2020254004 손의걸

1.

(a) 일종의 학습 기계로서 Rosenblatt가 제안한 것으로, 뇌의 학습 기능을 모델화한 기계이다

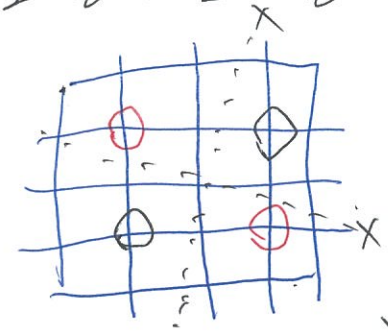
수용층, 연합층, 반응층 세 부분으로 되어있다.

수용층 : 외부 자극을 받아들임

연합층 : 수용층의 가중 입력을 받아 반응층으로 전달하는 기능을 담당

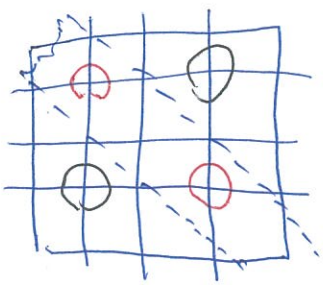
반응층 : 최종 출력을 내보내는 기능

(b) 헵에 퍼셉트론은 출력값이 1또는 0 이므로 선형분류 모델이라고 볼 수 있다.



아무리 선을 그어보아도 선 하나만으로는 검은점과 흰점을 분류할 수 없다 이것이 퍼셉트론의 한계로 등장하는 XOR 문제이다.

극복방안 : 단층레이어가 아닌 다층으로 해결이 가능하다. 인공신경망이라는 모델을 이용해서 멀티레이어를 이용한 마빈 민스키 교수가 말했던 방법대로 XOR 문제를 해결할 수 있다.



2020254004 손희경

2.

```
import numpy as np
def AND(x1,x2):  #AND 연산 정의
    X = np.array([x1, x2])
    w = np.array([0.5, 0.5])
    b = -0.7
    temp = np.sum(w*x)+b
    if temp<=0:
        return 0

    else:
        retunn 1

def NAND(x1, x2): #NAND 연산 정의
    x = np.array([[x1, x2]])
    w = np.array([[-0.5, -0.5]])
    b = 0.7

    temp = np.sum(w*x)+b
    if temp <= 0:
        return 0
    else :
        retunn 1

def OR(x1, x2): # OR 연산 정의
    x = np.array([[x1, x2]])
    w = np.array([[0.5, 0.5]])
    b = -0.2
    temp = np.sum(w*x)+b
    if temp <= 0:
        return 0
    else:
        return 1
```

2020254004 신의경

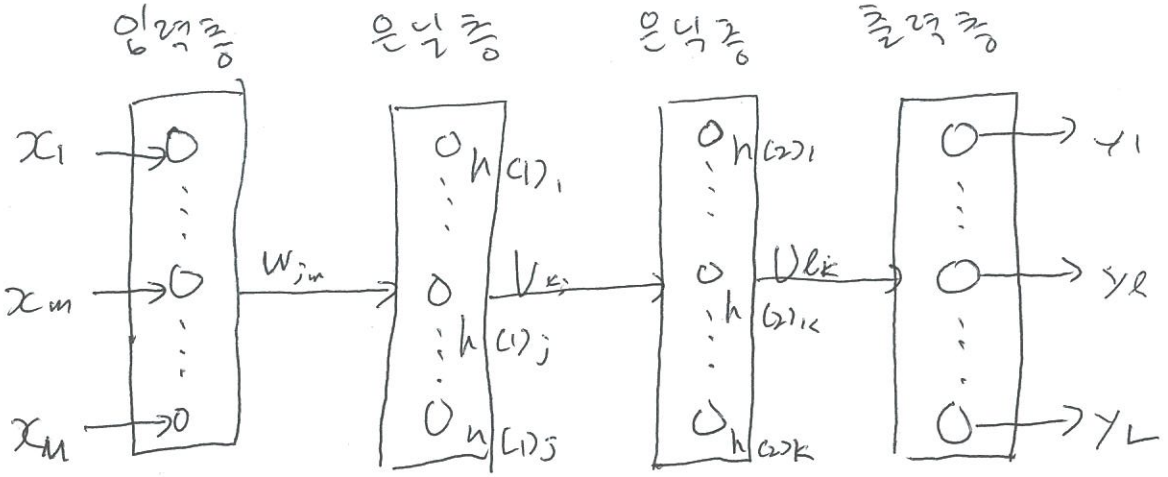
```
def XOR(x1, x2):          # XOR 연산정의
    s1 = NAND(x1, x2)
    s2 = OR(x1, x2)
    y = AND(s1, s2)
    return y

def Full_Adder(X, Y, Z):  # Full Adder 정의
    SUM = XOR(XOR(X, Y), Z)
    CARRY = OR(AND(X, Y), AND(XOR(X, Y), Z))
    return CARRY, SUM

print('X=0, Y=0, Z=0 =>')   # X=0, Y=0, Z=0 =>
print((Full_Adder(0, 0, 0)))  # Full_Adder(0, 0, 0) 실행
Print('X=0, Y=1, Z=0 =>')
Print((Full_Adder(0, 1, 0)))
```

: X, Y, Z 값을 변경하면서
: Full_Adder 결과출력
:

```
print('X=1, Y=1, Z=1 =>')
Print((Full_Adder(1, 1, 1)))
```

2020254004 손의꼰

3. (a)



신경망 모델을 학습 시킬때 오차 함수에 대한 최적 해를 구하기 위해 매개변수의 경사를 구해야 하는데 이과정에서 경사가 없어져 버리는(0이과버리는) 문제를 경사 소실 문제 라고 한다.

경사가 소실되면 오차 역전파을 제대로 학습 못하는데, 위 그림 그림에 은닉층이 늘어날수록 경사소실 문제가 한꺼번에 나타난다.

은닉층이 두개인 신경망 모델을 보면 각 층에서 나오는 뉴런의 출력은 식(A) 식(B) 식(C) 로 쓸수 있다.

- x : 입력층의 값
- $h^{(1)}$, $h^{(2)}$: 은닉층 2개
- y : 출력 층의 값
- W, V, U : 각층의 웨이트 행렬
- b, c, d : 각층의 바이어스 벡터
- $\sigma(\cdot)$: 활성화를 시키는 함수

식(A) $h^{(1)} = \sigma(Wx+b)$

식(B) $h^{(2)} = \sigma(Vh^{(1)}+c)$

식(C) $y = \mathrm{softmax}(Uh^{(2)}+d)$

2020254004 손의겸

3.(b) Softmax는 임의의 값을 출력으로 0~1 사이의 값을 모든 정규화하며 출력 값들의 총합은 항상 1이 되는 특성을 가진 함수이다.

분류하고 싶은 클래스 만큼 출력으로 구성한다.
가장 큰 출력 값을 부여받은 클래스가 확률이 가장 높은것으로 이용된다.

그러나 소프트맥스 결과값이 [0.4, 0.3, 0.2, 0.1] 으로 나오나 1등한 0.4와 [0.7, 0.1, 0.1, 0.1] 으로 나오나 1등한 0.7을 다를것이므로 그정도에 따라 추가 판단하기도 한다

$$f(\vec{x})_i = \frac{e^{x_i}}{\sum_{k=1}^{K} e^{x_k}} \quad \text{for } i = 1, \dots, K$$

결국 가장큰 값을 이어 소프트맥스 이전의 가장큰 값이였다 따라서 추론(운영)단계에서 연산 속도를 빠르게 하기 위해 생략하기도 한다.

소프트맥스 결과값을 One hot encoder 입력으로 연결하면 가장큰 값만 True, 나머지는 False 값이 나오게 하여 이용가능하다.

4.(a) 손실함수는 신경망의 성능을 나타내는 지표라할수 있다. 여기서 '정확도' 아닌, 같의 같인 '손실함수'를 택하는 이유는, 미분가능한 함수를 이용하여 신경망 학습을 효과적으로 하기 위함이다.

(b) MSE

$$MSE = \frac{1}{N}\sum_{i=1}^{N}(y_i - \hat{y}_i)^2$$

예측한 값과 실제 값 사이의 다중 간격 오차를 정의

(ㅂ) RMSE

$$RMSE = \sqrt{\sum_{i=1}^{n} \frac{(\hat{y}_i - y_i)^2}{n}}$$

MSE에 루트($\sqrt{}$)를 씌운것으로 MSE와 기본적으로 동일
MSE의 오류 제곱을 구하기 때문에 실제 오류 평균보다 더
커지는 특성이 있어 MSE에 루트를 씌운 RMSE 은 값의
왜곡을 줄여준다.

Binary Crossentropy
MSE와 RMSE와 다르게 실제 레이블과 예측 레이블간
교차 엔트로피손실을 계산한다. 2개의 레이블 클래스를 (0, 1로)
가질 이 있을때 사용하면 좋다.

5.) 주간, 야간 영상 융합
→ 비지도 학습으로 주간 및 야간 영상을 CNN 으로 특징을 추출하고
이를 ADD 한 후 다시 Decoding 하여 주간 및 야간 영상의
장점을 갖는 융합영상으로 생성

- 운전자의 행동 패턴을 기반으로 한 차량의 어댑티브 크루즈
→ 강화 학습으로서 운전자의 운전패턴을 학습하여 어댑티브
크루즈 실행시 차량의 가속 및 감속을 운전자의 평소 습관에
맞게 보다 편안한 반자율 주행 구현

- 보드의 불량검사
→ 지도 학습으로, PCB 기판의 불량품과 정상품의 이미지
데이터를 학습하여 카메라로 불량 여부를 자동으로
검사