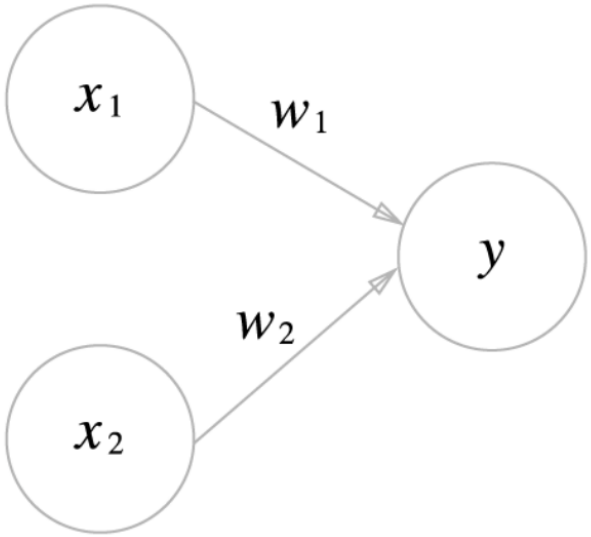
2장. 퍼셉트론

1957년(!) 프랑크 로젠블라크가 고안한 개념이다. (그랬구나!)

퍼셋트론 == 인공 뉴런 == 인공 신경 세포



- 신호 2개(x_1. x_2)를 받아서
- 가중치들(w_1, w_2)을 각각 곱해서
- 특정한 임계값($theta$)를 넘는 경우에는 1, 아니면 0을 출력신호로(ㄹ) 내보낸다. ($theta$ 는 그림에 없음)

In [1]:
```
import numpy as np
```

In [3]:
```
# 각 논리회로는 진리표 대로 나오게 하면 되는 것. (AND는 TT 일때 T)

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
```

```
print(AND(0,0), AND(0,1), AND(1,0), AND(1,1))
```

```
0 0 0 1
```

In [5]:
```
# 넘피 배열로 만들어도 동작한다.

def AND(x1, x2):
    x = np.array([x1, x2])
    w = np.array([0.5, 0.5])
    b = -0.7
    tmp = np.sum(x*w) + b
    return int(tmp > 0)

print(AND(0,0), AND(0,1), AND(1,0), AND(1,1))


# OR 이나 NAND도 비슷하게 구현할 수 있다.
def NAND(x1, x2):
    x = np.array([x1, x2])
    w = np.array([-0.5, -0.5])
    b = 0.7
    tmp = np.sum(x*w) + b
    return int(tmp > 0)

print(NAND(0,0), NAND(0,1), NAND(1,0), NAND(1,1))
```

```
0 0 0 1
1 1 1 0
```

In [6]:
```
def OR(x1, x2):
    x = np.array([x1, x2])
    w = np.array([0.5, 0.5])
    b = -0.2
    tmp = np.sum(x*w) + b
    return int(tmp > 0)

print(OR(0,0), OR(0,1), OR(1,0), OR(1,1))
```

```
0 1 1 1
```

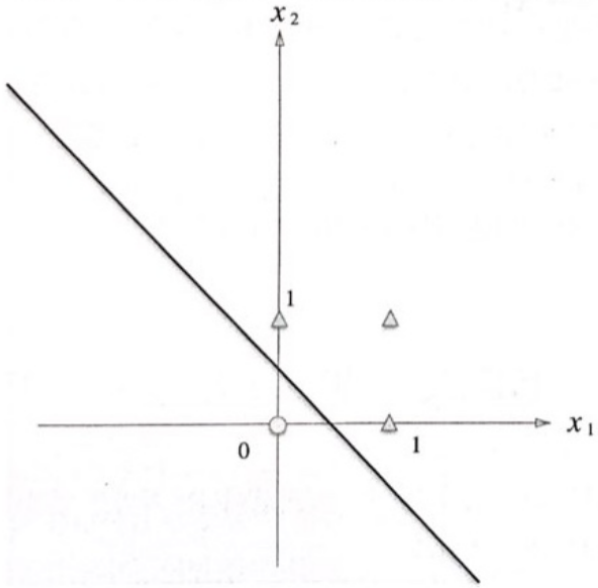
OR 게이트

위에서 만든 `OR` 게이트를 수식과 그림으로 보면 이렇다.

$$y = \begin{cases} 0 & (-0.5 + x_1 + x_2 \leq 0) \\ 1 & (-0.5 + x_1 + x_2 > 0) \end{cases} \quad \text{[식 2.3]}$$

[식 2.3]의 퍼셉트론은 직선으로 나뉜 두 영역을 만듭니다. 직선으로 나뉜 한쪽 영역은 1을 출력하고 다른 한쪽은 0을 출력합니다. 이를 그려보면 [그림 2-6]처럼 됩니다.

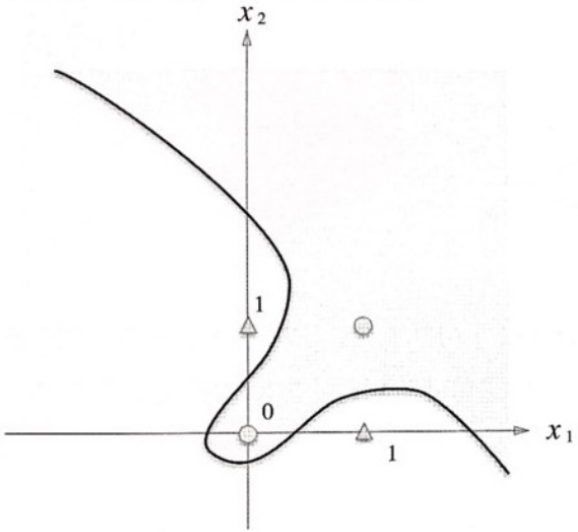
그림 2-6 퍼셉트론의 시각화 : 회색 영역은 0을 출력하는 영역이며, 전체 영역은 OR 게이트의 성질을 만족한다.



XOR 게이트

문제는 `XOR` 게이트.

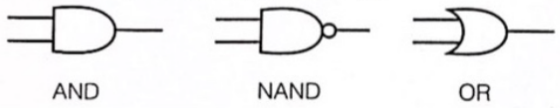
그림 2-8 곡선이라면 ○과 △을 나눌 수 있다.



퍼셉트론은 직선 하나로 나눈 영역만 표현할 수 있다는 한계가 있습니다. [그림 2-8] 같은 곡선은 표현할 수 없다는 것이죠. 덧붙여서 [그림 2-8]과 같은 곡선의 영역을 **비선형** 영역, 직선의 영역을 **선형** 영역이라고 합니다. 선형, 비선형이라는 말은 기계학습 분야에서 자주 쓰이는 용어로, [그림 2-6]과 [그림 2-8] 같은 이미지를 떠올리시면 됩니다.

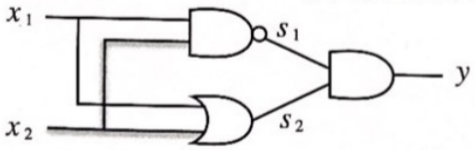
논리회로를 여러개 쌓으면 되긴 된다. (XOR함수안에서 NAND, OR, AND를 부르면 된다.)

그림 2-9 AND, NAND, OR 게이트 기호



[그림 2-11]과 같은 조합이라면 XOR 게이트를 구현할 수 있습니다. x_1과 x_2가 입력 신호, y가 출력 신호입니다. x_1과 x_2는 NAND와 OR 게이트의 입력이 되고, NAND와 OR의 출력이 AND 게이트의 입력으로 이어집니다.

그림 2-11 AND, NAND, OR 게이트를 조합해 구현한 XOR 게이트



In [5]:
```
def XOR(x1, x2):
    s1 = NAND(x1, x2)
    s2 = OR(x1, x2)
    y = AND(s1, s2)
    return y

print(XOR(0,0), XOR(0,1), XOR(1,0), XOR(1,1))
```

```
0 1 1 0
```

즉, 단층 퍼셉트론으로는 못만든다.

한층이 더 필요하다.

참고 : NAND 의 조합만으로 컴퓨터를 만들 수 있다. ('NAND에서 테트리스까지' 라는 책이 있다)

In [ ]: