밑바닥부터 시작하는 딥러닝 CHATER 2

퍼셉트론

: 로젠블라트 1957년에 고안한 알고리즘

: 다수의 신호를 입력으로 받아 하나의 신호를 출력

:흐른다(1), 안흐른다(1)

x1, x2 → 입력 신호 / y → 출력 신호 / w1, w2 →가중치

원 → 뉴런 혹은 노드

입력신호가 뉴런에 보내질 때는 각각 고유한 가중치를 곱해진다. 뉴런에서 보내온 신호의 총합이 정해진 한계를 넘어설 때만 1을 출력합니다. 한계를 임계값(θ)이라고 한다.

$y = 0 \leftarrow w1 \times 1 + w2 \times 2 \leq \theta$

$y = 1 \leftarrow w1 \times 1 + w2 \times 2 > \theta$

퍼셉트론은 복수의 입력 신호 각각에 고유한 가중치를 부여

가중치는 각 신호가 결과에 주는 영향력을 조절하는 요소

가중치가 클수록 해당 신호가 그만큼 더 중요함

가중치== 저항

AND

x1 ×2 y

0 0 0

1 0 1

0 1 1

1 1 1

AND 게이트 → 퍼셉트론으로 표현

(0.5,0.5,0.7) (0.5,0.5,0.8) (1.0,1.0,1.0) AND 게이트의 조건을 만족합니다.

NAND

x1 ×2 y

0 0 1

1 0 1

0 1 1

1 1 0

(w1, w2, θ)

(-0.5,-0.5,-0.7)

OR

x1 ×2 y

0 0 0

1 0 1

0 1 1

1 1 1

기계학습 → 매개변수의 값을 정하는 작업을 컴퓨터가 자동

학습 → 적절한 매개변수 값을 정하는 작업

사람은 퍼셉트론의 구조를 고민하고 컴퓨터에 학습할 데이터를 주는 일을 합니다.

퍼셉트론 구현하기

-간단한 구현 부터

```
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
# AND(0,0) 0
# AND(1,0) 0
# AND(0,1) 0
# AND(1,1) 1
```

- 가중치와 편향 도입
θ → -b로 치환

y =0 ←b+w1×1+w2×2 ≤ 0
y =1 ← b+w1×1+w2×2 >0

b=편향

```
import numpy as np
x=np.array([0,1]) # 입력
w=np.array([0.5,0.5]) # 가중치
b= -0.7 # 편향

w*x #array([0. , 0.5])
np.sum(w*x) #0.5
np.sum(w*x)+b # -0.19999999999999996 -> -0.2
```

-가중치와 편향 구현하기

```
import numpy as np

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
```

b→ 뉴런이 얼마나 쉽게 활성화하느냐를 조정하는 매개변수 1 -> 뉴련 활성화
w→ 각 입력 신호가 결과에 주는 영향력(중요도)

NAND

```
import numpy as np

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
```

OR

```
import numpy as np

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
```

-퍼셉트론의 한계

XOR

x1 ×2 y

0 0 0

0 1 1

1 0 1

1 1 0

(b, w1, w2)= (-0.5, 1.0, 1.0)

y=0 (-0.5+x1+ x2≤0)

y=1 (-0.5+x1+ x2>0)

-선형과 비선형

곡선 영역을 비선형 영역

직선의 영역은 선형 영역

-다층 퍼셉트론이 출동한다면

층을 쌓아 → 다층 퍼셉트론

XOR = s1 AND s2

s1 ← x1 NAND x2 / s2 ← x1 OR x2

-XOR 게이트 구현하기

```
import numpy as np

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

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

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

def XOR(x1, x2):
  s1= NAND(x1, x2)
  s2= OR(x1,x2)
```

```
    y= AND(s1,s2)
    return y

XOR(1,1)
```

AND,OR 단층 퍼셉트론

XOR 2층 퍼셉트론

-NAND에서 컴퓨터까지

:비선형인 시그몽드함수를 활성화 함수로 이용하면 임의의 함수를 표현할 수 있는다 사실이 증명되었다.

:퍼셉트론으로 표현하는 컴퓨터도 여러층을 다시 층층이 겹친 구조로 만드는 방형이 자연스러운 흐름입니다.