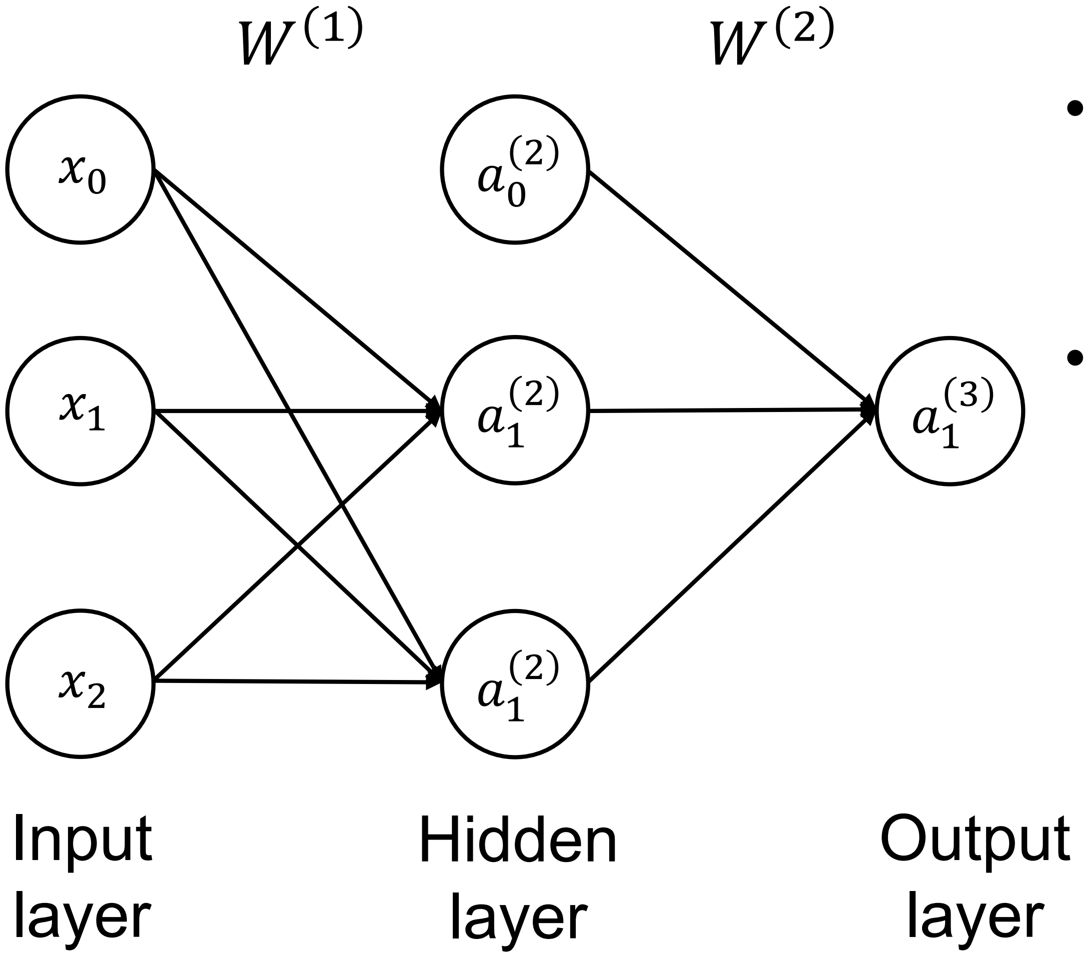
Tensorflow Playground

https://playground.tensorflow.org/



Forward Propagation

Forward Propagation



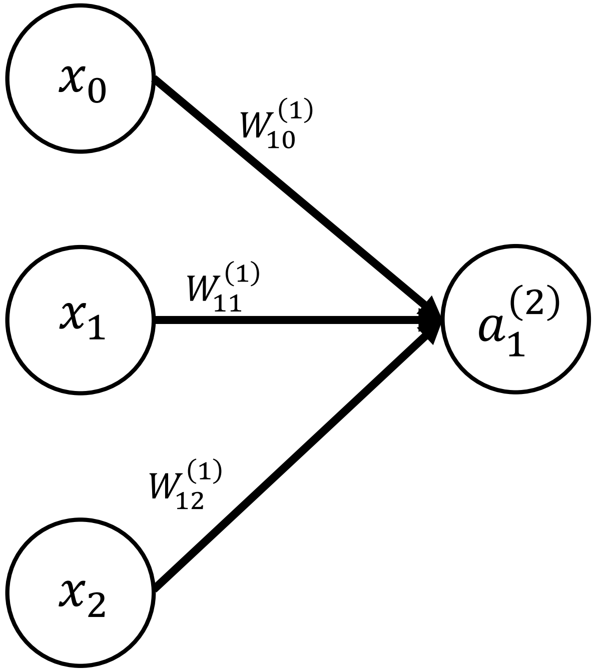
몇번째 노드?

- $a_j^{(i)}$: **"Activation"** of the i-th unit in the j-th layer

layer

- $W^{(j)}$: **"Weight Matrix"** mapping from the j-th layer to the $(j+1)$-th layer

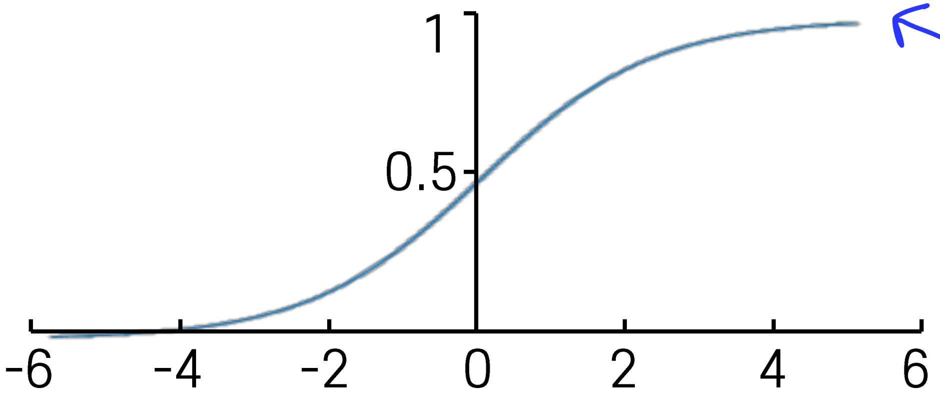
Forward Propagation



$$z_1^{(2)} = W_{10}^{(3)} x_0 + W_{11}^{(1)} + W_{12}^{(1)} x_2$$

$$= \begin{bmatrix} W_{10}^{(3)} & W_{11}^{(1)} & W_{12}^{(1)} \end{bmatrix} \begin{bmatrix} x_0 \\ x_1 \\ x_2 \end{bmatrix}$$

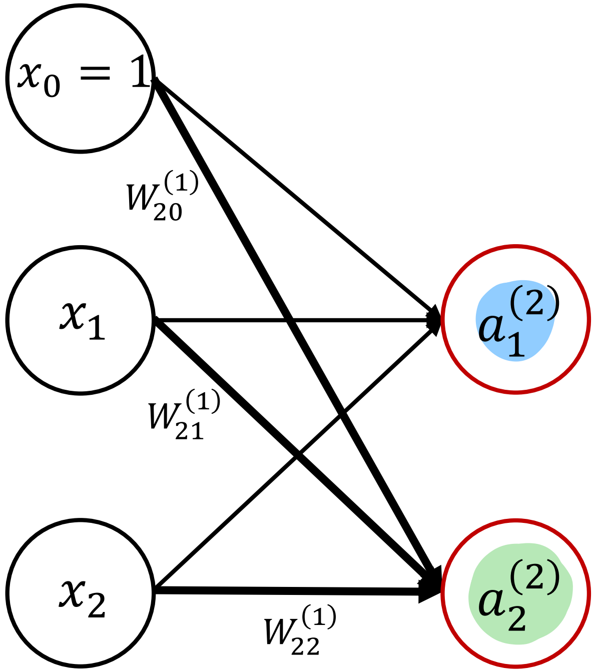
$$a_1^{(2)} = g(z_1^{(2)})$$



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

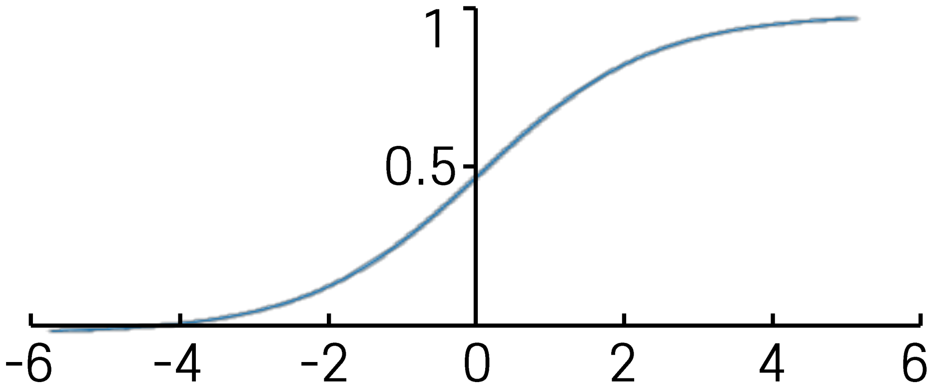
Logistic function
(Sigmoid function)

Forward Propagation



$$\begin{bmatrix} z_1^{(2)} \\ z_2^{(2)} \end{bmatrix} = \begin{bmatrix} W_{10}^{(3)} & W_{11}^{(1)} & W_{12}^{(1)} \\ W_{20}^{(3)} & W_{21}^{(1)} & W_{22}^{(1)} \end{bmatrix} \begin{bmatrix} x_0 \\ x_1 \\ x_2 \end{bmatrix}$$

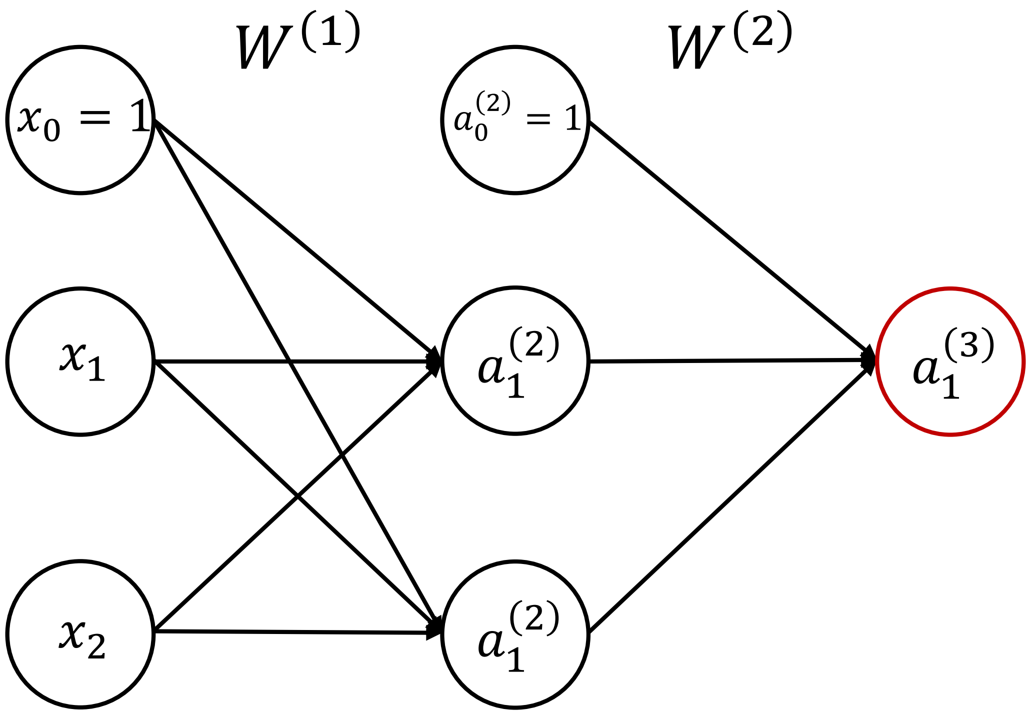
$$\boxed{\begin{bmatrix} a_1^{(2)} \\ a_2^{(2)} \end{bmatrix} = \begin{bmatrix} g\left(z_1^{(2)}\right) \\ g\left(z_2^{(2)}\right) \end{bmatrix}}$$



Logistic function
(Sigmoid function)

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

Forward Propagation



$$\begin{bmatrix} z_1^{(2)} \\ z_2^{(2)} \end{bmatrix} = \begin{bmatrix} W_{10}^{(3)} & W_{11}^{(1)} & W_{12}^{(1)} \\ W_{20}^{(3)} & W_{21}^{(1)} & W_{22}^{(1)} \end{bmatrix} \begin{bmatrix} x_0 \\ x_1 \\ x_2 \end{bmatrix}$$

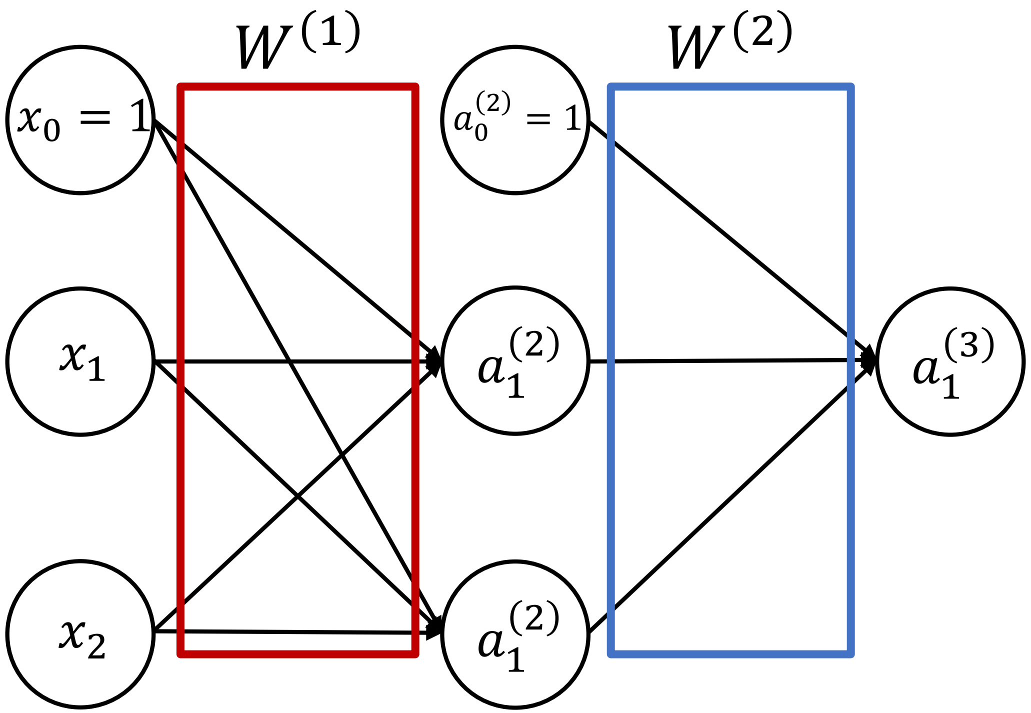
$$\begin{bmatrix} a_1^{(2)} \\ a_2^{(2)} \end{bmatrix} = \begin{bmatrix} g\left(z_1^{(2)}\right) \\ g\left(z_2^{(2)}\right) \end{bmatrix}$$

$$z_1^{(3)} = \begin{bmatrix} W_{20}^{(3)} & W_{21}^{(1)} & W_{22}^{(1)} \end{bmatrix} \begin{bmatrix} a_0^{(2)} \\ a_1^{(2)} \\ a_2^{(2)} \end{bmatrix}$$

$$\boxed{a_1^{(3)} = g\left(z_1^{(3)}\right)}$$

활성함수

Linear Layer



$$\begin{bmatrix} z_1^{(2)} \\ z_2^{(2)} \end{bmatrix} = \boxed{\begin{bmatrix} W_{10}^{(3)} & W_{11}^{(1)} & W_{12}^{(1)} \\ W_{20}^{(3)} & W_{21}^{(1)} & W_{22}^{(1)} \end{bmatrix}} \begin{bmatrix} x_0 \\ x_1 \\ x_2 \end{bmatrix}$$

$$z_1^{(3)} = \boxed{\begin{bmatrix} W_{20}^{(3)} & W_{21}^{(1)} & W_{22}^{(1)} \end{bmatrix}} \begin{bmatrix} a_0^{(2)} \\ a_1^{(2)} \\ a_2^{(2)} \end{bmatrix}$$

Each layer performs linear transformation, so it is also called a **linear layer**

Linear Layer and **Fully-connected Layer are the same thing.**

MNIST Dataset

MNIST (Modified National Institute of Standards and Technology)

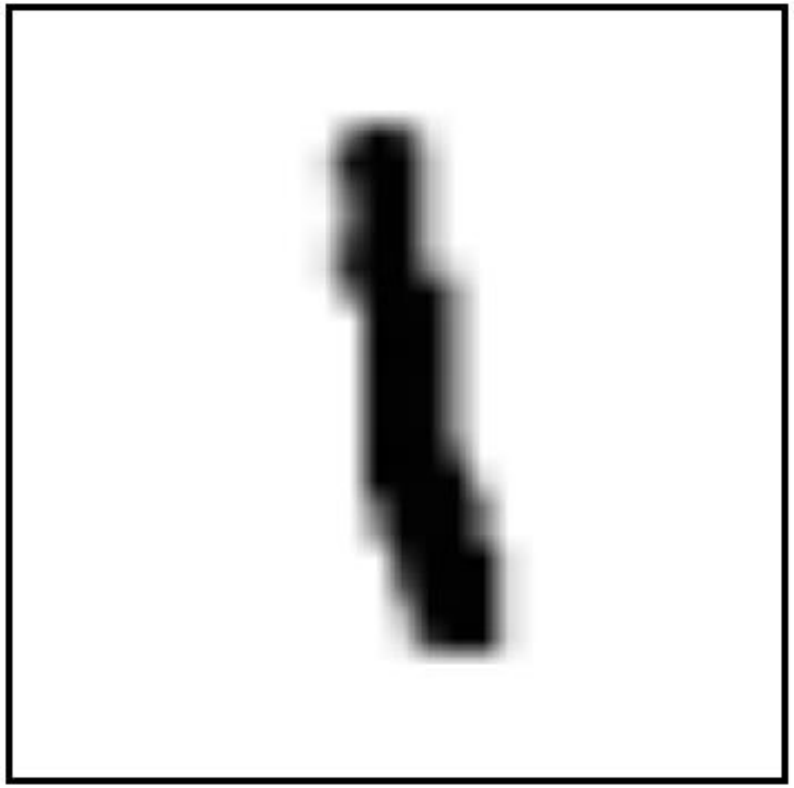
28×28

- Handwritten digits from 0 to 9
  - 55,000 training examples
  - 10,000 testing examples
- Each image has been preprocessed
  - Digits are center-aligned
  - Digit size is rescaled to similar size
  - Each image has fixed size of 28×28

→ Real number matrix from 0.0 to 1.0

http://yann.lecun.com/exdb/mnist/

Example of MNIST

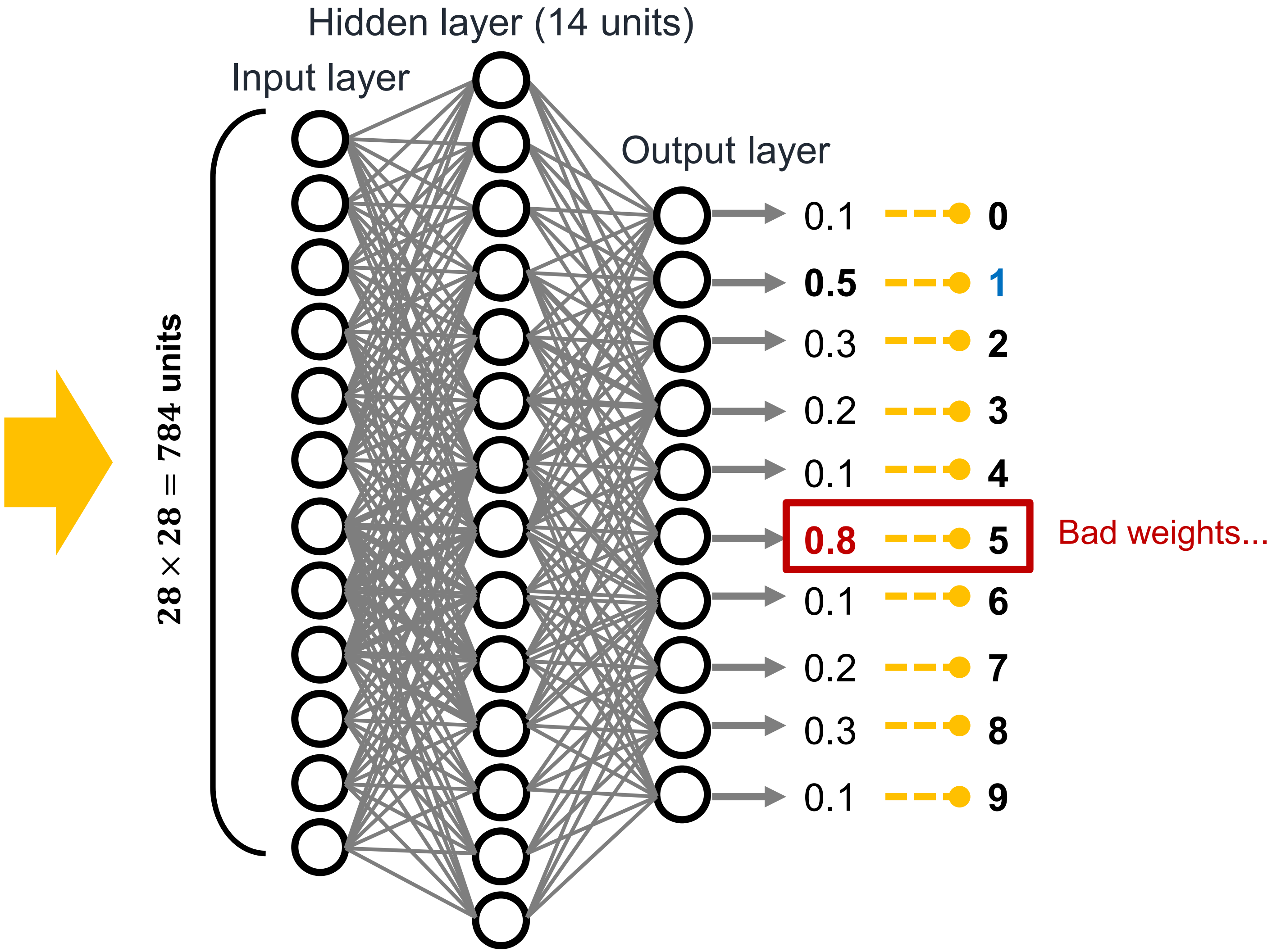


$$\approx \left.\underbrace{\begin{bmatrix}
0&0&0&0&0&0&0&0&0&0&0&0&0&0\\
0&0&0&0&0&0&0&0&0&0&0&0&0&0\\
0&0&0&0&0&0&.7&.1&0&0&0&0&0&0\\
0&0&0&0&0&0&.8&.2&0&0&0&0&0&0\\
0&0&0&0&0&0&.9&.3&0&0&0&0&0&0\\
0&0&0&0&0&0&.4&.5&0&0&0&0&0&0\\
0&0&0&0&0&0&.4&.5&0&0&0&0&0&0\\
0&0&0&0&0&0&.4&.6&0&0&0&0&0&0\\
0&0&0&0&0&0&.2&.9&0&0&0&0&0&0\\
0&0&0&0&0&0&0&1&.4&0&0&0&0&0\\
0&0&0&0&0&0&0&.8&.5&0&0&0&0&0\\
0&0&0&0&0&0&0&.3&.4&0&0&0&0&0\\
0&0&0&0&0&0&0&0&0&0&0&0&0&0\\
0&0&0&0&0&0&0&0&0&0&0&0&0&0
\end{bmatrix}}_{28}\right\} 28$$

http://yann.lecun.com/exdb/mnist/

MNIST Classification Model

$$
\begin{bmatrix}
0&0&0&0&0&0&0&0&0&0&0&0&0&0\\
0&0&0&0&0&0&0&0&0&0&0&0&0&0\\
0&0&0&0&0&0&.7&.1&0&0&0&0&0&0\\
0&0&0&0&0&0&.8&.2&0&0&0&0&0&0\\
0&0&0&0&0&0&.9&.3&0&0&0&0&0&0\\
0&0&0&0&0&0&.4&.5&0&0&0&0&0&0\\
0&0&0&0&0&0&.4&.5&0&0&0&0&0&0\\
0&0&0&0&0&0&.4&.6&0&0&0&0&0&0\\
0&0&0&0&0&0&.2&.9&0&0&0&0&0&0\\
0&0&0&0&0&0&0&1&.4&0&0&0&0&0\\
0&0&0&0&0&0&0&.8&.5&0&0&0&0&0\\
0&0&0&0&0&0&0&.3&.4&0&0&0&0&0\\
0&0&0&0&0&0&0&0&0&0&0&0&0&0\\
0&0&0&0&0&0&0&0&0&0&0&0&0&0
\end{bmatrix}
$$



MNIST Classification Model

$$
\begin{bmatrix}
0&0&0&0&0&0&0&0&0&0&0&0&0&0\\
0&0&0&0&0&0&0&0&0&0&0&0&0&0\\
0&0&0&0&0&0&.7&.1&0&0&0&0&0&0\\
0&0&0&0&0&0&.8&.2&0&0&0&0&0&0\\
0&0&0&0&0&0&.9&.3&0&0&0&0&0&0\\
0&0&0&0&0&0&.4&.5&0&0&0&0&0&0\\
0&0&0&0&0&0&.4&.5&0&0&0&0&0&0\\
0&0&0&0&0&0&.4&.6&0&0&0&0&0&0\\
0&0&0&0&0&0&.2&.9&0&0&0&0&0&0\\
0&0&0&0&0&0&0&1&.4&0&0&0&0&0\\
0&0&0&0&0&0&0&.8&.5&0&0&0&0&0\\
0&0&0&0&0&0&0&.3&.4&0&0&0&0&0\\
0&0&0&0&0&0&0&0&0&0&0&0&0&0\\
0&0&0&0&0&0&0&0&0&0&0&0&0&0
\end{bmatrix}
$$

$$
\left(\text{Prediction} - \text{Target} \right)^2 =
$$

Prediction		Target		=	
0.1	**0**	0	**0**	0.01	**0**
0.5	**1**	1	**1**	0.25	**1**
0.3	**2**	0	**2**	0.09	**2**
0.2	**3**	0	**3**	0.04	**3**
0.1	**4**	0	**4**	0.01	**4**
0.8	**5**	0	**5**	0.64	**5**
0.1	**6**	0	**6**	0.01	**6**
0.2	**7**	0	**7**	0.04	**7**
0.3	**8**	0	**8**	0.09	**8**
0.1	**9**	0	**9**	0.01	**9**

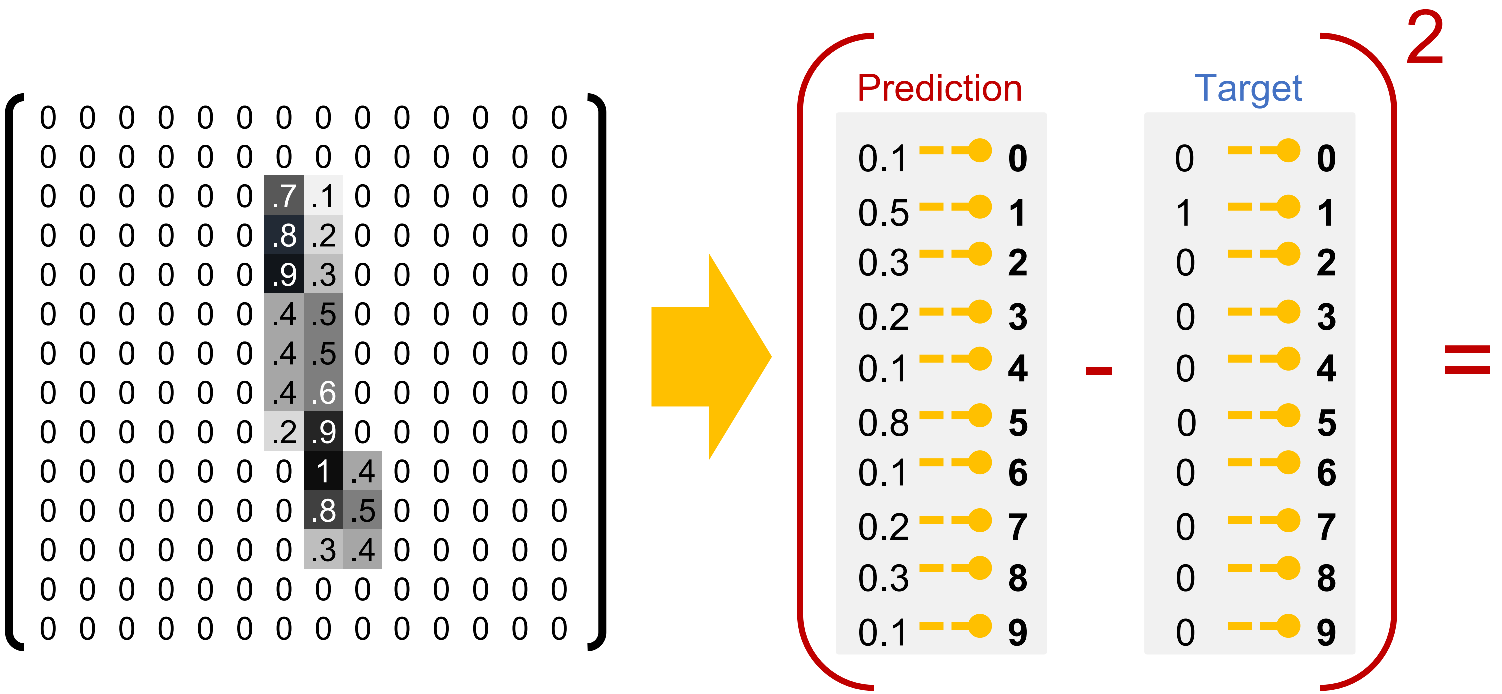
Squared Error: $\boxed{1.19}$

Let's find **the weight** which minimizes error!

ground truth 결과로
loss function

Softmax Layer (Softmax Classifier)

Problem of Sigmoid Outputs and Mean-Squared Error Loss



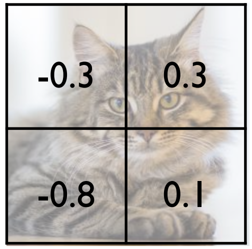
- Because of sigmoid outputs, Prediction $\in (0,1)$ & Target $\in \{0,1\}$

→ **Upper limits exist on loss and gradient magnitude with MSE Loss**

$$\max L = \max_{y_i \in \{0,1\}, \hat{y}_i \in (0,1)} \sum_{i=1}^{n} (\hat{y}_i - y_i)^2 < 1, \qquad \max \left| \frac{\partial L}{\partial \hat{y}_i} \right| < 2$$

- In addition, a better output would be a sum-to-one probability vector over multiple possible classes.

Softmax Layer (or Softmax Classifier) for Multi-Class Classification



3×5

	bias				
$\mathbf{w}_1^T$	1.1	-0.5	0.1	2	0.2
$\mathbf{w}_2^T$	3.2	1.3	2.1	1	1.5
$\mathbf{w}_3^T$	-1.2	0.3	0.7	-1.3	-0.2

$W^\top$

$\mathbf{x}$
1
-0.3
0.3
-0.8
0.1

softmax

Logit			Probability	
1.01	$e^{\mathbf{w}_1^T\mathbf{x}}$	2.75	$\frac{e^{\mathbf{w}_1^T\mathbf{x}}}{\sum_c e^{\mathbf{w}_c^T\mathbf{x}}}$	0.36 Dog
1.56	$e^{\mathbf{w}_2^T\mathbf{x}}$	4.76	$\frac{e^{\mathbf{w}_2^T\mathbf{x}}}{\sum_c e^{\mathbf{w}_c^T\mathbf{x}}}$	0.62 Cat
-1.74	$e^{\mathbf{w}_3^T\mathbf{x}}$	0.18	$\frac{e^{\mathbf{w}_3^T\mathbf{x}}}{\sum_c e^{\mathbf{w}_c^T\mathbf{x}}}$	0.02 Frog
$\in (-\infty, \infty)$		$\in (0, \infty)$		$\in (0,1)$

$e^{W^\top \mathbf{x}}$

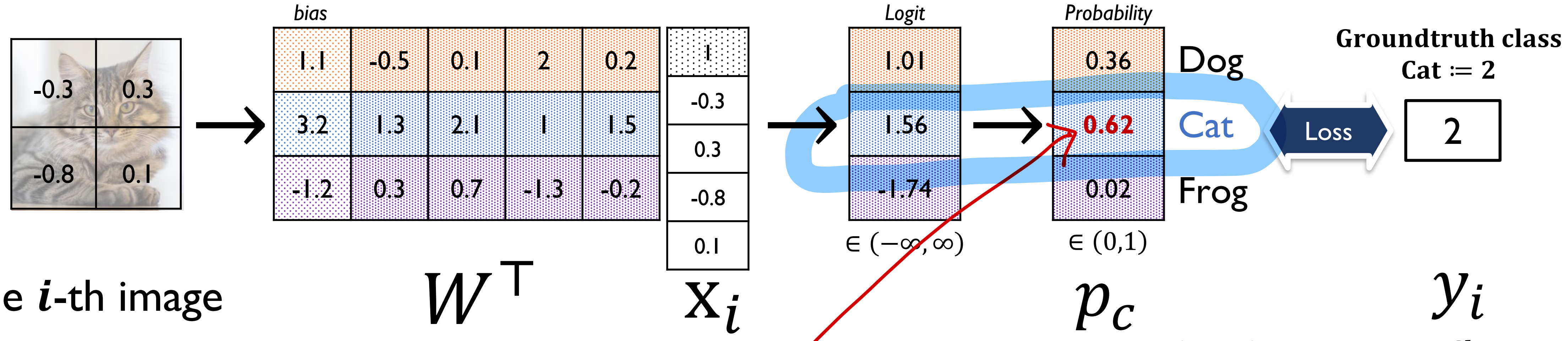
지수함수 통과시킴.

⊕값

상대적비율계산

- The softmax layer applies a monotonically increasing, exponential function to a logit vector:
  - Map the value in $(-\infty, \infty)$ to $(0, \infty)$
  - Preserve the order of values
- Calculate the relative proportions with respect to the sum of these positive values, resulting in a sum-to-one probability vector

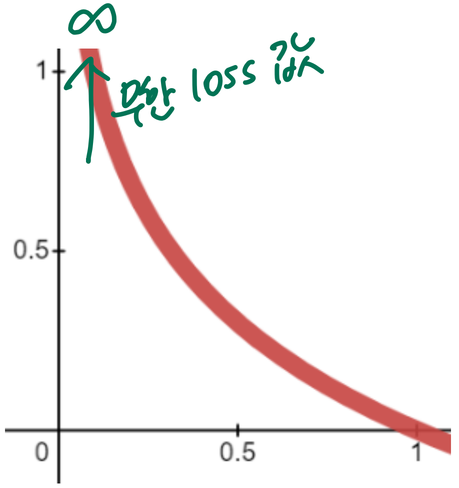
Softmax Loss (or Cross-Entropy Loss)



- **Softmax loss**, also known as **cross-entropy loss** or **negative log-likelihood** (NLL) loss used for training a softmax classifier is defined as

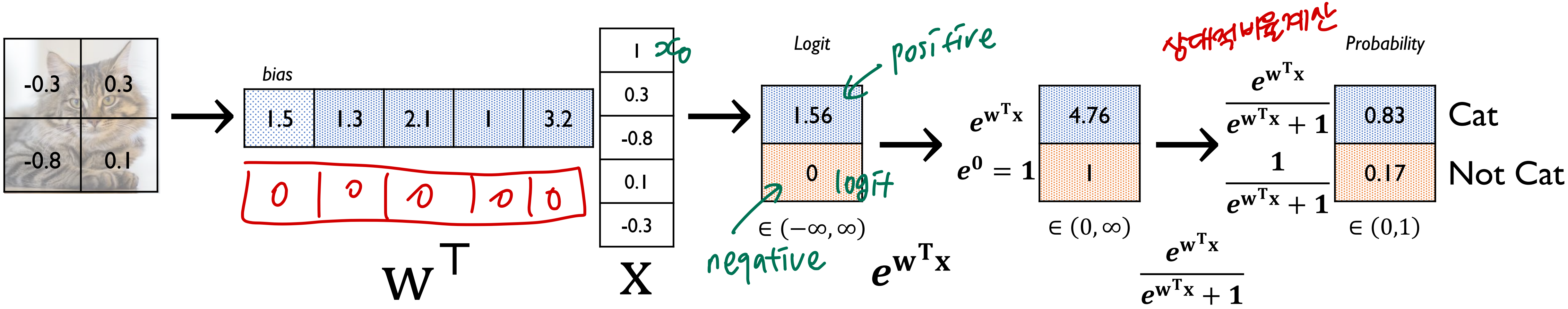
$$L = -\sum_{c=1}^{C} y_c \log(\hat{p}_c) = -\log(\hat{p}_{y_i})$$

ground truth vector



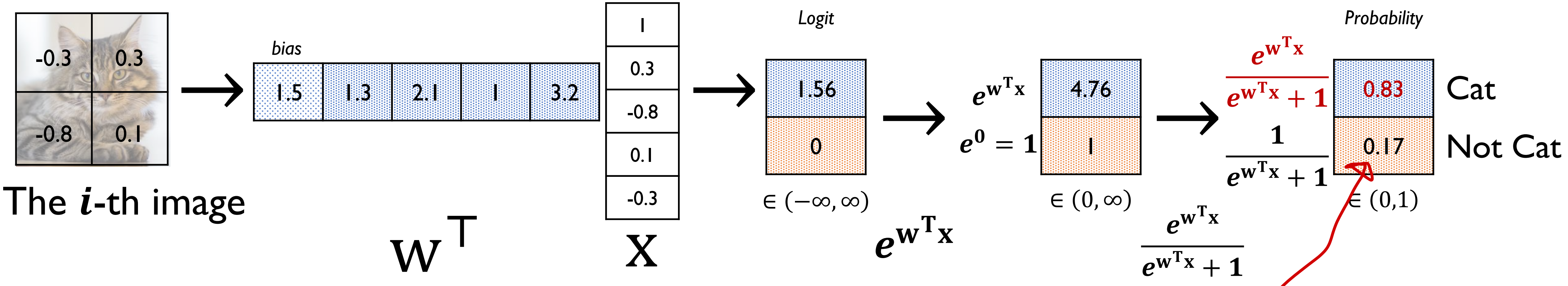
$-\log x$

Logistic Regression as a Special Case of Softmax Classifier



- Logistic regression ➔ Softmax classifier whose logit for a negative class is set as a constant value of 0.
- Logistic regression is used for a binary classification.
- The softmax classifier can also be used for two classes by using the matrix W with two columns, i.e., using the twice the number of parameters of a logistic regression.

Logistic Regression as a Special Case of Softmax Classifier



- Binary cross-entropy (BCE) loss for logistic regression is defined as

$$L = -\sum_{c=1}^{2} y_c \log(\hat{p}_c) = -y_i \log \hat{y}_i - (1 - y_i) \log(1 - \hat{y}_i)$$ ← BCE Loss

where $y_i = 1$ for a positive class, e.g., **cat**, and $y_i = 0$ for a negative class, e.g., **not cat**, and

https://machinelearningmastery.com/cross-entropy-for-machine-learning/