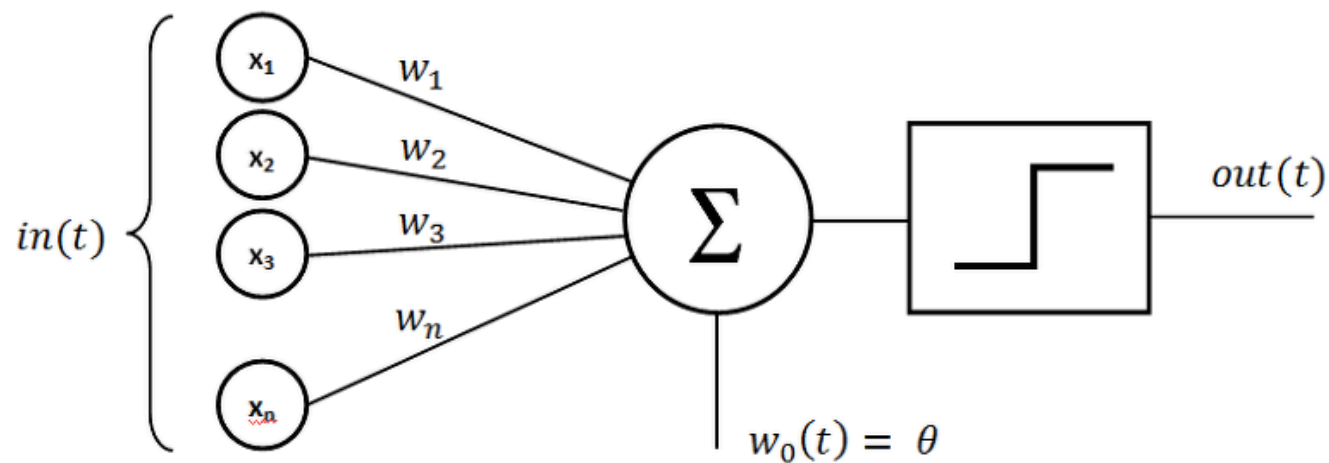


CHAPTER 2

퍼셉트론



2.1 퍼셉트론이란

- 1957년 프랑크 로젠블라트(Frank Rosenblatt)가 고안한 알고리즘
- 퍼셉트론은 **신경망의 기원**이 되는 알고리즘
- 퍼셉트론의 구조를 배우는 것은 신경망과 딥러닝으로 나아가는데 중요한 아이디어를 배우는 일



Frank Rosenblatt
(1928-1971)

2.1 퍼셉트론이란

- 다수의 신호를 입력으로 받아 하나의 신호를 출력하는 구조
- 출력 신호는 1 또는 0으로 표현
 - 1 = 신호가 흐른다
 - 0 = 신호가 흐르지 않는다
- x (입력 신호), y (출력 신호),
 w (가중치), 원(뉴런 또는 노드),
 θ (임계값)

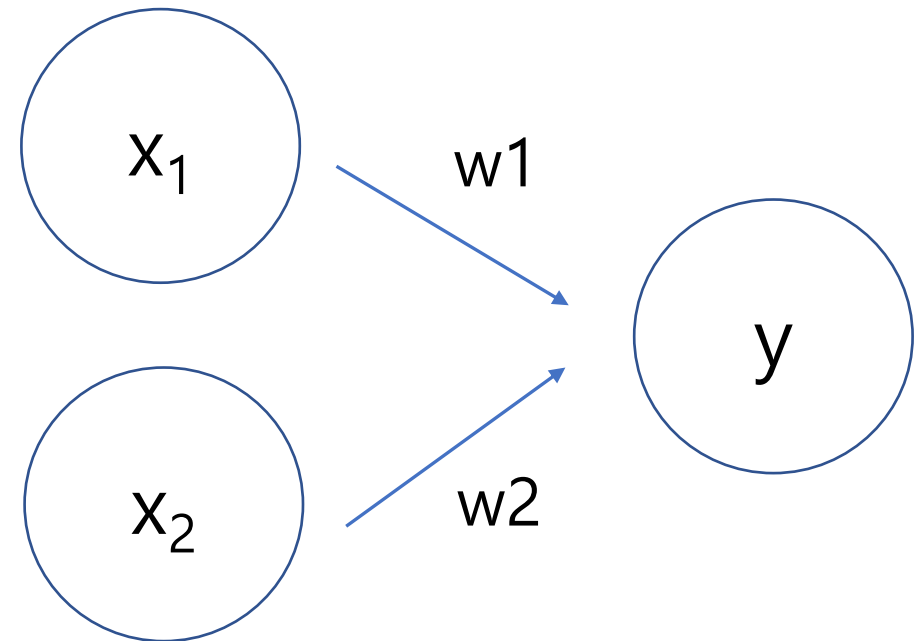
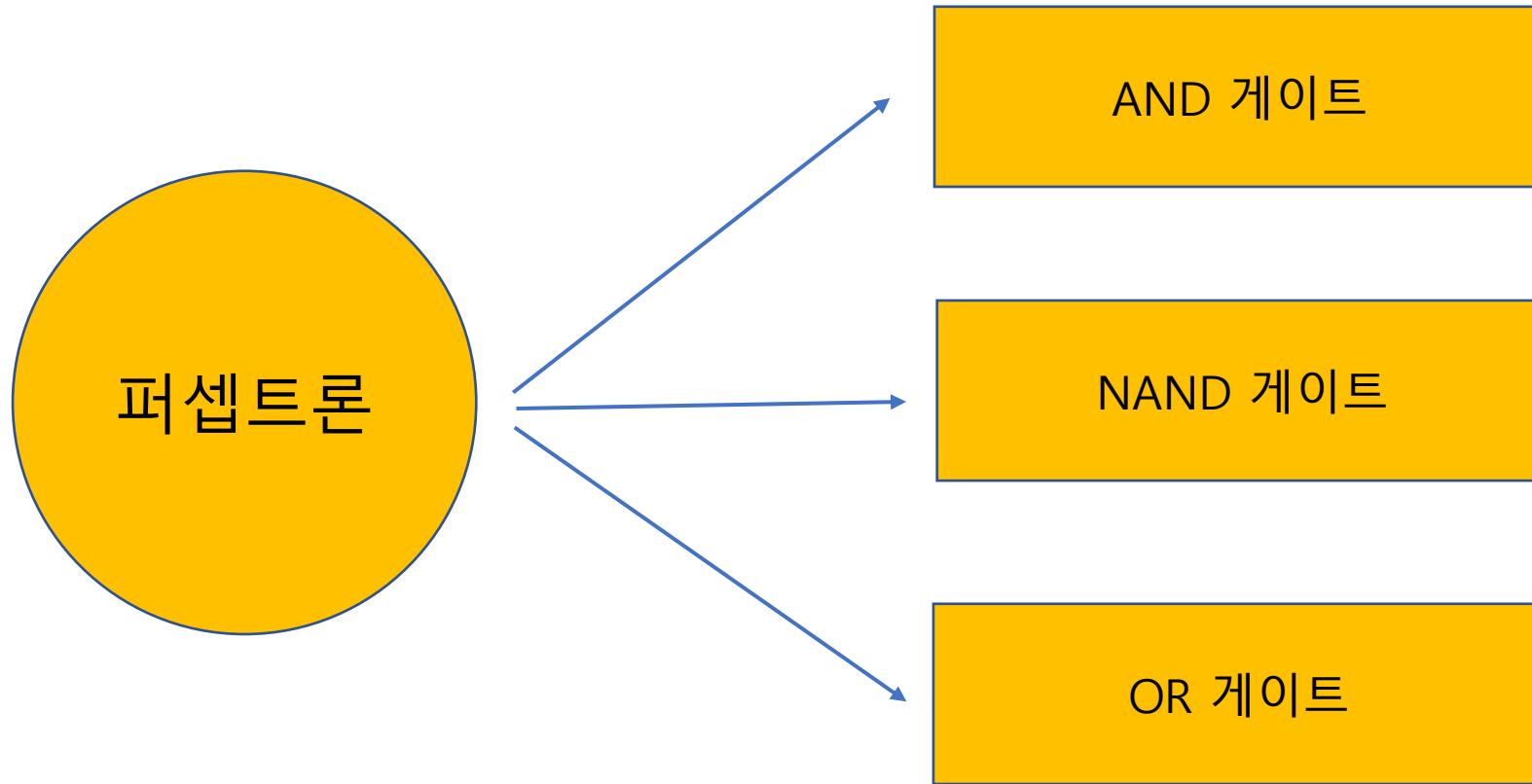


그림 2-1 입력이 2개인 퍼셉트론

$$y = \begin{cases} 0 & (w_1x_1 + w_2x_2 \leq \theta) \\ 1 & (w_1x_1 + w_2x_2 > \theta) \end{cases}$$

식 2.1 퍼셉트론의 구조

2.2 단순한 논리 회로



2.2.1 AND 게이트

두 입력이 모두 1일 때만 1을 출력,
그 외에는 0을 출력

그림 2-2를 만족하기 위해서
 $(\omega_1, \omega_2, \theta) = (0.5, 0.5, 0.7)$ or
 $(0.5, 0.5, 0.8), (1.0, 1.0, 1.0)$ 등 가능

Ex)

$$1 \times 0.5 + 0 \times 0.5 = 0.5 \leq 0.7(\theta) \rightarrow 0 \text{ 출력}$$

$$1 \times 0.5 + 1 \times 0.5 = 1.0 > 0.7(\theta) \rightarrow 1 \text{ 출력}$$

그림 2-2 AND 게이트의 진리표

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

$$y = \begin{cases} 0 & (w_1x_1 + w_2x_2 \leq \theta) \\ 1 & (w_1x_1 + w_2x_2 > \theta) \end{cases}$$

2.2.2 NAND 게이트 (Not AND)

두 입력이 모두 1일 때만 0을 출력,
그 외에는 1을 출력

그림 2-3를 만족하기 위해서
 $(\omega_1, \omega_2, \theta) = (-0.5, -0.5, -0.7)$ 조합 사용
→ AND 게이트를 구현하는 매개변수의 부호를
모두 반전한 구조

Ex)

$$1 \times -0.5 + 0 \times -0.5 = -0.5 > -0.7(\theta) \rightarrow 1 \text{ 출력}$$

$$1 \times -0.5 + 1 \times -0.5 = -1.0 \leq -0.7(\theta) \rightarrow 0 \text{ 출력}$$

그림 2-3 NAND 게이트의 진리표

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

2.2.2 OR 게이트

입력 신호 중 **하나 이상이 1이면** 출력이 1이 되는 논리 회로

그림 2-4를 만족하기 위해서
 $(\omega_1, \omega_2, \theta) = (-0.5, 1.0, 0.4)$ 조합 사용

Ex)

$$0 \times -0.5 + 0 \times 1.0 = 0 \leq 0.4(\theta) \rightarrow 0 \text{ 출력}$$

$$1 \times -0.5 + 1 \times 1.0 = 0.5 > 0.4(\theta) \rightarrow 1 \text{ 출력}$$

그림 2-4 OR 게이트의 진리표

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

2.2.2 OR 게이트

- 퍼셉트론으로 AND, NAND, OR 논리 회로를 표현할 수 있다.
- 여기서 중요한 점은 퍼셉트론 구조는 AND, NAND, OR 게이트 모두 똑같다.
- 세 가지 게이트에서 다른 것은 매개변수의 값 뿐이다.
- 따라서 매개변수의 값만 적절히 조정하면 AND, NAND, OR 게이트를 만들 수 있다.

2.3 퍼셉트론 구현하기(AND)

jupyter Untitled Last Checkpoint: a minute ago (unsaved changes)



Logout

File Edit View Insert Cell Kernel Widgets Help

Trusted

Python 3

Run Code

```
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
```

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

```
Out[2]: 0
```

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

```
Out[3]: 0
```

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

```
Out[4]: 0
```

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

```
Out[5]: 1
```

2.3 퍼셉트론 구현하기(AND)

jupyter Untitled Last Checkpoint: a minute ago (unsaved changes)



Logout

File Edit View Insert Cell Kernel Widgets Help

Trusted

Python 3

Run Code

```
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
```

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

```
Out[2]: 0
```

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

```
Out[3]: 0
```

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

```
Out[4]: 0
```

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

```
Out[5]: 1
```

그림 2-2 AND 게이트의 진리표

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

2.3.2 가중치와 편향 도입

앞에서 구현한 AND 게이트는
직관적이고 알기 쉽지만
다음 step을 위하여 $\theta = -b$ 로 치환!
 b 는 편향이라 한다.

$$y = \begin{cases} 0 & (w_1x_1 + w_2x_2 \leq \theta) \\ 1 & (w_1x_1 + w_2x_2 > \theta) \end{cases}$$



$$y = \begin{cases} 0 & (b + w_1x_1 + w_2x_2 \leq 0) \\ 1 & (b + w_1x_1 + w_2x_2 > 0) \end{cases}$$

The screenshot shows a Jupyter Notebook titled 'Untitled (unsaved changes)' with a menu bar (File, Edit, View, Insert, Cell, Kernel, Widgets, Help) and a toolbar. The notebook contains five code cells:

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

```
In [2]: x = np.array([0,1]) #입력신호
w = np.array([0.5,0.5]) #가중치
b = -0.7 #편향
```

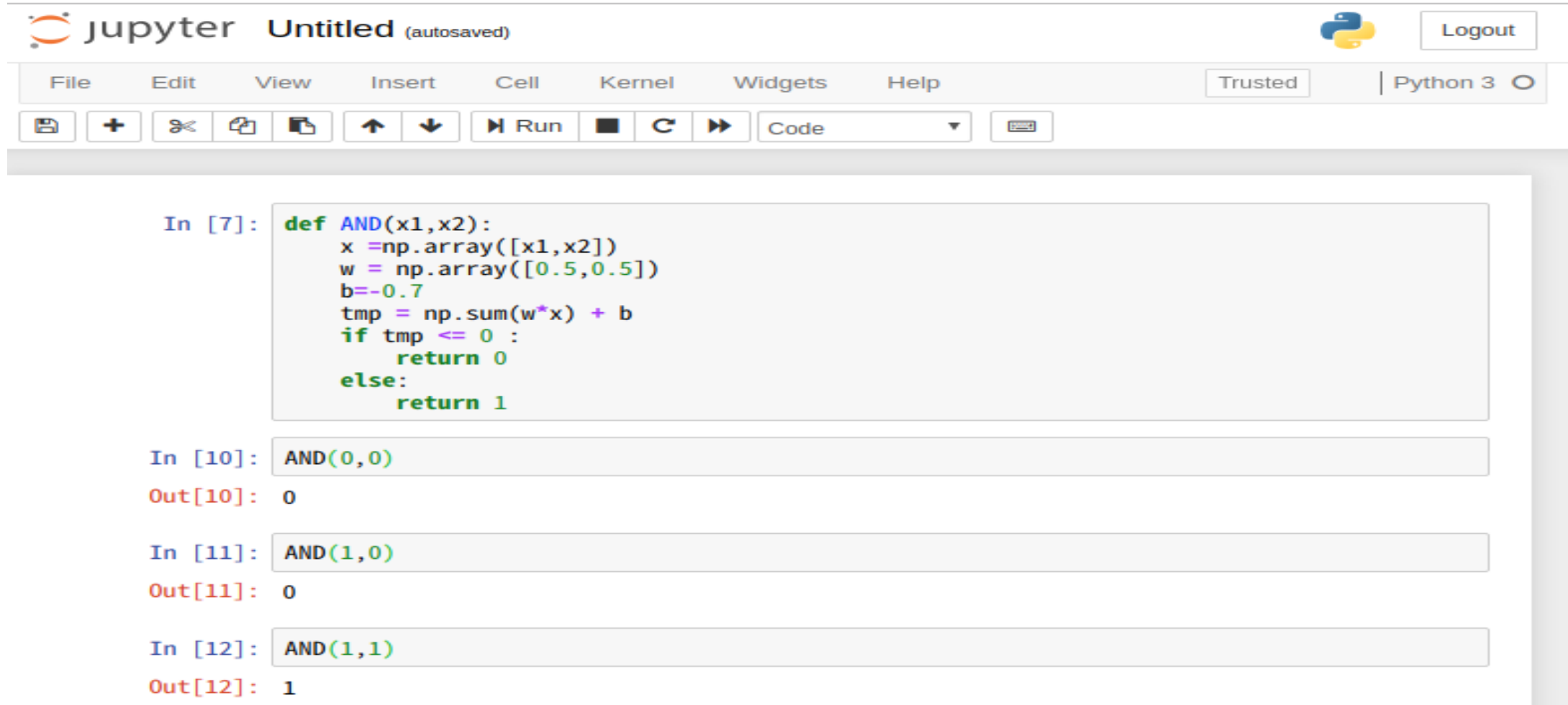
```
In [3]: print(w*x)
[0.  0.5]
```

```
In [4]: print(np.sum(w*x))
0.5
```

```
In [5]: print(np.sum(w*x)+b)
-0.19999999999999996
```

대략 -0.2 (부동소수점 수에 의한 연산 오차)

2.3.3 가중치와 편향 도입하여 구현(AND)



The image shows a Jupyter Notebook interface with the title "Untitled (autosaved)". The top bar includes a "Logout" button and a "Python 3" indicator. The menu bar contains "File", "Edit", "View", "Insert", "Cell", "Kernel", "Widgets", and "Help". The toolbar includes icons for saving, adding cells, undo, redo, and running code. The main area displays the following code and outputs:

```
In [7]: 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
```

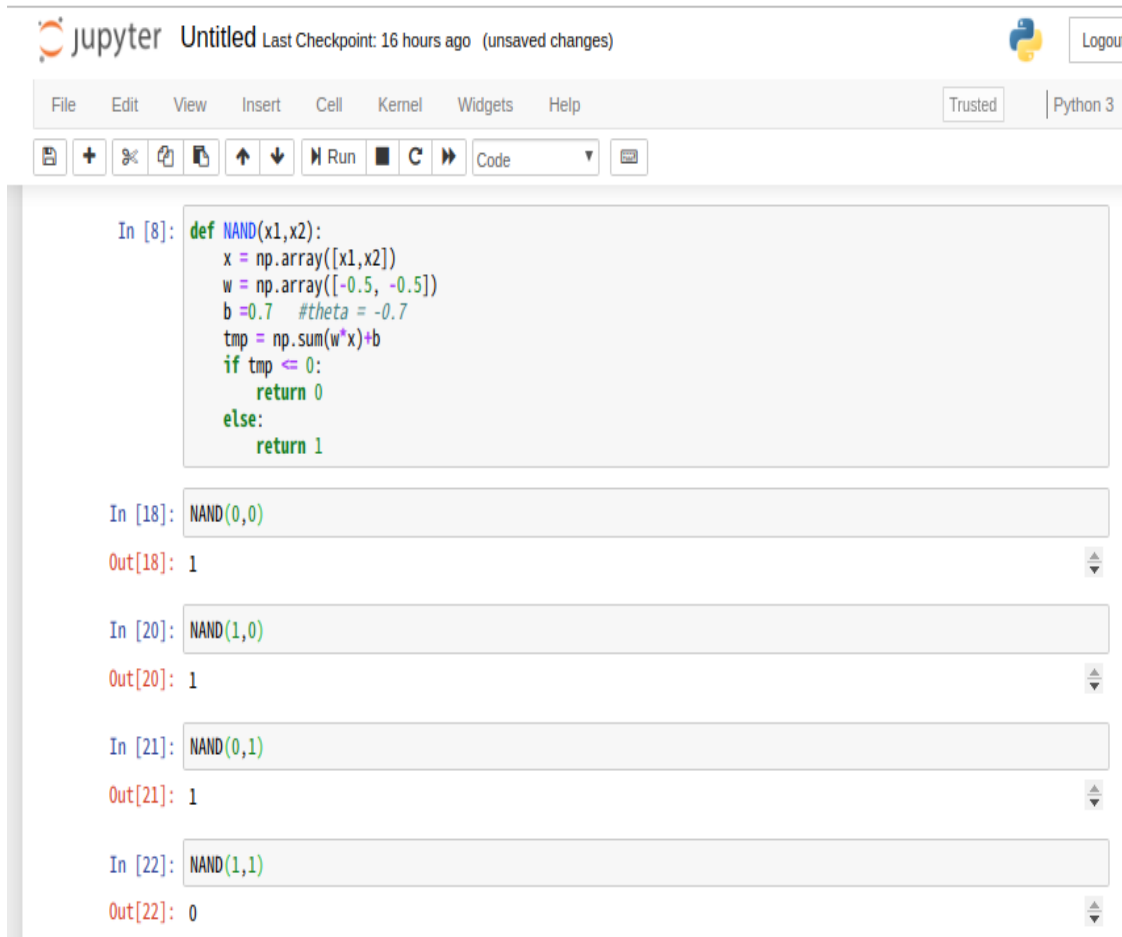
```
In [10]: AND(0,0)  
Out[10]: 0
```

```
In [11]: AND(1,0)  
Out[11]: 0
```

```
In [12]: AND(1,1)  
Out[12]: 1
```

2.3.3 가중치와 편향 도입하여 구현하기

<NAND>



The Jupyter Notebook interface shows the implementation of a NAND function. The code defines a function `NAND(x1, x2)` that takes two inputs, calculates a weighted sum with bias, and returns 0 if the sum is less than or equal to 0, otherwise returns 1. Below the code, four input-output pairs are shown, demonstrating the function's behavior for different combinations of 0 and 1 inputs.

```
In [8]: def NAND(x1,x2):  
        x = np.array([x1,x2])  
        w = np.array([-0.5, -0.5])  
        b = 0.7 #theta = -0.7  
        tmp = np.sum(w*x)+b  
        if tmp <= 0:  
            return 0  
        else:  
            return 1
```

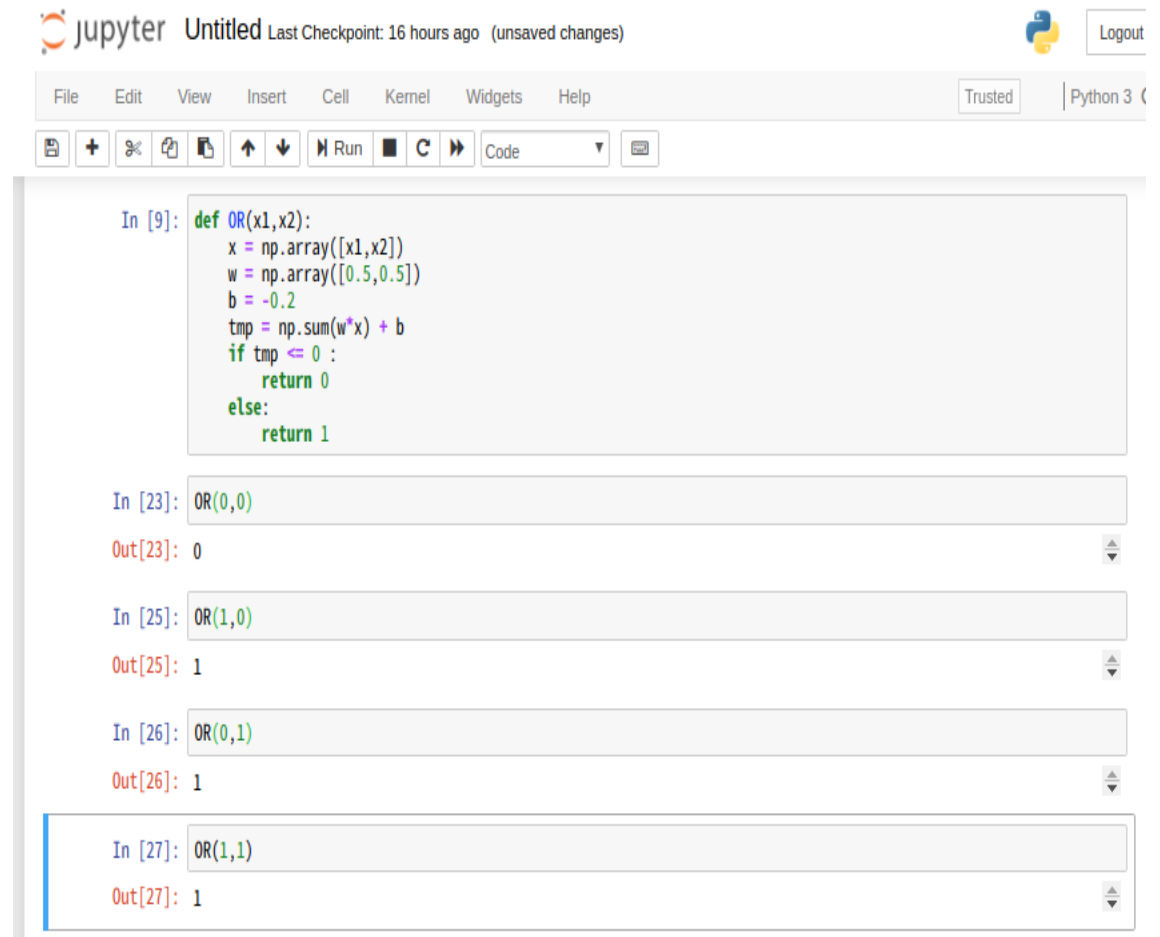
```
In [18]: NAND(0,0)  
Out[18]: 1
```

```
In [20]: NAND(1,0)  
Out[20]: 1
```

```
In [21]: NAND(0,1)  
Out[21]: 1
```

```
In [22]: NAND(1,1)  
Out[22]: 0
```

<OR>



The Jupyter Notebook interface shows the implementation of an OR function. The code defines a function `OR(x1, x2)` that takes two inputs, calculates a weighted sum with bias, and returns 0 if the sum is less than or equal to 0, otherwise returns 1. Below the code, four input-output pairs are shown, demonstrating the function's behavior for different combinations of 0 and 1 inputs.

```
In [9]: 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
```

```
In [23]: OR(0,0)  
Out[23]: 0
```

```
In [25]: OR(1,0)  
Out[25]: 1
```

```
In [26]: OR(0,1)  
Out[26]: 1
```

```
In [27]: OR(1,1)  
Out[27]: 1
```

2.3.3 가중치와 편향 도입하여 구현하기

<NAND>

Jupyter Untitled Last Checkpoint: 16 hours ago (unsaved changes)

File Edit View Insert Cell Kernel Widgets Help Trusted Python 3

```
In [8]: def NAND(x1,x2):
        x = np.array([x1,x2])
        w = np.array([-0.5, -0.5])
        b = 0.7 #theta = -0.7
        tmp = np.sum(w*x)+b
        if tmp <= 0:
            return 0
        else:
            return 1
```

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

In [20]: NAND(1,0)
Out[20]: 1

In [21]: NAND(0,1)
Out[21]: 1

In [22]: NAND(1,1)
Out[22]: 0

그림 2-3 NAND 게이트의 진리표

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

<OR>

Jupyter Untitled Last Checkpoint: 16 hours ago (unsaved changes)

File Edit View Insert Cell Kernel Widgets Help Trusted Python 3

```
In [9]: 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
```

In [23]: OR(0,0)
Out[23]: 0

In [25]: OR(1,0)
Out[25]: 1

In [26]: OR(0,1)
Out[26]: 1

In [27]: OR(1,1)
Out[27]: 1

그림 2-4 OR 게이트의 진리표

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

2.4 퍼셉트론의 한계

- 실제로 파이썬으로 작성한 NAND, OR게이트의 코드에서도 AND와 다른 곳은 **가중치와 편향 값을 설정하는 부분**이다
- **XOR 게이트**는 구현 불가
- XOR 게이트란 **배타적 논리합**이라는 논리 회로
- x_1 과 x_2 중 한쪽이 1일 때만 1을 출력

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

그림2-5 XOR 게이트의 진리표

2.4.1 XOR 게이트 구현

지금까지 구현한 퍼셉트론으로는
XOR을 구현할 수 없다!

Ex)OR 게이트

$$(b, \omega_1, \omega_2) = (-0.5, 1.0, 1.0)$$

$$y = \begin{cases} 0 & (-0.5 + x_1 + x_2 \leq 0) \\ 1 & (-0.5 + x_1 + x_2 > 0) \end{cases}$$

=> 퍼셉트론은 직선으로 나뉜
두 영역을 만든다

=> $\bullet = 0$, $\blacktriangle = 1$

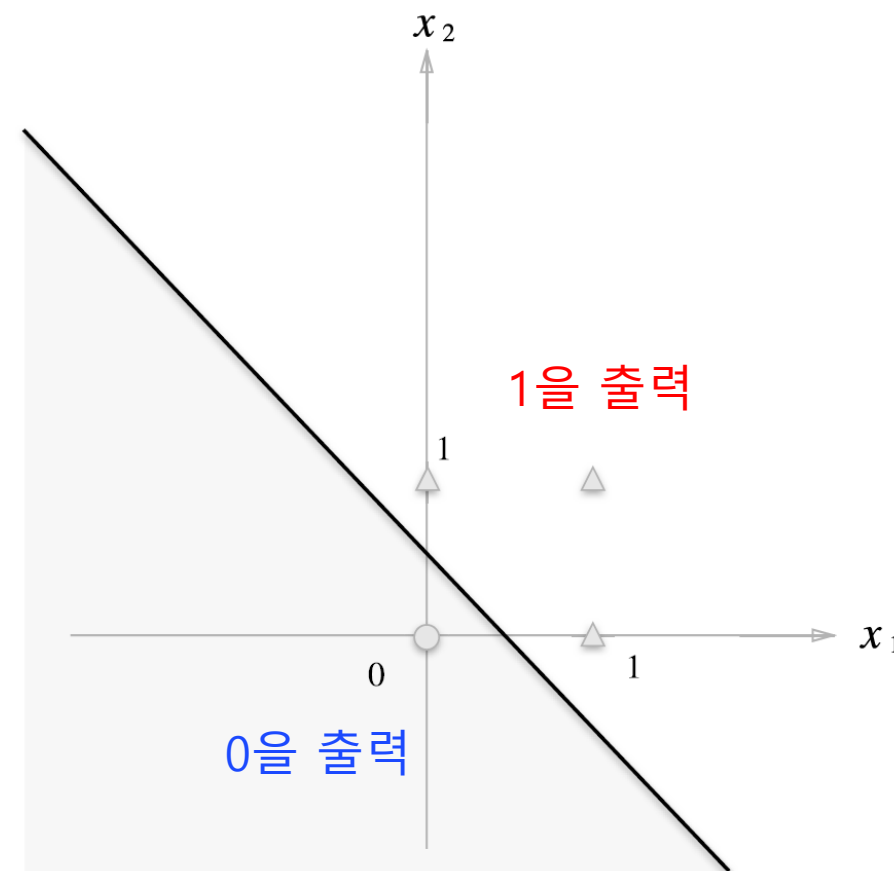
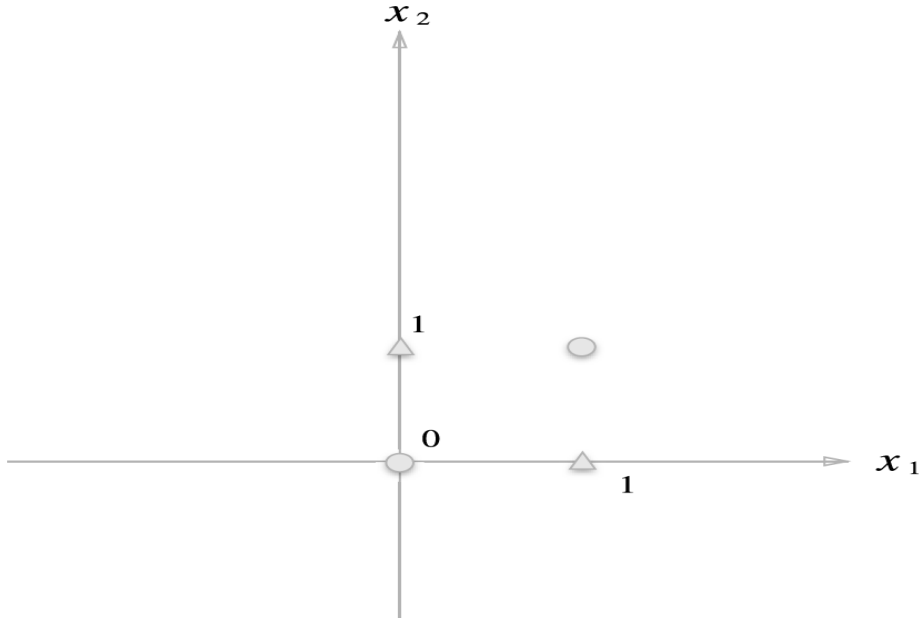


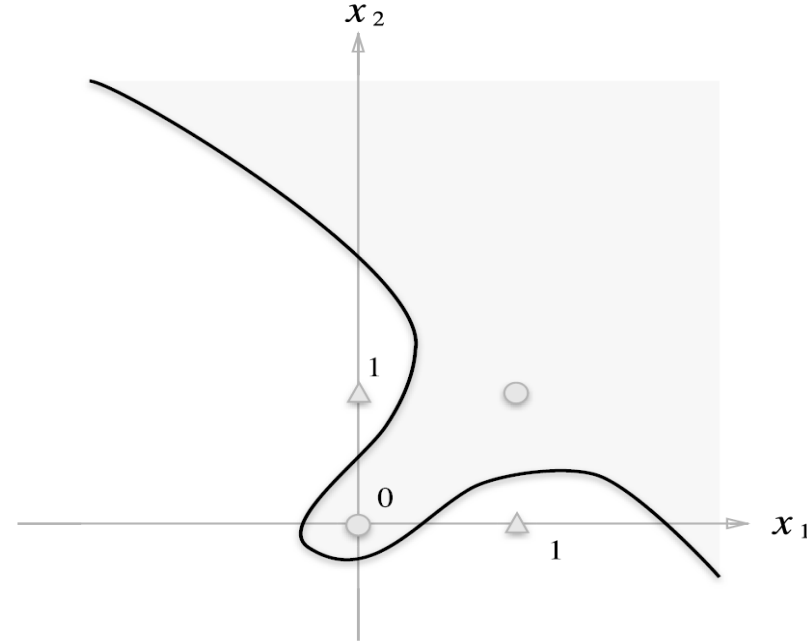
그림 2-6 퍼셉트론의 시각화

2.4.1 XOR 게이트 구현



XOR 게이트의 경우에는 OR게이트처럼 직선 하나로 \bullet , \blacktriangle 를 나눌 수 없다

퍼셉트론은 직선 하나로 나눈 영역만 표현 할 수 있다는 한계가 있다



따라서 직선의 제약을 없애고 곡선을 사용해야 한다

곡선의 영역을 **비선형 영역**, 직선의 영역을 **선형 영역**이라고 한다

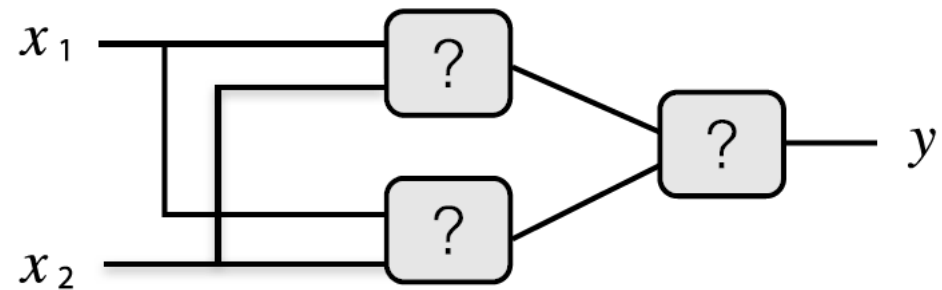
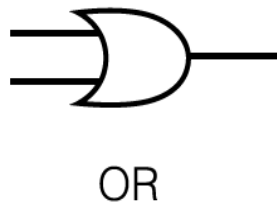
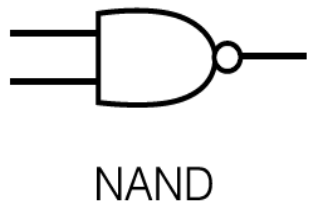
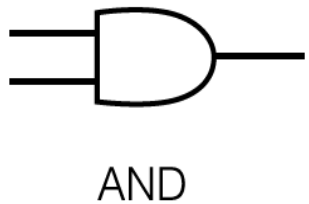
2.5 다층 퍼셉트론

퍼셉트론으로는 XOR게이트를 표현할 수 없었다.

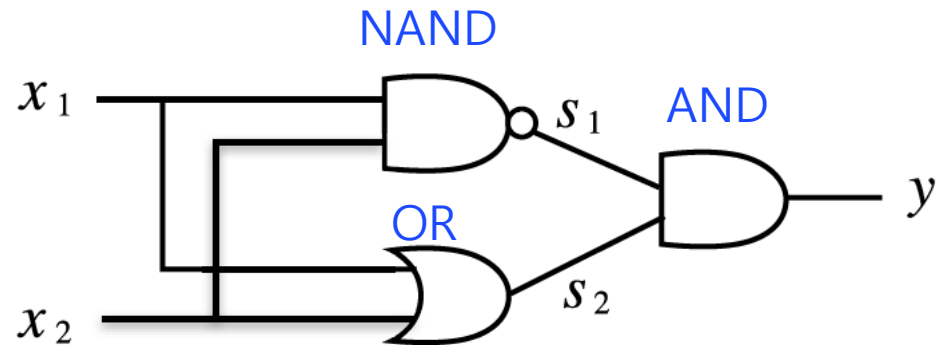
⇒ XOR 게이트를 만들기 위해서 **AND, NAND, OR 게이트를 조합**

⇒ **다층 퍼셉트론**을 사용

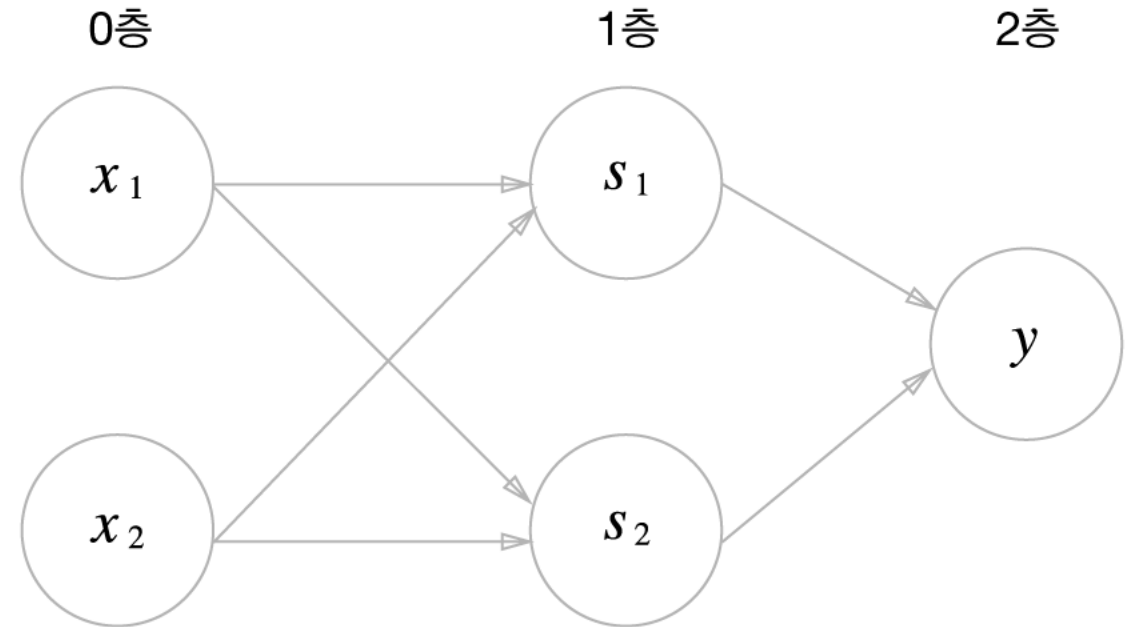
⇒ 어떻게 조합을 하면 XOR 게이트를 구현할 수 있을까?



2.5 다층 퍼셉트론



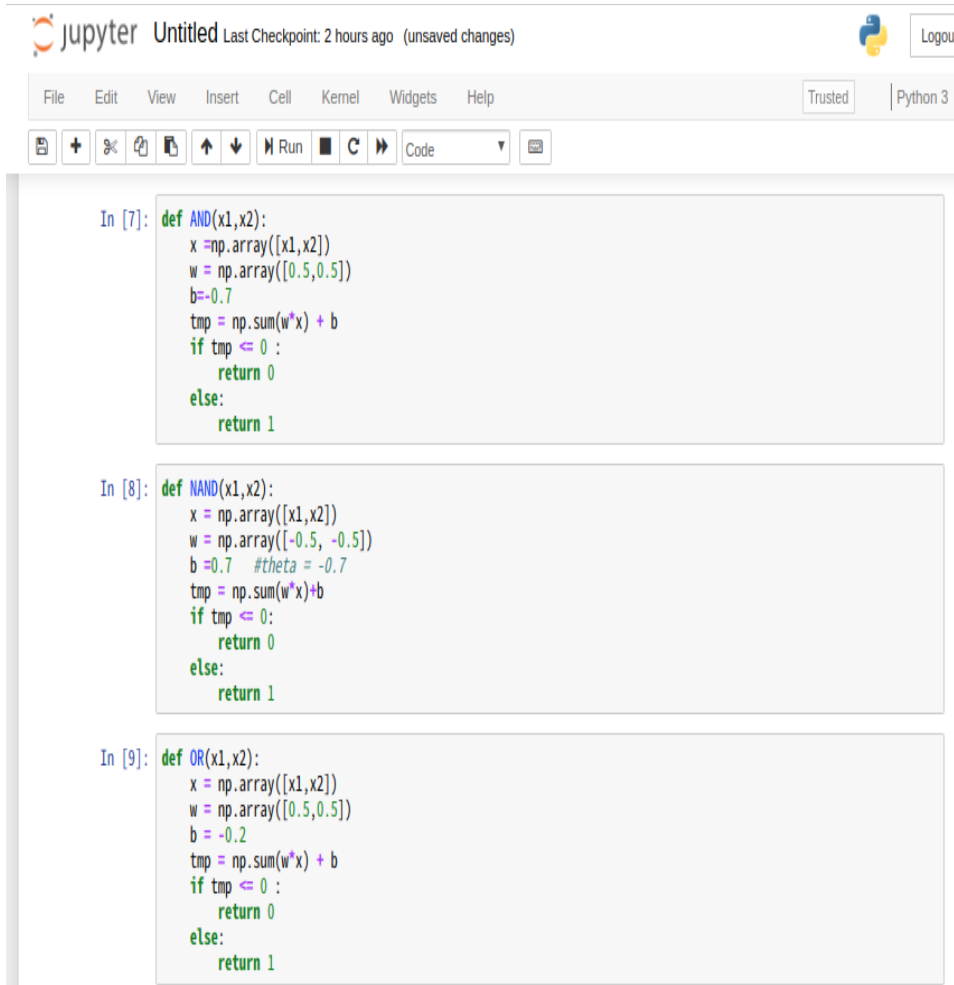
x_1	x_2	NAND s_1	OR s_2	AND y
0	0	1	0	0
1	0	1	1	1
0	1	1	1	1
1	1	0	1	0



XOR은 위와 같은 다층 구조의 네트워크

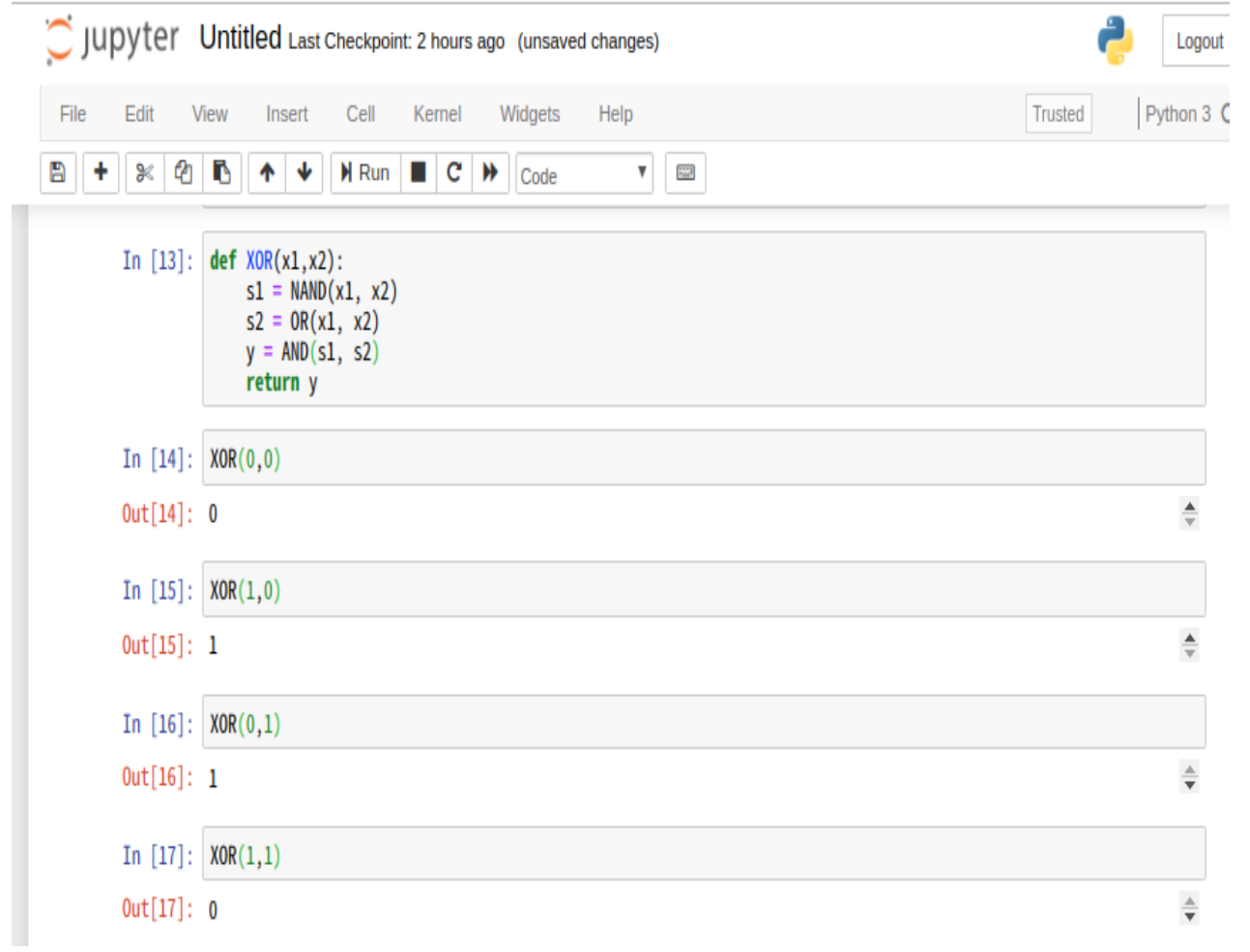
1. 0층의 두 뉴런이 입력신호를 받아 1층의 뉴런으로 신호보냄
2. 1층의 뉴런이 2층의 뉴런으로 신호를 보내고, 2층의 뉴런은 y 를 출력한다.

2.5.2 XOR게이트 구현하기



Jupyter Notebook interface showing code for AND, NAND, and OR gates. The notebook is titled "Untitled" and shows the last checkpoint from 2 hours ago. The code is written in Python 3.

```
In [7]: 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  
  
In [8]: def NAND(x1,x2):  
        x = np.array([x1,x2])  
        w = np.array([-0.5, -0.5])  
        b = 0.7 #theta = -0.7  
        tmp = np.sum(w*x)+b  
        if tmp <= 0:  
            return 0  
        else:  
            return 1  
  
In [9]: 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
```



Jupyter Notebook interface showing code for XOR gate and its test cases. The notebook is titled "Untitled" and shows the last checkpoint from 2 hours ago. The code is written in Python 3.

```
In [13]: def XOR(x1,x2):  
        s1 = NAND(x1, x2)  
        s2 = OR(x1, x2)  
        y = AND(s1, s2)  
        return y  
  
In [14]: XOR(0,0)  
Out[14]: 0  
  
In [15]: XOR(1,0)  
Out[15]: 1  
  
In [16]: XOR(0,1)  
Out[16]: 1  
  
In [17]: XOR(1,1)  
Out[17]: 0
```

2.6 NAND에서 컴퓨터까지

- 다층 퍼셉트론은 지금까지 보아온 회로보다 **복잡한 회로**를 만들 수 있다.
=> 컴퓨터도 표현 가능
- NAND 게이트의 조합으로 가능하다.
- 이론상 시그모이드 함수를 활성화 함수로 이용하면 **2층 퍼셉트론**으로도 컴퓨터를 만들 수 있다.
- 그러나 2층 퍼셉트론 구조에서 가중치를 적절히 설정하여 컴퓨터를 만들기란 매우 어렵다.
- NAND등의 저수준 소자에서 시작하여 컴퓨터를 만드는 데 필요한 모듈을 단계적으로 만들어 가는 쪽이 바람직하다.
AND,OR,NAND -> 반가산기, 전가산기, -> 산술 논리 연산 장치 -> CPU

2.7 정리

- 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.

Q&A
(DISCUSSION)