ROKEY BOOT CAMP

AI(Computer Vision)개론

Chapter 7. 퍼셉트론과 인공지능의 시작

OOO 강사

학습 목차

01

인공지능의 시작을 알린 퍼셉트론

인공지능의 시작을 알린 퍼셉트론

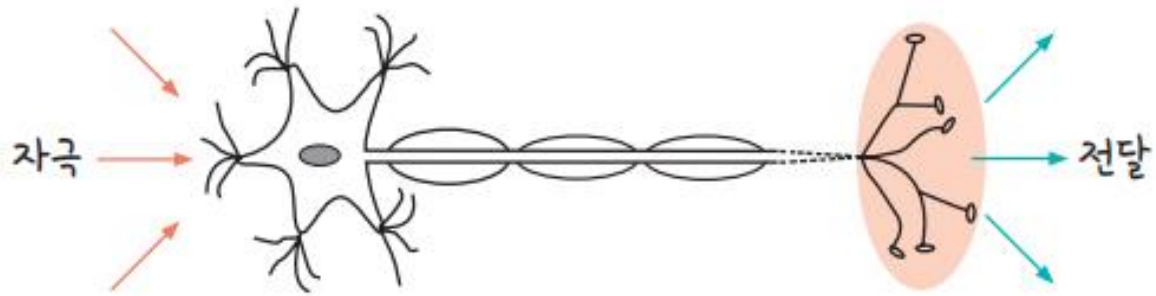
☑ 인공지능의 시작을 알린 퍼셉트론

- 인간의 뇌는 치밀하게 연결된 뉴런 약 1,000억 개로 이루어져 있음
- 뉴런과 뉴런 사이에는 시냅스라는 연결 부위가 있는데, 신경 말단에서 자극을 받으면 시냅스에서 화학 물질이 나와
  전위 변화를 일으킴
- 전위가 임계 값을 넘으면 다음 뉴런으로 신호를 전달하고, 임계 값에 미치지 못하면 아무것도 하지 않음
- 이 메커니즘은 우리가 앞서 배운 로지스틱 회귀와 많이 닮았음
- 이 간단한 회로는 입력 값을 놓고 활성화 함수에 의해 일정한 수준을 넘으면 참을, 그렇지 않으면 거짓을 내보내는 일을 하는데 뉴런과 유사함

인공지능의 시작을 알린 퍼셉트론

☑ 인공지능의 시작을 알린 퍼셉트론

▼ 그림 7-1 | 뉴런의 신호 전달



☑ 인공지능의 시작을 알린 퍼셉트론

- 우리 몸 안에 있는 수많은 뉴런은 서로 긴밀히 연결되어 신경 말단부터 뇌까지 곳곳에서 자신의 역할을 수행
- 이처럼 복잡하고 어려운 조합의 결과가 바로 우리의 '생각'
- 뉴런과 비슷한 메커니즘을 사용하면 인공적으로 '생각'하는 그 무언가를 만들 수 있지 않을까?

☑ 인공지능의 시작을 알린 퍼셉트론

- 이러한 상상과 함께 출발한 연구가 바로 인공 신경망(artificial neural network, 이하 줄여서 '신경망'이라고 함) 연구
- 맨 처음 시작은 '켜고 끄는 기능이 있는 신경'을 그물망 형태로 연결하면 사람의 뇌처럼 동작할 수 있다는 가능성을 처음으로 주장한 맥컬럭-월터 피츠(McCulloch-Walter Pitts)의 1943년 논문
- 그 후 1957년, 미국의 신경 생물학자 프랑크 로젠블랫(Frank Rosenblatt)이 이 개념을 실제 장치로 만들어 선보임
- 이것의 이름이 퍼셉트론(perceptron)

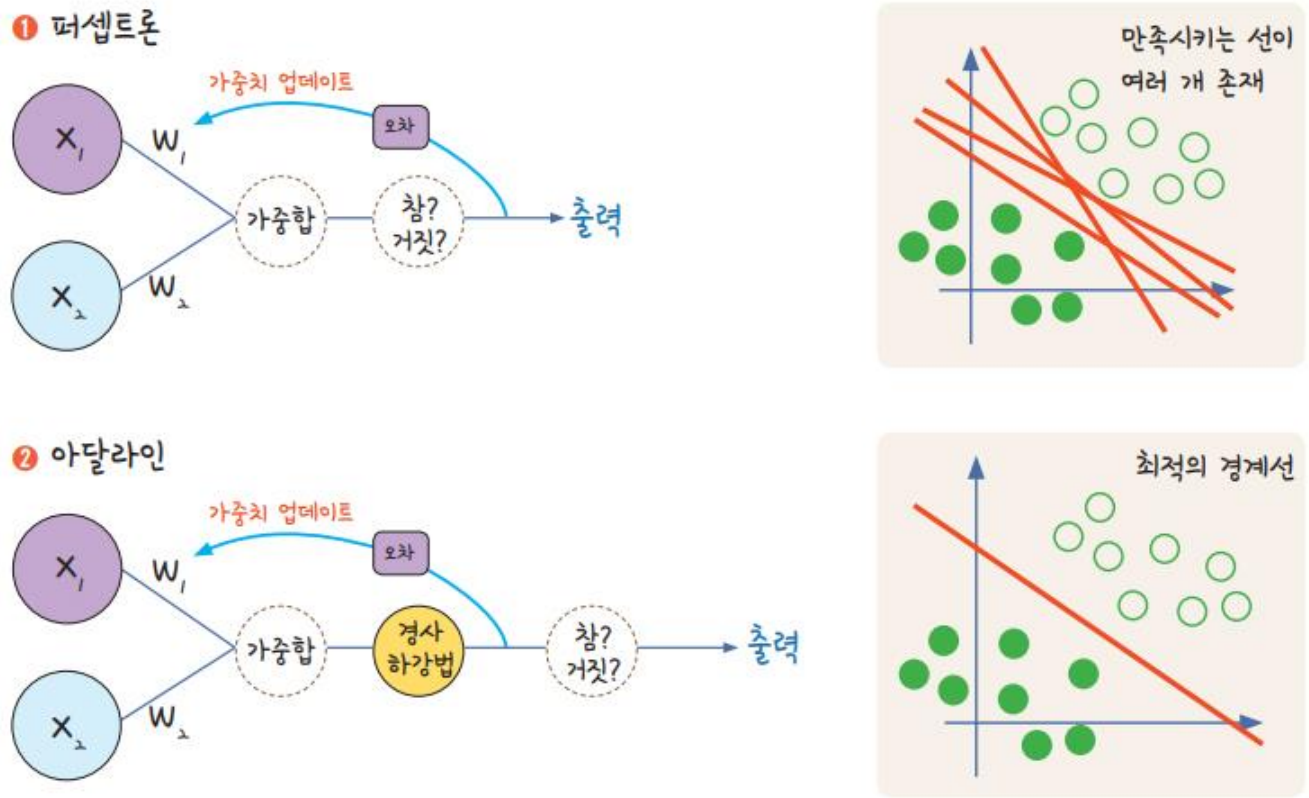
인공지능의 시작을 알린 퍼셉트론

- 퍼셉트론은 입력 값을 여러 개 받아 출력을 만드는데, 이때 입력 값에 가중치를 조절할 수 있게 만들어 최초로 '학습'을 하게 했음(그림 7-2 참조)
- 3년 후, 여기에 앞 장에서 다룬 경사 하강법을 도입해 최적의 경계선을 그릴 수 있게 한 아달라인(Adaline)이 개발
  (그림 7-2 참조)
- 특히 아달라인은 이후 서포트 벡터 머신(Support Vector Machine) 등 머신 러닝의 중요한 알고리즘들로 발전해 가는데, 이 중 시그모이드를 활성화 함수로 사용한 것이 바로 앞서 배웠던 로지스틱 회귀

인공지능의 시작을 알린 퍼셉트론

☑ 인공지능의 시작을 알린 퍼셉트론

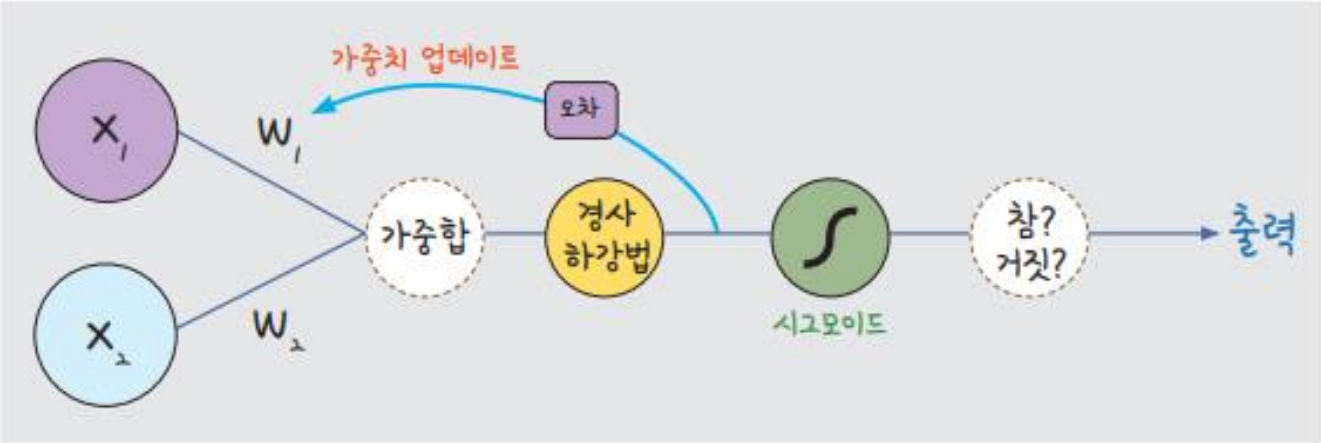
▼ 그림 7-2 | 퍼셉트론, 아달라인 그리고 로지스틱 회귀 모델



인공지능의 시작을 알린 퍼셉트론

☑ 인공지능의 시작을 알린 퍼셉트론

(비교) 로지스틱 회귀 모델



☑ 인공지능의 시작을 알린 퍼셉트론

- 가중합(weighted sum)이란 입력 값과 가중치를 모두 곱한 후 바이어스를 더한 값을 의미

02

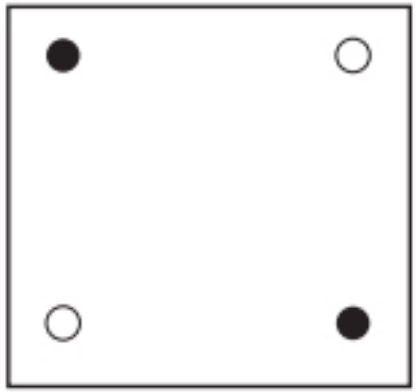
퍼셉트론의 과제

☑ 퍼셉트론의 과제

- 퍼셉트론이 완성되고 아달라인에 의해 보완되며 드디어 현실 세계의 다양한 문제를 해결하는 인공지능이 개발될 것으로 기대했음
- 곧 퍼셉트론의 한계가 보고
- 퍼셉트론의 한계가 무엇이었는지 알고 이를 극복하는 과정을 이해하는 것은 우리에게도 매우 중요함
- 이것을 해결한 것이 바로 딥러닝이기 때문임
- 지금부터는 퍼셉트론의 한계와 이를 해결하는 과정을 보며 신경망의 기본 개념을 확립해 보자

☑ 퍼셉트론의 과제

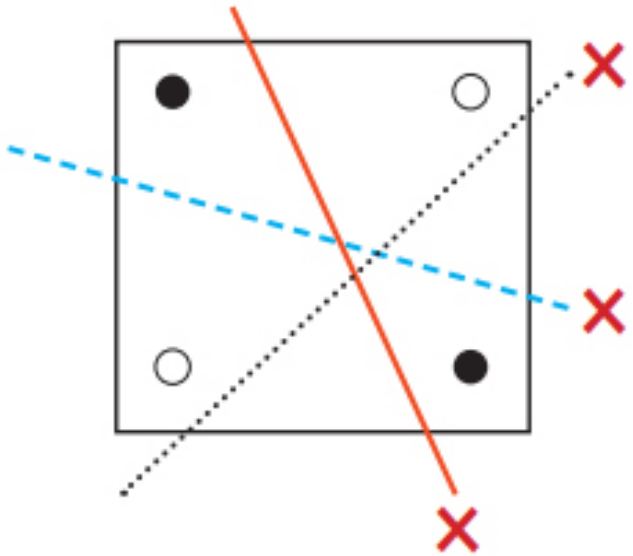
▼ 그림 7-3 | 사각형 종이에 놓인 검은색 점 두 개와 흰색 점 두 개



퍼셉트론의 과제

- 사각형 종이에 검은색 점 두 개와 흰색 점 두 개가 놓여 있음
- 이 네 점 사이에 직선을 하나 긋는다고 하자
- 이때 직선의 한쪽 편에는 검은색 점만 있고, 다른 한쪽에는 흰색 점만 있게끔 선을 그을 수 있을까?

▼ 그림 7-4 | 선으로는 같은 색끼리 나눌 수 없다: 퍼셉트론의 한계



☑ 퍼셉트론의 과제

- 선을 여러 개 아무리 그어 보아도 하나의 직선으로는 흰색 점과 검은색 점을 구분할 수 없음
- 퍼셉트론이나 아달라인은 모두 2차원 평면상에 직선을 긋는 것만 가능함
- 이 예시는 경우에 따라 선을 아무리 그어도 해결되지 않는 상황이 있다는 것을 말해 줌

03

XOR 문제

XOR 문제

- 이것이 퍼셉트론의 한계를 설명할 때 등장하는 XOR(exclusive OR) 문제
- XOR 문제는 논리 회로에 등장하는 개념
- 컴퓨터는 두 가지의 디지털 값, 즉 0과 1을 입력해 하나의 값을 출력하는 회로가 모여 만들어지는데, 이 회로를 '게이트(gate)'라고 함
- 그림 7-5는 AND 게이트, OR 게이트, XOR 게이트에 대한 값을 정리한 것
- AND게이트는 x1과 x2 둘 다 1일 때 결괏값이 1로 출력
- OR 게이트는 둘 중 하나라도 1이면 결괏값이 1로 출력
- XOR 게이트는 둘 중 하나만 1일 때 1이 출력

XOR 문제

☑ XOR 문제

▼ 그림 7-5 | AND, OR, XOR 게이트에 대한 진리표

AND
(논리곱)
두 개 모두 1일 때 1

x_1	x_2	결괏값
0	0	0
0	1	0
1	0	0
1	1	1

OR
(논리합)
두 개 중 한 개라도 1이면 1

x_1	x_2	결괏값
0	0	0
0	1	1
1	0	1
1	1	1

XOR
(배타적 논리합)
하나만 1이어야 1

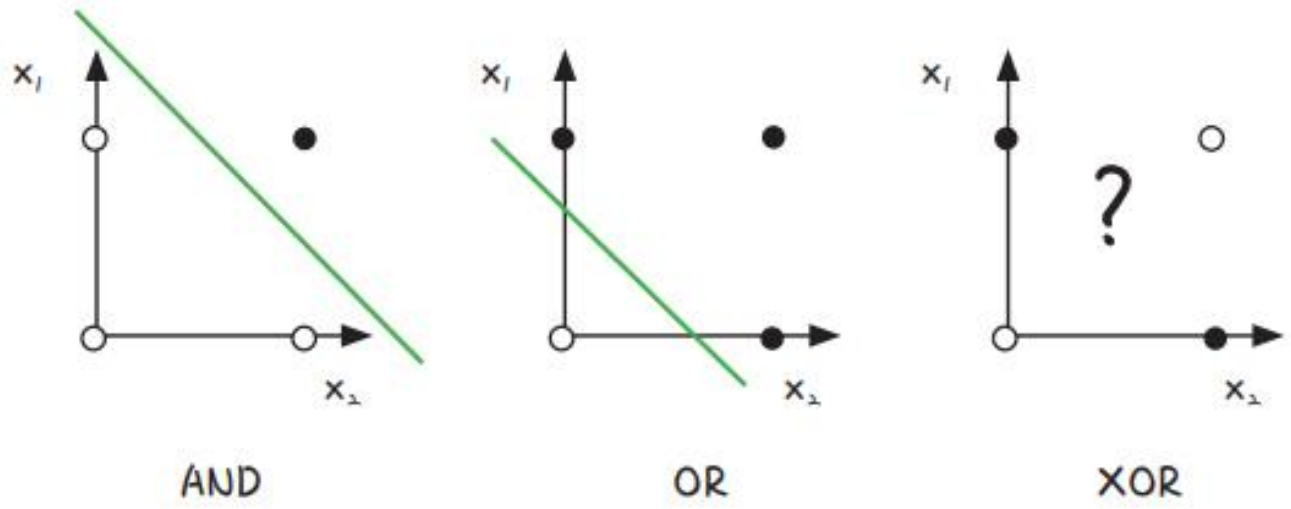
x_1	x_2	결괏값
0	0	0
0	1	1
1	0	1
1	1	0

XOR 문제

XOR 문제

- 그림 7-5를 각각 그래프로 좌표 평면에 나타내 보자
- 결괏값이 0이면 흰색 점으로, 1이면 검은색 점으로 나타낸 후 조금 전처럼 직선을 그어 위 조건을 만족할 수 있는지 보자

▼ 그림 7-6 | AND, OR, XOR 진리표대로 좌표 평면에 표현한 후 선을 그어 색이 같은 점끼리 나누기(XOR는 불가능)



XOR 문제

- AND와 OR 게이트는 직선을 그어 결괏값이 1인 값(검은색 점)을 구별할 수 있음
- XOR의 경우 선을 그어 구분할 수 없음
- 이는 인공지능 분야의 선구자였던 MIT의 마빈 민스키(Marvin Minsky) 교수가 1969년에 발표한 "퍼셉트론즈(Perceptrons)"라는 논문에 나오는 내용
- '뉴런 → 신경망 → 지능'이라는 도식에 따라 '퍼셉트론 → 인공 신경망 → 인공지능'이 가능하리라 꿈꾸었던 당시 사람들은 이것이 생각처럼 쉽지 않다는 사실을 깨닫게 됨
- 알고 보니 간단한 XOR 문제조차 해결할 수 없었던 것
- 이 논문 이후 인공지능 연구가 한동안 침체기를 겪게 됨
- 이 문제는 두 가지 방법이 순차적으로 개발되면서 해결
- 하나는 다층 퍼셉트론(multilayer perceptron), 그리고 또 하나는 오차 역전파(back propagation)

ROKEY BOOT CAMP

AI(Computer Vision)개론

Chapter 8. 다층 퍼셉트론

OOO 강사

학습 목차

01

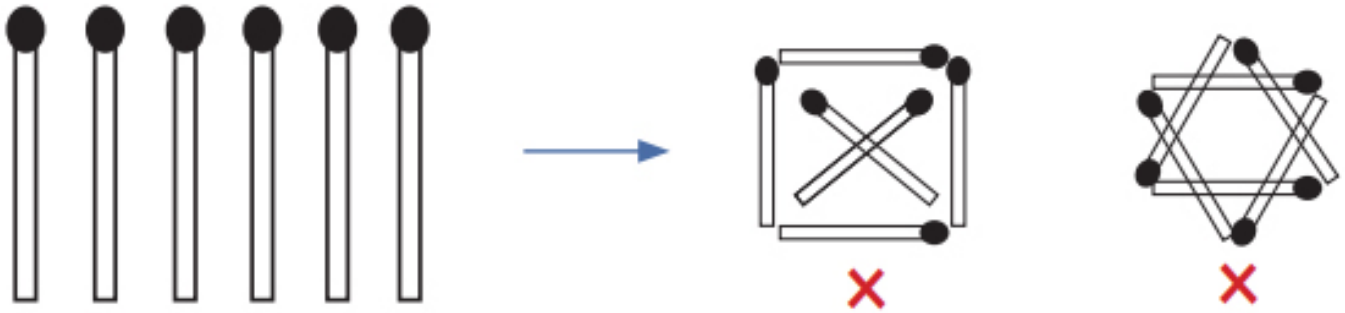
다층 퍼셉트론의 등장

다층 퍼셉트론의 등장

다층 퍼셉트론의 등장

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

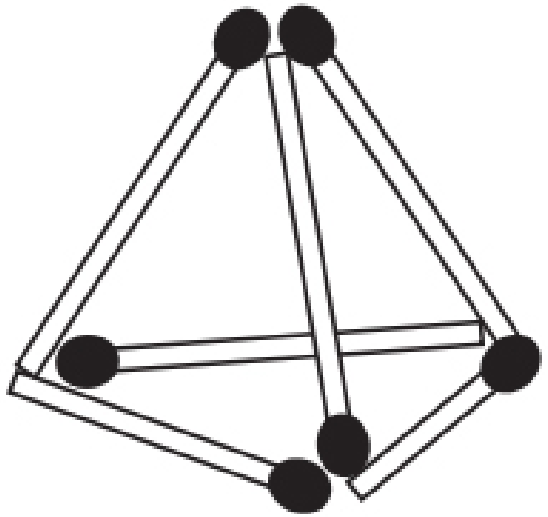
▼ 그림 8-1 | 성냥개비 여섯 개로 정삼각형 네 개를?



☑ 다층 퍼셉트론의 등장

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

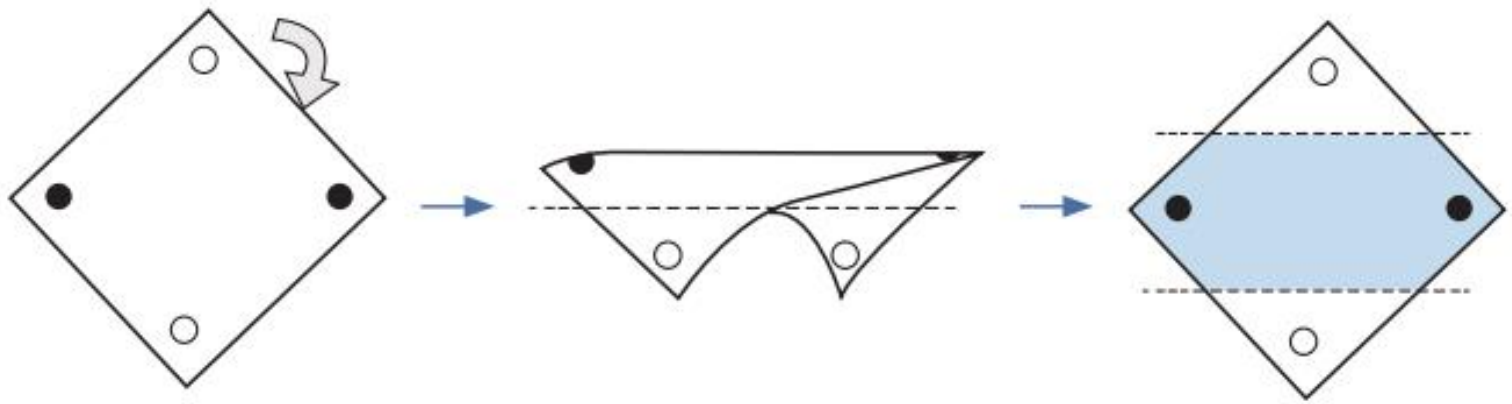
▼ 그림 8-2 | 차원을 달리하니 쉽게 완성!



다층 퍼셉트론의 등장

- 인공지능 학자들은 인공 신경망을 개발하기 위해 반드시 XOR 문제를 극복해야만 했음
- 이 문제를 해결하는 데도 고정 관념을 깨는 기발한 아이디어가 필요했음
- 이러한 노력은 결국 그림 8-3과 같은 아이디어를 낳았음

▼ 그림 8-3 | XOR 문제의 해결은 평면을 휘어 주는 것!



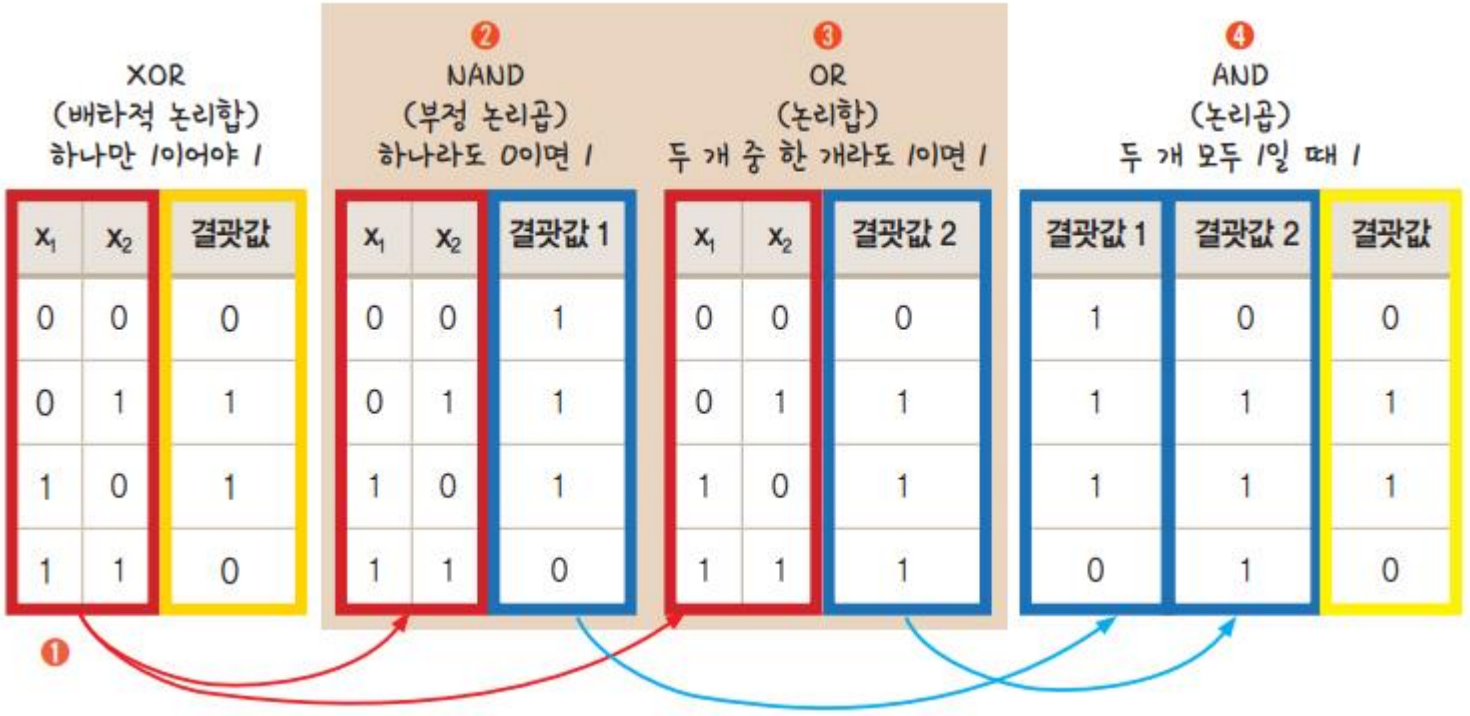
다층 퍼셉트론의 등장

- 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
- 은닉층을 만드는 것이 어떻게 XOR 문제를 해결하는지는 그림 8-4에 소개되어 있음

다층 퍼셉트론의 등장

☑ 다층 퍼셉트론의 등장

▼ 그림 8-4 | 은닉층이 XOR 문제를 해결하는 원리



x_1	x_2	결괏값
0	0	0
0	1	1
1	0	1
1	1	0

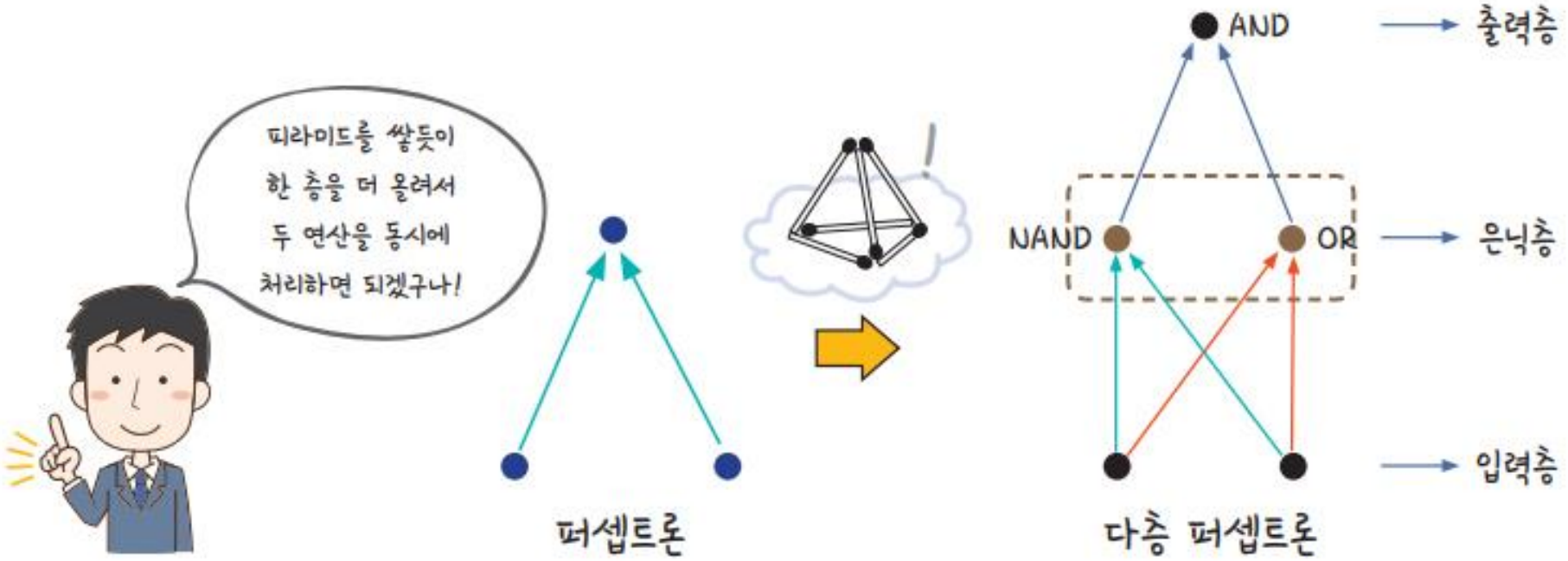
x_1	x_2	결괏값 1
0	0	1
0	1	1
1	0	1
1	1	0

x_1	x_2	결괏값 2
0	0	0
0	1	1
1	0	1
1	1	1

결괏값 1	결괏값 2	결괏값
1	0	0
1	1	1
1	1	1
0	1	0

다층 퍼셉트론의 등장

☑ 다층 퍼셉트론의 등장



다층 퍼셉트론의 등장

- 먼저 ❶ x1과 x2를 두 연산으로 각각 보냄
- ❷ 첫 번째 연산에서는 NAND 처리를 함
- ❸ 이와 동시에 두 번째 연산에서 OR 처리를 함
- ❷ 와 ❸ 을 통해 구한 결괏값 1과 결괏값 2를 ❹ 이고 AND 처리를 하면 우리가 구하고자 하는 출력 값을 만들 수 있음

02

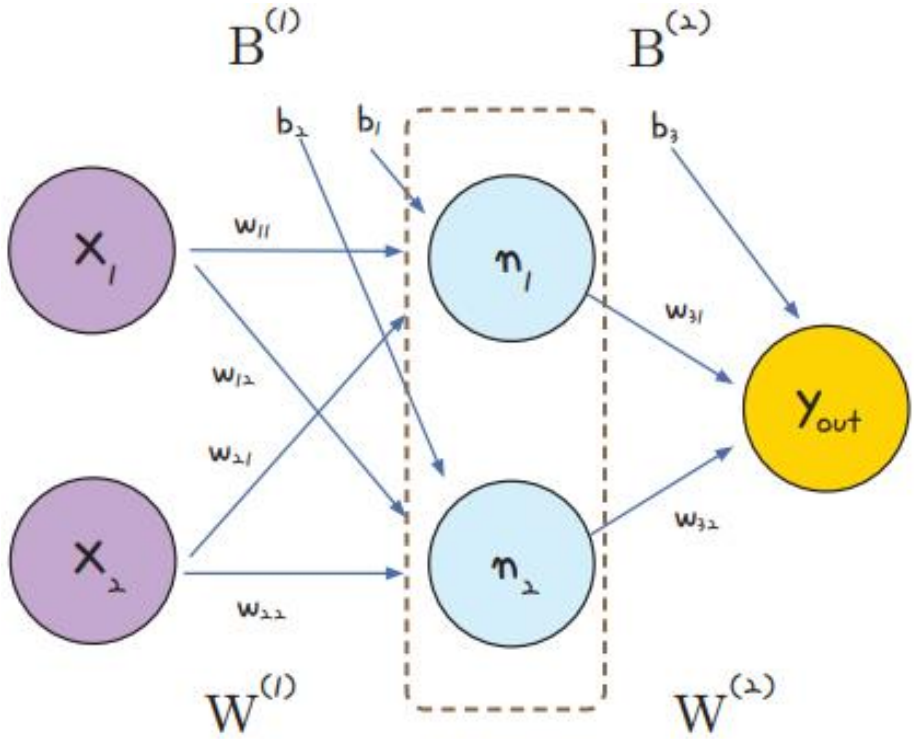
다층 퍼셉트론의 설계

다층 퍼셉트론의 설계

다층 퍼셉트론의 설계

- 다층 퍼셉트론이 입력층과 출력층 사이에 숨어 있는 은닉층을 만드는 것을 그림으로 나타내면 그림 8-5와 같음

그림 8-5 | 다층 퍼셉트론의 내부



다층 퍼셉트론의 설계

다층 퍼셉트론의 설계

- 가운데 점선으로 표시된 부분이 은닉층
- x1과 x2는 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 n1과 n2로 표시
- 은닉층에 취합된 값들은 활성화 함수를 통해 다음으로 보내는데, 만약 앞서 배운 시그모이드 함수((x))를 활성화 함수로 사용한다면 n1과 n2에서 계산되는 값은 각각 다음과 같음

$$n_1 = \sigma(x_1 w_{11} + x_2 w_{21} + b_1)$$
$$n_2 = \sigma(x_1 w_{12} + x_2 w_{22} + b_2)$$

다층 퍼셉트론의 설계

- 두 식의 결괏값이 출력층의 방향으로 보내어지고, 출력층으로 전달된 값은 마찬가지로 활성화 함수를 사용해 y 예측 값을 정하게 됨
- 이 값을 yout이라고 할 때 이를 식으로 표현하면 다음과 같음

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

다층 퍼셉트론의 설계

다층 퍼셉트론의 설계

- 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

$$W^{(1)} = \begin{bmatrix} w_{11} & w_{12} \\ w_{21} & w_{22} \end{bmatrix} \qquad B^{(1)} = \begin{bmatrix} b_1 \\ b_2 \end{bmatrix}$$

$$W^{(2)} = \begin{bmatrix} w_{31} \\ w_{32} \end{bmatrix} \qquad B^{(2)} = [b_3]$$

03

XOR 문제의 해결

XOR 문제의 해결

XOR 문제의 해결

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

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

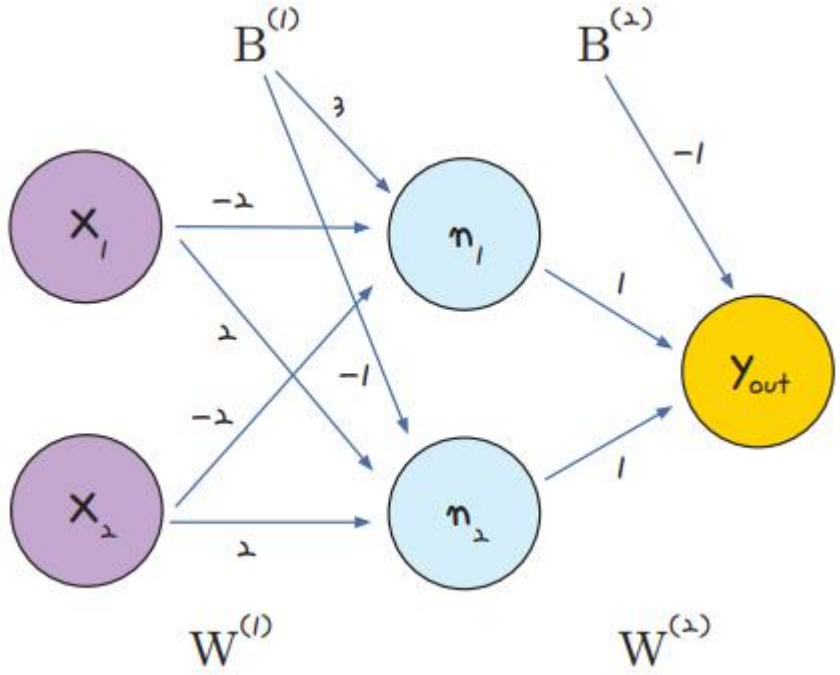
$$W^{(2)} = \begin{bmatrix} 1 \\ 1 \end{bmatrix} \qquad B^{(2)} = [-1]$$

XOR 문제의 해결

XOR 문제의 해결

- 이것을 그림에 대입하면 그림 8-6과 같음

▼ 그림 8-6 | 다층 퍼셉트론의 내부에 변수 채우기



XOR 문제의 해결

☑ XOR 문제의 해결

- 이제 x1 값과 x2 값을 각각 입력해 우리가 원하는 y 값이 나오는지 점검해 보자

▼ 표 8-1 | XOR 다층 문제 해결

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

- ≈ 기호는 '거의 같다'를 의미

XOR 문제의 해결

- 표 8-1에서 볼 수 있듯이 n1, n2, y를 구하는 공식에 차례로 대입하니 우리가 원하는 결과를 구할 수 있었음
- 숨어 있는 노드 두 개를 둔 다층 퍼셉트론을 통해 XOR 문제가 해결된 것

04

코딩으로 XOR 문제 해결하기

코딩으로 XOR 문제 해결하기

☑ 코딩으로 XOR 문제 해결하기

- 이제 주어진 가중치와 바이어스를 이용해 XOR 문제를 해결하는 파이썬 코드를 작성해 볼까?
- 정해진 가중치와 바이어스를 넘파이 라이브러리를 사용해 다음과 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
```

코딩으로 XOR 문제 해결하기

☑ 코딩으로 XOR 문제 해결하기

- 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
```

코딩으로 XOR 문제 해결하기

☑ 코딩으로 XOR 문제 해결하기

- 각 게이트의 정의에 따라 NAND 게이트, OR 게이트, AND 게이트, 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)
```

코딩으로 XOR 문제 해결하기

☑ 코딩으로 XOR 문제 해결하기

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

코딩으로 XOR 문제 해결하기

☑ 코딩으로 XOR 문제 해결하기

- 이제 x1 값과 x2 값을 번갈아 대입해 가며 최종 값을 출력해 보자

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

코딩으로 XOR 문제 해결하기

☑ 코딩으로 XOR 문제 해결하기

- 모두 정리하면 다음과 같음

실습 다층 퍼셉트론으로 XOR 문제 해결하기

```
import numpy as np

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

코딩으로 XOR 문제 해결하기

☑ 코딩으로 XOR 문제 해결하기

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

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

코딩으로 XOR 문제 해결하기

☑ 코딩으로 XOR 문제 해결하기

```
# 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))
```

코딩으로 XOR 문제 해결하기

☑ 코딩으로 XOR 문제 해결하기

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

코딩으로 XOR 문제 해결하기

☑ 코딩으로 XOR 문제 해결하기

실행 결과

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

코딩으로 XOR 문제 해결하기

☑ 코딩으로 XOR 문제 해결하기

- 우리가 원하는 XOR 문제의 정답이 도출
- 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 겨울이라고 함

코딩으로 XOR 문제 해결하기

☑ 코딩으로 XOR 문제 해결하기

- 이 겨울을 지나며 데이터 과학자들은 두 부류로 나뉨
- 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

코딩으로 XOR 문제 해결하기

☑ 코딩으로 XOR 문제 해결하기

▼ 그림 8-7 | 한눈에 보는 인공지능의 역사: 퍼셉트론에서 딥러닝까지