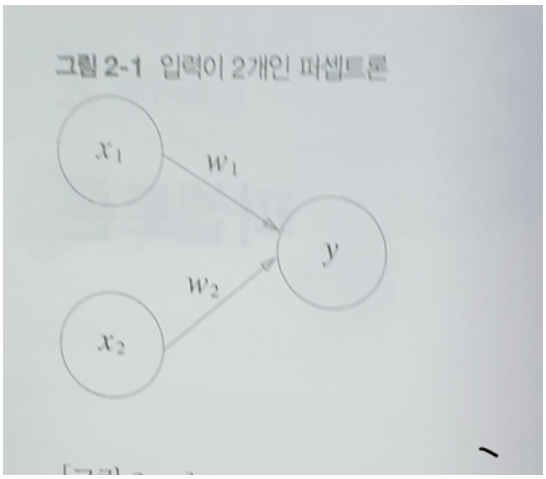
2. 퍼셉트론

이번 장에서는 **퍼셉트론** 알고리즘을 설명합니다. 퍼셉트론은 신경망(딥러닝)의 기원이 되는 알고리즘입니다. 이에 퍼셉트론의 구조를 배우는 것은 신경망과 딥러닝으로 나아가는 데 중요한 아이디어를 배우는 일도 된다.

이번 장에서는 퍼셉트론을 설명하고 퍼셉트론을 써서 간단한 문제를 풀어간다.

2.1 퍼셉트론이란?

퍼셉트론은 다수의 신호를 입력으로 받아 하나의 신호를 출력한다. 여기서 말하는 **신호**란 전류나 강물처럼 **흐름**이 있는 것을 상상하면 좋습니다. 전류가 전선을 타고 흐르는 전자를 내보내듯, 퍼셉트론 신호도 흐름을 만들고 정보를 앞으로 전달한다. 다만, 실제 전류와 달리 퍼셉트론 신호는 '흐른다/ 안 흐른다(0이나 1)' 의 두 가지 값을 가질 수 있다. 이 책에서는 1을 '신호가 흐른다' 0을 '신호가 흐르지 않는다'라는 의미로 쓰겠다.



위 그림은 입력으로 2개의 신호를 받은 퍼셉트론의 예이다. x_1, x_2는 입력신호, y는 출력신호, w_1과 w_2 는 가중치를 뜻한다. 그림의 원을 **뉴런** 혹은 **노드**라고 부른다. 입력 신호가 뉴런에 보내질 때는 각각 고유한 **가중치**가 곱해진다. 뉴런에서 보내온 신호의 총합이 정해진 한계를 넘어설 때만 1을 출력한다. (이를 뉴런이 활성화한다 라고 표현하기도 한다) 이 책에서는 그 한계를 **임계값** 이라 하며 θ 기호로 나타낸다.

퍼셉트론의 동작 원리는 이게 다이다. 이상을 수식으로 나타내면 다음과 같다.

$$y = 0(w_1x_1 + w_2x_2 \leq \theta)$$
$$1(w_1x_1 + w_2x_2 > \theta)$$

퍼셉트론은 복수의 입력 신호 각각에 고유한 가중치를 부여한다. 가중치는 각 신호가 결과에 주는

영향력을 조절하는 요소로 작용한다. 즉, 가중치가 클수록 해당 신호가 그만큼 더 중요함을 뜻한다.

*Note 가중치는 전류에서 말하는 **저항**에 해당한다. 저항은 전류의 흐름을 억제하는 매개변수로, 저항이 낮을수록 큰 전류가 흐른다. 한편 퍼셉트론의 가중치는 그 값이 클수록 강한 신호를 흘려보낸다. 이처럼 서로 작용하는 방향은 반대지만, 신호가 얼마나 잘(혹은 어렵게) 흐르는가를 통제한다는 점에서 저항과 가중치는 같은 기능을 한다.*

2.2 단순한 논리 회로

2.2.1 AND 게이트

퍼셉트론을 활용한 간단한 문제를 보자.

x_1	x_2	x_3
0	0	0
1	0	0
0	1	0
1	1	1

이 AND게이트를 퍼셉트론으로 표현하고자 한다. 이를 위해 할 일은 위 표대로 작동하도록 하는 w_1, w_2, θ의 값을 정하는 것이다. 그럼 어떤 값으로 설정하면 위 표의 조건을 충족하는 퍼셉트론이 만들어질까?

사실 위 표를 만족하는 매개변수 조합은 무한히 많다. 가령(w_1, w_2, θ) 가 (0.5,0.5,0.7)일 때, 또 (0.5,0.9,0.45) 등 모두 AND 게이트의 조건을 만족한다.

*Note 여기서 퍼셉트론의 매개변수 값을 정하는 것은 컴퓨터가 아닌 인간이다. 인간이 직접 진리표라는 "학습 데이터"를 보면서 매개변수의 값을 생각한다. 기계학습 문제는 이 매개변수의 값을 정하는 작업을 컴퓨터가 자동으로 하도록 한다. **학습**이란 적절한 매개변수 값을 정하는 작업이며, 사람은 퍼셉트론의 구조(모델)를 고민하고 컴퓨터에 학습할 데이터를 주는 일을 한다.*

이상과 같이 퍼셉트론으로 AND, NAND, OR 논리 회로를 표현할 수 있음을 알았다. 여기서 중요한 점은 퍼셉트론의 구조는 AND, NAND, OR 게이트 모두에서 똑같다는 것이다. 세 가지 게이트에서 다른 것은 매개변수(가중치와 임계값)의 값 뿐이다. 즉, 마치 팔색조 배우가 다양한 인물을 연기하는 것처럼 똑같은 구조의 퍼셉트론이 매개변수의 값만 적절히 조정하여 AND, NAND, OR로 변신하는 것이다.

2.3 퍼셉트론 구현하기

2.3.1 간단한 구현부터

이제 논리 회로를 파이썬으로 구현해보자. 다음은 x1과 x2를 인수로 받는 AND 함수이다.

```
In [1]: def AND(x1, x2):
   ...:     w1, w2, theta = 0.5,0.5,0.7
   ...:     tmp = x1*w1+x2*w2
   ...:     if tmp <= theta:
   ...:         return 0
   ...:     elif tmp > theta:
   ...:         return 1
```

매개변수 w1, w2, theta는 함수 안에서 초기화하고, 가중치를 곱한 입력의 총합이 임계값을 넘으면 1을 반환하고, 그 외에는 0을 반환한다. 이 함수의 출력이 다음과 같은지 확인해보자.

```
In [2]: AND(0,0)
Out[2]: 0

In [3]: AND(0,1)
Out[3]: 0

In [4]: AND(1,0)
Out[4]: 0

In [5]: AND(1,1)
Out[5]: 1
```

2.3.2 가중치와 편향 도입

앞에서 구현한 AND 게이트는 직관적이고 알기 쉽지만, 앞으로를 생각해서 다른 방식으로 수정하고자 한다. 그 전에

$$y = 0(w_1x_1 + w_2x_2 \leq \theta)$$
$$1(w_1x_1 + w_2x_2 > \theta)$$

위 이 식의 θ를 $-b$로 치환하면 퍼셉트론의 동작이 다음 식처럼 된다.

$$y = 0(b + w_1x_1 + w_2x_2 \leq 0)$$
$$1(b + w_1x_1 + w_2x_2 > 0)$$

위 식과 아래 식은 기호 표기만 바꿨을 뿐, 그 의미는 같다. 여기에서 b를 **편향**이라 하며, w_1과 w_2 는 그대로 가중치이다. 그럼 넘파이를 사용해서 아래식 방식으로 구현해 보자. 여기에서는 파이썬 인터프리터로 순서대로 결과를 확인해보자.

```
In [6]: import numpy as np

In [7]: x = np.array([0,1])

In [8]: w = np.array([0.5,0.5])

In [9]: b = -0.7

In [10]: w*x
Out[10]: array([0. , 0.5])

In [11]: np.sum(w*x)
Out[11]: 0.5

In [12]: np.sum(w*x)+b
Out[12]: -0.19999999999999996
```

넘파이 배열끼리의 곱셈은 두 배열의 원소 수가 같다면 각 원소끼리 곱한다. 그래서 이 예의 w*x에서는 인덱스가 같은 원소끼리 곱한다. 또, np.sum() 메서드는 입력한 배열에 담긴 모든 원소의 총합을 계산한다. 마지막으로 이 가중치에 편향을 더해라.

2.3.3 가중치와 편향 구현하기

'가중치와 편향을 도입'한 AND 게이트는 다음과 같이 구현할 수 있다.

```
In [13]: def AND(x1, x2):
    ...:     x= np.array([x1,x2])
    ...:     w = np.array([0.5,0.5])
    ...:     b=-0.7
    ...:     tmp = np.sum(w*x)+b
    ...:     if tmp <= 0:
    ...:         return 0
    ...:     else :
    ...:         return 1
```

여기에서 $-\theta$ 가 편향 b로 치환됐다. 그리고 편향은 가중치 w_1, w_2와 기능이 다르다는 사실에 주의하자. 구체적으로 말하면 w_1과 w_2 는 각 입력 신호가 결과에 주는 영향력(중요도)을 조절하는 매개변수고, 편향은 뉴런이 얼마나 쉽게 활성화(결과로 1을 출력)하느냐를 조정하는 매개변수이다.

예를 들어 b가 -0.1이면 각 입력 신호에 가중치를 곱한 값들의 합이 0.1을 초과할 때만 뉴런이 활성화한다. 반면 b가 -20이면 각 입력 신호에 가중치를 곱한 값들의 합이 20.0을 넘지 않으면 뉴런은 활성화하지 않는다. 이처럼 편향의 값은 뉴런이 얼마나 쉽게 활성화되는지를 결정한다. 한편 이 책에서는 w_1, w_2, b 셋 다 '가중치'라고 할 때도 있다.

이어서 NAND 게이트와 OR 게이트를 구현해보자.

```
In [14]: def NAND(x1,x2):
    ...:     x = np.array([x1,x2])
    ...:     w = np.array([-0.5,-0.5])
    ...:     b=0.7
    ...:     tmp = np.sum(w*x)+b
    ...:     if tmp<=0:
    ...:         return 0
    ...:     else:
    ...:         return 1
    ...:

In [15]: def OR(x1,x2):
    ...:     x = np.array([x1,x2])
    ...:     w = np.array([0.5,0.5])
    ...:     b=-0.2
    ...:     tmp = np.sum(w*x)+b
    ...:     if tmp<=0:
    ...:         return 0
    ...:     else:
    ...:         return 1
```

앞 절에서 AND, NAND, OR는 모두 같은 구조의 퍼셉트론이고, 차이는 가중치 매개변수의 값뿐이라 했다. 실제로 파이썬으로 작성한 NAND와 OR 게이트의 코드에서도 AND와 다른 곳은 가중치와 편향 값을 설정하는 부분뿐이다.

2.4 퍼셉트론의 한계

지금까지 살펴본 것 처럼 퍼셉트론을 이용하면 AND, NAND, OR의 3가지 논리 회로를 구현할 수 있었다. XOR게이트를 생각해보자.

2.4.1 XOR게이트

XOR게으트는 **베타적 논리합**이라는 논리 회로이다. 다음 표와 같이 x_1과 x_2 중 한쪽이 1일 때만 1을을 출력한다. ('배타적'이란 자기 외에는 거부한다는 의미이다.) 자 이 XOR 게이트를 퍼셉트론으로 구현하려면 가중치 매개변수 값을 어떻게 설정하면 될까?

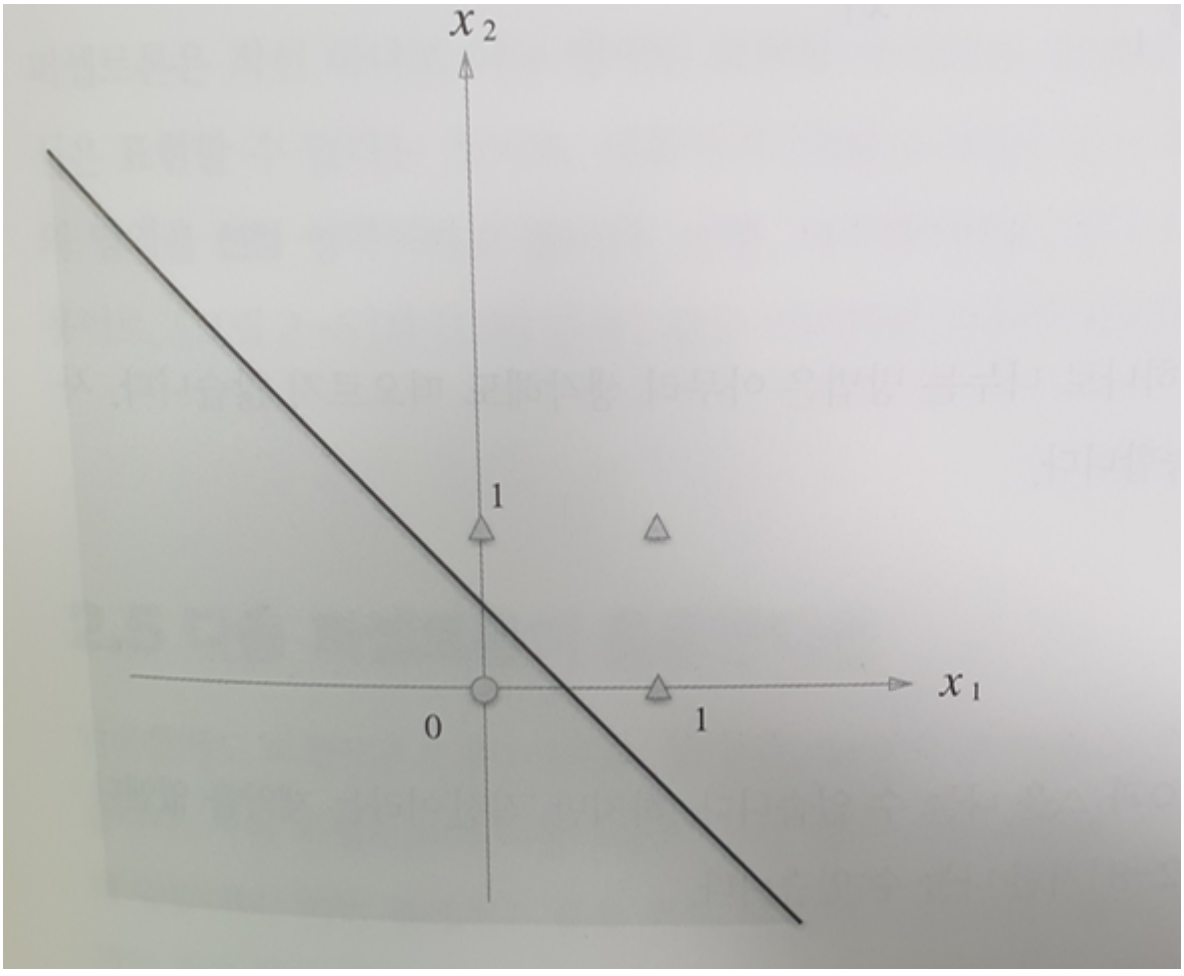
x_1	x_2	y
0	0	0
1	0	1
0	1	1
1	1	0

사실 지금까지 본 퍼셉트론으로는 이 XOR 게이트를 구현할 수 없다. 왜 AND와 OR은 되고 XOR은 안될까? 그림으로 설명하겠다.

우선 OR게이트의 동작을 시각적으로 생각해보자. OR 게이트는 예를 들어 가중치 매개변수가 (b, w_1, w_2) = (-0.5,1.0,1.0) 일 때 OR의 진리표를 만족한다. 이때의 퍼셉트론은 다음과 같이 표현된다.

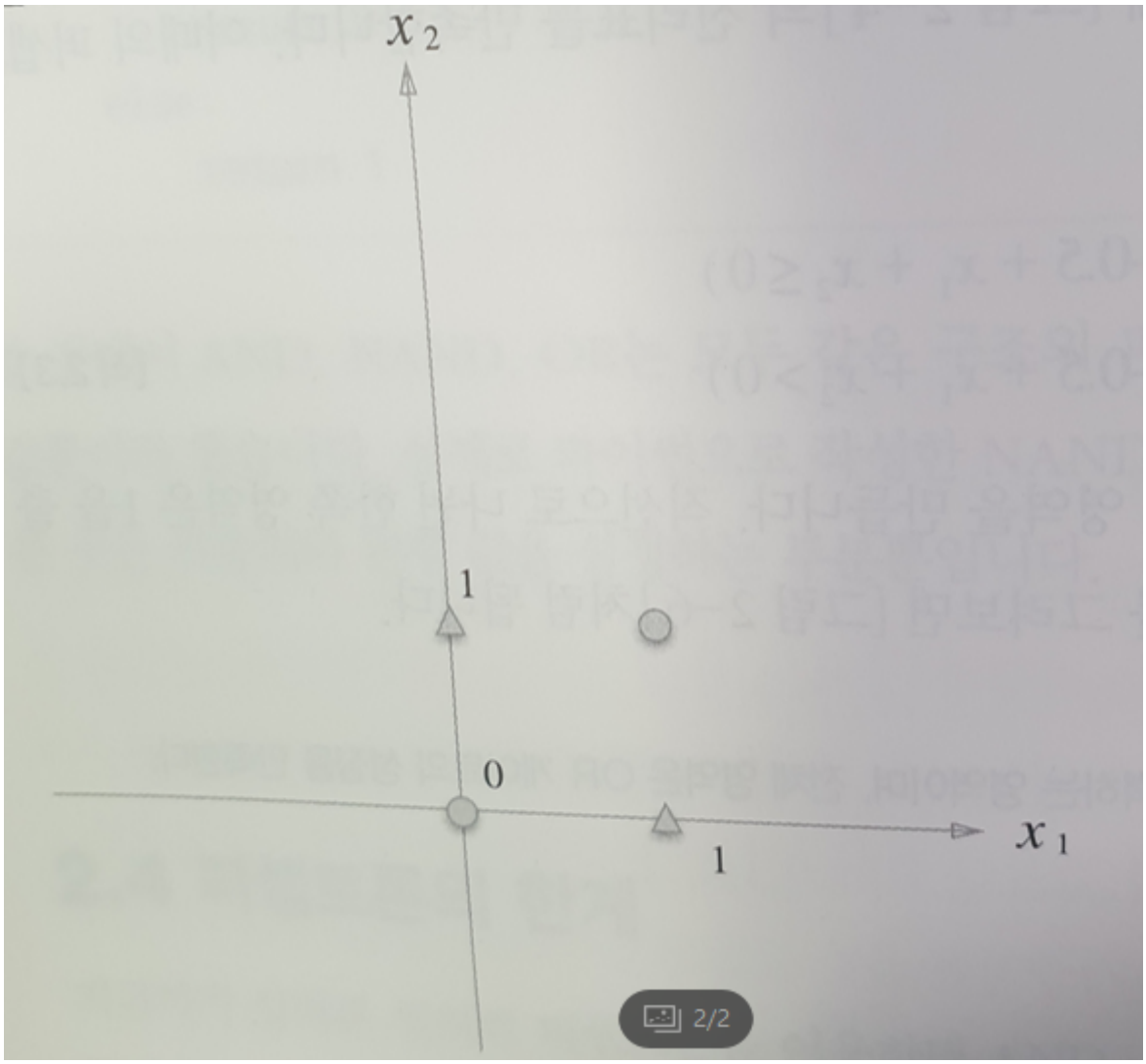
$$y = 0(-0.5 + x_1 + x_2 \leq 0)$$
$$1(-0.5 + x_1 + x_2 > 0)$$

위 식의 퍼셉트론은 직선으로 나뉜 두 영역을 만든다. 직선으로 나뉜 한쪽 영역은 1을 출력하고 다른 한쪽은 0을 출력한다. 이를 그려보면 다음과 같다.



OR 게이트는 (0,0) 일 때 0을 출력하고 나머지는 다 1을 출력한다.

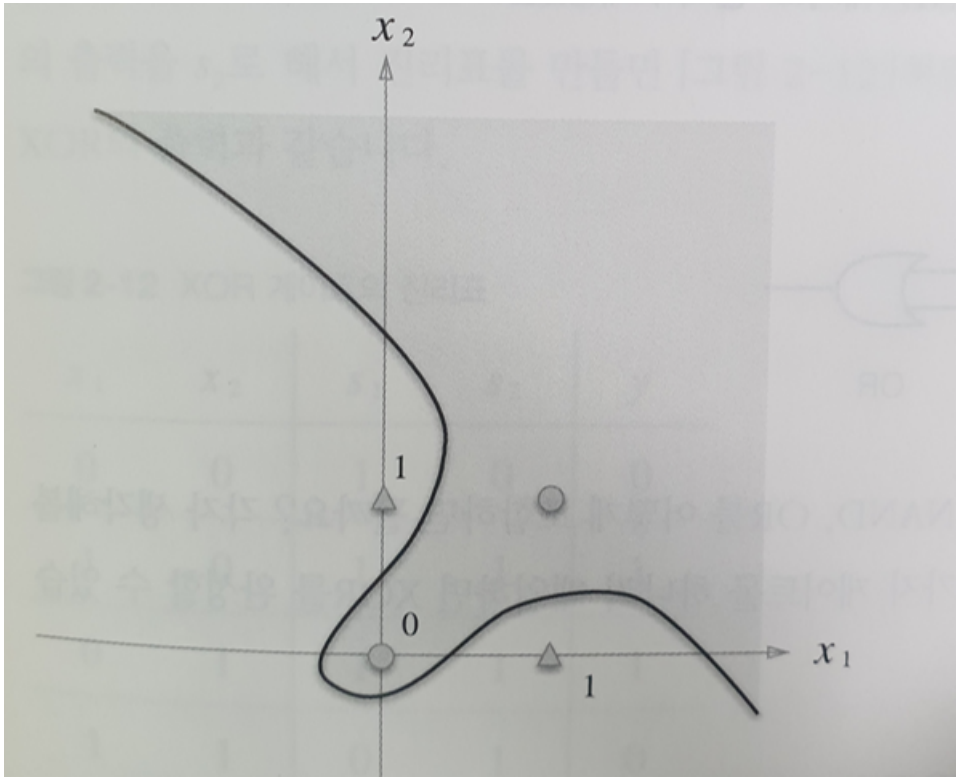
그럼 XOR 게이트의 경우는 어떨까? OR 게이트 때 처럼 직선 하나로 0과 1을 나누는 영역을 만들어낼 수 있을까?



사실 직선 하나로 나누기란 불가능하다.

2.4.2 선형과 비선형

직선 하나로는 0과 1을 나눌 수 없다. 하지만 '직선' 이라는 제약을 없앤다면 가능하다. 예를 들어 다음처럼 해보자.



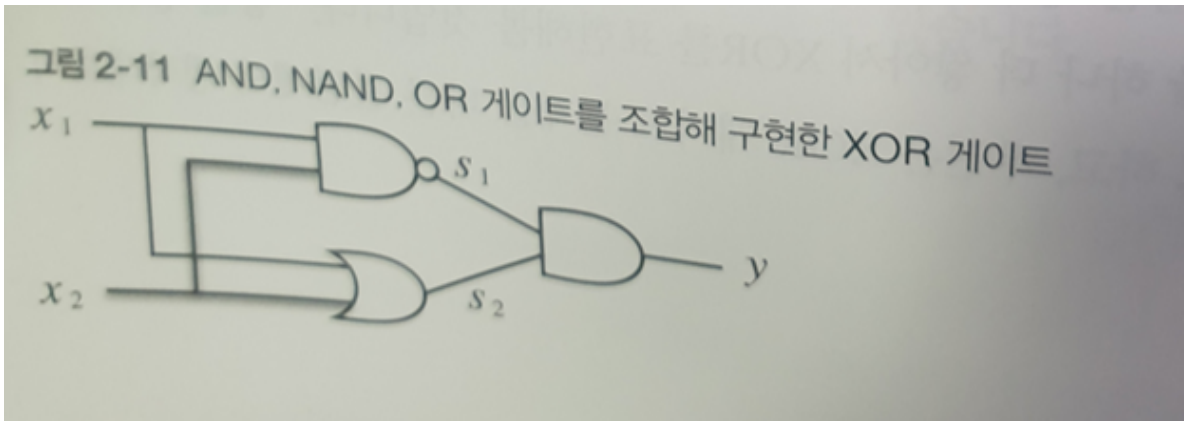
퍼셉트론은 직선 하나로 나눈 영역만 표현할 수 있다는 한계가 있다. 위와 같은 곡선은 표현할 수 없다는 것이다. 덧붙여서 위와 같은 곡선의 영역을 **비선형** 영역, 직선의 영역을 **선형** 영역이라고 한다. 선형, 비선형이라는 말은 기계학습 분야에서 자주 쓰이는 용어로 위와 OR 게이트 설명시 사용했던 그림을 떠올리면 된다.

2.5 다층 퍼셉트론이 충돌한다면

안타깝게도 퍼셉트론으로는 XOR게이트를 표현할 수 없었다. 그렇다고 슬퍼할 필요는 없다. 사실 퍼셉트론의 아름다움은 '층을 쌓아' **다층 퍼셉트론**을 만들 수 있다는 데 있다. 이번 절에서는 층을 하나 더 쌓아서 XOR를 표현해볼 것이다. ''층을 쌓는다''는 말의 뜻은 잠시 뒤에 살펴보기로 하고, 우선은 XOR 게이트 문제를 다른 관점에서 생각해보기로 하자.

2.5.1 기존 게이트 조합하기

XOR 게이트를 만드는 방법은 다양하다. 그중 하나는 앞서 만든 AND, NAND, OR 게이트를 조합하는 방법이다. 여기서는 AND, NAND, OR 게이트를 다음과 같은 기호로 하며 이를 조합해서 XOR게이트를 구현하자.



진리표는 다음과 같다.

x_1	x_2	s_1	s_2	y
0	0	1	0	0
1	0	1	1	1
0	1	1	1	1
1	1	0	1	0

2.5.2 XOR 게이트 구현하기

이어서 XOR 게이트를 파이썬으로 구현하고자 한다. 지금까지 정의한 함수 AND, NAND, OR 를 사용하면 다음과 같이 구현할 수 있다.

```
In [16]: def XOR(x1,x2):
    ...:     s1 = NAND(x1,x2)
    ...:     s2 = OR(x1,x2)
    ...:     y = AND(s1,s2)
    ...:     return y
    ...:
```

이 XOR 함수는 기대한 대로의 결과를 출력한다.

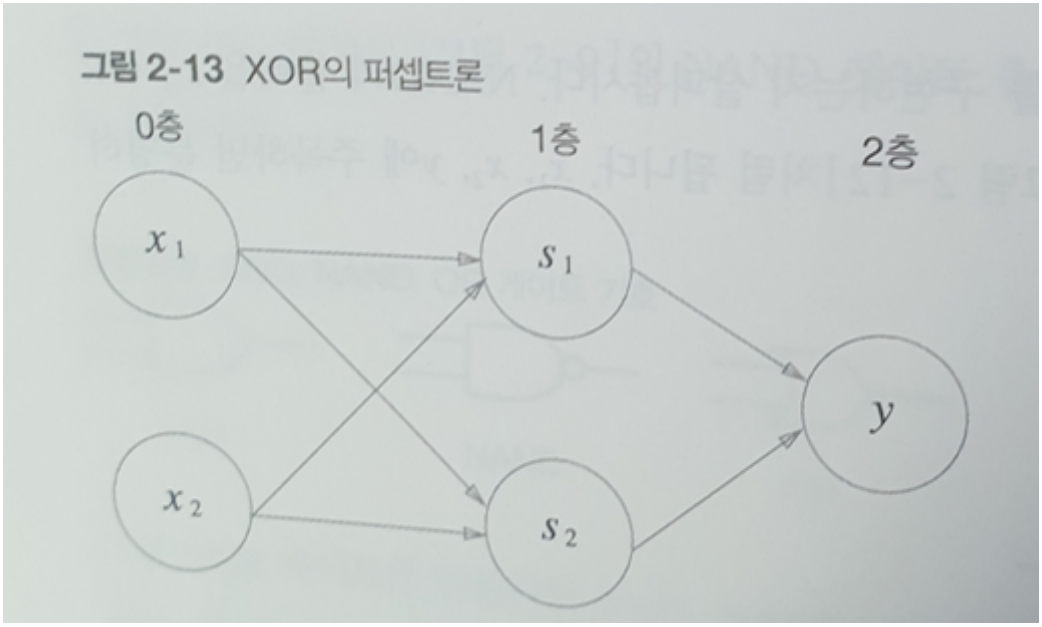
```
In [17]: XOR(0,0)
Out[17]: 0

In [18]: XOR(0,1)
Out[18]: 1

In [19]: XOR(1,0)
Out[19]: 1

In [20]: XOR(0,0)
Out[20]: 0
```

이로써 XOR 게이트를 완성했다. 지금 구현한 XOR를 뉴런을 이용한 퍼셉트론으로 표현하면 다음 그림처럼 된다.



XOR은 위 그림과 같은 다층 구조의 네트워크이다. 이 책에서는 왼쪽 부터 차례로 0층, 1층, 2층이라고 하겠다.

그런데 위 그림의 퍼셉트론은 지금까지 본 AND, OR 퍼셉트론과 형태가 다르다. 실제로 AND, OR가 단층 퍼셉트론인데 반해, XOR은 2층 퍼셉트론이다. 이처럼 층이 여러 개인 퍼셉트론을 다층 퍼셉트론이라고 한다.

WARNING_ 위 그림의 퍼셉트론은 모두 3층으로 구성됐으나, 가중치를 갖는 층은 사실 2개(0층과 1층 사이, 1층과 2층 사이)뿐이니 '2층 퍼셉트론'이라 부르기로 하자. 문헌에 따라서는 구성 층의 수를 기준으로 "2층 퍼셉트론'이라고 하는 경우도 있다.

위 그림과 같은 2층 퍼셉트론에서는 0층에서 1층으로 신호가 전달되고, 이어서 1층에서 2층으로 신호가 전달된다. 이 동작을 더 자세히 서술하면 다음과 같다.

1. 0층의 두 뉴런이 입력 신호를 받아 1층 뉴런으로 신호를 보낸다.
2. 1층의 뉴런이 2층의 뉴런으로 신호를 보내고, 2층의 뉴런은 y를 출력한다.

덧붙여서, 이 2층 퍼셉트론의 동작을 공장의 조립라인에 비유할 수 있다.

이상으로 2층 구조를 사용해 퍼셉트론으로 XOR 게이트를 구현할 수 있다. 다시 말해 **단층 퍼셉트론으로는 표현하지 못한 것을 층을 하나 늘려 구현** 할 수 있었다. 이처럼 퍼셉트론은 층을 쌓아 (깊게 하여) 더 다양한 것을 표현할 수 있다.

2.6 NAND에서 컴퓨터까지

다층 퍼셉트론은 지금까지 봐온 회로보타 복잡한 회로를 만들 수 있다. 예를 들면, 덧셈을 처리하는 가산기도 만들 수 있다. 2진수를 10진수로 변환하는 인코더, 어떤 조건을 충족하면 1을 출력하는 회로(패리티 검사 회로)도 페셉트론으로 표현할 수 있다. 사실은 퍼셉트론을 이용하면 '컴퓨터' 마저 표현할 수 있다.

컴퓨터는 정보를 처리하는 기계이다. 컴퓨터에 뭔가를 입력하면 정해진 방법으로 처리하고 그 결과를 출력한다. 정해진 방법으로 처리한다는 것은 컴퓨터도 마치 퍼셉트론처럼 입력과 출력으로 구성된 특정 규칙대로 계산을 수행한다는 뜻이다.

컴퓨터 내부에서 이뤄지는 처리가 매우 복잡할 것 같지만, 사실은 (놀랍게도) NAND 게이트의 조합만으로 컴퓨터가 수행하는 일을 재현할 수 있다. NAND게이트만으로 컴퓨터를 만들 수 있다? 이 말은 곧 퍼셉트론으로도 컴퓨터를 표현할 수 있다는 놀라운 사실로 이어진다. 지금까지 살펴본 것 처럼 NAND 게이트는 퍼셉트론으로 만들 수 있기 때문이다.

그렇다면 층을 얼마나 깊게 하면 컴퓨터를 만들 수 있을까? 그 답은 "이론상 2층 퍼셉트론이면 컴퓨터를 만들 수 있다"이다. 말도 안 되는 소리 같지만, 2층 퍼셉트론, 정확히는 비선형인 시그모이드 함수를 활성화 함수로 이용하면 임의의 함수를 표현할 수 있다는 사실이 증명됐다. 그러나 2층 퍼셉트론 구조에서 가중치를 적절히 설정하여 컴퓨터를 만들기란 너무 어렵다. 실제로도 NAND 등의 저수준 소자에서 시작하여 컴퓨터를 만드는 데 필요한 부품(모듈)을 단계적으로 만들어가는 쪽이 자연스러운 방법이다.

즉, 처음에는 AND와 OR 게 ㅐ 이트, 그다음에는 반가산기와 전가산기, 그 다음에는 산술 논리 연산 장치(ALU), 그 다음에 CPU라는 식이다. 그래서 퍼셉트론으로 표현하는 컴퓨터도 여러 층을 다시 층층이 겹친 구조로 만드는 방향이 자연스러운 흐름이다.

이 책에서 컴퓨터는 만들지 않는다. 그래도 퍼셉트론은 층을 거듭 쌓으면 비선형적인 표현도 가능하고, 이론상 컴퓨터가 수행하는 처리도 모두 표현할 수 있다는 점을 기억해둬라

2.7 정리

이번 장에서 퍼셉트론을 배웠다. 퍼셉트론은 간단한 알고리즘이라 그 구조를 쉽게 이해할 수 있다. 퍼셉트론은 다음 장에서 배울 신경망의 기초가 된다. 그러니 이번 장에서 배운 내용은 아주 중요하다.

- 퍼셉트론은 입출력을 갖춘 알고리즘이다. 입력을 주면 정해진 규칙에 따른 값을 출력한다.
- 퍼셉트론에서는 '가중치'와 '편향' 을 매개변수로 설정한다.
- 퍼셉트론으로 AND, OR 게이트 등의 논리 회로를 표현할 수 있다 .
- XOR 게이트는 단층 퍼셉트론으로는 표현할 수 없다.
- 2층 퍼셉트론을 이용하면 XOR 게이트를 표현할 수 있다.
- 단층 퍼셉트론은 직선형 영역만 표현할 수 있고, 다층 퍼셉트론은 비선형 영역도 표현할 수 있다.
- 다층 퍼셉트론은 (이론상) 컴퓨터를 표현할 수 있다.