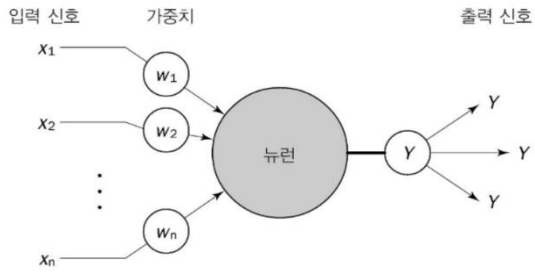
# 퍼셉트론 알고리즘



[그림 6-3] 뉴런의 도식

**퍼셉트론**이란 로젠 블랫이 제안한 인공신경망의 한 종류로 인공 뉴런이 여러 개가 모여 구성된 모델형이며 입력층과 출력층으로 이루어져 있다.

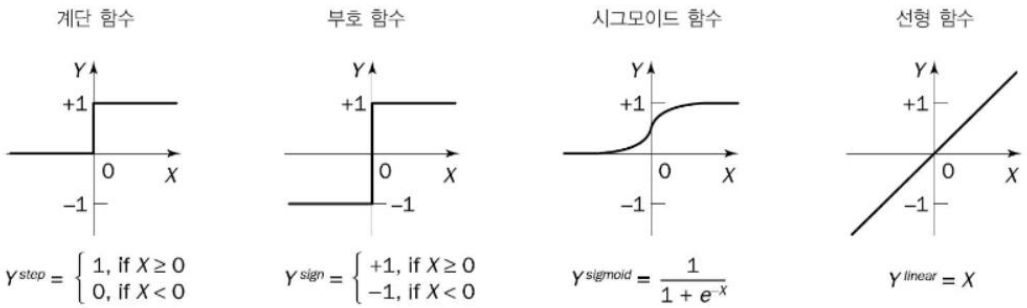
입력층을 통해 입력된 값에 가중치값을 곱해 나온 값을 활성화 함수를 거쳐 결과가 출력층으로 나오게 되는 것이다. 이러한 방법으로 가중치를 조절하여 반복을 통해 **실제출력과 목표출력 간의 차이를 줄여나가는 것이 최종 목표**이다.

퍼셉트론 훈련 알고리즘은 총 4단계로 **초기화, 활성화, 가중치 학습, 반복**으로 이루어져 있다.

**초기화**란 말 그대로 실험에 필요한 값들을 초기화하는 단계로 초기가중치 w1,w2,. ... ,wn 과 임계값 θ 를 [−0.5, 0.5] 구간의 임의의 값으로 설정한다.

**활성화**란 입력 x1(p), x2(p), . . ., xn(p)와 목표출력 Yd(p) 를 적용하여 퍼셉트론을 활성화하는 단계로 입력값과 가중치 값을 곱하여 더한 값을 활성화함수에 대입하여 출력값을 얻어낸다.

- **뉴런의 활성화 함수**
  - **가장 일반적인 활성화 함수로는 계단, 부호, 선형, 시그모이드 함수가 있다. : [그림 6-4]**



[그림 6-4] 뉴런의 활성화 함수들

**가중치 학습**이란 퍼셉트론의 가중치를 갱신하는 단계로 목표 출력과 실제 출력의 오차를 가지고 학습률과 오차와 입력값의 곱을 통해 결과가중치를 얻어낸다. 그 후 초기 가중치와 얻어낸 결과가중치의 차를 통해 최종가중치를 나타내고 다음 반복에 이 최종가중치를 사용한다.

**반복**은 말그대로 목표출력과 실제출력의 오차를 줄이기 위해 반복횟수 p값을 1 증가시키고, 2단계로 돌아가서 수렴할 때까지 과정을 반복하는 것을 의미한다.

# 퍼셉트론 알고리즘 코드 정리

**필요한 라이브러리를 호출한다. ( numpy: 수학적 연산을 수행하기 위한 라이브러리)**

```
import numpy as np
```

**연산에 필요한 변수를 선언하고 초기화한다. (inputs: 입력값(00 01 10 11) / targets: 각 입력값에 따른 목표 출력(사진은 AND 논리연산자의 목표출력이며 OR, XOR의 경우 각각 0111과 0110으로 수정해준다.) / initial_weights: 초기가중치 / bias: 임계값 / learning_rate: 학습률 )**

```
inputs = np.array([[0, 0], [0, 1], [1, 0], [1, 1]])
targets = np.array([0, 0, 0, 1])
initial_weights = np.array([0.3, -0.1])
bias = 0.2
learning_rate = 0.1
```

**오차 계산을 위한 활성화 함수는 step함수를 사용할 것이다. (step함수: 입력값이 0보다 크거나 같을 시 1 반환, 0보다 작을 시 0반환)**

```
def step_function(x):
    return 1 if x >= 0 else 0


# 에폭의 최대값 설정
max_epochs = 10

for epoch in range(1, max_epochs + 1):
    total_error = 0

    # 각 입력에 대한 계산과 최종가중치 업데이트
    for i in range(len(inputs)):
        input_data = inputs[i]
        target = targets[i]

        # 실제 출력 계산
        actual_output = np.dot(input_data, initial_weights) - bias
        actual_output = round(actual_output, 1)
        prediction = step_function(actual_output)

        # 오차 계산
        error = target - prediction
        total_error += abs(error)  # 오차의 절댓값을 누적

        # 최종가중치 업데이트
        initial_weights += learning_rate * error * input_data

        # 에폭별 입력값에 따른 결과 출력
        print(f"Epoch {epoch}, Input: {input_data}, Target: {target}, Actual Output: {ac
tual_output}, Prediction: {prediction}, Error: {error}")
        print(f"  Updated Weights: {initial_weights}₩n")

    # 모든 입력에 대한 누적된 오차가 0이면 종료
    if total_error == 0:
        print(f"Converged at Epoch {epoch}")
        break
    # 진행하면서 에폭이 최대 에폭 설정값과 동일하며 오차가 계속하여 존재할 경우
    if epoch == max_epochs and total_error != 0:
            print("Maximum epochs reached. Did not converge to zero error.")
```

**에폭의 최대값을 10으로 임의의로 설정 (에폭마다 00 01 10 11의 입력값을 가진다)**

**각 에폭의 오류를 누적시킬 total_error를 선언.**

**input_data: 00 01 10 11의 입력값 /target: 입력값에 대한 논리연산자 목표 출력**

**actual_output: step함수에 값을 넣기 위한 x1*w1+x2*w2 – bias의 연산을 가진다.(numpy 라이브러리함수np.dot을 이용하여 백터의 각 성분을 곱한 후 값들을 더해준다.)**

이렇게 얻은 값은 부동소수점 정밀도의 문제로 원하는 값이 나오지 않을 수 있기 때문에 round 함수를 통해 반올림하여 값을 얻는다.

이제 나온 값을 step함수에 넣어 실제출력값을 얻는다.

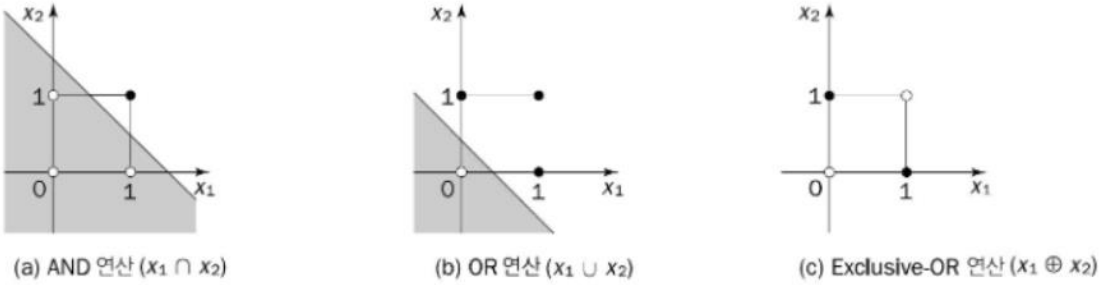
initial_weights 업데이트: 최종가중치는 초기가중치에서 결과가중치를 더한 값으로 학습률 * 입력값 * 오차이다. 에폭을 진행시킬때마다 이전 에폭의 결과가중치를 다음 에폭의 초기가중치로 사용할 것이기 때문에 +=을 통해 initial_weights를 초기화시켜준다.

마지막으로 에폭의 00 01 10 11의 입력이 종료됐을 때 누적된 오차가 0으로 수렴할 경우 반복문을 종료시킨다. 만약 에폭의 최대값까지 진행했음에도 누적된 오차가 0으로 수렴하지 않을 경우 도달하지 못했다는 에러메시지를 보여준다.

따라서 원하는 값이 나올 수 있도록 에폭의 최대값을 수정해주면 된다.

# 퍼셉트론 알고리즘 결과

### <AND / OR / XOR 논리연산자의 2차원 도면>




### <AND 논리연산자>

### AND 논리 연산자의 경우 에폭 5에서 종료되며 최종가중치는 0.1 / 0.1 이다.

```
Epoch 1, Input: [0 0], Target: 0, Actual Output: -0.2, Prediction: 0, Error: 0
  Updated Weights: [ 0.3 -0.1]

Epoch 1, Input: [0 1], Target: 0, Actual Output: -0.3, Prediction: 0, Error: 0
  Updated Weights: [ 0.3 -0.1]

Epoch 1, Input: [1 0], Target: 0, Actual Output: 0.1, Prediction: 1, Error: -1
  Updated Weights: [ 0.2 -0.1]

Epoch 1, Input: [1 1], Target: 1, Actual Output: -0.1, Prediction: 0, Error: 1
  Updated Weights: [0.3 0. ]

Epoch 2, Input: [0 0], Target: 0, Actual Output: -0.2, Prediction: 0, Error: 0
  Updated Weights: [0.3 0. ]

Epoch 2, Input: [0 1], Target: 0, Actual Output: -0.2, Prediction: 0, Error: 0
  Updated Weights: [0.3 0. ]

Epoch 2, Input: [1 0], Target: 0, Actual Output: 0.1, Prediction: 1, Error: -1
  Updated Weights: [0.2 0. ]

Epoch 2, Input: [1 1], Target: 1, Actual Output: -0.0, Prediction: 1, Error: 0
  Updated Weights: [0.2 0. ]

Epoch 3, Input: [0 0], Target: 0, Actual Output: -0.2, Prediction: 0, Error: 0
  Updated Weights: [0.2 0. ]

Epoch 3, Input: [0 1], Target: 0, Actual Output: -0.2, Prediction: 0, Error: 0
  Updated Weights: [0.2 0. ]

Epoch 3, Input: [1 0], Target: 0, Actual Output: -0.0, Prediction: 1, Error: -1
  Updated Weights: [0.1 0. ]

Epoch 3, Input: [1 1], Target: 1, Actual Output: -0.1, Prediction: 0, Error: 1
  Updated Weights: [0.2 0.1]

Epoch 4, Input: [0 0], Target: 0, Actual Output: -0.2, Prediction: 0, Error: 0
  Updated Weights: [0.2 0.1]

Epoch 4, Input: [0 1], Target: 0, Actual Output: -0.1, Prediction: 0, Error: 0
  Updated Weights: [0.2 0.1]

Epoch 4, Input: [1 0], Target: 0, Actual Output: -0.0, Prediction: 1, Error: -1
  Updated Weights: [0.1 0.1]

Epoch 4, Input: [1 1], Target: 1, Actual Output: -0.0, Prediction: 1, Error: 0
  Updated Weights: [0.1 0.1]

Epoch 5, Input: [0 0], Target: 0, Actual Output: -0.2, Prediction: 0, Error: 0
  Updated Weights: [0.1 0.1]

Epoch 5, Input: [0 1], Target: 0, Actual Output: -0.1, Prediction: 0, Error: 0
  Updated Weights: [0.1 0.1]

Epoch 5, Input: [1 0], Target: 0, Actual Output: -0.1, Prediction: 0, Error: 0
  Updated Weights: [0.1 0.1]

Epoch 5, Input: [1 1], Target: 1, Actual Output: -0.0, Prediction: 1, Error: 0
  Updated Weights: [0.1 0.1]

Converged at Epoch 5
```

## <OR 논리연산자>

## OR 논리 연산자의 경우 에폭 4에서 종료되며 최종가중치는 0.3 / 0.2 이다.

```
Epoch 1, Input: [0 0], Target: 0, Actual Output: -0.2, Prediction: 0, Error: 0
  Updated Weights: [ 0.3 -0.1]

Epoch 1, Input: [0 1], Target: 1, Actual Output: -0.3, Prediction: 0, Error: 1
  Updated Weights: [0.3 0. ]

Epoch 1, Input: [1 0], Target: 1, Actual Output: 0.1, Prediction: 1, Error: 0
  Updated Weights: [0.3 0. ]

Epoch 1, Input: [1 1], Target: 1, Actual Output: 0.1, Prediction: 1, Error: 0
  Updated Weights: [0.3 0. ]

Epoch 2, Input: [0 0], Target: 0, Actual Output: -0.2, Prediction: 0, Error: 0
  Updated Weights: [0.3 0. ]

Epoch 2, Input: [0 1], Target: 1, Actual Output: -0.2, Prediction: 0, Error: 1
  Updated Weights: [0.3 0.1]

Epoch 2, Input: [1 0], Target: 1, Actual Output: 0.1, Prediction: 1, Error: 0
  Updated Weights: [0.3 0.1]

Epoch 2, Input: [1 1], Target: 1, Actual Output: 0.2, Prediction: 1, Error: 0
  Updated Weights: [0.3 0.1]

Epoch 3, Input: [0 0], Target: 0, Actual Output: -0.2, Prediction: 0, Error: 0
  Updated Weights: [0.3 0.1]

Epoch 3, Input: [0 1], Target: 1, Actual Output: -0.1, Prediction: 0, Error: 1
  Updated Weights: [0.3 0.2]

Epoch 3, Input: [1 0], Target: 1, Actual Output: 0.1, Prediction: 1, Error: 0
  Updated Weights: [0.3 0.2]

Epoch 3, Input: [1 1], Target: 1, Actual Output: 0.3, Prediction: 1, Error: 0
  Updated Weights: [0.3 0.2]

Epoch 4, Input: [0 0], Target: 0, Actual Output: -0.2, Prediction: 0, Error: 0
  Updated Weights: [0.3 0.2]

Epoch 4, Input: [0 1], Target: 1, Actual Output: 0.0, Prediction: 1, Error: 0
  Updated Weights: [0.3 0.2]

Epoch 4, Input: [1 0], Target: 1, Actual Output: 0.1, Prediction: 1, Error: 0
  Updated Weights: [0.3 0.2]

Epoch 4, Input: [1 1], Target: 1, Actual Output: 0.3, Prediction: 1, Error: 0
  Updated Weights: [0.3 0.2]

Converged at Epoch 4
```

## <XOR 논리연산자>

## XOR 논리 연산자의 경우 에폭이 최대값 10에 도달했음에도 오차가 0으로 수렴하지 않는다.

```
Epoch 1, Input: [0 0], Target: 0, Actual Output: -0.2, Prediction: 0, Error: 0
  Updated Weights: [ 0.3 -0.1]

Epoch 1, Input: [0 1], Target: 1, Actual Output: -0.3, Prediction: 0, Error: 1
  Updated Weights: [0.3 0. ]

Epoch 1, Input: [1 0], Target: 1, Actual Output: 0.1, Prediction: 1, Error: 0
  Updated Weights: [0.3 0. ]

Epoch 1, Input: [1 1], Target: 0, Actual Output: 0.1, Prediction: 1, Error: -1
  Updated Weights: [ 0.2 -0.1]

Epoch 2, Input: [0 0], Target: 0, Actual Output: -0.2, Prediction: 0, Error: 0
  Updated Weights: [ 0.2 -0.1]

Epoch 2, Input: [0 1], Target: 1, Actual Output: -0.3, Prediction: 0, Error: 1
  Updated Weights: [0.2 0. ]

Epoch 2, Input: [1 0], Target: 1, Actual Output: -0.0, Prediction: 1, Error: 0
  Updated Weights: [0.2 0. ]

Epoch 2, Input: [1 1], Target: 0, Actual Output: -0.0, Prediction: 1, Error: -1
  Updated Weights: [ 0.1 -0.1]

Epoch 3, Input: [0 0], Target: 0, Actual Output: -0.2, Prediction: 0, Error: 0
  Updated Weights: [ 0.1 -0.1]

Epoch 3, Input: [0 1], Target: 1, Actual Output: -0.3, Prediction: 0, Error: 1
  Updated Weights: [0.1 0. ]

Epoch 3, Input: [1 0], Target: 1, Actual Output: -0.1, Prediction: 0, Error: 1
  Updated Weights: [0.2 0. ]

Epoch 3, Input: [1 1], Target: 0, Actual Output: -0.0, Prediction: 1, Error: -1
  Updated Weights: [ 0.1 -0.1]

Epoch 4, Input: [0 0], Target: 0, Actual Output: -0.2, Prediction: 0, Error: 0
  Updated Weights: [ 0.1 -0.1]

Epoch 4, Input: [0 1], Target: 1, Actual Output: -0.3, Prediction: 0, Error: 1
  Updated Weights: [0.1 0. ]

Epoch 4, Input: [1 0], Target: 1, Actual Output: -0.1, Prediction: 0, Error: 1
  Updated Weights: [0.2 0. ]

Epoch 4, Input: [1 1], Target: 0, Actual Output: -0.0, Prediction: 1, Error: -1
  Updated Weights: [ 0.1 -0.1]

Epoch 5, Input: [0 1], Target: 1, Actual Output: -0.3, Prediction: 0, Error: 1
  Updated Weights: [0.1 0. ]

Epoch 5, Input: [1 0], Target: 1, Actual Output: -0.1, Prediction: 0, Error: 1
  Updated Weights: [0.2 0. ]

Epoch 5, Input: [1 1], Target: 0, Actual Output: -0.0, Prediction: 1, Error: -1
  Updated Weights: [ 0.1 -0.1]

Epoch 6, Input: [0 0], Target: 0, Actual Output: -0.2, Prediction: 0, Error: 0
  Updated Weights: [ 0.1 -0.1]

Epoch 6, Input: [0 1], Target: 1, Actual Output: -0.3, Prediction: 0, Error: 1
  Updated Weights: [0.1 0. ]

Epoch 6, Input: [1 0], Target: 1, Actual Output: -0.1, Prediction: 0, Error: 1
  Updated Weights: [0.2 0. ]

Epoch 6, Input: [1 1], Target: 0, Actual Output: -0.0, Prediction: 1, Error: -1
  Updated Weights: [ 0.1 -0.1]

Epoch 7, Input: [0 0], Target: 0, Actual Output: -0.2, Prediction: 0, Error: 0
  Updated Weights: [ 0.1 -0.1]

Epoch 7, Input: [0 1], Target: 1, Actual Output: -0.3, Prediction: 0, Error: 1
  Updated Weights: [0.1 0. ]

Epoch 7, Input: [1 0], Target: 1, Actual Output: -0.1, Prediction: 0, Error: 1
  Updated Weights: [0.2 0. ]

Epoch 7, Input: [1 1], Target: 0, Actual Output: -0.0, Prediction: 1, Error: -1
  Updated Weights: [ 0.1 -0.1]

Epoch 8, Input: [0 0], Target: 0, Actual Output: -0.2, Prediction: 0, Error: 0
  Updated Weights: [ 0.1 -0.1]

Epoch 8, Input: [0 1], Target: 1, Actual Output: -0.3, Prediction: 0, Error: 1
  Updated Weights: [0.1 0. ]

Epoch 8, Input: [1 0], Target: 1, Actual Output: -0.1, Prediction: 0, Error: 1
  Updated Weights: [0.2 0. ]

Epoch 8, Input: [1 1], Target: 0, Actual Output: -0.0, Prediction: 1, Error: -1
  Updated Weights: [ 0.1 -0.1]

Epoch 9, Input: [0 0], Target: 0, Actual Output: -0.2, Prediction: 0, Error: 0
  Updated Weights: [ 0.1 -0.1]

Epoch 9, Input: [0 1], Target: 1, Actual Output: -0.3, Prediction: 0, Error: 1
  Updated Weights: [0.1 0. ]

Epoch 9, Input: [1 0], Target: 1, Actual Output: -0.1, Prediction: 0, Error: 1
  Updated Weights: [0.2 0. ]

Epoch 9, Input: [1 1], Target: 0, Actual Output: -0.0, Prediction: 1, Error: -1
  Updated Weights: [ 0.1 -0.1]

Epoch 10, Input: [0 0], Target: 0, Actual Output: -0.2, Prediction: 0, Error: 0
  Updated Weights: [ 0.1 -0.1]

Epoch 10, Input: [0 1], Target: 1, Actual Output: -0.3, Prediction: 0, Error: 1
  Updated Weights: [0.1 0. ]

Epoch 10, Input: [1 0], Target: 1, Actual Output: -0.1, Prediction: 0, Error: 1
  Updated Weights: [0.2 0. ]

Epoch 10, Input: [1 1], Target: 0, Actual Output: -0.0, Prediction: 1, Error: -1
  Updated Weights: [ 0.1 -0.1]

Maximum epochs reached. Did not converge to zero error.
```

## 그 이유는 XOR 논리 연산자는 선형 분리가 불가능하기 때문에 단층 퍼셉트론으로는 학습할 수 없다.

## 이러한 결과는 단층 퍼셉트론은 선형 분리가 가능한 함수만 학습할 수 있다는 결과를 보여준다.