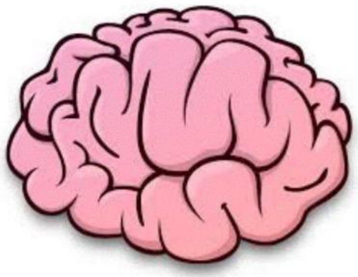
Perceptron

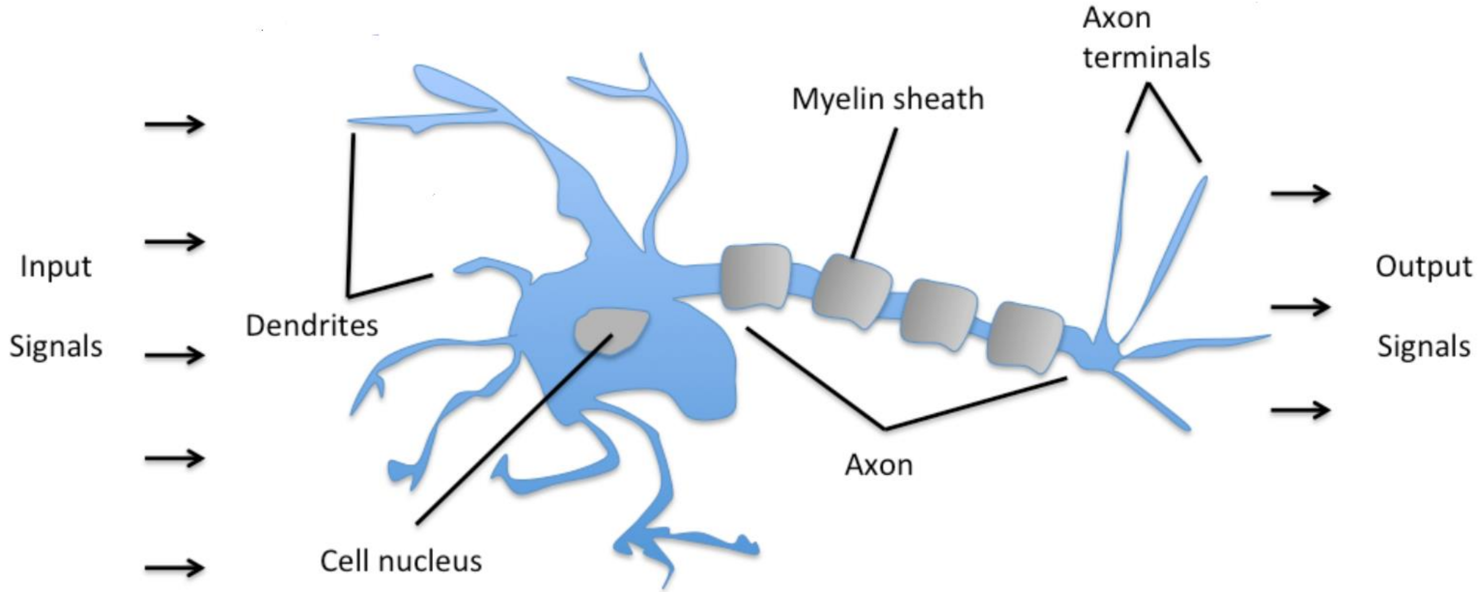
Computational Linguistics @ Seoul National University

DL from Scratch

By Hyopil Shin

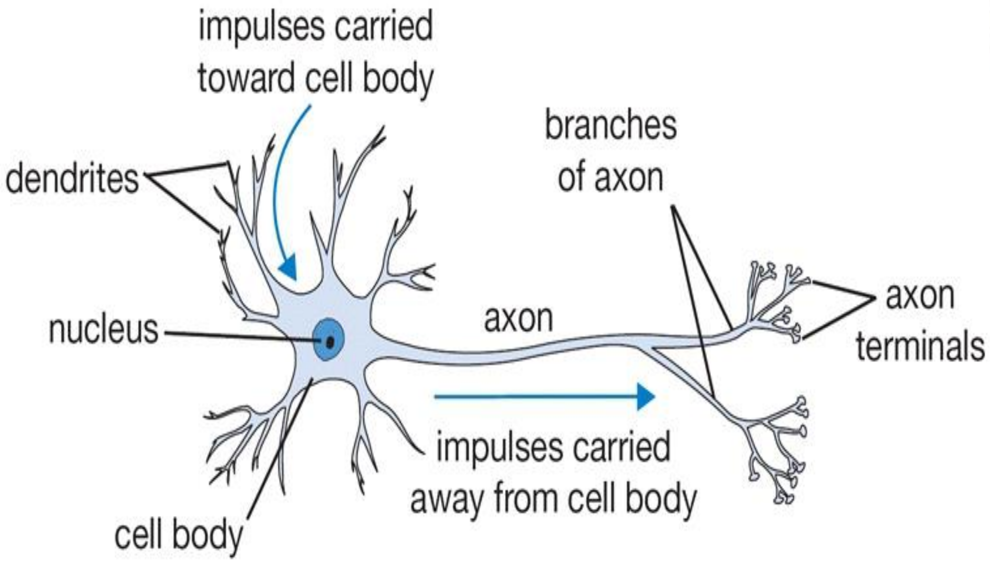
Schematic of a biological neuron

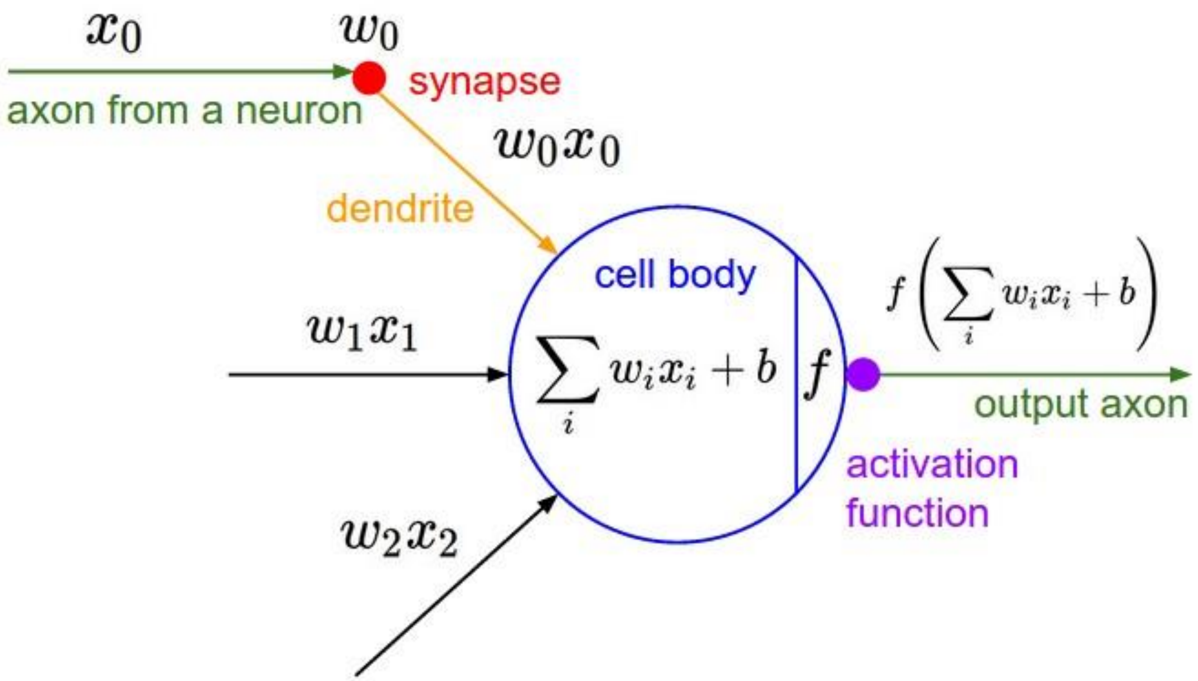




Schematic of a biological neuron.

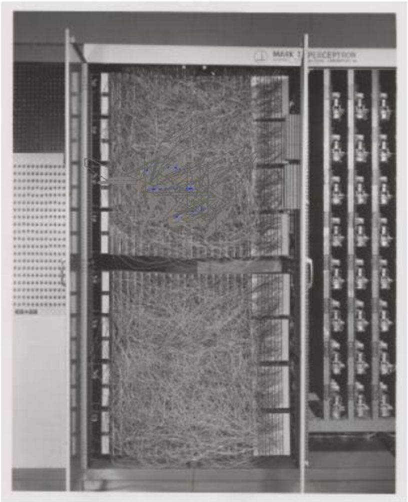
Modeling one Neuron





Hardware Implemenatations

Hardware implementations

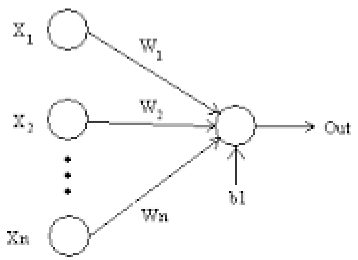


Frank Rosenblatt, ~1957: Perceptron

Widrow and Hoff, ~1960: Adaline/Madaline

Perceptron?

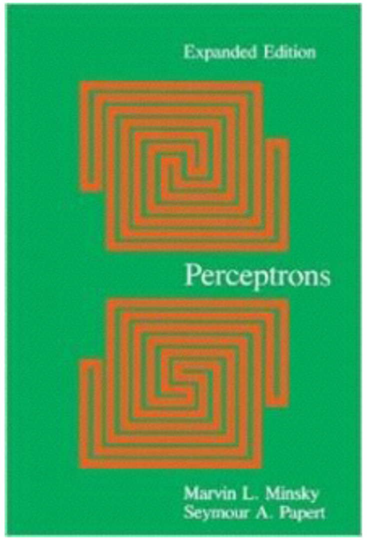
- Introduced by Frank Rosenblatt, author of the book *Principles of Neurodynamics*
- In machine learning, the **perceptron** is an algorithm for supervised learning of binary classifiers (functions that can decide whether an input, represented by a vector of numbers, belongs to some specific class or not).



Perceptrons

Perceptrons (1969)

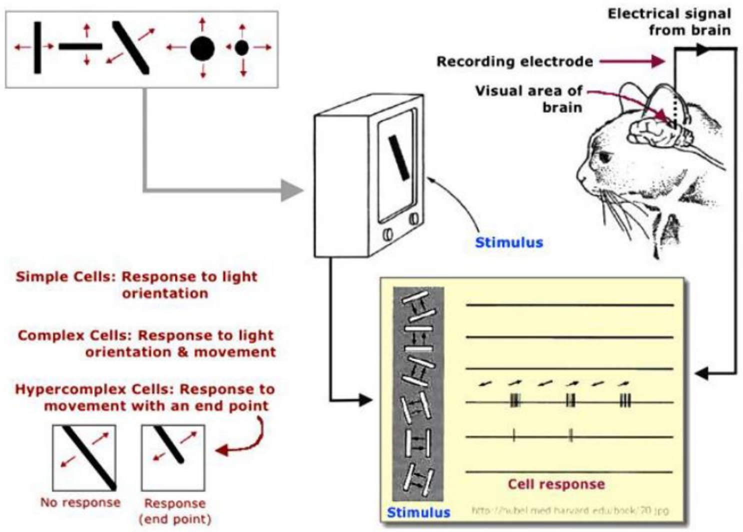
by Marvin Minsky, founder of the MIT AI Lab



- We need to use MLP, multilayer perceptrons (multilayer neural nets)
- No one on earth had found a viable way to train MLPs good enough to learn such simple functions.

Perceptrons

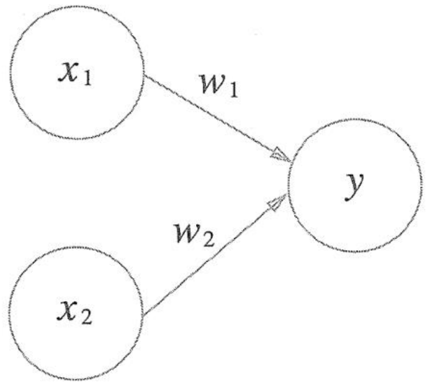
Convolutional Neural Networks



Hubel & Wiesel, 1959

Perceptron?

- Neuron
- Weight
- 임계값 (Θ)



$$y = \begin{cases} 0 & (w_1x_1 + w_2x_2 \leq \theta) \\ 1 & (w_1x_1 + w_2x_2 > \theta) \end{cases}$$

논리회로 – AND Gate

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

논리회로 – NAND, OR Gate

그림 2-3 NAND 게이트의 진리표

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

그림 2-4 OR 게이트의 진리표

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

Perceptron 구현하기

- Simple AND

```
def AND(x1, x2):
    w1, w2, theta = 0.5, 0.5, 0.7
    tmp = x1*w1 + x2*w2
    if tmp <= theta:
        return 0
    elif tmp > theta:
        return 1
```

- 가중치(weight)와 편향(bias)
  - Y = 0 ($b + w_1x_1 + w_2x_2 \leq 0$)
- 1($b + w_1x_1 + w_2x_2 \gg 0$)

$$y = \begin{cases} 0 & (w_1x_1 + w_2x_2 \leq \theta) \\ 1 & (w_1x_1 + w_2x_2 > \theta) \end{cases}$$

```
def AND(x1, x2):
    x = np.array([x1, x2])
    w = np.array([0.5, 0.5])
    b = -0.7
    tmp = np.sum(w*x) + b
    if tmp <= 0:
        return 0
    else:
        return 1
```

Perceptron 구현하기

- NAND, OR Gate 구현

```
def NAND(x1, x2):
    x = np.array([x1, x2])
    w = np.array([-0.5, -0.5]) # AND와는 가중치(w와 b)만 다르다!
    b = 0.7
    tmp = np.sum(w*x) + b
    if tmp <= 0:
        return 0
    else:
        return 1

def OR(x1, x2):
    x = np.array([x1, x2])
    w = np.array([0.5, 0.5])   # AND와는 가중치(w와 b)만 다르다!
    b = -0.2
    tmp = np.sum(w*x) + b
    if tmp <= 0:
        return 0
    else:
        return 1
```

Perceptron – XOR Gate

- Exclusive OR Gate
- XOR Gate → Perceptron으로 가능?
- OR Gate

(b, w1, w2) = (−0.5, 1.0, 1.0)

$$y = \begin{cases} 0 & (-0.5 + x_1 + x_2 \leq 0) \\ 1 & (-0.5 + x_1 + x_2 > 0) \end{cases}$$

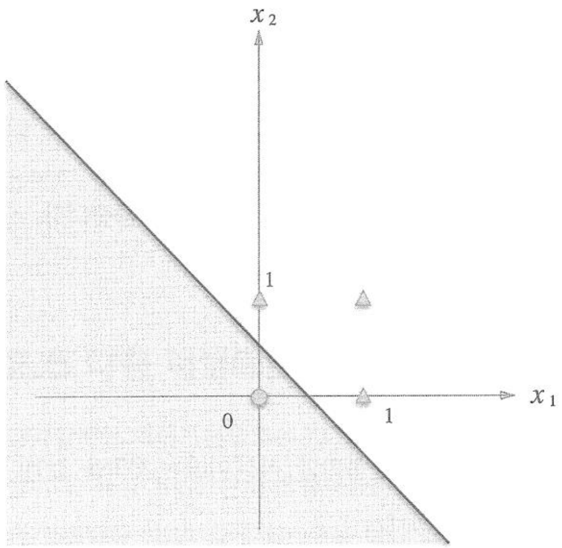
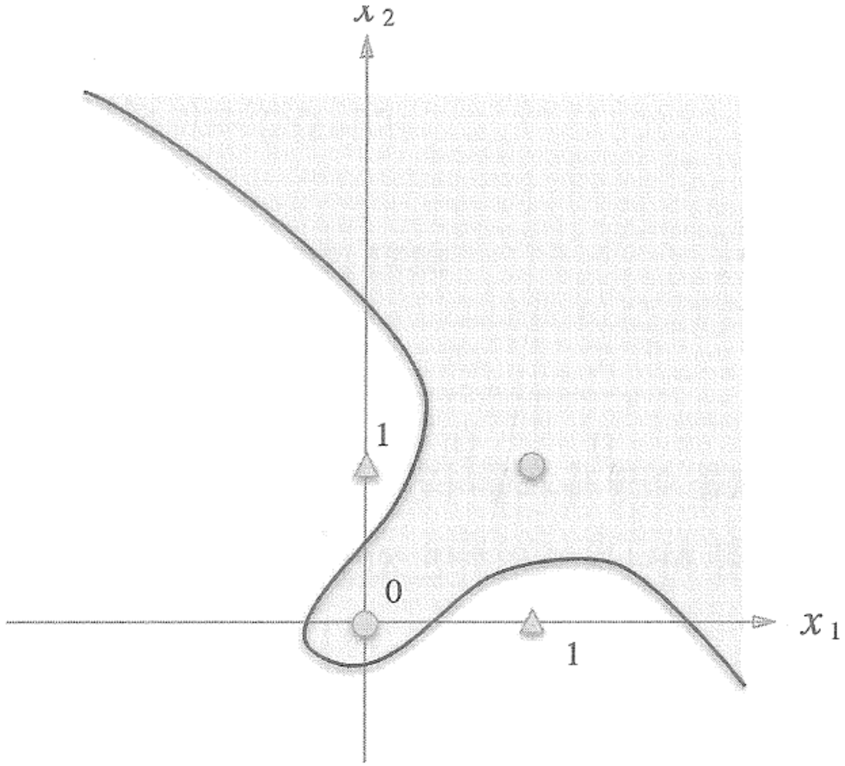


그림 2-5 XOR 게이트의 진리표

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

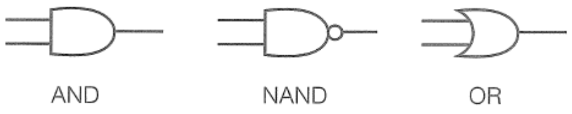
Perceptron – XOR Gate

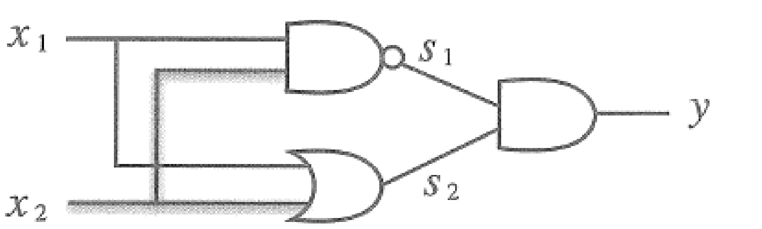
- Linear vs nonlinear
  - Single-layer perceptron으로는 XOR gate를 표현할 수 없다
  - Single-layer perceptron으로는 비선형 영역을 분리할 수 없다



Multi-layer Perceptron

- AND, NAND, OR Gate를 조합





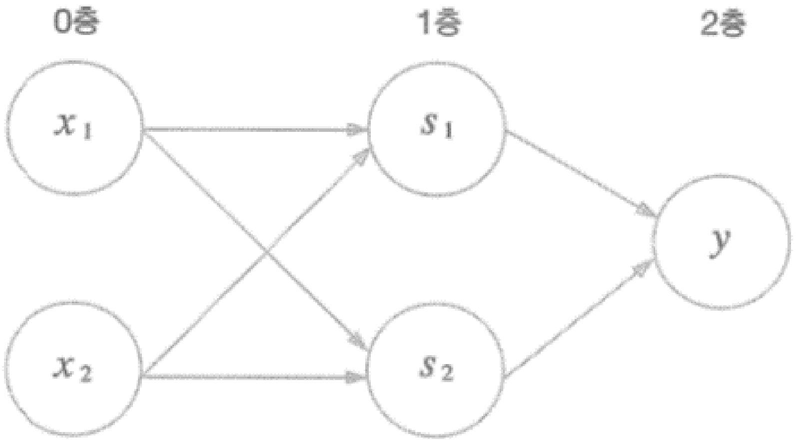
x_1	x_2	s_1	s_2	y
0	0	1	0	0
1	0	1	1	1
0	1	1	1	1
1	1	0	1	0

```
def XOR(x1, x2):
    s1 = NAND(x1, x2)
    s2 = OR(x1, x2)
    y = AND(s1, s2)
    return y
```

Multi-layer Perceptron

- Single-layer perceptron으로는 표현하지 못한 것을 층을 하나 늘려 구현할 수 있게 함
- 층을 쌓아(깊게 하여) 더 다양한 것을 표현할 수 있게 함
- Multi-layer perceptron은 아주 복잡한 표현도 가능하게 함

그림 2-13 XOR의 퍼셉트론



Round Up

- Perceptron은 입출력을 갖춘 알고리즘이다. 입력을 주면 정해진 규칙에 따른 값을 출력한다
- Perceptron에서는 ‘가중치 ' 와 ‘편향 ' 을 매개변수로 설정한다
- Perceptron으로는 AND, OR gate 등의 논리 회로를 표현할 수 있다
- XOR gate는 single-layer perceptron으로는 표현할 수 없다
- 2층 perceptron을 이용하면 XOR gate를 표현할 수 있다
- Single-perceptron은 직선형 영역만 표현할 수 있고, multi-layer perceptron은 비선형 영역도 표현할 수 있다