인공지능개론 정리노트 #1

ICT융합공학부 202204010 공성택

2장

Q) 일반적으로 퍼셉트론의 가중치 값에 제한된 범위가 있는가?

A) 조사해본 결과, 퍼셉트론의 가중치에는 일반적으로 직접적인 제한을 없지만 다중 퍼셉트론과 같은 신경망에서는 가중치 제한이나 정규화를 적용할 수 있다. 주로 사용되는 방법으로는 크게 두 가지가 존재하는데, 가중치 클리핑과 가중치 감쇠이다. 가중치 클리핑은 가중치의 크기를 최대값이나 최소값으로 제한하여 일정 범위 내에 유지하는 것이고, 가중치 감쇠는 가중치의 크기에 패널티를 부과하여 네트워크의 복잡도를 줄이고 가중치를 제한한다. 이러한 가중치 제한은 주로 신경망의 훈련을 안정화하는데 사용한다고 한다.

Q) 다층 퍼셉트론으로 x축과 y축 상의 데이터를 선형으로 분류할 수 있다고 배웠는데, 3차원(x, y, z) 상의 데이터도 분류할 수 있는가?

A) 조사해본 결과, 다층 퍼셉트론은 3차원 상의 데이터 또한 분류할 수 있다고 한다. 다층 퍼셉트론은 여러 개(3개)의 입력을 받아 각각의 가중치를 곱해, 활성화 함수를 통과시켜 계산한다. 이를 통해 다양한 형태의 입력 데이터를 분류할 수 있다. 보통 ReLU(Rectified Linear Unit)이나 시그모이드 함수와 같은 활성화 함수가 사용된다고 한다. 단계를 따라 다층 퍼셉트론을 구축하고 훈련해서, 3차원 상의 데이터를 분류할 수 있다고 한다.

3장

Q) ReLU 함수와 시그모이드 함수 이외에도 신경망에서 사용되는 다른 활성화 함수들은 무엇이 있을까?

A) 조사해본 결과,ReLU와 시그모이드 함수 외에 일반적으로 사용되는 다른 활성화 함수로는 하이퍼볼릭 탄젠트 함수, 리키 렐루 함수, 소프트맥스 함수가 있다. 하이퍼볼릭 탄젠트 함수는 시그모이드 함수와 비슷하지만, 출력 범위가 더 넓은 −1에서 1사이이고, 시그모이드 함수와 마찬가지로 기울기 소실 문제가 있다. 리키 렐루 함수는 ReLU 함수의 변종으로, 입력 값이 음수인 경우 기울기가 0이 아닌 작은 상수로 설정된다. 때문에 입력 값이 음수인 경우에도 그래디언트가 0이 되는 문제를 완화한다고 한다. 소프트맥스 함수는 강의 자료에 나와있듯, 모든 출력 값의 합이 1이 되도록하여 0에서 1사이의 확률값으로 변환한다. 출력 값을 각 클래스에 대한 확률로 해석하기 때문에, 분류 결과를 확률적으로 해석할 수 있도록 한다고 한다.

Q) 신경망은 활성화 함수로 비선형 함수를 사용하는 것이 일반적이고 효율적이라고 했는데, 선형 함수를 사용하는 경우는 없는가?

A) 신경망은 비선형 활성화 함수를 사용하여 모델이 복잡한 관계를 학습할 수 있도록 하는

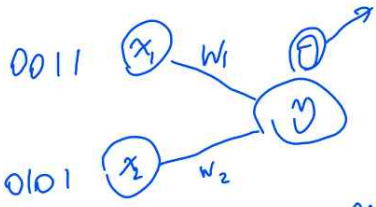
데, 선형 활성화 함수를 사용하는 경우 네트워크가 단순한 선형 변환만 수행하게 되기 때문에 층을 여러 개로 쌓아도 결과적으로 하나의 선형 변환으로 표현되는 한계가 있다. 조사해 본 결과, 선형 활성화 함수를 사용하는 경우는 선형 회귀 문제를 해결할 때 간단한 모델을 만드는데 사용한다. 또한, 특별한 제약이 있는 경우, 선형 활성화 함수를 사용한다고 한다. 이외에도 분류 문제에서 출력층이 선형 함수로 구성된 경우와 간단한 상호 작용 효과를 평가하고자 할 때 선형 함수를 사용한다고 한다. 이외의 경우는 비선형 함수가 훨씬 효율적이고 선형 함수의 한계를 극복하기 때문에 비선형 함수를 사용한다.

AND 게이트 가중치 손으로 구하기

2.2 단순한 논리 회로

- **AND 게이트**
  - AND 게이트의 진리표 – AND 게이트의 입력 신호와 출력 신호의 대응표

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



$y = x_1 w_1 + x_2 w_2$

- **AND 게이트를 퍼셉트론으로 표현**
  - AND 게이트 진리표대로 작동하는 w1, w2, θ 의 값을 정하는 것임
  - AND 게이트의 조건을 충족하는 퍼셉트론 설정하기
  - AND 게이트의 조건을 만족하는 매개변수 조합은 무한히 많음

NAND 게이트, OR 게이트 가중치 손으로 계산하기

정리노트

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

① NAND 게이트 진리표

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

② OR 게이트 진리표

① $(w_1, w_2, \theta) = 0.2, 0.3, 0.1$

0,0 일때, $0+0 \leq 0.1$ $y=0$ 이므로 X

θ가 음수? $\theta = -0.1$

0,0 일 때, $0 > -0.1$ $\therefore y=1$

1,0 일 때, $0.2 > -0.1$ $\therefore y=1$

0,1 일 때, $0.3 > -0.1$ $\therefore y=1$

1,1 일 때, $0.5 > -0.1$ $\therefore y=1$ 이므로 X

$x_1 w_1 + x_2 w_2 \leq \theta$

$\therefore w_1, w_2, \theta$ 모두 음수로 가정

$w_1 > \theta$, $w_2 > \theta$, $w_1 + w_2 \leq \theta$

$\therefore (w_1, w_2, \theta) = (-0.3, -0.4, -0.5)$

0, 0 일때 $0 > -0.5$ $\therefore y=1$

1, 0 일 때 $-0.3 > -0.5$ $\therefore y=1$

0, 1 일 때 $-0.4 > -0.5$ $\therefore y=1$

1, 1 일 때 $-0.7 \leq -0.5$ $\therefore y=0$ 이므로

$(w_1, w_2, \theta) = (-0.3, -0.4, -0.5)$ 가 충족

② 0, 0일 때 $y=0$ 이므로 $\theta \geq 0$

1, 0일 때 $y=1$ 이므로 $w_1 > \theta$

0, 1 일 때 $y=1$ 이므로 $w_2 > \theta$

1, 1 일 때 $y=1$ 이므로 $w_1 + w_2 > \theta$

$\therefore (w_1, w_2, \theta) = (0.9, 0.9, 0.3)$ 이라고 가정

0, 0 일때 $0 < 0.3$ $\therefore y=0$

1, 0 일 때 $0.9 > 0.3$ $\therefore y=1$

0, 1 일 때 $0.9 > 0.3$ $\therefore y=1$

1, 1 일 때 $1.6 > 0.3$ $\therefore y=1$

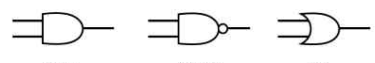
$\therefore (w_1, w_2, \theta) = (0.9, 0.9, 0.3)$ 가 충족

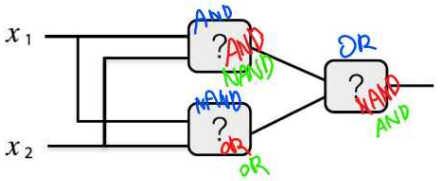
기존 게이트를 조합해서 XOR 게이트를 구현하기

2.5 다층 퍼셉트론 # 정리 노트

- **다층 퍼셉트론 (multi-layer perceptron)**
  - 단층 퍼셉트론으로는 XOR 게이트를 표현할 수 없음
  - 단층 퍼셉트론으로는 비선형 영역을 분리할 수 없음
  - 퍼셉트론을 여러 층으로 쌓아 만듦

- **기존 게이트 조합하기**
  - XOR 게이트를 만드는 방법은 다양함
  - AND, NAND, OR 게이트를 어떻게 조합하면 XOR 게이트를 구현할 수 있을까?





3장 신경망 프로그래밍 실습1

2024_03_27.ipynb

File Edit View Run Kernel Tabs Settings Help

Code

[7]:
```
import numpy as np
```

[4]:
```
def step_function(x):
    if x > 0:
        return 1
    else:
        return 0
```

[5]:
```
print(step_function(0.5))
print(step_function(-0.5))
print(step_function(5))
```

```
1
0
1
```

[10]:
```
import numpy as np

def step_function(x):
    y = x > 0
    return y.astype(np.int32)
```

[11]:
```
x = np.array([0.5, -0.5, 5.0])
print(step_function(x))
```

```
[1 0 1]
```

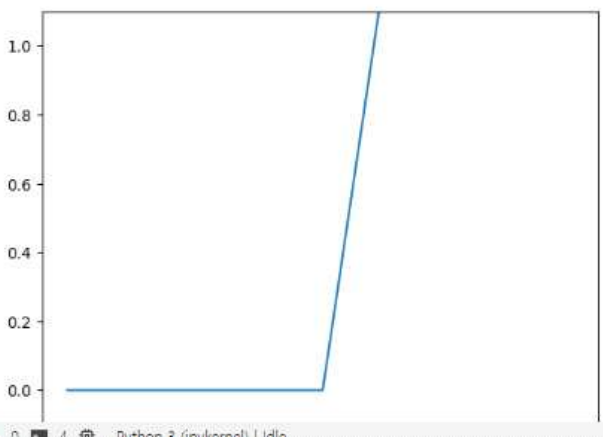
[23]:
```
import numpy as np
import matplotlib.pylab as plt

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

[24]:
```
def sigmoid(x):
    return 1 / (1 + np.exp(-x))
```

[25]:
```
def relu(x):
    return np.maximum(0, x)
```

[26]:
```
X = np.arange(-5.0, 5.0, 0.1)
#Y = step_function(X)
#Y = sigmoid(X)
Y = relu(X)
plt.plot(X, Y)
plt.ylim(-0.1, 1.1)
plt.show()
```



3장 신경망 프로그래밍 실습2

2024_03_27.ipynb

File Edit View Run Kernel Tabs Settings Help

Code

```
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
Z1 = sigmoid(A1)

#print(A1)
#print(Z1)

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)

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)
```

```
[0.31682708 0.77338016]
```

[36]:
```
def init_network():
    network = {}
    network['W1'] = np.array([[0.1, 0.3, 0.5], [0.2, 0.4, 0.6]])
    network['B1'] = np.array([0.1, 0.2, 0.3])
    network['W2'] = np.array([[0.1, 0.4], [0.2, 0.5], [0.3, 0.6]])
    network['B2'] = np.array([0.1, 0.2])
    network['W3'] = np.array([[0.1, 0.3], [0.2, 0.5]])
    network['B3'] = np.array([0.1, 0.2])

    return network
```

[37]:
```
def forward(network, x):
    W1, W2, W3 = network['W1'], network['W2'], network['W3']
    b1, b2, b3 = network['B1'], network['B2'], network['B3']

    a1 = np.dot(x, W1) + b1
    z1 = sigmoid(a1)
    a2 = np.dot(z1, W2) + b2
    z2 = sigmoid(a2)
    a3 = np.dot(z2, W3) + b3
    y = identity_function(a3)

    return y
```

[38]:
```
network = init_network()
x = np.array([0.3, 0.4])
y = forward(network, x)
print(y)
```

```
[0.31272088 0.76265591]
```