7장 다층 퍼셉트론

목 차

다층 퍼셉트론

- 종이 위에 각각 엇갈려 놓인 검은점 두 개와 흰점 두 개를 하나의 선으로는 구별할 수 없다는 것을 살펴봄
- 언뜻 보기에 해답이 없어 보이는 이 문제를 해결하려면 새로운 접근이 필요함

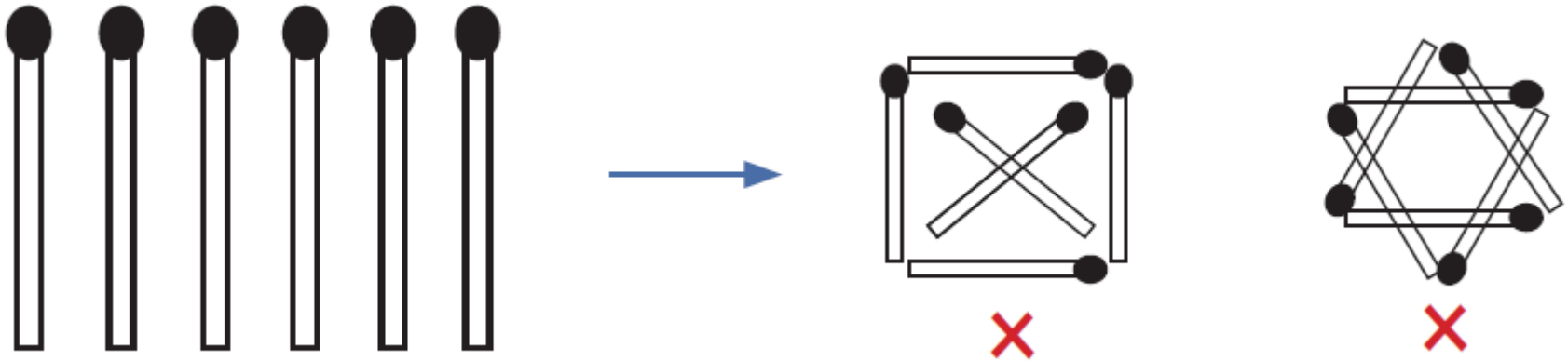

그림 7-1 성냥개비 여섯 개로 정삼각형 네 개를?

다층 퍼셉트론

- 골똘히 연구해도 답을 찾지 못했던 이 문제는 2차원 평면에서만 해결하려는 고정관념을 깨고 피라미드 모양으로 성냥개비를 쌓아 올리니 해결됨

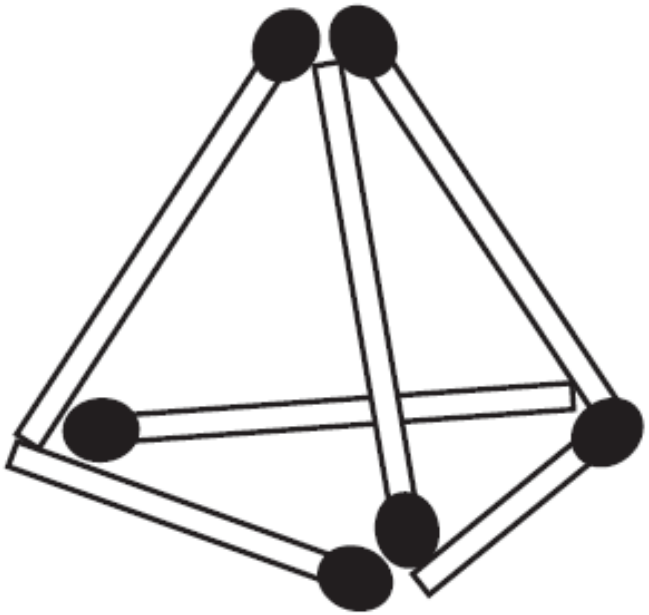

그림 7-2 차원을 달리하니 쉽게 완성!

다층 퍼셉트론

- 인공지능 학자들은 인공 신경망을 개발하기 위해서 반드시 XOR 문제를 극복해야만 했음
- 이 문제 역시 고정관념을 깬 기발한 아이디어에서 해결점이 보였음

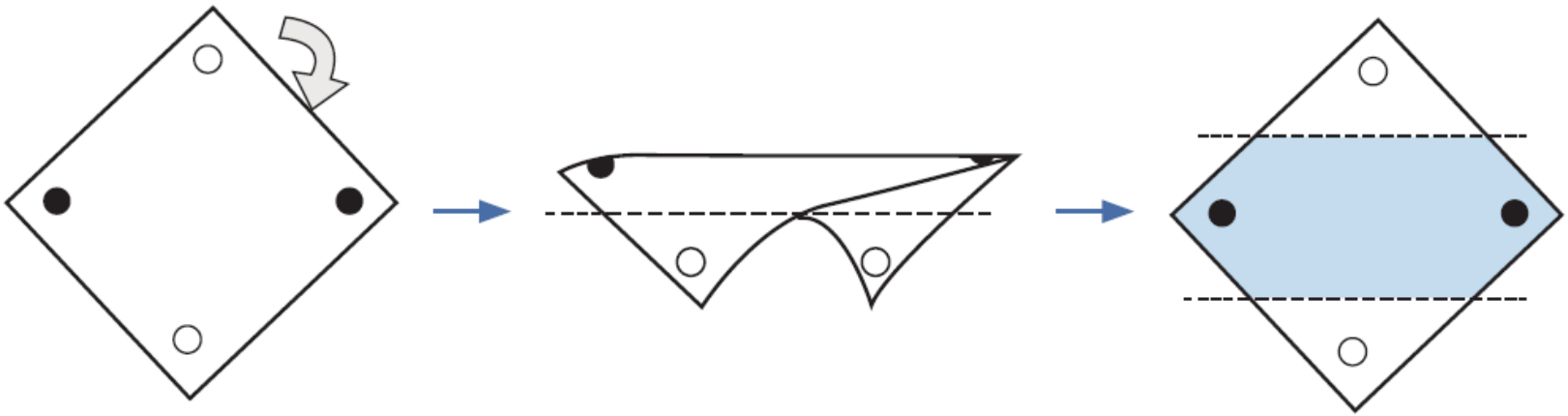

그림 7-3 XOR 문제의 해결은 평면을 휘어주는 것!

다층 퍼셉트론

- 좌표 평면 자체에 변화를 주는 것
- XOR 문제를 해결하기 위해서 우리는 두 개의 퍼셉트론을 한 번에 계산할 수 있어야 함
- 이를 가능하게 하려면 숨어있는 층, 즉 은닉층(hidden layer)을 만들면 됨

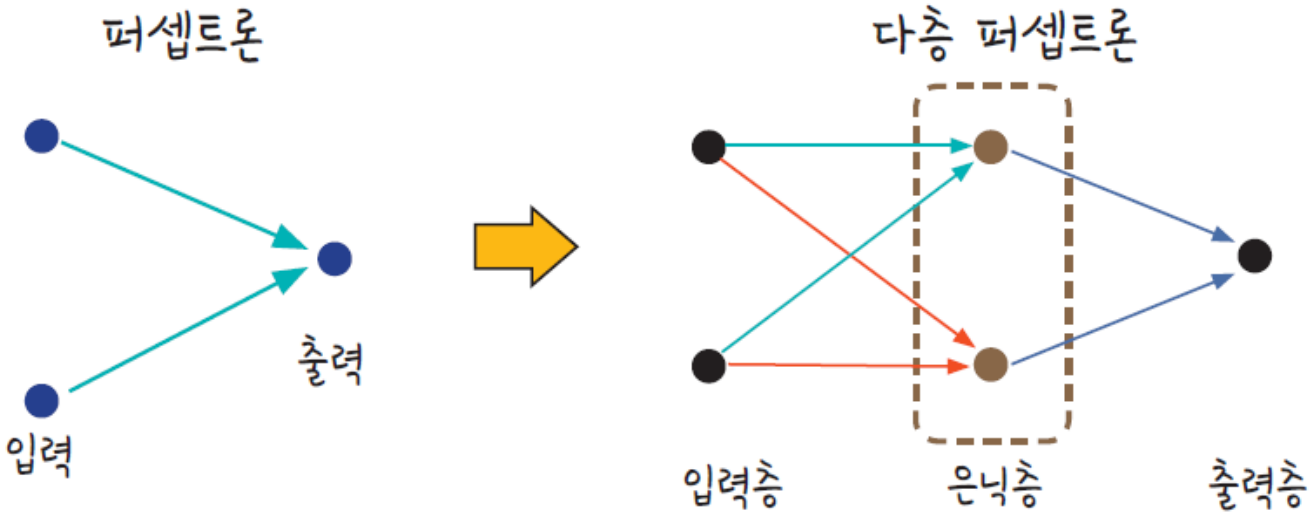

그림 7-4 퍼셉트론에서 다층 퍼셉트론으로

다층 퍼셉트론

- 은닉층이 좌표 평면을 왜곡시키는 결과를 가져옴

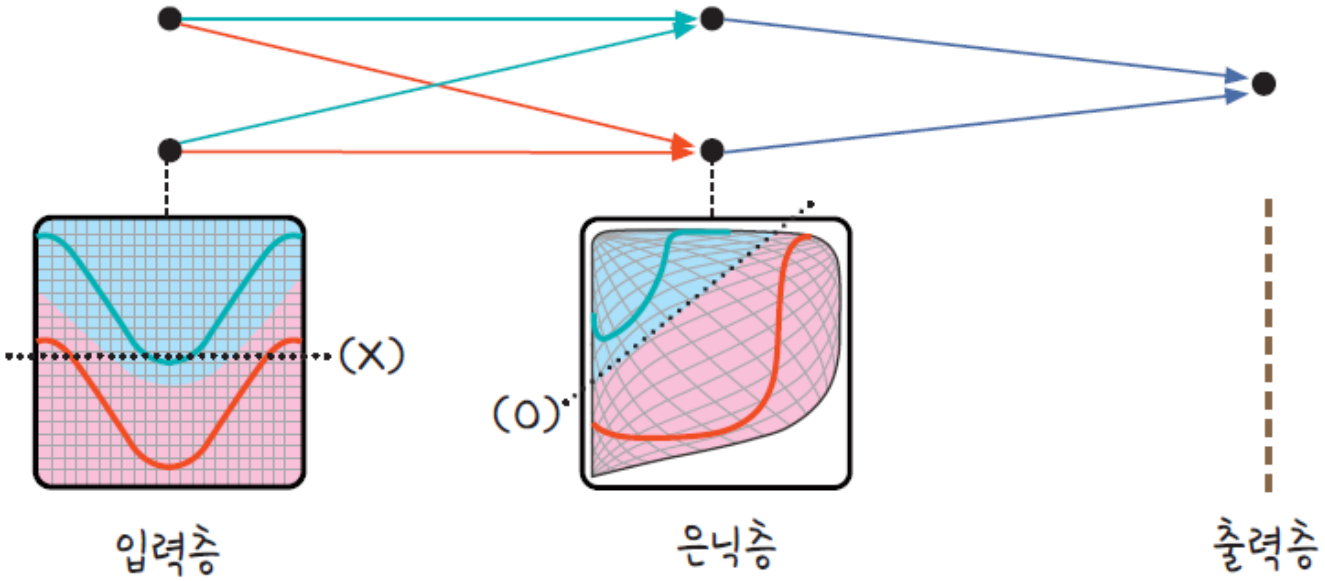

그림 7-5 은닉층의 공간 왜곡(https://goo.gl/8qEGHD 참조)

다층 퍼셉트론

- 입력 값(input)을 놓고 파란색과 빨간색의 영역을 구분한다고 할 때,

  그림 7-5의 왼쪽 그림을 보면 어떤 직선으로도 이를 해결할 수 없음
- 은닉층을 만들어 공간을 왜곡하면 두 영역을 가로지르는 선이 직선으로 바뀜

1 다층 퍼셉트론의 설계

- 다층 퍼셉트론이 입력층과 출력층 사이에 숨어있는 은닉층을 만드는 것을 도식으로 나타내면 그림 7-6과 같음

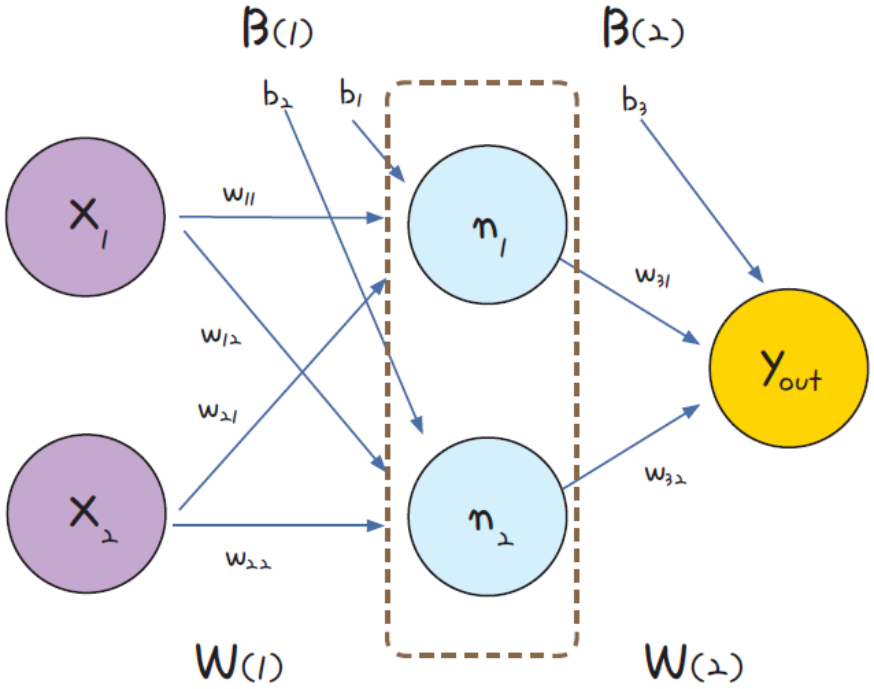

그림 7-6 다중 퍼셉트론의 내부

1 다층 퍼셉트론의 설계

- 가운데 숨어있는 은닉층으로 퍼셉트론이 각각 자신의 가중치(w)와 바이어스(b) 값을 보냄
- 이 은닉층에서 모인 값이 한 번 더 시그모이드 함수(기호로 σ 라고 표시함)를 이용해 최종 값으로 결과를 보냄
- 노드(node) :

  은닉층에 모이는 중간 정거장

  여기서는 n_1과 n_2로 표현함

1 다층 퍼셉트론의 설계

- n_1과 n_2의 값은 각각 단일 퍼셉트론의 값과 같음

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

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

- 위 두 식의 결괏값이 출력층으로 보내짐
- 출력층에서는 역시 시그모이드 함수를 통해 y값이 정해짐
- 이 값을 y_{out}라 할 때 식으로 표현하면 다음과 같음

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

1 다층 퍼셉트론의 설계

- 각각의 가중치(w)와 바이어스(b)의 값을 정할 차례임
- 2차원 배열로 늘어놓으면 다음과 같이 표시할 수 있음
- 은닉층을 포함해 가중치 6개와 바이어스 3개가 필요함

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

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

2 XOR 문제의 해결

- 각 변숫값을 정하고 이를 이용해 XOR 문제를 해결하는 과정을 알아보자

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

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

2 XOR 문제의 해결

- 이것을 도식에 대입하면 다음과 같음

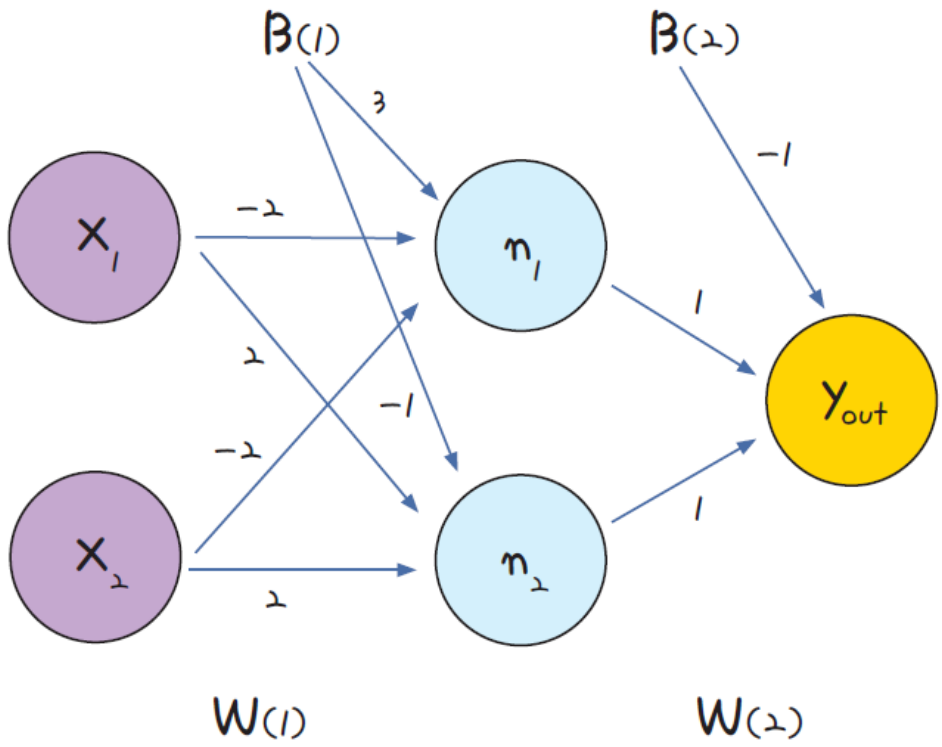

그림 7-7 다중 퍼셉트론의 내부에 변수를 채워보자.

2 XOR 문제의 해결

- 이제 X_1의 값과 X_2의 값을 각각 입력해 값이 우리가 원하는 값으로 나오는지를 점검해 보자

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

표 7-1 XOR 다층 문제 해

TIP

≈ 기호는 '거의 같다'를 의미하는 것으로 이해하면 됩니다.

3 코딩으로 XOR 문제 해결하기

- 먼저 표 7-1에서 n_1 의 값을 잘 보면 입력 값 X_1 , X_2 가 모두 1일 때 0을 출력하고 하나라도 0이 아니면 1을 출력하게 되어있음
- NAND(Negative And) 게이트 :

  AND 게이트의 정반대 값을 출력하는 방식
- n_2의 값을 잘 보면 X_1 , X_2 에 대한 OR 게이트에 대한 답
- NAND 게이트와 OR 게이트, 이 두 가지를 내재한 각각의 퍼셉트론이 다중 레이어 안에서 각각 작동함
- 이 두 가지 값에 대해 AND 게이트를 수행한 값이 바로 우리가 구하고자 하는 임을 알 수 있음

3 코딩으로 XOR 문제 해결하기

- 정해진 가중치와 바이어스를 numpy라이브러리를 사용해 다음과 같이 선언함

```
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
```

3 코딩으로 XOR 문제 해결하기

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

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

3 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)
```

3 코딩으로 XOR 문제 해결하기

```
# 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))
```

3 코딩으로 XOR 문제 해결하기

- 이제 X_1과 X_2 값을 번갈아 대입해 가며 최종 값을 출력해 보자

```
if __name__ == '__main__':
    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))
```

3 코딩으로 XOR 문제 해결하기

코드 7-1 다층 퍼셉트론으로 XOR 문제 해결하기

- 예제 소스: deeplearning_class/06_XOR.ipynb

```
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
```

3 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)
```

3 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))
```

3 코딩으로 XOR 문제 해결하기

```
# x1, x2 값을 번갈아 대입하며 최종값 출력
if __name__ == '__main__':
    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))
```

3 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
```

3 코딩으로 XOR 문제 해결하기

- 인공 신경망 :

  은닉층을 여러 개 쌓아올려 복잡한 문제를 해결하는 과정은 뉴런이 복잡한 과정을 거쳐 사고를 낳는 사람의 신경망을 닮았음
- 이를 간단히 줄여서 신경망이라고 통칭함