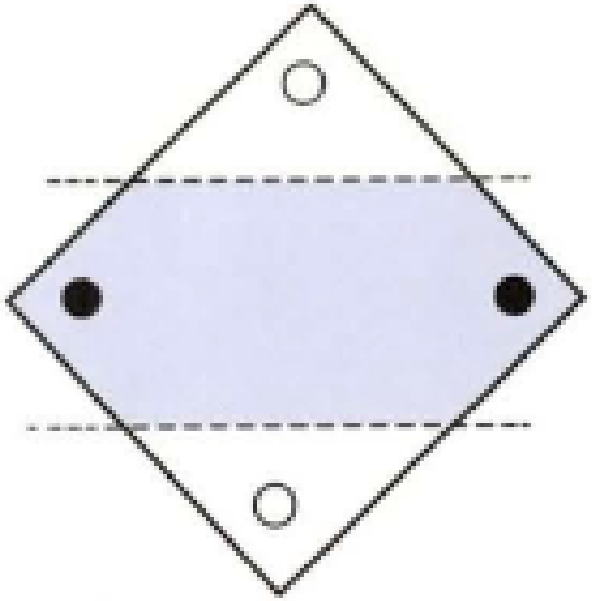
8. 다층 퍼셉트론

서론

- 앞서 종이 위에 각각 엇갈려 놓인 검은점 두 개와 흰점 두 개를 하나의 선으로는 구별할 수 없다는 것을 살펴보았습니다
- 언뜻 보기에 해답이 없어 보이는 이 문제를 해결하려면 새로운 접근이 필요합니다.
- 인공지능 학지들은 인공 신경망을 개발하기 위해서 반드시 XOR 문제를 극복해야만 했습니다.
- 이 문제 역시 고정관념을 깬 기발한 아이디어에서 해결점이 보였습니다.

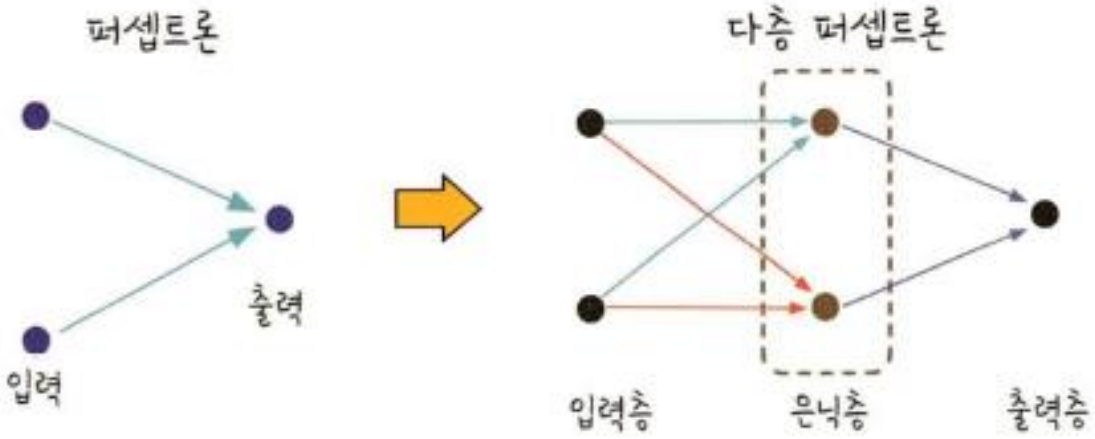
서론

- 바로 선을 2개 그어주는 것이었습니다.



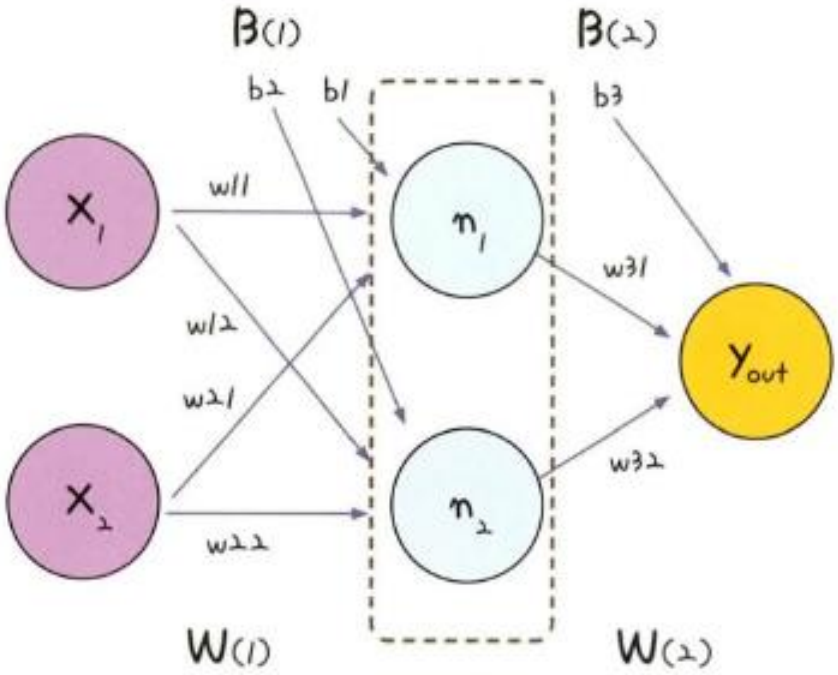
서론

- XOR 문제를 해결하기 위해서 우리는 두 개의 퍼셉트론을 한 번에 계산할 수 있어야 합니다.
- 이를 가능하게 하려면 숨어 있는 층, 즉 은닉층(hidden layer)을 만들면 됩니다.



다층 퍼셉트론의 설계

- 다층 퍼셉트론이 입력 층과 출력층 사이에 숨어 있는 은닉 층을 만드는 것을 도식으로 나타내면 다음 그림과 같습니다.



다층 퍼셉트론의 설계

- 가운데 숨어 있는 은닉층으로 퍼셉트론이 각각 자신의 가중치 (w)와 바이어스(b) 값을 보내고, 이 은닉층에서 모인 값이 한 번 더 시그모이드 함수(기 호로 σ라고 표시합니다)를 이용해 최종 값으로 결과를 보냅니다.
- 은닉층에 모이는 중간 정거장을 노드(node)라고 하며 여기서는 n_1과 n_2로 표현하였습니다.

다층 퍼셉트론의 설계

- n_1과 n_2의 값은 각각 단일 퍼셉트론의 값과 같습니다.

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

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

- 위 두 식의 결과값이 출력층으로 보내집니다.
- 출력층에서는 역시 시그모이드 함수를 통해 y 값이 정해집니다.

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

다층 퍼셉트론의 설계

- 이제 각각의 가중치(x)와 바이어스(b)의 값을 정할 차례입니다.
- 2차원 배열로 늘어놓으면 다음과 같이 표시할 수 있습니다.
- 은닉층을 포함해 가중치 6개와 바이어스 3개가 필요합니다.

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

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

XOR 문제의 해결

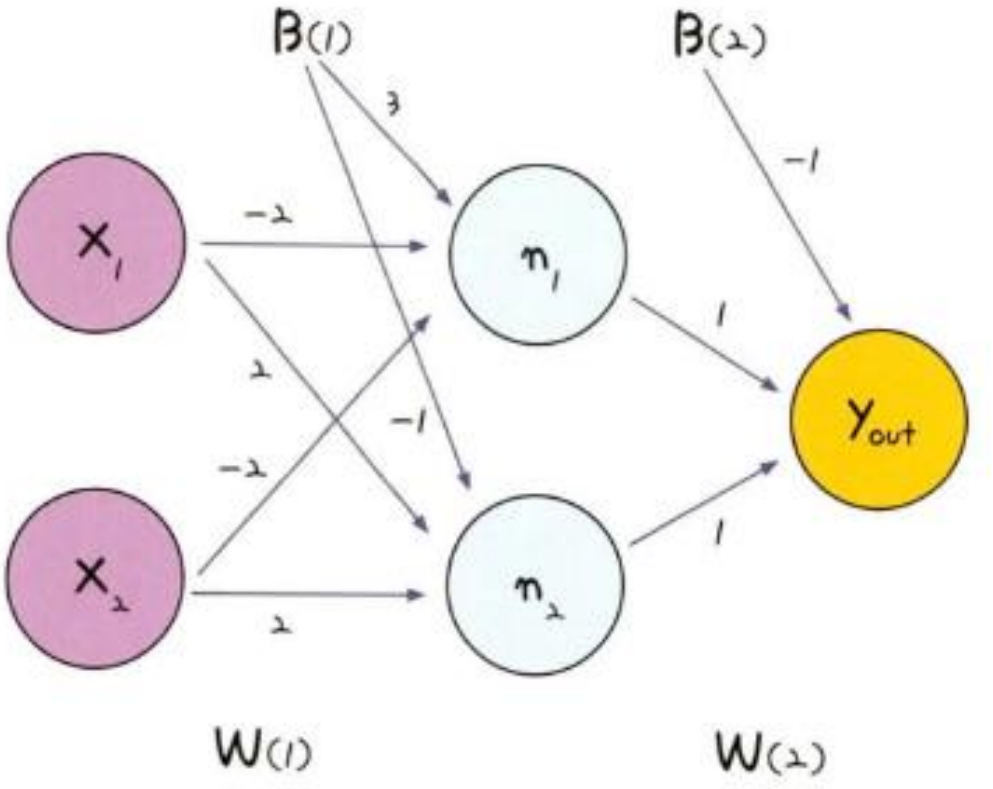
- 앞서 우리에게 어떤 가중치와 바이어스가 필요한지를 알아보았습니다.
- 이를 만족하는 가중치와 바이어스의 조합은 무수히 많습니다.
- 이를 구하는 방법은 8장에서 소개할 예정입니다. 지금은 먼저 다음과 같이 각 변수값을 정하고 이를 이용해 XOR 문제를 해결하는 과정을 알아보겠습니다

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

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

XOR 문제의 해결

- 이것을 도식에 대입하면 다음과 같습니다.



XOR 문제의 해결

- 이제 x_1의 값과 x_2의 값을 각각 입력해 y 값이 우리가 원하는 값으로 나오는 지를 점검해 보겠습니다.

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

- 숨어있는 두 개의 노드를 둔 다층 퍼셉트론을 통해 XOR 문제가 해결된 것입니다.

코딩으로 XOR 문제 해결하기

- 이제 주어진 가중치와 바이어스를 이용해 XOR 문제를 해결하는 파이씬 코드를 작성해볼까요?
- 먼저 표에서 n_1의 값을 잘 보면 입력 값 x_1, x_2가 모두 1일 때 0을 출력하고 하나라도 0이 아니면 1을 출력하게 되어 있습니다.
- 이는 AND 게이트의 정 반대 값을 출력하는 방식입니다.
- 이를 NAND 게이트라고 부릅니다.
- 그리고 n_2의 값을 잘보면 x_1, x_2에 대한 OR 게이트에 대한 답입니다.

코딩으로 XOR 문제 해결하기

- NAND 게이트와 OR 게이트 이 두 가지를 내재한 각각의 퍼셉트론이 다중 레이어 안에서 각각 작동하고 이 두가지의 값에 대해 AND 게이트를 수행한 값이 바로 우리가 구하고자 하는 Y_{out} 임 을 알 수 있습니다.

코딩으로 XOR 문제 해결하기

- 정해진 가중치와 바이어스를 넘파이 라이브러리를 사용해 다음과 같이 선언하겠습니다.

```
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

- 이제 퍼셉트론 함수를 만들어 줍니다.
- 0과 1 중에서 값을 출력하게 설정합니다.

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

코딩으로 XOR 문제 해결하기

- 각 게이트의 정의 에 따라 NAND 게이트, OR 게이트, AND 게이트, XOR 게이트 함수를 만들어 줍니다.

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

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

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

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

코딩으로 XOR 문제 해결하기

- 이제 x_1과 x_2 값을 번갈아 대입해 가며 최종 값을 출력해 봅시다.

```
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))
```

코딩으로 XOR 문제 해결하기

- 모두 정리하면 다음과 같습니다.

```
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

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

코딩으로 XOR 문제 해결하기

- 모두 정리하면 다음과 같습니다.

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

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

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

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

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))
```

코딩으로 XOR 문제 해결하기

- 코드를 실행하면 다음과 같은 결과가 출력됩니다.

```
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

- 우리가 원하는 XOR 문제의 정답이 도출되었습니다.
- 이렇게 퍼셉트론 하나로 해결되지 않던 문제를 은닉층을 만들어 해결했습니다.
- 은닉층을 여러 개 쌓아올려 복잡한 문제를 해결하는 과정이 뉴런이 복잡한 과정을 거쳐 사고를 낳는 사람의 신경망을 닮았습니다.
- 그래서 이 방법을 인공 신경망이라 부르기 시작했고, 이를 간단히 줄여서 신경망이라고 통칭합니다.

Q&A