AI 프로그래밍

- 6주차

인하 공전 컴퓨터 정보과

민 정혜

오늘 수업 순서

- 퍼셉트론, MLP
- MLP 계산
- 순방향 패스, 역방향 패스
- 분류 VS. 회귀
- 딥러닝
- 전체 리뷰

퍼셉트론 (Perceptron)

퍼셉트론(perceptron): 인공 뉴런 은 1957년에 로젠블라트(Frank Rosenblatt)가 고안한 인공 신경망이다.

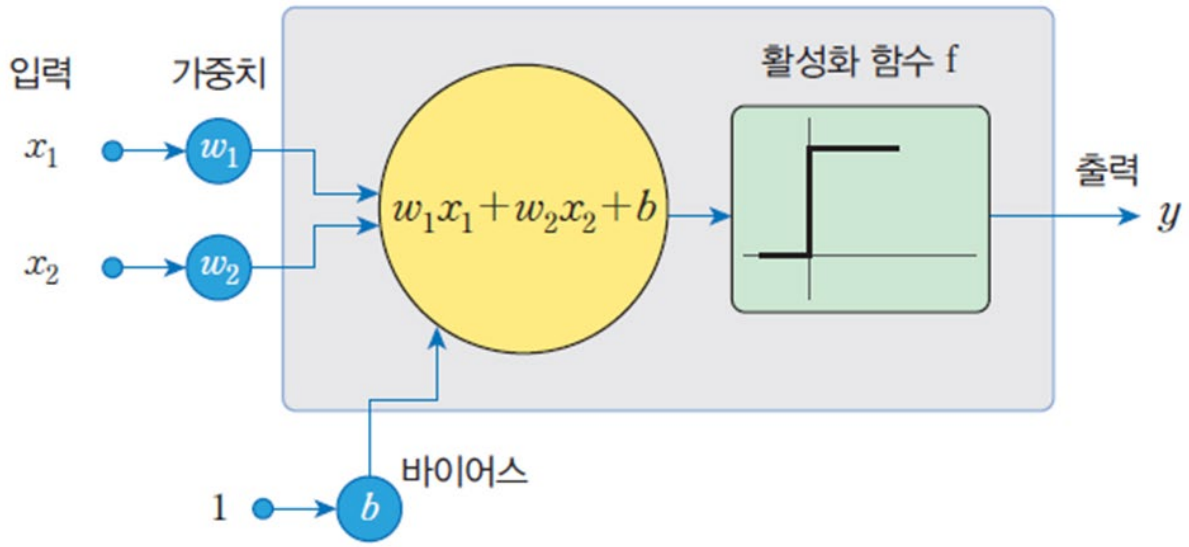


그림 5-5 퍼셉트론에서의 뉴런의 모델

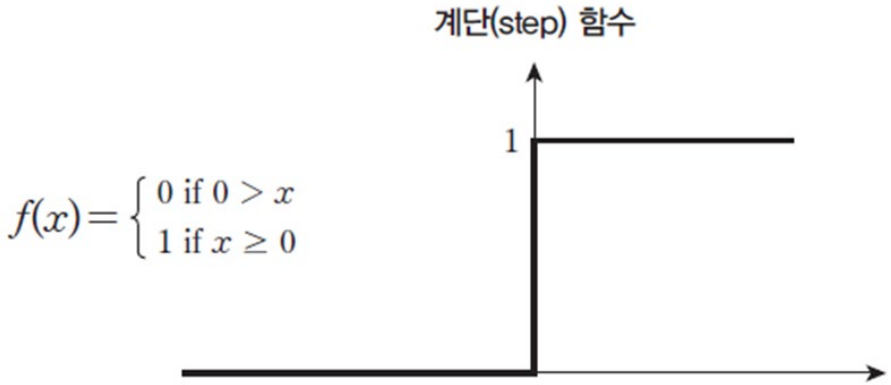


그림 5-7 퍼셉트론에서의 활성화 함수

-다수의 입력 신호 를 받아서 하나의 신호를 출력한다.
- 가중치는 입력 신호가 출력에 미치는 중요도를 조절
- 바이어스는 뉴런이 얼마나 쉽게 활성화 되는지 결정
- 가중합 : w1*x1+w2*x2+b
- 활성화 함수는 가중합의 결과를 놓고 0, 1을 판단

퍼셉트론 (Perceptron)

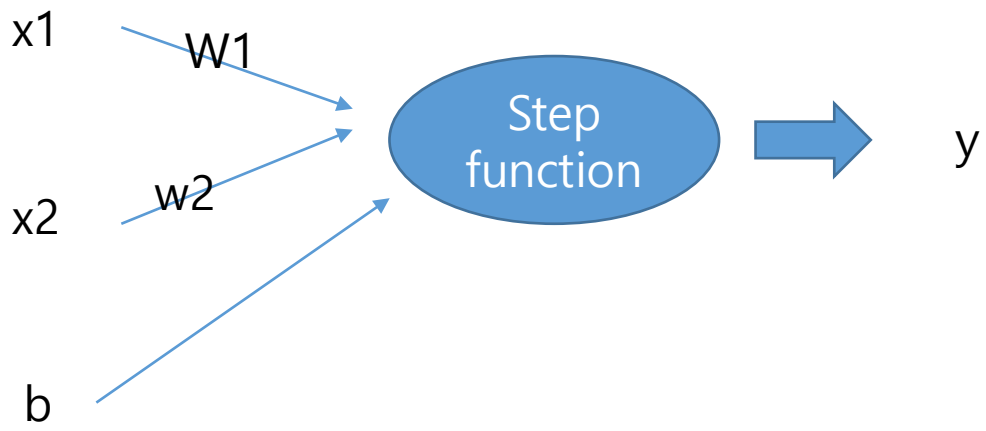
- 뉴런에서는 입력 신호의 가중치 합이 어떤 임계값을 넘는 경우에만 뉴런이 활성화되어서 1을 출력한다. 그렇지 않으면 0을 출력한다.

$$y = \begin{cases} 1 & \text{if } (w_1 x_1 + w_2 x_2 + b \geq 0) \\ 0 & otherwise \end{cases}$$

퍼셉트론 (Perceptron)

입력 (1,0)

w1: 1
w2: 1
B=-1.5



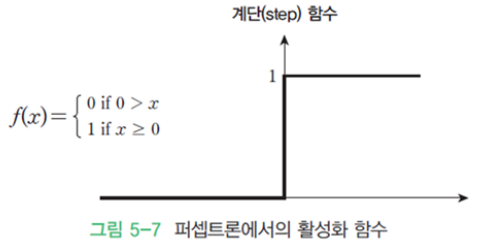


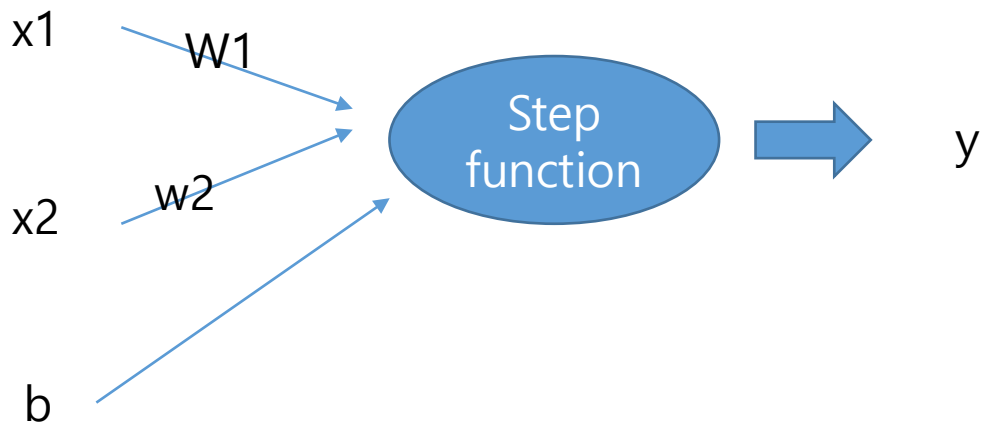
그림 5-7 퍼셉트론에서의 활성화 함수

출력값 y는?

퍼셉트론 (Perceptron)

입력 (1,0)

w1: 1
w2: 1
B=-1.5



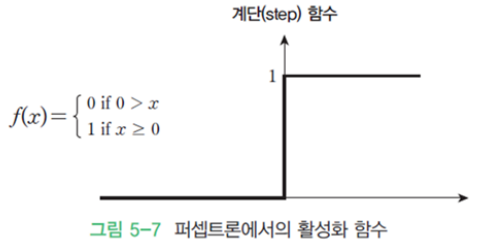


그림 5-7 퍼셉트론에서의 활성화 함수

출력값 y는? x1*w1+x2*w2+b= -0.5 -> step function -> 0

퍼셉트론 (Perceptron) – 논리 연산 수행

AND 연산

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

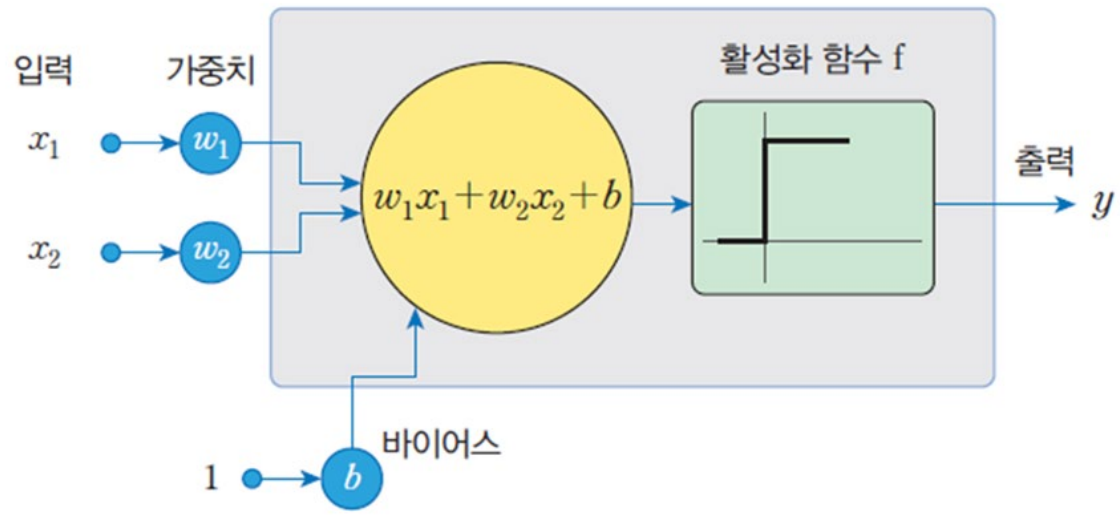


그림 5-5 퍼셉트론에서의 뉴런의 모델

만족시키는 w1,w2,b 찾기

w1=1,w2=1,b=-1.5 를 테스트 해보기

퍼셉트론 (Perceptron) - 논리 연산

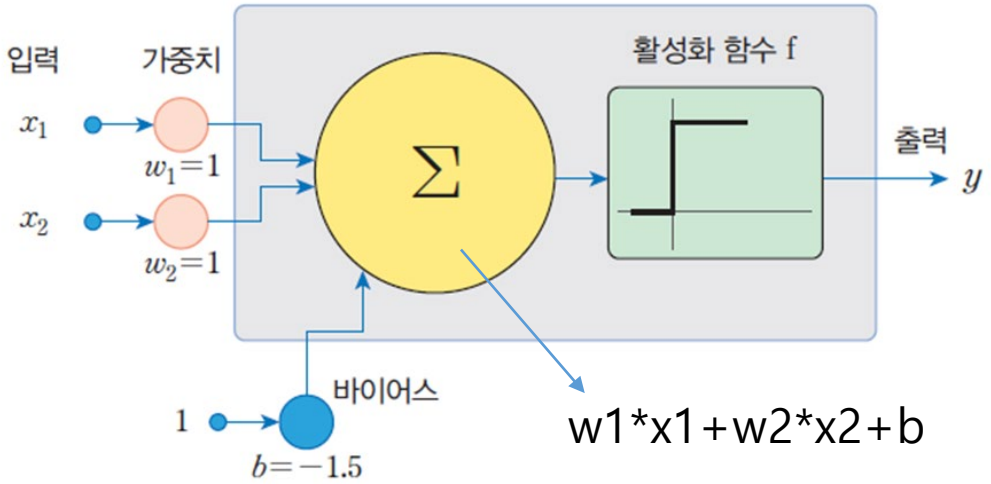


그림 5-6 논리 연산을 하는 퍼셉트론

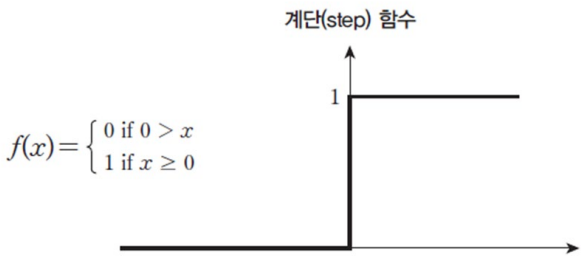


그림 5-7 퍼셉트론에서의 활성화 함수

x_1	x_2
0	0
1	0
0	1
1	1

?

가중치의 합

Y

퍼셉트론 (Perceptron) - 논리 연산

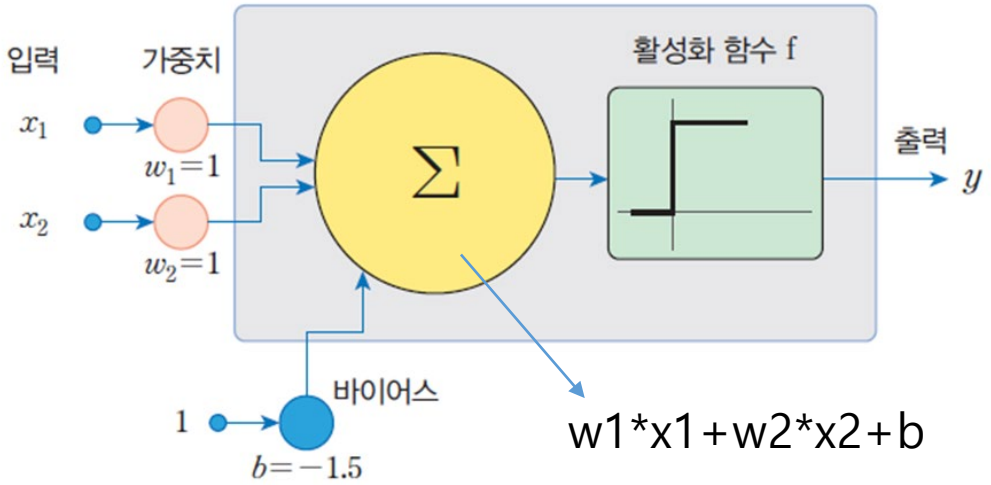


그림 5-6 논리 연산을 하는 퍼셉트론

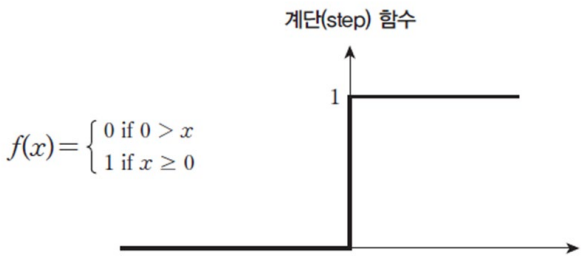


그림 5-7 퍼셉트론에서의 활성화 함수

x_1	x_2	$w_1x_1+w_2x_2$	b
0	0	1*0+1*0=0	−1.5
1	0	1*1+1*0=1	−1.5
0	1	1*0+1*1=1	−1.5
1	1	1*1+1*1=2	−1.5

-1.5
-0.5
-0.5
0.5

활성화 함수

y
0
0
0
1

AND 연산을 만족

퍼셉트론 (Perceptron) 구현

```
epsilon = 0.0000001

def perceptron(x1, x2):
    w1, w2, b = 1.0, 1.0, -1.5
    sum = x1*w1+x2*w2+b
    if sum > epsilon :                    # 부동소수점 오차를 방지하기 위하여
        return 1
    else :
        return 0

print(perceptron(0, 0))
print(perceptron(1, 0))
print(perceptron(0, 1))
print(perceptron(1, 1))
```

w1=1.0, w2=1.0,b=-0.5 일때 출력 구하기

1) 입력이 (0,0) 일 때
2) 입력이 (1,0) 일 때
3) 입력이 (0,1) 일 때
4) 입력이 (1,1) 일 때

퍼셉트론 (Perceptron) 학습

- 학습이라고 부르려면 신경망이 스스로 가중치를 자동으로 설정해주는 알고리즘이 필요하다. 퍼셉트론에서도 학습 알고리즘이 존재한다.

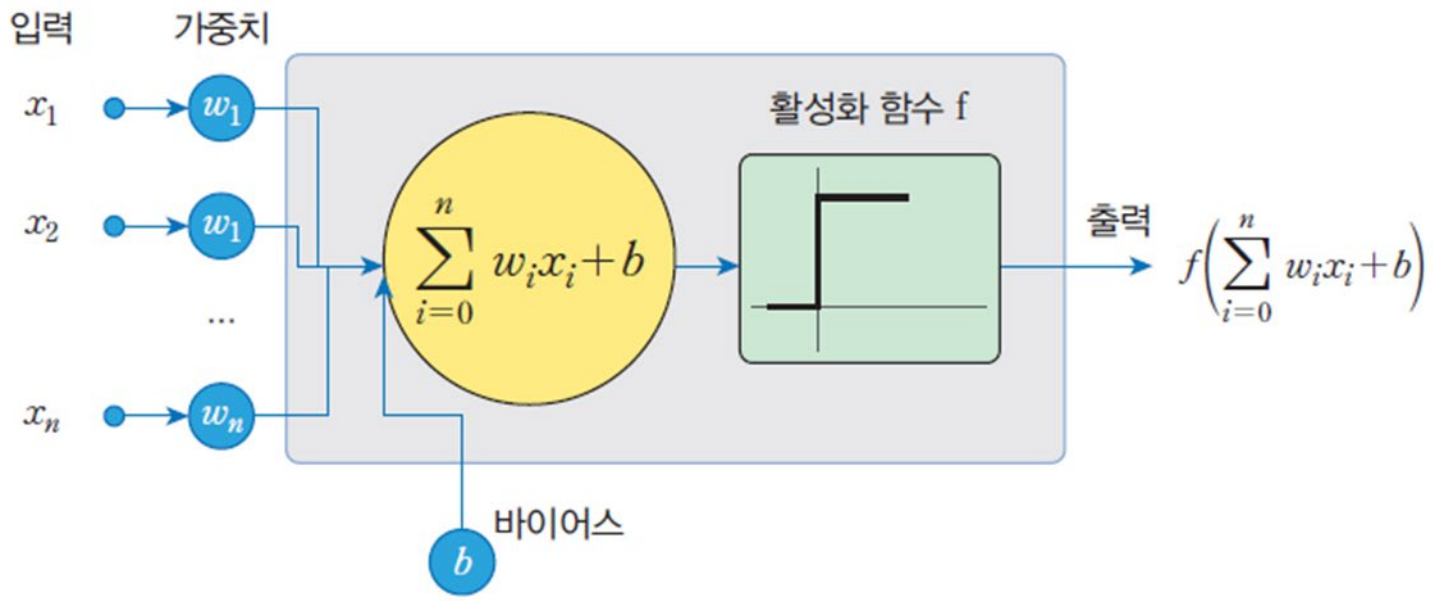


그림 5-8 퍼셉트론

퍼셉트론 (Perceptron) 학습 –scikit learn

- 학습이라고 부르려면 신경망이 스스로 가중치를 자동으로 설정해주는 알고리즘이 필요하다. 퍼셉트론에서도 학습 알고리즘이 존재한다.

```
from sklearn.linear_model import Perceptron

# 샘플과 레이블이다.
X = [[0,0],[0,1],[1,0],[1,1]]
y = [0, 0, 0, 1]

# 퍼셉트론을 생성한다. tol는 종료 조건이다. random_state는 난수의 시드이다.
clf = Perceptron(tol=1e-3, random_state=0)

# 학습을 수행한다.
clf.fit(X, y)

# 테스트를 수행한다.
print(clf.predict(X))
```

```
?
```

퍼셉트론 (Perceptron) 학습 OR 연산

x_1	x_2	Y
0	0	0
1	0	1
0	1	1
1	1	1

OR

SCIKIT LEARN 학습

퍼셉트론 (Perceptron) 학습 OR 연산

x_1	x_2	Y
0	0	0
1	0	1
0	1	1
1	1	1

OR

```
from sklearn.linear_model import Perceptron

# 샘플과 레이블이다.
X = [[0,0],[0,1],[1,0],[1,1]]
y = [0, 1, 1, 1]

# 퍼셉트론을 생성한다. tol는 종료 조건이다. random_state는 난수의 시드이다.
clf = Perceptron(tol=1e-3, random_state=0)

# 학습을 수행한다.
clf.fit(X, y)

# 테스트를 수행한다.
print(clf.predict(X))
```

```
?
```

퍼셉트론 (Perceptron) 학습 XOR 연산

x1	x2	y
0	0	0
1	0	1
0	1	1
1	1	0

XOR 연산을 수행하는 퍼셉트론 학습

퍼셉트론 (Perceptron) 학습 XOR 연산

x1	x2	y
0	0	0
1	0	1
0	1	1
1	1	0

XOR 연산을 수행하는 퍼셉트론 학습

```
from sklearn.linear_model import Perceptron

# 샘플과 레이블이다.
X = [[0,0],[0,1],[1,0],[1,1]]
y = [0, 1, 1, 0]

# 퍼셉트론을 생성한다. tol는 종료 조건이다. random_state는 난수의 시드이다.
clf = Perceptron(tol=1e-3, random_state=0)

# 학습을 수행한다.
clf.fit(X, y)

# 테스트를 수행한다.
print(clf.predict(X))
```

```
[0 0 0 0]
```

퍼셉트론 (Perceptron)

- XOR 연산은 퍼셉트론으로 학습이 불가능 하다
- 퍼셉트론도 w*x+b 의 형태. 직선을 이용한 분류.

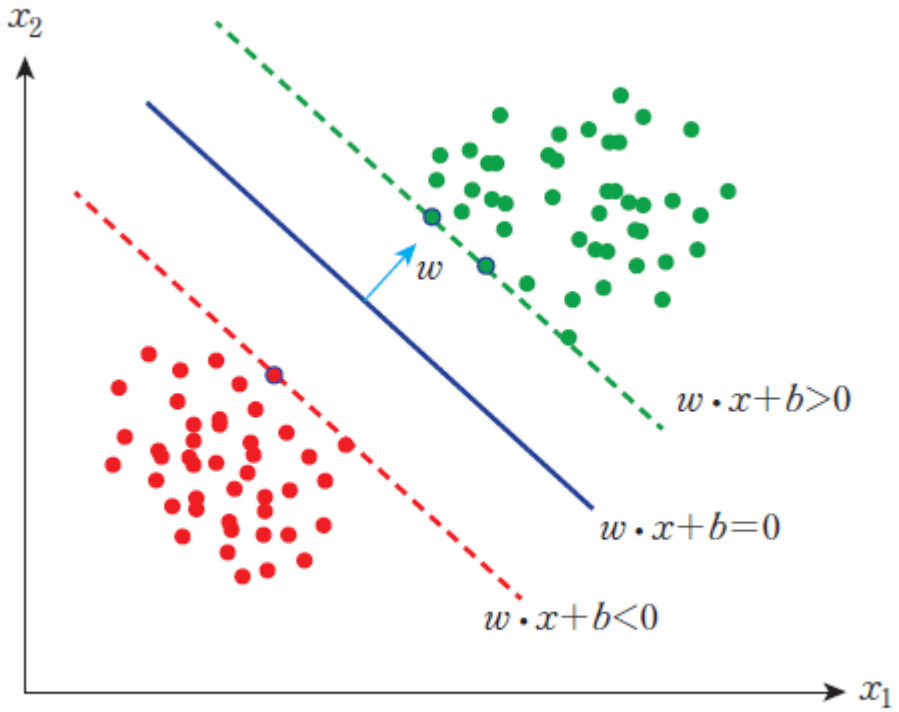


그림 5-10 선형 분류자

활성화 함수

-활성화 함수(activation function)은 입력의 총합을 받아서 출력값을 계산하는 함수이다
-MLP에서는 다양한 활성화 함수를 사용한다.

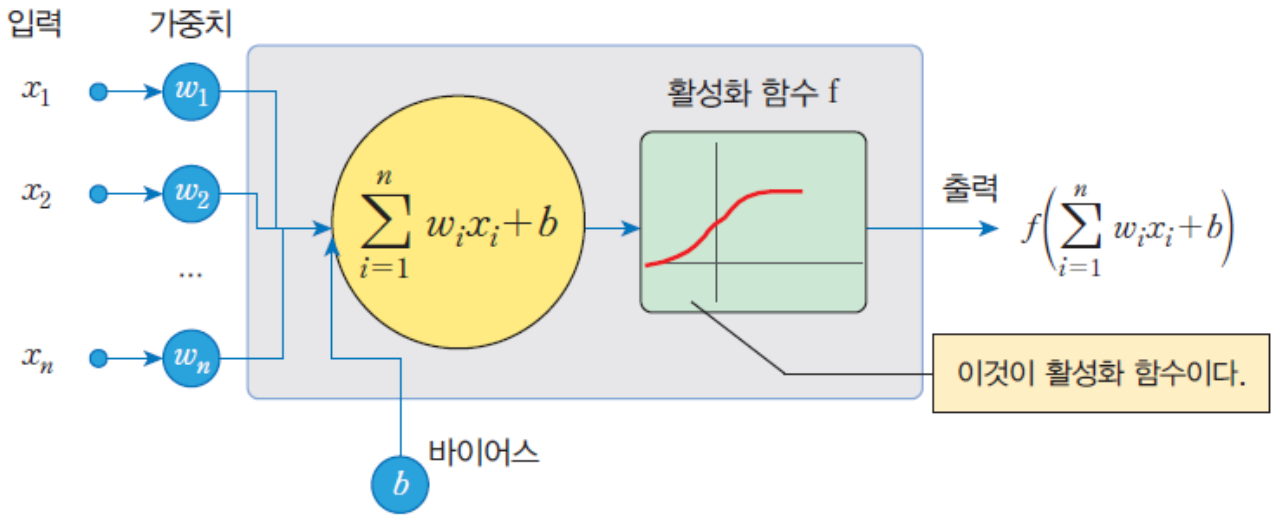


그림 6-2 활성화 함수

활성화 함수

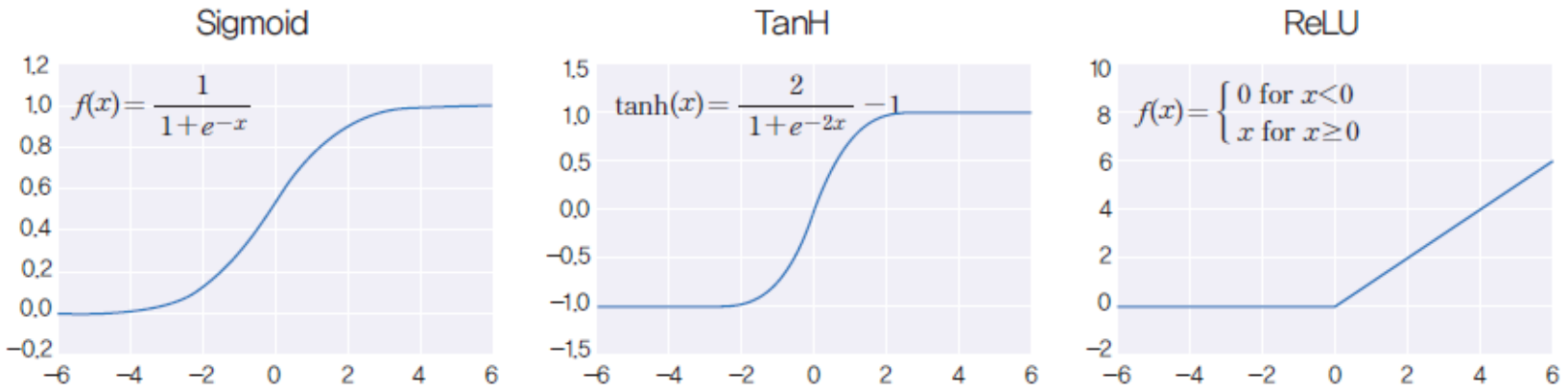
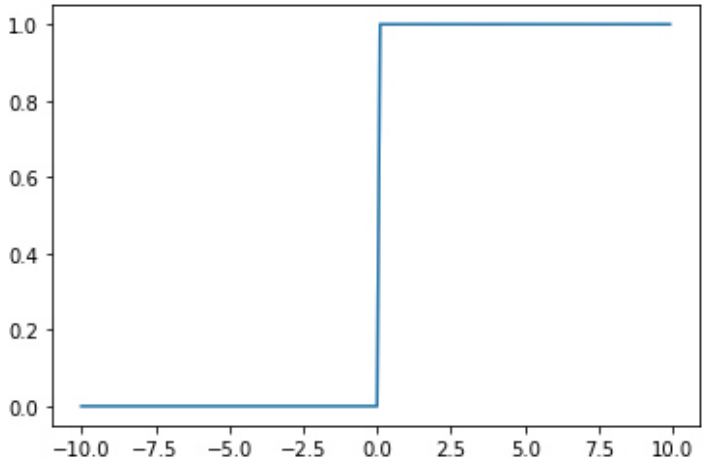


그림 6-3 많이 사용되는 활성화 함수

활성화 함수 (step)

- 계단 함수는 입력 신호의 총합이 0을 넘으면 1을 출력하고, 그렇지 않으면 0을 출력하는 함수이다.

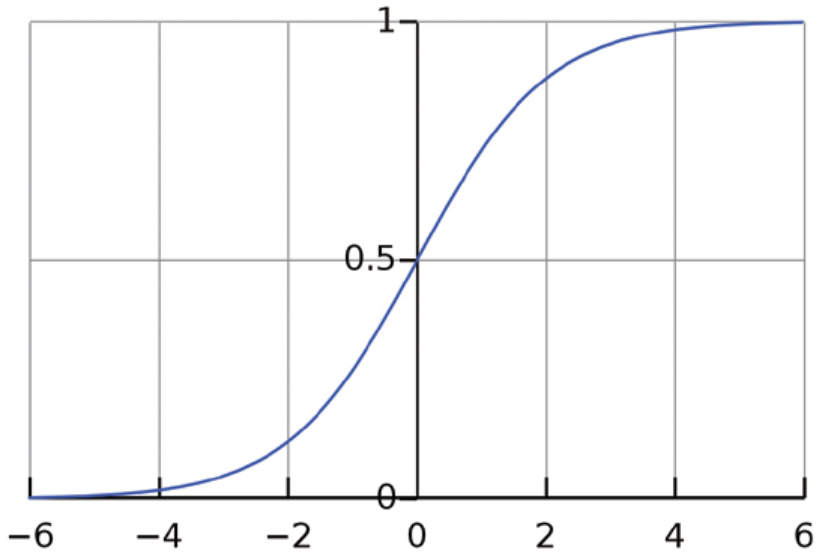
$$f(x) = \begin{cases} 1 & x > 0 \\ 0 & x \leq 0 \end{cases}$$



활성화 함수 (시그모이드)

- 1980년대부터 사용돼온 전통적인 활성화 함수이다. 시그모이드는 다음과 같이 s자와 같은 형태를 가진다.

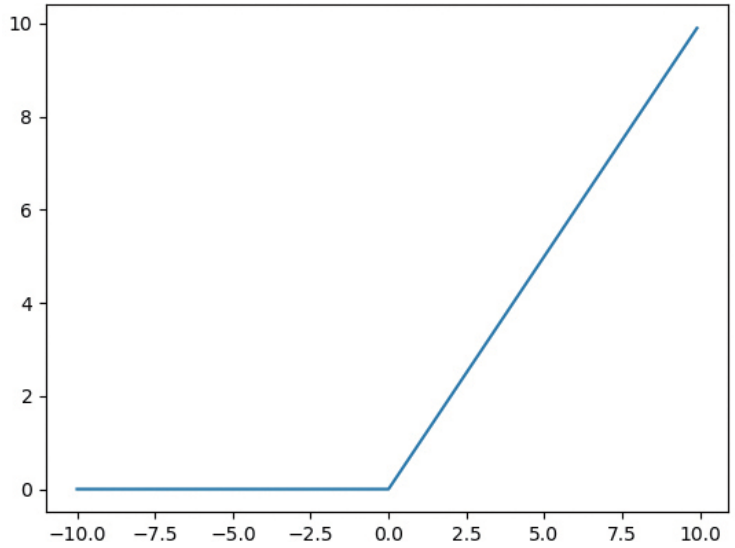
$$f(x) = \frac{1}{1+e^{-x}}$$



활성화 함수 (Rectifed Linear Unit function)

- ReLU 함수는 최근에 가장 인기 있는 활성화 함수이다. ReLU 함수는 입력이 0을 넘으면 그대로 출력하고, 입력이 0보다 적으면 출력은 0이 된다.

$$f(x) = \begin{cases} x & x > 0 \\ 0 & x \leq 0 \end{cases}$$



다중 퍼셉트론 (Multi Layer Perceptron: MLP)

- 좌표 평면 자체에 변화를 주는 것
- XOR 문제를 해결하기 위해서 우리는 두 개의 퍼셉트론을 한 번에 계산할 수 있어야 함
- 이를 가능하게 하려면 숨어있는 층, 즉 은닉층(hidden layer)을 만들면 됨

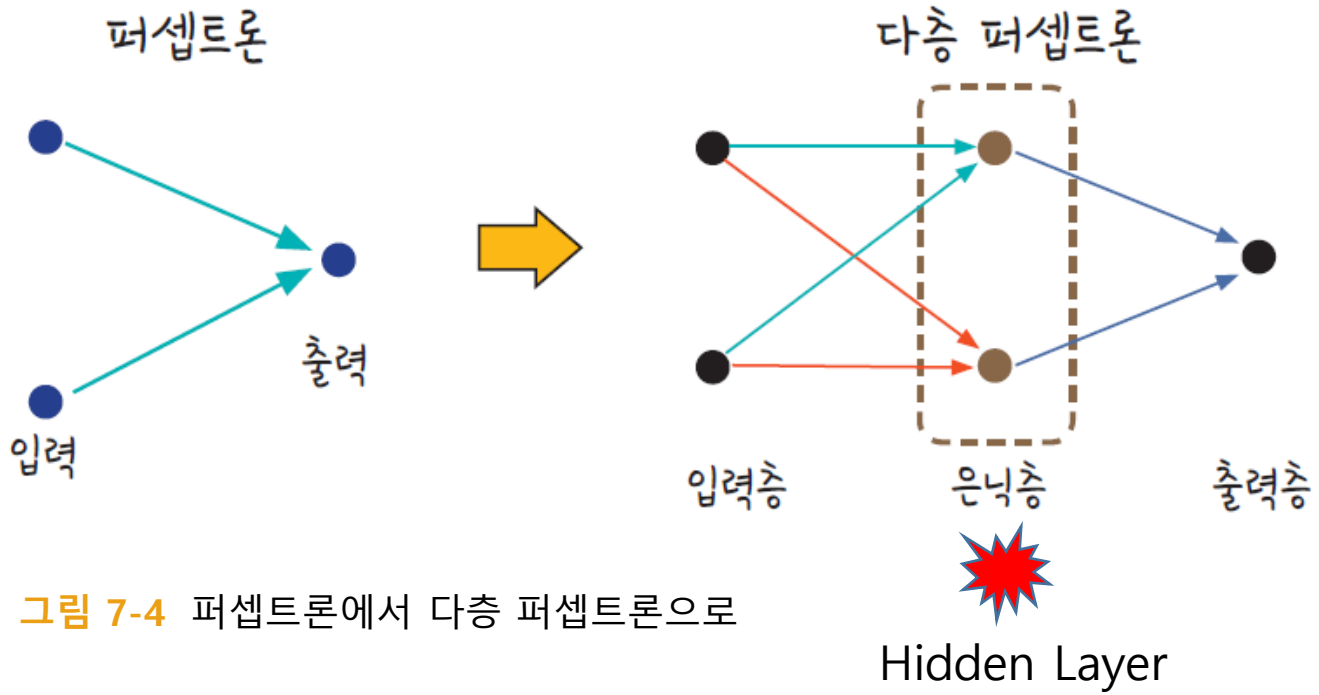


그림 7-4 퍼셉트론에서 다층 퍼셉트론으로

다중 퍼셉트론 (Multi Layer Perceptron: MLP)

- 다층 퍼셉트론이 입력층과 출력층 사이에 숨어있는 은닉층을 만드는 것을 도식으로 나타내면 그림 7-6과 같음

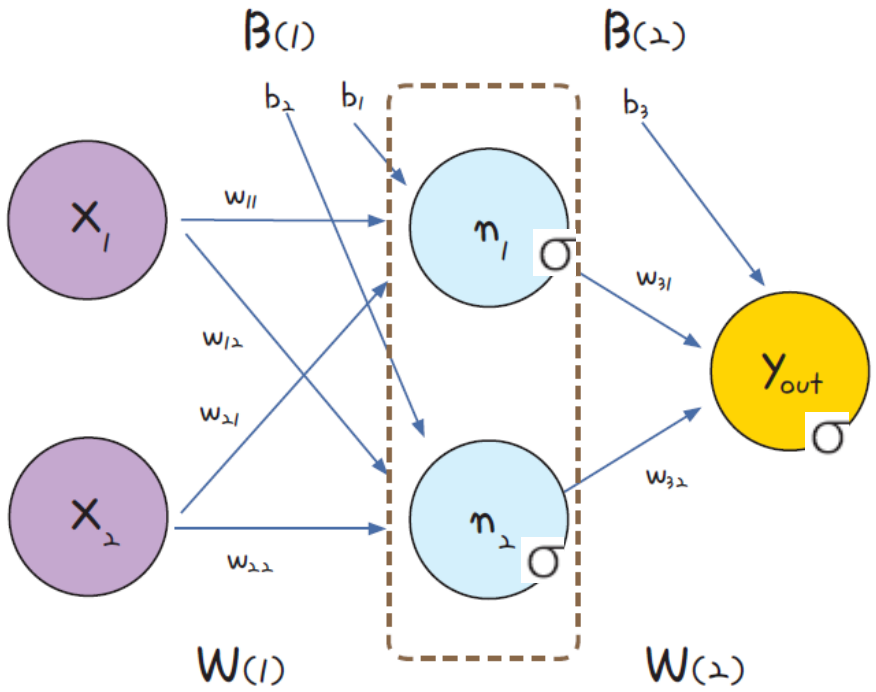


그림 7-6 다중 퍼셉트론의 내부

다중 퍼셉트론 (Multi Layer Perceptron: MLP)

2 | XOR 문제의 해결

- 각 변숫값을 정하고 이를 이용해 XOR 문제를 해결하는 과정을 알아 보자

$$W(1) = \begin{bmatrix} -2 & 2 \\ -2 & 2 \end{bmatrix} \quad B(1) = \begin{bmatrix} 3 \\ -1 \end{bmatrix}$$

$$W(2) = \begin{bmatrix} 1 \\ 1 \end{bmatrix} \qquad B(2) = [-1\,]$$

다중 퍼셉트론 (Multi Layer Perceptron: MLP)

2 | XOR 문제의 해결

- 이것을 도식에 대입하면 다음과 같음

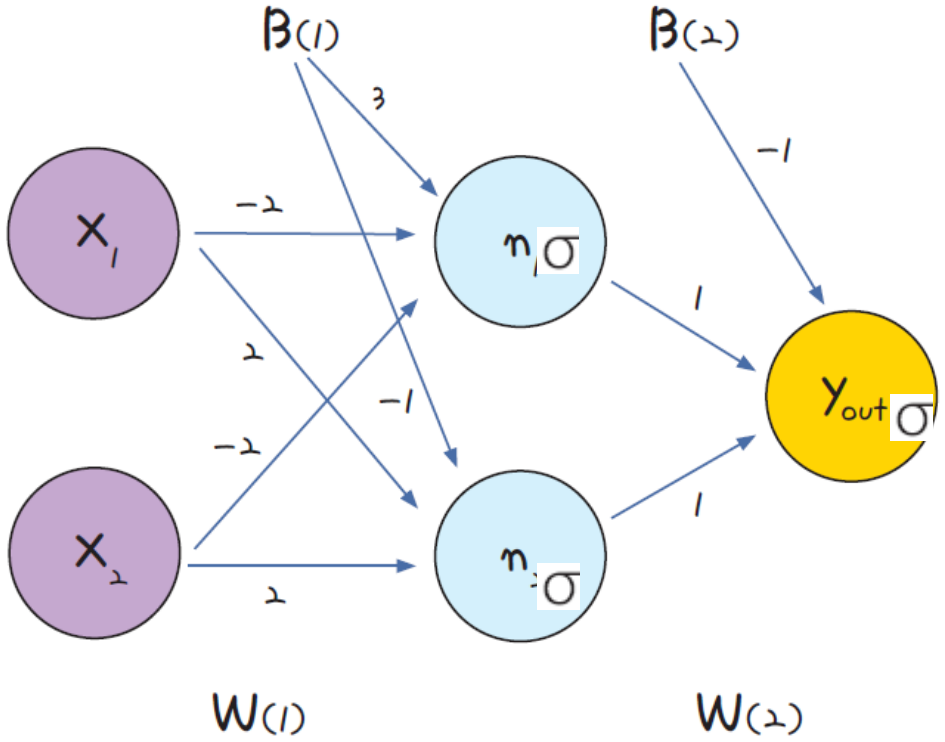
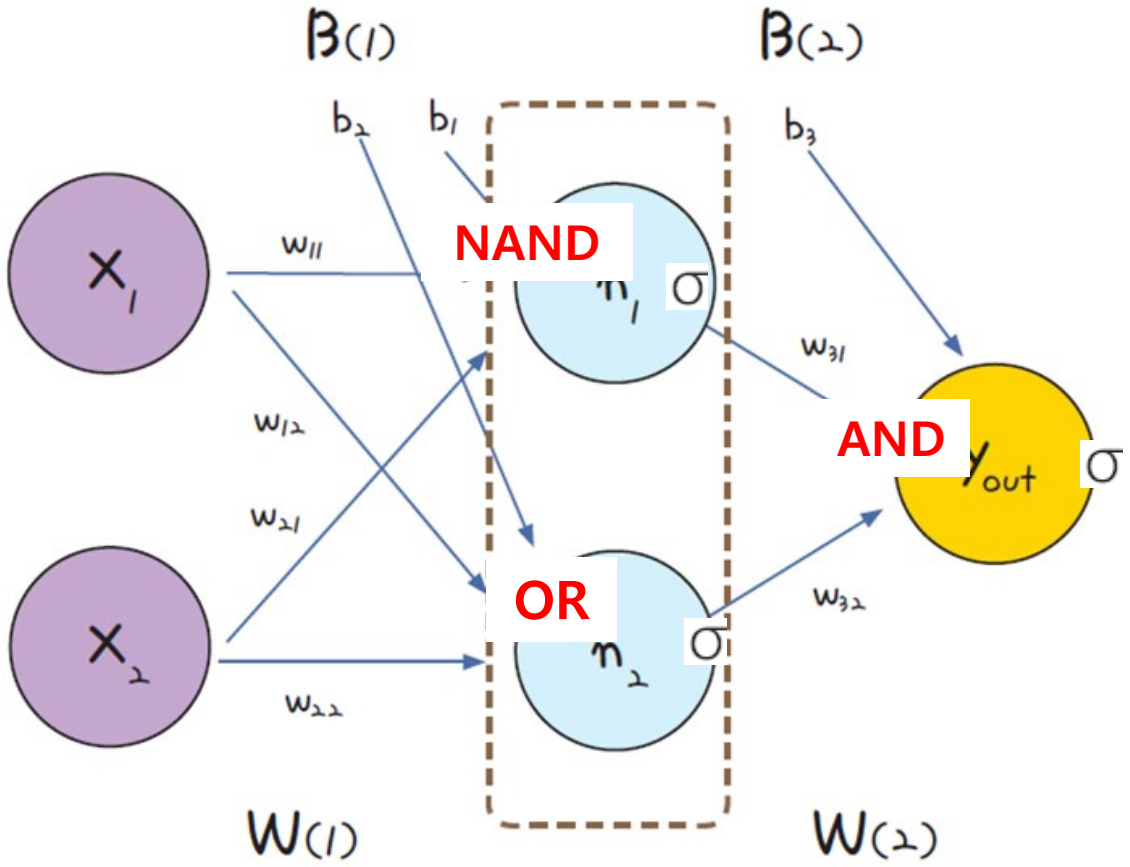


그림 7-7 다중 퍼셉트론의 내부에 변수를 채워보자.

다중 퍼셉트론 (Multi Layer Perceptron: MLP)



다중 퍼셉트론 (Multi Layer Perceptron: MLP)

- 이제 X_1 과 X_2 값을 번갈아 대입해 가며 최종 값을 출력해 보자

```
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))
```

NAND,OR,AND 기능을 하는 퍼셉트론이 연결되어

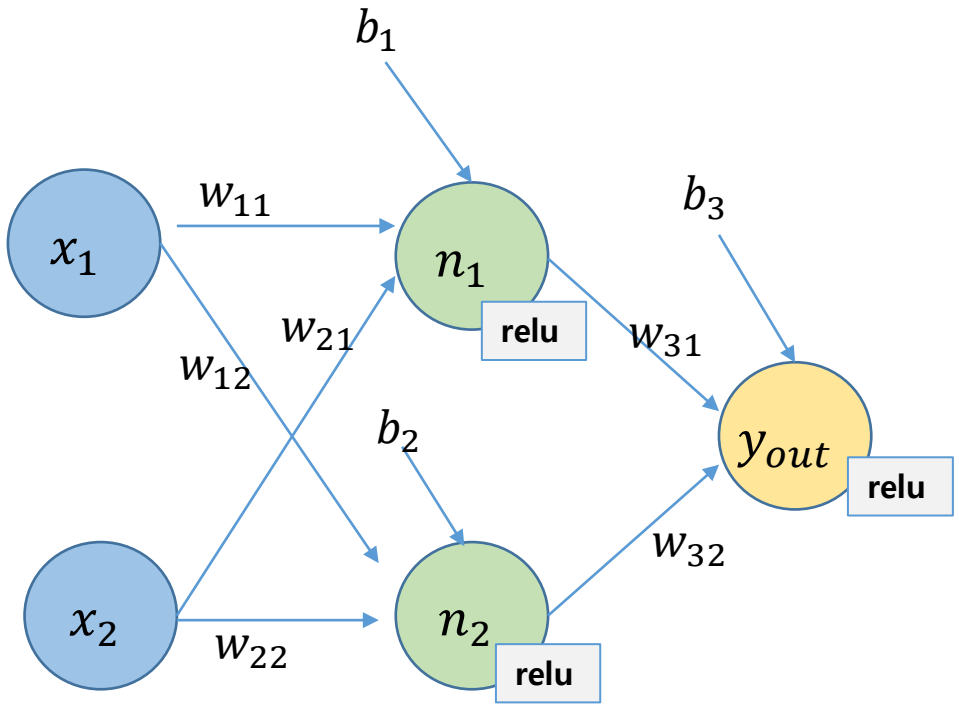
XOR 기능을 수행

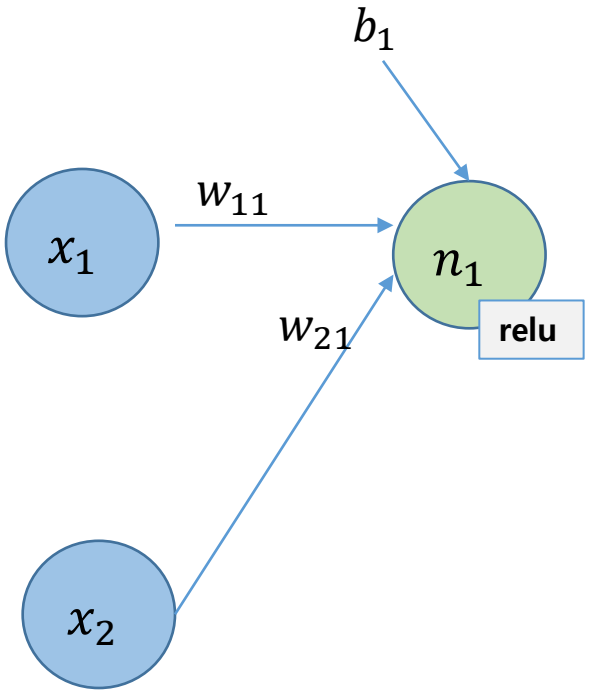
실행 결과

```
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
```

다중 퍼셉트론 (Multi Layer Perceptron: MLP)

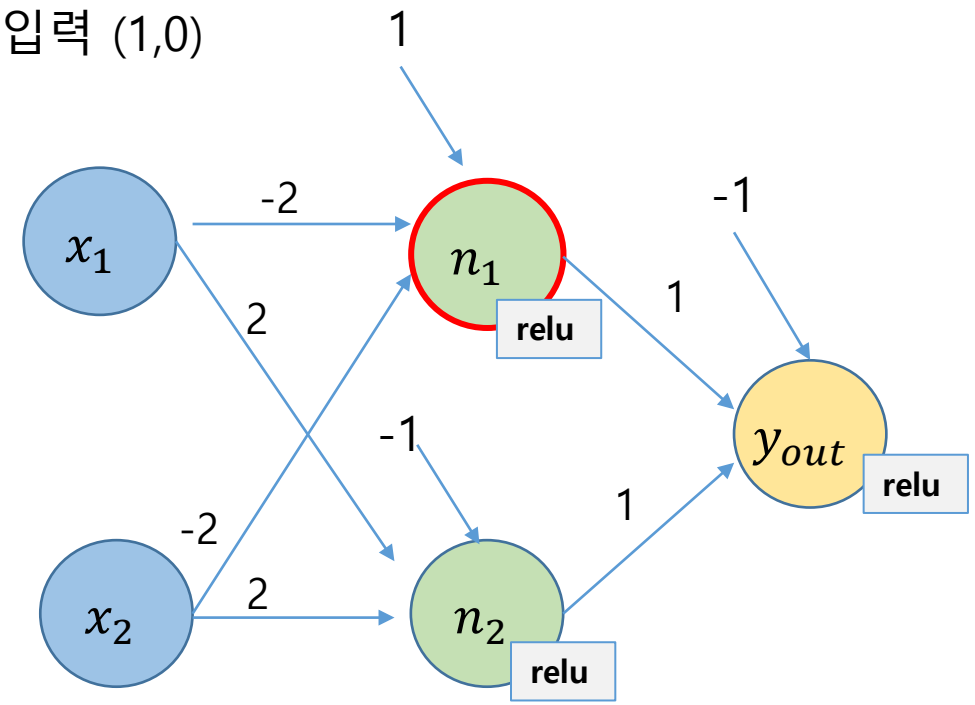
입력 (1,0)





다중 퍼셉트론 (Multi Layer Perceptron: MLP)

입력 (1,0)

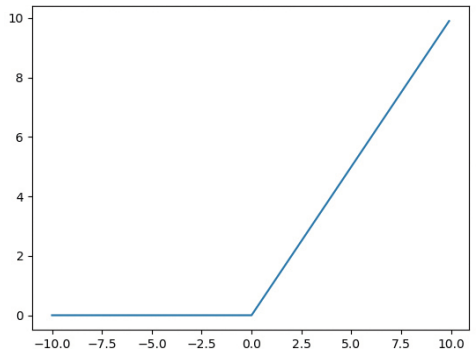


다음 다중 퍼셉트론의 출력은?

w_{12}

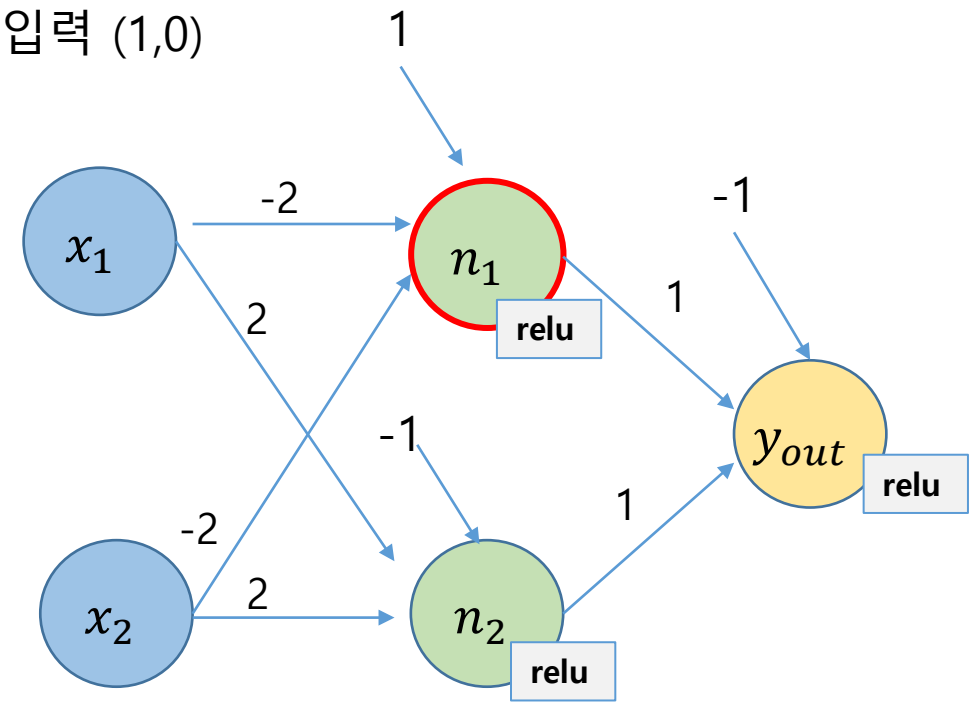
$$f(x) = \begin{cases} x & x > 0 \\ 0 & x \leq 0 \end{cases}$$

Relu



다중 퍼셉트론 (Multi Layer Perceptron: MLP)

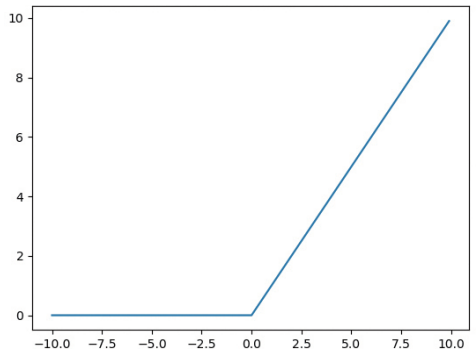
입력 (1,0)



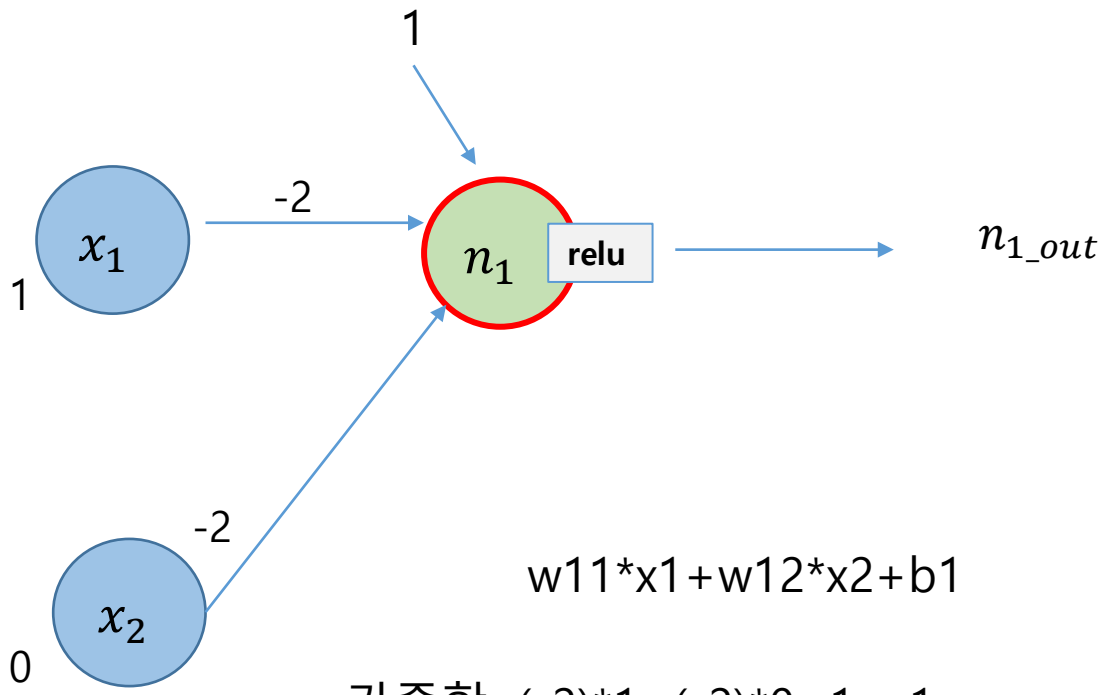
w_{12}

$$f(x) = \begin{cases} x & x > 0 \\ 0 & x \leq 0 \end{cases}$$

Relu



다음 다중 퍼셉트론의 출력은?



w11*x1+w12*x2+b1

가중합: (-2)*1+(-2)*0+1=-1

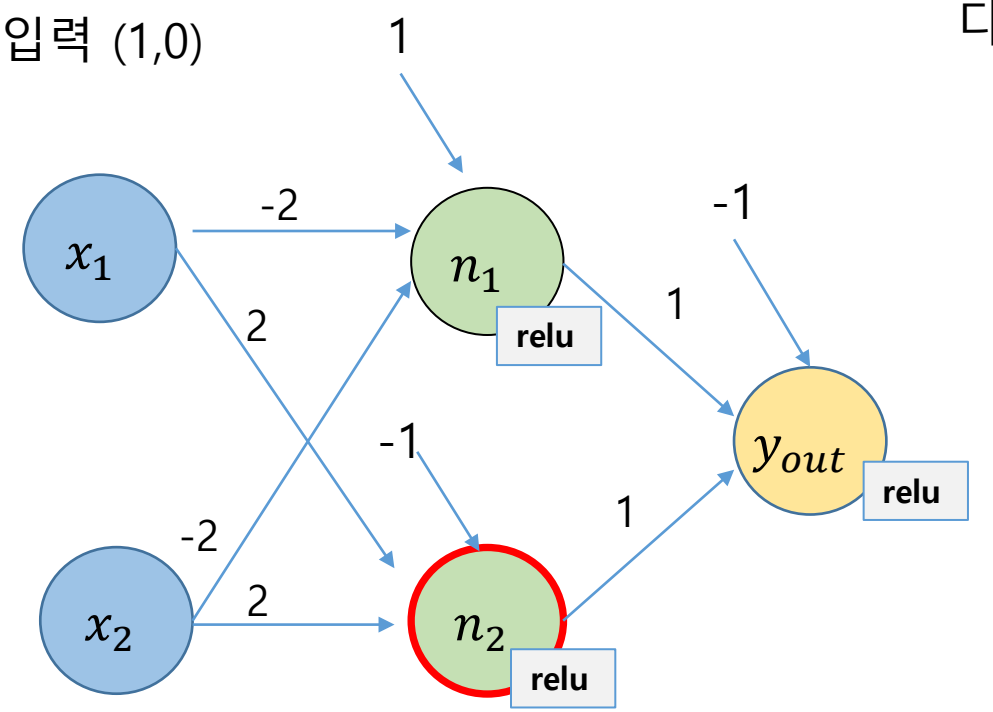
Relu(-1)=0

n1_out=0

다중 퍼셉트론 (Multi Layer Perceptron: MLP)

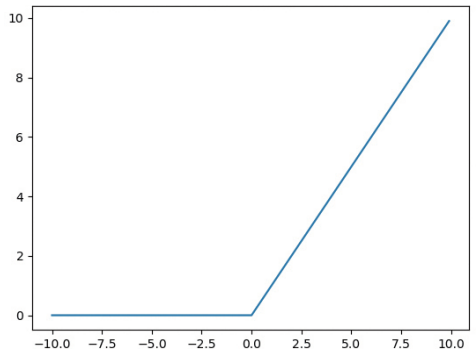
입력 (1,0)

다음 다중 퍼셉트론의 출력은?



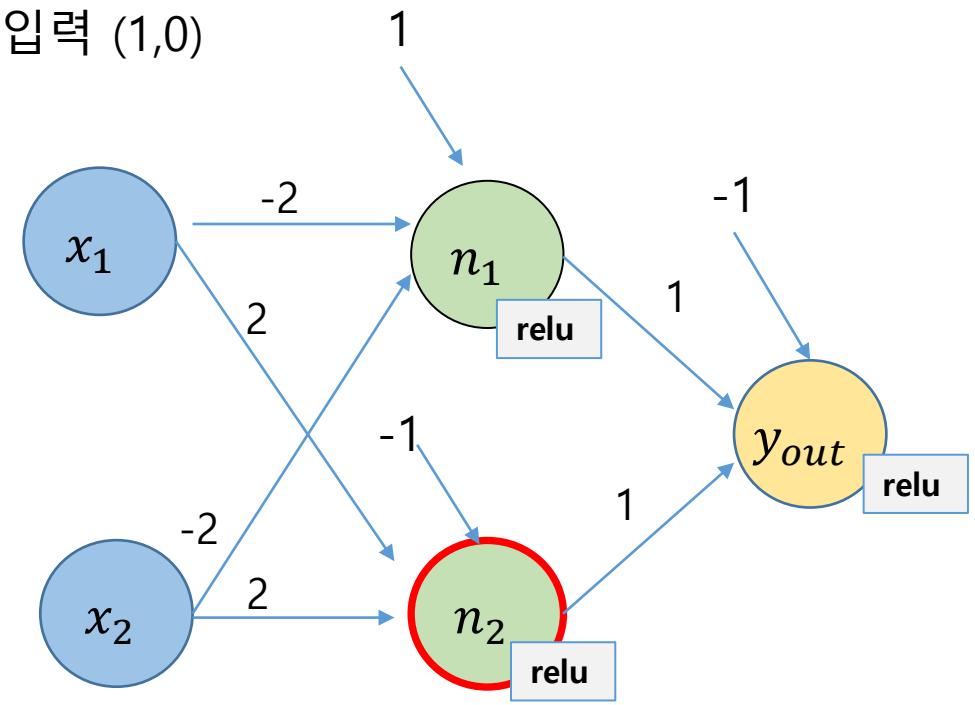
$$f(x) = \begin{cases} x & x > 0 \\ 0 & x \leq 0 \end{cases}$$

Relu



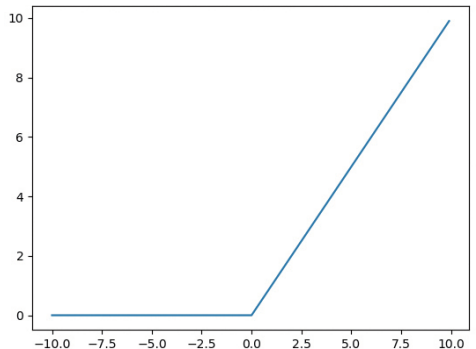
다중 퍼셉트론 (Multi Layer Perceptron: MLP)

입력 (1,0)



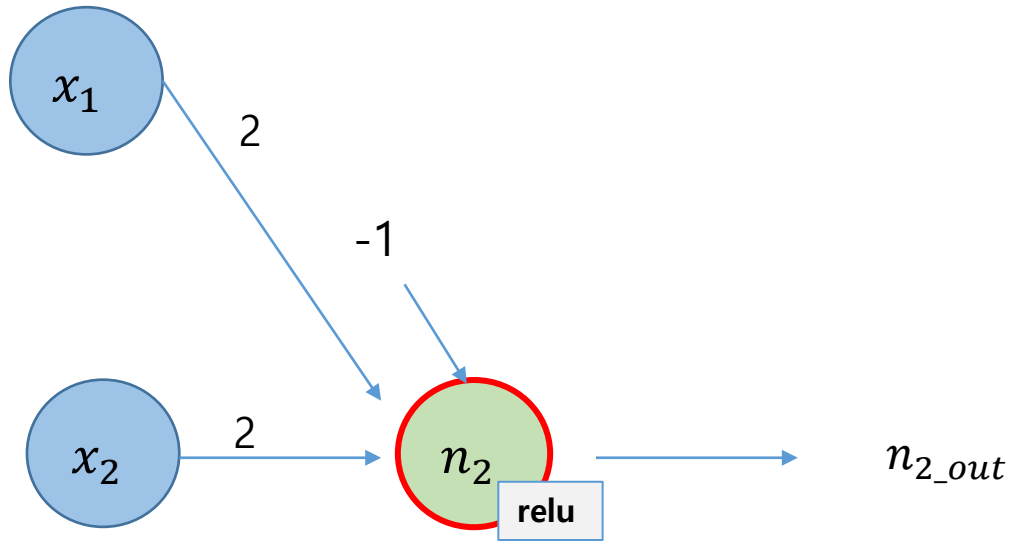
$$f(x) = \begin{cases} x & x > 0 \\ 0 & x \leq 0 \end{cases}$$

Relu



다음 다중 퍼셉트론의 출력은?

입력 (1,0)



w21*x1+w22*x2+b2

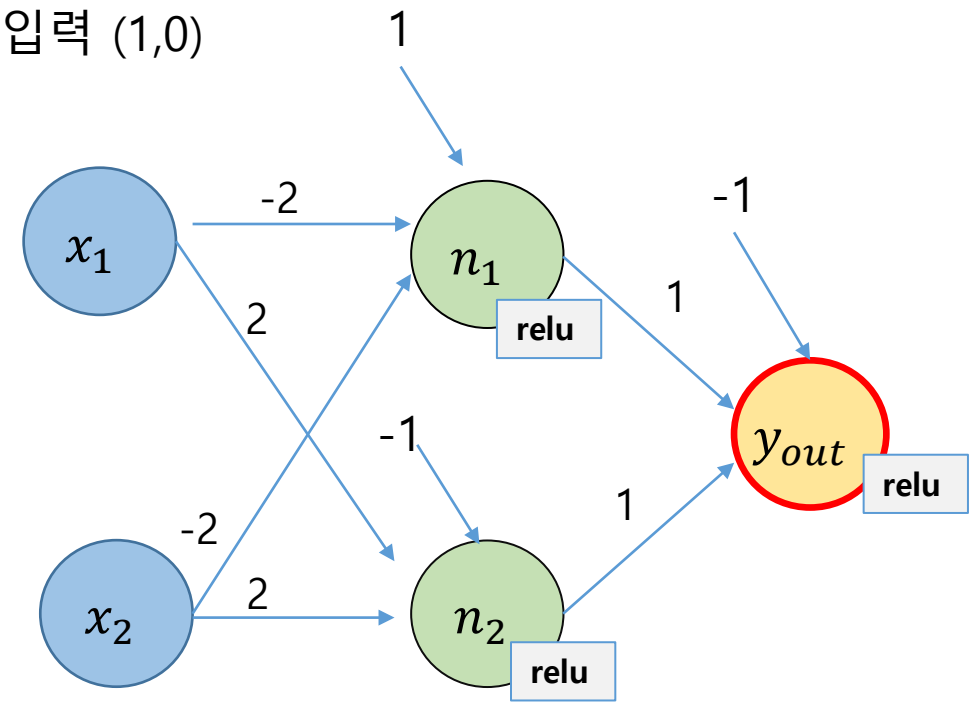
가중합: 1*(2)+2*(0)-1=1

Relu(1)=1

n2_out=1

다중 퍼셉트론 (Multi Layer Perceptron: MLP)

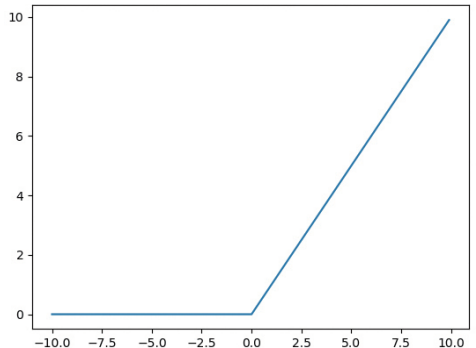
입력 (1,0)



다음 다중 퍼셉트론의 출력은?

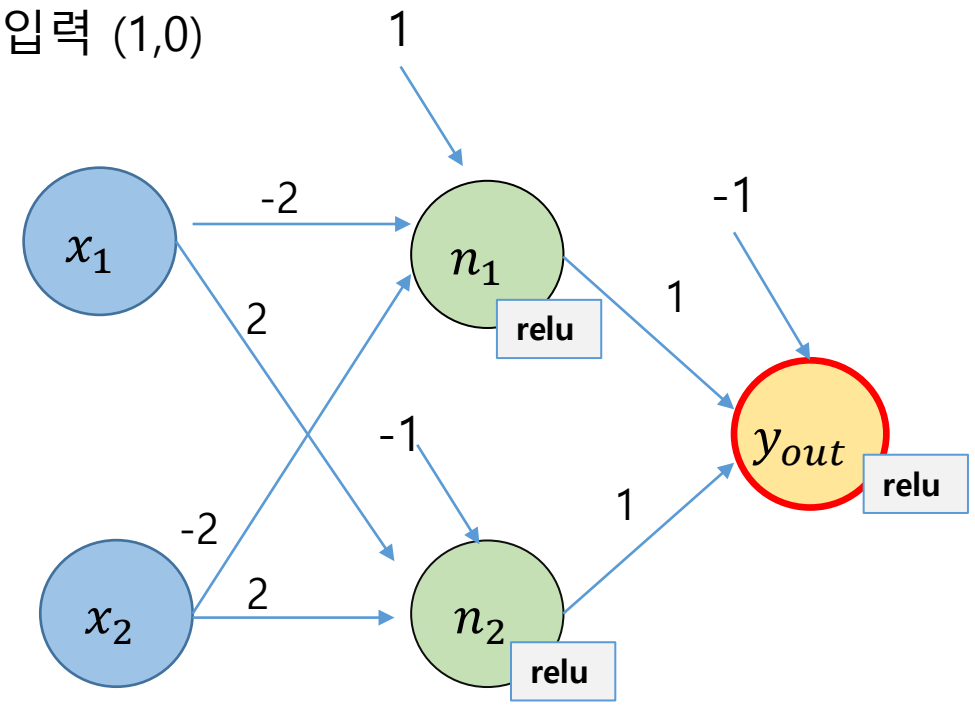
$$f(x) = \begin{cases} x & x > 0 \\ 0 & x \leq 0 \end{cases}$$

Relu



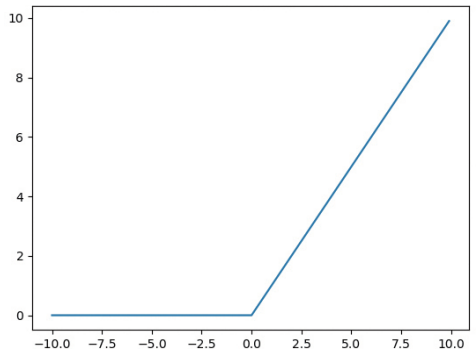
다중 퍼셉트론 (Multi Layer Perceptron: MLP)

입력 (1,0)

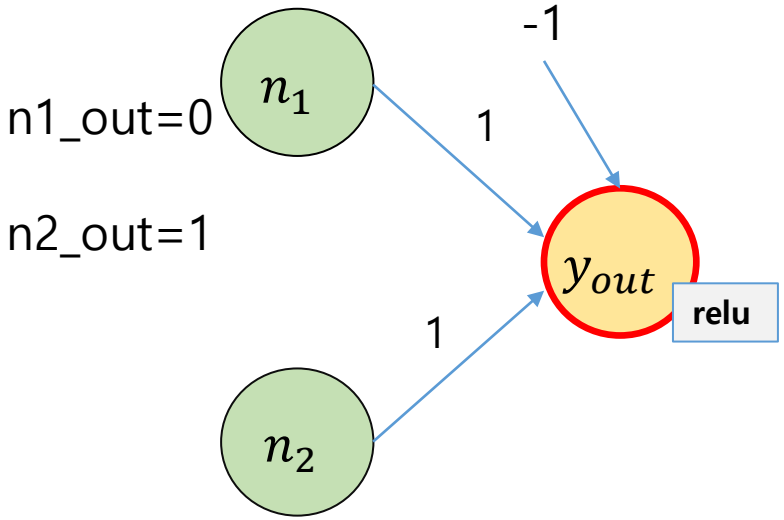


$$f(x) = \begin{cases} x & x > 0 \\ 0 & x \leq 0 \end{cases}$$

Relu



다음 다중 퍼셉트론의 출력은?



w31*n1_out+w32*n2_out+b3

가중합: 1*(0)+1*(1)-1=0

Relu(0)=0

Y_out=0

다중 퍼셉트론 (Multi Layer Perceptron: MLP)

- **순방향 패스**
  - 순방향 패스란 입력 신호가 입력층 유닛에 가해지고 이들 입력 신호가 은닉층을 통하여 출력층으로 전파되는 과정을 의미한다.

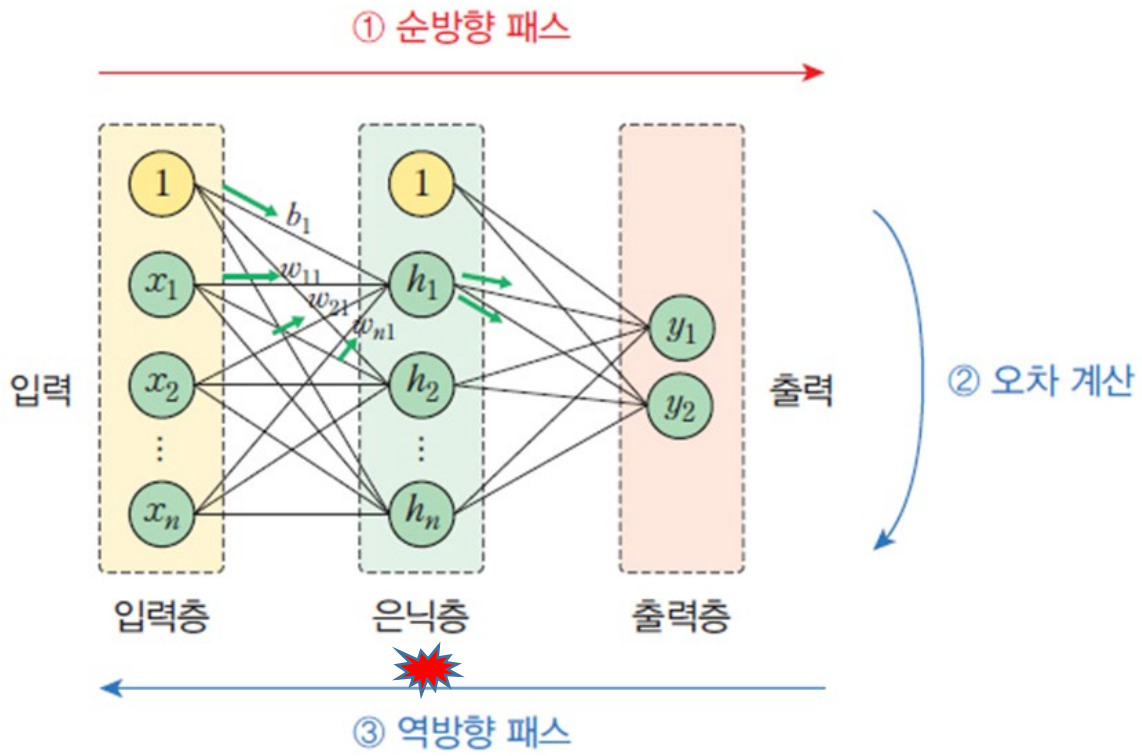


그림 6-5 순방향 패스

다중 퍼셉트론 (Multi Layer Perceptron: MLP)

▪ 역방향 패스 (오차 역전파)

- 신경망 내부의 가중치는 오차 역전파 방법을 사용해 수정함

▪ 결괏값의 오차를 구해 이를 토대로 하나 앞선 가중치를 차례로 거슬러 올라가며 조정함

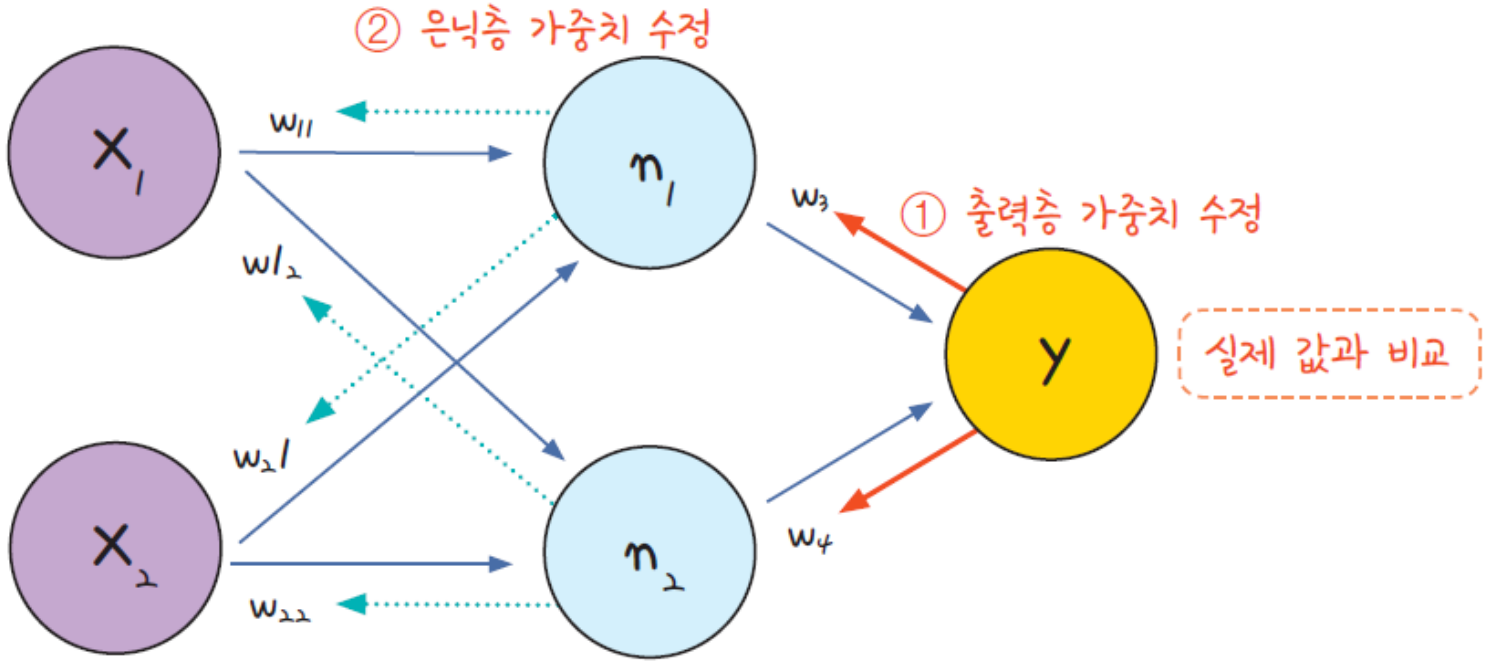


그림 8-2 다층 퍼셉트론에서의 오차 수정

다중 퍼셉트론 (Multi Layer Perceptron: MLP)

- **경사하강법**
  - 손실 함수를 입력 변수를 기준으로 미분하여 w, b를 업데이트
  - 반복 하면서 오차를 최소화 하는 방향으로 w와 b를 업데이트

```
iteration 0: loss 17.17 w 0.10 b 0.08
iteration 50: loss 0.61 w 1.73 b 1.71
iteration 100: loss 0.33 w 1.73 b 2.05
iteration 150: loss 0.24 w 1.63 b 2.23
iteration 200: loss 0.19 w 1.53 b 2.36
iteration 250: loss 0.16 w 1.47 b 2.45
iteration 300: loss 0.15 w 1.41 b 2.52
iteration 350: loss 0.14 w 1.37 b 2.58
iteration 400: loss 0.13 w 1.34 b 2.62
iteration 450: loss 0.13 w 1.32 b 2.65
########## final w,b 1.3033228991130752
2.676018429308869 4


1.25 2.74
```

다중 퍼셉트론 (Multi Layer Perceptron: MLP)

- 오차 역전파(back propagation) :

  다층 퍼셉트론에서의 최적화 과정

- 오차 역전파 구동 방식은 다음과 같이 정리할 수 있음

  1 | 임의의 초기 가중치( )를 준 두 과( )를 계산함

  2 | 계산 결과와 우리가 원하는 값 사이의 오차를 구함

  3 | 경사 하강법을 이용해 바로 앞 가중치를 오차가 작아지는 방향으로 업데이트함

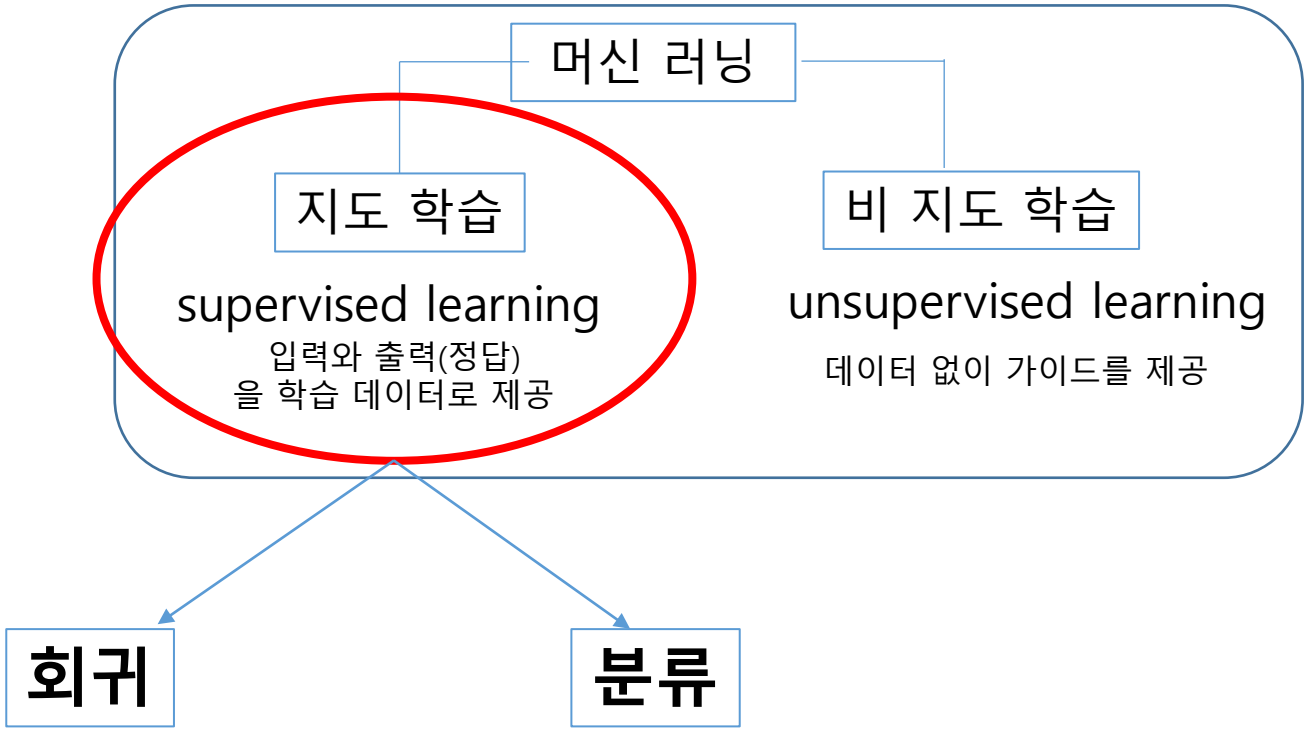
  4 | 위 과정을 더이상 오차가 줄어들지 않을 때까지 반복함

  동영상

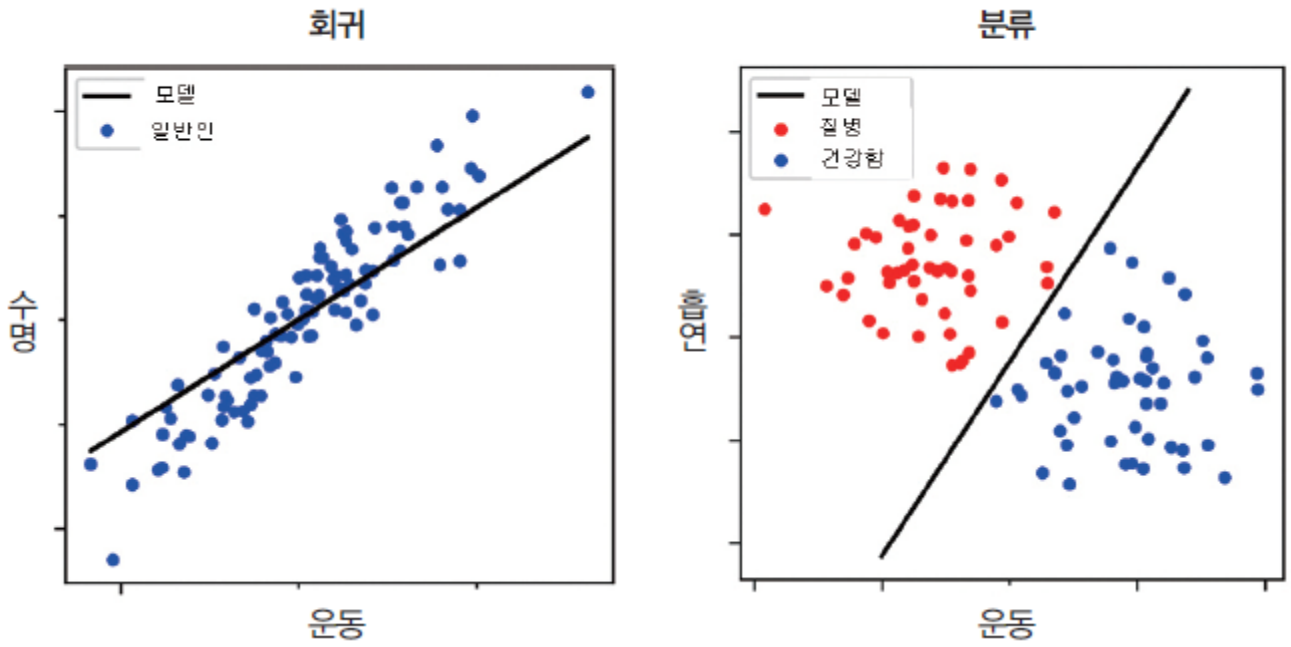
다중 퍼셉트론 (Multi Layer Perceptron: MLP)

- 역전파 알고리즘은 입력이 주어지면 순방향으로 계산하여 출력을 계산한 후에 실제 출력과 우리가 원하는 출력 간의 오차를 계산한다. 이 오차를 역방향으로 전파하면서 오차를 줄이는 방향으로 가중치를 변경한다.

지도 학습 : 분류 vs. 회귀


머신 러닝
지도 학습
supervised learning
입력와 출력(정답)
을 학습 데이터로 제공
비 지도 학습
unsupervised learning
데이터 없이 가이드를 제공
회귀
분류

지도 학습 : 분류 vs. 회귀



지도 학습 : 분류 vs. 회귀

- 회귀(regression)는 주어진 입력-출력 쌍을 학습한 후에 새로운 입력값이 들어왔을 때, 합리적인 출력값을 예측
- 회귀는 입력(x)과 출력(y)이 주어질 때, 입력에서 출력으로의 매핑 함수를 학습하는 것이라 할 수 있다.

$$y = f(x)$$

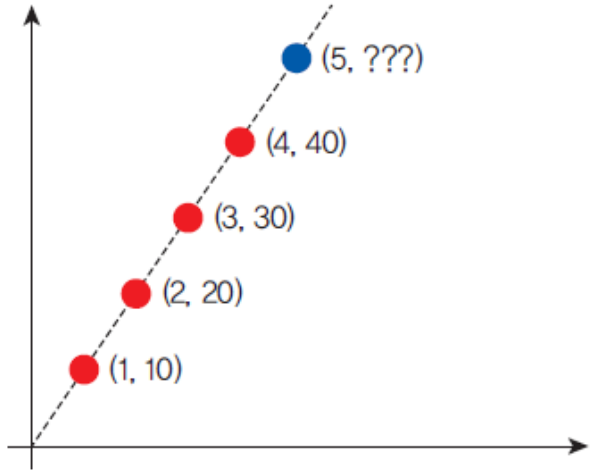
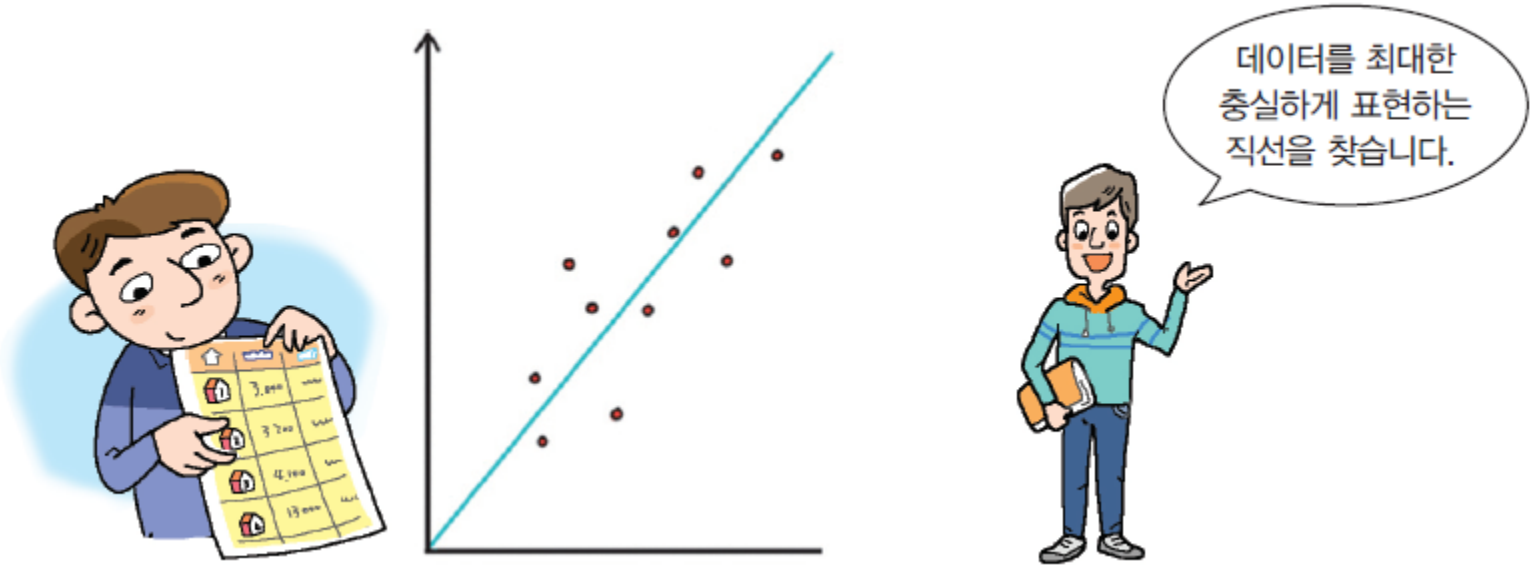


그림 3-7 회귀

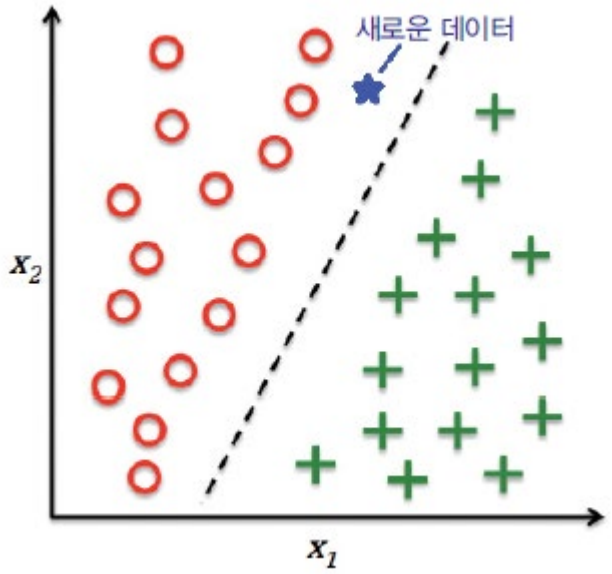
지도 학습 : 분류 vs. 회귀

- **회귀(regression) : 회귀에서는 입력과 출력이 모두 실수이다.**
  - "사용자가 이 광고를 클릭할 확률이 얼마인가요?"
  - "면적에 따른 각 아파트의 가격은 어떻게 되나요?"
  - "키에 따른 몸무게"



지도 학습 : 분류 vs. 회귀

- 앞에 나왔던 식 y = f(x)에서 출력 y가 이산적(discrete)인 경우에 이것을 분류 문제(또는 인식 문제)라고 부른다.
- 분류에서는 입력을 2개 이상의 클래스(부류)로 나누는 것이다.
- 예를 들어서 사진을 보고 "강아지", 또는 "고양이"로 분류하는 것도 분류 문제이다.



지도 학습 : 분류 (classification)

- 많은 과일로 채워진 과일 바구니를 보고, 프로그램이 바나나, 오렌지와 같은 올바른 레이블을 예측

번호	크기	색상	모양	과일 이름
1	크다.	빨강색	동근 모양에 꼭지가 있음	사과
2	작다.	빨강색	심장모양	체리
3	크다.	녹색	길고 곡선 형태의 원통 모양	바나나
4	작다.	녹색	타원형, 다발 형태	포도

색상(nm)	산성도(pH)	라벨
610	3.8	오렌지주스
380	2.5	콜라
390	2.6	콜라
...	...	...

분류 예시

독립변수	종속변수	학습시킬 데이터를 만드는 방법
공부시간	합격 여부 (합격/불합격)	사람들의 공부시간을 입력받고, 최종 합격여부를 확인한다.
X-ray 사진과 영상 속 종양의 크기, 두께	악성 종양 여부 (양성/음성)	의학적으로 양성과 음성이 확인된 사진과 영상 데이터를 모은다.
품종, 산도, 당도, 지역, 연도	와인의 등급	소믈리에를 통해서 등급이 확인된 와인을 가지고 품종, 산도 등의 독립변수를 정하고 기록한다.
메일 발신인, 제목, 본문 내용 (사용된 단어, 이모티콘 등)	스팸 메일 여부	이제까지 받은 메일을 모으고, 이들을 스팸 메일과 일반 메일로 구분한다.
고기의 지방함량, 지방색, 성숙도, 육색	소고기 등급	소고기의 정보를 토대로 등급을 측정한다.

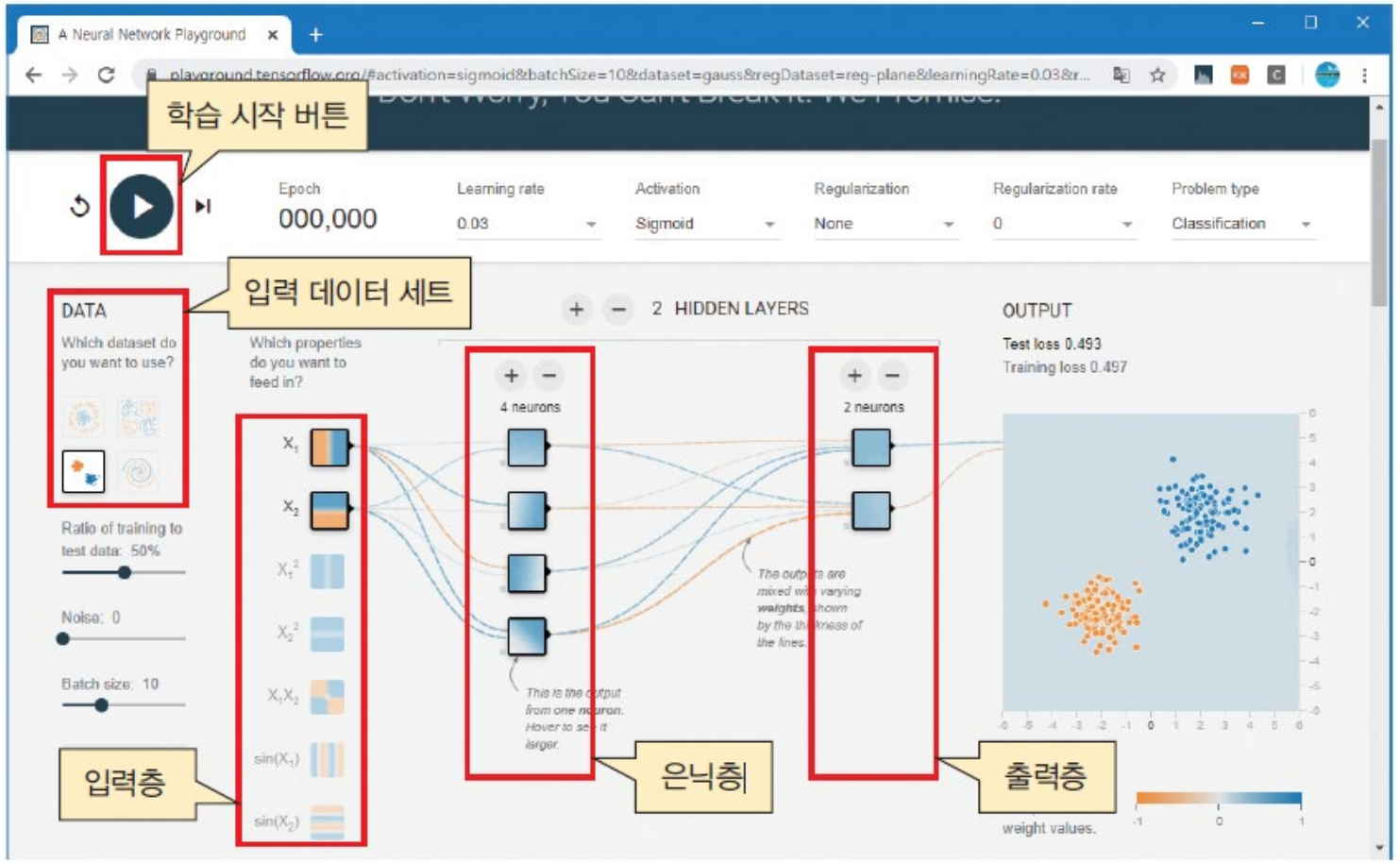
회귀 예시

독립변수	종속변수	학습시킬 데이터를 만드는 방법
공부시간	시험점수 (10점, 20점)	사람들의 공부시간을 입력받고 점수를 확인한다.
온도	레모네이드 판매량	온도와 그날의 판매량을 기록한다.
역세권, 조망 등	집 값	집과 역까지의 거리, 수치화된 조망의 평점 등을 집 값과 함께 기록한다
온실 기체량	기온 변화량	과거에 배출된 온실 기체량과 기온의 변화량을 기록한다.
자동차 속도	충돌 시 사망 확률	충돌시 속도와 사상자를 기록한다.
나이	키	학생들의 나이

구글 플레이 그라운드 실습

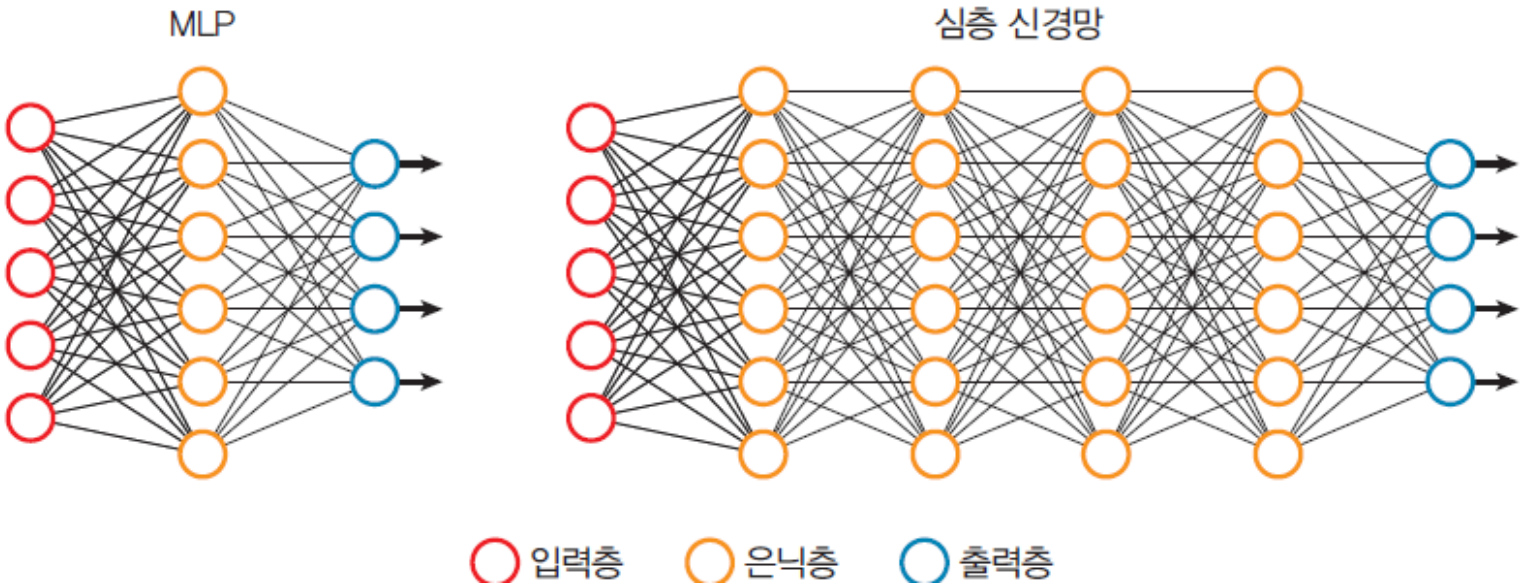
- https://playground.tensorflow.org
- 은닉층 (hidden layer) 유무에 따른 성능 비교
- 분류 (classification) 예제 실습
- Activation : 활성화 함수
- Classification : 분류
- Regression : 회귀
- Learning rate : 학습률
- Epoch : 반복 횟수

구글의 플레이그라운드


A Neural Network Playground
학습 시작 버튼
Epoch
000,000
Learning rate
0.03
Activation
Sigmoid
Regularization
None
Regularization rate
0
Problem type
Classification
입력 데이터 세트
DATA
Which dataset do you want to use?
Which properties do you want to feed in?
2 HIDDEN LAYERS
4 neurons
2 neurons
OUTPUT
Test loss 0.493
Training loss 0.497
Ratio of training to test data: 50%
Noise: 0
Batch size: 10
입력층
은닉층
출력층
weight values.

딥러닝 (Deep learning)

- **심층 신경망 (Deep Neural Network)을 사용하는 학습 방법**
- 심층 신경망(DNN: Deep Neural Networks)은 MLP(다층 퍼셉트론)에서 은닉층의 개수를 증가시킨 것이다.
- 은닉층을하나만 사용하는 것이 아니고 여러 개를 사용한다.
- 최근에 딥러닝은 컴퓨터 시각, 음성 인식, 자연어 처리, 소셜 네트워크 필터링, 기계 번역 등에 적용되어서 인간 전문가에 필적하는 결과를 얻고 있다.



딥러닝 (Deep learning) 실습

- 케라스로 모델 설계 하기 (keras 라이브러리 이용)
- Tensorflow + keras 사용
- Import tensorflow.keras
- 폐암 수술 환자의 생존율 예측 하기 다시 돌아보기 (01_My_First_Deeplearning.ipynb)

딥러닝 (Deep learning) 실습

- 자료실 colab_01_My_First_Deeplearning_no_drive.ipynb 실습
- 데이터 ThoracicSurgery

폐암 환자 생존율 실습

```
# 딥러닝을 구동하는 데 필요한 케라스 함수 호출
from tensorflow.keras.models import Sequential
from tensorflow.keras.layers import Dense

# 필요한 라이브러리 불러옴
import numpy as np
import tensorflow as tf

# 실행할 때마다 같은 결과를 출력하기 위해 설정하는 부분
np.random.seed(3)
tf.random.set_seed(3)
```

폐암 환자 생존율 실습

```
# 준비된 수술 환자 데이터를 불러옴
Data_set = np.loadtxt("../dataset/ThoracicSurgery.csv",
delimiter=",")

# 환자의 기록과 수술 결과를 X와 Y로 구분하여 저장
X = Data_set[:,0:17]
Y = Data_set[:,17]

# 딥러닝 구조를 결정(모델을 설정하고 실행하는 부분)
model = Sequential( )
model.add(Dense(30, input_dim=17, activation='relu'))
model.add(Dense(1, activation='sigmoid'))
```

*MLP 정의

폐암 환자 생존율 실습

속성 / 클래스

	정보 1	정보 2	정보 3	…	정보 17	생존 여부
1번째 환자	293	1	3.8	…	62	0
2번째 환자	1	2	2.88	…	60	0
3번째 환자	8	3	3.19	…	66	1
…	…	…	…	…	…	…
470번째 환자	447	8	5.2	…	49	0

샘플

표 10-2 폐암 환자 생존율 예측 데이터의 샘플, 속성, 클래스 구분

17개의 속성이 있을때 생존 여부를 예측 하는 모델 생성

폐암 환자 생존율 실습

```
# 딥러닝 실행
model.compile(loss=' binary_crossentropy ', optimizer='adam',
metrics=['accuracy'])
model.fit(X, Y, epochs=100, batch_size=10)
```

폐암 환자 생존율 실습

- 딥러닝의 모델을 설정하고 구동하는 부분은 모두 model이라는 함수를 선언하며 시작이 됨
- 먼저 model = Sequential( )로 시작되는 부분은 딥러닝의 구조를 짜고 층을 설정하는 부분임
- 이어서 나오는 model.compile( ) 부분은 위에서 정해진 모델을 컴퓨터가 알아들을 수 있게끔 컴파일 하는 부분임
- model.fit( )으로 시작하는 부분은 모델을 실제로 수행하는 부분임

```
# 딥러닝 구조를 결정(모델을 설정하고 실행하는 부분)
model = Sequential( )
model.add(Dense(30, input_dim=17, activation='relu'))
model.add(Dense(1, activation='sigmoid'))
```

폐암 환자 생존율 실습

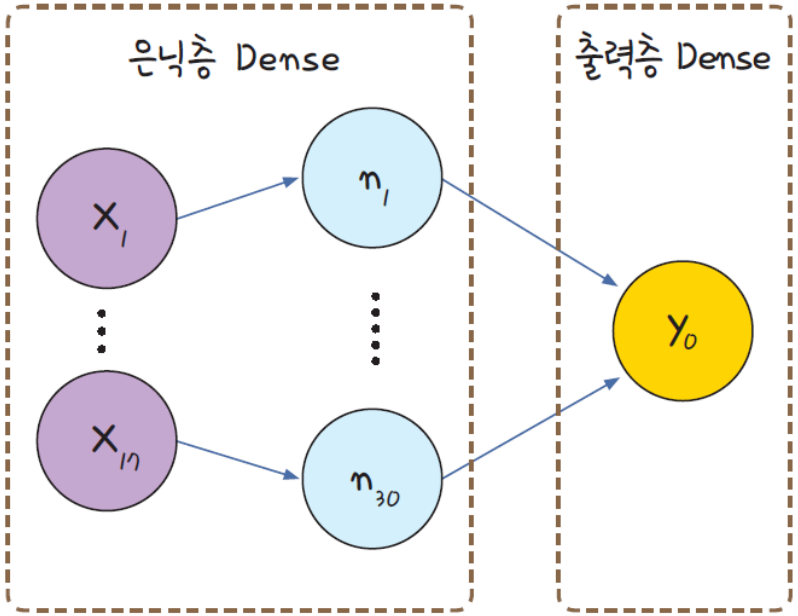
- 맨 마지막 층은 결과를 출력하는 '출력층'이 됨
- 나머지는 모두 '은닉층'의 역할을 함
- 지금 만들어진 이 두 개의 층은 각각 은닉층과 출력층임

폐암 환자 생존율 실습

- 각각의 층은 Dense라는 함수를 통해 구체적으로 그 구조가 결정됨
- Dense(30, input_dim=17, activation='relu')부분을 더 살펴보자
  - → model.add( ) 함수를 통해 새로운 층을 만들고 나면 Dense() 함수를 통해 이 층에 몇 개의 노드를 만들 것인지를 숫자로 써줌
  - → 30이라고 되어 있는 것은 이 층에 30개의 노드를 만들겠다는 것
  - → 이어서 input_dim이라는 변수가 나옴
  - → 이는 입력 데이터에서 몇 개의 값을 가져올지를 정하는 것

폐암 환자 생존율 실습

- keras는 입력층을 따로 만드는 것이 아니라, 첫 번째 은닉층에 input_dim을 적어줌으로써 첫 번째 Dense가 은닉층 + 입력층의 역할을 겸함
- 우리가 다루고 있는 폐암 수술 환자의 생존 여부 데이터에는 17개의 입력 값들이 있음
- 데이터에서 17개의 값을 받아 은닉층의 30개 노드로 보낸다는 뜻




```
model.add(Dense(30, input_dim=17, activation='relu'))
model.add(Dense(1, activation='sigmoid'))
```

폐암 환자 생존율 실습

3 | 모델 컴파일

```
model.compile(loss=' binary_crossentropy ', optimizer='adam',
metrics=['accuracy'])
```

- model.compile 부분은 앞서 지정한 모델이 효과적으로 구현될 수 있게 여러 가지 환경을 설정해 주면서 컴파일하는 부분임
- 먼저 어떤 오차 함수를 사용할지를 정해야 함
- 여기서는 평균 제곱 오차 함수(binary_crossentropy)를 사용함

폐암 환자 생존율 실습

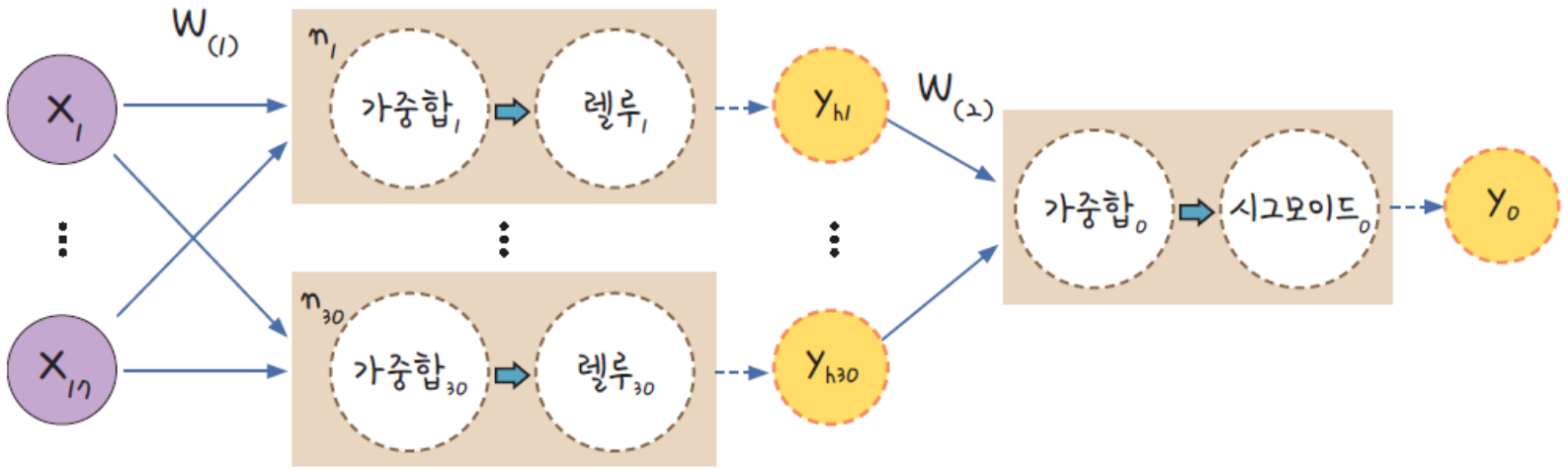


그림 10-2 폐암 환자 생존율 예측 신경망 모델의 도식화. 여기서 W는 각 층별 가중치(w)들의 집합을 말함.

폐암 환자 생존율 실습

모델 실행 하기. 학습 하기

- 컴파일 단계에서 정해진 환경을 주어진 데이터를 불러 실행시킬 때 사용되는 함수는 다음과 같이 model.fit( ) 부분임

```
model.fit(X, Y, epochs=100, batch_size=10)
```

- 1 epoch('에포크'라고 읽음) :
  학습 프로세스가 모든 샘플에 대해 한 번 실행되는 것
- 코드에서 epochs=100으로 지정한 것은 각 샘플이 처음부터 끝까지 100번 재사용될 때까지 실행을 반복하라는 뜻

폐암 환자 생존율 실습

- batch_size
  - 샘플을 한 번에 몇 개씩 처리할지를 정하는 부분으로 batch_size=10은 전체 470개의 샘플을 10개씩 끊어서 집어넣으라는 뜻
  - batch_size가 너무 크면 학습 속도가 느려지고, 너무 작으면 각 실행 값의 편차가 생겨서 전체 결괏값이 불안정해질 수 있음
  - 자신의 컴퓨터 메모리가 감당할 만큼의 batch_size를 찾아 설정해 주는 것이 좋음

수고 하셨습니다

jhmin@inhatc.ac.kr