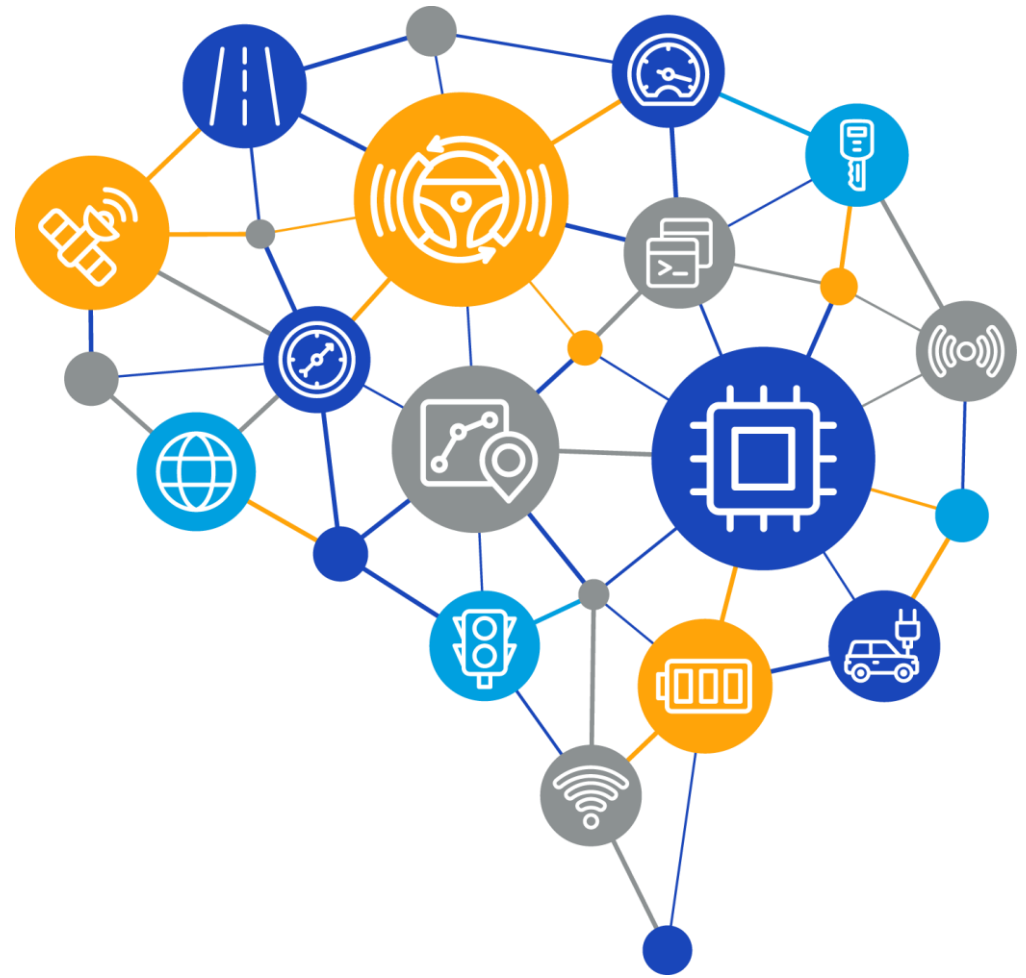


# 퍼셉트론과 신경망

Perceptron and Neural Network

2022.07

강환수 교수



# AI Experts Who Lead The Future

## CONTENTS

- 01 | 인공지능경망 개요
- 02 | 인공뉴런 퍼셉트론 개요
- 03 | 다층 퍼셉트론
- 04 | 딥러닝 개요

AI Experts  
Who Lead  
The Future

# 01

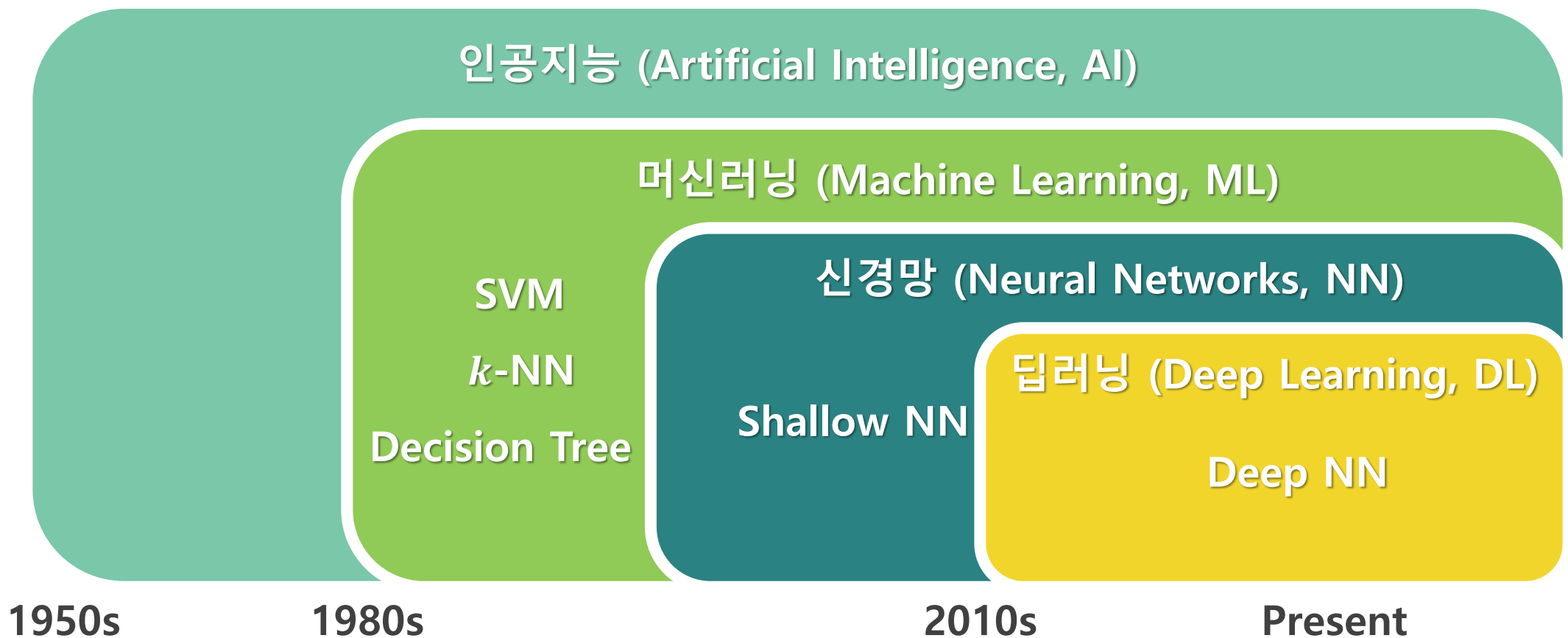
## 인공신경망 개요

다음 자료를 기반으로 제작  
난생처음 인공지능 입문 (출판사: 한빛아카데미)

## 01. 인공지능망이란 무엇인가요?

인공지능 활용 Python language

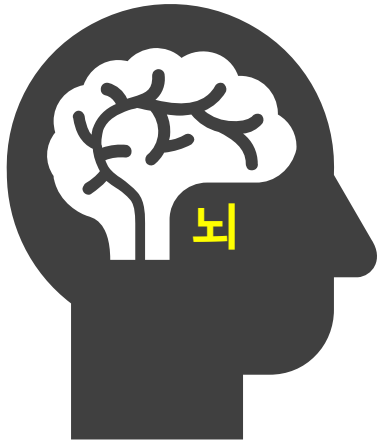
- 인공지능, 머신러닝, 딥러닝의 관계



# 인간의 뇌 (Brain)

인공지능 활용 Python language

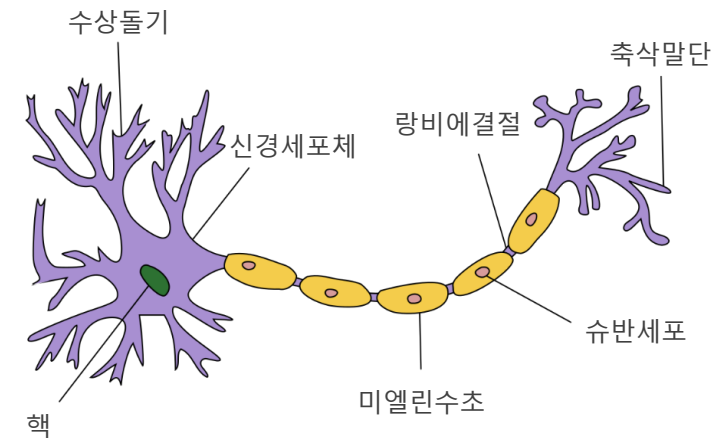
- 뇌는 신경세포 (= Neuron, 뉴런)들이 뭉쳐 큰 군집을 이루고 있는 덩어리이며 동물의 중추 신경계를 관장하는 기관
- 뇌는 움직임, 행동 대부분을 관장하고, 신체 항상성을 유지하며 인지, 감정, 기억, 학습 등을 담당



인간의 뇌는 대략 1000억개에서 860억개의 신경세포로 구성되어 있습니다.

[정보출처] [https://www.hani.co.kr/arti/science/science\\_general/755976.html](https://www.hani.co.kr/arti/science/science_general/755976.html)

## 신경세포 (= 뉴런)



[사진출처] [https://ko.wikipedia.org/wiki/신경\\_세포](https://ko.wikipedia.org/wiki/신경_세포)

## • 인공신경망 (Artificial Neural Network, ANN)

- 인간의 뇌를 구성하는 신경세포인 뉴런 (Neuron)의 연결 구조를 흉내 내서 만든 머신러닝 모델
- 뉴런은 수상돌기로부터 입력 받는 신호의 총합이 일정 값 이상이 되면 축삭돌기로 신호를 전달하는 구조를 가짐

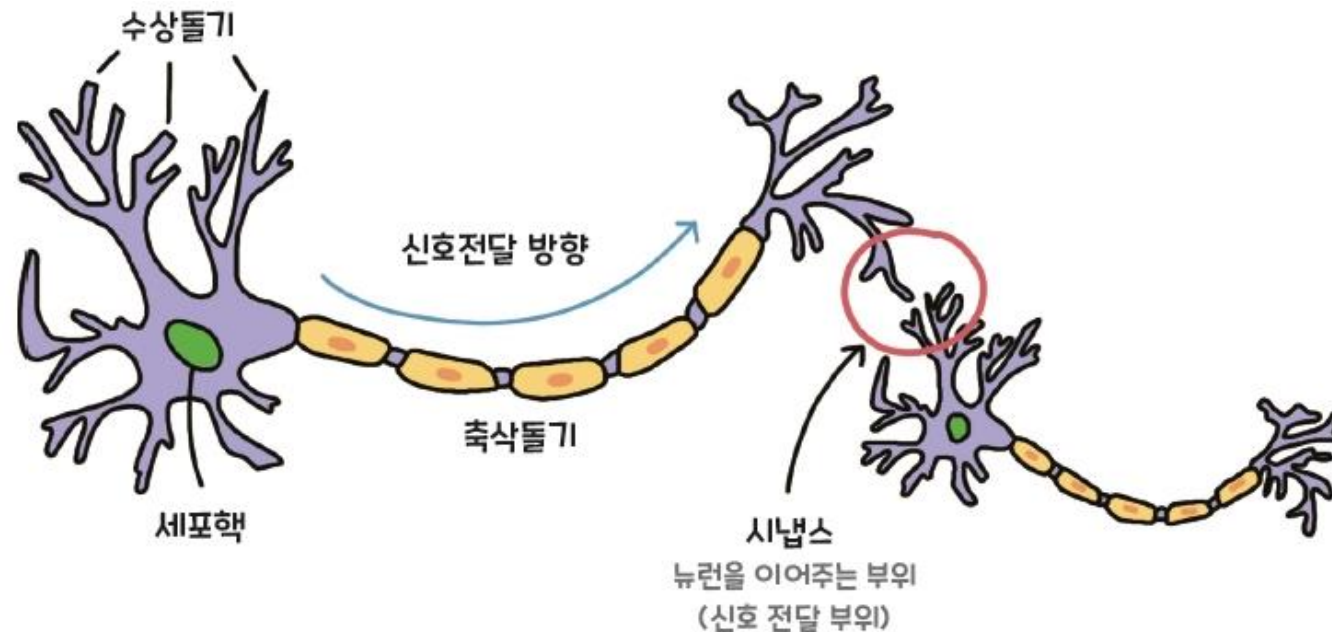
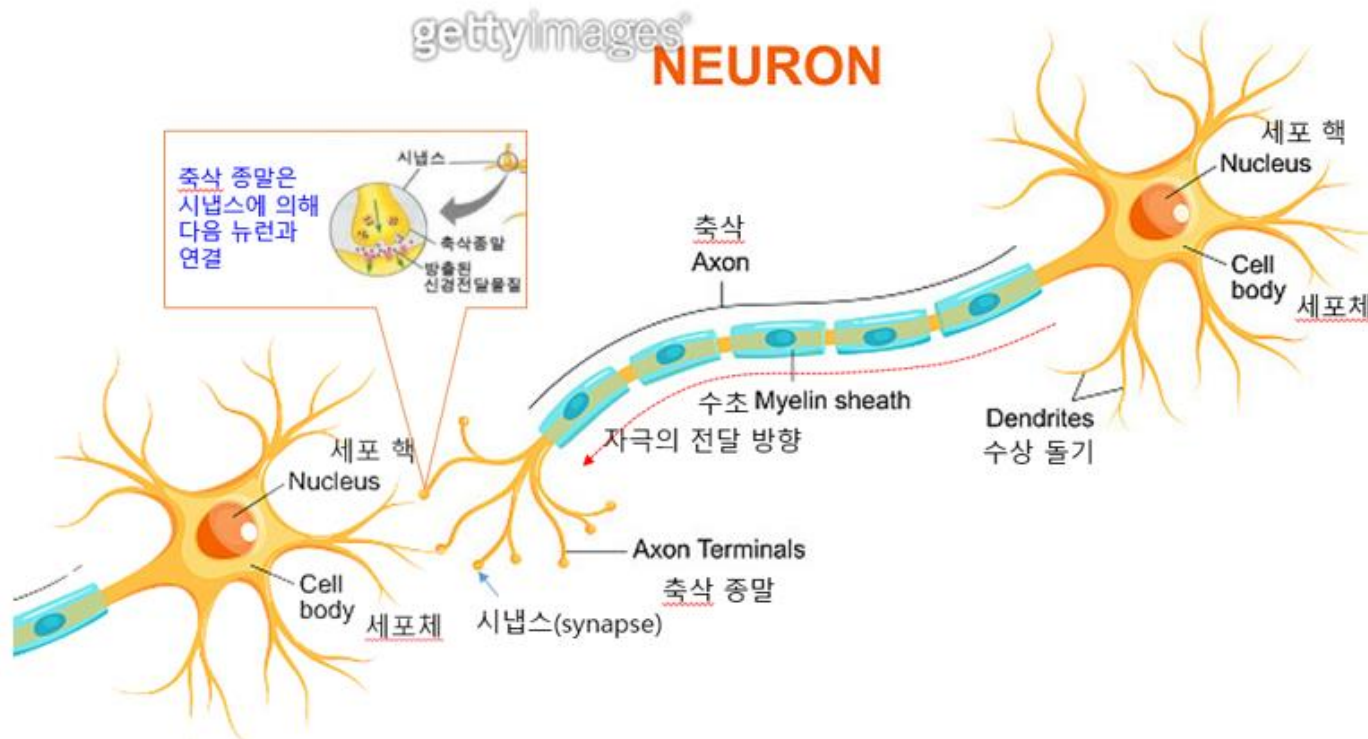


그림 3-1 뉴런의 연결 구조

- 신경계를 구성하는 세포: 한 뉴런에서 다른 뉴런으로 신호를 전달
  - 뉴런은 신경 접합부인 시냅스(Synapse)로 연결
    - 하나의 뉴런에는 수천 개 이상의 시냅스가 존재하므로 수 백조 이상의 시냅스가 있음
  - 사람의 뇌는 1,000억 개의 뉴런으로 구성
- 수상돌기(dendrite)
  - 세포핵(nucleus)이 있는 세포체(cell body, soma)에서 멀리 뻗어 나온 형태
  - 뉴런에서 뻗어 나온 나뭇가지 모양이라 가지돌기라 부름



# 신경세포(神經細胞)인 뉴런(neuron) 2/2

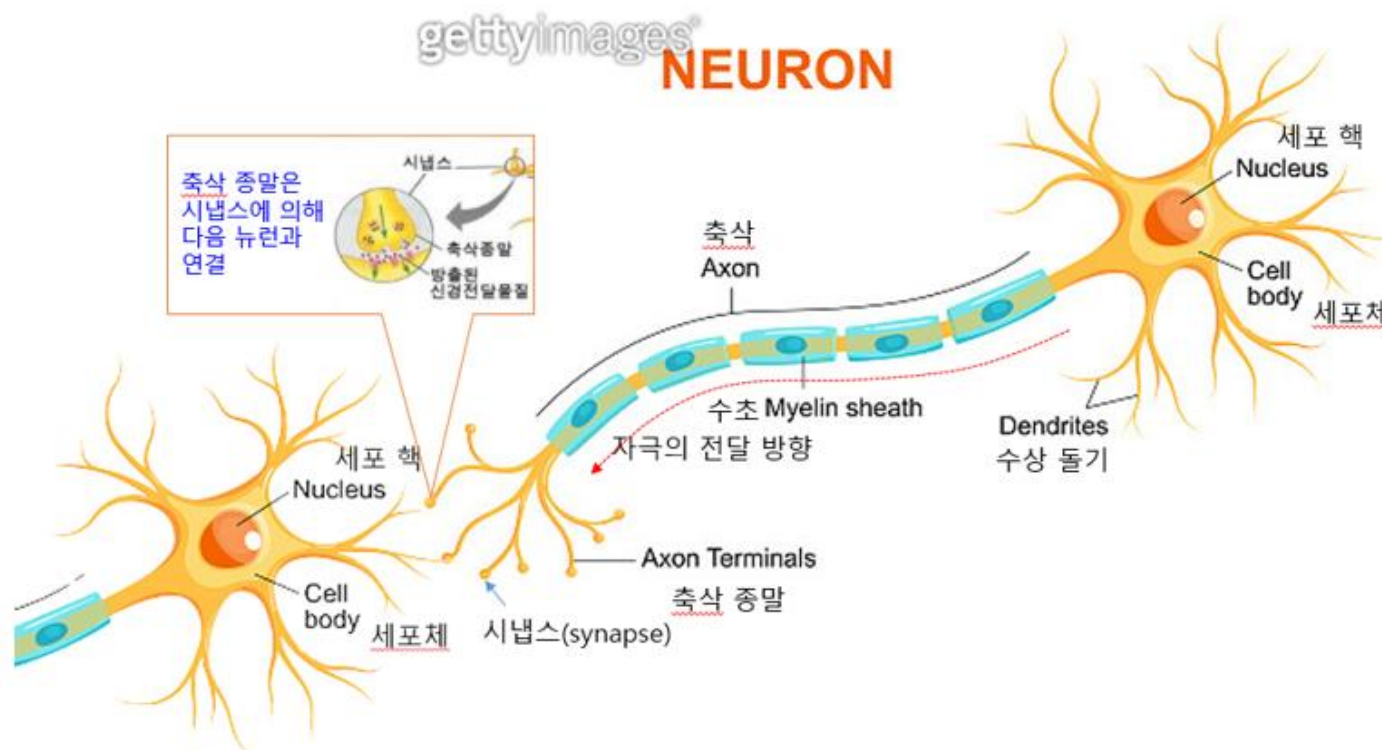
인공지능 활용 Python language

## • 축삭(axon)

- 수상돌기보다 훨씬 긴 관 형태로 세포체에서 길게 뻗어 있고,
  - **미엘린 수초(myelin sheath)로 절연되어 있음**
- 일정한 간격으로 절연되지 않은 부위인 랑비에 결절(nodes of ranvier)이 있음
- 축삭의 끝부분은 절연이 벗겨지고 여러 갈래로 나뉘어 축삭 종말(axon terminals)을 형성

## • 시냅스(Synapse)

- 이전 뉴런의 축삭 종말과 다음 뉴런의 수상돌기 사이 지점인 뉴런과 뉴런을 연결하는 부분

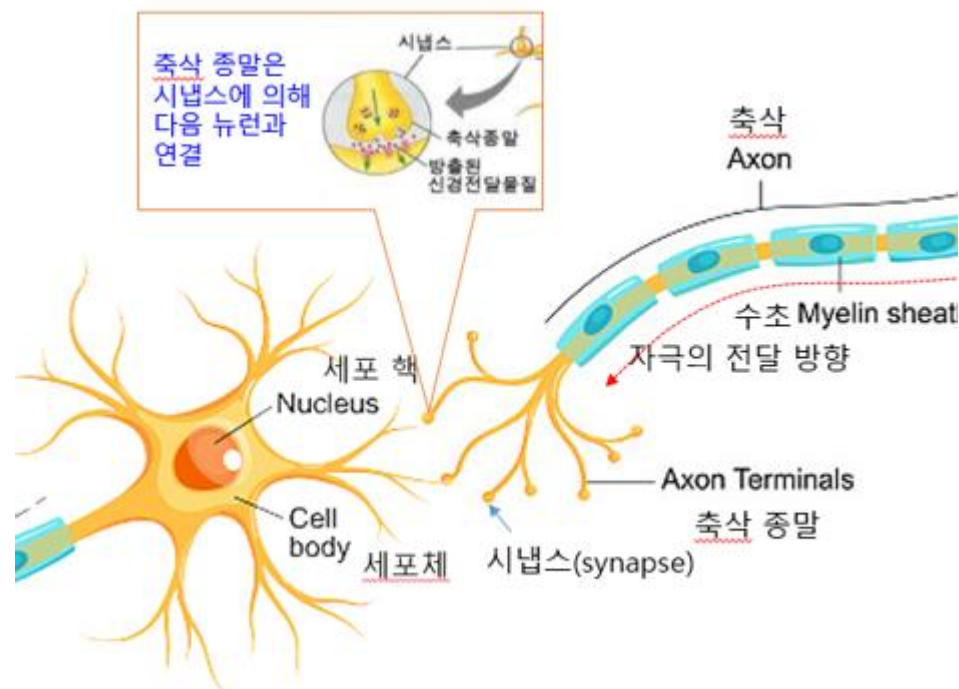




# 자극의 전달 과정

인공지능 활용 Python language

- 외부에서 들어온 자극은 여러 뉴런을 통해 뇌에 전달
  - 뉴런은 시냅스에서 전달 받은 신호를 수상돌기를 통해 세포체에 전달
  - 다시 나트륨 통로와 칼륨 통로 등의 이온 통로인 축삭을 통해 전기적인 방법으로 신호가 전달
- 시냅스
  - 인접한 다른 뉴런에게 시냅스라는 구조를 통해
    - 화학적 방법으로 신호를 다음 뉴런의 수상돌기 말단으로 전달
      - 이때 수상돌기는 일정수준을 넘는 강한 신호만 세포체로 전달
      - 수상돌기에서 돌출된 가시는 축삭 종말의 시냅스전(presynaptic) 뉴런에서 방출되는 신경전달물질을 포착할 수 있게 되어 있음



# 시냅스(synapse)의 정보 전달과정

- 시냅스는 뉴런이나 세포들과 접촉하여 정보가 전달되는 부분
  - 신호를 주는 뉴런을 시냅스전(pre-synapse) 뉴런이라 부르며
  - 신호를 받는 부분을 시냅스후(post-synapse)라고 부름
  - 시냅스전과 후 뉴런의 막은 20~50nm 정도의 시냅스 틈(synaptic cleft)으로 분리
- 시냅스의 시냅스전 부위
  - 시냅스전 요소(presynaptic element)
  - 보통 뉴런의 축삭 말단 부위
    - 직경 50nm정도의 수십 개의 시냅스 소낭(synaptic vesicles)들을 가지고 있음
    - 시냅스 소낭들은 신경전달물질들을 저장하고 있는 주머니
  - 시냅스 전에서 나온 신경전달물질은 시냅스 후 막의 수용체로 전달
    - 주름이 집중된 구조인 수용체는 신경전달물질이 다른 곳으로 새지 않고 집중적으로 수용체 위에 전달되도록 함

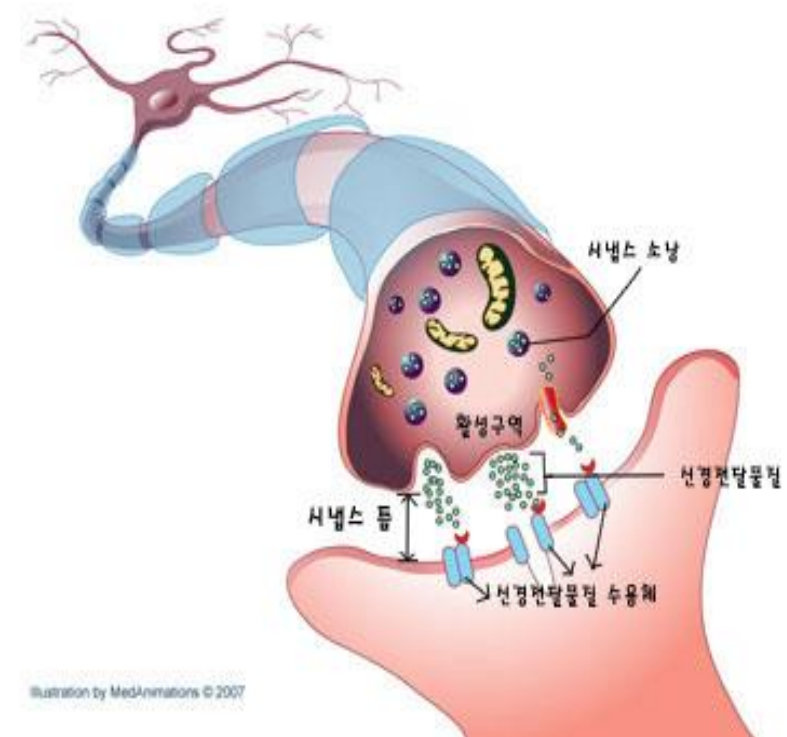
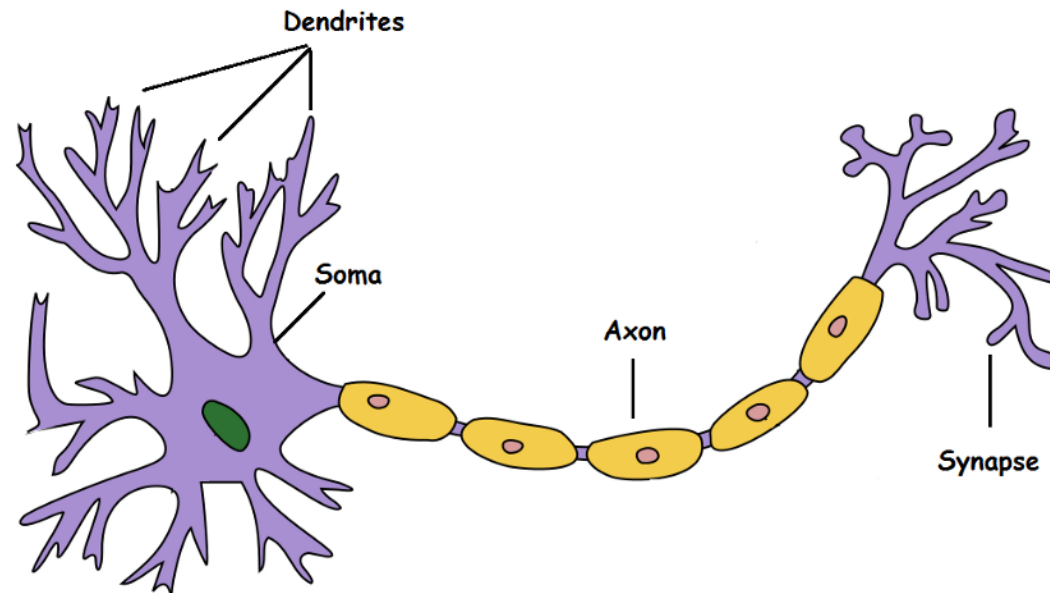


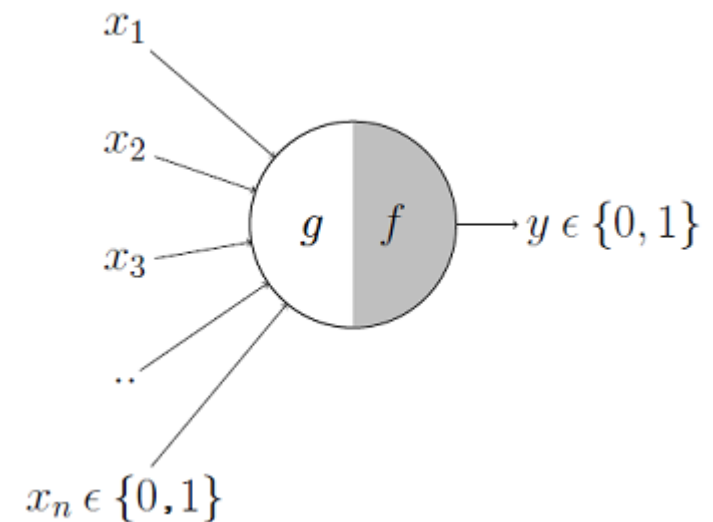
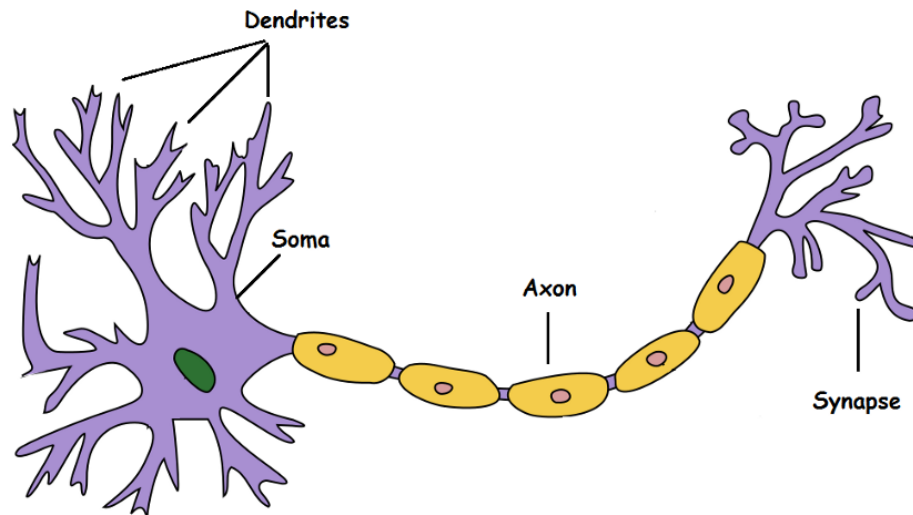
Illustration by MedAnimations © 2007

- 생물학적 뉴런에 대한 인류 최초의 수학적 모델
  - 1943년 McCulloch와 Pitts에 의해 생물학적 뉴런의 기능을 모방



- 수상돌기 : 다른 뉴런으로부터 신호를 받습니다.
- 세포체(Soma, cell body) : 정보를 처리합니다.
- 축삭 돌기(Axon) : 이 뉴런의 출력을 전달
- 시냅스(synapse) : 다른 뉴런 과 연결되는 지점

- 1943년 Warren MuCulloch(신경과학자)와 Walter Pitts(논리학자)에 의해 고안
  - 뉴런의 첫 번째 계산 모델
  - 2 부분 구성
    - 첫 번째 부분인  $g$ 는 입력(dendrite)을 취하여 집계를 수행
    - 두 번째 부분인  $f$ 는 집계된 값을 기반으로 결정을 내림
      - 입출력은 모두 논리 값인 0, 1



# 인공뉴런 TLU의 개념 (1/2)

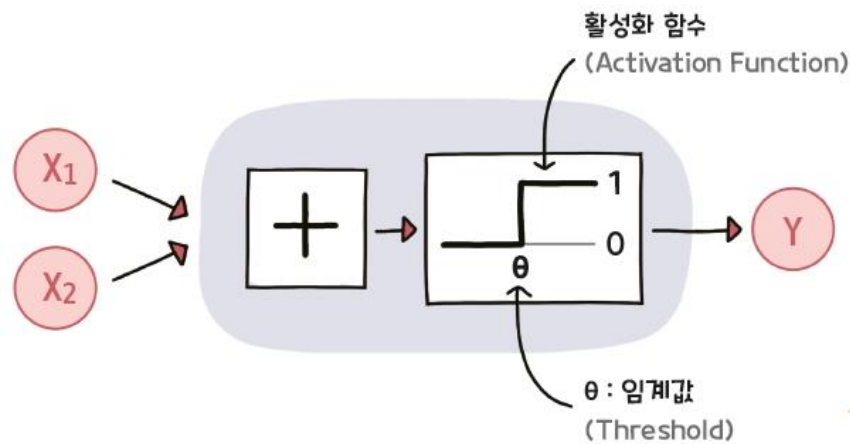
인공지능 활용 Python language

- 1943년, 워런 맥컬록과 월터 피츠는 뉴런을 모델로 한 인공뉴런 TLU를 제안함
  - MCP 뉴런 == TLU
- TLU (Threshold Logic Unit)
  - 뉴런의 수상돌기처럼 외부로부터 신호들을 입력 받아 그 총합을 구하는 부분과, 입력 신호들의 총합이 어느 임계 값 (Threshold) 이상일 경우에만 신호를 출력하는 부분으로 구성 되어 있음
- 활성화 함수 (Activation Function)
  - 입력 신호들의 총합을 임계 값과 비교한 뒤 출력을 결정하는 함수
  - 활성화 함수는 임계 값 전과 후에 따라 그 값이 0과 1로 분명하게 구분되는 특징을 갖고 있음
  - 활성화 함수는 입력 신호의 총합이 임계 값보다 클 경우 1을 출력하고, 작을 경우에는 0을 출력

## 인공뉴런 TLU의 개념 (2/2)

인공지능 활용 Python language

- 인공뉴런 TLU의 동작 원리는 입력 값  $X_1$ 과  $X_2$ 를 더한 값이 임계 값  $\theta$ 보다 같거나 크면 출력 값  $Y$ 를 1로 출력하고,  $\theta$ 보다 작으면 0으로 출력하는 것



$$Y = \begin{cases} 1, & \text{if } X_1 + X_2 \geq \theta \\ 0, & \text{if } X_1 + X_2 < \theta \end{cases}$$

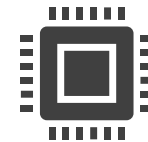
여기서  $\theta$ 은 '세타'라고 읽습니다.

그림 3-2 인공뉴런 TLU의 구조 © Hanbit Academy Inc.

# TLU를 이용하여 논리연산을 구현하려는 시도

인공지능 활용 Python language



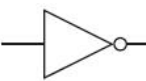

- 컴퓨터는 전자 소자들의 집합체로, 이 전자 소자들이 0과 1을 표현하는데, 이 소자들의 연산 수행을 가능케 하는 것이 논리회로임
- 논리회로 (Logic Circuit)란?
  - 논리연산을 수행하는 회로
- 논리연산자의 종류
  - AND 연산자: 두 조건이 모두 참일 때만 결과가 참이 되는 논리연산자
  - OR 연산자: 두 조건이 모두 거짓일 때만 결과가 거짓이 되는 논리연산자
  - NOT 연산자: 참과 거짓이 반대되는 논리연산자
  - XOR 연산자: 두 조건이 서로 다를 때만 결과가 참이 되는 논리연산자
  - NAND 연산자:
    - (AND 연산자와 NOT 연산자를 결합)
    - AND 연산자의 연산 결과를 부정하는 논리연산자
  - NOR 연산자:
    - (OR 연산자와 NOT 연산자를 결합)
    - 양쪽의 변수가 거짓일 때만 진리표의 참값이 되는 논리연산자



# 논리게이트 (Logic Gate)

인공지능 활용 Python language

표 2-3 논리연산자의 논리도, 논리식, 진리표

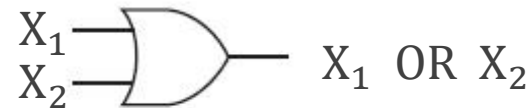
종류	논리도(심벌)	논리식(기호)	진리표																		
AND		$A \cdot B$ $A$ $B$	<table><tr><th colspan="2">입력</th><th>출력</th></tr><tr><th><math>A</math></th><th><math>B</math></th><th><math>A \text{ AND } B</math></th></tr><tr><td>0</td><td>0</td><td>0</td></tr><tr><td>0</td><td>1</td><td>0</td></tr><tr><td>1</td><td>0</td><td>0</td></tr><tr><td>1</td><td>1</td><td>1</td></tr></table>	입력		출력	$A$	$B$	$A \text{ AND } B$	0	0	0	0	1	0	1	0	0	1	1	1
입력		출력																			
$A$	$B$	$A \text{ AND } B$																			
0	0	0																			
0	1	0																			
1	0	0																			
1	1	1																			
OR		$A + B$ $A$ $B$	<table><tr><th colspan="2">입력</th><th>출력</th></tr><tr><th><math>A</math></th><th><math>B</math></th><th><math>A \text{ OR } B</math></th></tr><tr><td>0</td><td>0</td><td>0</td></tr><tr><td>0</td><td>1</td><td>1</td></tr><tr><td>1</td><td>0</td><td>1</td></tr><tr><td>1</td><td>1</td><td>1</td></tr></table>	입력		출력	$A$	$B$	$A \text{ OR } B$	0	0	0	0	1	1	1	0	1	1	1	1
입력		출력																			
$A$	$B$	$A \text{ OR } B$																			
0	0	0																			
0	1	1																			
1	0	1																			
1	1	1																			
NOT		$\overline{A}$	<table><tr><th>입력</th><th>출력</th></tr><tr><th><math>A</math></th><th>NOT <math>A</math></th></tr><tr><td>0</td><td>1</td></tr><tr><td>1</td><td>0</td></tr></table>	입력	출력	$A$	NOT $A$	0	1	1	0										
입력	출력																				
$A$	NOT $A$																				
0	1																				
1	0																				
XOR		$A \oplus B$ $A$ $B$	<table><tr><th colspan="2">입력</th><th>출력</th></tr><tr><th><math>A</math></th><th><math>B</math></th><th><math>A \text{ XOR } B</math></th></tr><tr><td>0</td><td>0</td><td>0</td></tr><tr><td>0</td><td>1</td><td>1</td></tr><tr><td>1</td><td>0</td><td>1</td></tr><tr><td>1</td><td>1</td><td>0</td></tr></table>	입력		출력	$A$	$B$	$A \text{ XOR } B$	0	0	0	0	1	1	1	0	1	1	1	0
입력		출력																			
$A$	$B$	$A \text{ XOR } B$																			
0	0	0																			
0	1	1																			
1	0	1																			
1	1	0																			

- 1: 참, True
- 0: 거짓, False



# TLU의 논리연산 구현 (1/6)

- TLU를 활용하면 논리연산인 논리합 (OR)과 논리곱 (AND)을 구현할 수 있음
- 참 또는 거짓의 두 진리 값을 입력 받아 새로운 진리 값을 계산
- ① 논리합 (OR)
  - 논리합은 입력 받은 두 진리 값 중 하나라도 참이면 결과를 참으로 계산하는 연산으로, 입력 받은 두 진리 값이 모두 거짓일 경우에만 결과를 거짓으로 계산함



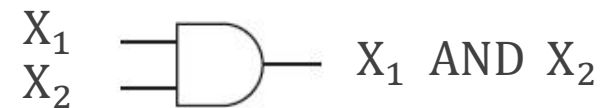
$X_1$	$X_2$	$X_1 \text{ OR } X_2$
참	참	참
참	거짓	참
거짓	참	참
거짓	거짓	거짓

$X_1$	$X_2$	$X_1 \text{ OR } X_2$
1	1	1
1	0	1
0	1	1
0	0	0

## TLU의 논리연산 구현 (2/6)

### ② 논리곱 (AND)

- 논리곱은 입력 받은 두 진리 값 중 하나라도 거짓이면 결과를 거짓으로 계산하는 연산으로, 입력 받은 두 진리 값이 모두 참일 경우에만 결과를 참으로 계산함



$X_1$	$X_2$	$X_1 \text{ AND } X_2$
참	참	참
참	거짓	거짓
거짓	참	거짓
거짓	거짓	거짓

$X_1$	$X_2$	$X_1 \text{ AND } X_2$
1	1	1
1	0	0
0	1	0
0	0	0

## TLU의 논리연산 구현 (3/6)

인공지능 활용 Python language

- TLU는 활성화 함수의 임계 값  $\theta$ 를 어떤 값으로 정하느냐에 따라 구현하는 논리연산이 결정됨
- 임계 값  $\theta$ 를 0.5로 정하여 논리합 (OR) 연산을 구현하면?

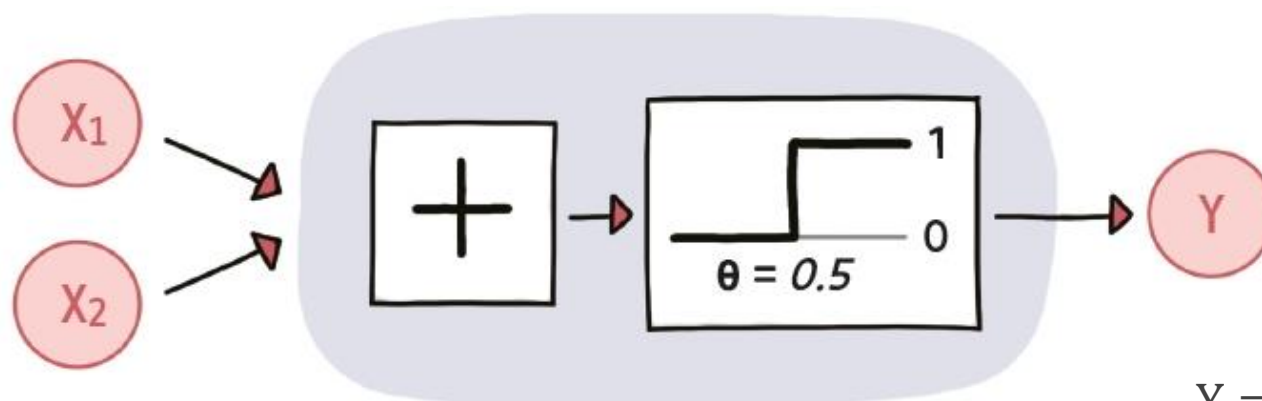


그림 3-3 TLU의 논리합(OR) 구현 : 임계값  $\theta = 0.5$

© Hanbit Academy Inc.

$$Y = \begin{cases} 1, & \text{if } X_1 + X_2 \geq 0.5 \\ 0, & \text{if } X_1 + X_2 < 0.5 \end{cases}$$

- 입력 값  $X_1$ 과  $X_2$ 를 더한 값이 0.5보다 크거나 같으면 출력 값  $Y$ 는 1로, 입력 값  $X_1$ 과  $X_2$ 를 더한 값이 0.5보다 작으면 출력 값  $Y$ 는 0으로 결정됨

# TLU의 논리연산 구현 (4/6)

인공지능 활용 Python language

- 이 관계를 표로 정리하면 [표 3-1]과 같음
- TLU의 입력 값과 출력 값에 사용된 1과 0을 각각 진리 값 참과 거짓으로 해석하면, 논리합 연산 ( $X_1 \text{ OR } X_2$ ) 결과와  $Y$ 의 값이 서로 같다는 것을 알 수 있음

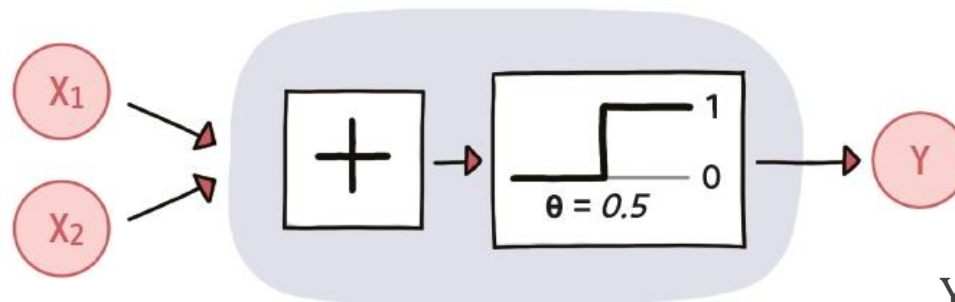


그림 3-3 TLU의 논리합(OR) 구현 : 임계값  $\theta = 0.5$

© Hanbit Academy Inc.

$$Y = \begin{cases} 1, & \text{if } X_1 + X_2 \geq 0.5 \\ 0, & \text{if } X_1 + X_2 < 0.5 \end{cases}$$

© Hanbit Academy Inc.

표 3-1 TLU의 논리합(OR) 구현 결과 : 임계값  $\theta = 0.5$

$X_1$	$X_2$	$X_1 + X_2$	$Y$	$X_1 \text{ OR } X_2$
1	1	2	1 ( $X_1 + X_2 = 2 \geq 0.5$ )	1
1	0	1	1 ( $X_1 + X_2 = 1 \geq 0.5$ )	1
0	1	1	1 ( $X_1 + X_2 = 1 \geq 0.5$ )	1
0	0	0	0 ( $X_1 + X_2 = 0 < 0.5$ )	0

# TLU의 논리연산 구현 (5/6)

- 임계 값  $\theta$ 를 1.5로 정하여 논리곱 연산을 구현하면?

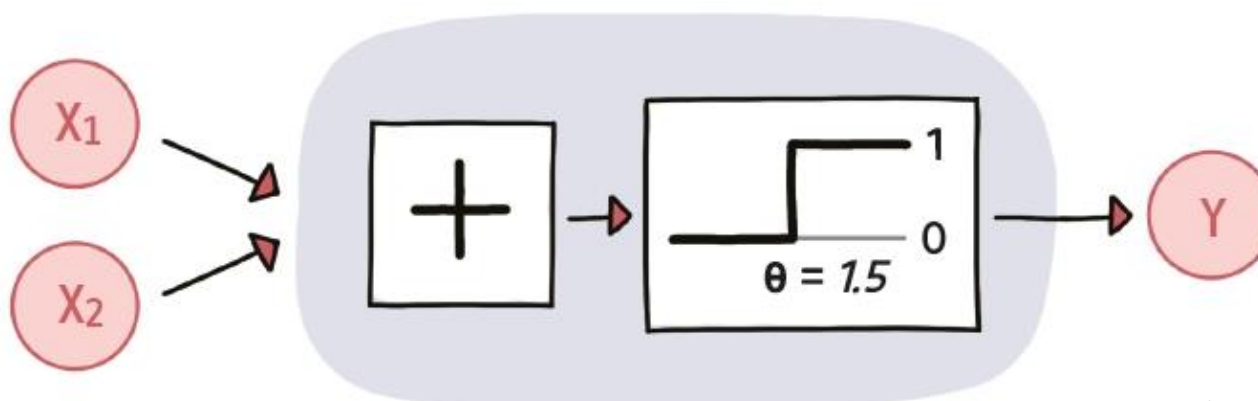


그림 3-4 TLU의 논리곱(AND) 구현 : 임계값  $\theta = 1.5$

© Hanbit Academy Inc.

$$Y = \begin{cases} 1, & \text{if } X_1 + X_2 \geq 1.5 \\ 0, & \text{if } X_1 + X_2 < 1.5 \end{cases}$$

- 입력 값  $X_1$ 과  $X_2$ 를 더한 값이 1.5보다 크거나 같으면 출력 값  $Y$ 는 1로, 입력 값  $X_1$ 과  $X_2$ 를 더한 값이 1.5보다 작으면 출력 값  $Y$ 는 0으로 결정됨

# TLU의 논리연산 구현 (6/6)

인공지능 활용 Python language

- 이 관계를 정리하면 [표 3-2]와 같은데, 논리곱 연산 ( $X_1$  AND  $X_2$ ) 결과와  $Y$ 의 값이 서로 같다는 것을 알 수 있음

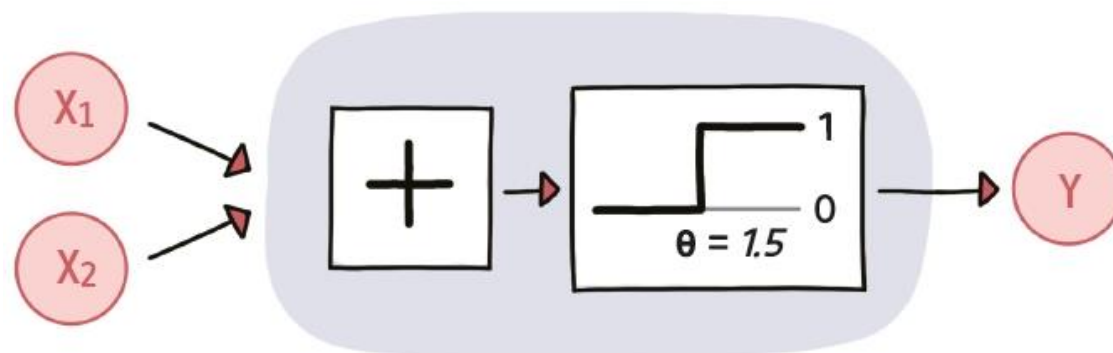


그림 3-4 TLU의 논리곱(AND) 구현 : 임계값  $\theta = 1.5$

© Hanbit Academy Inc.

$$Y = \begin{cases} 1, & \text{if } X_1 + X_2 \geq 1.5 \\ 0, & \text{if } X_1 + X_2 < 1.5 \end{cases}$$

표 3-2 TLU의 논리곱(AND) 구현 결과 : 임계값  $\theta = 1.5$

© Hanbit Academy Inc.

$X_1$	$X_2$	$X_1 + X_2$	$Y$	$X_1 \text{ AND } X_2$
1	1	2	1 ( $X_1 + X_2 = 2 \geq 1.5$ )	1
1	0	1	0 ( $X_1 + X_2 = 1 < 1.5$ )	0
0	1	1	0 ( $X_1 + X_2 = 1 < 1.5$ )	0
0	0	0	0 ( $X_1 + X_2 = 0 < 1.5$ )	0

AI Experts  
Who Lead  
The Future

## 02

### 인공뉴런 퍼셉트론 개요

다음 자료를 기반으로 제작  
난생처음 인공지능 입문 (출판사: 한빛아카데미)

## 02. 인공뉴런 퍼셉트론에 대해 알아봅시다

인공지능 활용 Python language

### • 퍼셉트론의 등장

- 1957년 코넬 항공 연구소의 프랑크 로젠블라트 (Frank Rosenblatt)가 퍼셉트론 (Perceptron)이라는 신경망 모델 발표
- 인간의 두뇌 움직임을 수학적으로 구성하여 당시 굉장한 이슈가 되었던 모델
- 퍼셉트론이 발표된 후 인공신경망에 대한 기대 폭증

### The New York Times

#### NEW NAVY DEVICE LEARNS BY DOING

July 8, 1958

"The Navy revealed the embryo of an electronic computer today that it expects will be able to walk, talk, see, write, reproduce itself and be conscious of its existence... Dr. Frank Rosenblatt, a research psychologist at the Cornell Aeronautical Laboratory, Buffalo, said Perceptrons might be fired to the planets as mechanical space explorers"

"컴퓨터는 걷고, 말하고, 보고, 쓰고, 재생산하고, 자의식을 갖출 것이다."

그림 9-1 퍼셉트론 관련 뉴욕타임즈 기사 © Hanbit Academy Inc.



## 02. 인공지능 퍼셉트론에 대해 알아봅시다

인공지능 활용 Python language

### • 퍼셉트론의 동작 원리 (1/2)

#### - 퍼셉트론 (Perceptron)

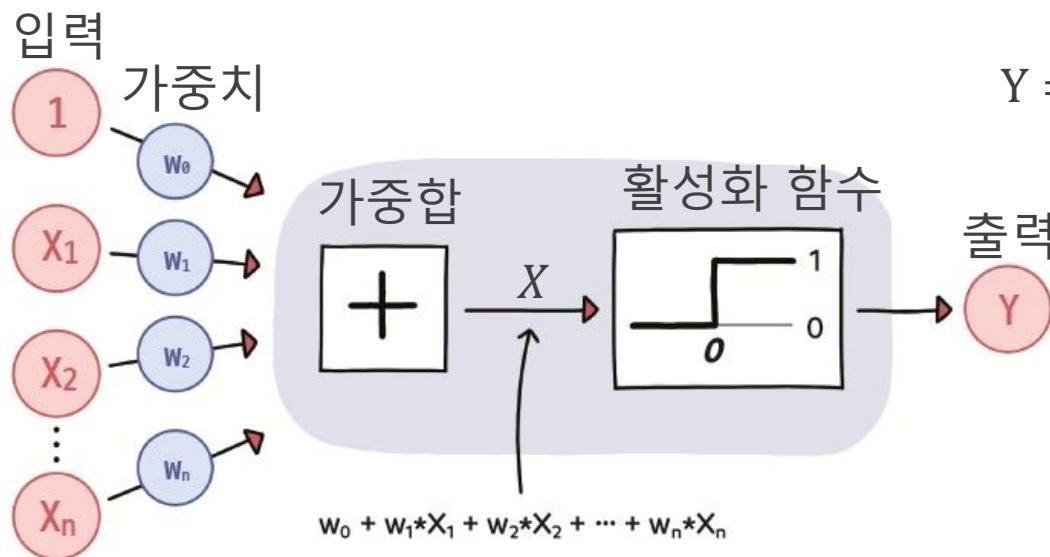
- TLU를 기반으로 하며, 가중치 (Weight)라는 개념을 추가한 모델
  - 맨 처음 시작은 '켜고 끄는 기능이 있는 신경'을 그물망 형태로 연결하면 사람의 뇌처럼 동작할 수 있다는 가능성을 처음으로 주장한 맥컬록-윌터 피츠(McCulloch-Walter Pitts)의 1943년 논문
- 가중치라는 개념은 시냅스에 신호가 반복적으로 전달되면 그 시냅스가 강화되는 생물학적 특성을 반영한 것
- 각 입력 값에 고유한 가중치를 부여하고, 입력 값에 가중치를 곱한 값의 총합을 임계 값과 비교하여 출력 값을 결정 (가중치가 클수록 결과에도 영향을 많이 미친다고 볼 수 있음)
- 임계 값은 0으로 정해 놓고 가중치를 적절히 선택하여 논리합 (OR) 또는 논리곱 (AND)을 결정함
- 학습 개념이 도입
  - 즉, 퍼셉트론으로 구성된 인공신경망에서 학습이란 입력의 가중치를 결정하는 작업이라고 할 수 있음

## 02. 인공뉴런 퍼셉트론에 대해 알아봅시다

인공지능 활용 Python language

### • 퍼셉트론의 동작 원리 (2/2)

- 퍼셉트론의 동작 원리는 입력 값에 가중치를 곱한 값들의 총합을  $X$ 라고 했을 때
  - 즉,  $X = w_0 + w_1 * X_1 + w_2 * X_2 + \dots + w_n * X_n$
- $X$ 의 값이 0 이상이면 출력 값  $Y$ 를 1로, 0보다 작으면 출력 값  $Y$ 를 0으로 결정



- 입력 (Input): AND 또는 OR 연산을 위한 입력 신호
- 가중치 (Weight): 입력 신호에 부여되는 중요도
- 가중합 (Weighted Sum): 입력값과 가중치의 곱을 모두 합한 값
- 활성화 함수 (Activation Function): 어떠한 신호를 입력 받아 이를 적절하게 처리하여 출력해 주는 함수로, 가중합이 임계치 (Threshold)를 넘어가면 1, 그렇지 않으면 0을 출력함
- 출력 (Output): 최종 결과 (분류)

그림 3-5 인공뉴런 퍼셉트론의 구조와 동작 원리 © Hanbit Academy Inc.

## 02. 인공뉴런 퍼셉트론에 대해 알아봅시다

### • 퍼셉트론의 논리합 (OR) 연산 구현 (1/2)

- 퍼셉트론은 가중치를 적절히 선택하여 논리합 (OR)을 구현해 봅시다
- 가중치  $w_0 = -5$ ,  $w_1 = 8$ ,  $w_2 = 8$ 을 사용하여 논리합 연산을 구현하면?

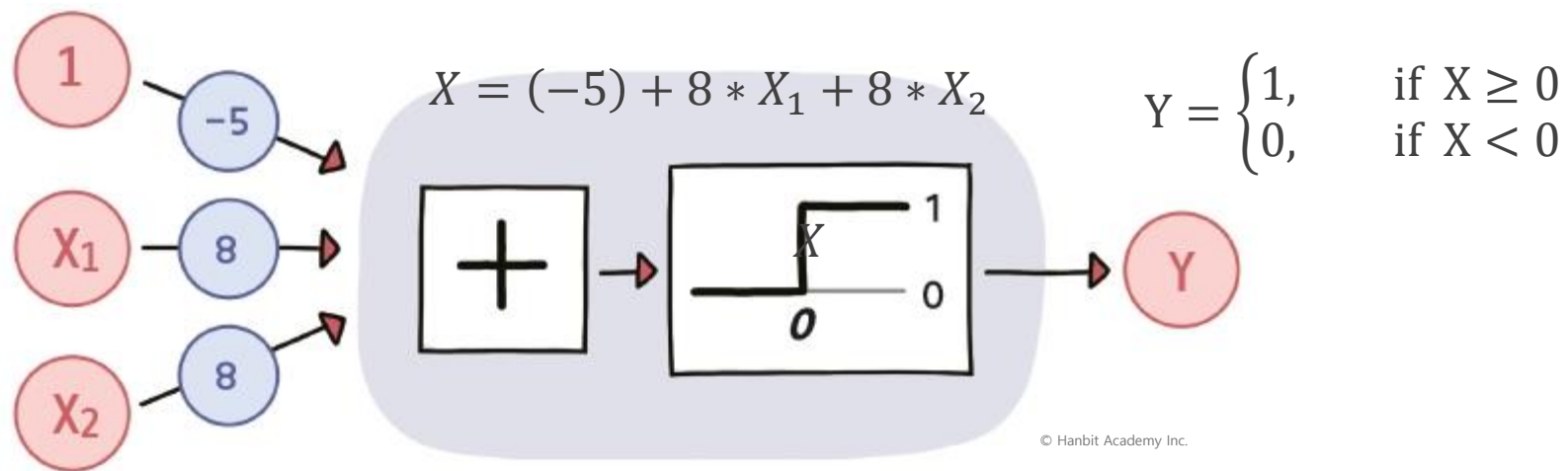


그림 3-6 퍼셉트론의 논리합(OR) 구현 : 가중치  $w_0 = -5$ ,  $w_1 = 8$ ,  $w_2 = 8$

- 입력 값에 가중치를 곱한 값들의 총합  $((-5) + 8 * X_1 + 8 * X_2)$ 이 0보다 크거나 같으면
  - 출력 값  $Y$ 는 1로,
  - 0보다 작으면 출력 값  $Y$ 는 0으로 결정됨

## 02. 인공뉴런 퍼셉트론에 대해 알아봅시다

인공지능 활용 Python language

### • 퍼셉트론의 논리합 (OR) 연산 구현 (2/2)

- 논리합 연산 ( $X_1$  OR  $X_2$ ) 결과와  $Y$ 의 값이 서로 같다는 것을 알 수 있음

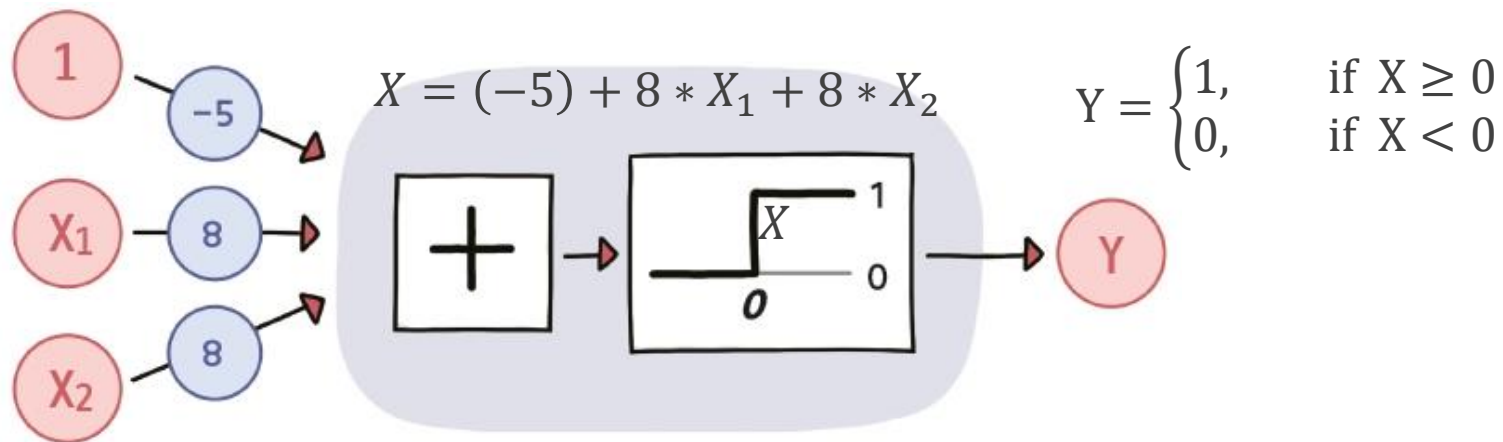


그림 3-6 퍼셉트론의 논리합(OR) 구현 : 가중치  $w_0 = -5$ ,  $w_1 = 8$ ,  $w_2 = 8$

© Hanbit Academy Inc.

표 3-3 퍼셉트론의 논리합(OR) 구현 결과 : 가중치  $w_0 = -5$ ,  $w_1 = 8$ ,  $w_2 = 8$

© Hanbit Academy Inc.

$X_1$	$X_2$	$(-5) + 8 * X_1 + 8 * X_2$	$Y$	$X_1$ OR $X_2$
1	1	11	1 ( $11 \geq 0$ )	1
1	0	3	1 ( $3 \geq 0$ )	1
0	1	3	1 ( $3 \geq 0$ )	1
0	0	-5	0 ( $-5 < 0$ )	0

## 02. 인공지능 퍼셉트론에 대해 알아봅시다

인공지능 활용 Python language

### • 퍼셉트론의 논리곱 (AND) 연산 구현 (1/2)

- 퍼셉트론은 가중치를 적절히 선택하여 논리곱 (AND)을 구현해 봅시다
- 가중치  $w_0 = -10$ ,  $w_1 = 8$ ,  $w_2 = 8$ 을 사용하여 논리곱 연산을 구현하면?

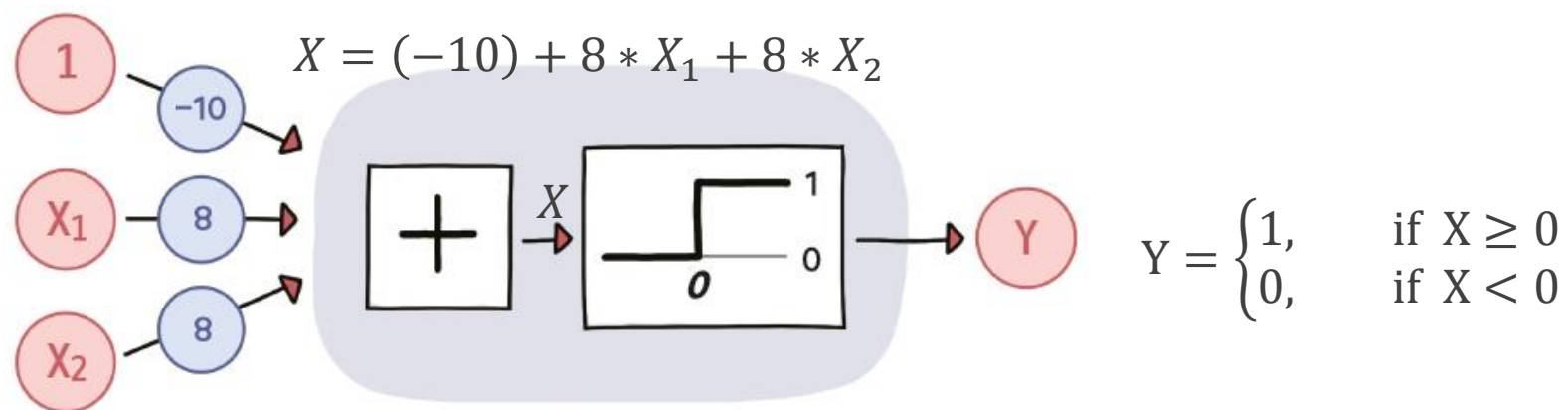


그림 3-7 퍼셉트론의 논리곱(AND) 구현 : 가중치  $w_0 = -10$ ,  $w_1 = 8$ ,  $w_2 = 8$  © Hanbit Academy Inc.

- 입력 값에 가중치를 곱한 값들의 총합  $((-10) + 8 * X_1 + 8 * X_2)$ 
  - 0보다 크거나 같으면 출력값  $Y$ 는 1로,
  - 0보다 작으면 출력 값  $Y$ 는 0으로 결정됨

## 02. 인공뉴런 퍼셉트론에 대해 알아봅시다

인공지능 활용 Python language

### • 퍼셉트론의 논리곱 (AND) 연산 구현 (2/2)

- 논리곱 연산 ( $X_1$  AND  $X_2$ ) 결과와  $Y$ 의 값이 서로 같음을 알 수 있음

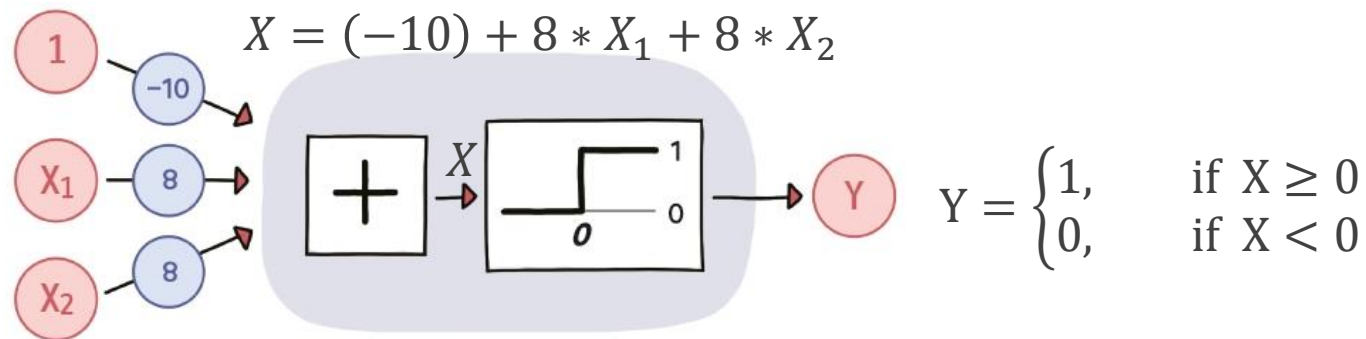


그림 3-7 퍼셉트론의 논리곱(AND) 구현 : 가중치  $w_0 = -10$ ,  $w_1 = 8$ ,  $w_2 = 8$  © Hanbit Academy Inc.

표 3-4 퍼셉트론의 논리곱(AND) 구현 결과 : 가중치  $w_0 = -10$ ,  $w_1 = 8$ ,  $w_2 = 8$  © Hanbit Academy Inc.

$X_1$	$X_2$	$(-10) + 8 * X_1 + 8 * X_2$	$Y$	$X_1$ AND $X_2$
1	1	6	1 ( $6 \geq 0$ )	1
1	0	-2	0 ( $-2 < 0$ )	0
0	1	-2	0 ( $-2 < 0$ )	0
0	0	-10	0 ( $-10 < 0$ )	0

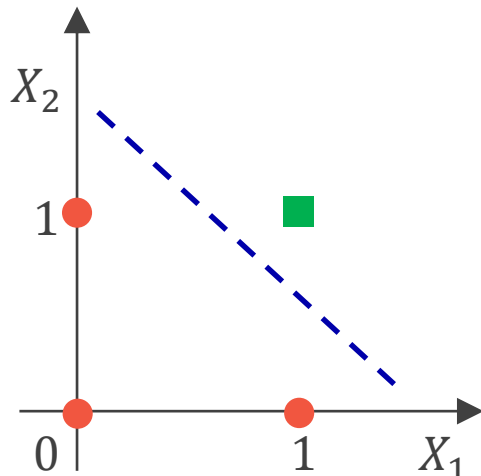
## 02. 인공뉴런 퍼셉트론에 대해 알아봅시다

인공지능 활용 Python language

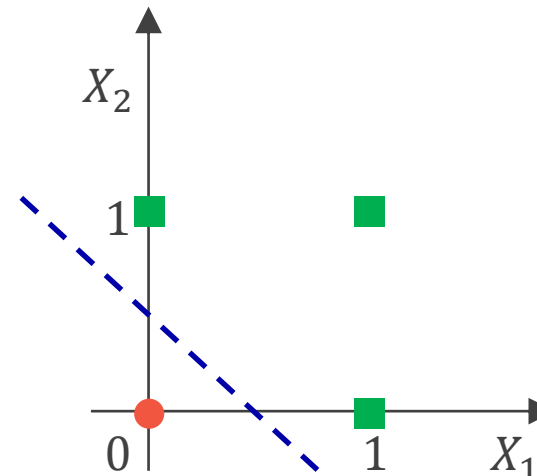
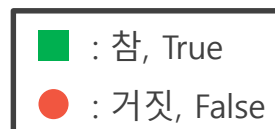
### • 퍼셉트론의 한계 (1/4)

- 초기 퍼셉트론을 이용한 문제해결은 지금보면 AND와 OR 같은 간단한 연산이었으나, 당시 기계가 AND와 OR 연산을 스스로 풀 수 있으면 이를 조합해 어떤 문제라도 전부 풀어낼 수 있다고 생각했음
- AND와 OR 연산의 경우 퍼셉트론을 통한 선형 분리가 가능한 것을 확인 가능

$$Y = \begin{cases} 1, & \text{if } X = w_0 + w_1 * X_1 + w_2 * X_2 \geq 0 \\ 0, & \text{if } X = w_0 + w_1 * X_1 + w_2 * X_2 < 0 \end{cases}$$



AND 연산



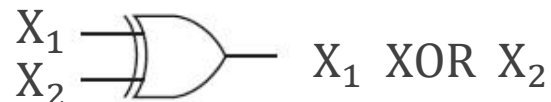
OR 연산

## 02. 인공지능 퍼셉트론에 대해 알아봅시다

인공지능 활용 Python language

### • 퍼셉트론의 한계 (2/4)

- 퍼셉트론이 처음 등장했을 때 사람들은 인간의 뇌와 같은 인공지능이 곧 나올 것이라는 희망을 가졌음
- 1969년, 마빈 민스키와 시모어 페퍼트가 퍼셉트론으로는 배타적 논리합 (XOR) 같은 논리연산이 불가능하다는 사실을 수학적으로 증명 → 인공지능에 대한 실망감이 커짐
  - 배타적 논리합 (XOR : Exclusive-OR)
    - 두 입력 값이 서로 다를 때만 참을 갖는 논리연산으로, 주로 네트워크로 전송된 데이터의 오류 검증 등에 사용됨



$X_1$	$X_2$	$X_1 \text{ XOR } X_2$
참	참	거짓
참	거짓	참
거짓	참	참
거짓	거짓	거짓

$X_1$	$X_2$	$X_1 \text{ XOR } X_2$
1	1	0
1	0	1
0	1	1
0	0	0



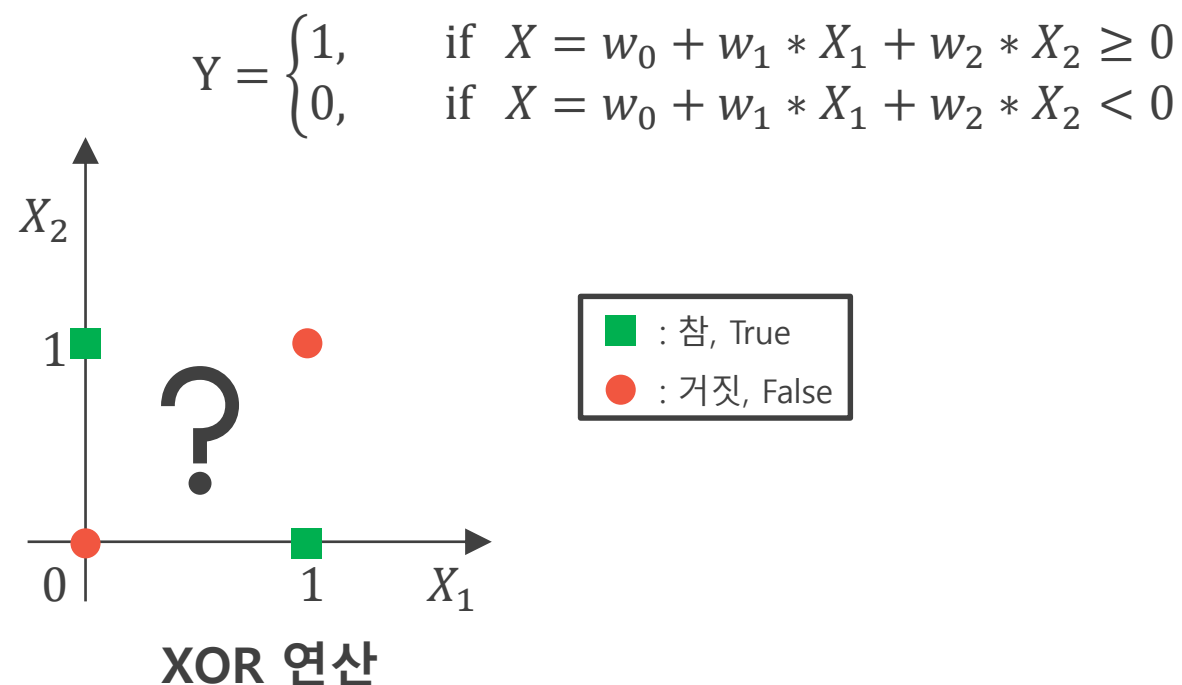
## 02. 인공뉴런 퍼셉트론에 대해 알아보시다

인공지능 활용 Python language

### • 퍼셉트론의 한계 (3/4)

- XOR 연산의 경우 퍼셉트론을 통한 선형 분리가 불가능한 것을 확인 가능

$X_1$	$X_2$	$X_1 \text{ XOR } X_2$
1	1	0
1	0	1
0	1	1
0	0	0



## 02. 인공지능 퍼셉트론에 대해 알아봅시다

인공지능 활용 Python language

- 퍼셉트론의 한계 (4/4)

- 퍼셉트론을 여러 개 쌓아 올린 다층 퍼셉트론(Multi Layer Perceptron, MLP)을 통해 XOR 연산에 대한 문제는 해결될 수 있지만,  
“각각의 가중치 (Weight)와 편향 (Bias)을 학습시킬 방법이 없다”라는 결론을 내놓으면서 퍼셉트론을 이용한 학습의 한계가 언급됨
- 또한 당시 인공지능 기술이 복잡하고 현실적인 문제를 해결하기에는 역부족이라는 사실이 알려지면서, 1970년대 초부터 1980년까지 인공지능 역사의 첫 번째 겨울 (First AI Winter)을 겪게 됨

AI Experts  
Who Lead  
The Future

# 03

## 다층 퍼셉트론

다음 자료를 기반으로 제작  
난생처음 인공지능 입문 (출판사: 한빛아카데미)

## 03. 다층 퍼셉트론에 대해 살펴봅시다

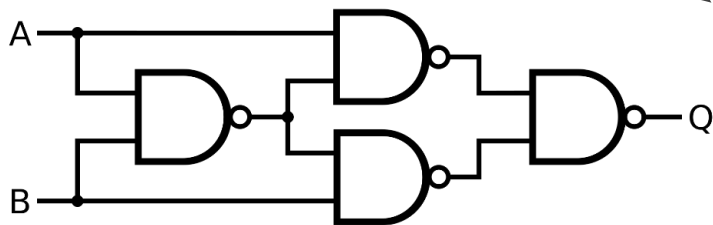
인공지능 활용 Python language

### • 다층 퍼셉트론 (MLP)의 등장 (1/9)

- 퍼셉트론의 한계로 배타적 논리합 (XOR) 연산을 구현하지 못한다고 배웠음

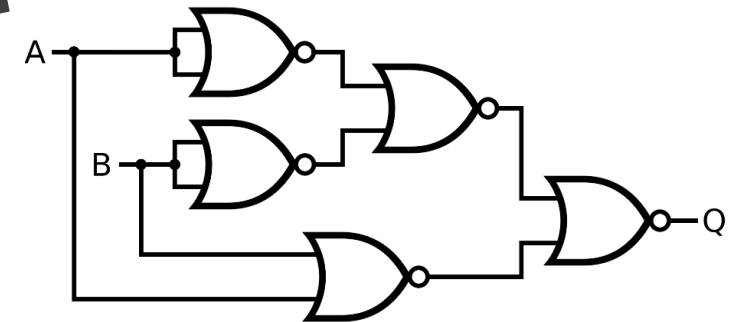
그렇다면, 기존에 XOR 연산자는 어떻게 구현되었을까?

NAND 게이트만 사용하여 구성



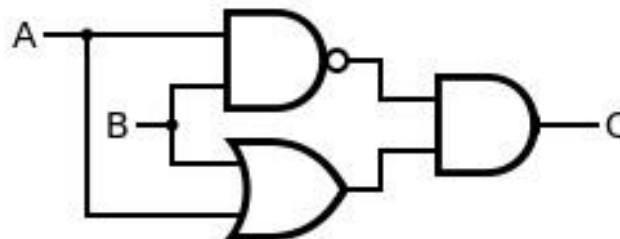
[사진출처] [https://ko.wikipedia.org/wiki/XOR\\_게이트](https://ko.wikipedia.org/wiki/XOR_게이트)

NOR 게이트만 사용하여 구성



[사진출처] [https://ko.wikipedia.org/wiki/XOR\\_게이트](https://ko.wikipedia.org/wiki/XOR_게이트)

3개의 게이트를 혼합하여 구성



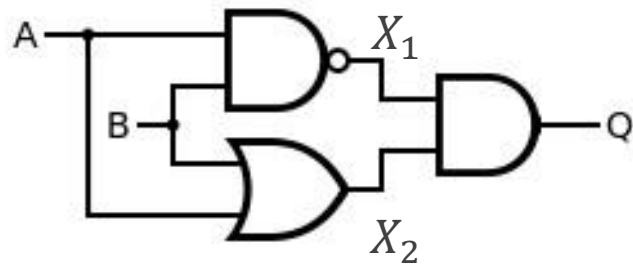
[사진출처] [https://ko.wikipedia.org/wiki/XOR\\_게이트](https://ko.wikipedia.org/wiki/XOR_게이트)

## 03. 다층 퍼셉트론에 대해 살펴봅시다

인공지능 활용 Python language

### • 다층 퍼셉트론 (MLP)의 등장 (2/9)

- 3개의 게이트 (Gate)를 혼합하여 구성된 배타적 논리합 (XOR) 게이트를 살펴봅시다



A	B	$X_1 (= A \text{ NAND } B)$	$X_2 (= A \text{ OR } B)$	$Q (= X_1 \text{ AND } X_2)$
1	1	0	1	0
1	0	1	1	1
0	1	1	1	1
0	0	1	0	0

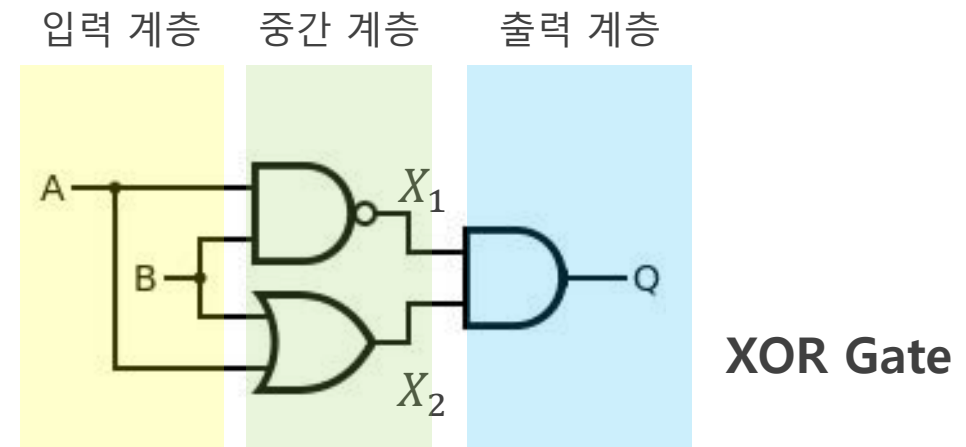
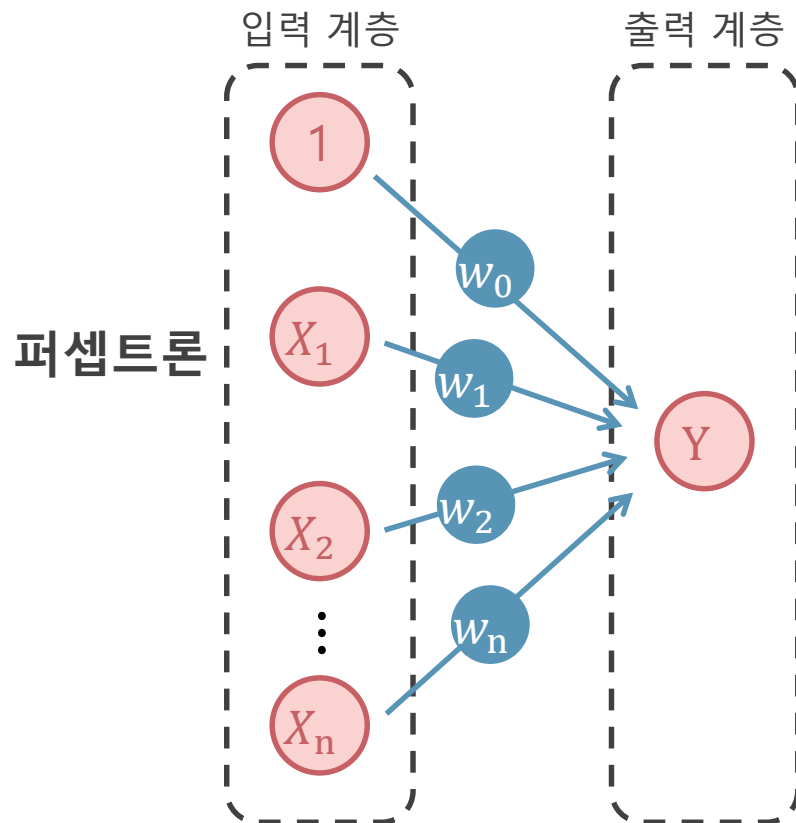
- A, B: 입력값
- $X_1, X_2$ : 중간 결과값
- Q: (최종) 결과값

## 03. 다층 퍼셉트론에 대해 살펴봅시다

인공지능 활용 Python language

### • 다층 퍼셉트론 (MLP)의 등장 (3/9)

- 퍼셉트론과 배타적 논리합 (XOR) 게이트를 비교해 봅시다



- $A$ ,  $B$ : 입력값
- $X_1$ ,  $X_2$ : 중간 결과값
- $Q$ : (최종) 결과값

퍼셉트론에 중간 계층을 추가하면, XOR 게이트를 구현할 수 있지 않을까?

## 03. 다층 퍼셉트론에 대해 살펴봅시다

### • 다층 퍼셉트론 (MLP)의 등장 (4/9)

- 퍼셉트론의 한계는 1986년, 데이비드 럼멜하트 (David Everett Rumelhart)와 데이비드 클라렌스 맥클랜드 (David Clarence McClelland)가 다층 퍼셉트론 개념과 다층 퍼셉트론을 학습시킬 수 있는 방법인 역전파 (Backpropagation) 알고리즘을 발표하면서 극복할 수 있게 됨
- 다층 퍼셉트론 (Multi Layer Perceptron, MLP)
  - 퍼셉트론의 입력 계층과 출력 계층 사이에 은닉 계층 (Hidden Layer)을 추가한 것
- 은닉 계층을 추가함으로써 단층 퍼셉트론으로는 해결하지 못한 배타적 논리합 (XOR)의 연산을 구현할 수 있게 됨

### 03. 다층 퍼셉트론에 대해 살펴봅시다

인공지능 활용 Python language

#### • 다층 퍼셉트론 (MLP)의 등장 (5/9)

- 입력 계층과 출력 계층 사이에 하나의 은닉 계층을 추가한 다층 퍼셉트론으로 배타적 논리합 연산을 구현하면?

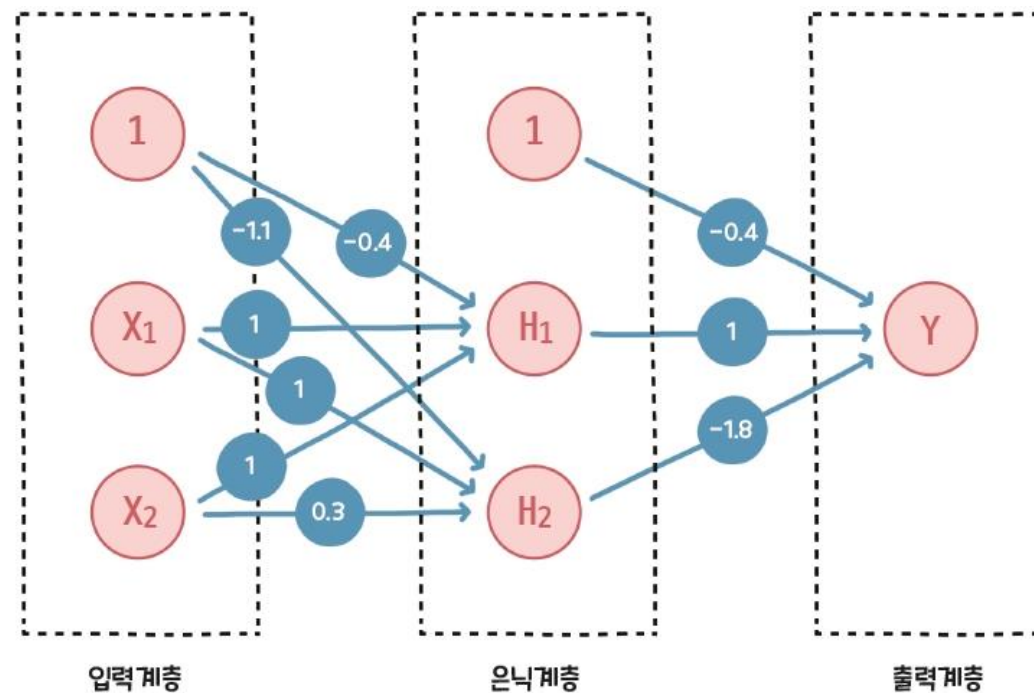


그림 3-8 다층 퍼셉트론의 배타적 논리합(XOR) 구현 © Hanbit Academy Inc.



### 03. 다층 퍼셉트론에 대해 살펴봅시다

인공지능 활용 Python language

#### • 다층 퍼셉트론 (MLP)의 등장 (6/9)

- [그림 3-8]의  $H_1$ 과  $H_2$ 의 출력 값을 계산하면 [표 3-5]와 같음
- 그림상  $H_1$ 과  $H_2$  값의 입력이 복잡하므로 출력 값 계산에 적용될 가중치를 정확히 구별하는데 유의해야 함

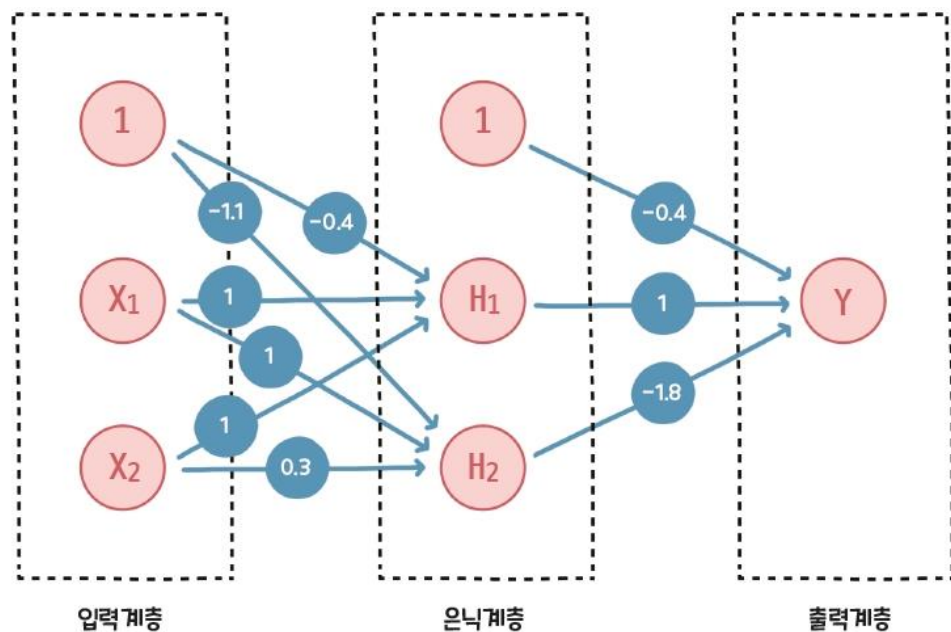


그림 3-8 다층 퍼셉트론의 배타적 논리합(XOR) 구현

© Hanbit Academy Inc.

표 3-5 은닉계층  $H_1$ 과  $H_2$ 의 출력값 계산

© Hanbit Academy Inc.

$X_1$	$X_2$	$H_1 (= -0.4 + 1 \cdot X_1 + 1 \cdot X_2)$	$H_2 (= -1.1 + 1 \cdot X_1 + 0.3 \cdot X_2)$
0	0	0 ( $-0.4 + 1 \cdot 0 + 1 \cdot 0 < 0$ )	0 ( $-1.1 + 1 \cdot 0 + 0.3 \cdot 0 < 0$ )
0	1	1 ( $-0.4 + 1 \cdot 0 + 1 \cdot 1 > 0$ )	0 ( $-1.1 + 1 \cdot 0 + 0.3 \cdot 1 < 0$ )
1	0	1 ( $-0.4 + 1 \cdot 1 + 1 \cdot 0 > 0$ )	0 ( $-1.1 + 1 \cdot 1 + 0.3 \cdot 0 < 0$ )
1	1	1 ( $-0.4 + 1 \cdot 1 + 1 \cdot 1 > 0$ )	1 ( $-1.1 + 1 \cdot 1 + 0.3 \cdot 1 > 0$ )

### 03. 다층 퍼셉트론에 대해 살펴봅시다

인공지능 활용 Python language

#### • 다층 퍼셉트론 (MLP)의 등장 (7/9)

- $H_1$ 과  $H_2$ 의 값을 입력으로 받는 출력 계층  $Y$ 의 값을 계산하면 [표 3-6]과 같음
- 결과적으로 출력  $Y$ 값이  $X_1$ 과  $X_2$ 의 배타적 논리합 ( $X_1 \text{ XOR } X_2$ )의 연산 결과와 동일함을 알 수 있음

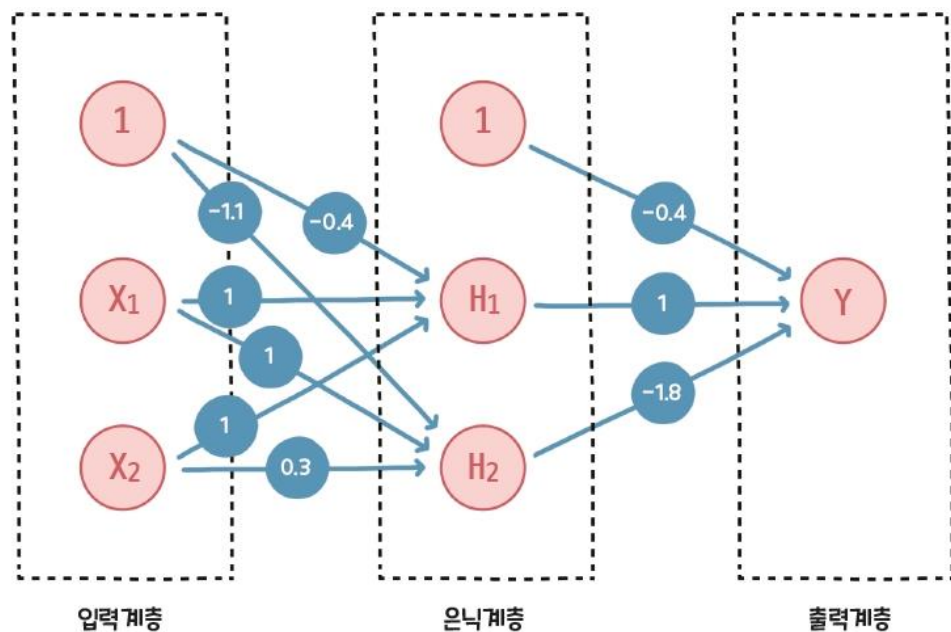


그림 3-8 다층 퍼셉트론의 배타적 논리합(XOR) 구현 © Hanbit Academy Inc.

표 3-6 출력계층 Y 값 계산( $X_1 \text{ XOR } X_2$ 와 동일) © Hanbit Academy Inc.

$X_1$	$X_2$	$H_1$	$H_2$	$Y (= -0.4 + 1*H_1 + (-1.8)*H_2)$	$X_1 \text{ XOR } X_2$
0	0	0	0	$0 (-0.4 + 1*0 + (-1.8)*0 < 0)$	0
0	1	1	0	$1 (-0.4 + 1*1 + (-1.8)*0 > 0)$	1
1	0	1	0	$1 (-0.4 + 1*1 + (-1.8)*0 > 0)$	1
1	1	1	1	$0 (-0.4 + 1*1 + (-1.8)*1 < 0)$	0

퍼셉트론에 은닉 계층을 추가하면, XOR 게이트를 구현 가능!

### 03. 다층 퍼셉트론에 대해 살펴봅시다

인공지능 활용 Python language

#### • 다층 퍼셉트론 (MLP)의 등장 (8/9)

- 은닉 계층을 추가함으로써 인공지능망이 비선형 (Nonlinear) 문제를 잘 풀 수 있게 됨

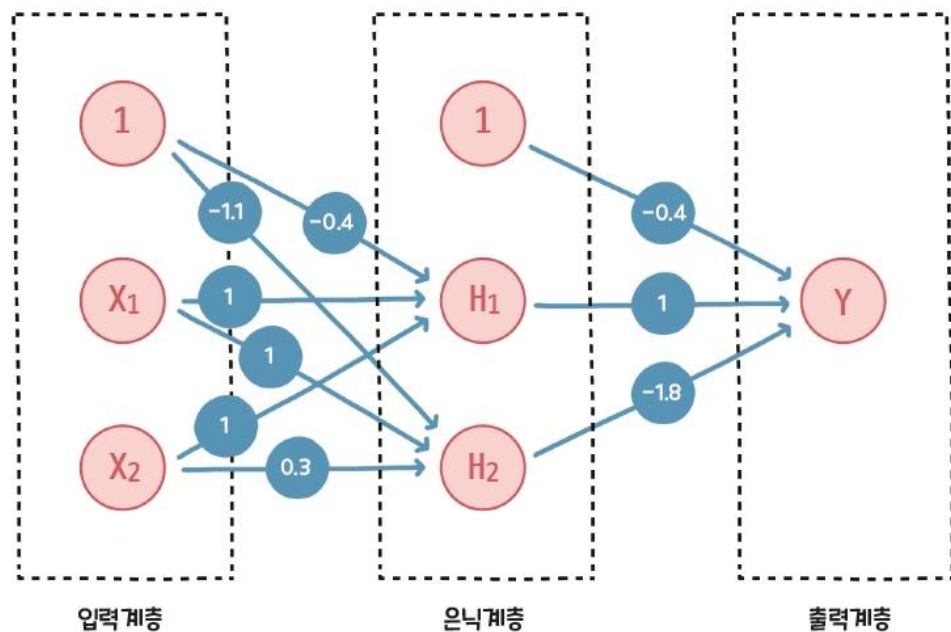
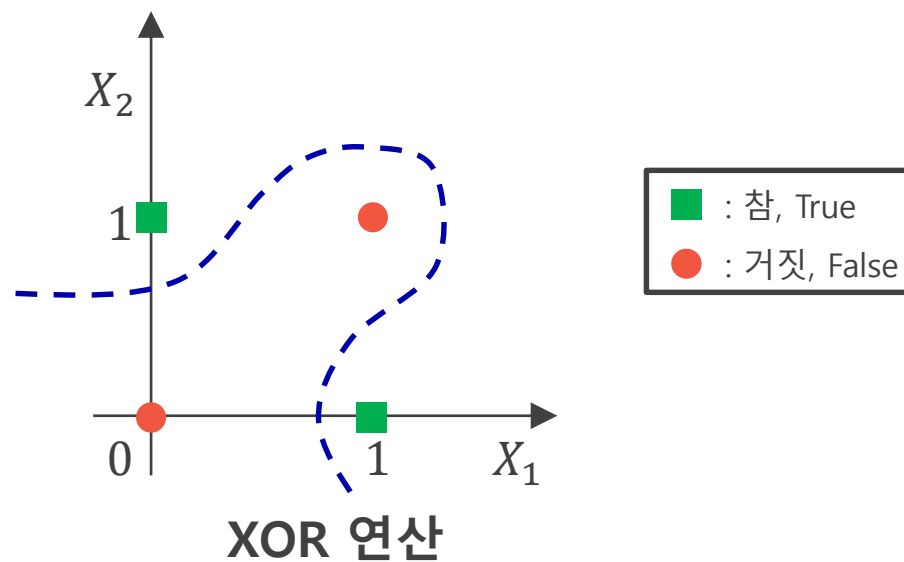


그림 3-8 다층 퍼셉트론의 배타적 논리합(XOR) 구현

© Hanbit Academy Inc.



### 03. 다층 퍼셉트론에 대해 살펴봅시다

인공지능 활용 Python language

#### • 다층 퍼셉트론 (MLP)의 등장 (9/9)

- 당시에도 다층 퍼셉트론 (Multiple Layer Perceptron, MLP)을 이용하면 XOR 문제를 해결할 수 있다는 사실을 알고 있었음
  - 하지만 은닉 계층의 가중치를 학습시킬 방법이 없다는 점이 문제였음
- 그러다가 이 문제를 해결할 방법인 역전파 (backpropagation) 알고리즘이 등장하면서 퍼셉트론의 한계를 극복할 수 있게 됨

이 많은 가중치를 어떻게 학습시키지?

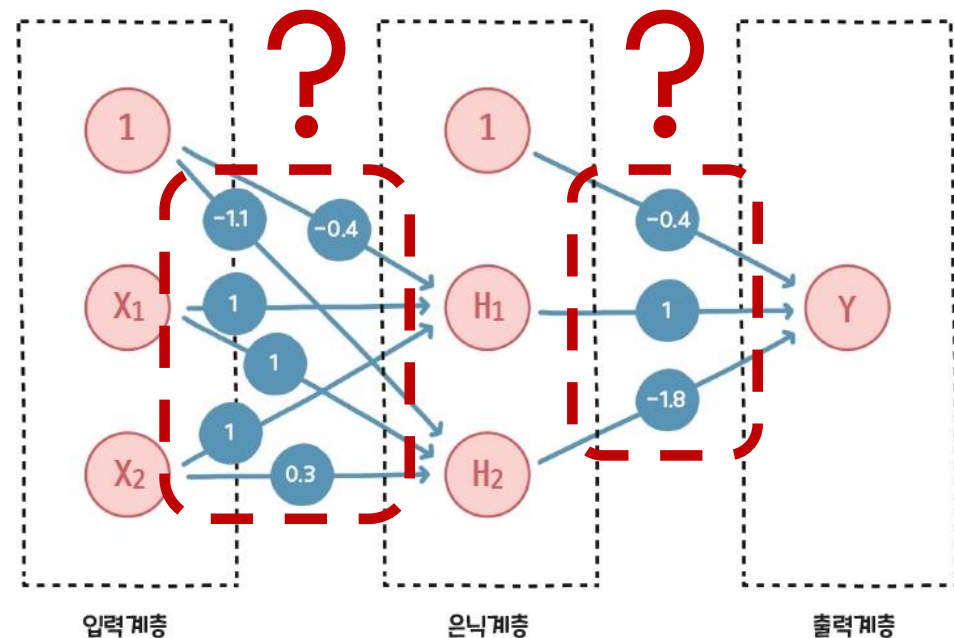


그림 3-8 다층 퍼셉트론의 배타적 논리합(XOR) 구현

© Hanbit Academy Inc.

### 03. 다층 퍼셉트론에 대해 살펴봅시다

인공지능 활용 Python language

#### • 역전파 (Backpropagation) 알고리즘 (1/3)

- 출력 계층에서 계산된 오차 값을 입력 계층 방향으로 다시 보내 가중치를 수정하는 방법
- 개념적으로 봤을 때 가중치가 높은 입력 값이 오차 발생에 더 큰 영향을 줬다고 가정하여 보다 큰 수정의 책임을 부여하는 것
  - [그림 3-9]는 출력 계층  $Y$ 의 값에 오차가 1.0만큼 발생하였을 때, 입력 값  $H_1$ 과  $H_2$ 로 가중치 비중만큼 오차를 역전파하여 수정의 책임을 전가하는 원리를 보여 줌
  - 이에 따라 입력 값  $H_1$ 에게는 오차 0.8에 대한,  $H_2$ 에게는 오차 0.2에 대한 가중치 수정의 책임이 주어짐

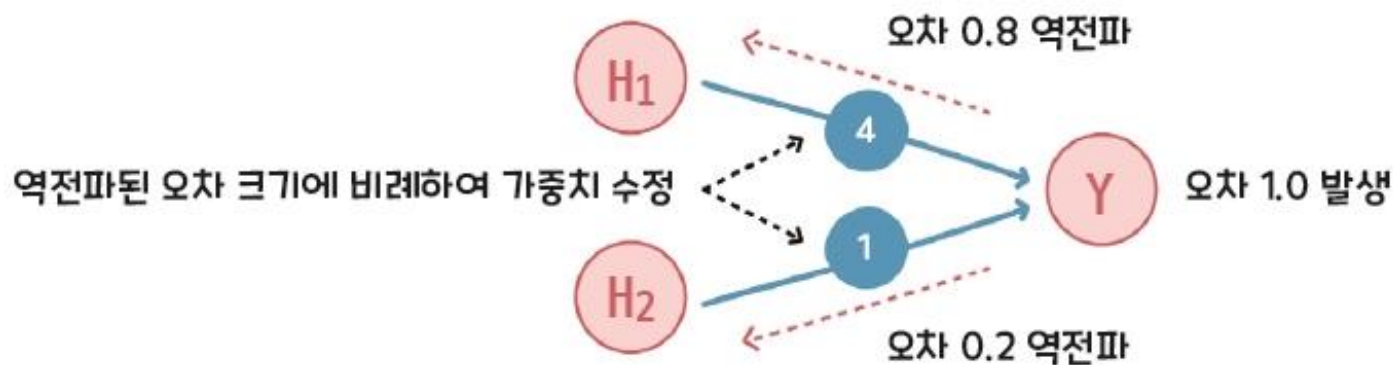


그림 3-9 오차 역전파 알고리즘의 원리

© Hanbit Academy Inc.

### 03. 다층 퍼셉트론에 대해 살펴봅시다

인공지능 활용 Python language

#### • 역전파 (Backpropagation) 알고리즘 (2/3)

- 사라지는 경사도 (Vanishing Gradient, 기울기 소멸) 문제
  - 여러 계층을 역으로 올라가다 보니 전파되는 오차 값이 현저히 작아져 학습 효과가 사라진다는 문제 발생
- 복잡한 문제를 해결하기 위해 은닉 계층의 개수를 계속 늘려갈 때 치명적인 약점으로 작용

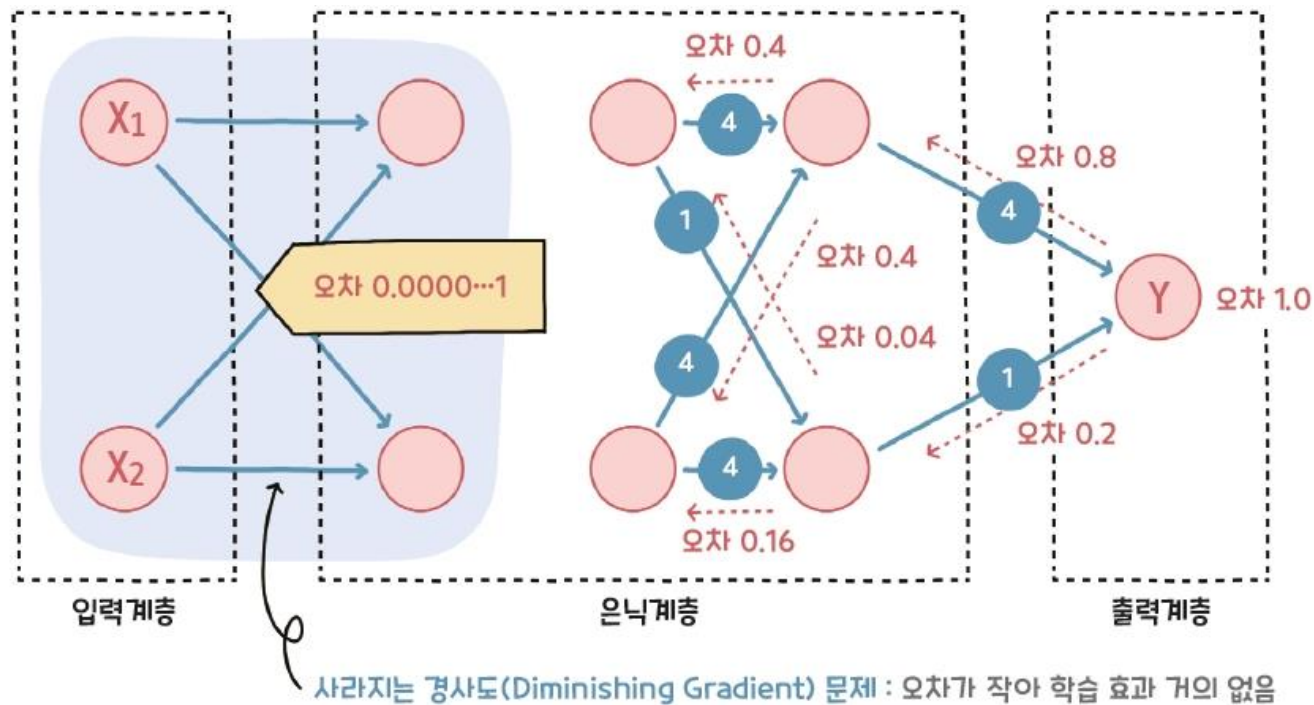


그림 3-10 오차 역전파 알고리즘 문제점 : 사라지는 경사도

© Hanbit Academy Inc.

## 03. 다층 퍼셉트론에 대해 살펴봅시다

인공지능 활용 Python language

- 역전파 (Backpropagation) 알고리즘 (3/3)
  - 심층신경망 (Deep Neural Network, DNN)
    - 은닉 계층이 2개 이상인 인공신경망 (Artificial Neural Network, ANN)
  - 여러 개의 은닉 계층으로 구성된 DNN은 “사라지는 경사도” 문제 때문에 학습이 잘 안 된다는 점이 문제로 거론됨
  - 이로 인해, 인공신경망 분야는 2000년대 초반까지 두 번째 겨울 (Second AI Winter)을 겪음
  - 2000년대에 들어,
    - 가중치 계산에 확률 개념을 도입하거나
    - 입력 값을 출력 값에 복사하는 방식의 사전학습 방법 (Stacked Autoencoder for Pretraining),
    - 확률적인 방법으로 DNN의 일부 뉴런 (퍼셉트론)을 일부러 사용하지 않게 만드는 드롭아웃 (Dropout) 방법 등의 등장으로 인공지능 분야는 딥러닝 (Deep Learning)을 대표로 한 “세 번째 봄”을 맞이함

AI Experts  
Who Lead  
The Future

# 04

---

## 딥러닝 개요

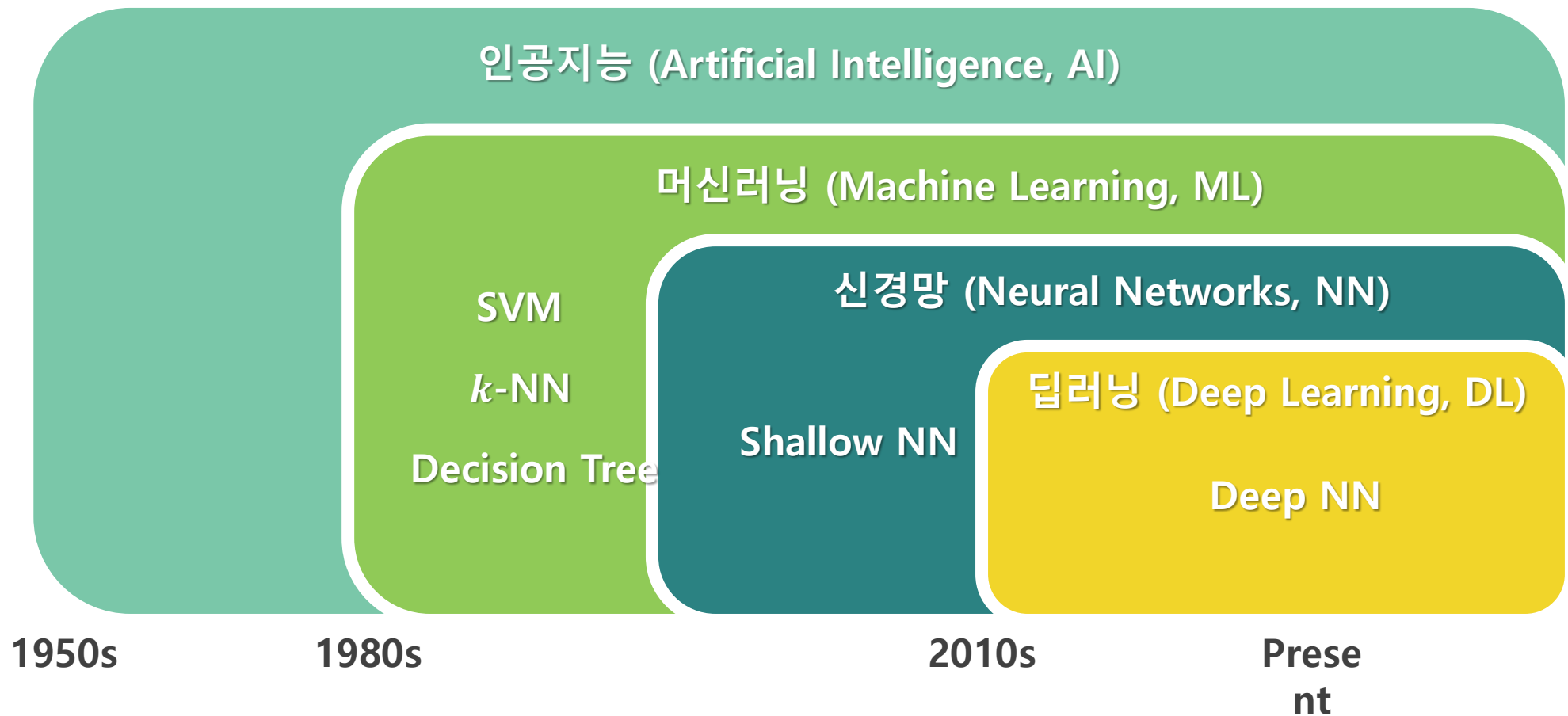
다음 자료를 기반으로 제작  
난생처음 인공지능 입문 (출판사: 한빛아카데미)



# 01. 딥러닝에 대해 알아봅시다

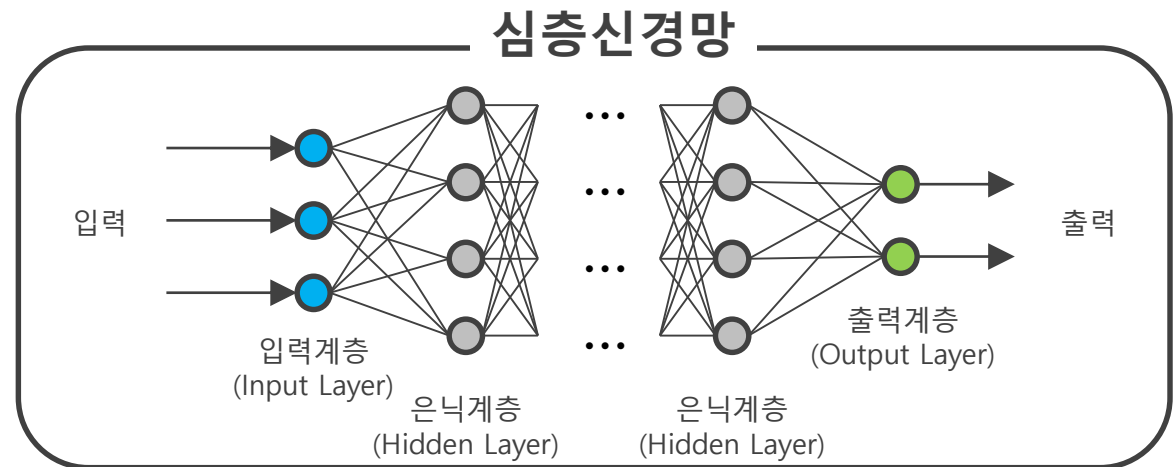
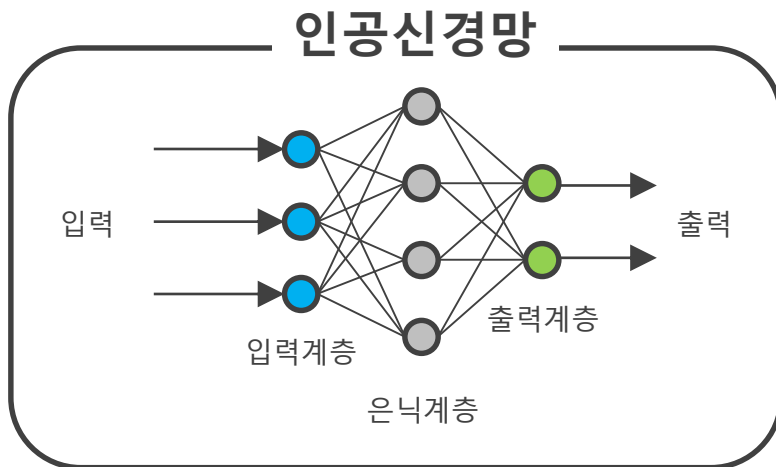
인공지능 활용 Python language

- 인공지능, 머신러닝, 딥러닝의 관계



## • 딥러닝의 개념

- 딥러닝 (Deep Learning, 심층 학습)
  - **인공신경망 (Artificial Neural Network, ANN)을 기반으로 하는 머신러닝 방법들의 집합**
    - 심층신경망 (Deep Neural Network, DNN), 합성곱 신경망 (Convolution NN, CNN), 순환 신경망 (Recurrent NN, RNN) 등
  - **2000년대 초반까지 형성된 인공신경망에 대한 부정적인 인식 때문에, 기존 인공신경망의 한계를 뛰어넘은 인공신경망에 대해 “딥러닝”이라고 용어를 붙여 리브랜딩 (Re-branding)**
- 심층신경망 (Deep Neural Network, DNN)
  - **입력계층과 출력계층 사이에 2개 이상의 은닉계층 (Hidden Layer)을 갖는 신경망을 뜻함**



# 01. 딥러닝에 대해 알아봅시다

인공지능 활용 Python language

## • 딥러닝의 특징 (1/3)

### - 특성/특징 추출 (Feature Extraction)

- 데이터로부터 올바른 결과 도출에 도움이 되는 정보를 뽑아내는 작업
- 예를 들어,
  - 고양이와 개를 구분하고자 할 때 올바른 결과 도출에 도움이 되는 중요한 특징을 추출하는 작업을 사람이 수행하는 경우, 얼마나 좋은 특징을 잘 추출 하느냐는 전적으로 그 사람의 능력에 좌우됨
- 데이터에 대한 이해도가 부족하다면, 중요한 특징을 빠뜨리거나 잘못된 정보를 선택하여 머신러닝의 학습 결과에 안 좋은 영향을 미칠 수 있음
- 만약 “발가락 개수”를 특징으로 쓴다면 고양이와 개를 구분하는 데 전혀 도움이 되지 않아, 머신러닝 모델의 성능도 향상되지 않음

기존 머신러닝 방법에서는 Feature Extraction을 사람이 수행

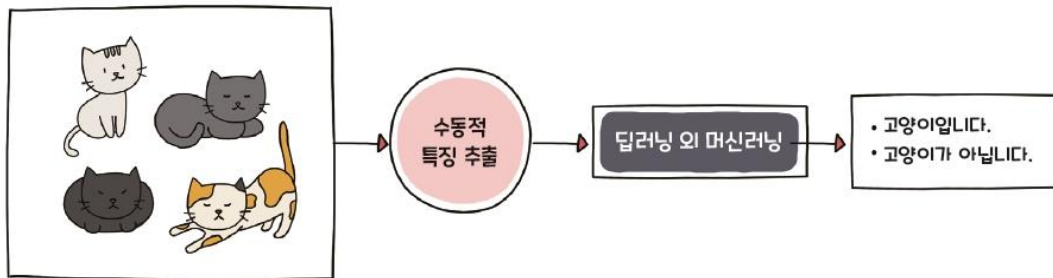


“좋은 Features를 뽑을 수 있는가?”는 결국 사람의 능력에 좌우됨

# 01. 딥러닝에 대해 알아봅시다

## • 딥러닝의 특징 (2/3)

- 딥러닝은 사람의 도움 없이 학습 데이터로부터 특징 추출 작업을 수행함
- 고양이와 개를 구분하는 예에서, 딥러닝은 고양이와 개 이미지로부터 그 둘을 구별하는 데 유용한 특징들을 스스로 찾아냄



(a) 딥러닝 외 머신러닝 : 사람이 수동적으로 특징 추출 작업 수행



(b) 딥러닝 : 학습 데이터로부터 스스로 특징 추출 작업 수행

그림 3-11 딥러닝의 특징 1 : 학습 데이터로부터 자동으로 특징 추출 작업 수행 © Hanbit Academy Inc.

[주의]

딥러닝에서도 사람이 추출한 Features를 입력 받아 모델을 학습시키는 방법을 사용할 수 있음

# 01. 딥러닝에 대해 알아봅시다

인공지능 활용 Python language

## • 딥러닝의 특징 (3/3)

- 딥러닝 외 대부분의 머신러닝 알고리즘들은 데이터의 양이 일정 수준을 넘어가면 더 이상 성능이 향상되지 않는 한계를 보이지만, 딥러닝은 데이터가 많을수록 성능이 좋아지는 것이 특징
- 따라서 “얼마나 더 많은 양의 데이터를 확보하느냐”에 따라 인공지능 (딥러닝) 역량에 차이가 커짐

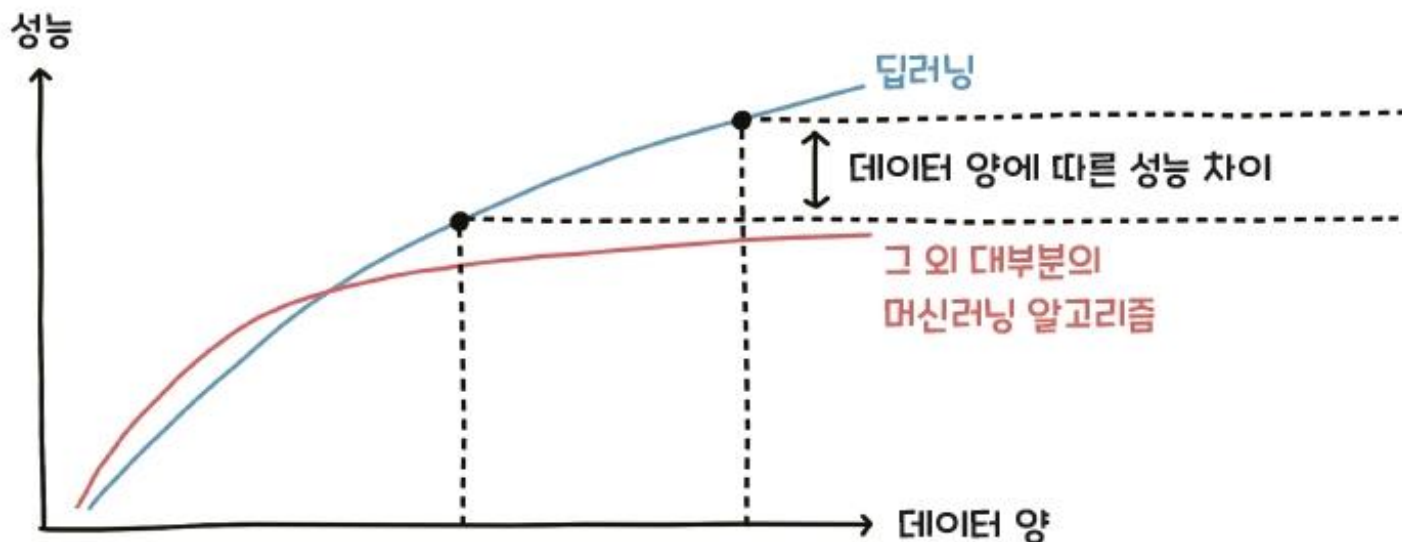


그림 3-12 딥러닝의 특징 2 : 데이터의 양이 많아질수록 성능이 지속적으로 향상

© Hanbit Academy Inc.