퍼셉트론 보고서

제출일: 25.06.25
제출인: 서윤철

<순서>

- AND GATE
- OR GATE
- NAND GATE
- XOR GATE
- 단층 퍼셉트론 구조
- 다층 퍼셉트론 구조
- 부연설명
- 결론
- 해결방안
- 다층 퍼셉트론을 사용한 결과
- 참고

AND GATE

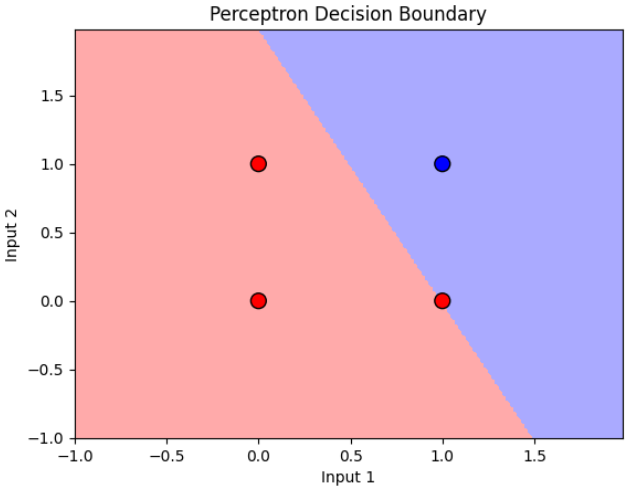
- 예측한 결과

```
Epoch 1/10, Errors: 1
Epoch 2/10, Errors: 3
Epoch 3/10, Errors: 3
Epoch 4/10, Errors: 2
Epoch 5/10, Errors: 1
Epoch 6/10, Errors: 0
Epoch 7/10, Errors: 0
Epoch 8/10, Errors: 0
Epoch 9/10, Errors: 0
Epoch 10/10, Errors: 0

AND GATE TEST:
Input: [0 0], Predicted Output: 0
Input: [0 1], Predicted Output: 0
Input: [1 0], Predicted Output: 0
Input: [1 1], Predicted Output: 1
```

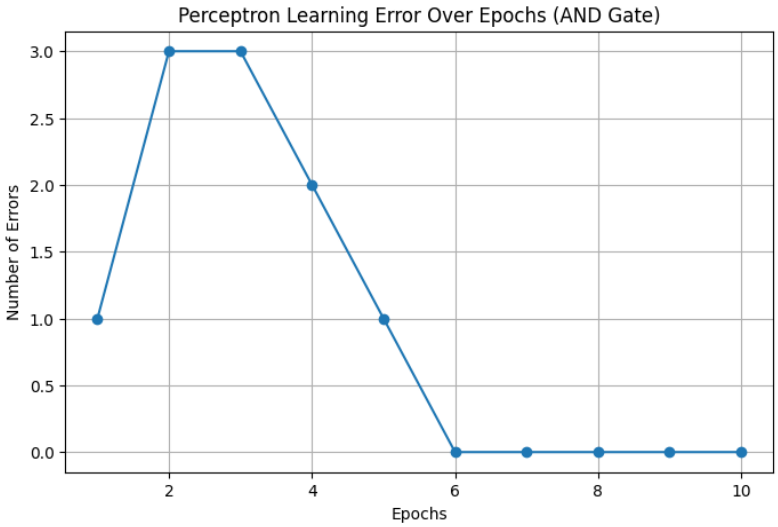
=> Epoch 6 부터 Error = 0

- 결정 경계 시각화



=> True

- 오류 시각화



OR GATE

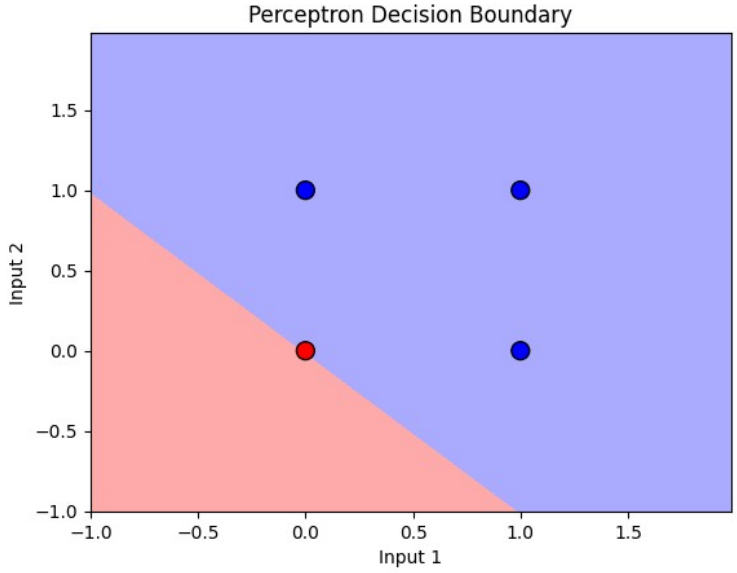
- 예측한 결과

```
Epoch 1/10, Errors: 1
Epoch 2/10, Errors: 2
Epoch 3/10, Errors: 1
Epoch 4/10, Errors: 0
Epoch 5/10, Errors: 0
Epoch 6/10, Errors: 0
Epoch 7/10, Errors: 0
Epoch 8/10, Errors: 0
Epoch 9/10, Errors: 0
Epoch 10/10, Errors: 0

OR GATE TEST:
Input: [0 0], Predicted Output: 0
Input: [0 1], Predicted Output: 1
Input: [1 0], Predicted Output: 1
Input: [1 1], Predicted Output: 1
```

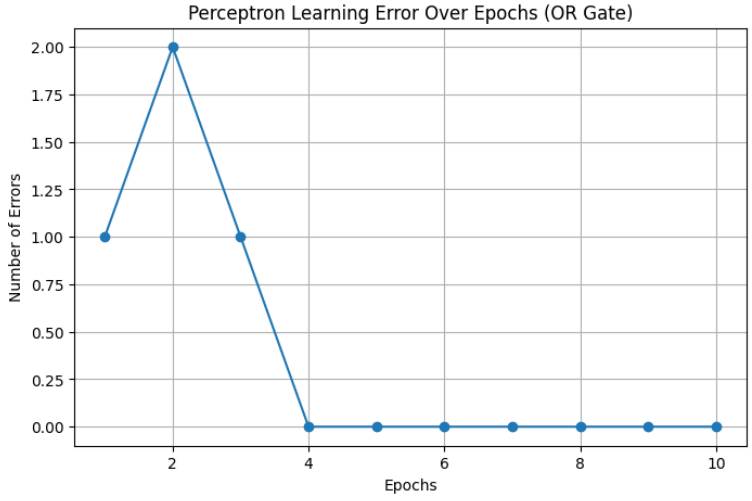
=> Epoch 4 부터 Error = 0

- 결정 경계 시각화



=> True

- 오류 시각화



NAND GATE

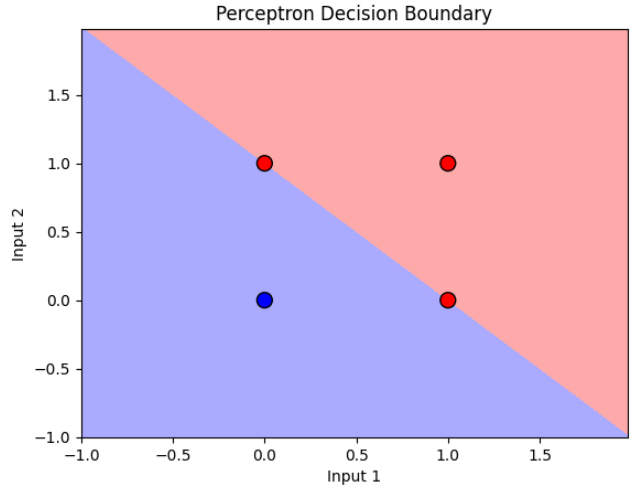
- 예측한 결과

```
Epoch 1/10, Errors: 2
Epoch 2/10, Errors: 2
Epoch 3/10, Errors: 1
Epoch 4/10, Errors: 0
Epoch 5/10, Errors: 0
Epoch 6/10, Errors: 0
Epoch 7/10, Errors: 0
Epoch 8/10, Errors: 0
Epoch 9/10, Errors: 0
Epoch 10/10, Errors: 0

NAND GATE TEST:
Input: [0 0], Predicted Output: 1
Input: [0 1], Predicted Output: 0
Input: [1 0], Predicted Output: 0
Input: [1 1], Predicted Output: 0
```

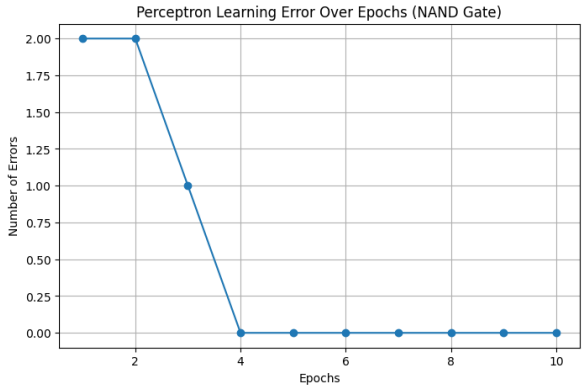
=> Epoch 4 부터 Error = 0

- 결정 경계 시각화



=> True

- 오류 시각화



XOR GATE

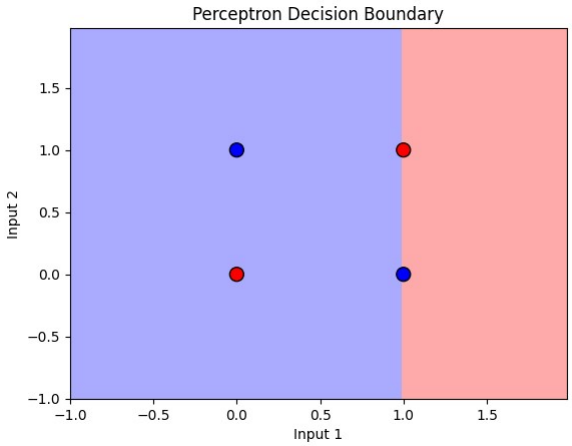
- 예측한 결과

```
Epoch 1/10, Errors: 2
Epoch 2/10, Errors: 3
Epoch 3/10, Errors: 4
Epoch 4/10, Errors: 4
Epoch 5/10, Errors: 4
Epoch 6/10, Errors: 4
Epoch 7/10, Errors: 4
Epoch 8/10, Errors: 4
Epoch 9/10, Errors: 4
Epoch 10/10, Errors: 4

XOR GATE TEST:
Input: [0 0], Predicted Output: 1
Input: [0 1], Predicted Output: 1
Input: [1 0], Predicted Output: 0
Input: [1 1], Predicted Output: 0
```

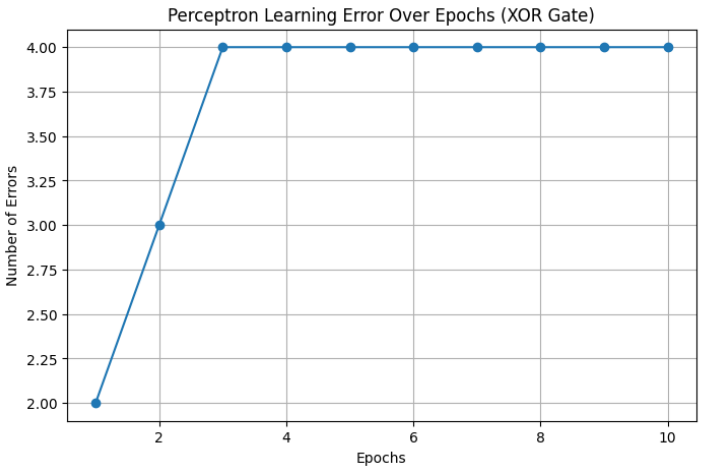
=> 예측한 결과와 실제 결과 사이에 **오류** 발생!!

- 결정 경계 시각화



=> ERROR

- 오류 시각화



단층 퍼셉트론 구조

```python
class Perceptron:
    def __init__(self, input_size, lr=0.1, epochs=10): # lr: 학습률
        self.weights = np.zeros(input_size)            # 가중치 초기화
        self.bias = 0                                  # 바이어스 초기화
        self.lr = lr
        self.epochs = epochs                           # 학습 횟수
        self.errors = []                               # 학습 과정에서의 오류 저장

    def activation(self, x):  # 이진 계단 함수
        return np.where(x > 0, 1, 0)

    def predict(self, x):
        linear_output = np.dot(x, self.weights) + self.bias # 선형 결합
        return self.activation(linear_output)               # 활성화 함수 통과

    def train(self, x, y):
        for epoch in range(self.epochs):
            total_error = 0
            for xi, target in zip(x, y):
                prediction = self.predict(xi)               # 현재 예측
                update = self.lr * (target - prediction)    # 가중치 조정량 계산
                self.weights += update * xi                 # 가중치 업데이트
                self.bias += update                         # 바이어스 업데이트
                total_error += abs(target - prediction)     # 오차 누적
            self.errors.append(total_error)
            print(f"Epoch {epoch+1}/{self.epochs}, Errors: {total_error}")
```

다층 퍼셉트론 구조

```python
#다중 퍼셉트론
import numpy as np
import matplotlib.pyplot as plt
from matplotlib.colors import ListedColormap

# 시그모이드 함수 및 도함수
def sigmoid(x):
    return 1 / (1 + np.exp(-x))

def sigmoid_derivative(x):
    return x * (1 - x)

# XOR 데이터
X = np.array([[0,0],[0,1],[1,0],[1,1]])
y = np.array([[0],[1],[1],[0]])

# 시드 고정
np.random.seed(42)

# 신경망 구조
input_size = 2
hidden_size = 4
output_size = 1
lr = 0.1
epochs = 10000

# 가중치 및 바이어스 초기화
w1 = np.random.randn(input_size, hidden_size)
b1 = np.zeros((1, hidden_size))
w2 = np.random.randn(hidden_size, output_size)
b2 = np.zeros((1, output_size))

loss_history = []

# 학습
for epoch in range(epochs):
    # 순전파
    z1 = np.dot(X, w1) + b1
    a1 = sigmoid(z1)
    z2 = np.dot(a1, w2) + b2
    y_pred = sigmoid(z2)

    # 손실 계산
    loss = np.mean((y - y_pred)**2)
    loss_history.append(loss)

    # 역전파
    d_loss_y = 2 * (y_pred - y)
    d_output = d_loss_y * sigmoid_derivative(y_pred)
    d_hidden = np.dot(d_output, w2.T) * sigmoid_derivative(a1)

    # 가중치 업데이트
    w2 -= lr * np.dot(a1.T, d_output)
    b2 -= lr * np.sum(d_output, axis=0, keepdims=True)
    w1 -= lr * np.dot(X.T, d_hidden)
    b1 -= lr * np.sum(d_hidden, axis=0, keepdims=True)

    # 로그 출력
    if (epoch + 1) % 1000 == 0 or epoch == 0:
        print(f"Epoch {epoch+1}/{epochs}, Loss: {loss:.6f}")
```

XOR gate 부연 설명

오류:

-Epoch 0

Input	Target	Prediction	Error	Update
[0,0]	0	1	-1	w += 0, b -= 0.1
[0,1]	1	1	0	no change
[1,0]	1	1	0	no change
[1,1]	0	1	-1	w -= 0.1, b -= 0.1

-Epoch 1

- [0, 0] → 0 (정답)
- [0, 1] → 1 (정답)
- [1, 0] → 1 (정답)
- [1, 1] → **0 or 1** (**틀릴 수도 있음**)

-Epoch 2,3,4,...
비선형 => 가중치 계속 변화 => 불안정한 결과 도출

오류 원인:

단층 퍼셉트론은 입력 공간을 단 하나의 직선(또는 초평면)으로 나눌 수 있어야 학습 가능하다. 즉, `[0,0]`과 `[1,1](input1, input2)`의 **결과는** 0이고 `[0,1]`, `[1,0]`은 1인데, 이 두 결과는 서로 다른 대각선에 위치한다. 그러므로 XOR GATE의 0과 1의 영역을 **한 개의 선**으로 구분하는 건 불가능하다.

결론: XOR GATE 를 단층 퍼셉트론으로 학습시키는 것은 불가능하다.

해결방안: 은닉층을 가지는 다층 퍼셉트론을 사용한다.

다층 퍼셉트론(MLP)이란?

구조: 입력층 → 은닉층(1개 이상) → 출력층

각 층(layer)은 여러 개의 노드(퍼셉트론)로 구성되며, 각 노드는 입력을 받아 가중치와 곱하고 합산한 뒤,비선형 활성화 함수를 거쳐 다음 층으로 전달하고 최종적으로 출력층에서 은닉층의 출력값을 선형 결합하여 결과를 도출해낸다.

항목	단층 퍼셉트론	다층 퍼셉트론 (MLP)
층 구조	입력층 → 출력층	입력층 → **은닉층** → 출력층
표현 가능성	선형 문제만 해결가능	비선형 문제 (XOR, etc.) 해결 가능
활성화 함수	계단 함수(혹은 선형)	**비선형 함수** (ReLU, Sigmoid 등)
학습 알고리즘	간단한 퍼셉트론 학습 규칙	**역전파** 사용

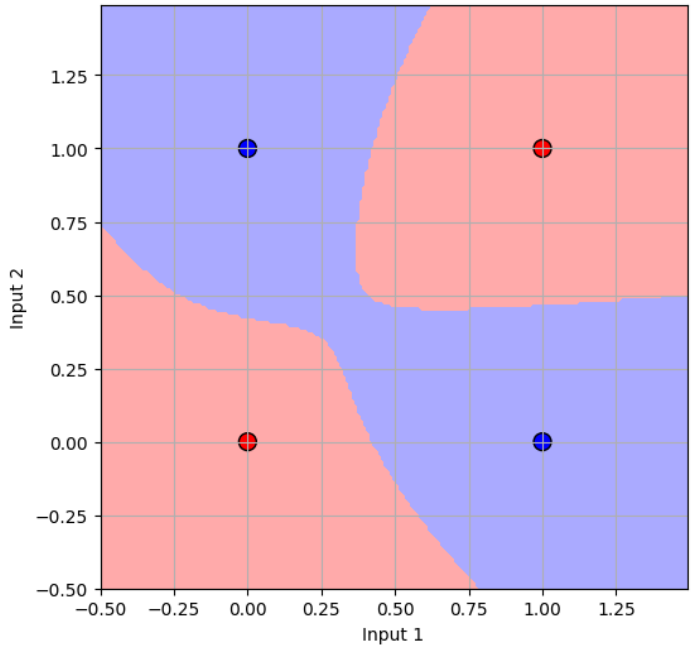
다층 퍼셉트론(sigmoid)을 사용한 결과

- 예측한 결과

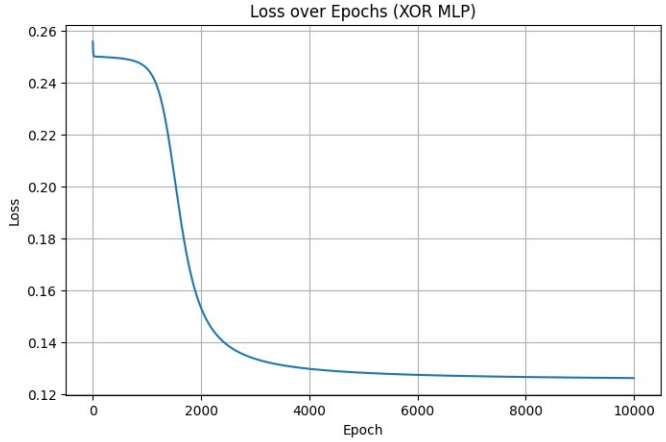
```
Epoch 1/10000, Loss: 0.283190
Epoch 1000/10000, Loss: 0.212605
Epoch 2000/10000, Loss: 0.057486
Epoch 3000/10000, Loss: 0.010708
Epoch 4000/10000, Loss: 0.004686
Epoch 5000/10000, Loss: 0.002788
Epoch 6000/10000, Loss: 0.001916
Epoch 7000/10000, Loss: 0.001430
Epoch 8000/10000, Loss: 0.001127
Epoch 9000/10000, Loss: 0.000921
Epoch 10000/10000, Loss: 0.000775

XOR 테스트 결과:
Input: [0 0], Predicted: 0.0197, Rounded: 0
Input: [0 1], Predicted: 0.9742, Rounded: 1
Input: [1 0], Predicted: 0.9700, Rounded: 1
Input: [1 1], Predicted: 0.0339, Rounded: 0
```

- 결정 경계 시각화



- 오류 시각화



참고

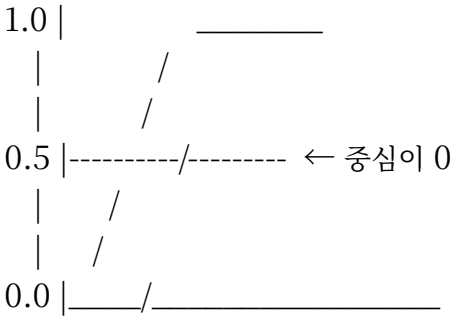
ReLU: f(x)=max(0,x)

- 입력 x가 **양수**면: 그대로 출력
- 입력 x가 **0 이하**면: 0으로 출력

장점: 계산간단, 딥러닝에서 성능 우수, 일정한 기울기
단점: 음수 입력 항상 0(학습 도중 일부 뉴런이 비활성화될 수 있음), 미분 불연속

Sigmoid: σ(x)=1/(1+e^(−x))

- 입력값이 클수록 (양수) → 출력은 1에 가까움
- 입력값이 작을수록 (음수) → 출력은 0에 가까움
- 입력값이 0이면 → 출력은 0.5



장점: 비선형성, 출력 범위 제한(0~1)
단점:

- 입력이 너무 크거나 작으면, 도함수가 0에 가까워져서 학습이 거의 되지 않음
- 중심이 0.5 → 출력값이 항상 양수 → 학습 시 편향 발생 가능
- 지수함수 연산 필요 → ReLU에 비해 느릴 수 있음

역전파: 신경망이 오차(error)를 계산하고, 이를 바탕으로 가중치(weight)를 조정하는 알고리즘
"모델이 잘못된 출력을 냈을 때, 그 실수를 추적해서 어디서 잘못된 건지 역으로 거슬러 올라가며 가중치를 수정하는 방법"

출처

chat gpt

Minsky, Marvin & Papert, Seymour (1969). *Perceptrons*
Goodfellow, Bengio, Courville (2016). Deep Learning (MIT Press)
Glorot, Xavier, & Bengio (2010). "Understanding the difficulty of training deep feedforward neural networks"
PyTorch & TensorFlow 공식 문서
Rumelhart, Hinton, and Williams (1986). "Learning representations by back-propagating errors"
Michael Nielsen – Neural Networks and Deep Learning (2015)