



# 인공신경망과딥러닝심화

## Lecture 08. 다층 퍼셉트론

동덕여자대학교  
데이터사이언스 전공  
권 범

# 목차

- ❖ 01. 다층 퍼셉트론의 등장
- ❖ 02. 다층 퍼셉트론의 설계
- ❖ 03. XOR 문제의 해결
- ❖ 04. 코딩으로 XOR 문제 해결하기

# 01. 다층 퍼셉트론의 등장

02. 다층 퍼셉트론의 설계

03. XOR 문제의 해결

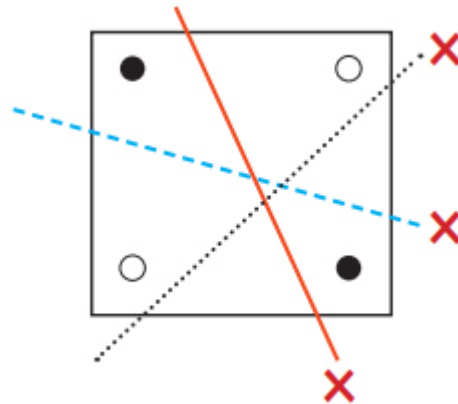
04. 코딩으로 XOR 문제 해결하기

# 01. 다층 퍼셉트론의 등장

## ❖ 다층 퍼셉트론의 등장 과정에 대해 알아보기 (1/6)

- 앞서 종이 위에 각각 엇갈려 놓인 검은색 점 두 개와 흰색 점 두 개를 하나의 선으로는 구별할 수 없다는 것을 살펴보았음
- 언뜻 보기에 해답이 없어 보이는 이 문제를 해결하려면 새로운 접근이 필요함

선으로는 같은 색끼리 나눌 수 없다: 퍼셉트론의 한계

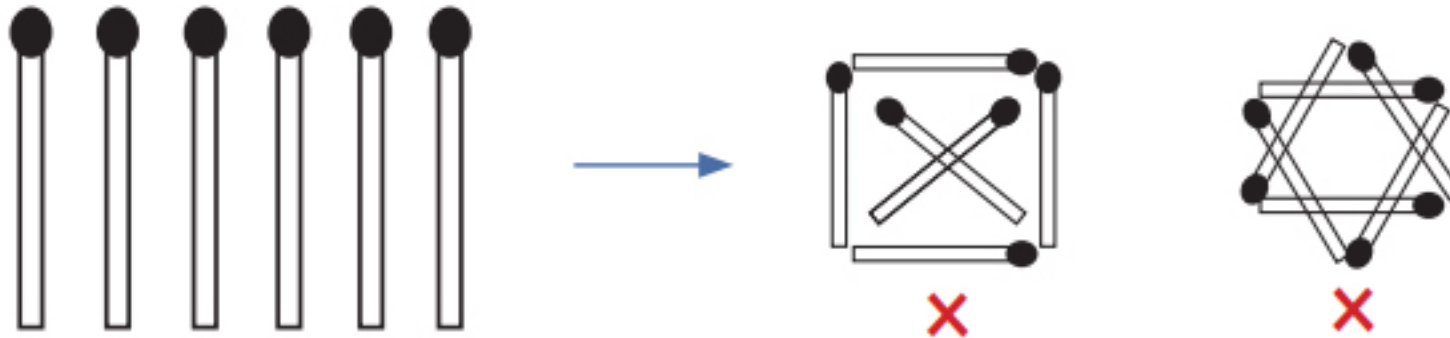


# 01. 다층 퍼셉트론의 등장

## ❖ 다층 퍼셉트론의 등장 과정에 대해 알아보기 (2/6)

- 어릴 적 친구들에게 장난처럼 들곤 했던 문제가 의외로 기발한 해답이었던 기억이 있음
- '성냥개비 여섯 개로 정삼각형 네 개를 만들 수 있는가'라는 문제를 기억하나요?

성냥개비 여섯 개로 정삼각형 네 개를?

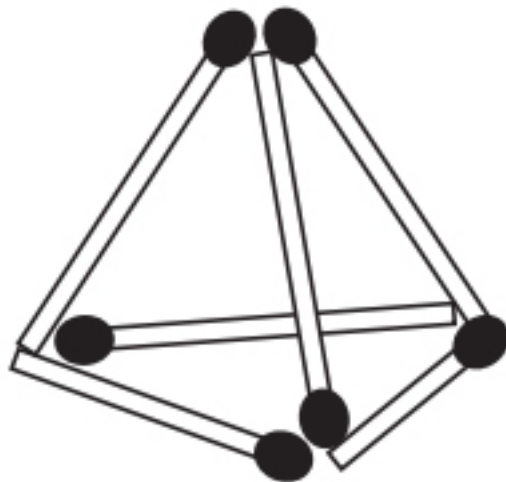


# 01. 다층 퍼셉트론의 등장

## ❖ 다층 퍼셉트론의 등장 과정에 대해 알아보기 (3/6)

- 골똥히 연구해도 답을 찾지 못했던 이 문제는  
2차원 평면에서만 해결하려는 고정 관념을 깨고  
피라미드 모양으로 성냥개비를 쌓아 올리니 해결

차원을 달리하니 쉽게 해결!



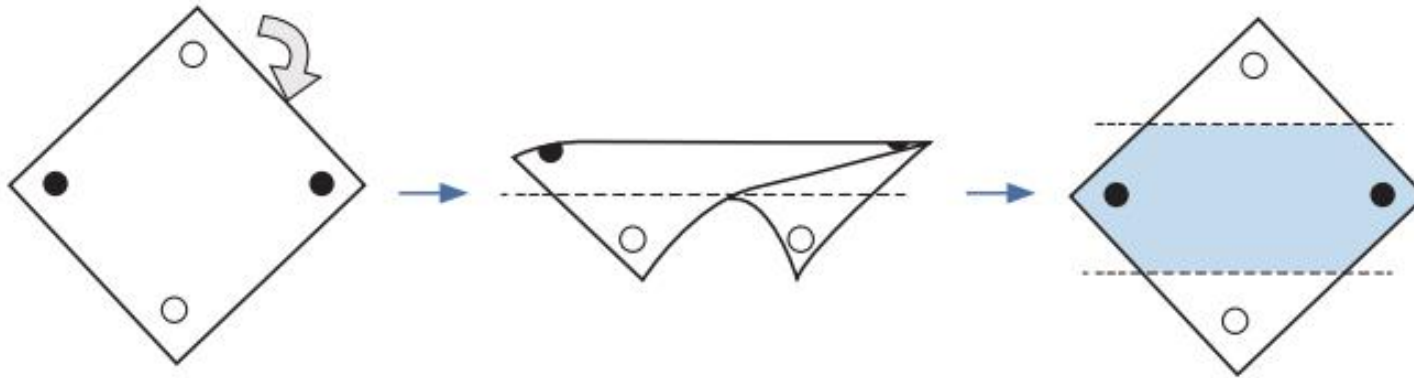
# 01. 다층 퍼셉트론의 등장

## ❖ 다층 퍼셉트론의 등장 과정에 대해 알아보기 (4/6)

- 인공지능 학자들은 인공신경망을 개발하기 위해 반드시 XOR 문제를 극복해야만 했음
- 이 문제를 해결하는 데도 고정 관념을 깨는 기발한 아이디어가 필요했음

이러한 노력은 결국 아래 그림과 같은 아이디어를 낳았음

**XOR 문제의 해결은 평면을 휘어 주는 것!**



- ✓ 즉, 종이를 휘어 주어 선 두 개를 동시에 긋는 방법
- ✓ 이것을 XOR 문제에 적용하면 '퍼셉트론 두 개를 한 번에 계산'하면 된다는 결론에 이름
- ✓ 이를 위해 퍼셉트론 두 개를 각각 처리하는 **은닉층(Hidden Layer)**을 만듦

# 01. 다층 퍼셉트론의 등장

## ❖ 다층 퍼셉트론의 등장 과정에 대해 알아보기 (5/6)

- 은닉층을 만드는 것이 어떻게 XOR 문제를 해결하는지 살펴보자

XOR (배타적 논리합) 하나만 1이어야 1			② NAND (부정 논리곱) 하나라도 0이면 1			③ OR (논리합) 두 개 중 한 개라도 1이면 1			④ AND (논리곱) 두 개 모두 1일 때 1		
$x_1$	$x_2$	결괏값	$x_1$	$x_2$	결괏값 1	$x_1$	$x_2$	결괏값 2	결괏값 1	결괏값 2	결괏값
0	0	0	0	0	1	0	0	0	1	0	0
0	1	1	0	1	1	0	1	1	1	1	1
1	0	1	1	0	1	1	0	1	1	1	1
1	1	0	1	1	0	1	1	1	0	1	0

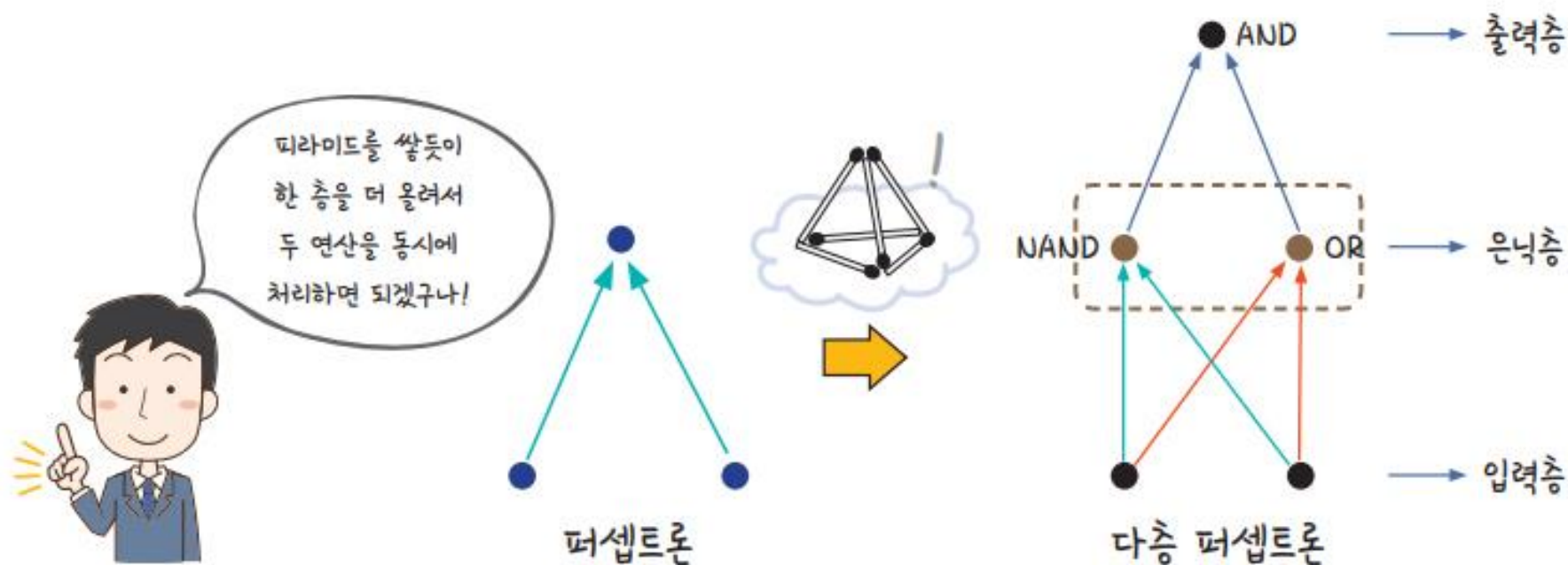
- ✓ 먼저 ①  $x_1$ 과  $x_2$ 를 두 연산으로 각각 보냄
- ✓ ② 첫 번째 연산에서는 NAND 연산을, ③ 두 번째 연산에서 OR 연산을 함
- ✓ ②와 ③을 통해 구한 결괏값에 대해 ④ AND 연산을 하면 구하고자 하는 출력 값을 만들 수 있음



# 01. 다층 퍼셉트론의 등장

## ❖ 다층 퍼셉트론의 등장 과정에 대해 알아보기 (6/6)

### 은닉층이 XOR 문제를 해결하는 원리



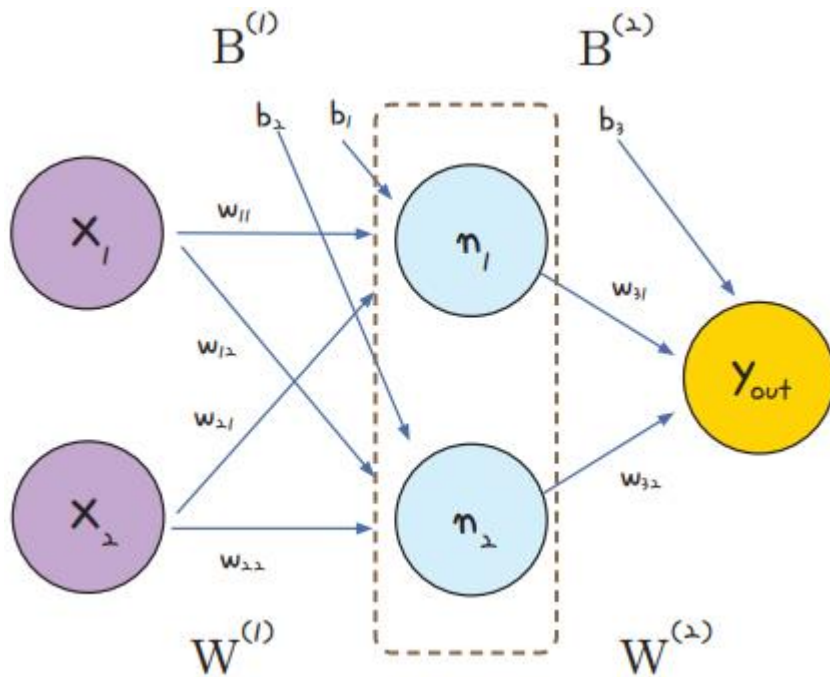
## 02. 다층 퍼셉트론의 설계

- 01. 다층 퍼셉트론의 등장
- 03. XOR 문제의 해결
- 04. 코딩으로 XOR 문제 해결하기

## 02. 다층 퍼셉트론의 설계

### ❖ 다층 퍼셉트론 설계하기 (1/4)

- 다층 퍼셉트론이 입력층과 출력층 사이에 숨어 있는 은닉층을 만드는 것을 그림으로 나타내면 아래 그림과 같음



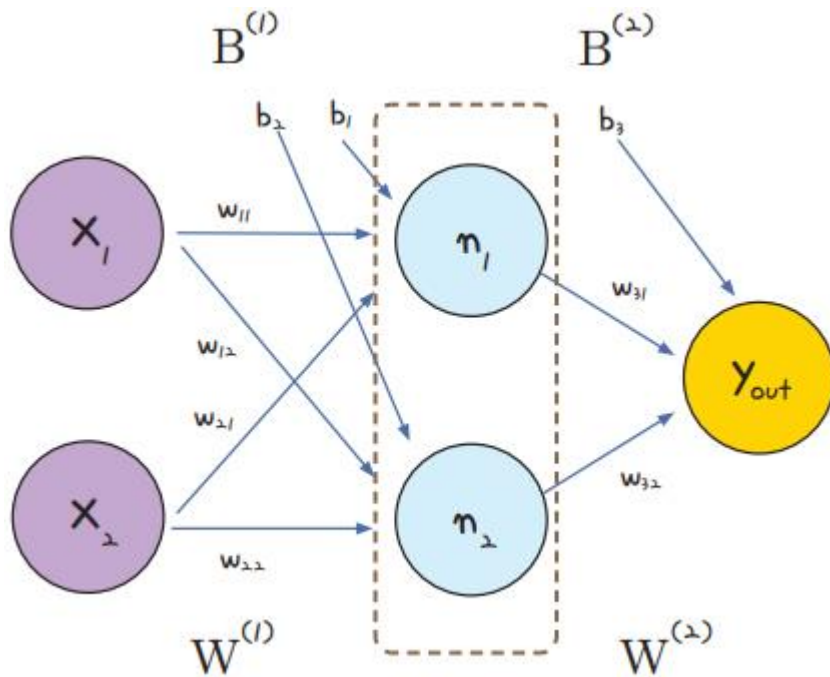
### 다층 퍼셉트론의 내부

- ✓ 가운데 점선으로 표시된 부분이 은닉층
- ✓  $x_1$ 과  $x_2$ 는 입력 값인데, 각 값에 가중치( $w$ )를 곱하고 바이어스( $b$ )를 더해 은닉층으로 전송
- ✓ 이 값들이 모이는 은닉층의 중간 정거장을 노드(Node)라고 하며, 여기서는  $n_1$ 과  $n_2$ 로 표시

## 02. 다층 퍼셉트론의 설계

### ❖ 다층 퍼셉트론 설계하기 (2/4)

- 은닉층에 취합된 값들은 활성화 함수를 통해 다음으로 보내는데, 만약 앞서 배운 시그모이드 함수  $\sigma(\cdot)$ 를 활성화 함수로 사용한다면  $n_1$ 과  $n_2$ 에서 계산되는 값은 각각 다음과 같음



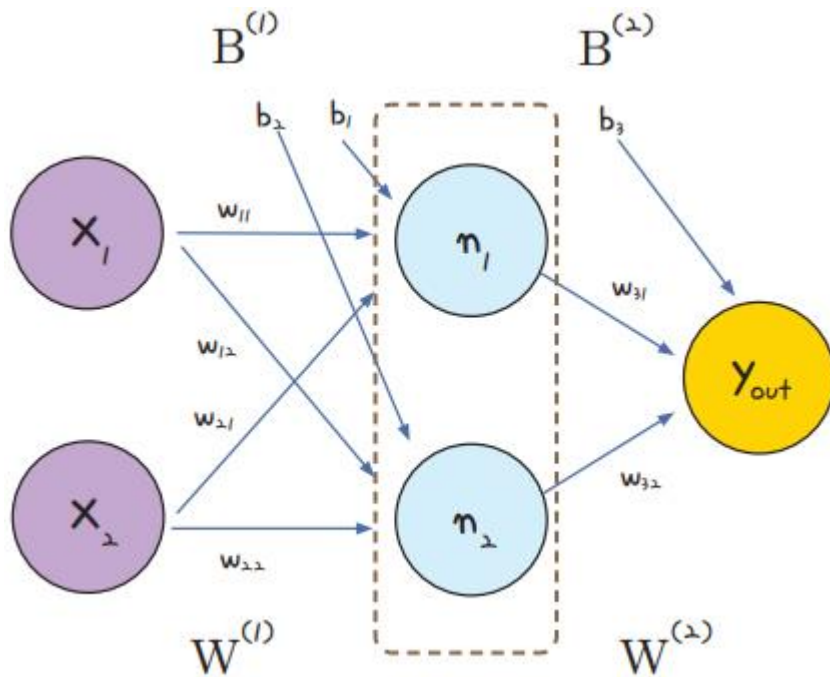
$$n_1 = \sigma(x_1 w_{11} + x_2 w_{21} + b_1)$$

$$n_2 = \sigma(x_1 w_{12} + x_2 w_{22} + b_2)$$

## 02. 다층 퍼셉트론의 설계

### ❖ 다층 퍼셉트론 설계하기 (3/4)

- 두 식의 결과값이 출력층의 방향으로 보내어지고,  
출력층으로 전달된 값은 마찬가지로 활성화 함수를 사용해  $y$  예측 값을 정하게 됨
- 이 값을  $y_{out}$  이라고 할 때, 이를 식으로 표현하면 다음과 같음

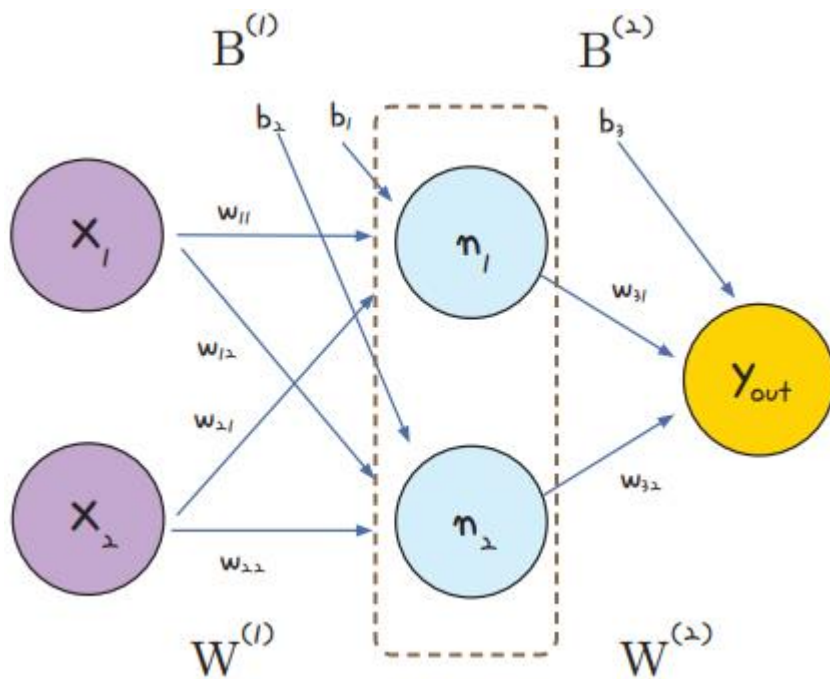


$$y_{out} = \sigma(n_1 w_{31} + n_2 w_{32} + b_3)$$

## 02. 다층 퍼셉트론의 설계

### ❖ 다층 퍼셉트론 설계하기 (4/4)

- 이제 각각의 가중치( $w$ )와 바이어스( $b$ ) 값을 정할 차례
- 2차원 배열로 늘어놓으면 다음과 같이 표시할 수 있음
- 은닉층을 포함해 가중치 여섯 개와 바이어스 세 개가 필요함



$$W^{(1)} = \begin{bmatrix} w_{11} & w_{12} \\ w_{21} & w_{22} \end{bmatrix}$$

$$B^{(1)} = \begin{bmatrix} b_1 \\ b_2 \end{bmatrix}$$

$$W^{(2)} = \begin{bmatrix} w_{31} \\ w_{32} \end{bmatrix}$$

$$B^{(2)} = [b_3]$$

## 03. XOR 문제의 해결

- 01. 다층 퍼셉트론의 등장
- 02. 다층 퍼셉트론의 설계
- 04. 코딩으로 XOR 문제 해결하기

## 03. XOR 문제의 해결

### ❖ XOR 문제를 해결하는 과정 살펴보기 (1/3)

- 앞서 우리에게 어떤 가중치와 바이어스가 필요한지 알아보았음  
이를 만족하는 가중치와 바이어스의 조합은 무수히 많음
- 지금은 먼저 아래와 같이 각 변수 값을 정하고,  
이를 이용해 XOR 문제를 해결하는 과정을 알아보자

$$W^{(1)} = \begin{bmatrix} -2 & 2 \\ -2 & 2 \end{bmatrix} \quad B^{(1)} = \begin{bmatrix} 3 \\ -1 \end{bmatrix}$$

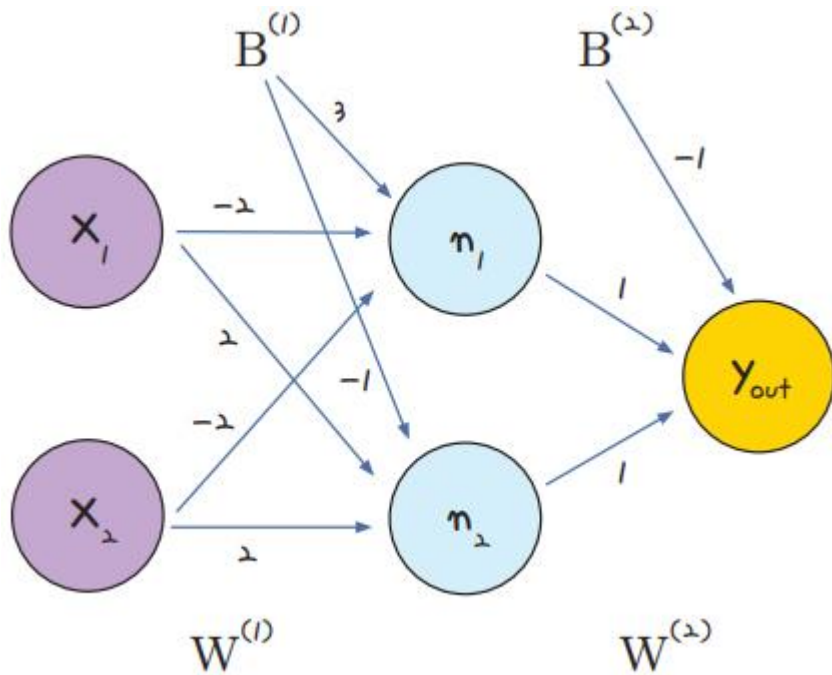
$$W^{(2)} = \begin{bmatrix} 1 \\ 1 \end{bmatrix} \quad B^{(2)} = [-1]$$



### 03. XOR 문제의 해결

#### ❖ XOR 문제를 해결하는 과정 살펴보기 (2/3)

- 각 변수 값을 다층 퍼셉트론에 대입하면 아래 그림과 같음



다층 퍼셉트론의 내부에 변수 채우기

$$W^{(1)} = \begin{bmatrix} -2 & 2 \\ -2 & 2 \end{bmatrix}$$

$$B^{(1)} = \begin{bmatrix} 3 \\ -1 \end{bmatrix}$$

$$W^{(2)} = \begin{bmatrix} 1 \\ 1 \end{bmatrix}$$

$$B^{(2)} = [-1]$$

# 03. XOR 문제의 해결

## ❖ XOR 문제를 해결하는 과정 살펴보기 (3/3)

- 이제  $x_1$  값과  $x_2$  값을 각각 입력해서, 원하는  $y$  값이 나오는지 점검해 보자

### XOR 다층 문제 해결

$\approx$  기호는 '거의 같다'를 의미

$x_1$	$x_2$	$n_1$	$n_2$	$y_{out}$	우리가 원하는 값
0	0	$\sigma(0 * (-2) + 0 * (-2) + 3) \approx 1$	$\sigma(0 * 2 + 0 * 2 - 1) \approx 0$	$\sigma(1 * 1 + 0 * 1 - 1) \approx 0$	0
0	1	$\sigma(0 * (-2) + 1 * (-2) + 3) \approx 1$	$\sigma(0 * 2 + 1 * 2 - 1) \approx 1$	$\sigma(1 * 1 + 1 * 1 - 1) \approx 1$	1
1	0	$\sigma(1 * (-2) + 0 * (-2) + 3) \approx 1$	$\sigma(1 * 2 + 0 * 2 - 1) \approx 1$	$\sigma(1 * 1 + 1 * 1 - 1) \approx 1$	1
1	1	$\sigma(1 * (-2) + 1 * (-2) + 3) \approx 0$	$\sigma(1 * 2 + 1 * 2 - 1) \approx 1$	$\sigma(0 * 1 + 1 * 1 - 1) \approx 0$	0

- ✓ 위의 표에서 볼 수 있듯이 원하는 결과를 구할 수 있었음
- ✓ 숨어 있는 노드 두 개를 둔 다층 퍼셉트론을 통해 XOR 문제가 해결된 것

## 04. 코딩으로 XOR 문제 해결하기

- 01. 다층 퍼셉트론의 등장
- 02. 다층 퍼셉트론의 설계
- 03. XOR 문제의 해결

## 04. 코딩으로 XOR 문제 해결하기

### ❖ 파이썬으로 XOR 문제 해결하기 (1/10)

- 이제 주어진 가중치와 바이어스를 이용해 XOR 문제를 해결하는 파이썬 코드를 작성해 보자
- 정해진 가중치와 바이어스를 넘파이 라이브러리를 사용해 아래와 같이 선언하겠음

```
import numpy as np

w11 = np.array([-2, -2])
w12 = np.array([2, 2])
w2 = np.array([1, 1])
b1 = 3
b2 = -1
b3 = -1
```

## 04. 코딩으로 XOR 문제 해결하기

### ❖ 파이썬으로 XOR 문제 해결하기 (2/10)

- 이제 퍼셉트론 함수를 만들어 줌
- 0과 1 중에서 값을 출력하게 설정

```
def MLP(x, w, b):  
    y = np.sum(w * x) + b  
    if y <= 0:  
        return 0  
    else:  
        return 1
```

## 04. 코딩으로 XOR 문제 해결하기

### ❖ 파이썬으로 XOR 문제 해결하기 (3/10)

- 각 게이트의 정의에 따라 NAND, OR, AND, XOR 게이트 함수를 만들어 줌

```
# NAND 게이트
def NAND(x1, x2):
    return MLP(np.array([x1, x2]), w11, b1)

# OR 게이트
def OR(x1, x2):
    return MLP(np.array([x1, x2]), w12, b2)

# AND 게이트
def AND(x1, x2):
    return MLP(np.array([x1, x2]), w2, b3)

# XOR 게이트
def XOR(x1, x2):
    return AND(NAND(x1, x2), OR(x1, x2))
```

## 04. 코딩으로 XOR 문제 해결하기

### ❖ 파이썬으로 XOR 문제 해결하기 (4/10)

- 이제  $x_1$  값과  $x_2$  값을 번갈아 대입해 가며 최종 값을 출력해 보자

```
for x in [(0, 0), (0, 1), (1, 0), (1, 1)]:  
    y = XOR(x[0], x[1])  
    print("입력 값: " + str(x) + " 출력 값: " + str(y))
```

## 04. 코딩으로 XOR 문제 해결하기

### ❖ 파이썬으로 XOR 문제 해결하기 (5/10)

- 모두 정리하면 다음과 같음

```
1 import numpy as np
2
3 # 가중치와 바이어스
4 w11 = np.array([-2, -2])
5 w12 = np.array([2, 2])
6 w2 = np.array([1, 1])
7 b1 = 3
8 b2 = -1
9 b3 = -1
10
11 # 퍼셉트론
12 def MLP(x, w, b):
13     y = np.sum(w * x) + b
14     if y <= 0:
15         return 0
16     else:
17         return 1
```



## 04. 코딩으로 XOR 문제 해결하기

### ❖ 파이썬으로 XOR 문제 해결하기 (6/10)

```
18 # NAND 게이트
19 def NAND(x1, x2):
20     return MLP(np.array([x1, x2]), w11, b1)
21
22 # OR 게이트
23 def OR(x1, x2):
24     return MLP(np.array([x1, x2]), w12, b2)
25
26 # AND 게이트
27 def AND(x1, x2):
28     return MLP(np.array([x1, x2]), w2, b3)
29
30 # XOR 게이트
31 def XOR(x1, x2):
32     return AND(NAND(x1, x2), OR(x1, x2))
33
34
35
```

## 04. 코딩으로 XOR 문제 해결하기

### ❖ 파이썬으로 XOR 문제 해결하기 (7/10)

```
36 # x1 값, x2 값을 번갈아 대입하며 최종 값 출력
37 for x in [(0, 0), (0, 1), (1, 0), (1, 1)]:
38     y = XOR(x[0], x[1])
39     print("입력 값: " + str(x) + " 출력 값: " + str(y))
```

## 04. 코딩으로 XOR 문제 해결하기

### ❖ 파이썬으로 XOR 문제 해결하기 (8/10)

#### 실행결과

입력 값:	(0, 0)	출력 값:	0
입력 값:	(0, 1)	출력 값:	1
입력 값:	(1, 0)	출력 값:	1
입력 값:	(1, 1)	출력 값:	0

XOR 문제의 정답 도출

## 04. 코딩으로 XOR 문제 해결하기

### ❖ 파이썬으로 XOR 문제 해결하기 (9/10)

- 이렇게 퍼셉트론 하나로 해결되지 않던 문제를 은닉층을 만들어 해결
- 퍼셉트론의 문제가 완전히 해결된 것은 아니었음
- 다층 퍼셉트론을 사용할 경우 XOR 문제는 해결되었지만,  
은닉층에 들어 있는 **가중치를 데이터를 통해 학습하는 방법이 아직 없었기 때문임**
- 다층 퍼셉트론의 적절한 학습 방법을 찾기까지 그 후로 약 20여 년의 시간이 더 필요했음  
이 기간을 흔히 **인공지능의 겨울**이라고 함

## 04. 코딩으로 XOR 문제 해결하기

### ❖ 파이썬으로 XOR 문제 해결하기 (10/10)

- 이 겨울을 지나며 데이터 과학자들은 두 부류로 나뉘
- 하나는 최적화된 예측선을 잘 그려 주던 아달라인을 발전시켜 SVM이나 로지스틱 회귀 모델을 만든 그룹
- 또 하나의 그룹은 여러 어려움 속에서도 끝까지 다층 퍼셉트론의 학습 방법을 찾던 그룹
- 이 두 번째 그룹에 속해 있던 **제프리 힌튼(Geoffrey Hinton)** 교수가 바로 딥러닝의 아버지로 칭송 받는 사람
- 힌튼 교수는 여러 가지 혁신적인 아이디어로 인공지능의 겨울을 극복해 냈음  
첫 번째 아이디어는 1986년에 발표한 **오차 역전파**

# 04. 코딩으로 XOR 문제 해결하기

## ❖ 한눈에 보는 인공지능의 역사: 퍼셉트론에서 딥러닝까지

**1943**  
맥컬릭 - 월터피츠



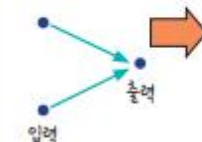
"온·오프 기능이 있는 신경을 그물망 형태로 연결하면 사람의 뇌처럼 동작할 수 있다!"




**1957**  
로젠블랫



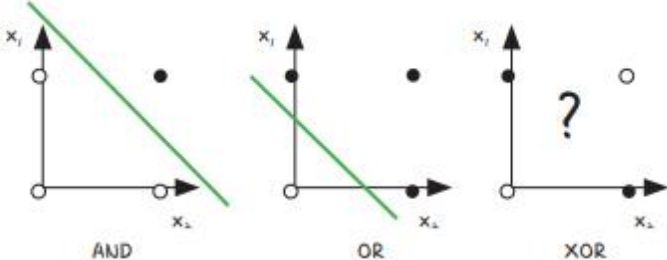
여기에 학습을 더하면?  
→ 퍼셉트론



**1969**  
민스키 - 페퍼트



XOR 문제?

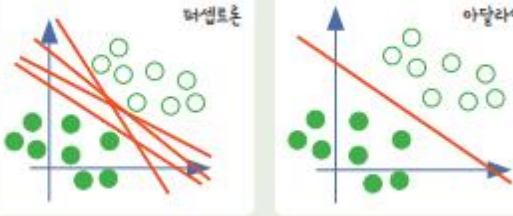


잠깐만요



**1960**  
위드로우


여기에 경사 하강법을 없으면?  
→ 아달라인



끝까지 해 보자  
(인공지능의 겨울)

**1986** **1988**

럼멜하트 힌튼 레쿰 벤지오



→ 오차 역전파(1986)  
→ 새로운 활성화 함수(2000)  
→ 초기 가중치 설정(2006)  
→ GPU 등 하드웨어의 발전  
→ 딥러닝!

쓸 만한 걸 발전시키자

→ 서포트 벡터 머신  
→ 로지스틱 회귀

- ❖ 01. 다층 퍼셉트론의 등장
- ❖ 02. 다층 퍼셉트론의 설계
- ❖ 03. XOR 문제의 해결
- ❖ 04. 코딩으로 XOR 문제 해결하기

# THANK YOU!

## Q & A

- Name: 권범
- Office: 동덕여자대학교 인문관 B821호
- Phone: 02-940-4752
- E-mail: [bkwon@dongduk.ac.kr](mailto:bkwon@dongduk.ac.kr)