Chapter 08

# Artificial Neural Network

Sejong Oh

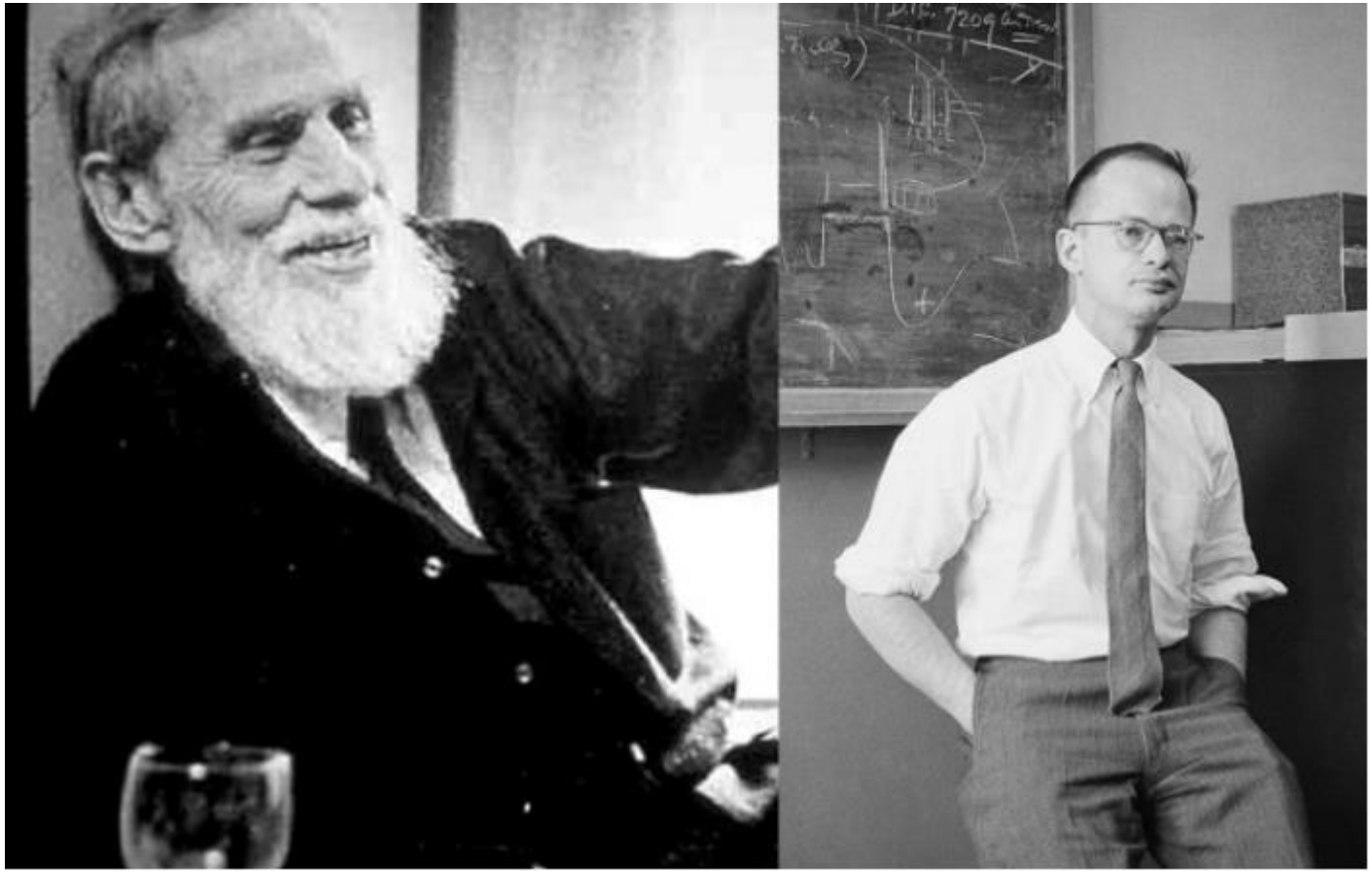
Bio Information technology Lab.

# Contents

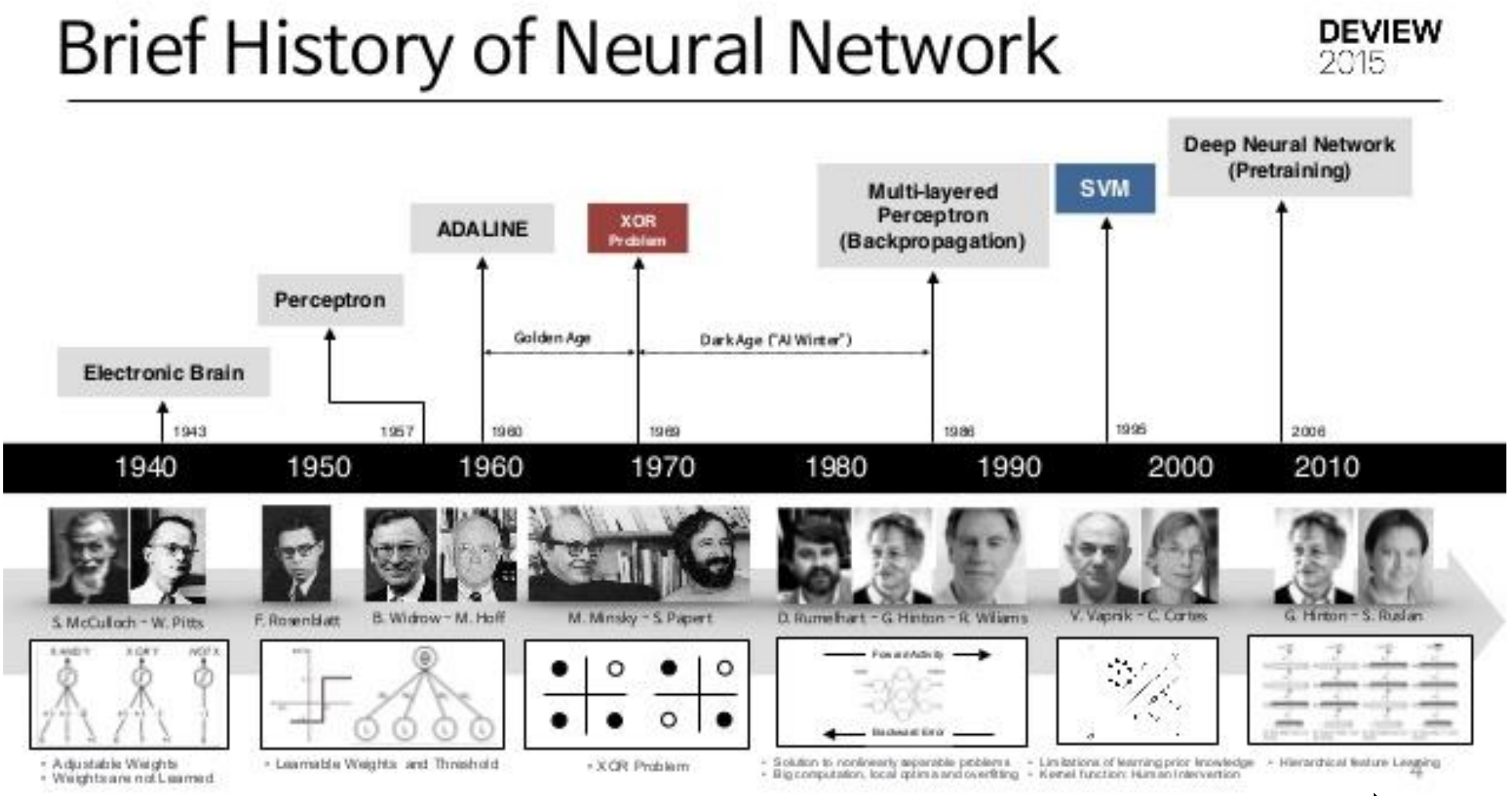
- History
- Terms & concept
- Perceptron
- AND logic example
- Limitation of simple perceptron

# 1. History

- In1943, Warren McCulloch & Walter Pitts
  - Suggest neural network



# 1. History




Deep learning

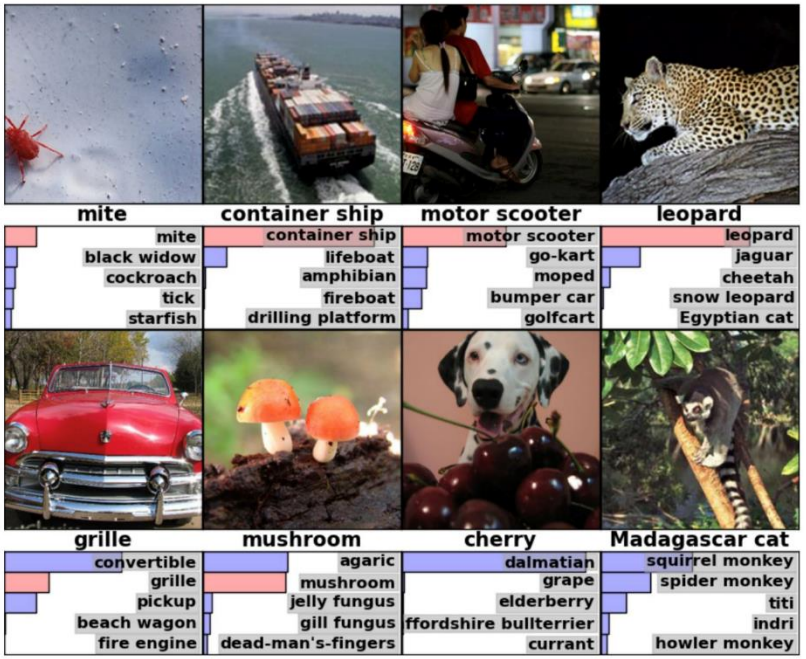
# 1. History

- Deep learning 의 성과 사례

## ImageNet Challenge

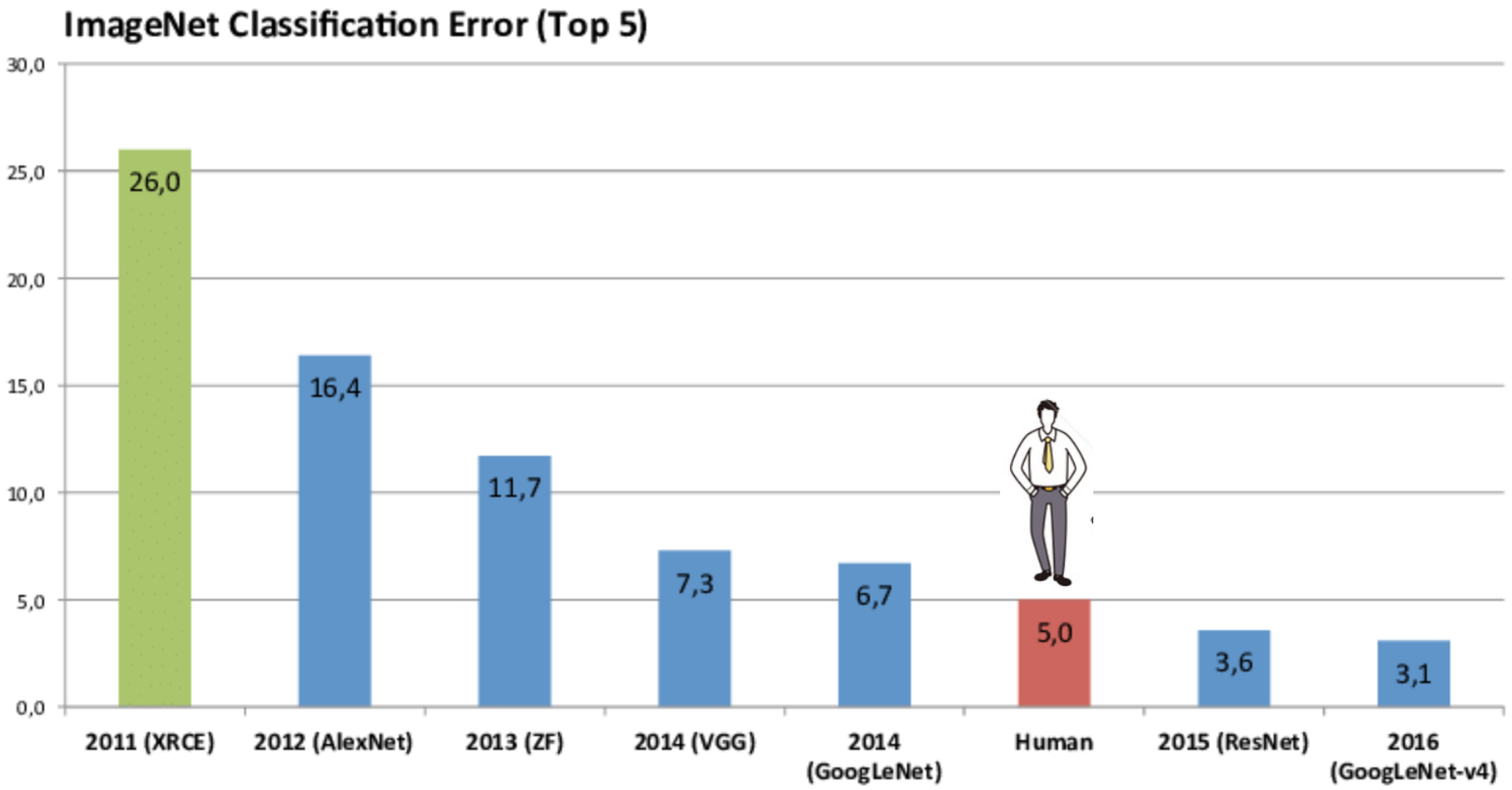
IMAGENET

- 1,000 object classes (categories).
- Images:
  - 1.2 M train
  - 100k test.




http://ruder.io/nlp-imagenet/

# 1. History




https://www.researchgate.net/figure/Winner-results-of-the-ImageNet-large-scale-visual-recognition-challenge-LSVRC-of-the_fig7_324476862

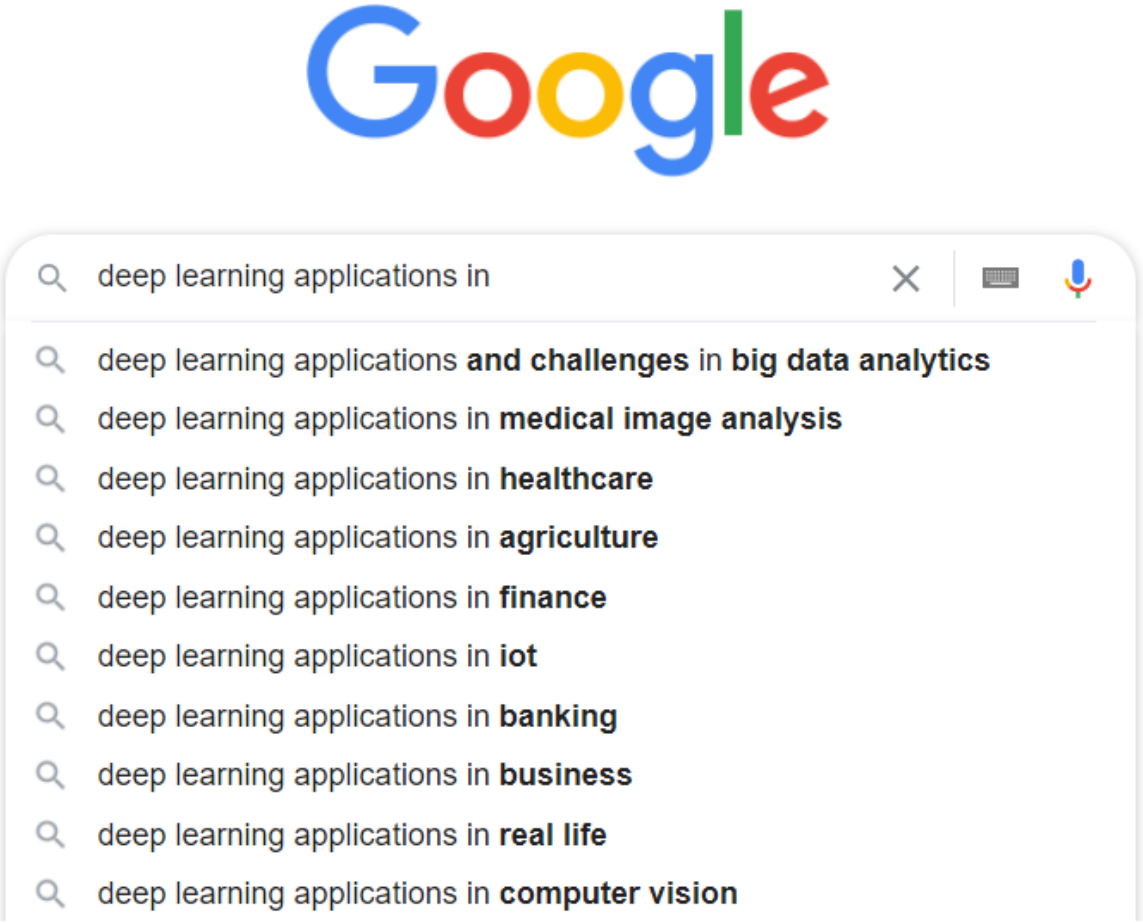
# 1. History

글로벌 기업 딥러닝 기술 경쟁

| | |
|---|---|
| 구글 | 4억달러에 '딥마인드' 인수, 제프리 힌튼 교수 영입 |
| 페이스북 | 얼굴 인지하는 '딥페이스' 프로젝트, 얀 르쿤 교수 영입 |
| 마이크로소프트 | 이미지 인식 연구 '프로젝트 아담' 진행 |
| 바이두 | 미국 실리콘밸리에 딥러닝 연구소 설립, 앤드루 응 교수 영입 |
| 트위터 | 사진분석 기업 '매드비츠' 인수 |
| 네이버 | 음성 검색에 딥러닝 기술 적용 |
| 카카오 | 김범수 의장이 딥러닝 스타트업 '클디'에 투자 |

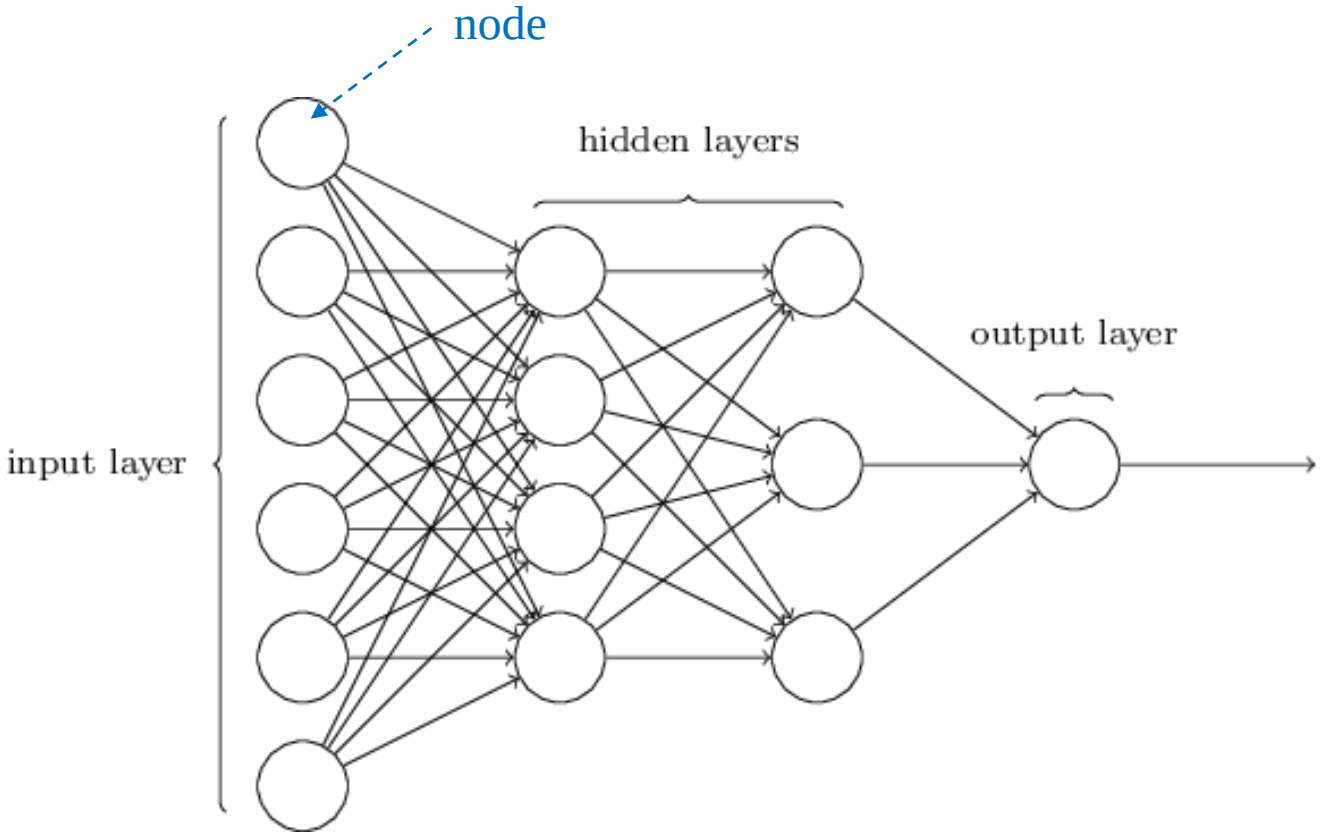
https://m.blog.naver.com/PostView.nhn?blogId=vinylx&logNo=220263868724&proxyReferer=https%3A%2F%2Fwww.google.co.kr%2F

# 1. History


Google
deep learning applications in
deep learning applications and challenges in big data analytics
deep learning applications in medical image analysis
deep learning applications in healthcare
deep learning applications in agriculture
deep learning applications in finance
deep learning applications in iot
deep learning applications in banking
deep learning applications in business
deep learning applications in real life
deep learning applications in computer vision

# 2. Terms & concept

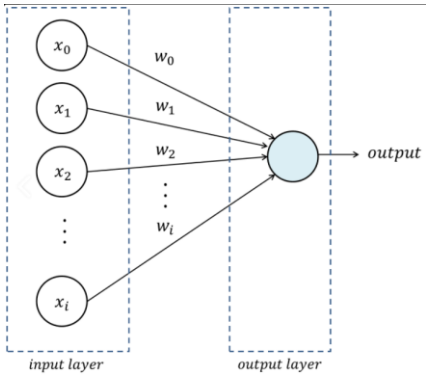
- Layers of neural network



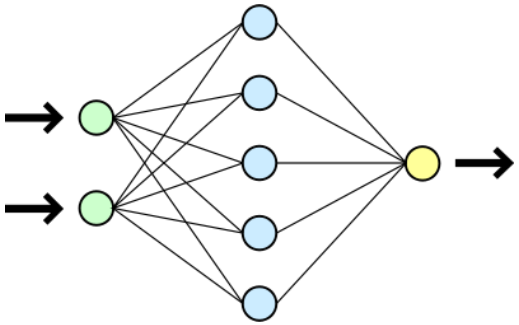

입력층 은닉층 출력층

# 2. Terms & concept

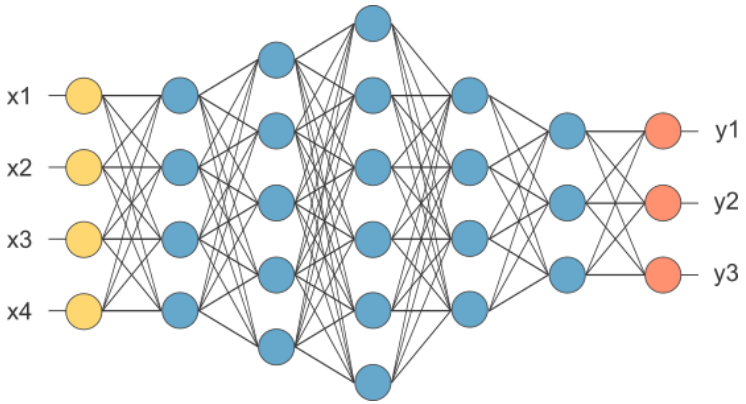
- Types of neural network




단층신경망



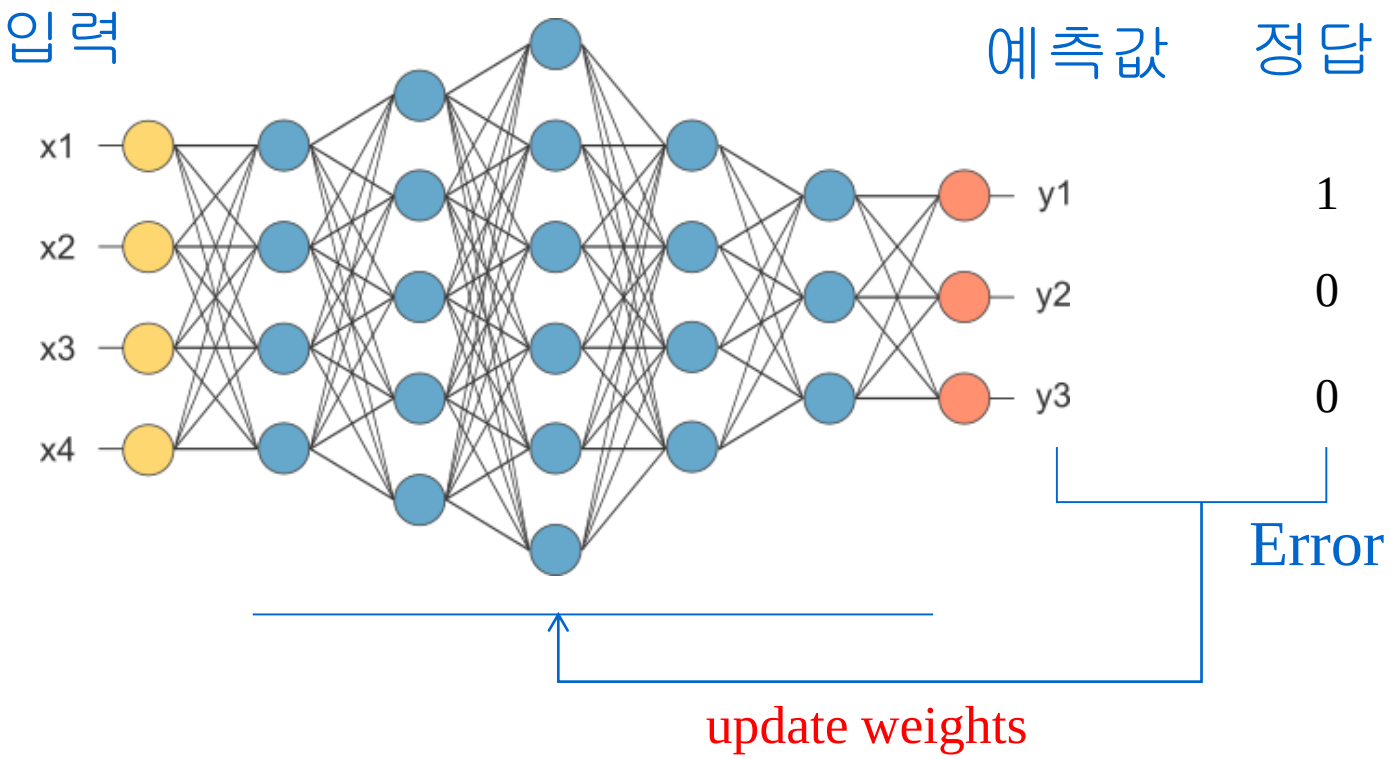
얕은 신경망




심층신경망 (Deep neural network)

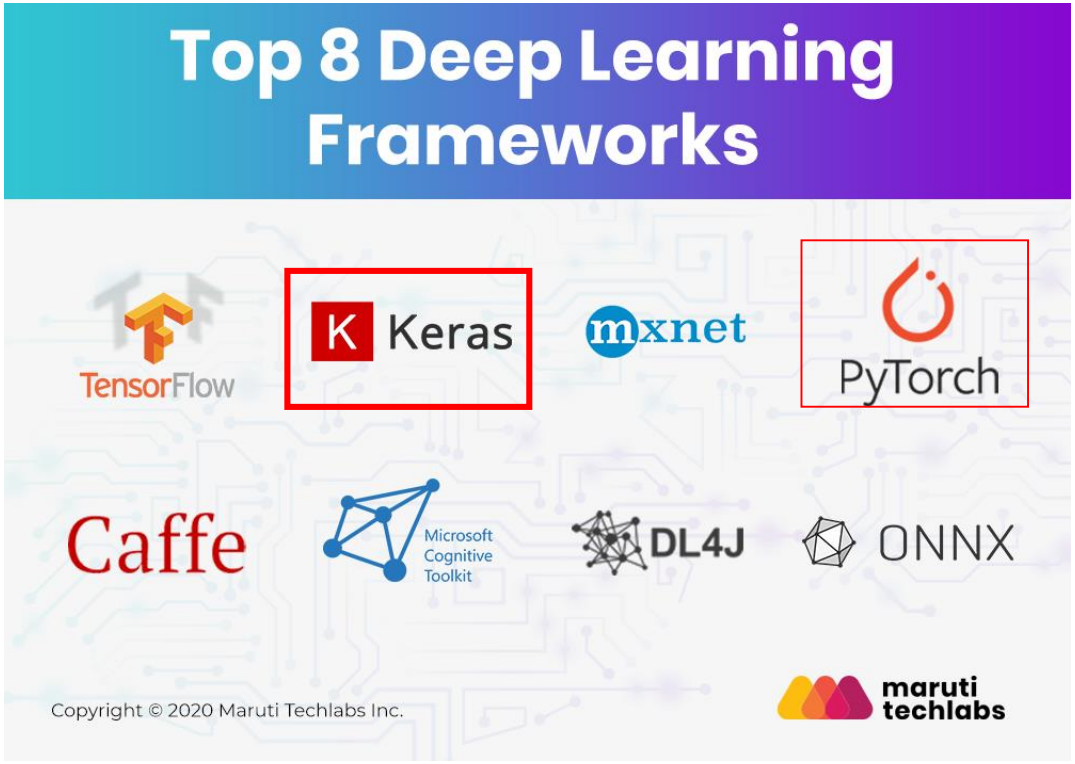
# 2. Terms & concept

- Learning in Neural network



# 2. Terms & concept

- Deep learning frameworks

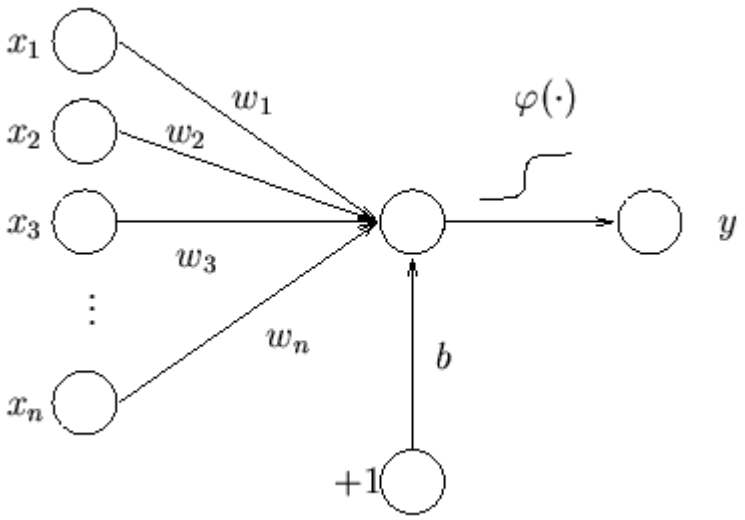



https://marutitech.com/top-8-deep-learning-frameworks/

인공신경망에 대한
기본 개념들을 이해하면
이런 패키지들을 이용해서
쉽게 문제를 해결할 수 있다

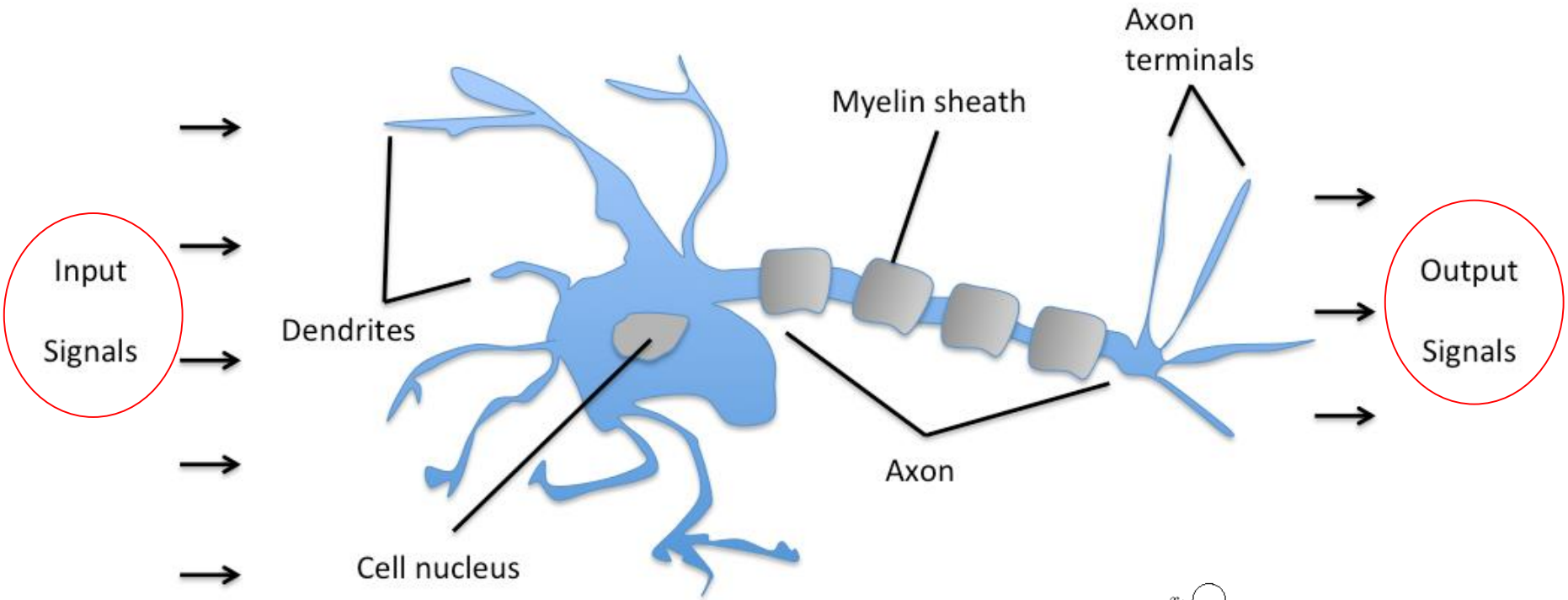
# 3. Perceptron

- Perceptron
  - 가장 단순한 형태의 신경망
  - Frank Rosenblatt 가 제안 (1957)
  - 다수의 신호를 입력으로 받아들여 하나의 (원하는) 신호를 출력하는 알고리즘

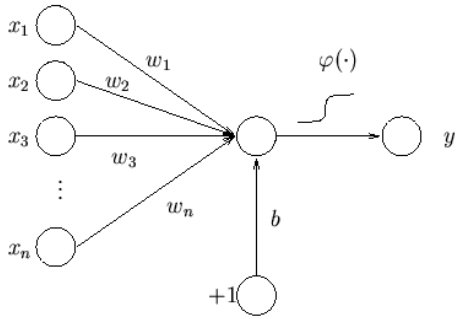


# 3. Perceptron

- 생물학적 뉴런

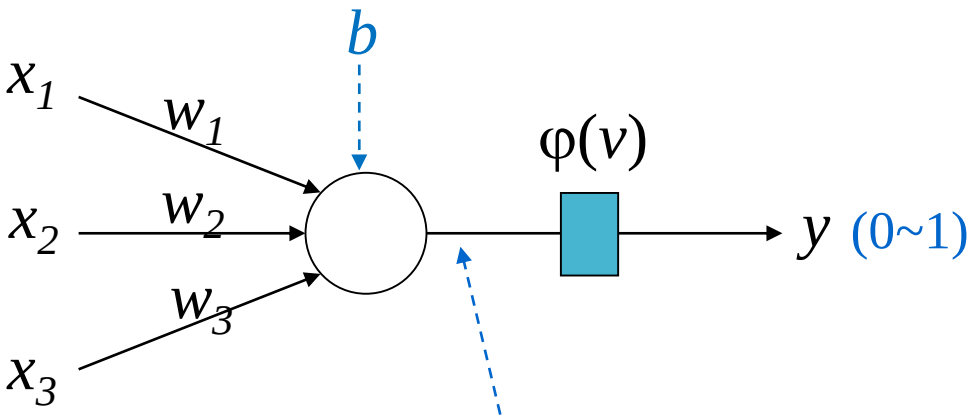



**Schematic of a biological neuron.**




http://sebastianraschka.com/Articles/2015_singlelayer_neurons.html

# 3. Perceptron




$$v = w_1x_1 + w_2x_2 + w_3x_3 + b$$

$$\varphi(v) = \begin{cases} 0, \text{ if } v \le \theta \\ 1, \text{ if } v > \theta \end{cases}$$

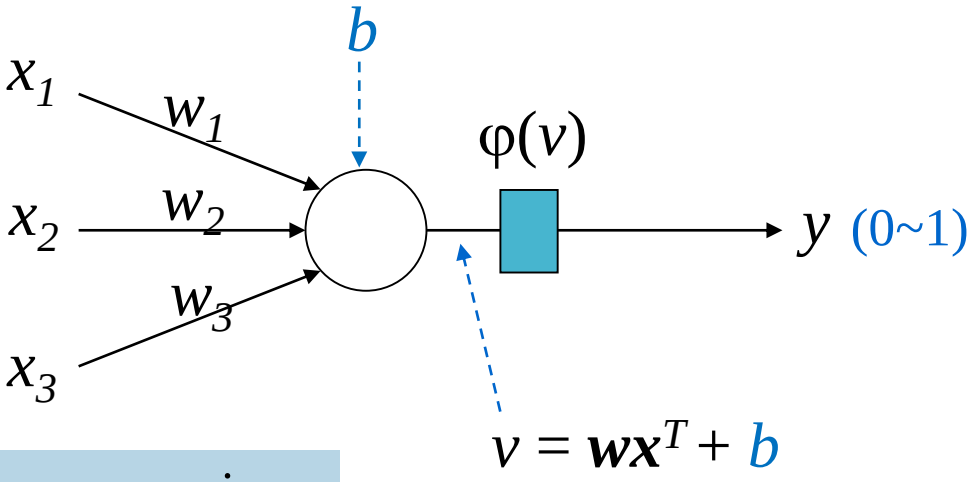
$w_1$, $w_2$, $w_3$: weights (가중치)

$b$ : bias (편향)

$v$ : weighted sum (가중합)

φ() : activation function (활성함수)

# 3. Perceptron




vector expression

$$\boldsymbol{w} = (w_1, w_2, w_3)$$

$$\boldsymbol{x} = (x_1, x_2, x_3)$$

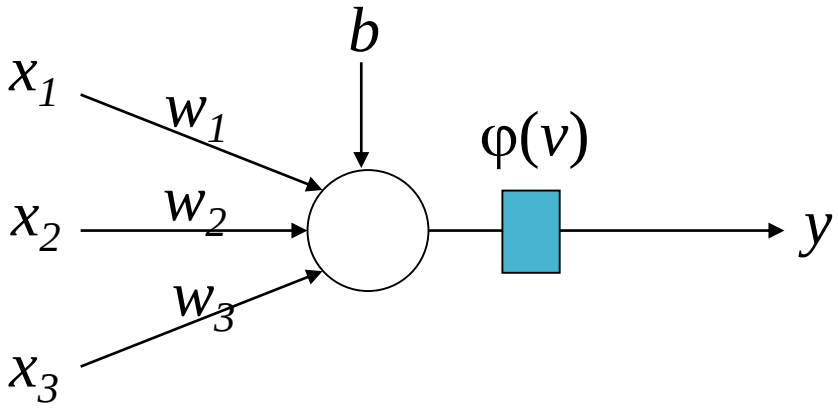
$$\boldsymbol{x}^T = \begin{pmatrix} x_1 \\ x_2 \\ x_3 \end{pmatrix}$$

$$\boldsymbol{w}\boldsymbol{x}^T = (w_1, w_2, w_3) \bullet \begin{pmatrix} x_1 \\ x_2 \\ x_3 \end{pmatrix}$$

$$= w_1x_1 + w_2x_2 + w_3x_3$$

# 3. Perceptron

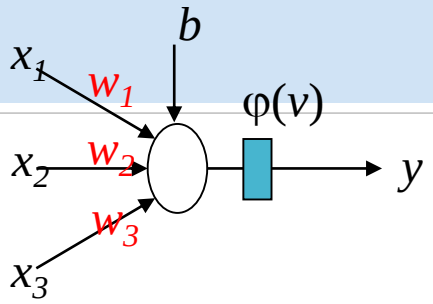
- Machine learning by perceptron



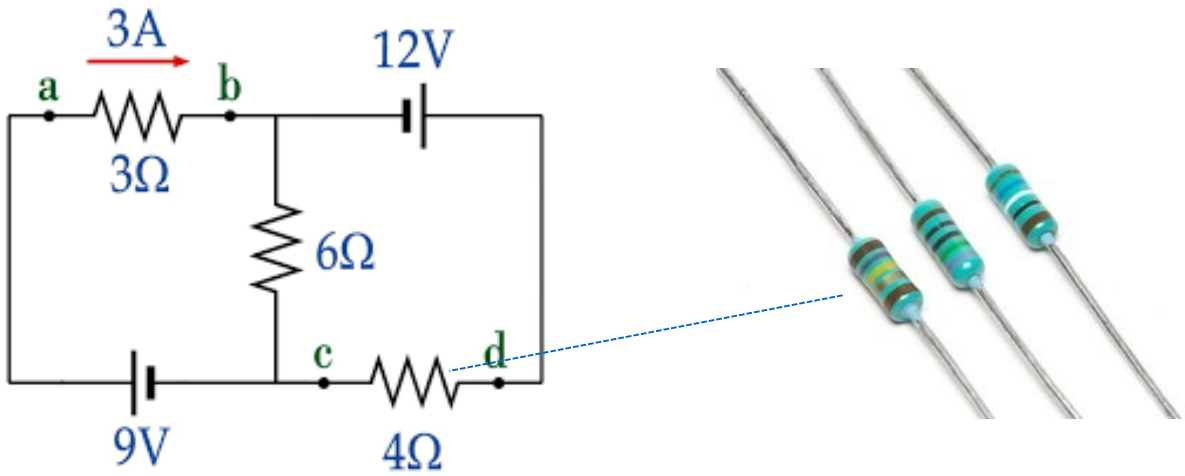

| $x_1$ | $x_2$ | $x_3$ | $x_4$ | $y$ |
|---|---|---|---|---|
| Sepal.Length | Sepal.Width | Petal.Length | Petal.Width | Species |
| 5.1 | 3.5 | 1.4 | 0.2 | setosa |
| 4.9 | 3.0 | 1.4 | 0.2 | setosa |
| 4.7 | 3.2 | 1.3 | 0.2 | setosa |
| 4.6 | 3.1 | 1.5 | 0.2 | setosa |
| 5.0 | 3.6 | 1.4 | 0.2 | setosa |
| 5.4 | 3.9 | 1.7 | 0.4 | setosa |

**w, b, φ 를 잘 조절하면 $x_1,..,x_n$ 으로부터 y 를 도출할 수 있다!**

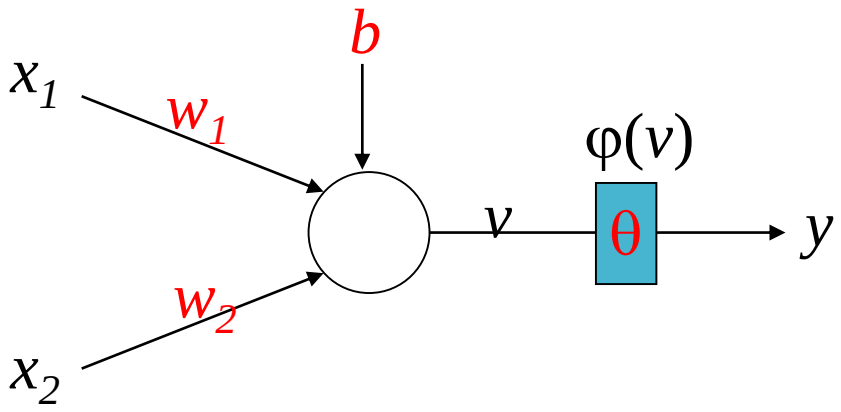
# 4. Weight value




- Weight value
  - 전기 회로에서 저항에 해당
  - 저항값이 크면 전기가 적게 흐르고, 저항값이 작으면 전류가 많으 흐름
  - 인공 신경망에서는 Weight value 가 크면 입력값이 출력에 많이 전달되고, Weight value 가 작으면 입력값이 출력에 적게 전달



# 5. Bias (편향)




$$\varphi(v) = \begin{cases} 0, \text{ if } v \le \theta \\ 1, \text{ if } v > \theta \end{cases}$$

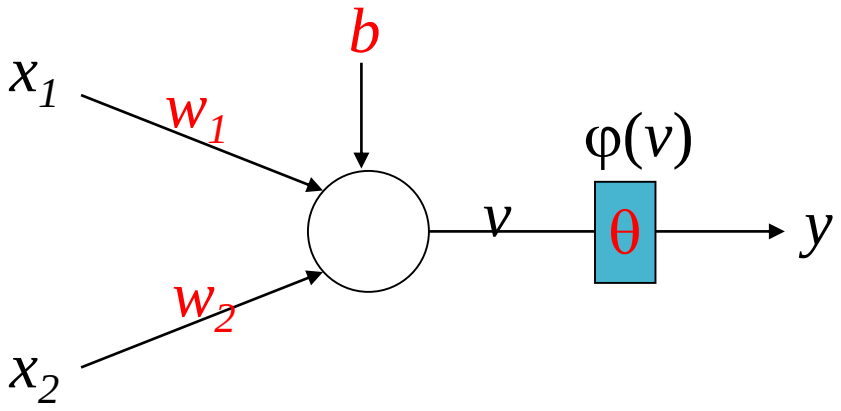
| $w_1$ | $w_2$ | b | θ |
|---|---|---|---|
| 0.5 | 0.5 | -0.7 | 0 |
| 0.5 | 0.5 | -0.8 | 0 |
| 1.0 | 1.0 | -1.0 | 0 |

| $x_1$ | $x_2$ | $y$ |
|---|---|---|
| 0 | 0 | 0 |
| 0 | 1 | 0 |
| 1 | 0 | 0 |
| 1 | 1 | 1 |

$w_1x_1 + w_2x_2 > |b|$ 일때 뉴런은 활성화 된다 (b<0인 경우)
$|b|$ 가 큰 값이면 활성화가 억제됨
$|b|$ 가 작은 값이면 활성화가 촉진됨

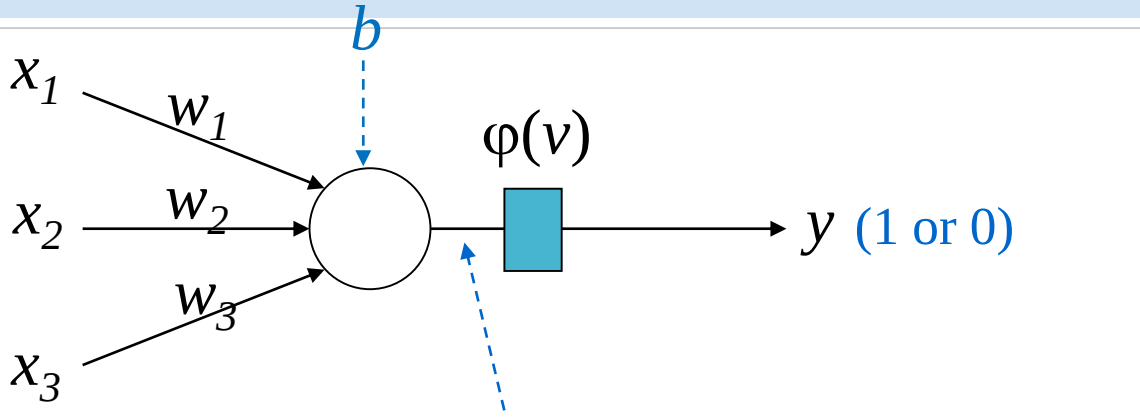
# 5. Bias (편향)

- b 값에 따라 y 값이 0 또는 1에 치우친 값이 나오므로 편향(bias) 라고 한다
- 예) 환자(0), 정상인(1) 을 진단하는 인공 신경망이 있을 때, 환자를 정상인으로 오진하는 경우를 최대한 막으려면 |b| 값을 크게 한다. 즉, 정상인이라는 증거가 충분히 클 때만 정상인으로 판단하겠다는 의미 (b<0인 경우)




- Weight 의 역할 : 각각의 입력신호가 출력에 영향을 주는 정도를 조절
- Bias 의 역할 : 뉴런이 얼마나 쉽게 활성화 되는지를 조절

# 6. Activation function

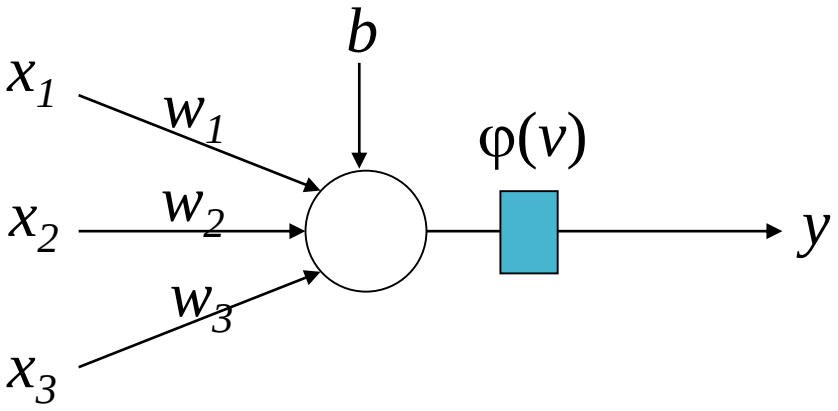



$$\varphi(v) = \begin{cases} 0, \text{ if } v \le \theta \\ 1, \text{ if } v > \theta \end{cases}$$

- Neural network 에서 output y 의 값은 0 ~ 1
- 활성화함수 φ() 는 가중합 $v$ 를 0 (deactivate) ~ 1 (activate) 사이의 값으로 변환하는 역할
- 여러 종류의 활성화 함수가 존재함

# 7. Perceptron example 1

- 연습




$$\varphi(v) = \begin{cases} 0, \text{ if } v \le \theta \\ 1, \text{ if } v > \theta \end{cases}$$

$w_1 = 0.4$, $w_2 = -0.1$, $w_3 = 0.5$, $b = -0.5$, $\theta = 0$

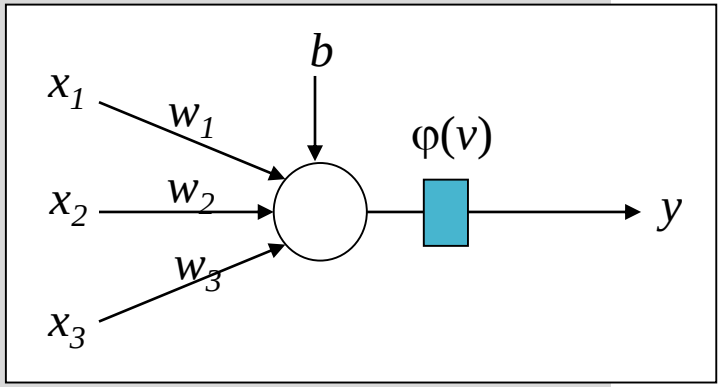
| x1 | x2 | x3 | y |
|---|---|---|---|
| 0.3 | 0.1 | 0.8 | |
| 0.5 | 0.6 | 0.3 | |
| 0.1 | 0.2 | 0.1 | |
| 0.8 | 0.7 | 0.7 | |
| 0.5 | 0.5 | 0.6 | |

# 7. Perceptron example 1

- Python code

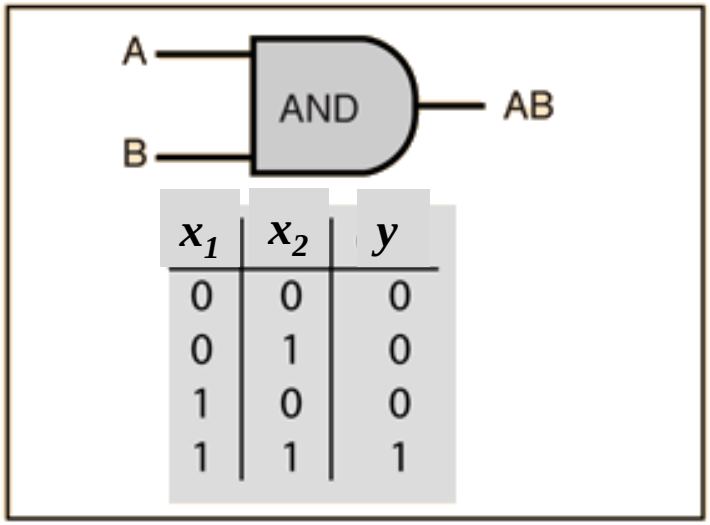
08.perceptron.py

```
import numpy as np

def myNN(x):
    W = np.array([0.4, -0.1, 0.5])
    b = -0.5
    seta = 0

    v = np.sum(x * W) + b
    y = 1 if v>seta else 0
    return y

ds = np.array([[0.3,0.1,0.8],
               [0.5,0.6,0.3],
               [0.1,0.2,0.1],
               [0.8,0.7,0.7],
               [0.5,0.5,0.6]])

for i in range(5):
    print(myNN(ds[i:]))
```



# 8. Perceptron example 2

- AND logic

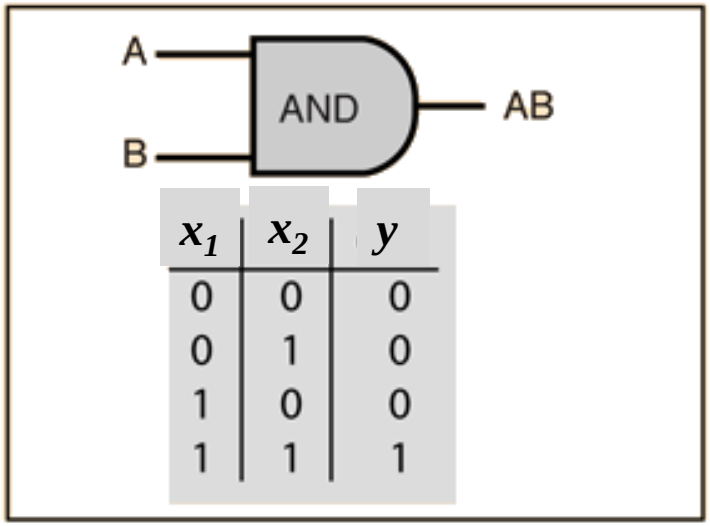



| $x_1$ | $x_2$ | $y$ |
|---|---|---|
| 0 | 0 | 0 |
| 0 | 1 | 0 |
| 1 | 0 | 0 |
| 1 | 1 | 1 |

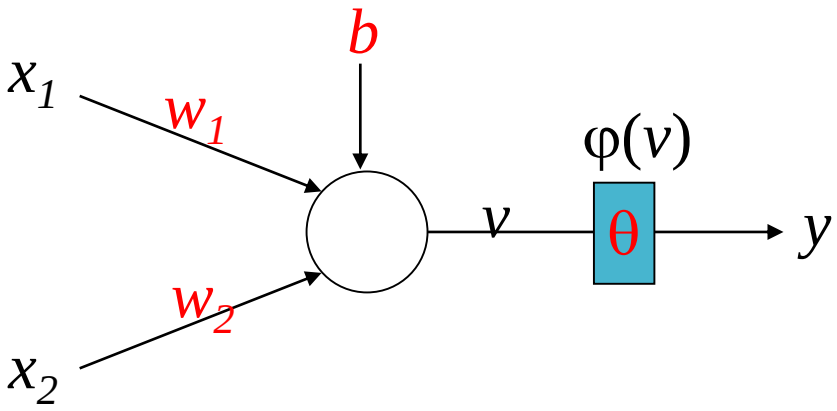
http://hyperphysics.phy-astr.gsu.edu/hbase/Electronic/and.html

- Let's implement AND gate using Perceptron
  - Weight, bias, θ 값을 정해주는 문제가 된다.

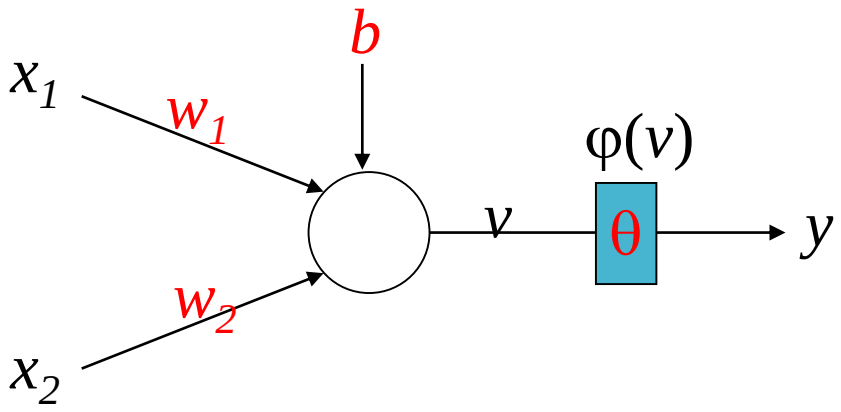
# 8. Perceptron example 2




| $x_1$ | $x_2$ | $y$ |
|---|---|---|
| 0 | 0 | 0 |
| 0 | 1 | 0 |
| 1 | 0 | 0 |
| 1 | 1 | 1 |



# 8. Perceptron example 2




Possible answers :

| $w_1$ | $w_2$ | b | θ |
|---|---|---|---|
| 0.5 | 0.5 | 0 | 0.7 |
| 0.5 | 0.5 | 0 | 0.4 |
| 1.0 | 1.0 | 0 | 1.0 |

| $x_1$ | $x_2$ | $y$ |
|---|---|---|
| 0 | 0 | 0 |
| 0 | 1 | 0 |
| 1 | 0 | 0 |
| 1 | 1 | 1 |

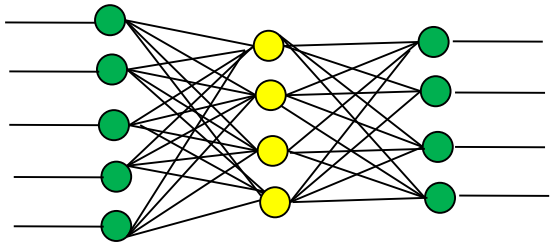
How to find **w**, b, θ ?

⇒ Machine learning

# 9. Design of neural network

- Neural network for Classification
  - Input & output

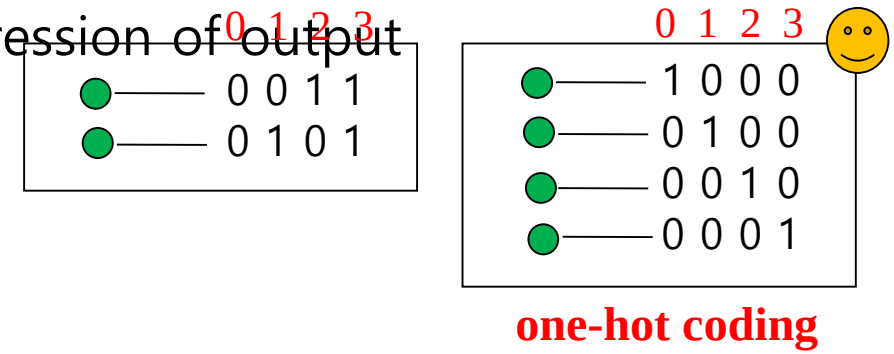
[dataset]
5 features
4 classes



    - No of input nodes = No of feature
    - No of output nodes = No of class
    - Generally, the input for a single neuron is a value between 0 and 1 (or -1 ~1)

  - Two types expression of output

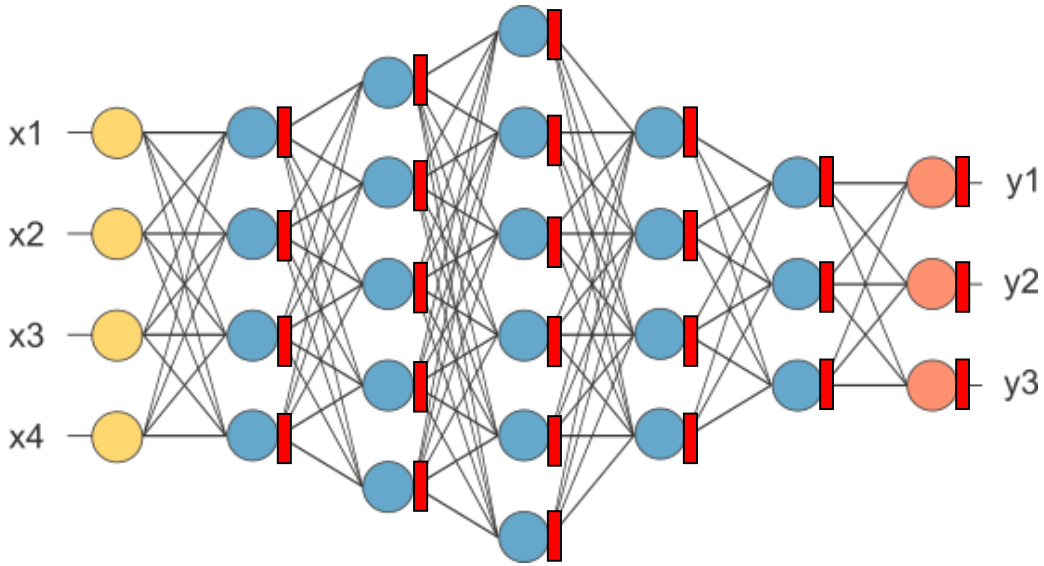
4 classes




**one-hot coding**

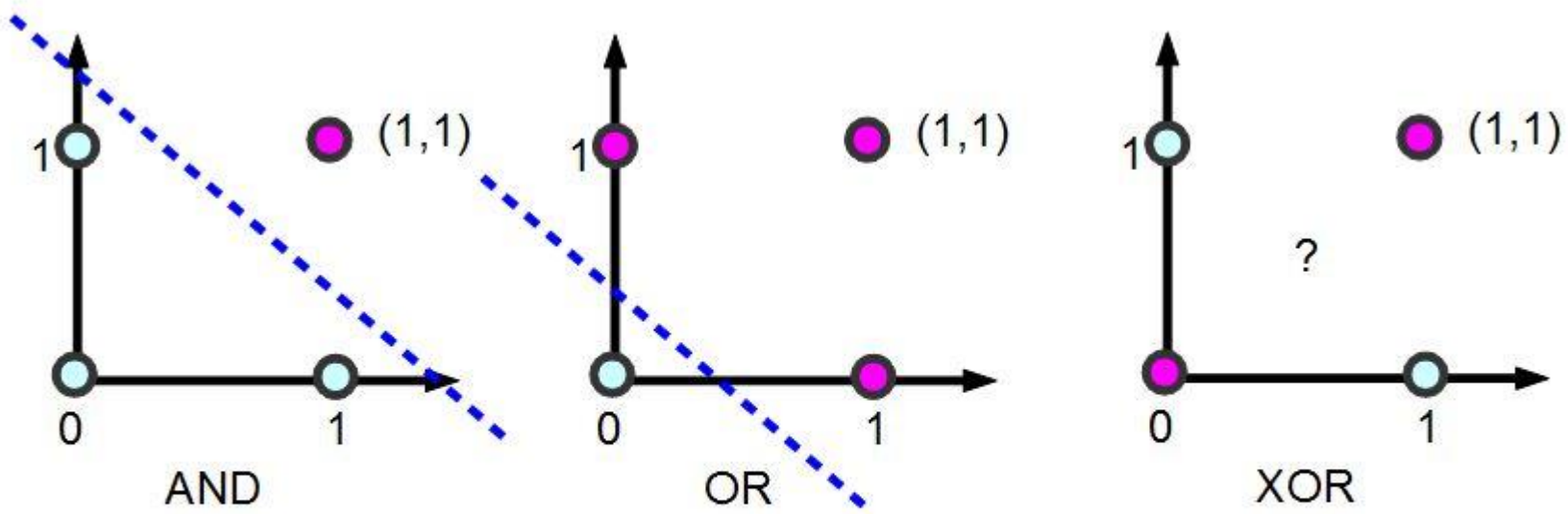
# 9. Design of neural network

- Number of hidden layers ?
- Number of nodes for each hidden layer ?
- Activation function ?
- Weight values ? ⟹ updated by learning process
  - Generally, each initial weight value is between -1 and 1



# 10. Limitation of simple perceptron

- 선형 분리 문제 만 해결가능 하다
- XOR 논리는 학습이 안됨

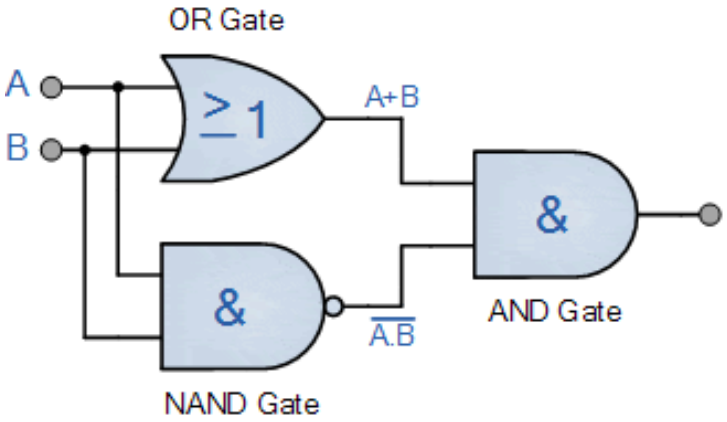



| in1 | in2 | out |
|---|---|---|
| 0 | 0 | 0 |
| 0 | 1 | 1 |
| 1 | 0 | 1 |
| 1 | 1 | 0 |

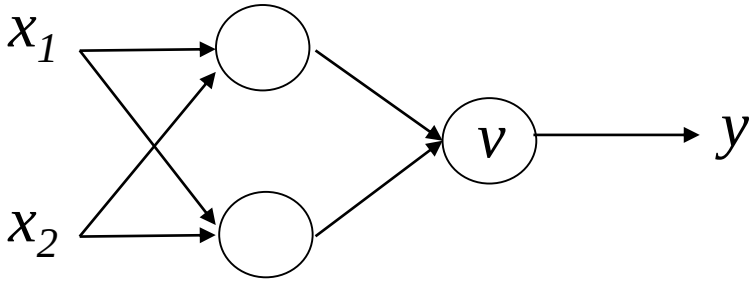
https://www.slideshare.net/SanghunYun/convolutional-neural-networks-75256236

# 10. Limitation of simple perceptron

- 해결 방법 : 다층 perceptron




복잡한 문제를 해결하기 위해서는
다층 신경망이 필요하다



# Keywords

- Perceptron
- Weight value
- Bias
- Activation function
- Input layer, hidden layer, output layer