강사 윤영주
yj.youn1103@gmail.com

셋째마당
딥러닝의 시작, 신경망

8장 다층 퍼셉트론

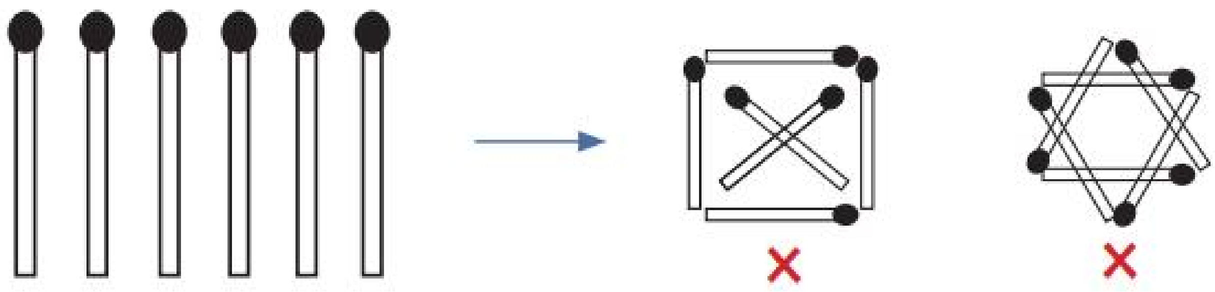
1 다층 퍼셉트론의 등장

1 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?

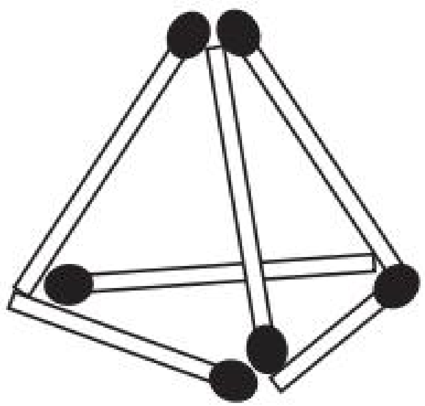
▼ 그림 8-1 | 성냥개비 여섯 개로 정삼각형 네 개를?



1 다층 퍼셉트론의 등장

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

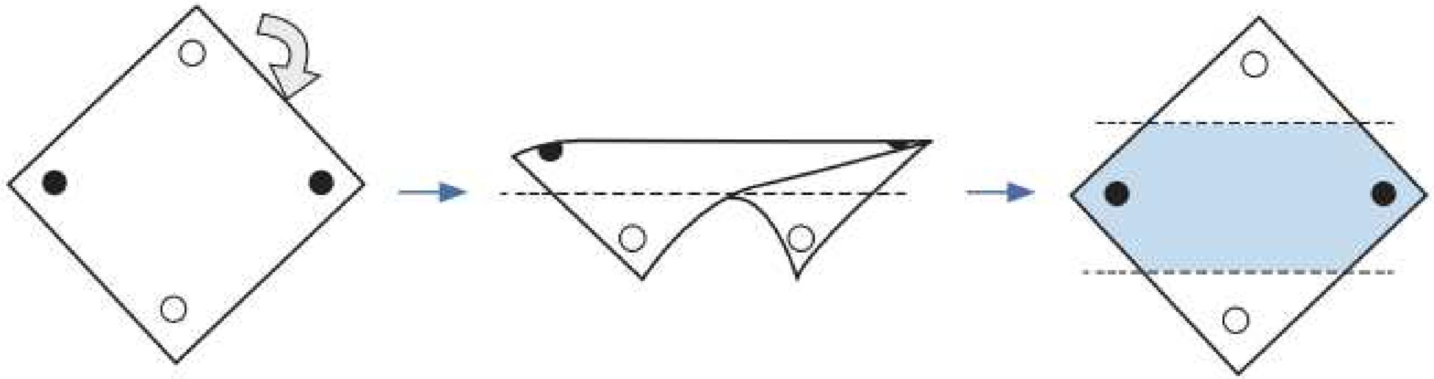
▼ 그림 8-2 | 차원을 달리하니 쉽게 완성!



1 다층 퍼셉트론의 등장

- 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!



1 다층 퍼셉트론의 등장

- **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

1 다층 퍼셉트론의 등장

▼ 그림 8-4 | 은닉층이 XOR 문제를 해결하는 원리

❶ XOR (배타적 논리합) 하나만 1이어야 1

x_1	x_2	결괏값
0	0	0
0	1	1
1	0	1
1	1	0

❷ NAND (부정 논리곱) 하나라도 0이면 1

x_1	x_2	결괏값 1
0	0	1
0	1	1
1	0	1
1	1	0

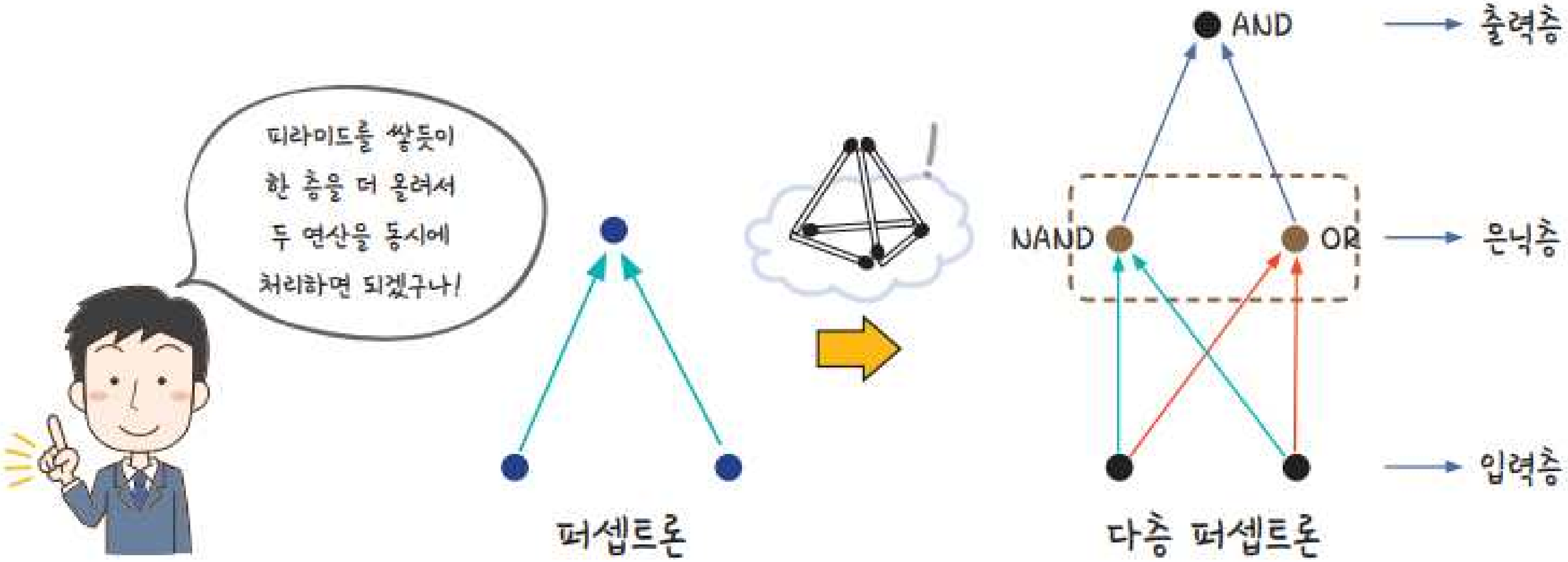
❸ OR (논리합) 두 개 중 한 개라도 1이면 1

x_1	x_2	결괏값 2
0	0	0
0	1	1
1	0	1
1	1	1

❹ AND (논리곱) 두 개 모두 1일 때 1

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

1 다층 퍼셉트론의 등장



피라미드를 쌓듯이
한 층을 더 올려서
두 연산을 동시에
처리하면 되겠구나!
퍼셉트론
AND
출력층
NAND
OR
은닉층
입력층
다층 퍼셉트론

1 다층 퍼셉트론의 등장

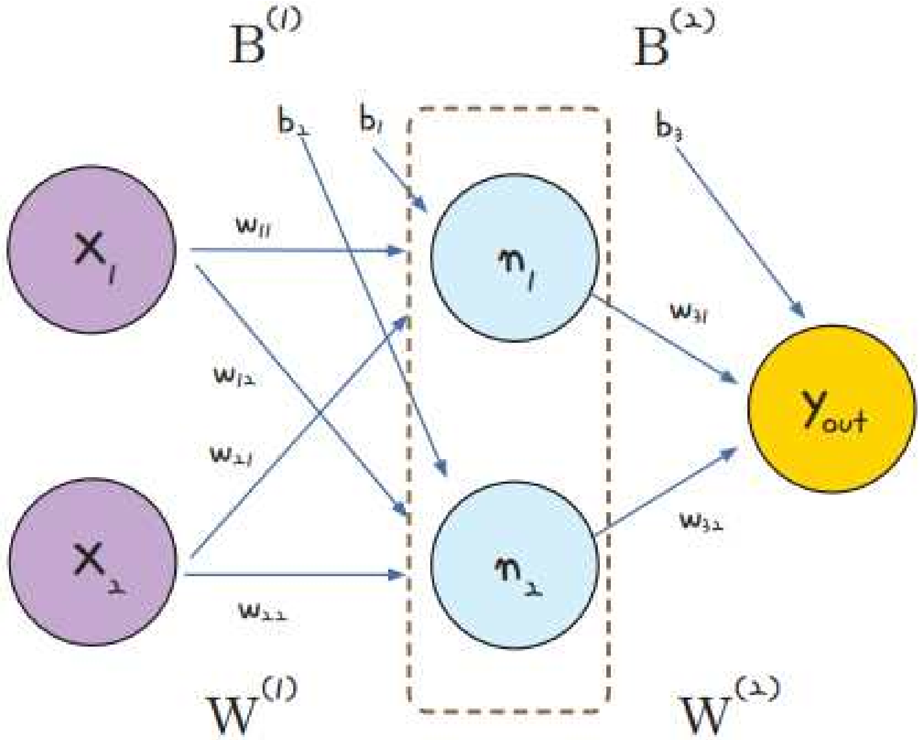
- **다층 퍼셉트론의 등장**
  - 먼저 ❶ x_1과 x_2를 두 연산으로 각각 보냄
  - ❷ 첫 번째 연산에서는 NAND 처리를 함
  - ❸ 이와 동시에 두 번째 연산에서 OR 처리를 함
  - ❷와 ❸을 통해 구한 결괏값 1과 결괏값 2를 가지고 ❹ AND 처리를 하면 우리가 구하고자 하는 출력 값을 만들 수 있음

2 다층 퍼셉트론의 설계

2 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**



2 다층 퍼셉트론의 설계

- **다층 퍼셉트론의 설계**
  - 가운데 점선으로 표시된 부분이 은닉층
  - x_1과 x_2는 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_1과 n_2로 표시
  - 활성화 함수를 통해 다음으로 보내는데, 만약 앞서 배운 활성화 함수로 사용한다면 n_1과 n_2에서 계산되는 값은

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

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

2 다층 퍼셉트론의 설계

- **다층 퍼셉트론의 설계**
  - 두 식의 결괏값이 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

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

2 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

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

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

3 XOR 문제의 해결

3 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

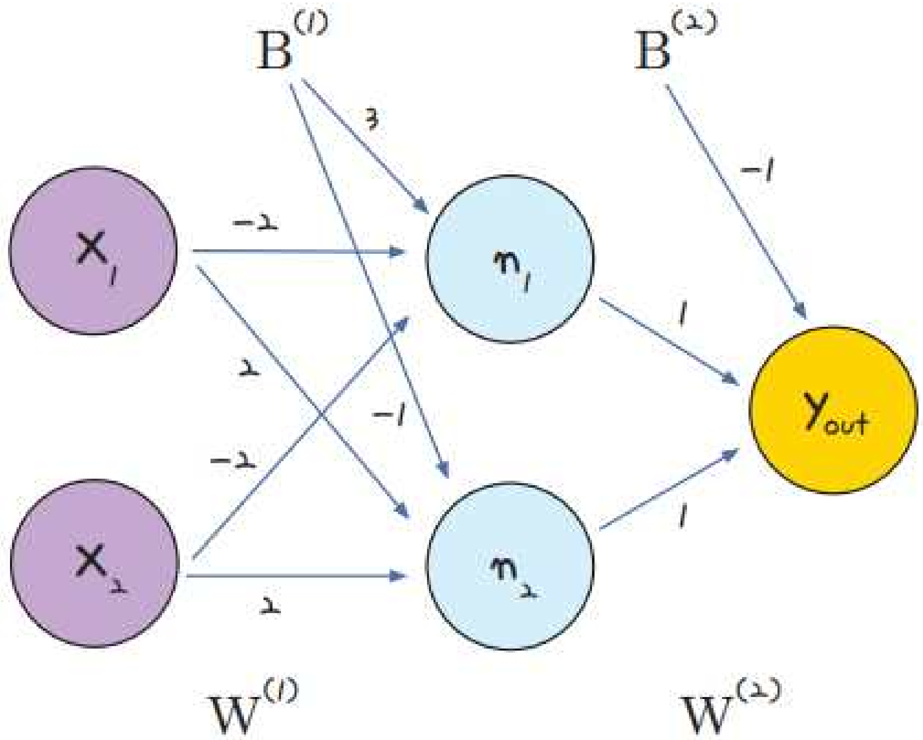
$$W^{(1)} = \begin{bmatrix} -2 & 2 \\ -2 & 2 \end{bmatrix} \qquad B^{(1)} = \begin{bmatrix} 3 \\ -1 \end{bmatrix}$$

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

3 XOR 문제의 해결

- **XOR 문제의 해결**
  - 이것을 그림에 대입하면 그림 8-6과 같음

▼ **그림 8-6 | 다층 퍼셉트론의 내부에 변수 채우기**



3 XOR 문제의 해결

XOR 문제의 해결

- 이제 x_1 값과 x_2 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

3 XOR 문제의 해결

- **XOR 문제의 해결**
  - 표 8-1에서 볼 수 있듯이 n_1, n_2, y를 구하는 공식에 차례로 대입하니 우리가 원하는 결과를 구할 수 있었음
  - 숨어 있는 노드 두 개를 둔 다층 퍼셉트론을 통해 XOR 문제가 해결된 것

4 코딩으로 XOR 문제 해결하기

4 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

```
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
```

4 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
```

4 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)
```

4 코딩으로 XOR 문제 해결하기

● 코딩으로 XOR 문제 해결하기

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

4 코딩으로 XOR 문제 해결하기

- 코딩으로 XOR 문제 해결하기
  - 이제 x_1 값과 x_2 값을 번갈아 대입해 가며 최종 값을 출력해 보자

```python
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))
```

4 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
```

4 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)
```

4 코딩으로 XOR 문제 해결하기

코딩으로 XOR 문제 해결하기

```python
# 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))
```

4 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))
```

4 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
```

4 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

4 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**

4 코딩으로 XOR 문제 해결하기

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