인공지능개론 정리 노트#1

소프트웨어 202284011 김연재

(1) 2, 3장 학습 내용 요약

퍼셉트론: 다수의 신호를 입력으로 받아 하나의 신호를 출력한다.

퍼셉트론 하나로는 XOR게이트를 구현할 수 없다. 층을 쌓아 다층 퍼셉트론을 만들어야 한다.

NAND 게이트의 조합만으로 컴퓨터가 수행하는 일을 재현할 수 있다. 퍼셉트론으로 NAND 게이트를 만들 수 있기 때문에, 이론상 퍼셉트론으로 컴퓨터를 재현할 수 있다.

신경망의 가장 왼쪽은 입력층, 오른쪽은 출력층, 중간은 은닉층

입력 신호의 총합을 출력 신호로 변환하는 함수를 활성화 함수라 한다.

활성화 함수는 임계값을 경계로 출력이 바뀌는데 이런 함수를 계단 함수라 한다.

예) 시그모이드 함수: 1/(1+np.exp(-x))

계단 함수와 시그모이드 함수는 모두 비선형 함수이다. 선형 함수는 층을 쌓는 의미가 없다.

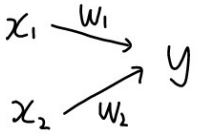
최근에는 ReLU 함수를 사용한다. 0을 넘으면 그 입력을 그대로 출력하고, 0 이하이면 0을 출력한다.

(2) 2, 3장 테스트 내용

-퍼셉트론 가중치 손으로 구하기

AND 게이트 퍼셉트론

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



$$y = \begin{cases} 0 & (x_1w_1 + x_2w_2 \le \theta) \\ 1 & (x_1w_1 + x_2w_2 > \theta) \end{cases}$$

(w_1, w_2, θ)가 $(1, 1, 1.5)$ 일때

$y = 0+0 \le 1.5$

$= 0+1.5 \le 1.5$

$= 1.5+0 \le 1.5$

$= 1.5+1.5 > 1.5$

NAND 게이트 퍼셉트론

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

(w_1, w_2, θ)가 $(-1, -1, -1.5)$일때

$y = 0+0 > -1.5$

$= 0-1 > -1.5$

$= -1+0 > -1.5$

$= -1+(-1) \le -1.5$

OR 게이트 퍼셉트론

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

(w_1, w_2, θ)가 $(10, 10, 5)$일때

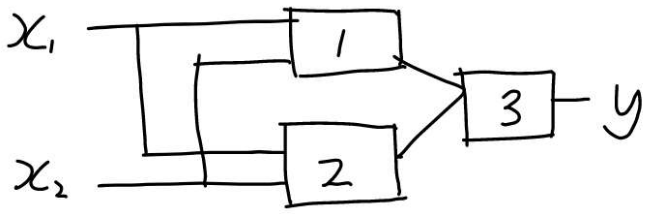
$y = 0+0 < 5$

$0+10 \ge 5$

$10+0 \ge 5$

$10+10 \ge 5$

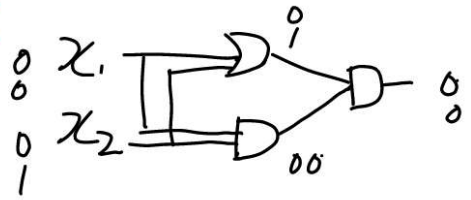
XOR 게이트 만드는 방법



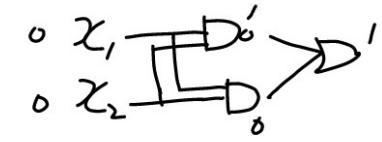
XOR 게이트

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

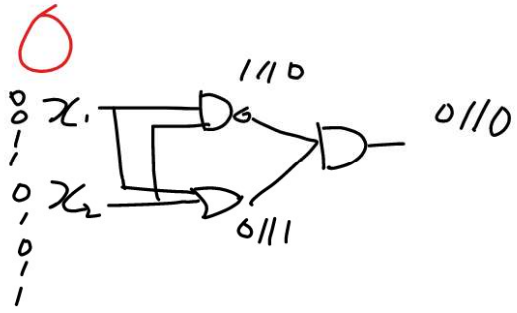
X



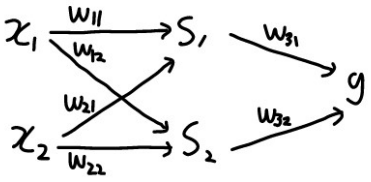
X



O



다층 퍼셉트론



$$S_1 = x_1 w_{11} + x_2 w_{21} + b_1$$

$$S_2 = x_1 w_{12} + x_2 w_{22} + b_2$$

$$\begin{pmatrix} w_{11} & w_{12} \\ w_{21} & w_{22} \end{pmatrix} \begin{pmatrix} x_1 \\ x_2 \end{pmatrix} + \begin{pmatrix} b_1 \\ b_2 \end{pmatrix}$$

(3) 2, 3장 파이썬 코드

-논리회로 파이썬으로 구현

```python
import numpy as np

# 가중치와 바이어스
w11 = np.array([-2, -2])
w12 = np.array([2, 2])
w2 = np.array([1, 1])
b1 = 3
b2 = -1
b3 = -1

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

# NAND 게이트
def NAND(x1, x2):
    return perceptron(np.array([x1, x2]), w11, b1)

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

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

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

# x1, x2 값을 번갈아 대입하며 최종값 출력
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))
```

[34]

```
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
```

-파이썬으로 신경망 구현하기

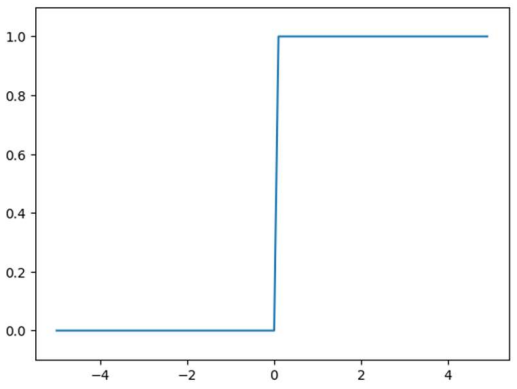
```python
import matplotlib.pylab as plt

def step_function(x):
    return np.array(x > 0, dtype = np.int32)
```

[16]

```python
x = np.arange(-5.0, 5.0, 0.1)
y = step_function(x)
plt.plot(x,y)
plt.ylim(-0.1, 1.1)
plt.show()
```

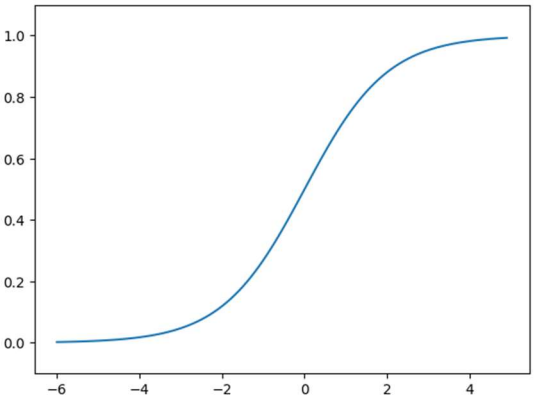
[17]




```python
import numpy as np
import matplotlib.pylab as plt
def sigmoid(x):
    return 1/(1+np.exp(-x))

x=np.arange(-6.0, 5.0, 0.1)
y=sigmoid(x)
plt.plot(x,y)
plt.ylim(-0.1,1.1)
plt.show()
```

[3]



```
#신경망 구현하기

X = np.array([1.0, 0.5])
W1 = np.array([[0.1, 0.3, 0.5], [0.2, 0.4, 0.6]])
B1 = np.array([0.1, 0.2, 0.3])

print(W1.shape)
print(X.shape)
print(B1.shape)

A1 = np.dot(X, W1) + B1
print(A1)
```
[9]

```
(2, 3)
(2,)
(3,)
[0.3 0.7 1.1]
```

```
Z1 = sigmoid(A1)

print(A1)
print(Z1)
```
[10]

```
[0.3 0.7 1.1]
[0.57444252 0.66818777 0.75026011]
```

```
#1층에서 2층으로 신호 전달

def sigmold(x):
    return 1/(1+np.exp(-x))

W2 = np.array([[0.1, 0.4], [0.2, 0.5], [0.3, 0.6]])
B2 = np.array([0.1, 0.2])

print(Z1.shape)
print(W2.shape)
print(B2.shape)

A2 = np.dot(Z1, W2) + B2
Z2 = sigmoid(A2)
```
[11]

```
(3,)
(3, 2)
(2,)
```

```
#2층에서 출력층으로 신호 전달

def identity_function(x):
    return X

W3 = np.array([[0.1, 0.3], [0.2, 0.5]])
B3 = np.array([0.1, 0.2])

A3 = np.dot(Z2, W3) + B3
Y = identity_function(A3)

print(Y)
```
[13]

```
[1.  0.5]
```

내용이 많아 일부만 첨부했습니다.

2장, 3장에서 학습한 코드 전체는 아래 깃허브에 있습니다.

https://github.com/kyj0503/pi-deeplearning/tree/main

(3) 2장 질문

1. 퍼셉트론 연산, 구현만을 위한 하드웨어가 있을까?

A. 초기 신경망 하드웨어는 주로 퍼셉트론 연산을 가속하기 위해 설계되었다. 1960년대에는 퍼셉트론을 구현하기 위한 전자회로가 만들어졌으며, 이는 단층 퍼셉트론 형태로 제한적인 문제 해결에 사용됐다. 최근에는 훨씬 더 복잡한 심층 신경망 연산을 가속하기 위해 GPU, TPU 등 일반적인 컴퓨팅 장치를 주로 사용한다.

2. 가중치를 손으로 구하는게 너무 힘든데 자동으로 구하는 방법이 없을까?

A. 오차역전파 알고리즘이라는 것을 사용해 신경망의 출력과 실제 값의 오차를 역방향으로 전파하여 가중치를 조정할 수 있다. 텐서플로우나 파이토치 등의 딥러닝 프레임워크는 역전파 알고리즘을 자동으로 구현하고 가중치를 업데이트하는 기능을 제공한다.

(4) 3장 질문

1. 최근 마이크로소프트의 연구팀에서 가중치를 -1, 0, 1의 세 가지 값으로만 사용함으로써 대규모 언어 모델의 계산 비용을 급감시키는데 성공했다는 기사를 봤다. 우리가 파이썬으로 작성하는 얕은 수준의 신경망에서는 적용시킬 수 없는 것일까?

A. 마소 연구팀의 기술은 트러너리 가중치라 부르는데 이는 대규모 언어 모델에 특화된 방식이고, 일반적으로 모든 종류의 신경망에 적용되진 않는다. 파이썬으로 우리가 구현하는 작은 규모의 신경망에서는 가중치 압축을 위한 특별한 기술을 적용하기 어려울 수도 있을 것이다. 오히려 성능이 떨어질수도 있다!

2. 3층 신경망까지 만드는 방법을 배웠는데, 실제로 상용화된 인공지능 서비스에서 사용하는 신경망은 몇층일까?

A. 간단한 이미지 분류나 텍스트 분류에는 상대적으로 얕은 신경망이 사용되지만, 상용화된 많은 서비스는 깊은 신경망을 사용한다. 구글의 BERT 모델은 24층, OpenAI의 GPT-3는 96층의 신경망을 사용한다.