AI 프로그래밍

- 2024

6주차

오늘 수업 순서

- 퍼셉트론
- 행렬
- 활성화 함수
- 다중 퍼셉트론

중간평가

- **4월 25일 목요일 11:30**

1 차원 배열


```
import numpy as np
a=np.array([1,2,3,4,5])
print(a.shape)
```

```
(5,)
```

```
import numpy as np
a=np.array([1,
            2,
            3,
            4,
            5])
print(a.shape)
```

```
(5,)
```

numpy


```
import numpy as np
a = np.array([1, 2, 3])
a1 = a[np.newaxis, :]
a2 = a[:, np.newaxis]
print('a',a,a.shape)
print('a1',a1,a1.shape)
print('a2',a2,a2.shape)
```

```
a   [1 2 3]          (3,)
a1 [[1 2 3]]         (1, 3)
a2 [[1]
    [2]
    [3]] (3, 1)
```

numpy


```
import numpy as np

a=np.array([1,2,3])
a3=a.reshape(1,-1)
a4=a.reshape(-1,1)
print(a3,a3.shape)
print(a4,a4.shape)
```

numpy


```
import numpy as np

a=np.array([1,2,3])
a3=a.reshape(1,-1)
a4=a.reshape(-1,1)
print(a3,a3.shape)
print(a4,a4.shape)
```

```
[[1 2 3]] (1, 3)
[[1]
 [2]
 [3]] (3, 1)
```

numpy


```
a1 [[1 2 3]]       (1, 3)
a2 [[1]
    [2]
    [3]] (3, 1)
```

numpy


```
a1 [[1 2 3]]    shape->  (1, 3)
a2 [[1]
    [2]
    [3]] shape-> (3, 1)
```

```
arr=np.array(
   [ [ [ 0,  1,  2,  3],
       [ 4,  5,  6,  7],
       [ 8,  9, 10 ,11] ],
     [ [12, 13, 14, 15],
       [16 ,17 ,18, 19],
       [20 ,21 ,22, 23] ] ])

print(arr,arr.shape)
```

Scikit learn parameter

https://scikit-learn.org/

- **fit (X,y) :학습**

Parameters:
X{array} of shape (n_samples, n_features).

- **Predict (X) :예측**

Parameters:
X{array} of shape (n_samples, n_features).

농어의 무게 예측

- **입력 데이터 : 길이**
- **출력 데이터 무게**

```
import numpy as npperch_length = np.array(     [8.4, 13.7, 15.0, 16.2,
17.4, 18.0, 18.7, 19.0, 19.6, 20.0,       21.0, 21.0, 21.0, 21.3, 22.0, 22.0,
22.0, 22.0, 22.0, 22.5,       22.5, 22.7, 23.0, 23.5, 24.0, 24.0, 24.6, 25.0,
25.6, 26.5,       27.3, 27.5, 27.5, 27.5, 28.0, 28.7, 30.0, 32.8, 34.5, 35.0,
36.5, 36.0, 37.0, 37.0, 39.0, 39.0, 39.0, 40.0, 40.0, 40.0,       40.0, 42.0,
43.0, 43.0, 43.5, 44.0]      )
```

```
perch_weight = np.array(     [5.9, 32.0, 40.0, 51.5, 70.0, 100.0, 78.0, 80.0,
85.0, 85.0,       110.0, 115.0, 125.0, 130.0, 120.0, 120.0, 130.0, 135.0,
110.0,       130.0, 150.0, 145.0, 150.0, 170.0, 225.0, 145.0, 188.0, 180.0,
197.0, 218.0, 300.0, 260.0, 265.0, 250.0, 250.0, 300.0, 320.0,       514.0,
556.0, 840.0, 685.0, 700.0, 700.0, 690.0, 900.0, 650.0,       820.0, 850.0,
900.0, 1015.0, 820.0, 1100.0, 1000.0, 1100.0,       1000.0, 1000.0]      )
```

참고자료 : 혼자 공부하는 머신 러닝 + 딥러닝

```
from sklearn.model_selection import train_test_split# 훈련 세트와 테스트 세트로 나눕니다

rain_input, test_input, train_target, test_target = train_test_split(     perch_length, perch_weight,
random_state=42)
from sklearn.linear_model import LinearRegression

  lr = LinearRegression() # 선형 회귀 모델 훈련
  lr.fit(train_input, train_target)


# 50cm 농어에 대한 예측 p
rint(lr.predict([[50]]))

print(lr.coef_, lr.intercept_)

plt.scatter(train_input, train_target)

plt.plot([15, 50], [15*lr.coef_+lr.intercept_, 50*lr.coef_+lr.intercept_])
# 50cm 농어 데이터
plt.scatter(50, 1241.8, marker='^')
plt.show()
```

```
from sklearn.model_selection import train_test_split# 훈련 세트와 테스트 세트로 나눕니다

rain_input, test_input, train_target, test_target = train_test_split(     perch_length, perch_weight,
random_state=42)
from sklearn.linear_model import LinearRegression

  lr = LinearRegression() # 선형 회귀 모델 훈련
  lr.fit(train_input, train_target)


# 50cm 농어에 대한 예측 p
rint(lr.predict([[50]]))

print(lr.coef_, lr.intercept_)

plt.scatter(train_input, train_target)

plt.plot([15, 50], [15*lr.coef_+lr.intercept_, 50*lr.coef_+lr.intercept_])
# 50cm 농어 데이터
plt.scatter(50, 1241.8, marker='^')
plt.show()
```

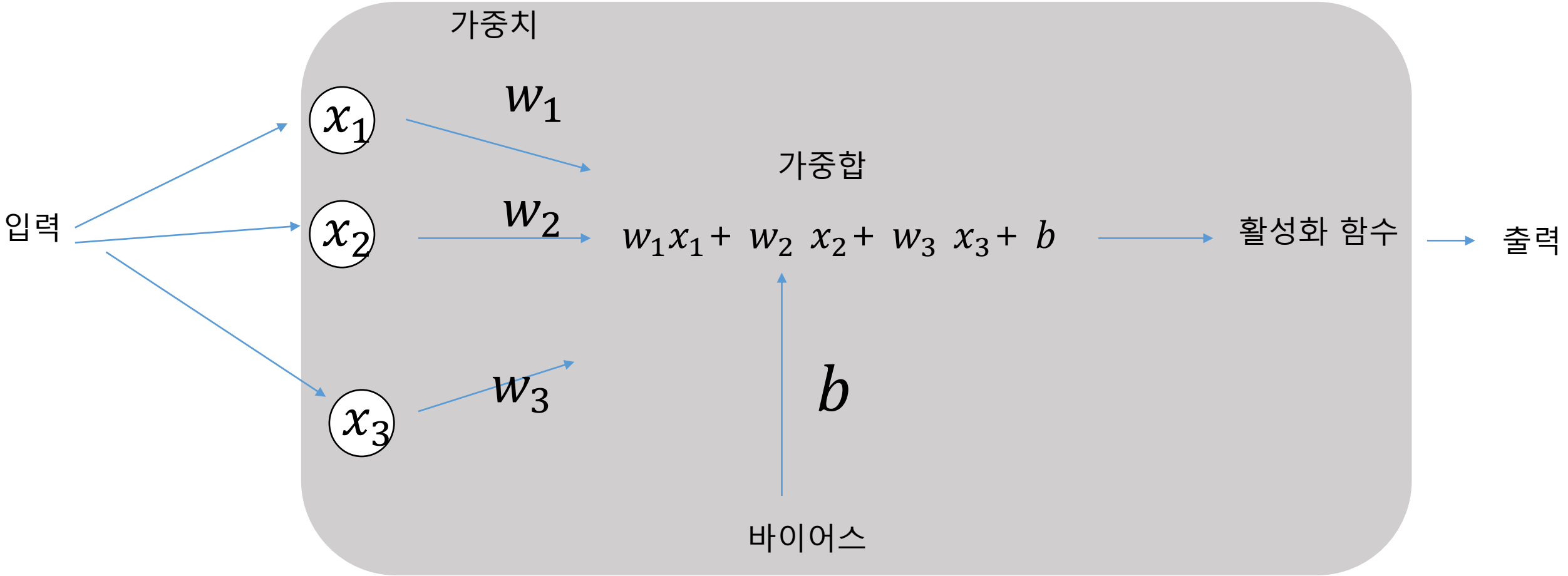
```
from sklearn.model_selection import train_test_split# 훈련 세트와 테스트 세트로 나눕니다
train_input, test_input, train_target, test_target = train_test_split(    perch_length, perch_weight,
random_state=42)
                    train_input = train_input.reshape(-1, 1)
from sklearn.linear_model import LinearRegression

lr = LinearRegression() # 선형 회귀 모델 훈련
lr.fit(train_input, train_target)


# 50cm 농어에 대한 예측 p                print(lr.predict([ [25], [50]    ]))
print(lr.predict([[50]]))

print(lr.coef_, lr.intercept_)

plt.scatter(train_input, train_target)

plt.plot([15, 50], [15*lr.coef_+lr.intercept_, 50*lr.coef_+lr.intercept_])
# 50cm 농어 데이터
plt.scatter(50, 1241.8, marker='^')
plt.show()
```

머신 러닝/ 딥러닝

신경망 (Neural Network)

인공 뉴런(퍼셉트론)으로 이루어짐

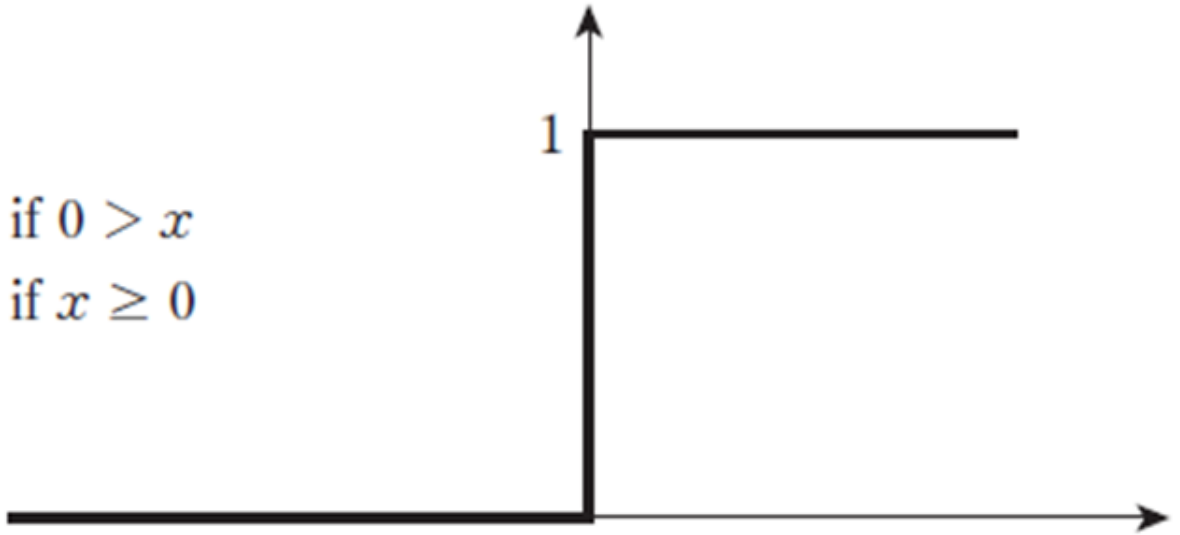
퍼셉트론 (입력 3개)



퍼셉트론 (Perceptron)

계단(step) 함수

$$f(x) = \begin{cases} 0 \text{ if } 0 > x \\ 1 \text{ if } x \geq 0 \end{cases}$$



.

- 가중치는 입력 신호가 출력에 미치는 중요도를 조절
- 바이어스는 뉴런의 활성화 기준
- 가중합 (입력 2개) : w1*x1+w2*x2+b

퍼셉트론 (Perceptron)

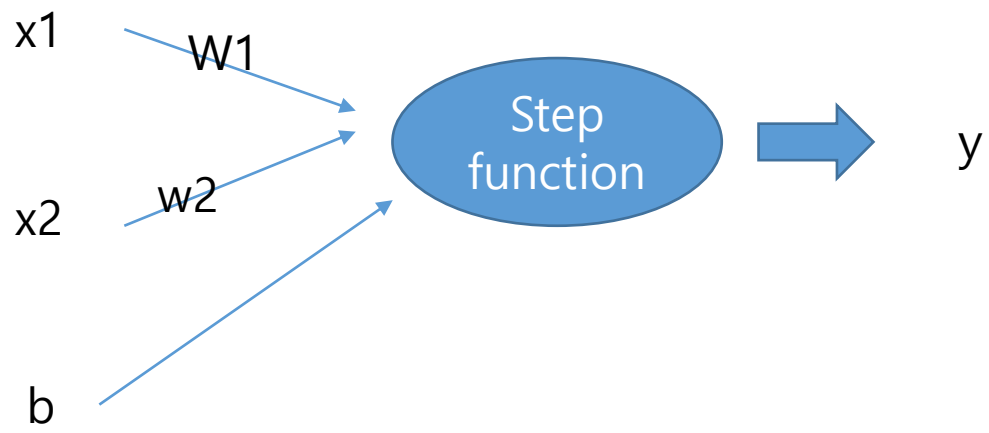
- 뉴런에서는 입력 신호의 가중치 합이 어떤 임계값을 넘는 경우에만 뉴런이 활성화되어서 1을 출력한다. 그렇지 않으면 0을 출력한다.

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

퍼셉트론 (Perceptron)

입력 (1,0)

w1: 1
w2: 1
B=-1.5



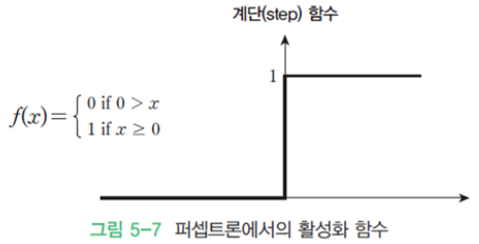


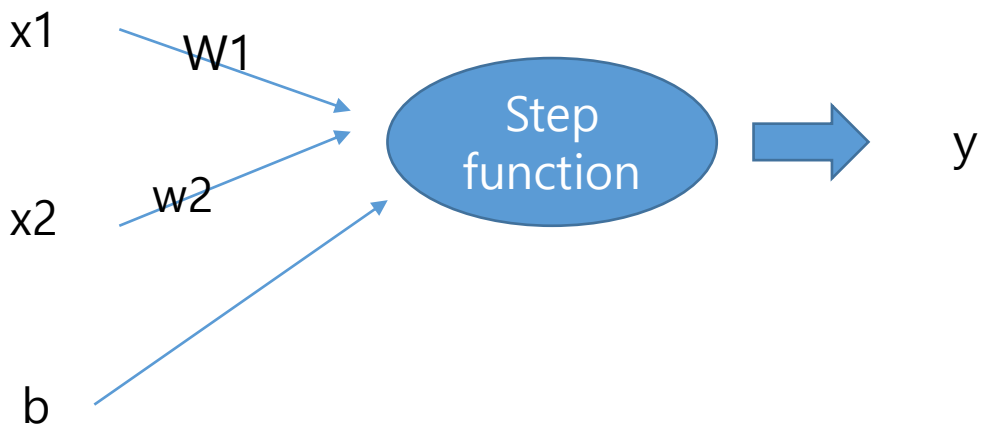
그림 5-7 퍼셉트론에서의 활성화 함수

출력값 y는?

퍼셉트론 (Perceptron)

입력 (1,0)

w1: 1
w2: 1
B=-1.5



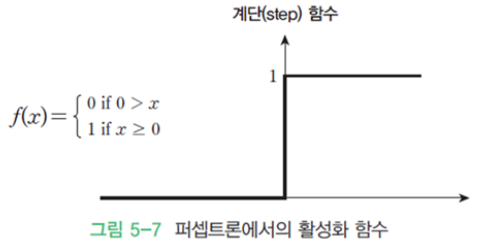


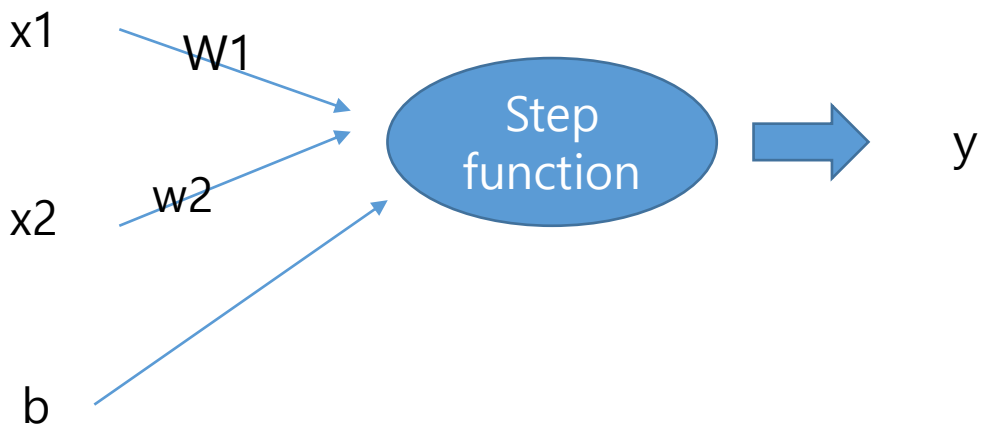
그림 5-7 퍼셉트론에서의 활성화 함수

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

퍼셉트론 (Perceptron)

입력 (1,1)

w1: 1
w2: 1
B=-1.5



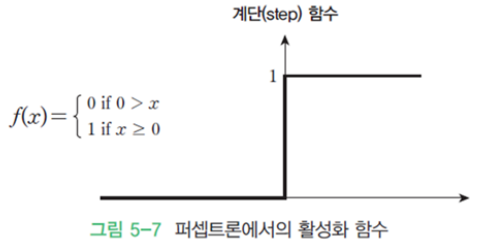


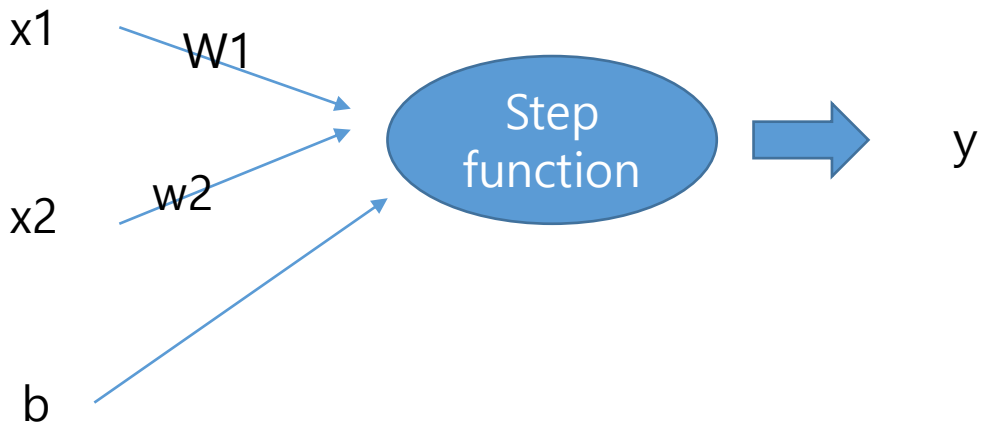
그림 5-7 퍼셉트론에서의 활성화 함수

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

퍼셉트론 (Perceptron)

입력 (1,1)

w1: 1
w2: 1
B=-0.5



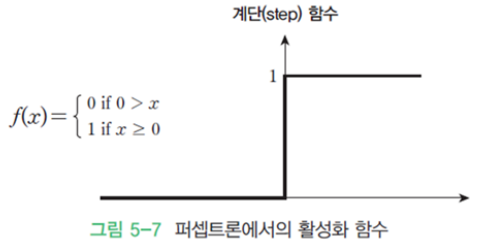


그림 5-7 퍼셉트론에서의 활성화 함수

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

퍼셉트론 (Perceptron) – 논리 연산 수행

AND 연산

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

만족시키는 w1,w2,b 찾기

w1=1,w2=1,b=-1.5 를 테스트 해보기

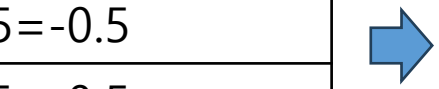
퍼셉트론 (Perceptron) – 논리 연산 수행

w1=1,w2=1,b=-1.5

가중합: x1*w1+x2*w2+b

x1	x2
0	0
1	0
0	1
1	1

가중합
1*0+1*0-1.5=-1.5
1*1+1*0-1.5=-0.5
1*0+1*1-1.5=-0.5
1*1+1*1-1.5=0.5



활성화 함수
(step function)

출력(y)
0
0
0
1

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

과제 1) code upload(*.py), 출력값 입력

퍼셉트론 (Perceptron) 학습

- **퍼셉트론 도 scikit에서 학습 이 가능**

퍼셉트론 (Perceptron) 학습 –scikit learn


```
from sklearn.linear_model import Perceptron

# 논리적 AND 연산 샘플과 정답이다.
X = [[0,0],[0,1],[1,0],[1,1]]                    # 항상 2차원 배열이어야 한다.
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))
```

과제 2) code upload(*.py), 출력 값 입력

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 한 후 예측 결과를 제출

과제 3) 결과값 제출

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

과제4) 결과값 제출

퍼셉트론 (Perceptron)

- **XOR 연산은 퍼셉트론으로 학습이 불가능 하다**

퍼셉트론 (Perceptron)

- 퍼셉트로은 간단한 xor문제고 해결 못함.

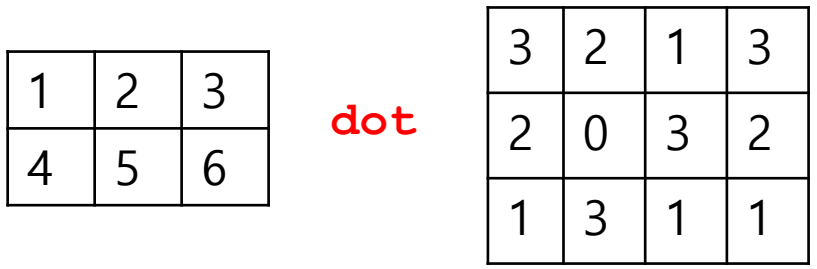
- 해결 하기 위해선 => 다층 퍼셉트론 필요 (Multilayer perceptron)

곱셈


```
arr1 = np.array([[1, 2], [3, 4], [5, 6]])
arr2 = np.array([[2, 2], [2, 2], [2, 2]])
result = arr1 * arr2
result
array([[ 2,  4],
       [ 6,  8],
       [10, 12]])
```

Numpy 행렬 곱셈


```
import numpy as np

a = np.array([[1, 2, 3], [4, 5, 6]])
b = np.array([[3, 2,1,3], [2, 0, 3, 2], [1, 3, 1, 1])

c1=a@b
c2=np.dot(a,b)
c3=a.dot(b)
C4=np.matmult(a,b)
```



c1,c2,c3,c4 모두 행렬 곱셉을 나타냄

1차원 행렬 곱셈

3	2	1

dot

4	5	6

=3*4+2*5+1*6=28

- **A=[1,2,3]**
- **B=[4,5,6]**

- **A*B 일반 곱셈**

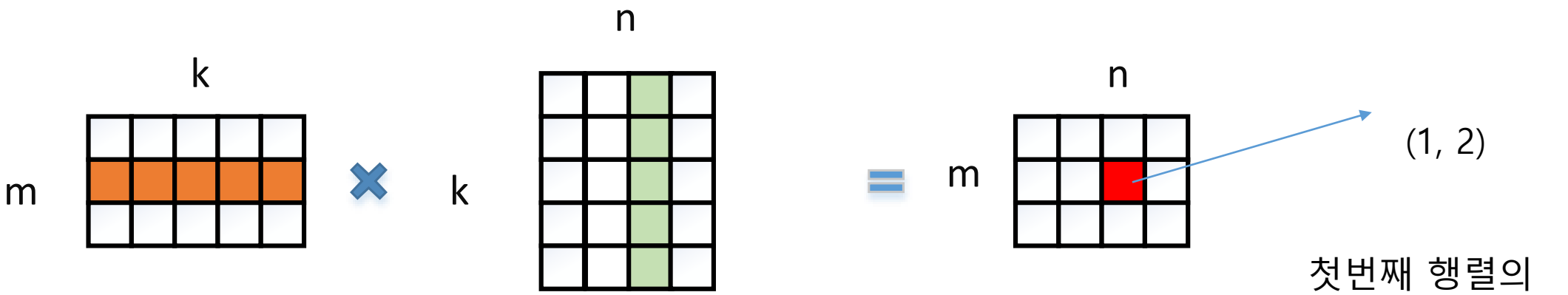
  => A*B => [4,10,18]

- **A*B 행렬 곱셈**

  np.dot(A,B)

  - 1*4+2*5+3*6= 4+10+18= 32

행렬 곱셈



첫번째 행렬의
1 행

두번째 행렬의
2열

1차원 행렬 곱샘

(m,k) x (k,n) = (m, n)

행렬 곱셈

1	2	3
4	5	6

dot

3	2	1	3
2	0	3	2
1	3	1	1

=

1)	2)	3)	10
28	26	25	28

(2,3) (3,4) (2,4)

1) 1*3+2*2+3*1=10

행렬 곱셈

1	2	3
4	5	6

dot

3	2	1	3
2	0	3	2
1	3	1	1

=

1)	2)	3)	10
28	26	25	28

(2,3) (3,4) (2,4)

2) 1*2+2*0+3*3=11

Numpy 행렬 곱셈

1	2	3
4	5	6

dot

3	2	1	3
2	0	3	2
1	3	1	1

=

1)	2)	3)	10
28	26	25	28

(2,3)

(3,4)

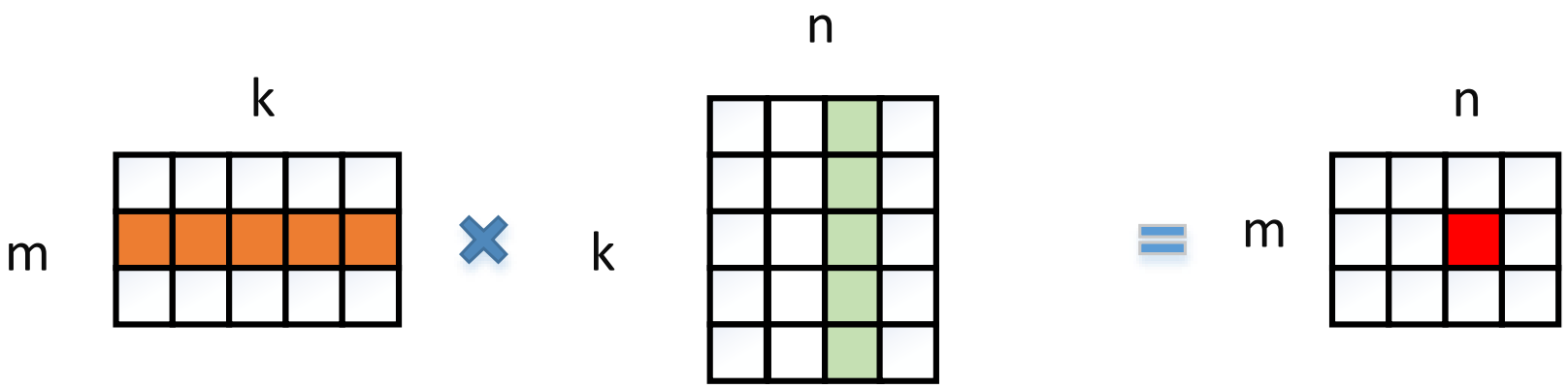
(2,4)

3)

행렬 곱셈


```
>>> arr1 = np.array([[1, 2, 3], [4, 5, 6], [7, 8, 9]])
>>> arr2 = np.array([[2, 2], [2, 2], [2, 2]])
>>> result = arr1 @ arr2              # arr1.dot(arr2) 도 가능
array([[12, 12],
       [30, 30],
       [48, 48]])
```

행렬 곱셈



(m,k) x (k,n) = (m, n)

행렬 곱셈이 가능하지 않은것은?

1) $\begin{bmatrix}1 & 2 & 3\\4 & 5 & 6\end{bmatrix} \times \begin{bmatrix}1 & 2\\3 & 4\\5 & 6\end{bmatrix}$

2) $\begin{bmatrix}1 & 2\\3 & 4\\5 & 6\end{bmatrix} \times \begin{bmatrix}1 & 2 & 3\\4 & 5 & 6\end{bmatrix}$

3) $\begin{bmatrix}3 & 2 & 1\\1 & 1 & 1\\1 & 2 & 3\end{bmatrix} \times \begin{bmatrix}5\\4\\3\end{bmatrix}$

4) $\begin{bmatrix}1 & 2 & 3\end{bmatrix} \times \begin{bmatrix}3 & 2 & 4\\1 & 5 & 1\\1 & 2 & 3\end{bmatrix}$

5) $\begin{bmatrix}1 & 2 & 3\end{bmatrix} \times \begin{bmatrix}5\\4\\3\end{bmatrix}$

6) $\begin{bmatrix}3 & 2 & 4\\1 & 5 & 1\\1 & 2 & 3\end{bmatrix} \times \begin{bmatrix}1 & 2 & 3\\4 & 5 & 6\end{bmatrix}$

- **A=[1,2,3]**
- **B=[4,5,6]**

- **A*B 일반 곱셈**

  => A*B => [4,10,18]

- **A*B 행렬 곱셈**
  - 1*4+2*5+3*6= 4+10+18= 32

행렬 곱셈

W=[w1, w2], X=[x1,x2]

WX=w1*x1+w2*x2 WX=np. matmult(W, X)

W=[1,2], X= [3,4] WX= ?

np. matmult(W,X)

W= $\begin{bmatrix} w1 & w2 \\ w3 & w4 \end{bmatrix}$ X=$\begin{bmatrix} x1 \\ x2 \end{bmatrix}$

W = $\begin{bmatrix} 1 & 2 \\ 3 & 4 \end{bmatrix}$ X=$\begin{bmatrix} 5 \\ 6 \end{bmatrix}$

WX= $\begin{bmatrix} w1x1 + & w2x2 \\ w3x1 + & w4x2 \end{bmatrix}$

WX=?

np. matmult(W,X)

행렬 곱셈

W=[w1, w2], X=[x1,x2]

WX=w1*x1+w2*x2

WX=np. matmult(W, X)

W=[1,2], X= [3,4] WX= ?

np. matmult(W,X)

```
import numpy as np
W=np.array([ 1 ,2 ])
X=np.array([3,4])

print(W*X)       => [3 8]
print(np. matmult(W,X))  => 11
```

W= $\begin{bmatrix} w1 & w2 \\ w3 & w4 \end{bmatrix}$ X=$\begin{bmatrix} x1 \\ x2 \end{bmatrix}$

WX= $\begin{bmatrix} w1x1 + & w2x2 \\ w3x1 + & w4x2 \end{bmatrix}$

W = $\begin{bmatrix} 1 & 2 \\ 3 & 4 \end{bmatrix}$ X=$\begin{bmatrix} 5 \\ 6 \end{bmatrix}$

WX=?

np. matmult(W,X)

```
import numpy as np
W=np.array([[ 1 ,2 ] ,[3, 4]])
X=np.array([5,6])
print(np. matmult(W,X))


 [17 39]
```

행렬 곱셈 연습 (과제 5)

5-1. W=[1, 2], X=[0,3]

1) W*X 의 결과는?

2) WX 의 결과는?

5-2. $W = \begin{bmatrix} 1 & 2 \\ 0 & 4 \end{bmatrix}$ $X = \begin{bmatrix} 5 \\ 2 \end{bmatrix}$

1) WX의 결과는?

5-3. $W = \begin{bmatrix} 7 & 3 \\ 1 & 2 \end{bmatrix}$ $X = \begin{bmatrix} 0 \\ 4 \end{bmatrix}$

1) WX 의 결과는?

5-4. W=[3, 1], X=[2,1]

1) W*X 의 결과는?

2) WX 의 결과는?

5-5. $W = \begin{bmatrix} 1 & 2 \\ 0 & 4 \end{bmatrix}$ $X = \begin{bmatrix} 5 \\ 2 \end{bmatrix}$

1) WX 의 결과는?

결과 입력

WX=np.matmult(W,X)

행렬 곱셈 연습 (과제 6)

Matrix.ipynb

1) $\begin{bmatrix}1 & 2 & 3\\4 & 5 & 6\end{bmatrix} \times \begin{bmatrix}1 & 2\\3 & 4\\5 & 6\end{bmatrix}$ 를 행렬 곱셈으로 구현하고 실행 하시오

행렬 곱셈

W=[w1, w2], X=[x1,x2]

WX=w1*x1+w2*x2

$$W=\begin{bmatrix} w1 & w2 \\ w3 & w4 \end{bmatrix} \quad X=\begin{bmatrix} x1 \\ x2 \end{bmatrix}$$

$$WX=\begin{bmatrix} w1x1+ & w2x2 \\ wx1+ & w4x2 \end{bmatrix}$$

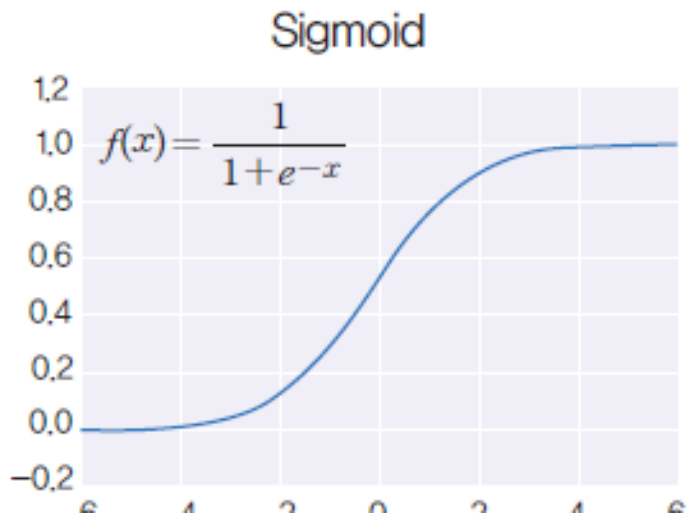
w1*x1+w2*x2+b =>WX+b

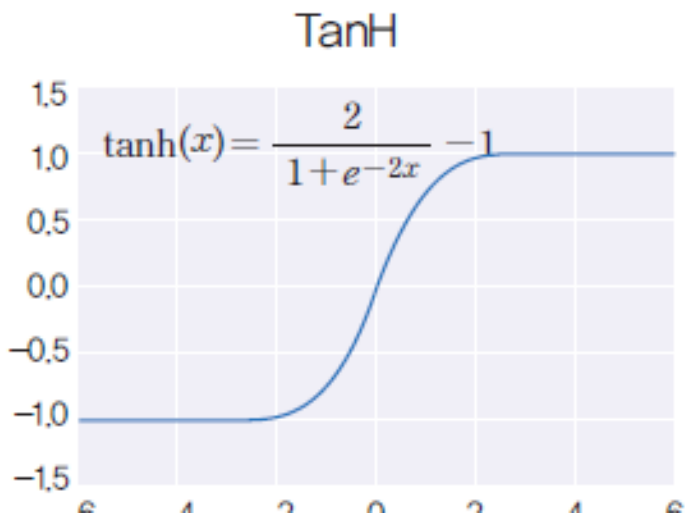
Np dot 하고 그냥 곱셈 차이

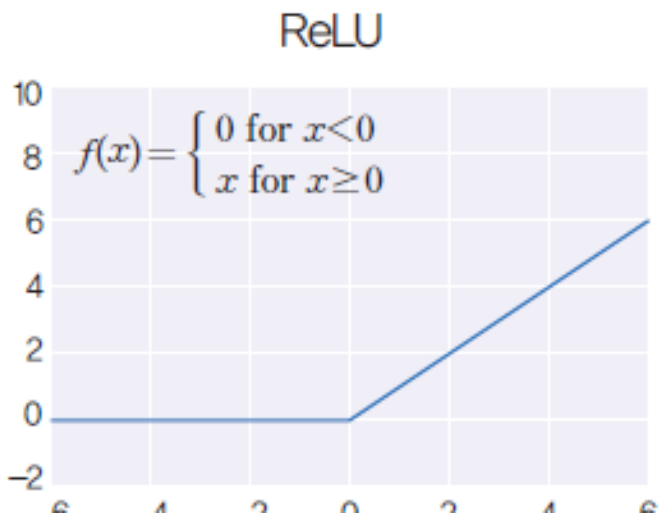
활성화 함수

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

활성화 함수


Sigmoid
$f(x)=\frac{1}{1+e^{-x}}$
1.2
1.0
0.8
0.6
0.4
0.2
0.0
−0.2
6
4
2
0
2
4
6

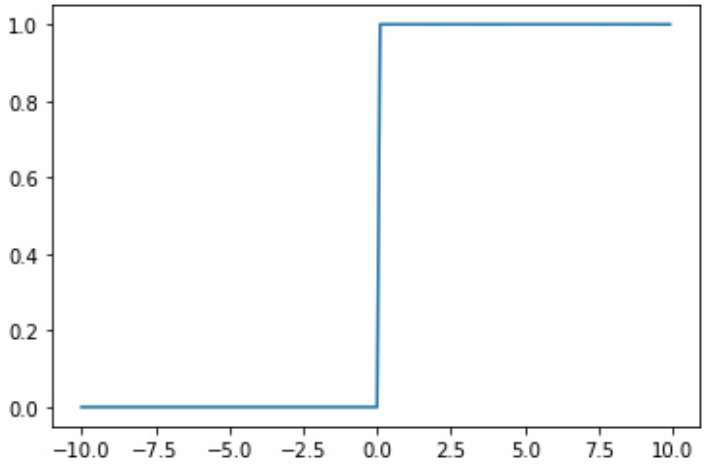

TanH
$\tanh(x)=\frac{2}{1+e^{-2x}}-1$
1.5
1.0
0.5
0.0
−0.5
−1.0
−1.5
6
4
2
0
2
4
6


ReLU
$f(x)=\begin{cases}0 \text{ for } x<0\\ x \text{ for } x\geq 0\end{cases}$
10
8
6
4
2
0
−2
6
4
2
0
2
4
6

활성화 함수 (step)

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

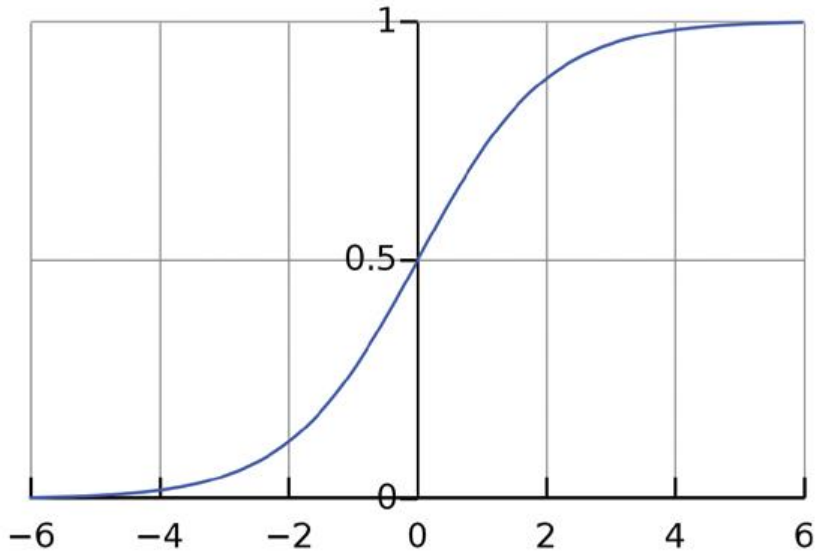
$$f(x) = \begin{cases} 1 & x > 0 \\ 0 & x \leq 0 \end{cases}$$



활성화 함수 (시그모이드)

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

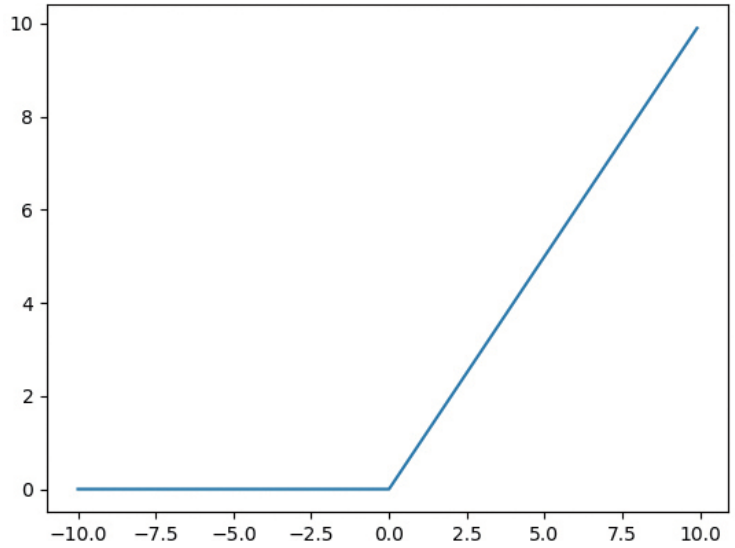
$$f(x) = \frac{1}{1+e^{-x}}$$



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.

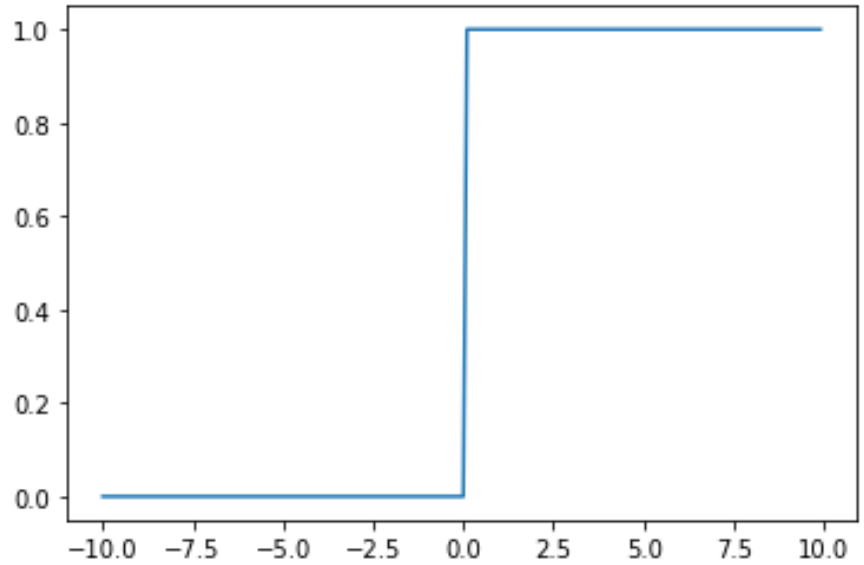
$$f(x) = \begin{cases} x & x > 0 \\ 0 & x \leq 0 \end{cases}$$



Step 함수


```
def step(x):
  result = x > 0.000001                 # True 또는 False
  return result.astype(np.int)          # 정수로 반환
```

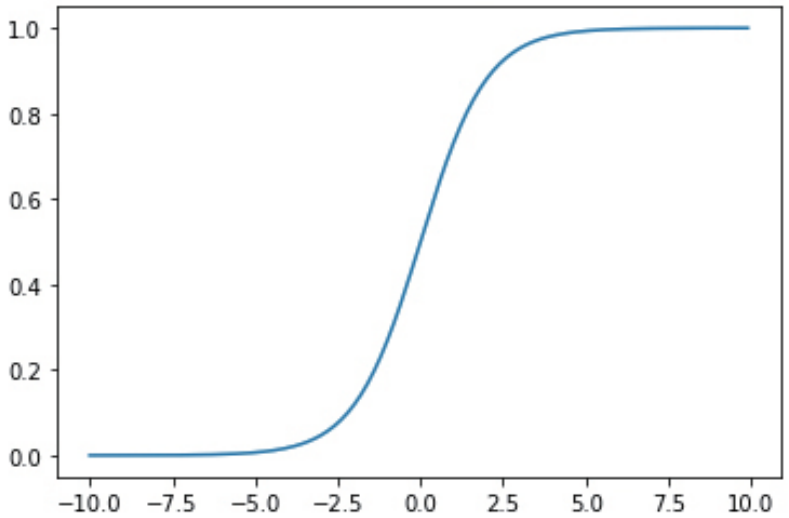
```
import numpy as np
import matplotlib.pyplot as plt

x = np.arange(-10.0, 10.0, 0.1)
y = step(x)
plt.plot(x, y)
plt.show()
```



시그모이드 함수

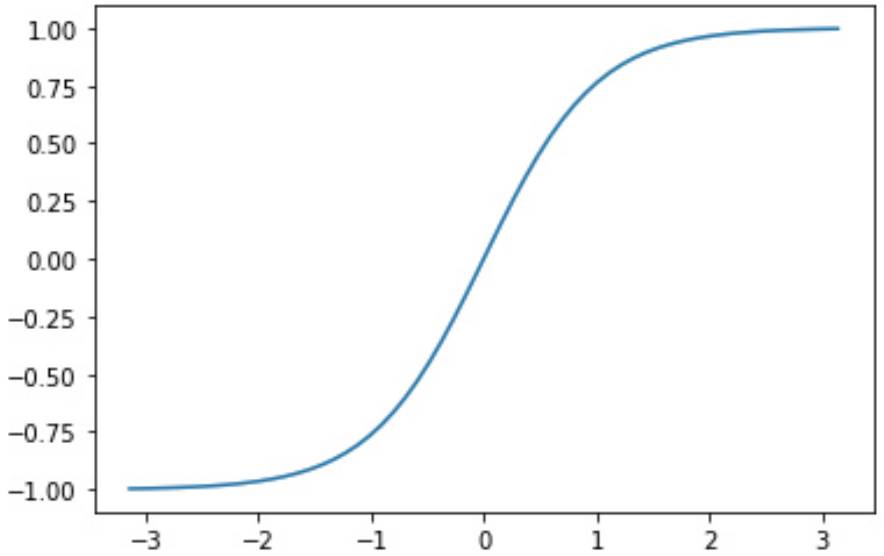

```
import numpy as np
import matplotlib.pyplot as plt

def sigmoid(x):
  return 1.0 / (1.0 + np.exp(-x))

x = np.arange(-10.0, 10.0, 0.1)
y = sigmoid(x)
plt.plot(x, y)
plt.show()
```

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



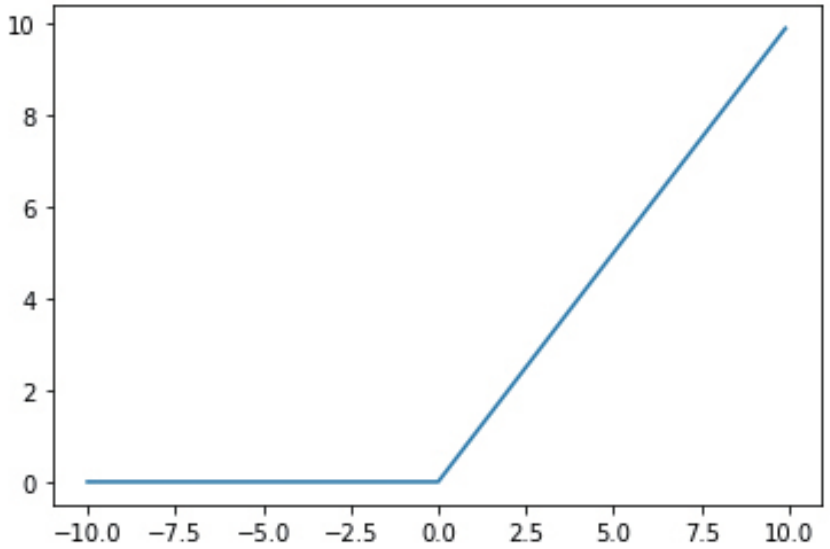
Tanh 함수


```
import numpy as np
import matplotlib.pyplot as plt

x = np.linspace(-np.pi, np.pi, 60)
y = np.tanh(x)
plt.plot(x, y)
plt.show()
```



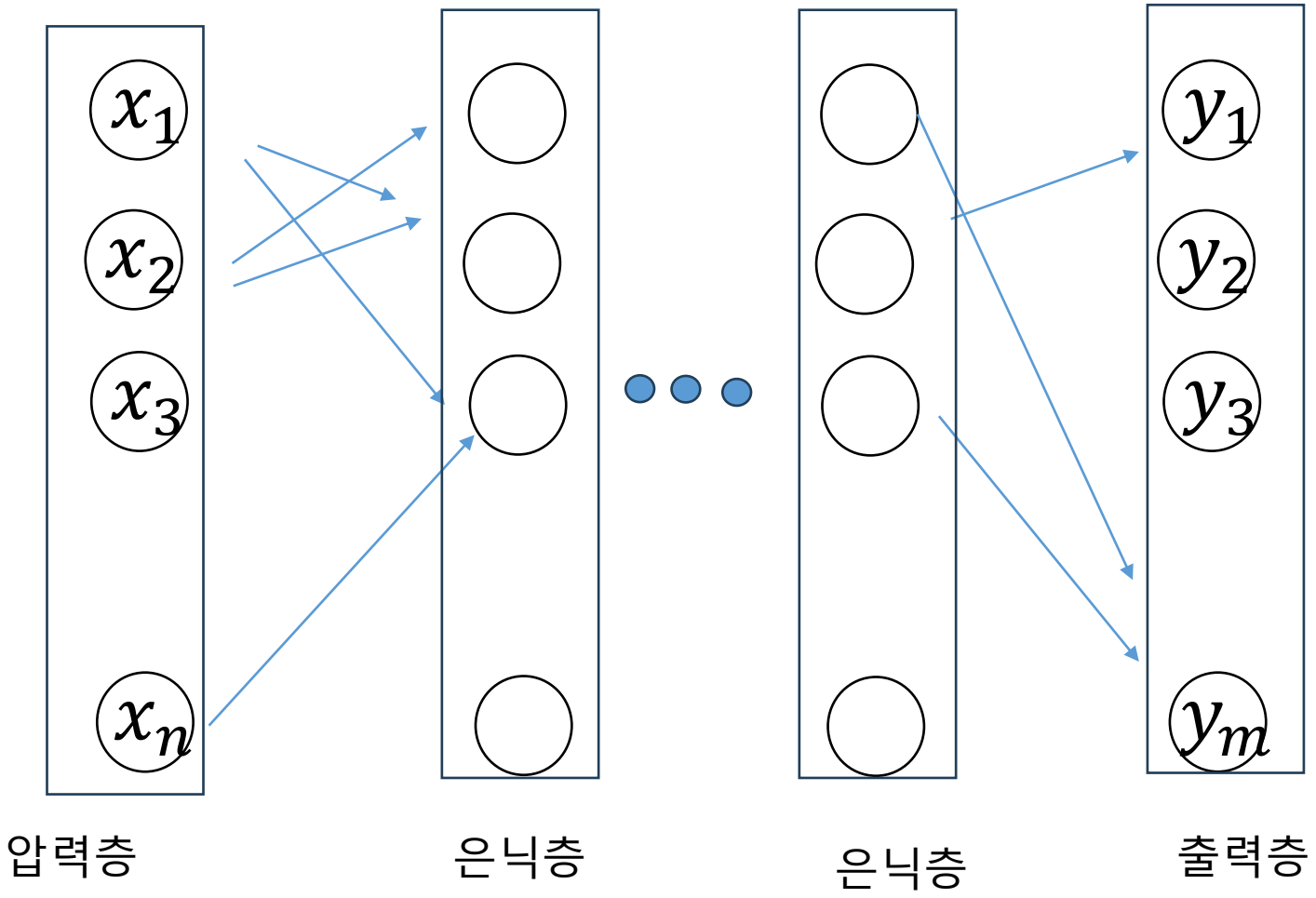
relu 함수


```
import numpy as np
import matplotlib.pyplot as plt

def relu(x):
  return np.maximum(x, 0)

x = np.arange(-10.0, 10.0, 0.1)
y = relu(x)
plt.plot(x, y)
plt.show()
```

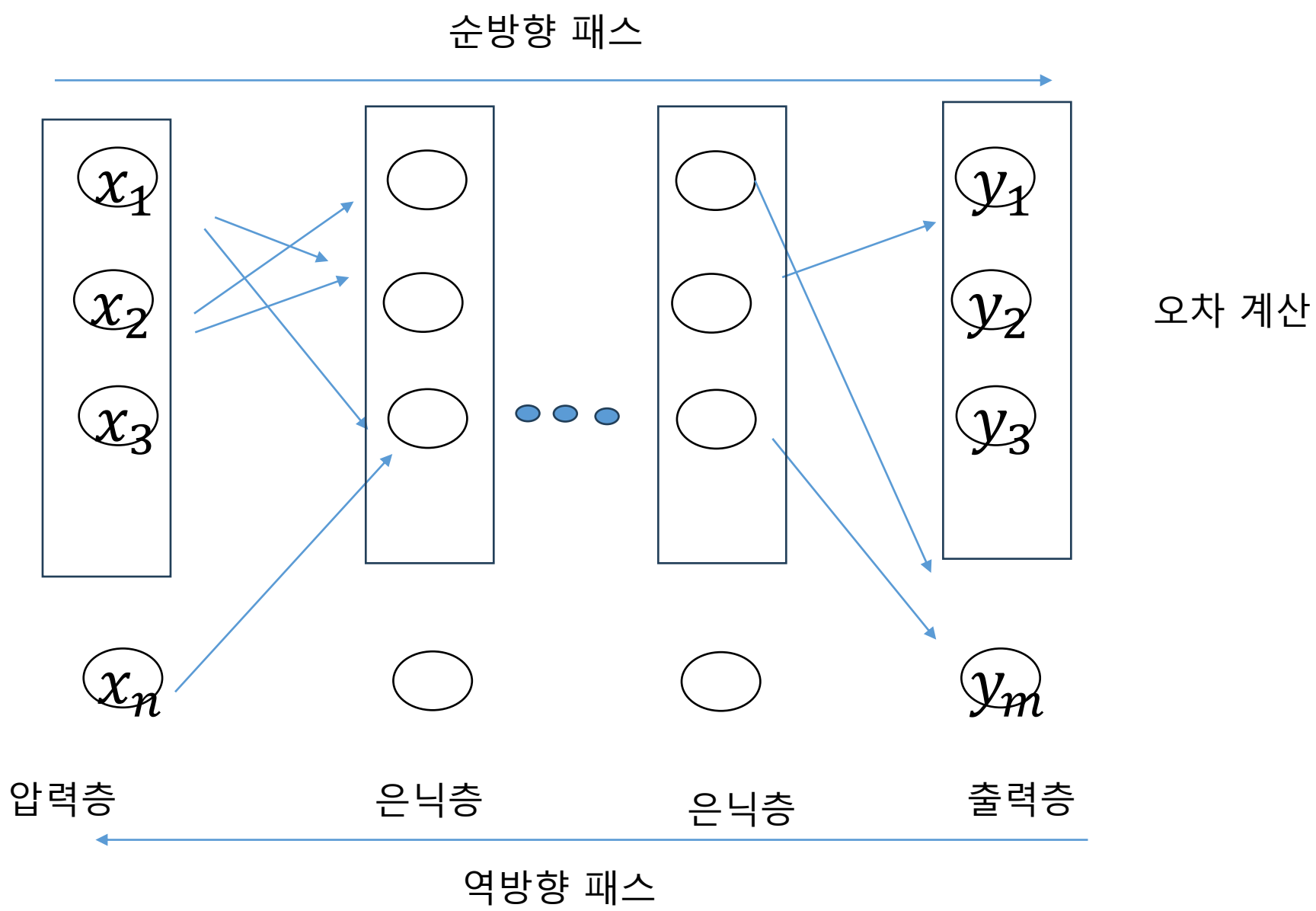


MLP

- **다층 퍼셉트론(multilayer perceptron: MLP): 입력층과 출력층 사이에 은닉층(hidden layer)을 존재**

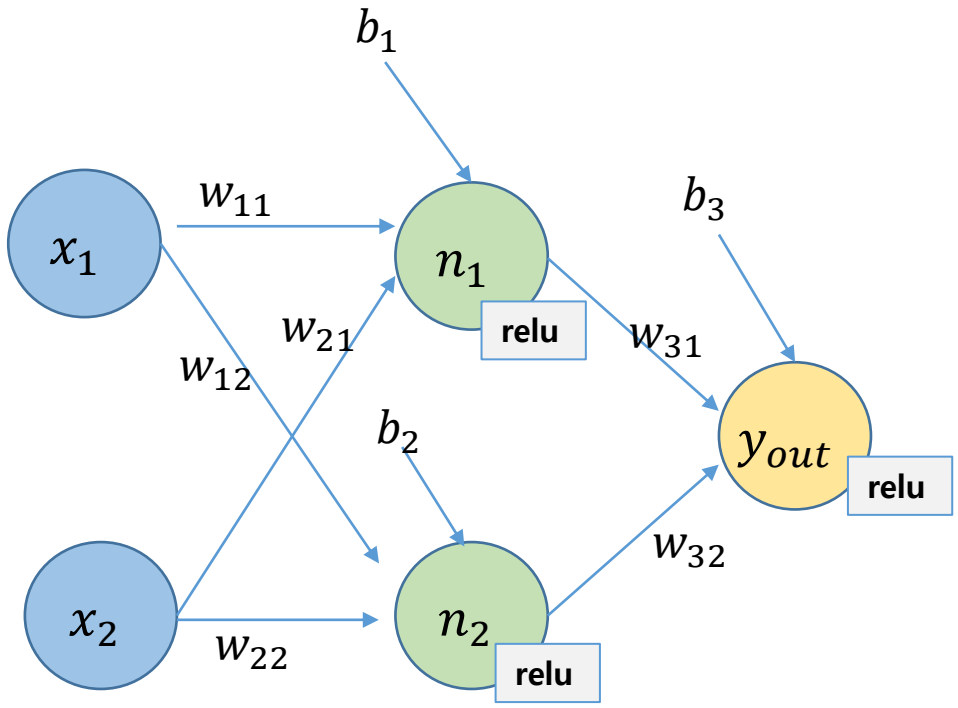


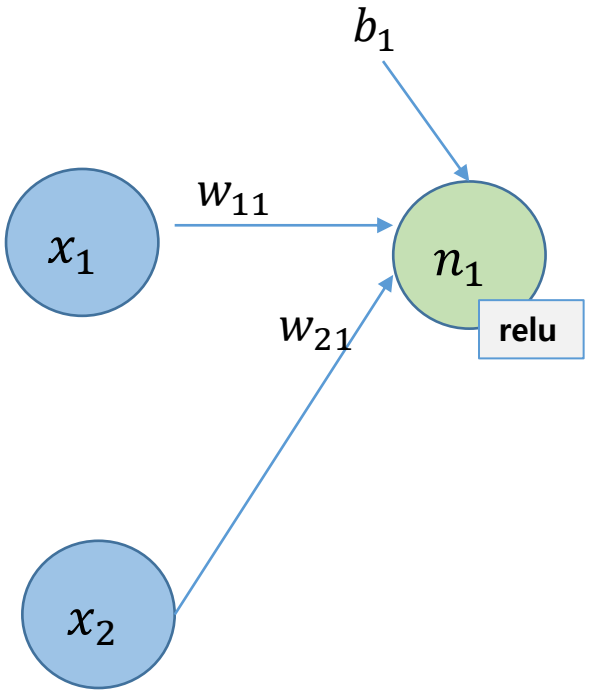
MLP



다중 퍼셉트론 (Multi Layer Perceptron: MLP)

입력 (1,0)

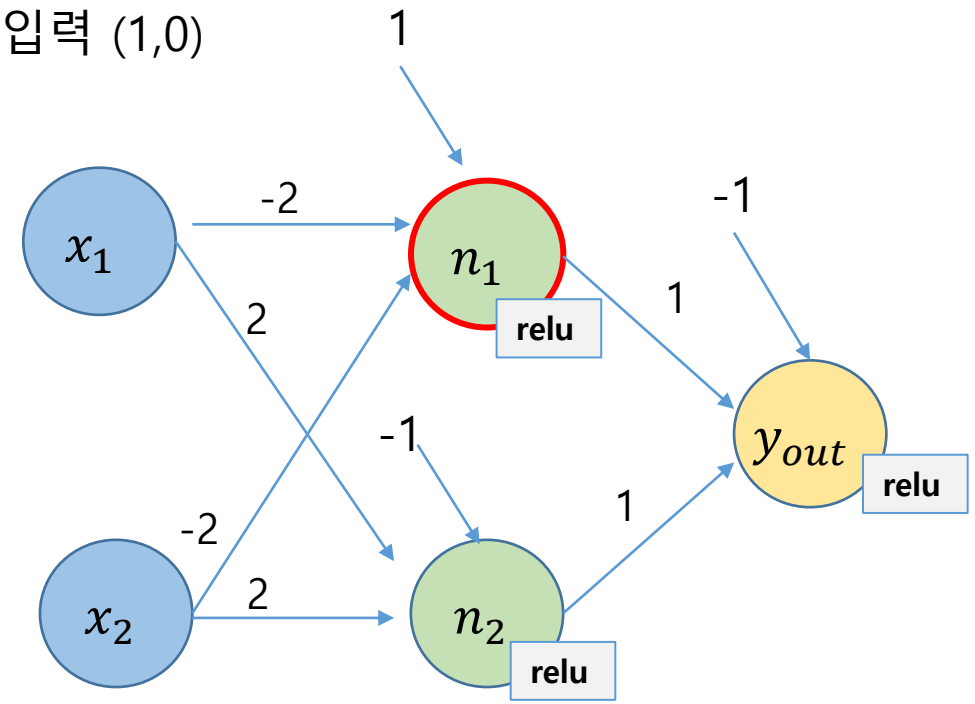




다중 퍼셉트론 (Multi Layer Perceptron: MLP)

인하공전 컴퓨터 정보과

입력 (1,0)

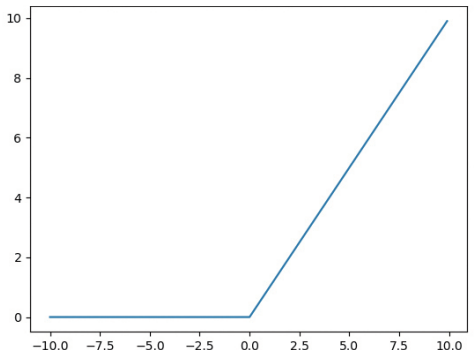


다음 다중 퍼셉트론의 출력은?

w_{12}

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

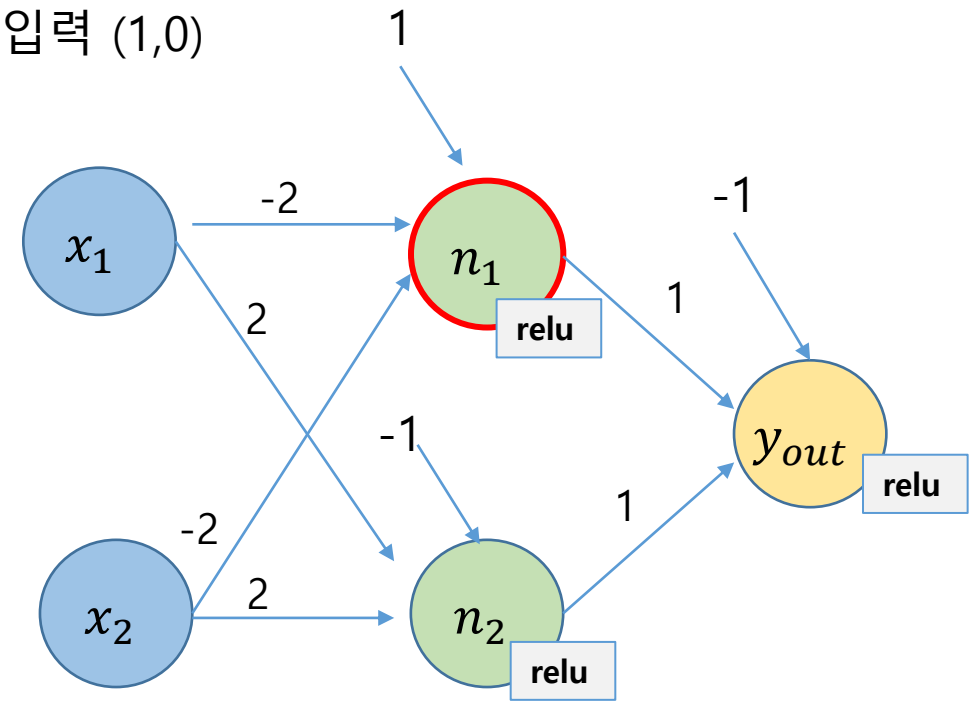
Relu



다중 퍼셉트론 (Multi Layer Perceptron: MLP)

인하공전 컴퓨터 정보과

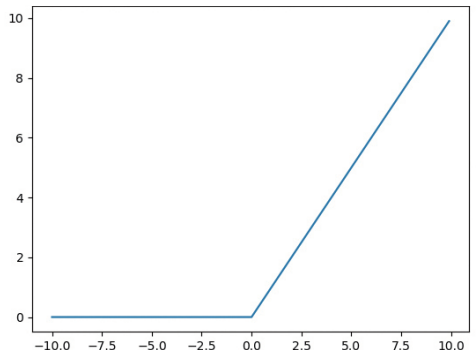
입력 (1,0)



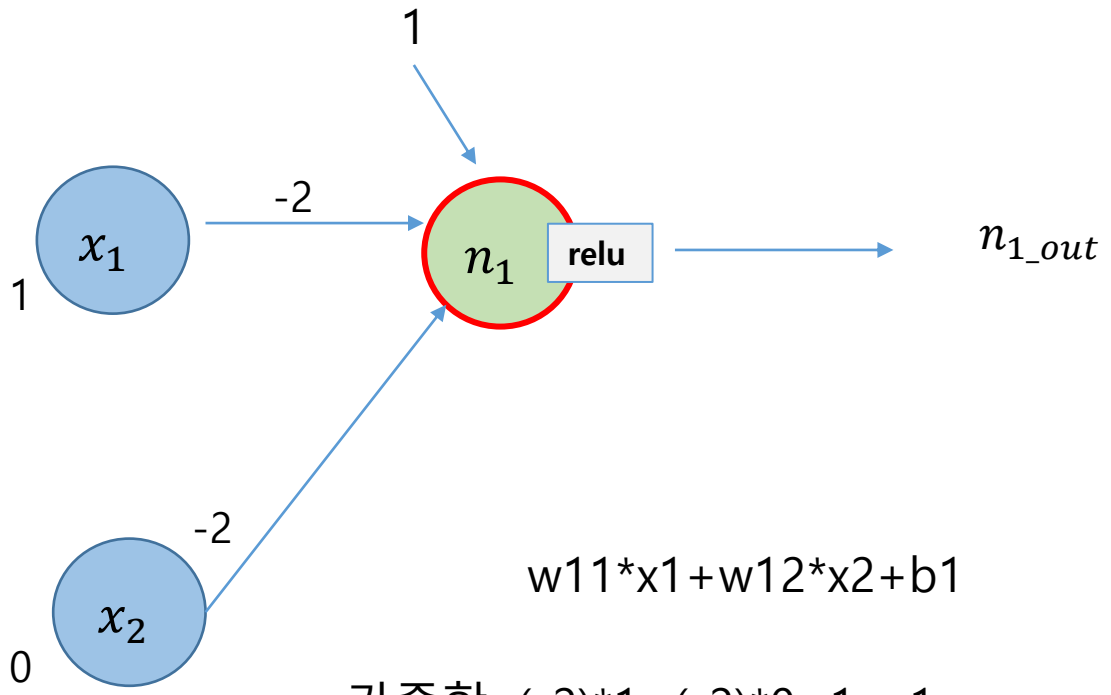
w_{12}

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

Relu



다음 다중 퍼셉트론의 출력은?



w11*x1+w12*x2+b1

가중합: (-2)*1+(-2)*0+1=-1

Relu(-1)=0

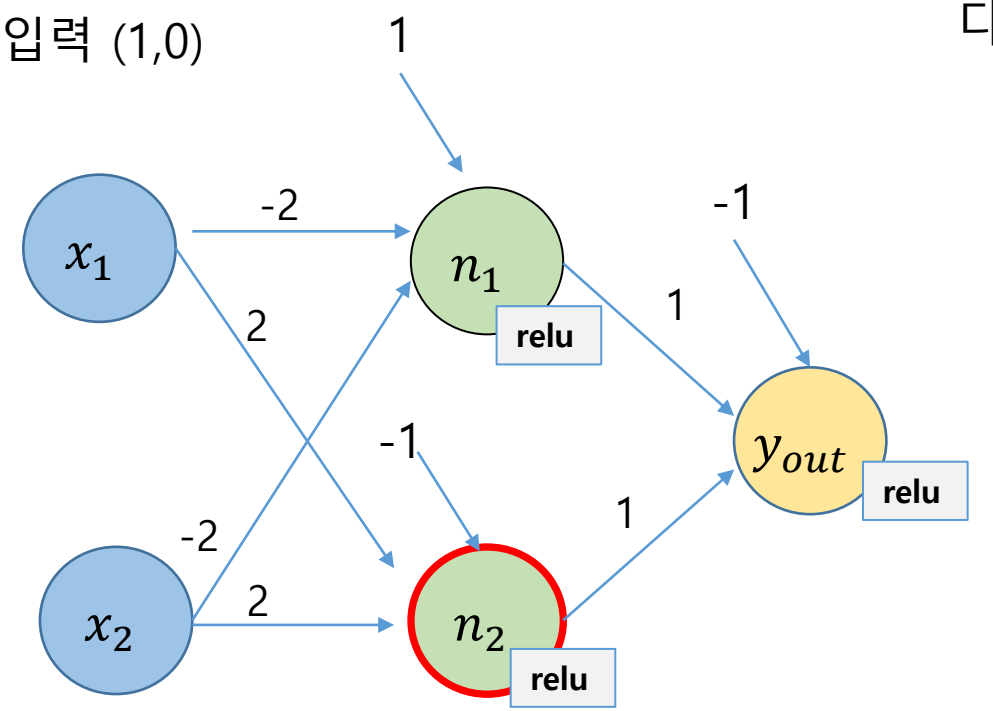
n1_out=0

다중 퍼셉트론 (Multi Layer Perceptron: MLP)

인하공전 컴퓨터 정보과

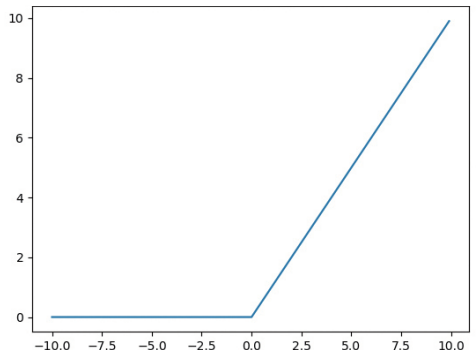
입력 (1,0)

다음 다중 퍼셉트론의 출력은?



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

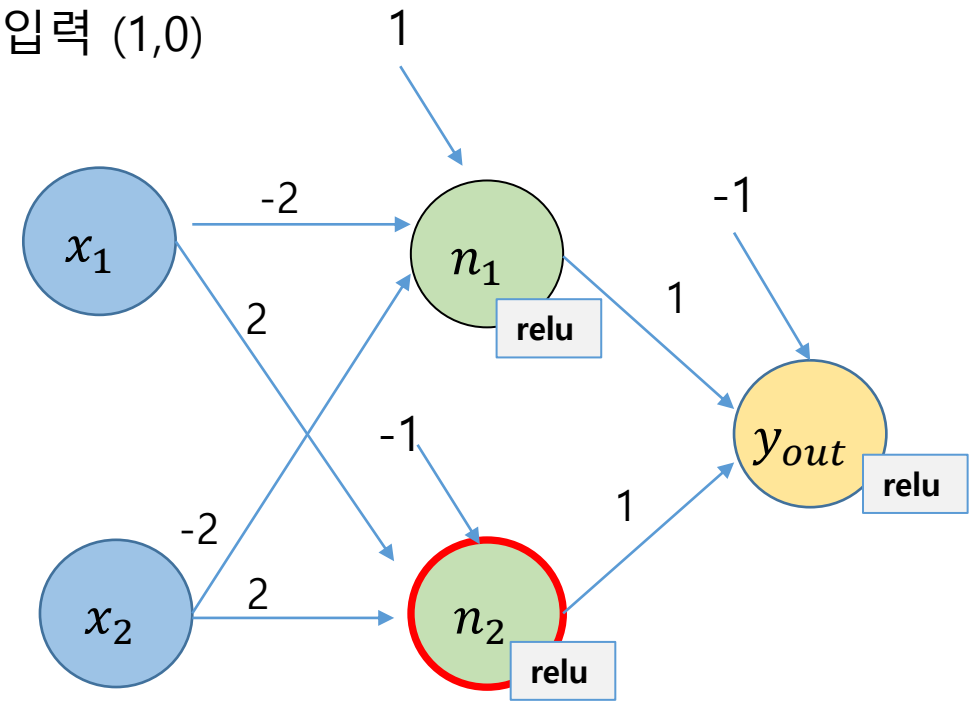
Relu



다중 퍼셉트론 (Multi Layer Perceptron: MLP)

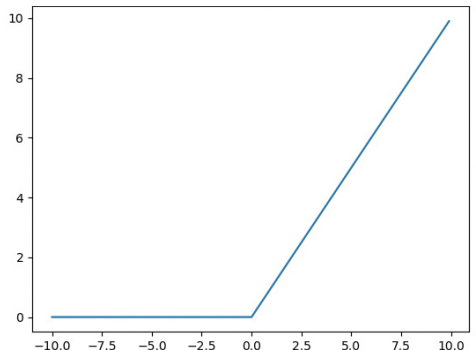
인하공전 컴퓨터 정보과

입력 (1,0)



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

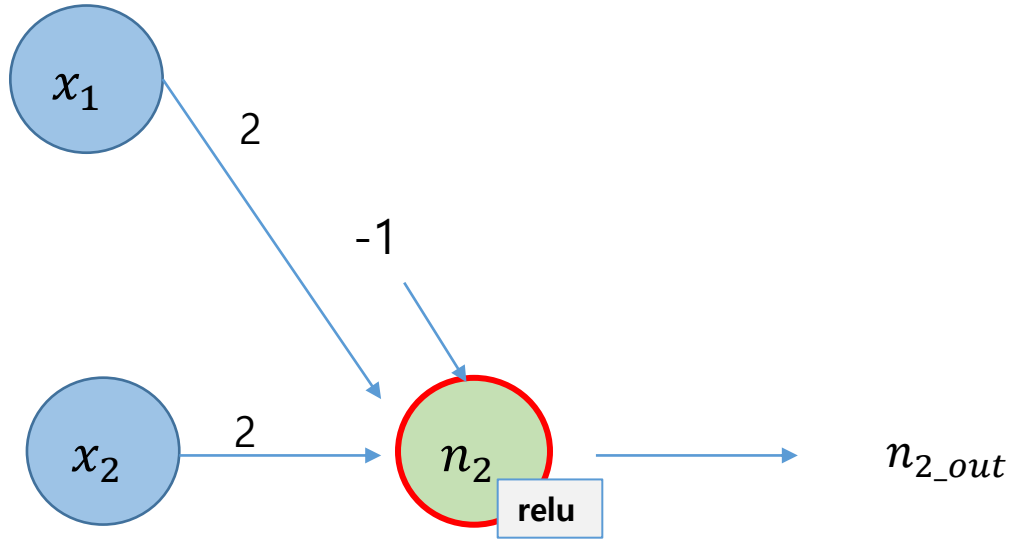
Relu



참고자료 : 딥러닝 express

다음 다중 퍼셉트론의 출력은?

입력 (1,0)



w21*x1+w22*x2+b2

가중합: 1*(2)+2*(0)-1=1

Relu(1)=1

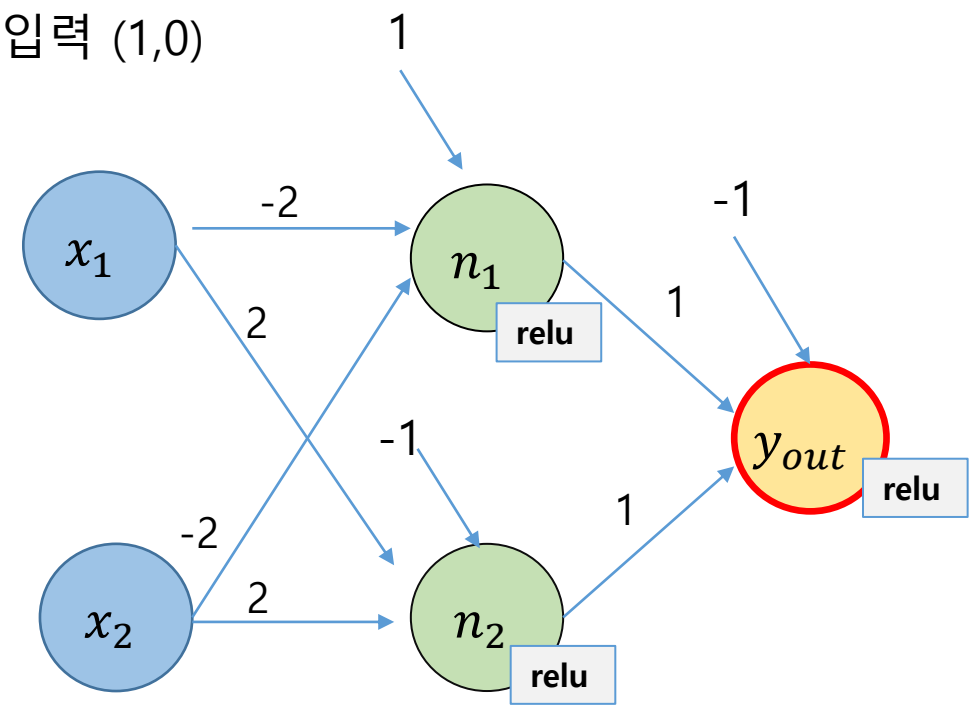
n2_out=1

다중 퍼셉트론 (Multi Layer Perceptron: MLP)

인하공전 컴퓨터 정보과

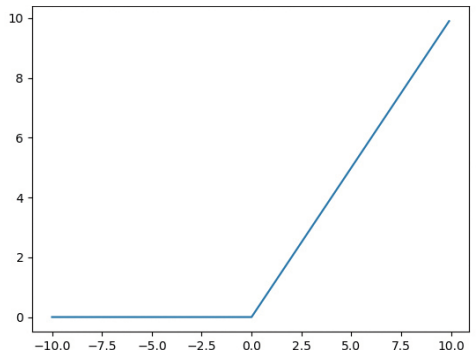
입력 (1,0)

다음 다중 퍼셉트론의 출력은?



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

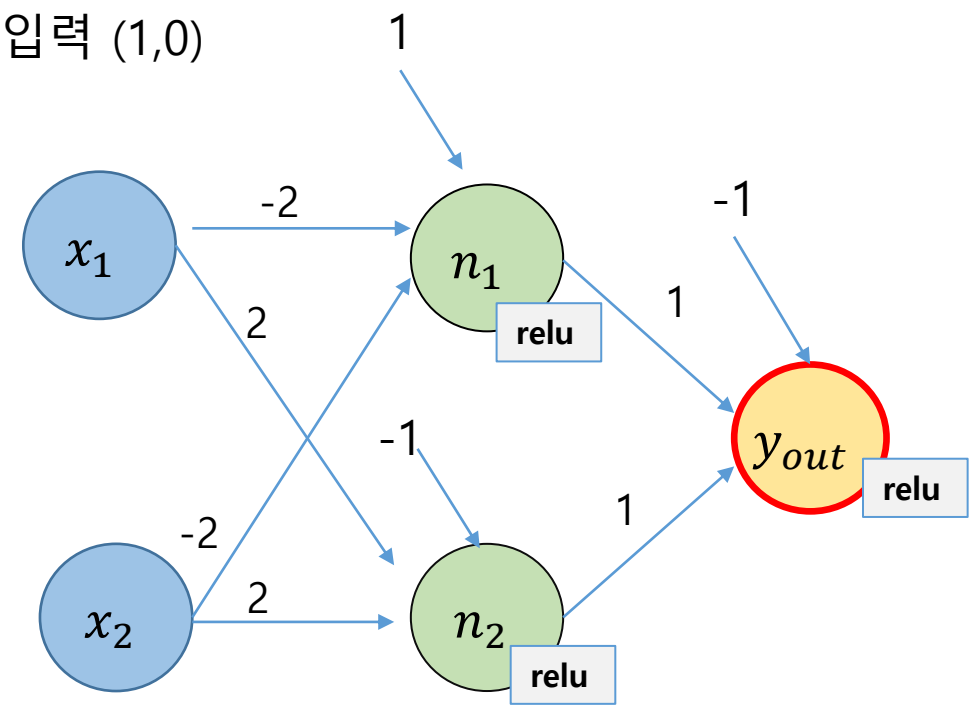
Relu



다중 퍼셉트론 (Multi Layer Perceptron: MLP)

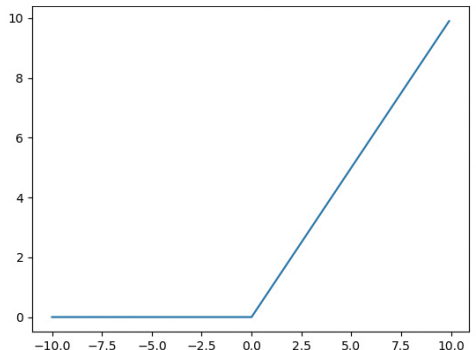
인하공전 컴퓨터 정보과

입력 (1,0)

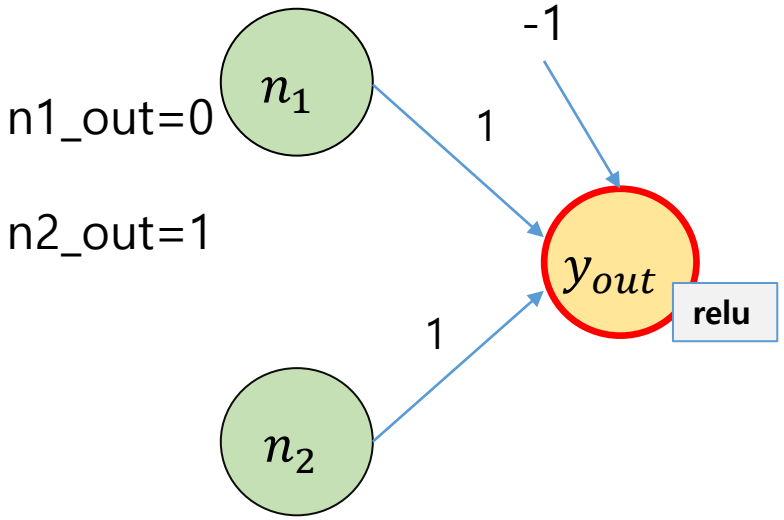


$$f(x) = \begin{cases} x & x > 0 \\ 0 & x \leq 0 \end{cases}$$

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.

역방향 패스 (오차 역전파)
신경망 내부의 가중치는 오차 역전파 방법을 사용해 수정함

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
```

참고자료 : 딥러닝 express

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(w)를 준 뒤 결과(y_{out})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.

참고자료 : 딥러닝 express

손실 함수 계산

- 신경망에서 학습을 시킬 때 는 실제 출력과 원하는 출력 사이의 오차를 이용한다.

- 오차를 계산하는 함수를 손실함수(loss function)라고 한다.

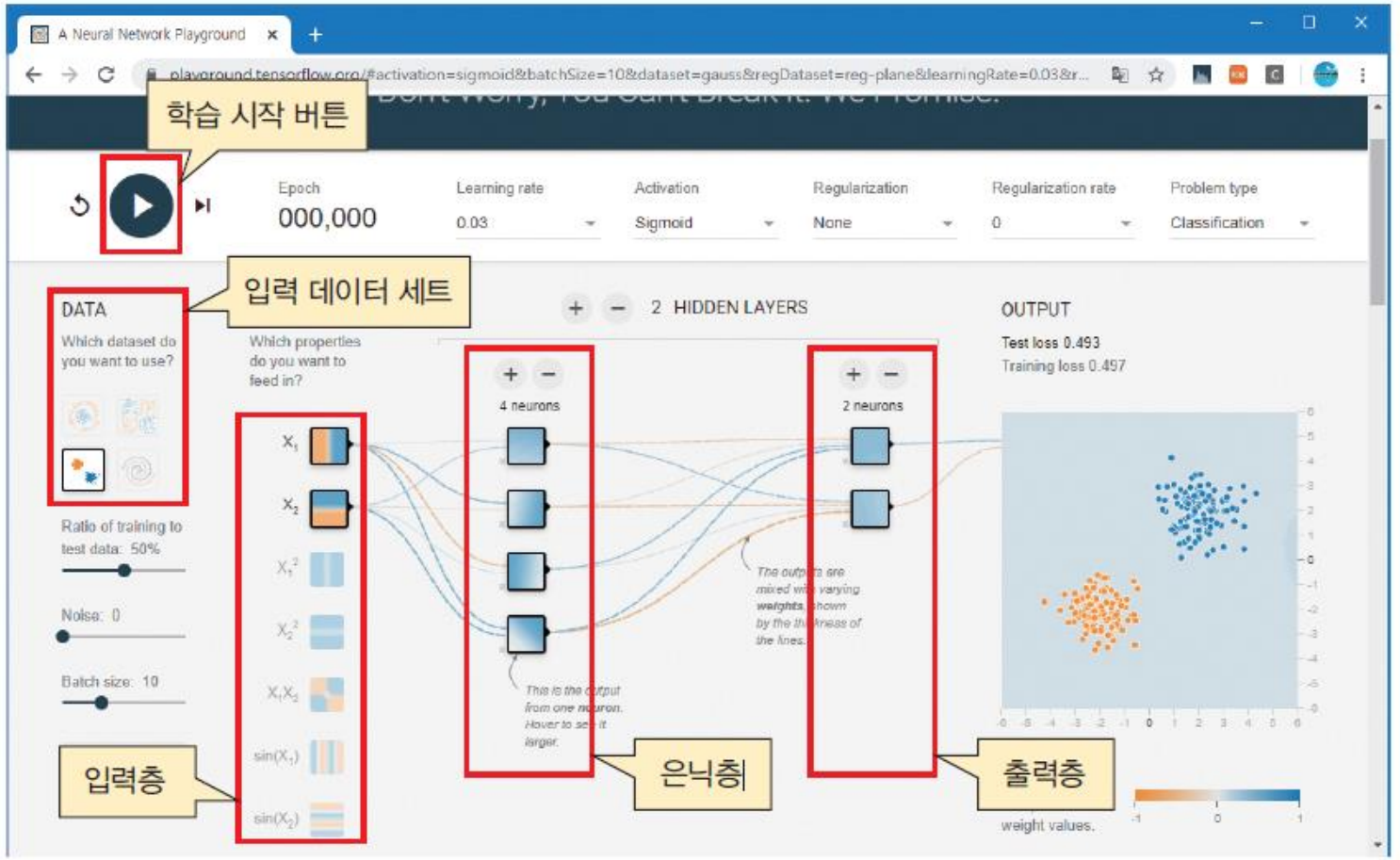
평균 제곱 오차(MSE)

- 예측값과 정답 간의 평균 제곱 오차

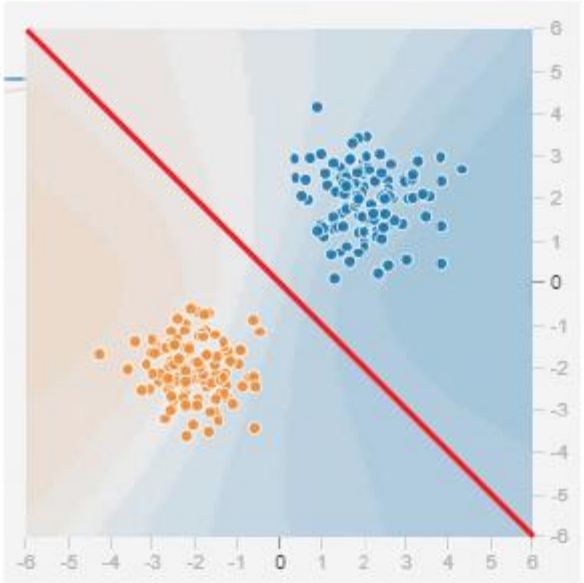
구글의 플레이그라운드

- 사이트(https://playground.tensorflow.org)
- 텐서 플로우 플레이그라운드는 자바 스크립트로 작성된 웹 애플리케이션으로 웹 브라우저에서 실행
- 이 사이트에서는 사용자가 딥러닝 모델을 구성하고 여러 가지 매개 변수를 조정하면서 실험할 수 있는 기능을 제공한다.

구글의 플레이그라운드

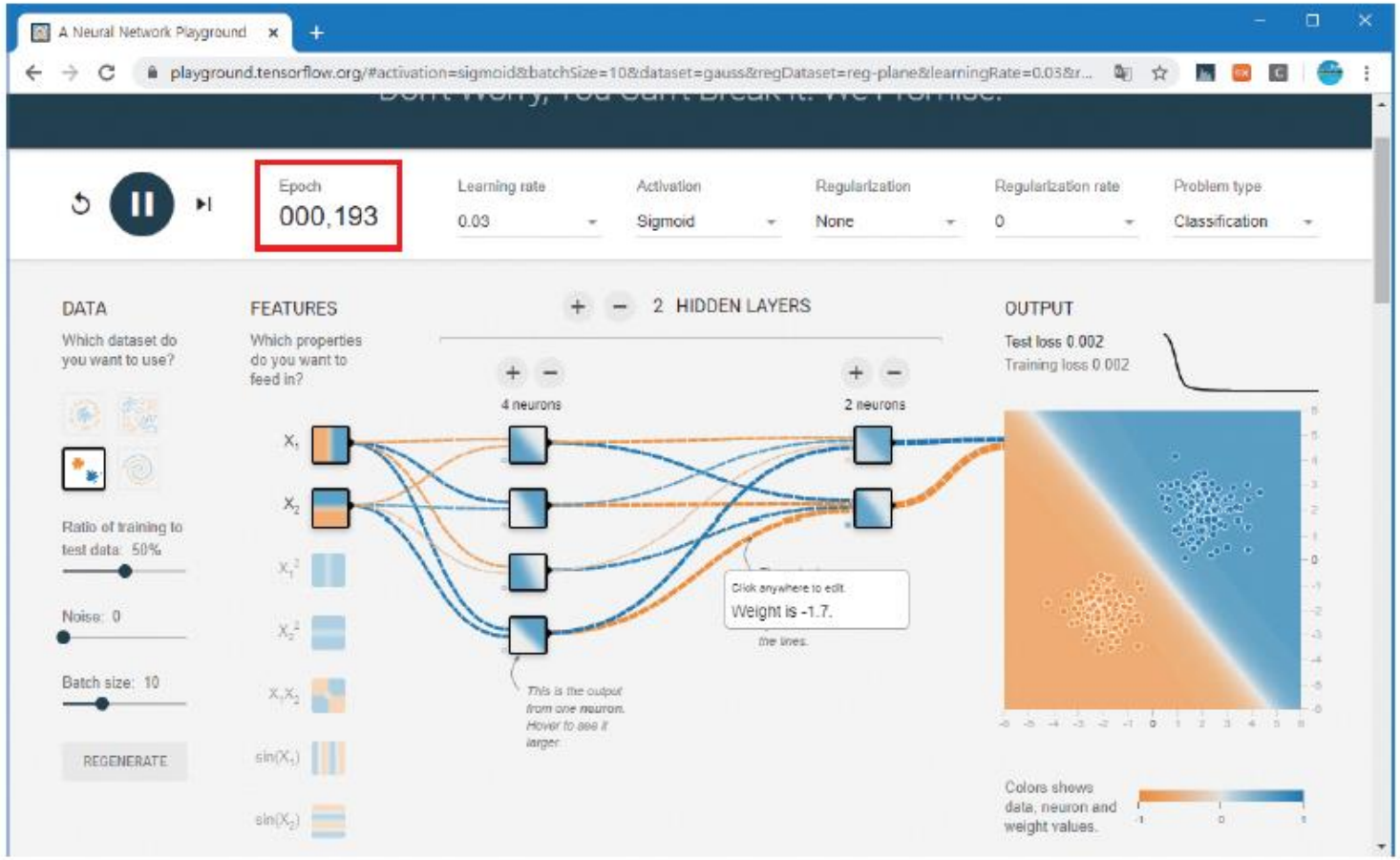

A Neural Network Playground
학습 시작 버튼
Epoch
000,000
Learning rate
0.03
Activation
Sigmoid
Regularization
None
Regularization rate
0
Problem type
Classification
입력 데이터 세트
DATA
Which dataset do you want to use?
Which properties do you want to feed in?
2 HIDDEN LAYERS
4 neurons
2 neurons
OUTPUT
Test loss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

선형 분리 가능한 입력 데이터

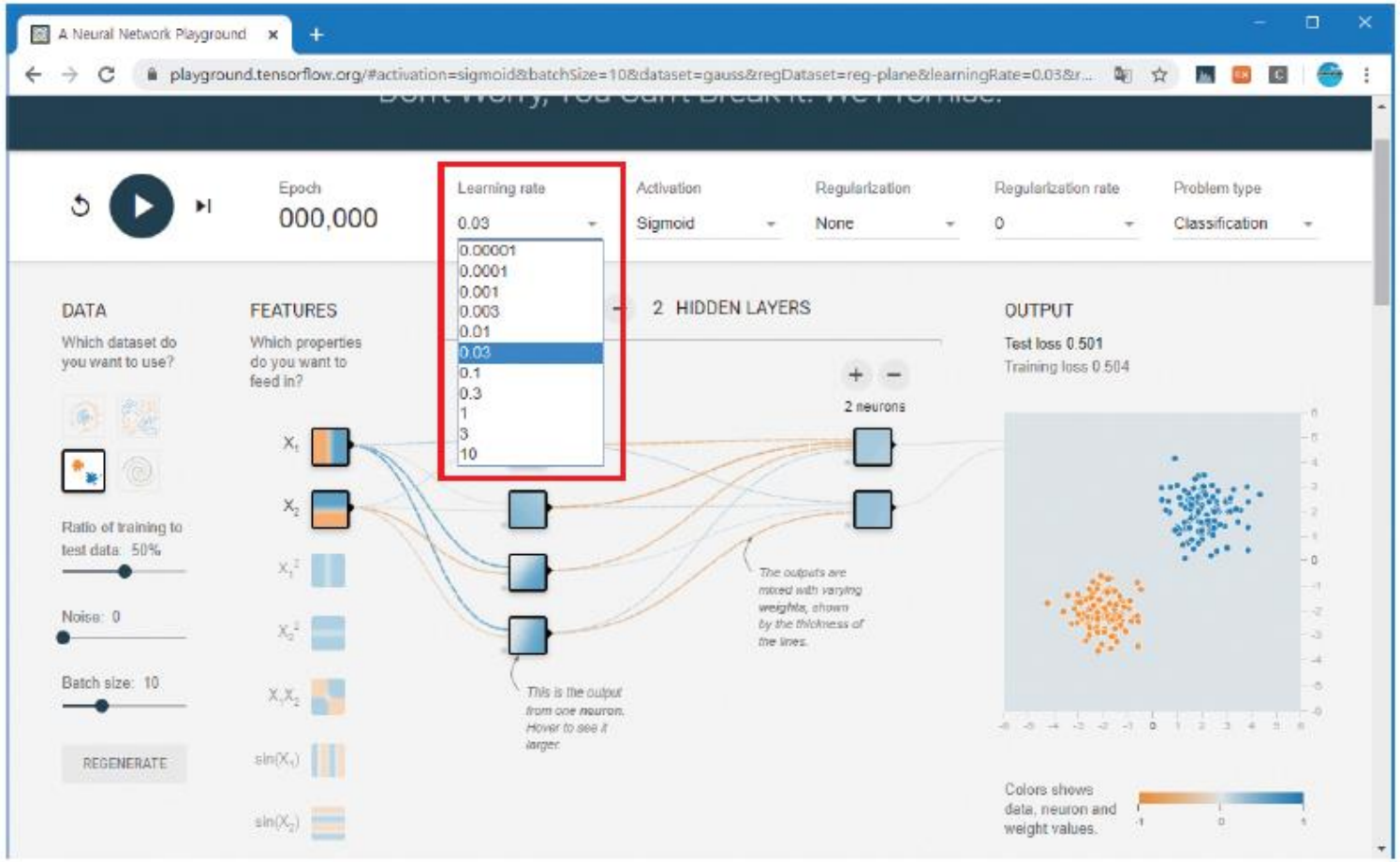


$$w_1x_1+w_2x_2+b=0$$

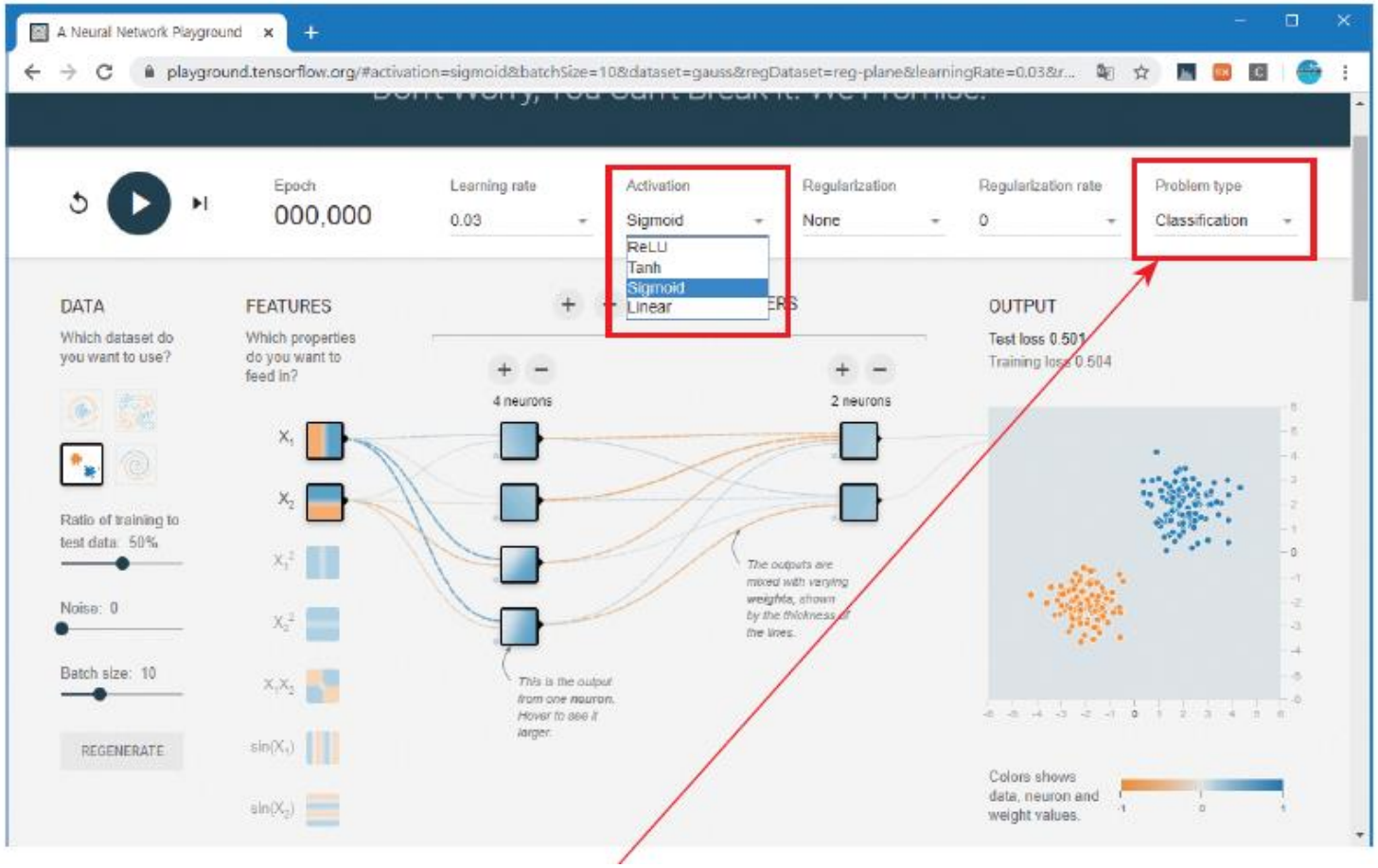
에포크


A Neural Network Playground
playground.tensorflow.org/#activation=sigmoid&batchSize=10&dataset=gauss&regDataset=reg-plane&learningRate=0.03&r...
Epoch
000,193
Learning rate
0.03
Activation
Sigmoid
Regularization
None
Regularization rate
0
Problem type
Classification
DATA
Which dataset do you want to use?
Ratio of training to test data: 50%
Noise: 0
Batch size: 10
REGENERATE
FEATURES
Which properties do you want to feed in?
2 HIDDEN LAYERS
4 neurons
2 neurons
OUTPUT
Test loss 0.002
Training loss 0.002
Click anywhere to edit.
Weight is -1.7.
the lines.
This is the output from one neuron. Hover to see it larger.
Colors shows data, neuron and weight values.

학습률

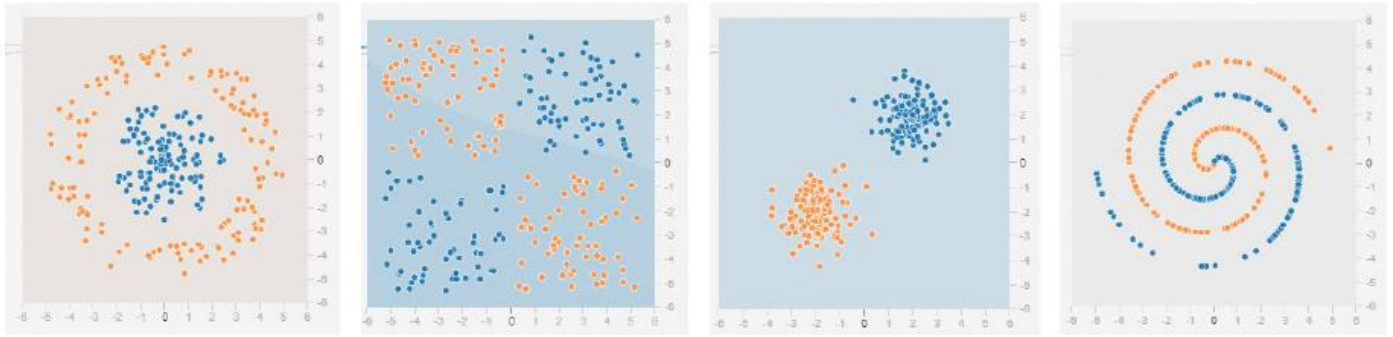

A Neural Network Playground
playground.tensorflow.org/#activation=sigmoid&batchSize=10&dataset=gauss&regDataset=reg-plane&learningRate=0.03&r...
Epoch
000,000
Learning rate
0.03
0.00001
0.0001
0.001
0.003
0.01
0.03
0.1
0.3
1
3
10
Activation
Sigmoid
Regularization
None
Regularization rate
0
Problem type
Classification
DATA
Which dataset do you want to use?
Ratio of training to test data: 50%
Noise: 0
Batch size: 10
REGENERATE
FEATURES
Which properties do you want to feed in?
2 HIDDEN LAYERS
2 neurons
The outputs are mixed with varying weights, shown by the thickness of the lines.
This is the output from one neuron. Hover to see it larger.
OUTPUT
Test loss 0.501
Training loss 0.504
Colors shows data, neuron and weight values.

활성화 함수 선택

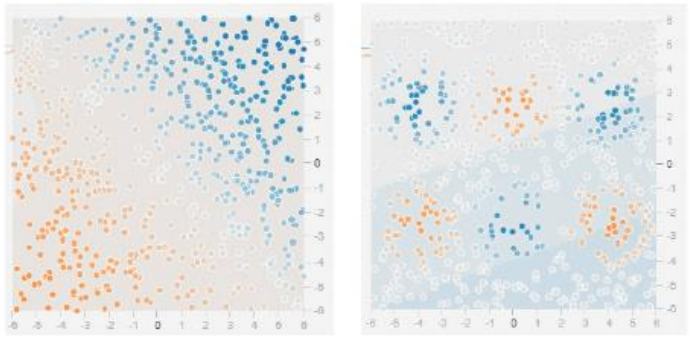

A Neural Network Playground
playground.tensorflow.org/#activation=sigmoid&batchSize=10&dataset=gauss&regDataset=reg-plane&learningRate=0.03&r...
Epoch
000,000
Learning rate
0.03
Activation
Sigmoid
ReLU
Tanh
Sigmoid
Linear
Regularization
None
Regularization rate
0
Problem type
Classification
DATA
Which dataset do you want to use?
Ratio of training to test data: 50%
Noise: 0
Batch size: 10
REGENERATE
FEATURES
Which properties do you want to feed in?
4 neurons
2 neurons
OUTPUT
Test loss 0.501
Training loss 0.504
The outputs are mixed with varying weights, shown by the thickness of the lines.
This is the output from one neuron. Hover to see it larger.
Colors shows data, neuron and weight values.

문제 유형

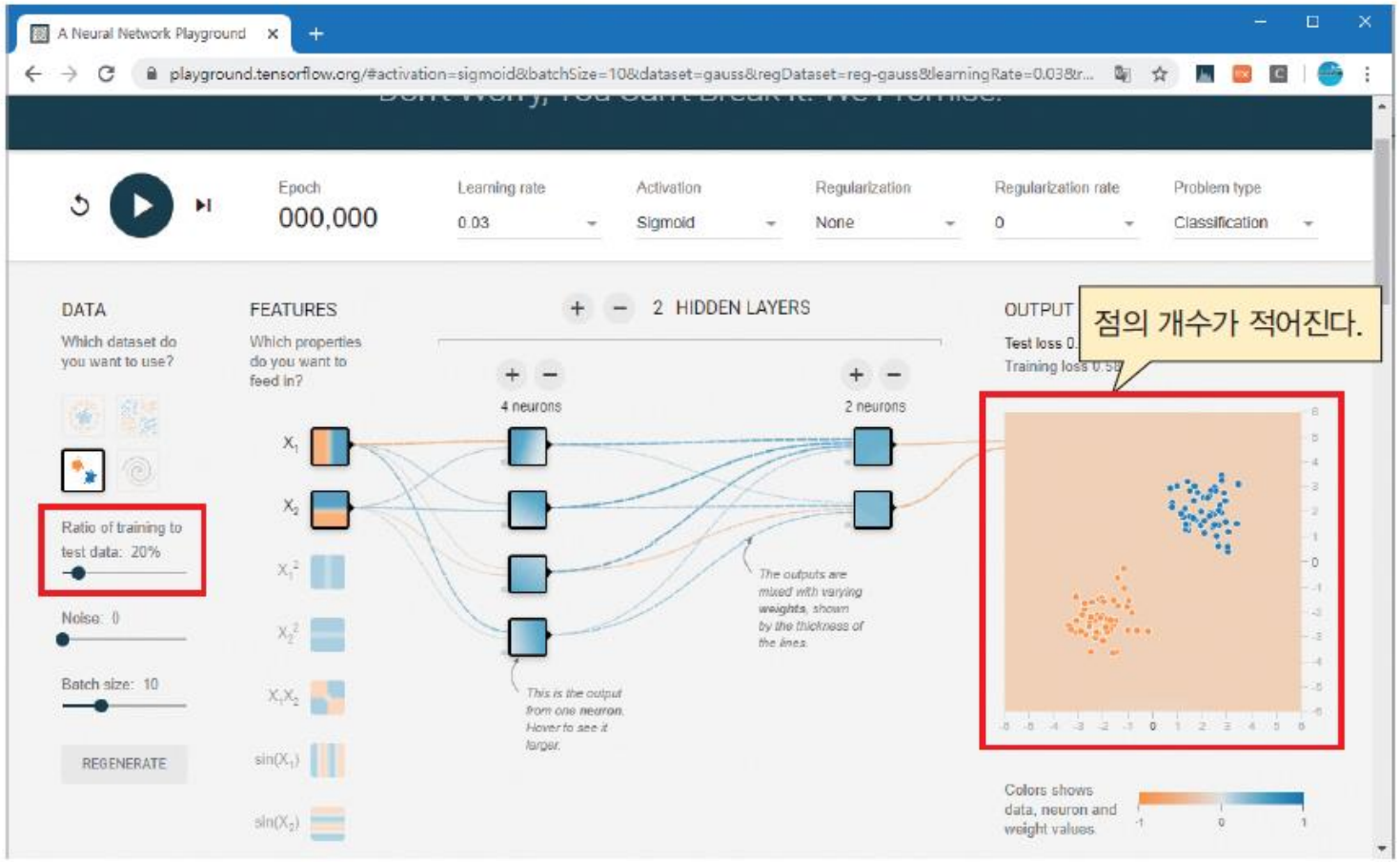
- 분류 문제



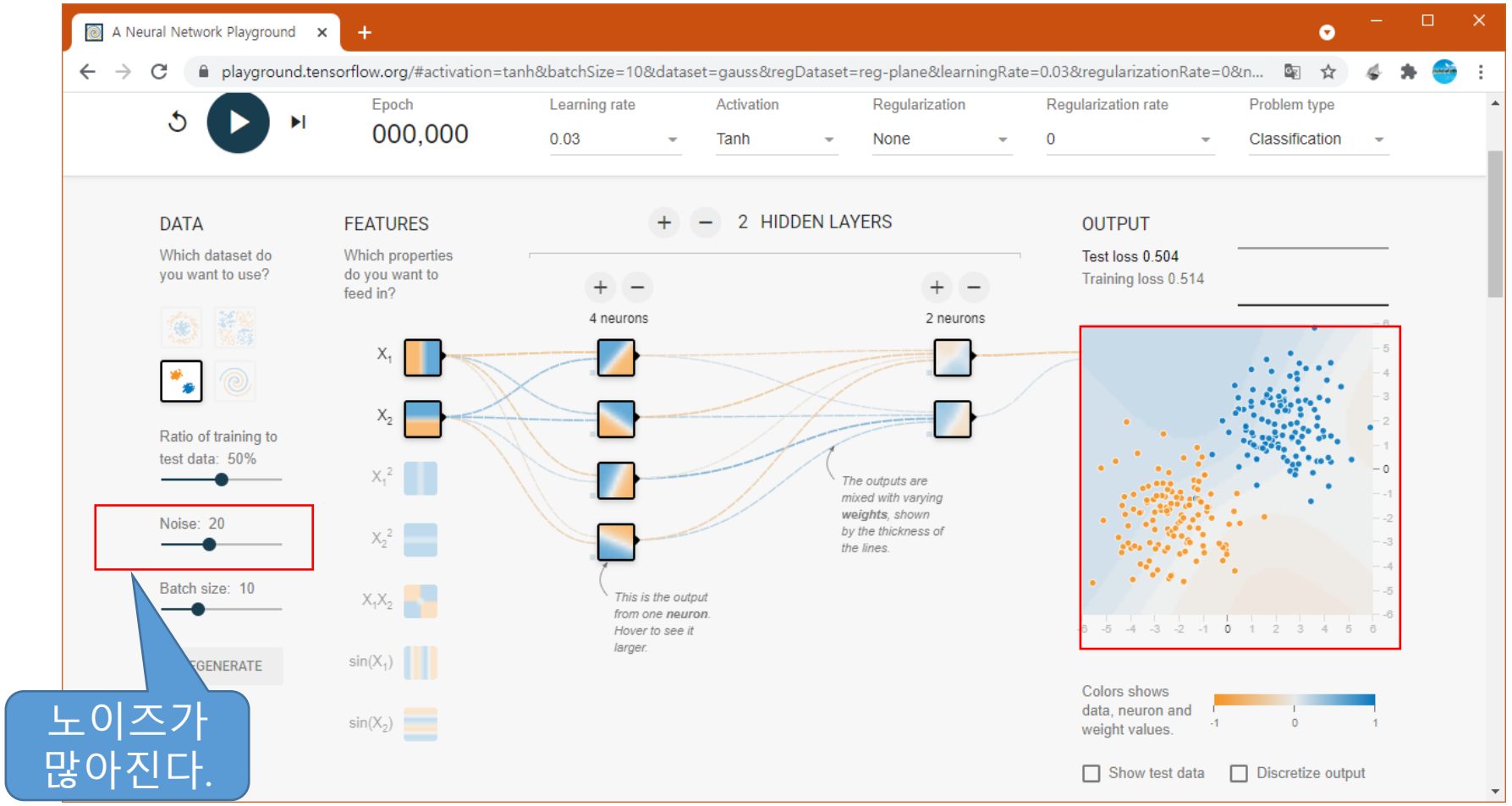
- 회귀 문제



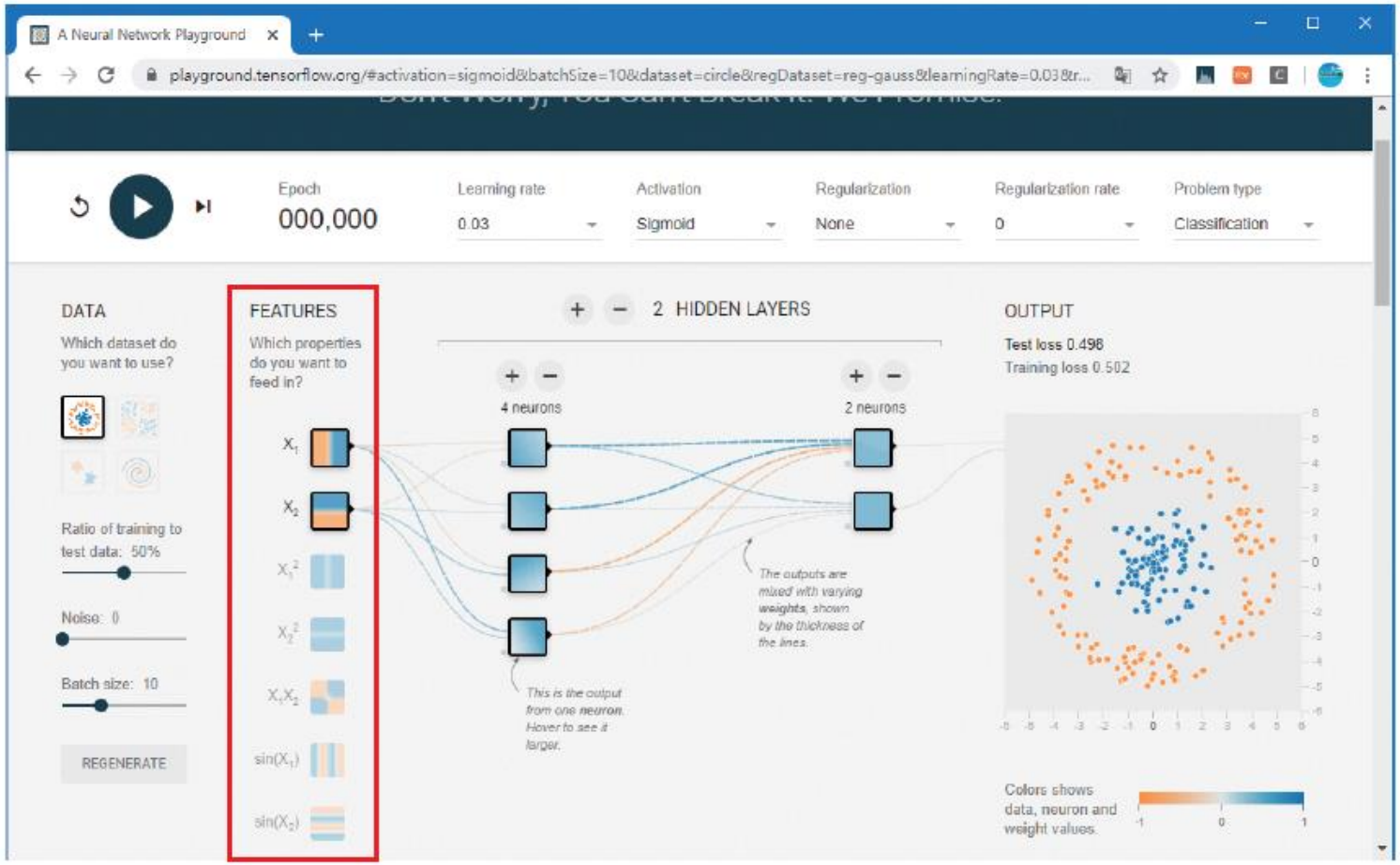
학습 데이터와 테스트 데이터의 비율



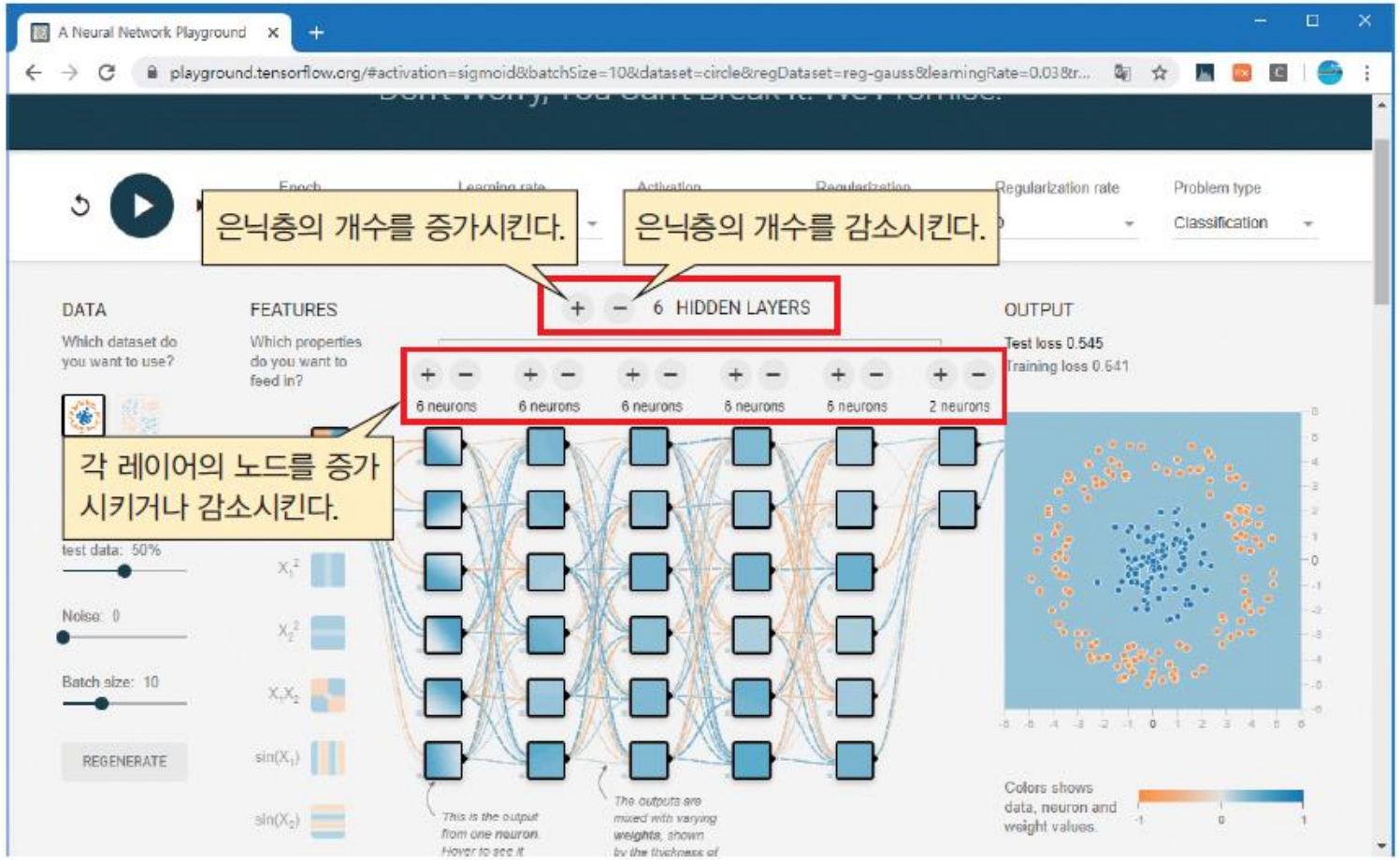
노이즈 비율


A Neural Network Playground
playground.tensorflow.org/#activation=tanh&batchSize=10&dataset=gauss&regDataset=reg-plane&learningRate=0.03&regularizationRate=0&n...
Epoch
000,000
Learning rate
0.03
Activation
Tanh
Regularization
None
Regularization rate
0
Problem type
Classification
DATA
Which dataset do you want to use?
Ratio of training to test data: 50%
Noise: 20
Batch size: 10
FEATURES
Which properties do you want to feed in?
2 HIDDEN LAYERS
4 neurons
2 neurons
The outputs are mixed with varying weights, shown by the thickness of the lines.
This is the output from one neuron. Hover to see it larger.
OUTPUT
Test loss 0.504
Training loss 0.514
Colors shows data, neuron and weight values.
Show test data
Discretize output
노이즈가 많아진다.

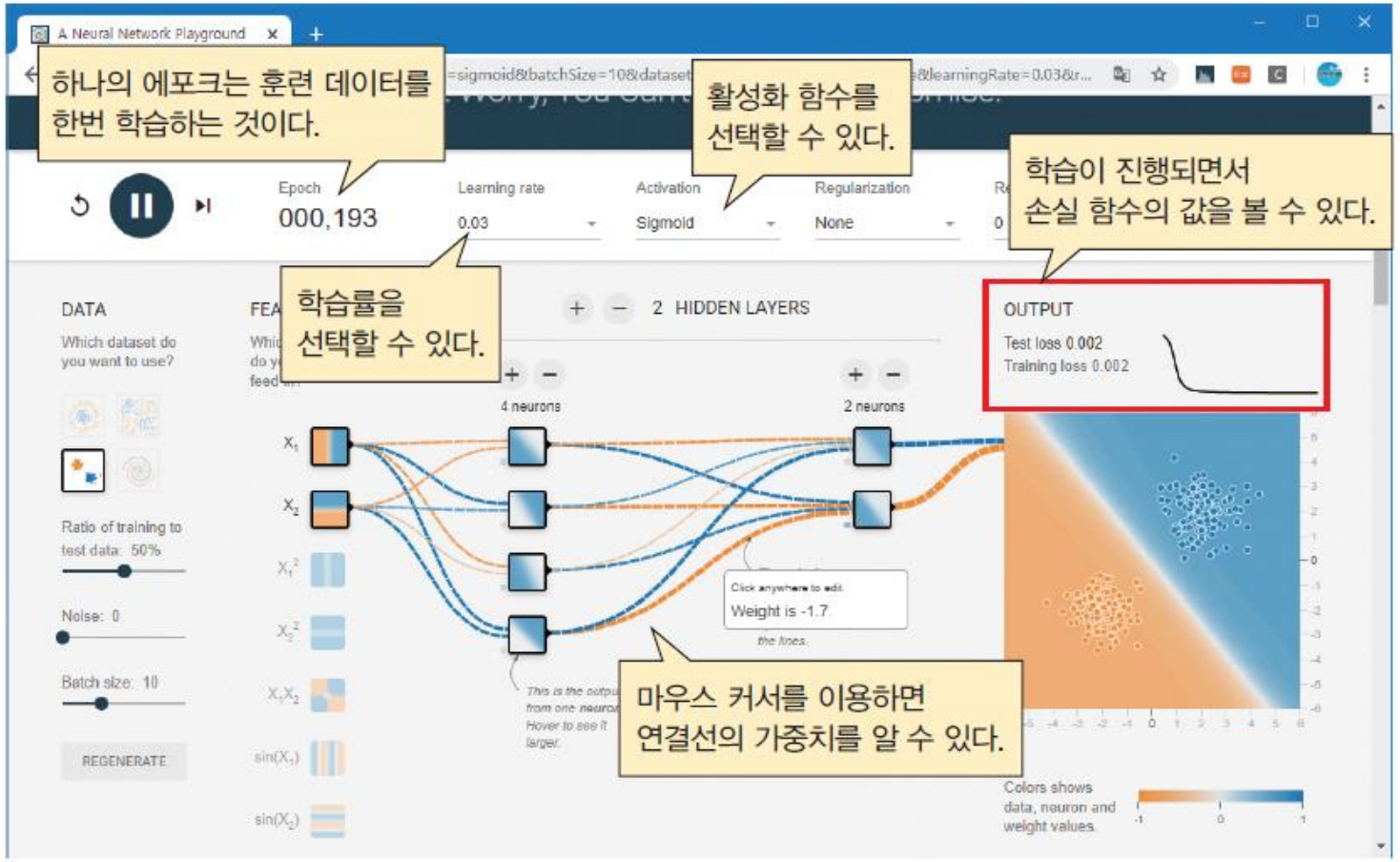
입력 특징 선택


A Neural Network Playground
playground.tensorflow.org/#activation=sigmoid&batchSize=10&dataset=circle&regDataset=reg-gauss&learningRate=0.03&r...
Epoch
000,000
Learning rate
0.03
Activation
Sigmoid
Regularization
None
Regularization rate
0
Problem type
Classification
DATA
Which dataset do you want to use?
Ratio of training to test data: 50%
Noise: 0
Batch size: 10
REGENERATE
FEATURES
Which properties do you want to feed in?
2 HIDDEN LAYERS
4 neurons
2 neurons
This is the output from one neuron. Hover to see it larger.
The outputs are mixed with varying weights, shown by the thickness of the lines.
OUTPUT
Test loss 0.498
Training loss 0.502
Colors shows data, neuron and weight values.

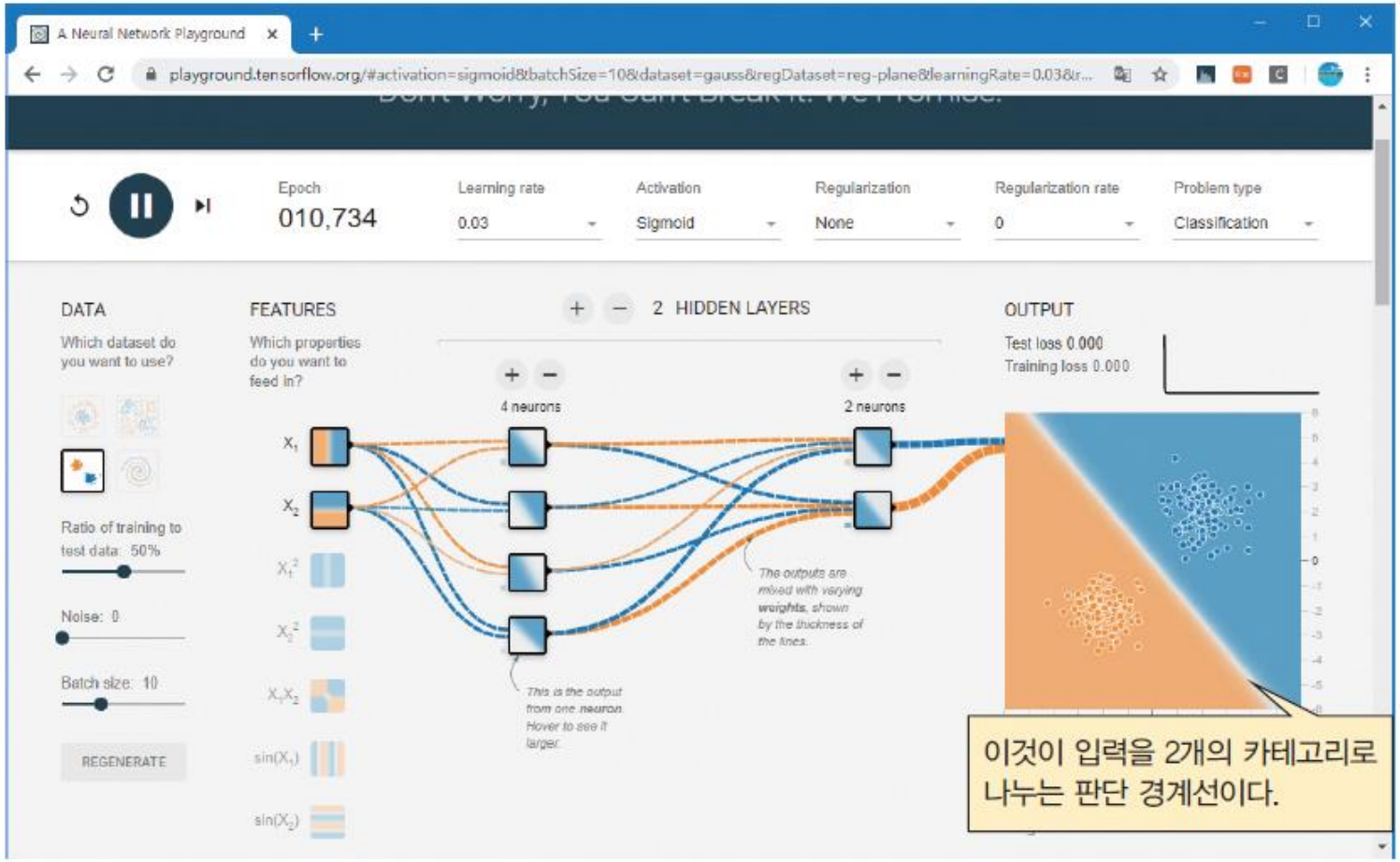
은닉층 추가하기


A Neural Network Playground
playground.tensorflow.org/#activation=sigmoid&batchSize=10&dataset=circle&regDataset=reg-gauss&learningRate=0.03&r...
은닉층의 개수를 증가시킨다.
은닉층의 개수를 감소시킨다.
Regularization rate
Problem type
Classification
DATA
Which dataset do you want to use?
FEATURES
Which properties do you want to feed in?
6 HIDDEN LAYERS
OUTPUT
Test loss 0.545
Training loss 0.641
6 neurons
6 neurons
6 neurons
6 neurons
6 neurons
2 neurons
각 레이어의 노드를 증가시키거나 감소시킨다.
test data: 50%
Noise: 0
Batch size: 10
REGENERATE
Colors shows data, neuron and weight values.

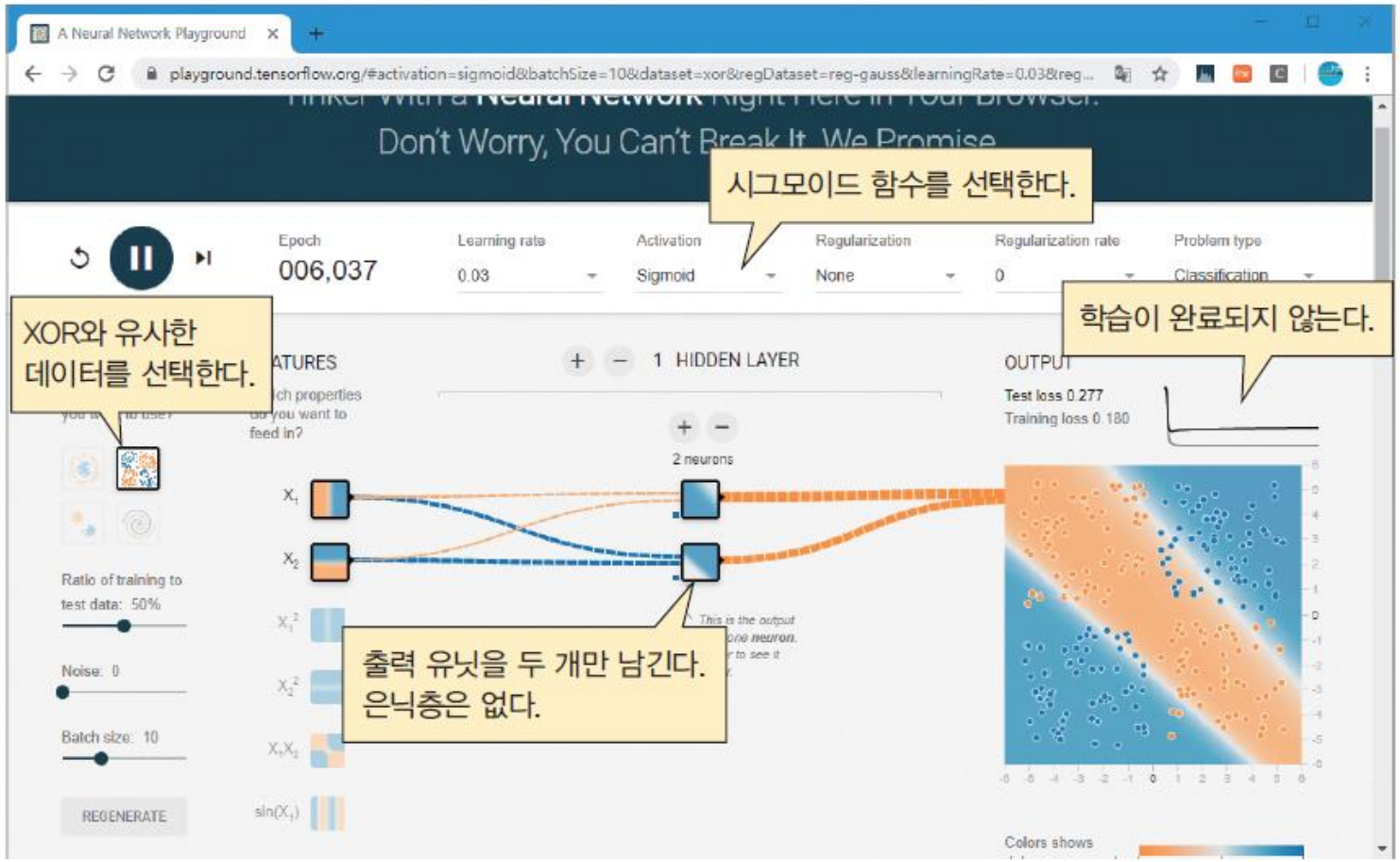
학습 시작


A Neural Network Playground
하나의 에포크는 훈련 데이터를 한번 학습하는 것이다.
활성화 함수를 선택할 수 있다.
학습이 진행되면서 손실 함수의 값을 볼 수 있다.
학습률을 선택할 수 있다.
Epoch
000,193
Learning rate
0.03
Activation
Sigmoid
Regularization
None
DATA
Which dataset do you want to use?
Ratio of training to test data: 50%
Noise: 0
Batch size: 10
REGENERATE
2 HIDDEN LAYERS
4 neurons
2 neurons
OUTPUT
Test loss 0.002
Training loss 0.002
Weight is -1.7
마우스 커서를 이용하면 연결선의 가중치를 알 수 있다.
Colors shows data, neuron and weight values.

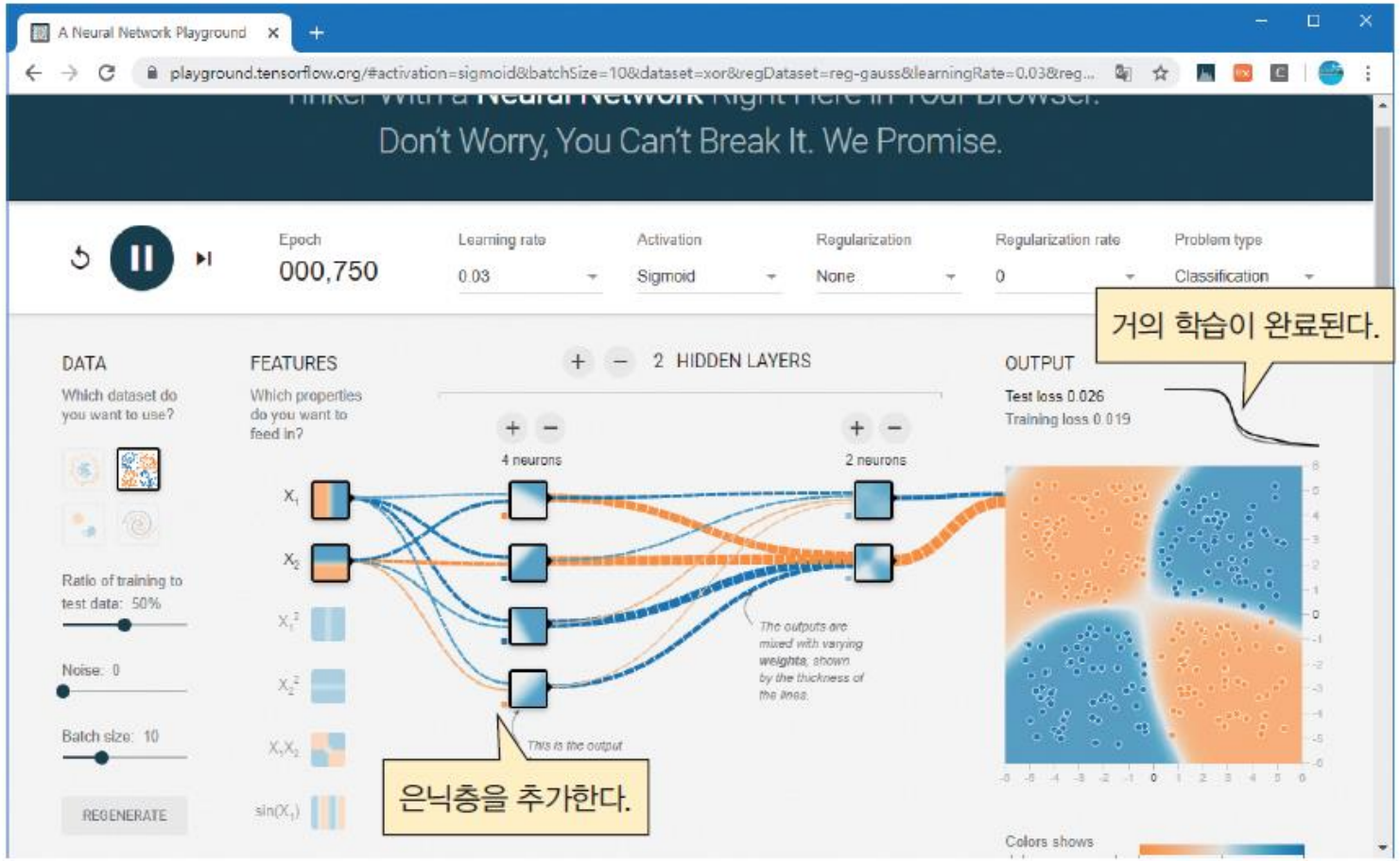
학습 완료


A Neural Network Playground
playground.tensorflow.org/#activation=sigmoid&batchSize=10&dataset=gauss&regDataset=reg-plane&learningRate=0.03&r...
Epoch
010,734
Learning rate
0.03
Activation
Sigmoid
Regularization
None
Regularization rate
0
Problem type
Classification
DATA
Which dataset do you want to use?
Ratio of training to test data: 50%
Noise: 0
Batch size: 10
REGENERATE
FEATURES
Which properties do you want to feed in?
2 HIDDEN LAYERS
4 neurons
2 neurons
The outputs are mixed with varying weights, shown by the thickness of the lines.
This is the output from one neuron. Hover to see it larger.
OUTPUT
Test loss 0.000
Training loss 0.000
이것이 입력을 2개의 카테고리로 나누는 판단 경계선이다.

은닉층 없이 분류 실습


A Neural Network Playground
playground.tensorflow.org/#activation=sigmoid&batchSize=10&dataset=xor&regDataset=reg-gauss&learningRate=0.03&reg...
Don't Worry, You Can't Break It. We Promise.
시그모이드 함수를 선택한다.
학습이 완료되지 않는다.
XOR와 유사한 데이터를 선택한다.
출력 유닛을 두 개만 남긴다.
은닉층은 없다.
Epoch
006,037
Learning rate
0.03
Activation
Sigmoid
Regularization
None
Regularization rate
0
Problem type
Classification
1 HIDDEN LAYER
2 neurons
OUTPUT
Test loss 0.277
Training loss 0.180
Ratio of training to test data: 50%
Noise: 0
Batch size: 10
REGENERATE
Colors shows

은닉층을 추가한 실습

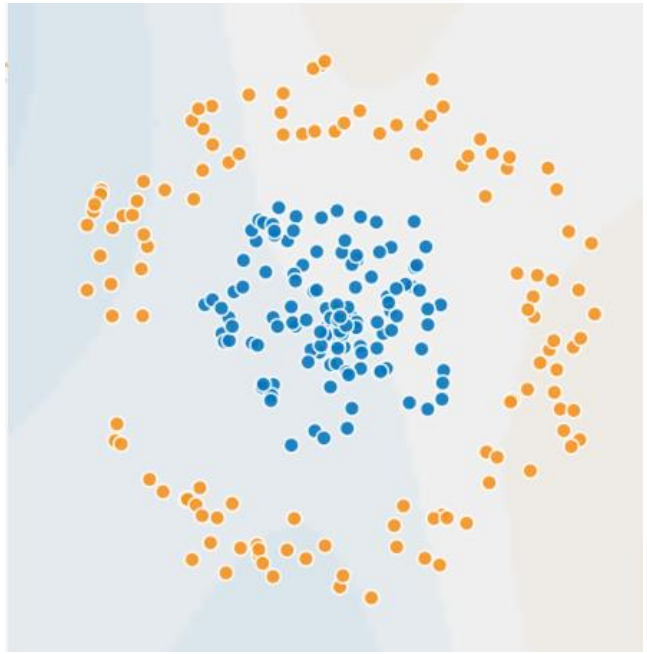

A Neural Network Playground
playground.tensorflow.org/#activation=sigmoid&batchSize=10&dataset=xor&regDataset=reg-gauss&learningRate=0.03&reg...
Don't Worry, You Can't Break It. We Promise.
Epoch
000,750
Learning rate
0.03
Activation
Sigmoid
Regularization
None
Regularization rate
0
Problem type
Classification
거의 학습이 완료된다.
DATA
Which dataset do you want to use?
Ratio of training to test data: 50%
Noise: 0
Batch size: 10
REGENERATE
FEATURES
Which properties do you want to feed in?
2 HIDDEN LAYERS
4 neurons
2 neurons
The outputs are mixed with varying weights, shown by the thickness of the lines.
This is the output
은닉층을 추가한다.
OUTPUT
Test loss 0.026
Training loss 0.019
Colors shows

과제 7

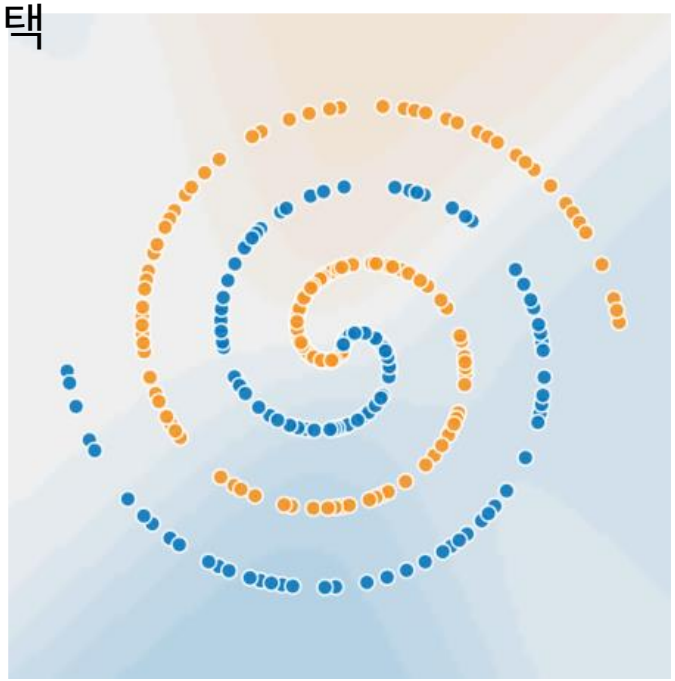
- **Google play ground (problem classification)**

에서 learning rate, hidden layer 수, neuron 수, activation, features

ratio of training data 를 조절 하여 아래 dataset 을 분류한 결과 image를 upload 하시오.



선택



수고하셨습니다

jhmin@inhatc.ac.kr