AI 프로그래밍

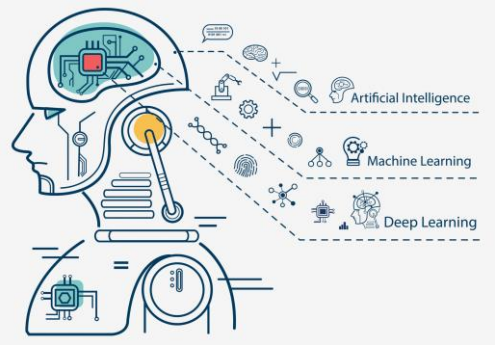
- REVIEW 0411

인하 공전 컴퓨터 정보과

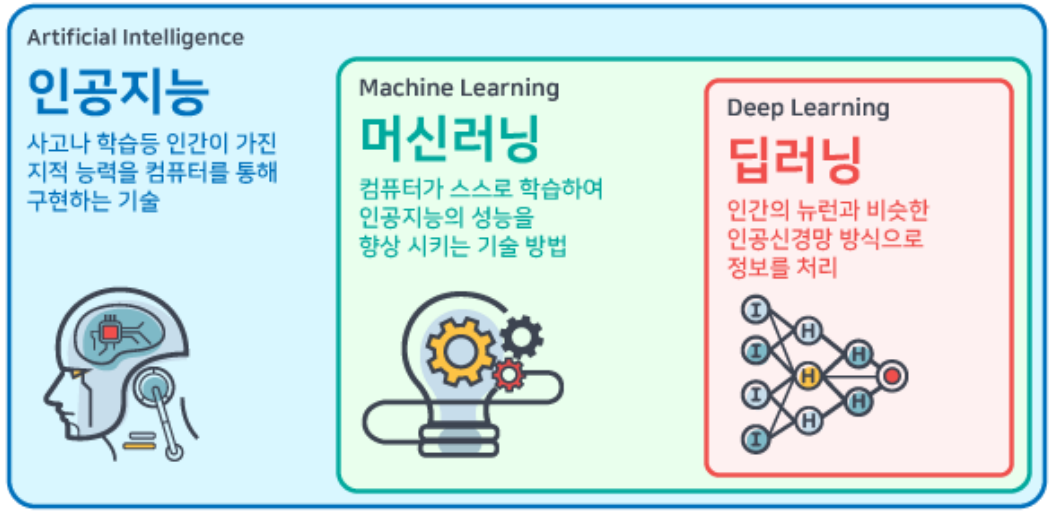
민 정혜

인공 지능 이란

- **인공 지능** (Artificial Intelligence)
  - **인간**이 가진 **지적 능력**을 **컴퓨터**를 통해 구현 하는 기술
- **머신 러닝** (Machine Learning)
  - 데이터와 **알고리즘**을 통해 컴퓨터를 **학습** 시켜 인공 지능의 성능을 향상 시키는 기술
- **딥 러닝** (Deep Learning)
  - **신경망 (Neural Network)** 을 이용한 머신 러닝 방법.
  - 여러 층으로 이루어진 신경망을 사용.



Machine Learning: A Primer to Laboratory Applications (thermofisher.com)



https://m.blog.naver.com/pwj6971/221614497987

선형 회귀 예제 실습

다음은 집의 면적 당 가격을 정리한 표이다.

X 입력	면적	5	7	12	13	19
Y 출력	가격	12	19	28	37	48

자료실의 regression1.ipnb 프로그램을 이용하여
1) 선형회귀로 분석 하여 직선의 방정식 y=wx+b를 구하여라

선형 회귀 예제 실습

code

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



X = np.array([5.0, 7.0, 12.0, 13.0, 19.0])
y = np.array([12.0,19.0,28.0,37.0,48.0])

w = 0       # 기울기
b = 0       # 절편

lrate = 0.003  # 학습률
epochs = 500  # 반복 횟수

n = float(len(X)) # 입력 데이터의 개수

# 경사 하강법

for i in range(epochs):
    y_pred = w*X + b                                  # 선형 회귀 예측값
    dw = (2/n) * sum(X * (y_pred-y))
    db = (2/n) * sum(y_pred-y)
    w = w - lrate * dw                    # 기울기 수정
    b = b - lrate * db                    # 절편 수정
    if (i%50==0):
      print('iteration %3d:  w %3.2f b %3.2f '%(i,w,b))


# 기울기와 절편을 출력한다.
print ('########## final w %3.2f b %3.2f '%(w, b))
```

실행 결과

```
iteration   0:  w 2.31 b 0.17
iteration  50:  w 2.56 b 0.19
iteration 100:  w 2.56 b 0.19
iteration 150:  w 2.56 b 0.18
iteration 200:  w 2.56 b 0.18
iteration 250:  w 2.56 b 0.18
iteration 300:  w 2.56 b 0.18
iteration 350:  w 2.56 b 0.17
iteration 400:  w 2.56 b 0.17
iteration 450:  w 2.56 b 0.17
########## final w 2.56 b 0.17
```

반복 횟수

면적이 10일때 예측 값은?

W*x+b=2.56*10+0.17

선형 회귀 예제 실습

다음은 CPU 속도와 프로그래밍 수행 시간을 정리한 표이다.

CPU	30	50	80	90	120
수행 시간	70	140	145	170	260

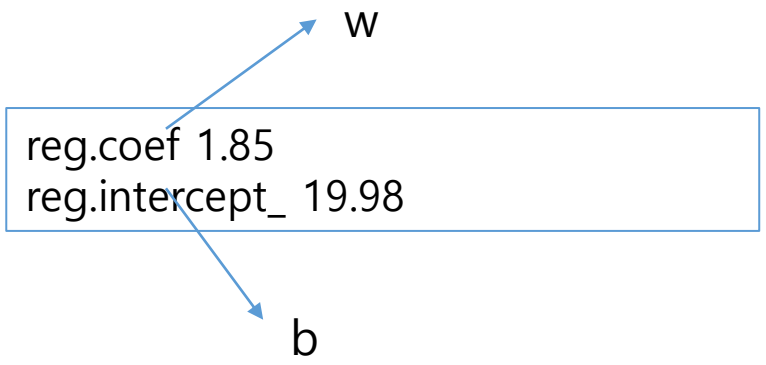
자료실의 regression2.ipnb 프로그램을 이용하여
1) 선형회귀로 분석 하여 직선의 방정식 y=wx+b를 구하여라
1) CPU 속도가 100일때의 수행 시간을 예측 하시오.

```
# 데이터는 파이썬의 리스트로 만들어도 되고 아니면 넘파이의 배열로 만들어도 됨
X = [[30], [50], [80], [90], [120]]   # 반드시 2차원으로 만들어야 함
y = [70, 140, 145,170,260]
```

w,b, CPU 속도가 100일때의 수행시간 예측 값을 예측 값을 채팅으로 보내주세요.

선형 회귀 예제 실습

```
# 선형 회귀 모델을 생성한다.
reg = linear_model.LinearRegression()

# 데이터는 파이썬의 리스트로 만들어도 되고 아니면 넘파이의 배열로 만들어도 됨
X = [[30], [50], [80], [90], [120]]   # 반드시 2차원으로 만들어야 함
y = [70, 140, 145,170,260]

# 학습을 시킨다.
reg.fit(X, y)

print('reg.coef %3.2f'% (reg.coef_))     # 직선의 기울기
print('reg.intercept_ %3.2f'%(reg.intercept_))  # 직선의 y-절편
```

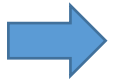


CPU속도가 100일때의 수행 시간 예측 값은?
w*100+b

선형 회귀 예제 실습

```
# 선형 회귀 모델을 생성한다.
reg = linear_model.LinearRegression()

# 데이터는 파이썬의 리스트로 만들어도 되고 아니면 넘파이의 배열로 만들어도 됨
X = np.array([30, 50, 80, 90, 120])
 # 반드시 2차원으로 만들어야 함
y = np.array([70, 140, 145,170,260]   )

# 학습을 시킨다.
reg.fit(X, y)

print('reg.coef %3.2f'% (reg.coef_))     # 직선의 기울기
print('reg.intercept_ %3.2f'%(reg.intercept_))  # 직선의 y-절편
```

```
ValueError: Expected 2D array, got 1D array instead: array=[ 30 50 80 90 120]. Reshape
your data either using array.reshape(-1, 1) if your data has a single feature or
array.reshape(1, -1) if it contains a single sample.
```



Error 를 고치려면 ? X 직접 수정 or numpy 함수 사용

오차 수정하기 : 경사 하강법

- Y=ax
- 오차가 가장 작은 지점은?

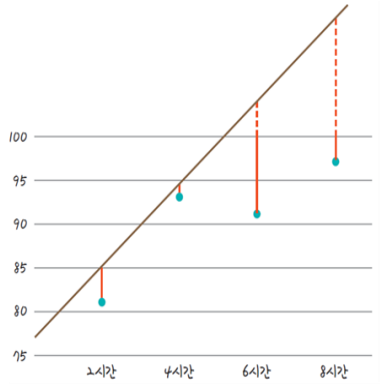


그림 3-7 기울기를 너무 크게 잡았을 때의 오차

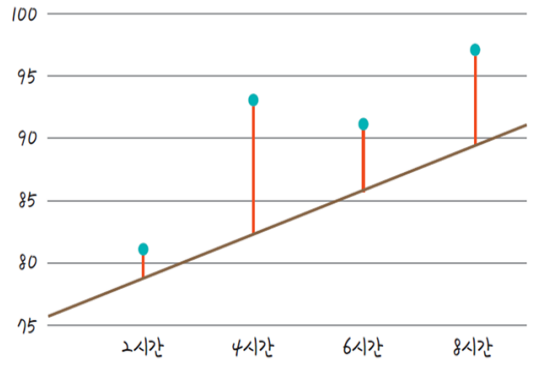
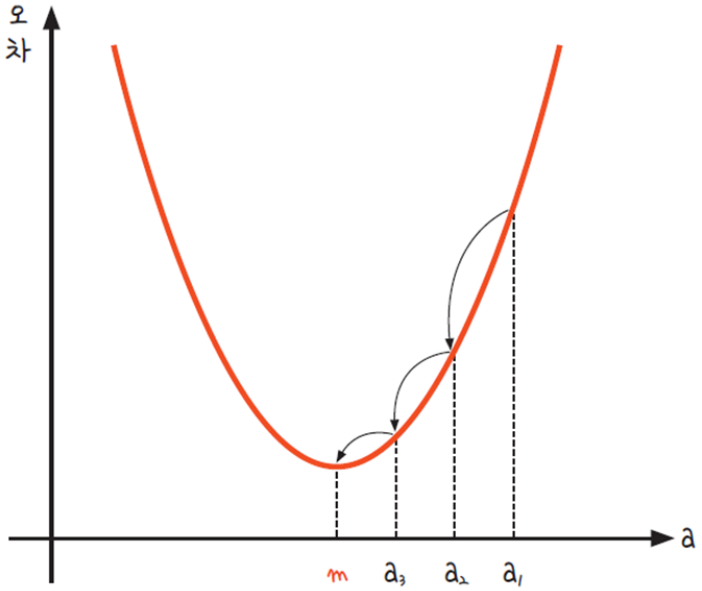


그림 3-8 기울기를 너무 작게 잡았을 때의 오차



- 컴퓨터를 이용해 m의 값을 구하려면 임의의 한 점(a_1)을 찍고 이 점을 m 에 가까운 쪽으로 점점 이동($a_1 \rightarrow a_2 \rightarrow a_3$)시키는 과정이 필요함
- 경사 하강법(gradient descent) :

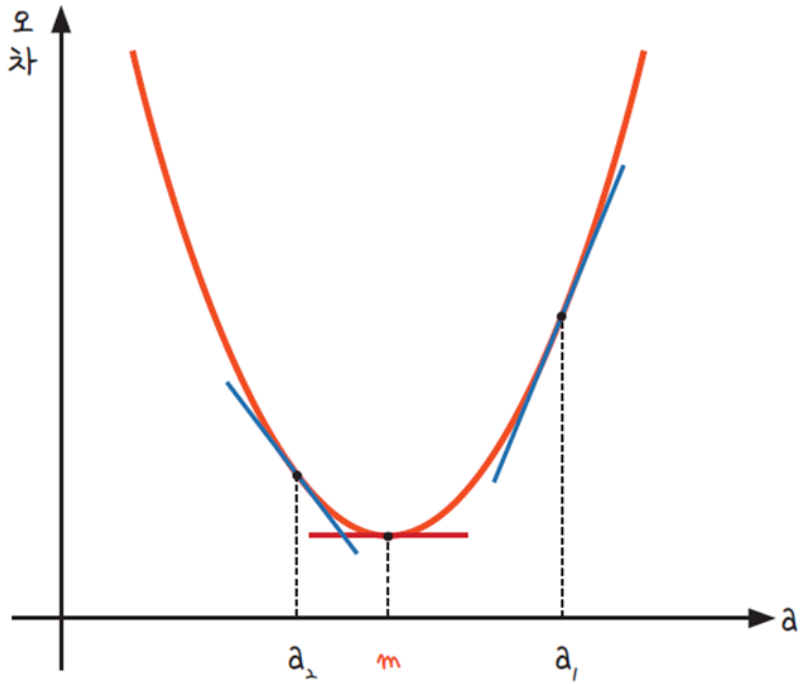
  그래프에서 오차를 비교하여 가장 작은 방향으로 이동시키는 방법이 있는데 바로 미분 기울기를 이용

경사 하강법

1 | a_1 에서 미분을 구함

2 | 구해진 기울기의 반대 방향(기울기가 +면 음의 방향, -면 양의 방향)으로 얼마간 이동시킨 a_2 에서 미분을 구함(그림 4-3 참조).

3 | 위에서 구한 미분 값이 0이 아니면 위 과정을 반복함



기울기=그래디언트=미분값

경사 하강법 실습

- 손실 함수 $y = (x-3)^2 + 10$
- 그래디언트(미분된 함수): y' = 2x-6
- 학습률 : 0.2

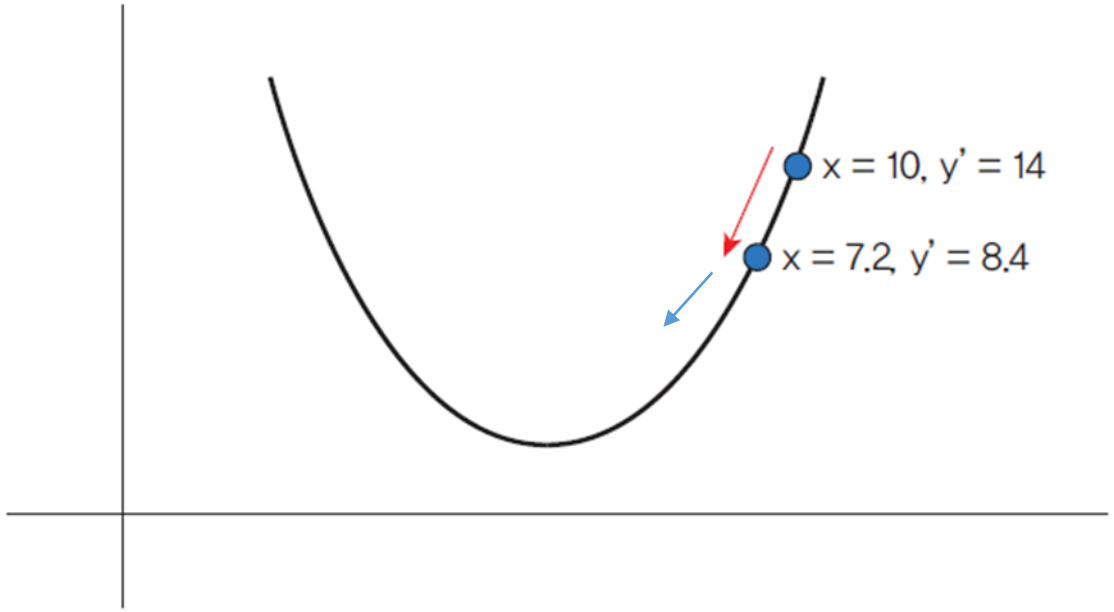


그림 6-12 그래디언트의 계산

- X=10, y'=14, 0.2*14=2.8,

Gradient의 반대 방향 => -2.8

10-2.8=7.2

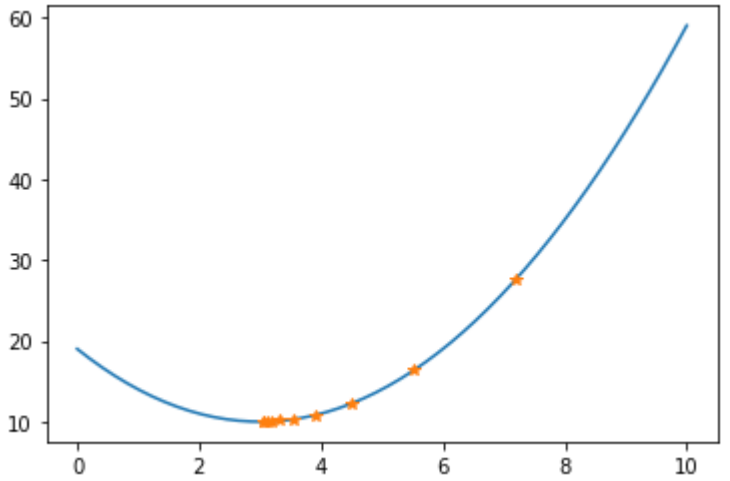
X초기값 10. 두번째 위치. 7.2

참고자료 : 딥러닝 express

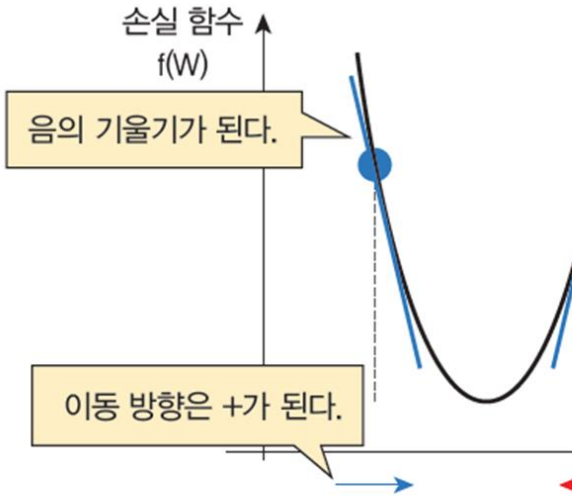
경사 하강법 실습

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

x = 10
learning_rate = 0.2
precision = 0.00001
max_iterations = 10

# 손실함수를 람다식으로 정의한다.
loss_func = lambda x: (x-3)**2 + 10

# 그래디언트를 람다식으로 정의한다. 손실함수의 1차 미분값이다.
gradient = lambda x: 2*x-6

list1 = []
list2 = []

# 그래디언트 강하법
for i in range(max_iterations):
    x = x - learning_rate * gradient(x)
    list1.append(x)
    list2.append(loss_func(x))
    print("i=%d X=%3.2f loss =%3.2f "%(i, x, loss_func(x)))

print("최소값 = ", x)

x1 = np.linspace(0.0, 10.0)
y1 = loss_func(x1)
fig, ax = plt.subplots()  # Create a figure containing a single axes.
ax.plot(x1,y1)  # Plot some data on the axes.
ax.plot(list1,list2, '*')  # Plot some data on the axes.
```

```
i=0 X=7.20 loss =27.64
i=1 X=5.52 loss =16.35
i=2 X=4.51 loss =12.29
i=3 X=3.91 loss =10.82
i=4 X=3.54 loss =10.30
i=5 X=3.33 loss =10.11
i=6 X=3.20 loss =10.04
i=7 X=3.12 loss =10.01
i=8 X=3.07 loss =10.00
i=9 X=3.04 loss =10.00
최소값 = 3.0423263232
```



경서 하강법 실습



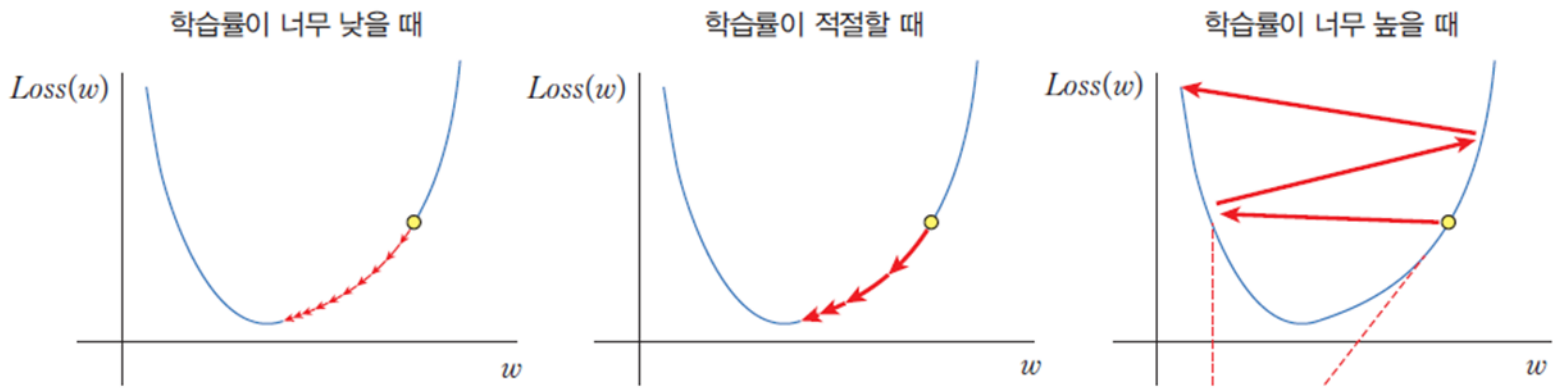
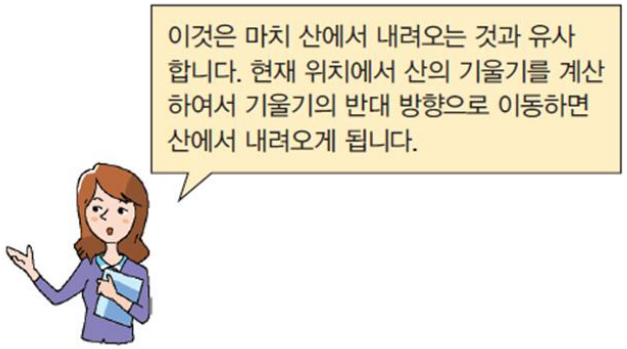


그림 4-9 학습률

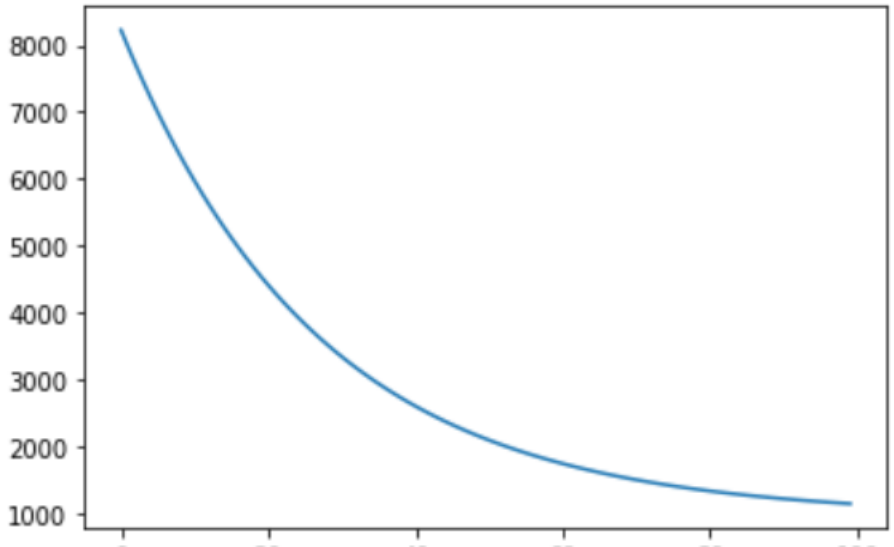


참고자료 : 딥러닝 express

경사 하강법 실습

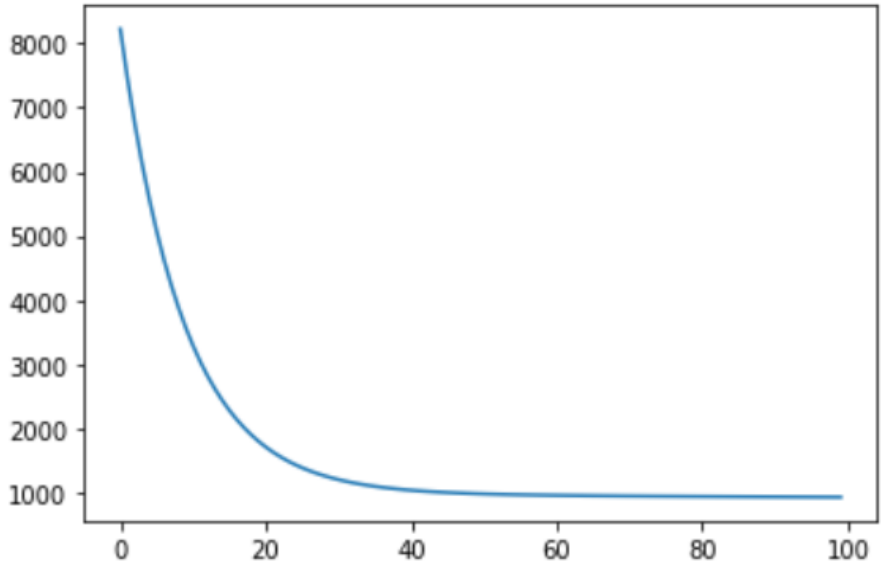
- 학습률=0.0003

```
epoch=0, 기울기=0.2784, 절편=0.0543 loss =8225.00000
epoch=10, 기울기=2.7931, 절편=0.5521 loss =5965.81863
epoch=20, 기울기=4.8772, 절편=0.9777 loss =4410.22195
epoch=30, 기울기=6.6042, 절편=1.3434 loss =3338.90852
epoch=40, 기울기=8.0349, 절편=1.6594 loss =2600.93165
epoch=50, 기울기=9.2197, 절편=1.9341 loss =2092.39373
epoch=60, 기울기=10.2006, 절편=2.1745 loss =1741.78132
epoch=70, 기울기=11.0123, 절편=2.3865 loss =1499.87125
epoch=80, 기울기=11.6836, 절편=2.5749 loss =1332.78285
epoch=90, 기울기=12.2385, 절편=2.7438 loss =1217.19585
[<matplotlib.lines.Line2D at 0x7f140432c8d0>]
```



- 학습률=0.0009

```
epoch=0, 기울기=0.8352, 절편=0.1629 loss =8225.00000
epoch=10, 기울기=7.0087, 절편=1.4265 loss =3288.43886
epoch=20, 기울기=10.4800, 절편=2.2366 loss =1709.21635
epoch=30, 기울기=12.4249, 절편=2.7902 loss =1201.43535
epoch=40, 기울기=13.5076, 절편=3.1987 loss =1035.60302
epoch=50, 기울기=14.1032, 절편=3.5249 loss =978.91795
epoch=60, 기울기=14.4237, 절편=3.8044 loss =957.08669
epoch=70, 기울기=14.5889, 절편=4.0572 loss =946.40002
epoch=80, 기울기=14.6663, 절편=4.2946 loss =939.29155
epoch=90, 기울기=14.6943, 절편=4.5230 loss =933.34650
[<matplotlib.lines.Line2D at 0x7f1404a82190>]
```



선형 회귀 예제 실습 – 당뇨병 예제

Bmi와 혈당간의 관계 예측

```
diabetes_X, diabetes_y = datasets.load_diabetes(return_X_y=True)

print('diabetes_X',diabetes_X.shape )
```

→ `diabetes_X (442, 10)`

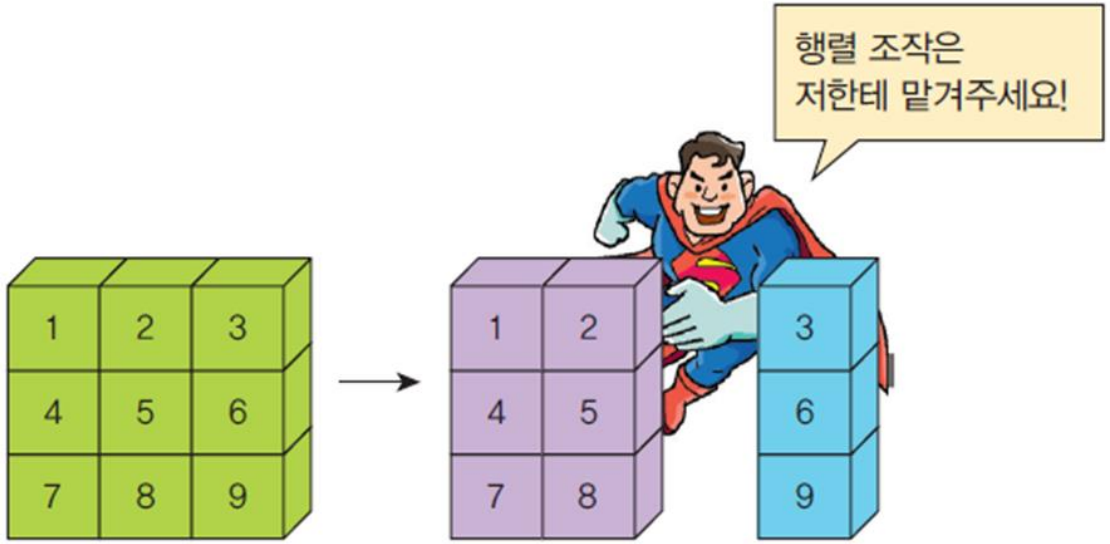
```
# 하나의 특징(BMI)만 추려내서 2차원 배열로 만든다. BMI 특징의 인덱
스가 2이다.
diabetes_X_new0 = diabetes_X[:, 2]
print('diabetes_X_new0',diabetes_X_new0.shape )
diabetes_X_new = diabetes_X_new0[:, np.newaxis]
print('diabetes_X_new',diabetes_X_new.shape )
```

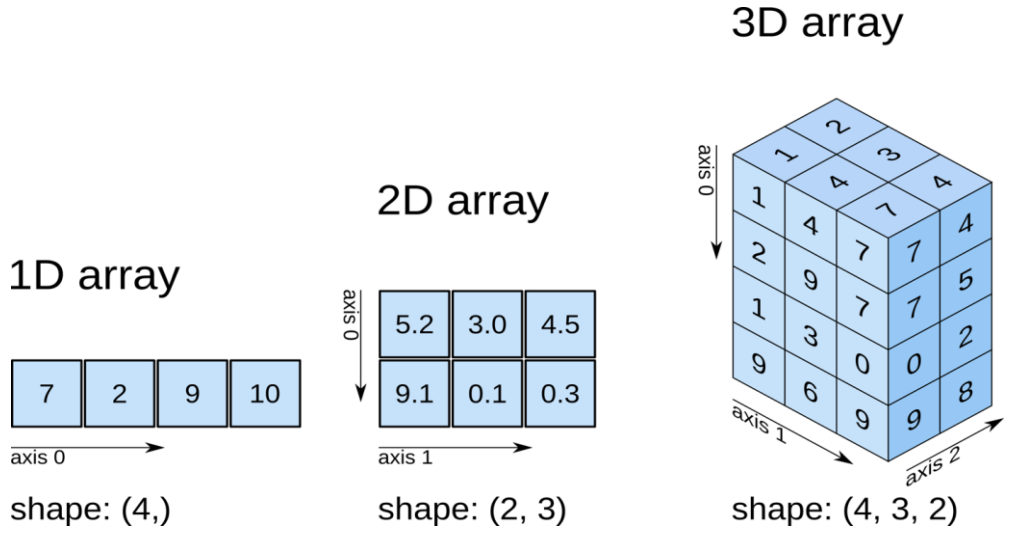
→ `diabetes_X_new0 (442,)`

→ `diabetes_X_new (442, 1)`

Numpy

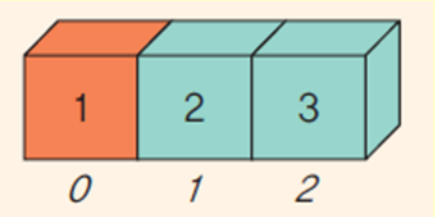
- 고성능의 수치 계산을 위한 라이브러리
- 넘파이 라이브러리에는 다차원 배열 데이터 구조가 포함되어 있다.
- 넘파이는 데이터 전처리 및 다양한 수학적인 행렬 연산을 수행하는 데 사용할 수 있다.
- **Import numpy as np**

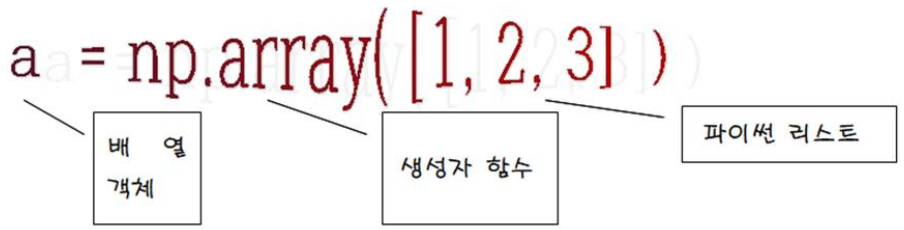




Numpy

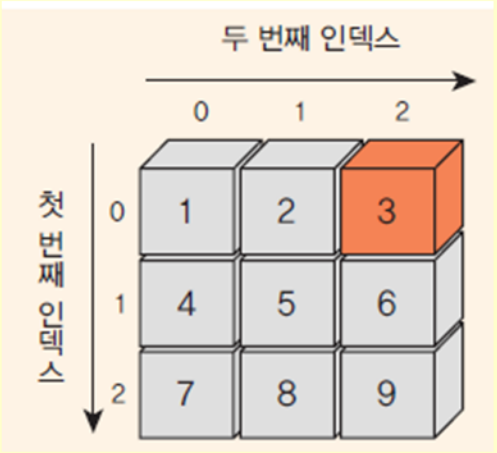
```
>>> import numpy as np

>>> a = np.array([1, 2, 3])
>>> a
array([1, 2, 3])

>>> a[0]
1
```





Numpy

```
>>> b = np.array([[1, 2, 3], [4, 5, 6], [7, 8, 9]])
>>> b
array([[1, 2, 3],
       [4, 5, 6],
       [7, 8, 9]])

>>> b[0][2]
3
```



Numpy

- 배열의 차원 및 항목 수는 형상(shape)에 의해 정의된다. 배열의 형상은 각 차원의 크기를 지정하는 정수의 튜플이다.
- 넘파이에서는 차원을 축(axes)이라고 한다

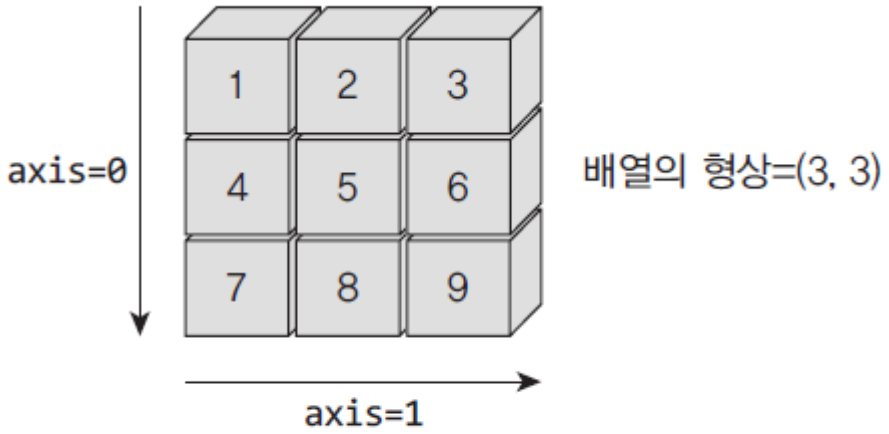
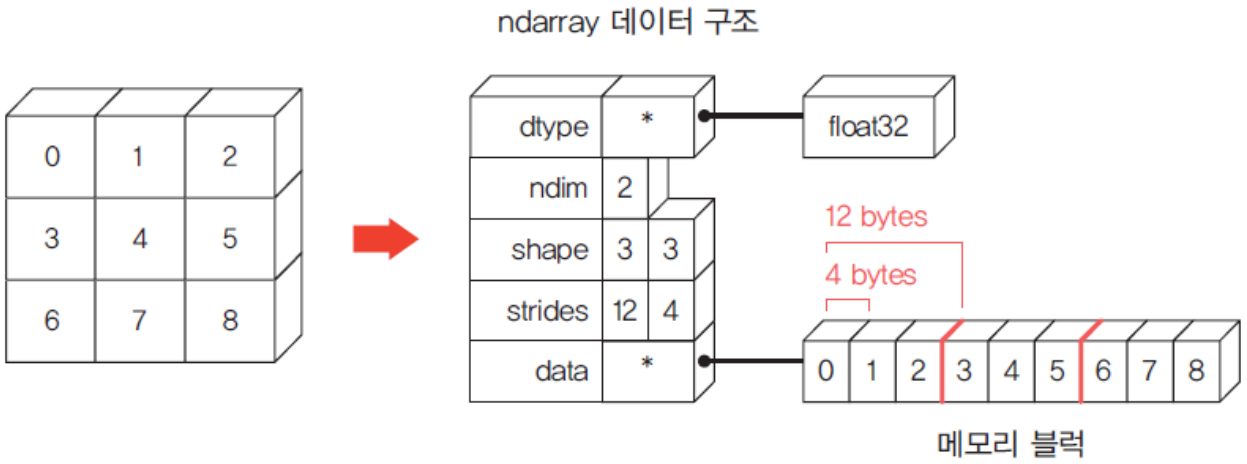


그림 2-4 넘파이 배열의 형상

Numpy- 배열의 속성

속성	설명
ndim	축의 개수. 2차원 배열이면 ndim은 20이다.
shape	배열의 형상. 형상은 정수 튜플로 나타낸다. 예를 들어서 n개의 행과 m개의 열이 있는 행렬의 경우, shape는 (n, m)이다.
size	배열 안에 있는 요소들의 총 개수
dtype	배열 요소의 자료형, numpy.int32, numpy.int16, numpy.float64가 그 예이다.
itemsize	배열을 이루는 요소의 크기로서 단위는 바이트이다. 예를 들어, float64은 itemsize가 80이다.
data	실제 데이터가 저장되는 메모리 블럭의 주소



Numpy

```
>>> import numpy as np
>>> a = np.array([[ 0, 1, 2],
          [ 3, 4, 5],
          [ 6, 7, 8]])
>>> a.shape                    # 배열의 형상
(3, 3)
>>> a.ndim                     # 배열의 차원 개수
2
```

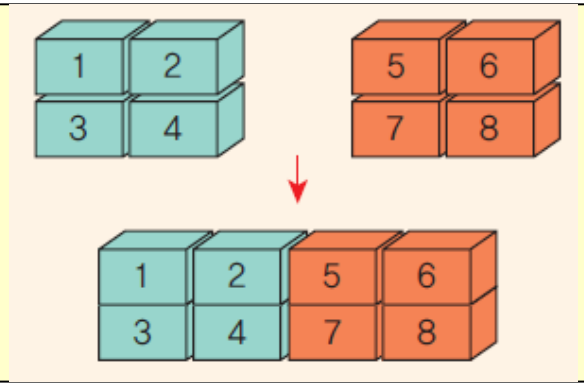
```
>>> np.zeros( (3, 4) )                    # (3, 4)는 배열의 형상(행의 개수, 열의 개수)
array([[ 0.,  0.,  0.,  0.],
       [ 0.,  0.,  0.,  0.],
       [ 0.,  0.,  0.,  0.]])
```

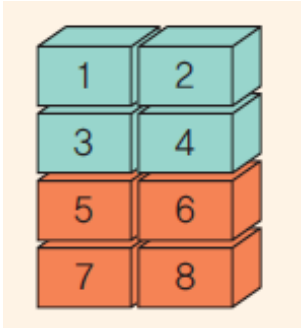
Numpy-arange

```
>>> np.arange(5)
array([0, 1, 2, 3, 4])

# 시작값을 지정하려면 다음과 같이 한다.
>>> np.arange(1, 6)
array([1, 2, 3, 4, 5])

# 증가되는 간격을 지정하려면 다음과 같이 한다.
>>> np.arange(1, 10, 2)
array([1, 3, 5, 7, 9])
```

Numpy-배열 합치기

```
>>> x = np.array([[1, 2], [3, 4]])
>>> y = np.array([[5, 6], [7, 8]])

>>> np.concatenate((x, y), axis=1)
array([[1, 2, 5, 6],
       [3, 4, 7, 8]])
```

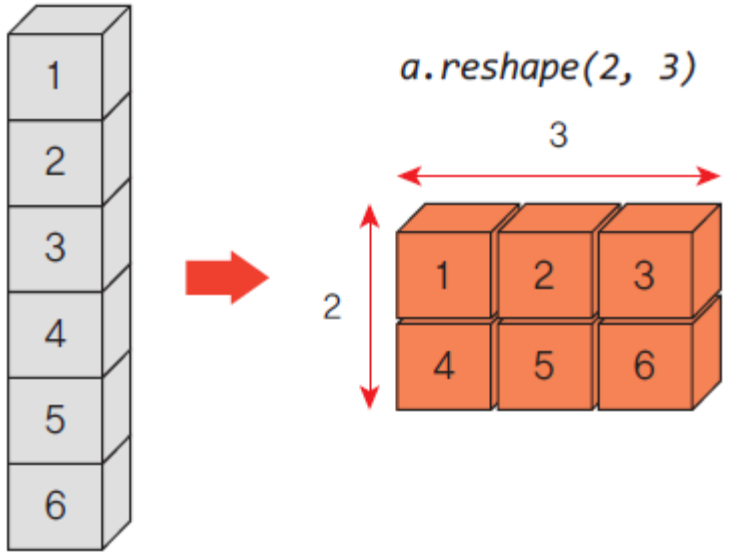




Numpy-reshape

```
new_array = old_array.reshape((2, 3))
```

새로운 배열 — 원래의 배열 — 새로운 배열의 형상



Numpy

```
>>> a = np.arange(12)
array([ 0,  1,  2,  3,  4,  5,  6,  7,  8,  9, 10, 11])

# a에 대하여 reshape(3, 4)를 호출하면 1차원 배열이 2차원 배열로 바뀌게 된다.
>>> a.reshape(3, 4)
array([[ 0,  1,  2,  3],
       [ 4,  5,  6,  7],
       [ 8,  9, 10, 11]])

>>> a.reshape(6, -1)
array([[ 0,  1],
       [ 2,  3],
       [ 4,  5],
       [ 6,  7],
       [ 8,  9],
       [10, 11]])
```

Numpy

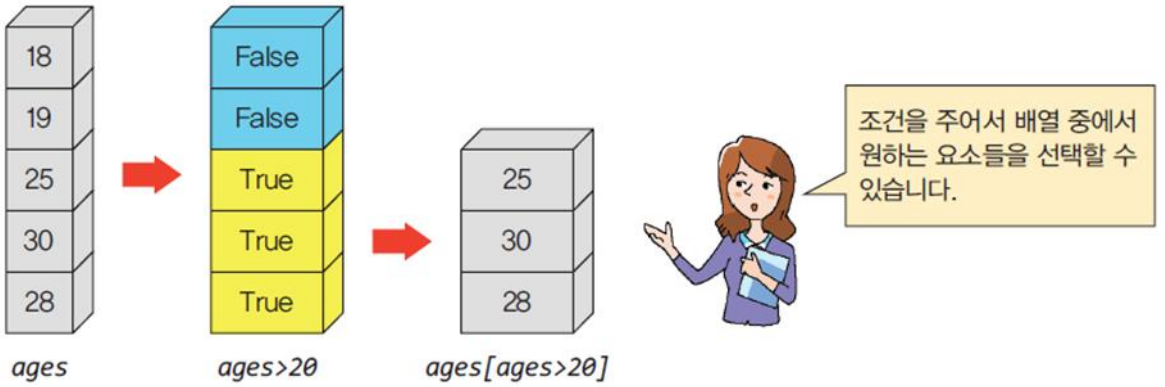
```
>>> array = np.arrange(30).reshape(-1, 10)
>>> array
Array([[ 0,  1,  2,  3,  4,  5,  6,  7,  8,  9],
       [10, 11, 12, 13, 14, 15, 16, 17, 18, 19],
       [20, 21, 22, 23, 24, 25, 26, 27, 28, 29]])
```

Numpy

```
>>> a = np.array([1, 2, 3, 4, 5, 6])
>>> a.shape
(6,)

>>> a1 = a[np.newaxis, :]
>>> a1
array([[1, 2, 3, 4, 5, 6]])
>>> a1.shape
(1, 6)

>>> a2 = a[:, np.newaxis]
array([[1],
       [2],
       [3],
       [4],
       [5],
       [6]])
>>> a2.shape
(6, 1)
```

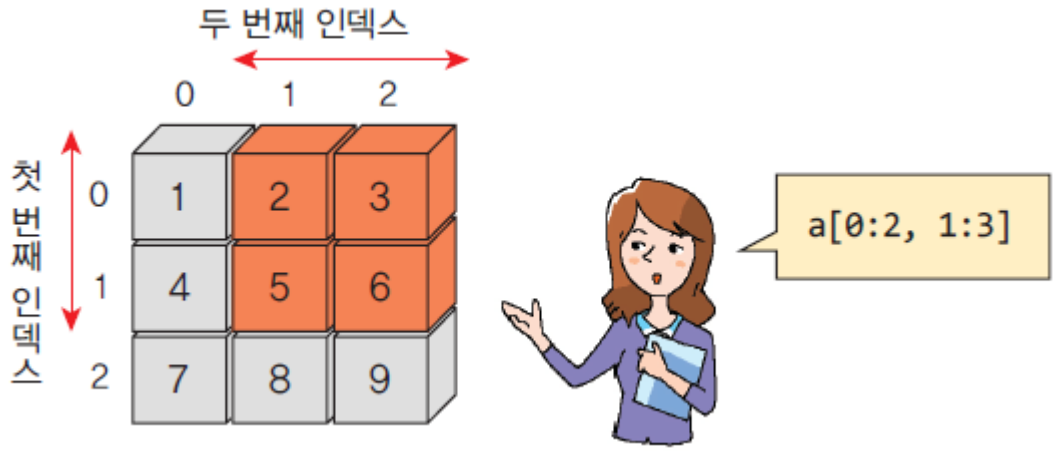
Numpy-인덱싱과 슬라이싱

```
>>> ages = np.array([18, 19, 25, 30, 28])
>>> ages[1:3] # 인덱스 1에서 인덱스 2까지
array([19, 25])
>>> ages[:2] # 인덱스 0에서 인덱스 1까지
array([18, 19])

# 논리적인 인덱싱(logical indexing)
>>> y = ages > 20
>>> y
array([False, False,  True,  True,  True])
>>> ages[ages > 20 ]
array([25, 30, 28])
```



Numpy-2차원 배열의 슬라이싱

```
>>> a = np.array([[1, 2, 3], [4, 5, 6], [7, 8, 9]])
>>> a[0:2, 1:3]
array([[2, 3],
       [5, 6]])
```



Numpy-2차원 배열의 슬라이싱

data[1,:]

(0, 0)	(0, 1)	(0, 2)	(0, 3)
(1, 0)	**(1, 1)**	**(1, 2)**	**(1, 3)**
(2, 0)	(2, 1)	(2, 2)	(2, 3)
(3, 0)	(3, 1)	(3, 2)	(3, 3)

data[:,2]

(0, 0)	(0, 1)	**(0, 2)**	(0, 3)
(1, 0)	(1, 1)	**(1, 2)**	(1, 3)
(2, 0)	(2, 1)	**(2, 2)**	(2, 3)
(3, 0)	(3, 1)	**(3, 2)**	(3, 3)

data[0:2,0:2]

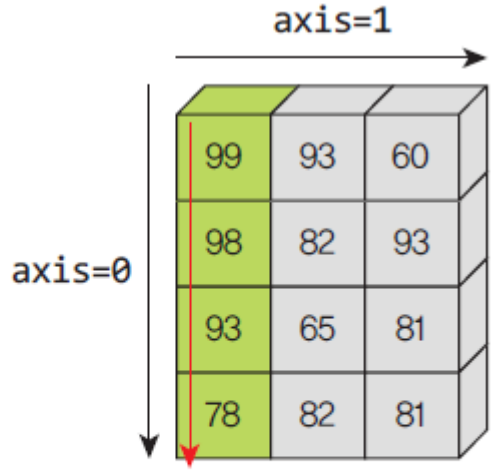
(0, 0)	**(0, 1)**	(0, 2)	(0, 3)
(1, 0)	**(1, 1)**	(1, 2)	(1, 3)
(2, 0)	(2, 1)	(2, 2)	(2, 3)
(3, 0)	(3, 1)	(3, 2)	(3, 3)

data[0:2,2:4]

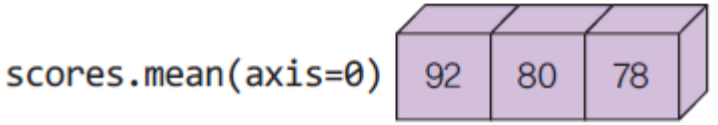
(0, 0)	(0, 1)	**(0, 2)**	**(0, 3)**
(1, 0)	(1, 1)	**(1, 2)**	**(1, 3)**
(2, 0)	(2, 1)	(2, 2)	(2, 3)
(3, 0)	(3, 1)	(3, 2)	(3, 3)

Numpy-특정한 행과 열을 이용한 연산

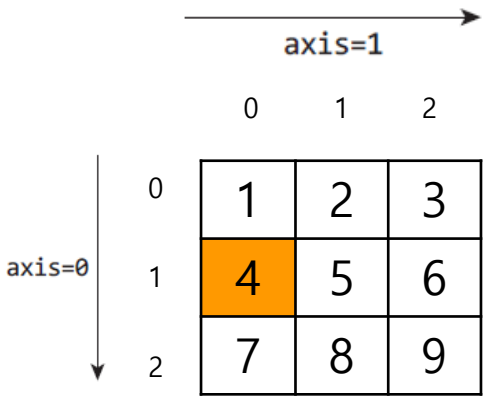
```
>>> scores = np.array([[99, 93, 60], [98, 82, 93],
  ...:                 [93, 65, 81], [78, 82, 81]])
>>> scores.mean(axis=0)
array([92.  , 80.5 , 78.75])
```




```
scores = np.array([[99, 93, 60], [98, 82, 93],
                   [93, 65, 81], [78, 82, 81]])
scores.sum(axis=0)

=> array([368, 322, 315])
```




```
scores = np.array([[99, 93, 60], [98, 82, 93],
                   [93, 65, 81], [78, 82, 81]])
scores.sum(axis=1)
```




```
import numpy as np
a = np.array([[1, 2, 3], [4, 5, 6], [7, 8, 9]])
a1=a[1:2,0:1]
a2=a[1:2,0]
a3=a[1,0:1]
a4=a[1,0]
print('a1',a1,a1.shape)
print('a2',a2,a2.shape)
print('a3',a3,a3.shape)
print('a4',a4,a4.shape)
```

```
a1 [[4]]   (1, 1)
a2 [4]   (1,)
a3 [4]   (1,)
a4 4    ()
```

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)
```

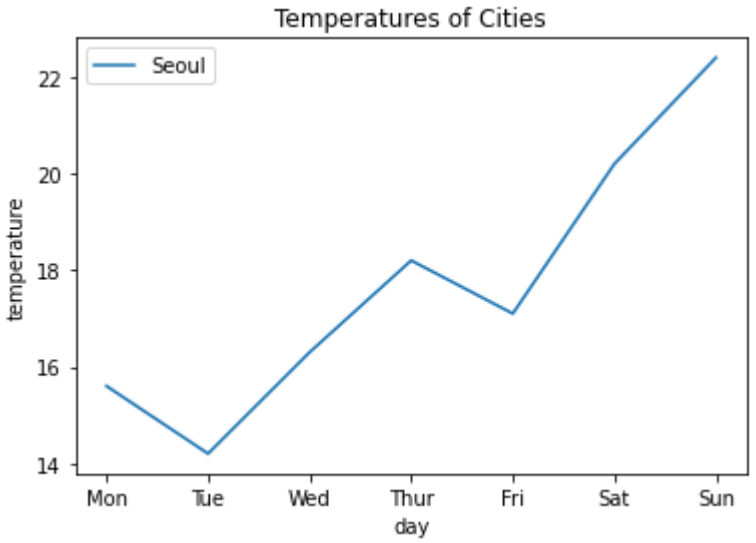
```
a  [1 2 3]          (3,)
a1 [[1 2 3]]        (1, 3)
a2 [[1]
    [2]
    [3]] (3, 1)
```

매트플롯

- 그래프를 그리는 라이브러리이다.

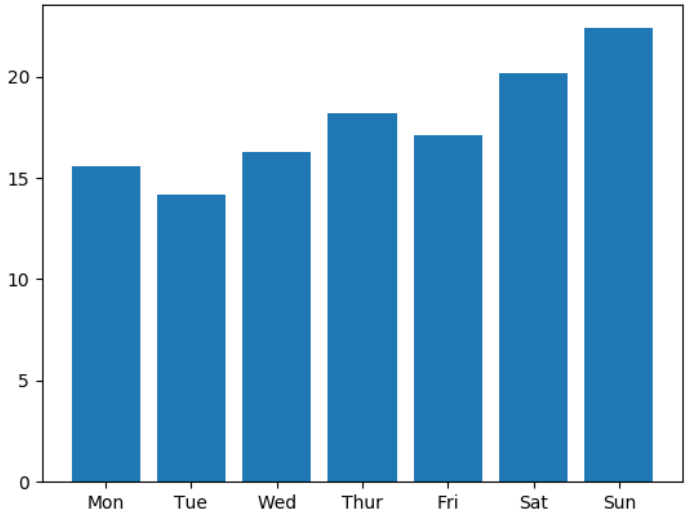
import matplotlib.pyplot as plt

```
X = [ "Mon", "Tue", "Wed", "Thur", "Fri",  "Sat", "Sun" ]
Y1 = [15.6, 14.2, 16.3, 18.2, 17.1, 20.2, 22.4]

plt.plot(X, Y1, label="Seoul")   # 분리시켜서 그려도 됨
plt.xlabel("day")
plt.ylabel("temperature")
plt.legend(loc="upper left")
plt.title("Temperatures of Cities")
plt.show()
```



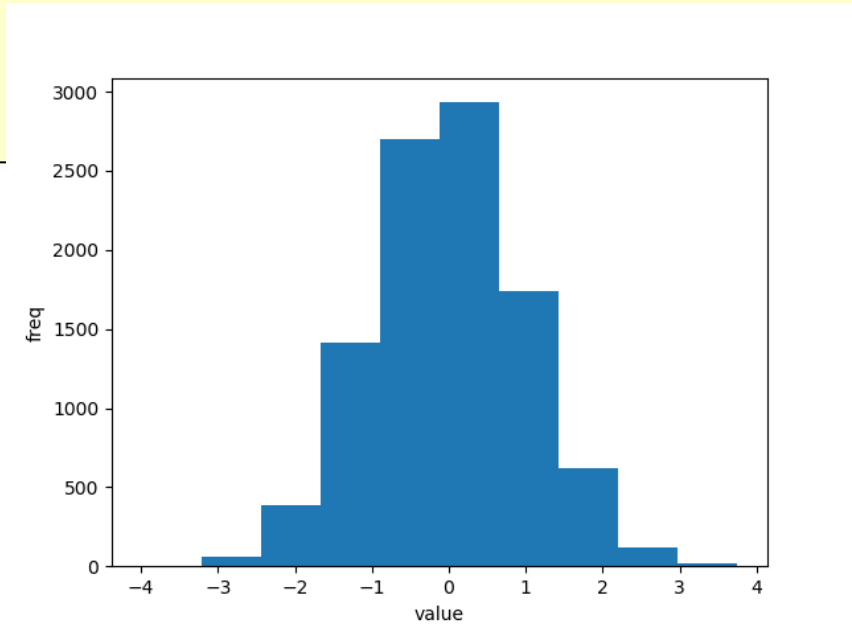
매트플롯

```
import matplotlib.pyplot as plt

X = [ "Mon", "Tue", "Wed", "Thur", "Fri",  "Sat", "Sun" ]
Y = [15.6, 14.2, 16.3, 18.2, 17.1, 20.2, 22.4]
plt.bar(X, Y)
plt.show()
```



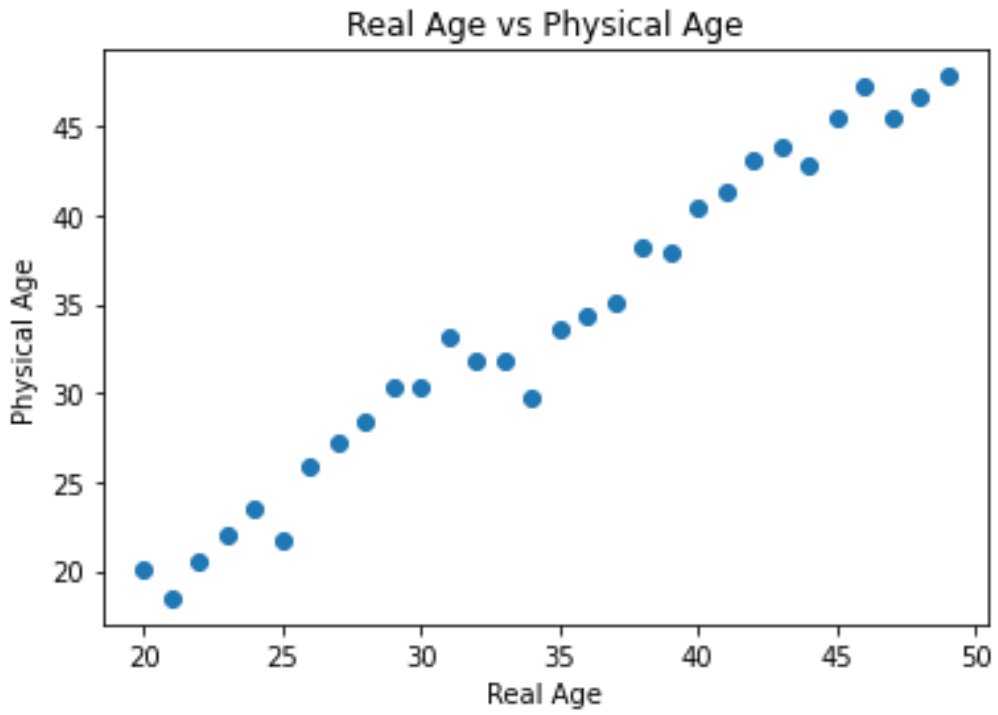
매트플롯

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

numbers = np.random.normal(size=10000)

plt.hist(numbers)
plt.xlabel("value")
plt.ylabel("freq")
plt.show()
```



맷플롯 -산포도

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

xData=np.arange(20,50)
yData=xData+2*np.random.randn(30)

plt.scatter(xData,yData)
plt.title('Real Age vs Physical Age')
plt.xlabel('Real Age')
plt.ylabel('Physical Age')
plt.show()
```



퍼셉트론 (Perceptron)

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

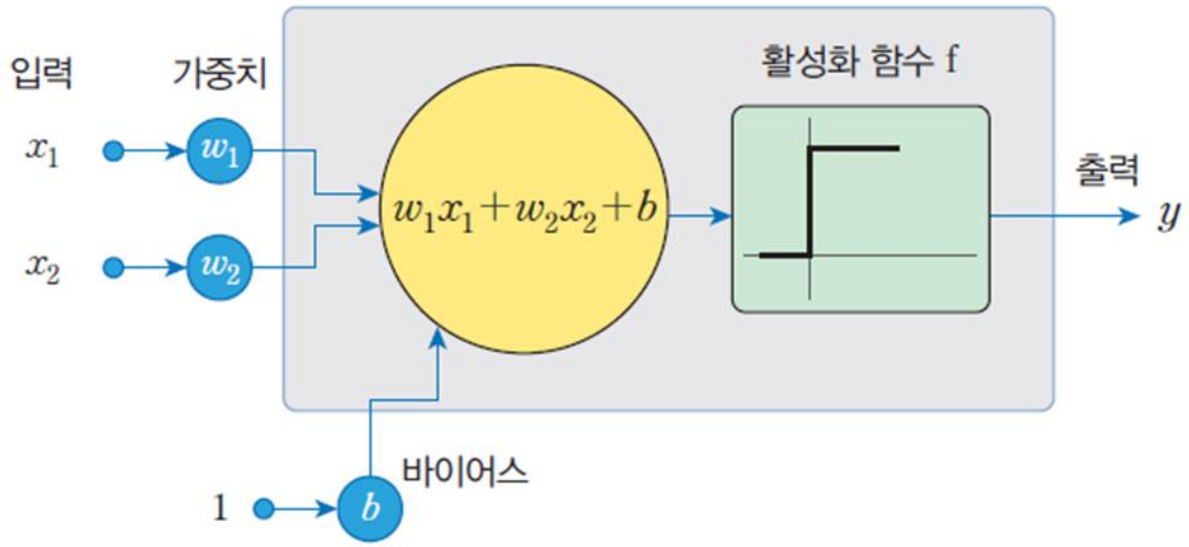


그림 5-5 퍼셉트론에서의 뉴런의 모델

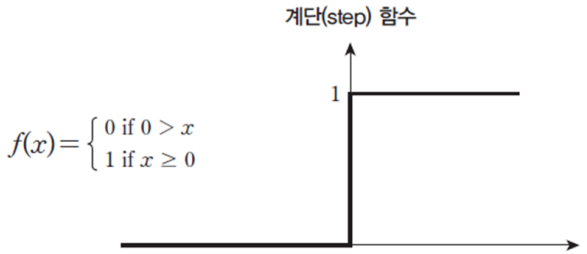


그림 5-7 퍼셉트론에서의 활성화 함수

-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)

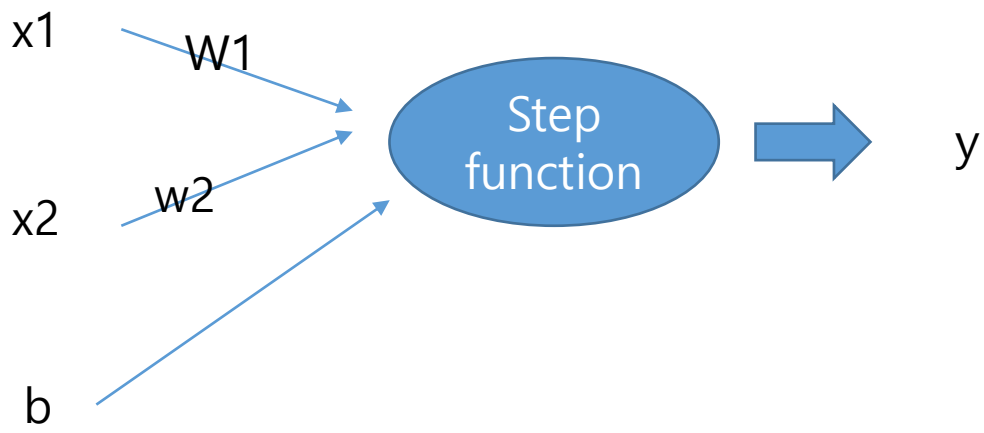
- 뉴런에서는 입력 신호의 가중치 합이 어떤 임계값을 넘는 경우에만 뉴런이 활성화되어서 1을 출력한다. 그렇지 않으면 0을 출력한다.

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

퍼셉트론 (Perceptron)

입력 (1,0)

w1: 1
w2: 1
B=-1.5



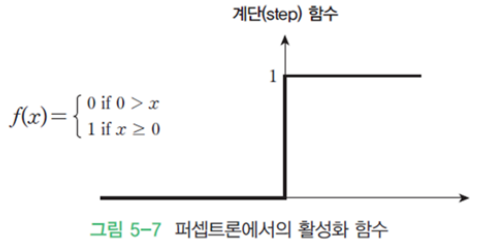


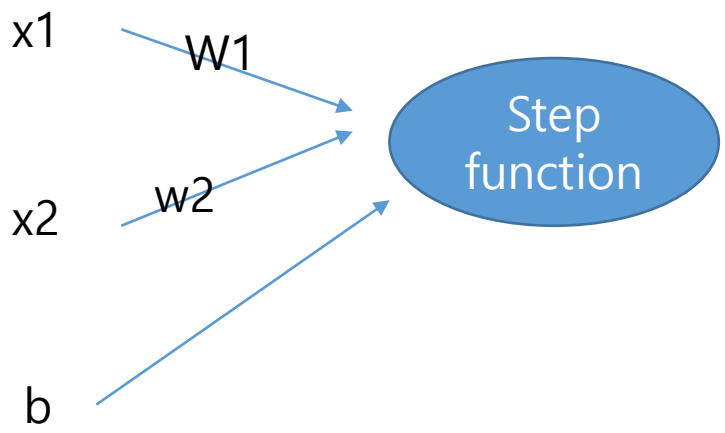
그림 5-7 퍼셉트론에서의 활성화 함수

출력값 y는?

퍼셉트론 (Perceptron)

입력 (1,0)

w1: 1
w2: 1
B=-1.5



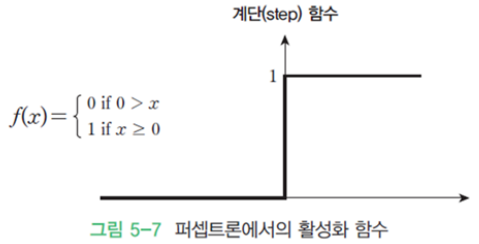


그림 5-7 퍼셉트론에서의 활성화 함수

출력값 y는? x1*w1+x2*w2+b= -> 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

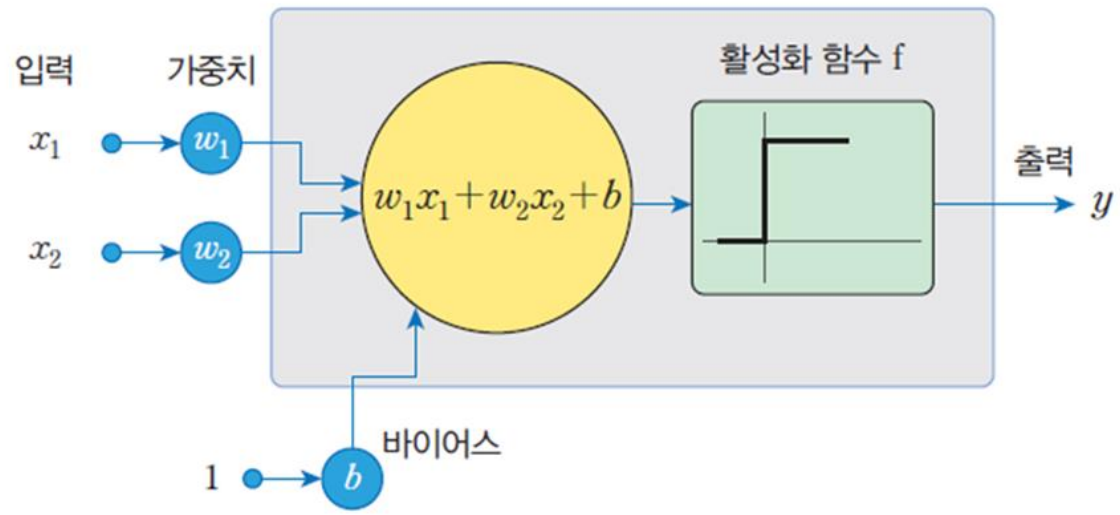


그림 5-5 퍼셉트론에서의 뉴런의 모델

만족시키는 w1,w2,b 찾기

w1=1,w2=1,b=-1.5 를 테스트 해보기

퍼셉트론 (Perceptron) - 논리 연산

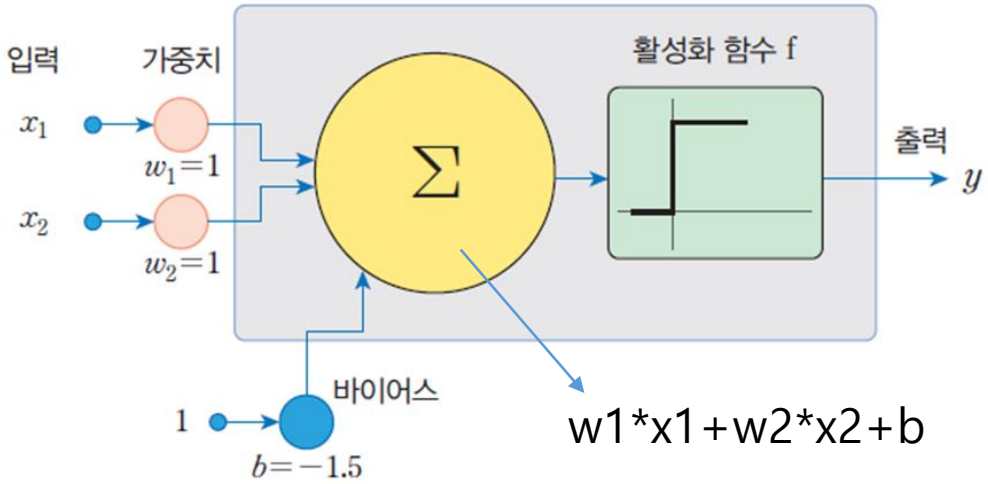


그림 5-6 논리 연산을 하는 퍼셉트론

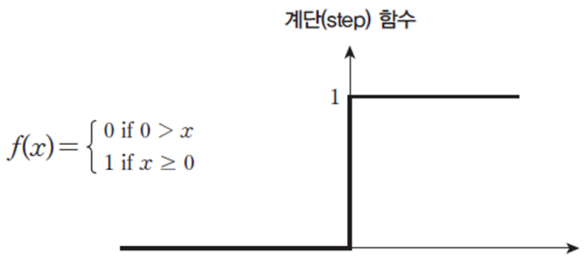


그림 5-7 퍼셉트론에서의 활성화 함수

x_1	x_2
0	0
1	0
0	1
1	1

?

가중치의 합

Y

퍼셉트론 (Perceptron) - 논리 연산

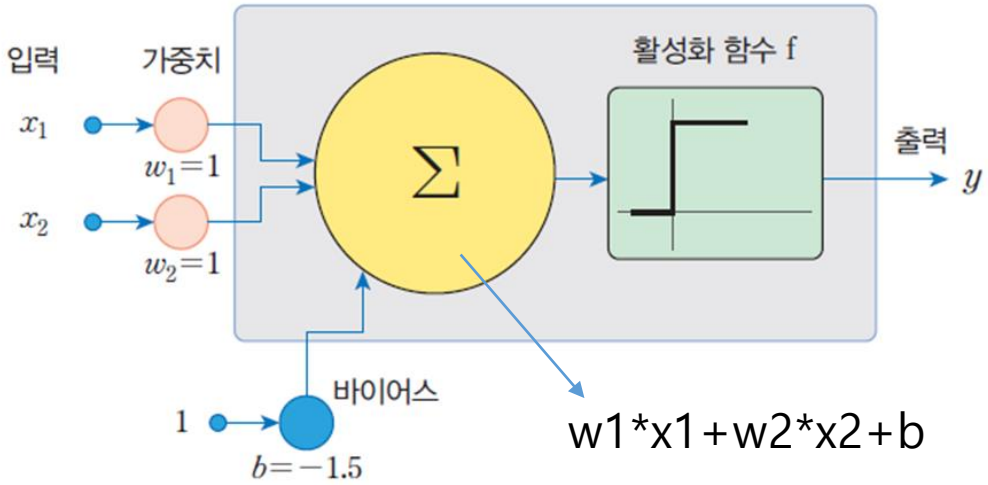


그림 5-6 논리 연산을 하는 퍼셉트론

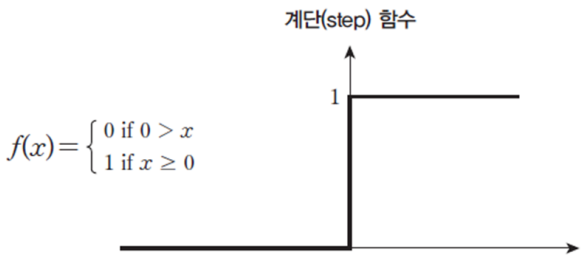


그림 5-7 퍼셉트론에서의 활성화 함수

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

활성화 함수

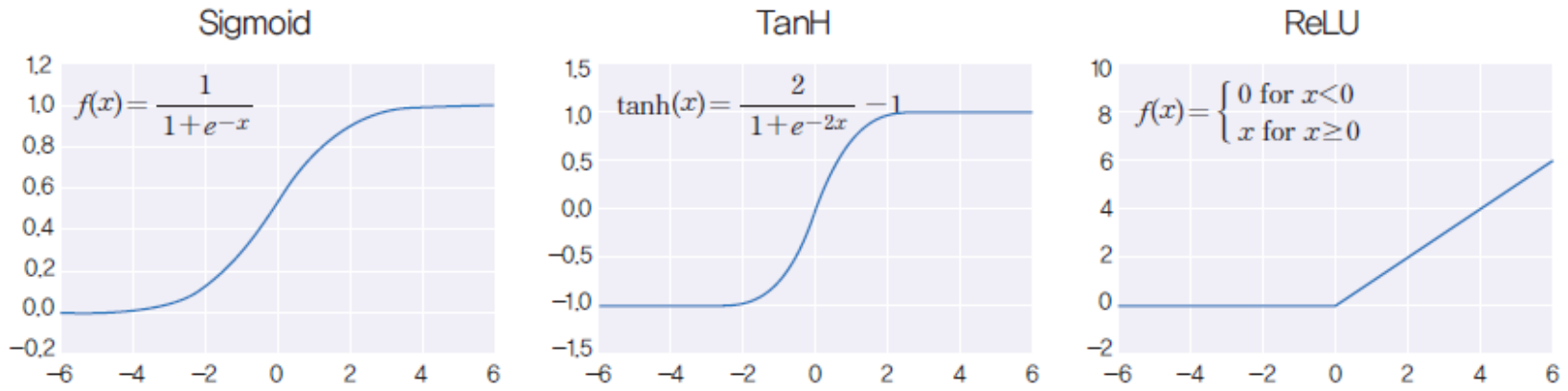
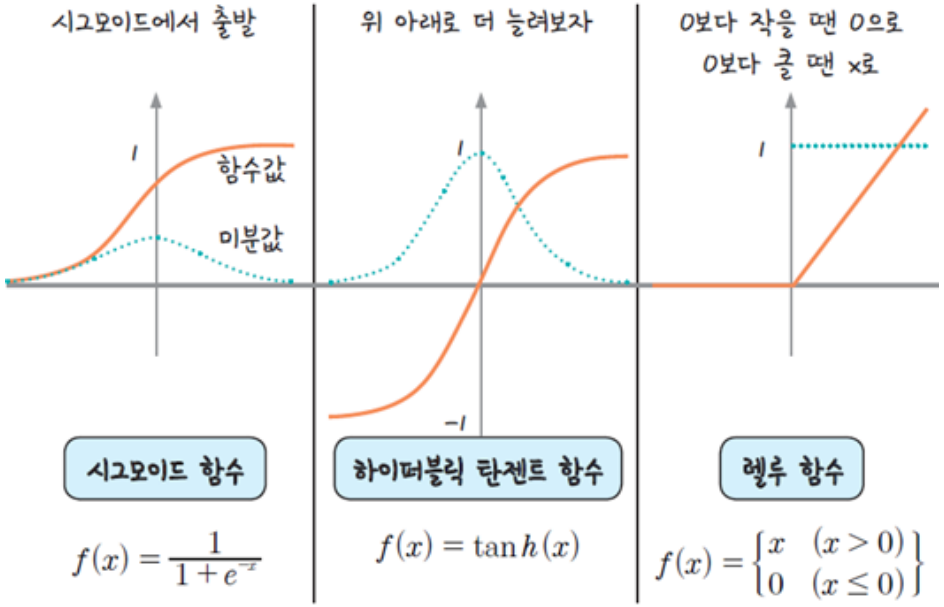


그림 6-3 많이 사용되는 활성화 함수



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

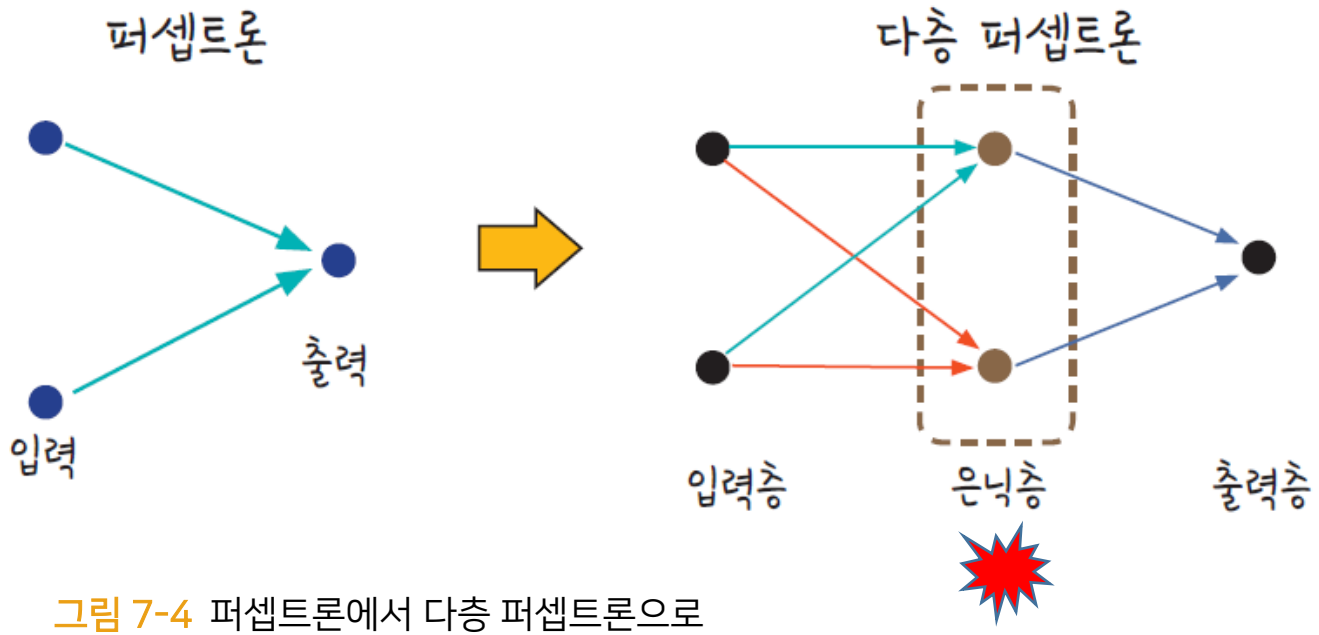
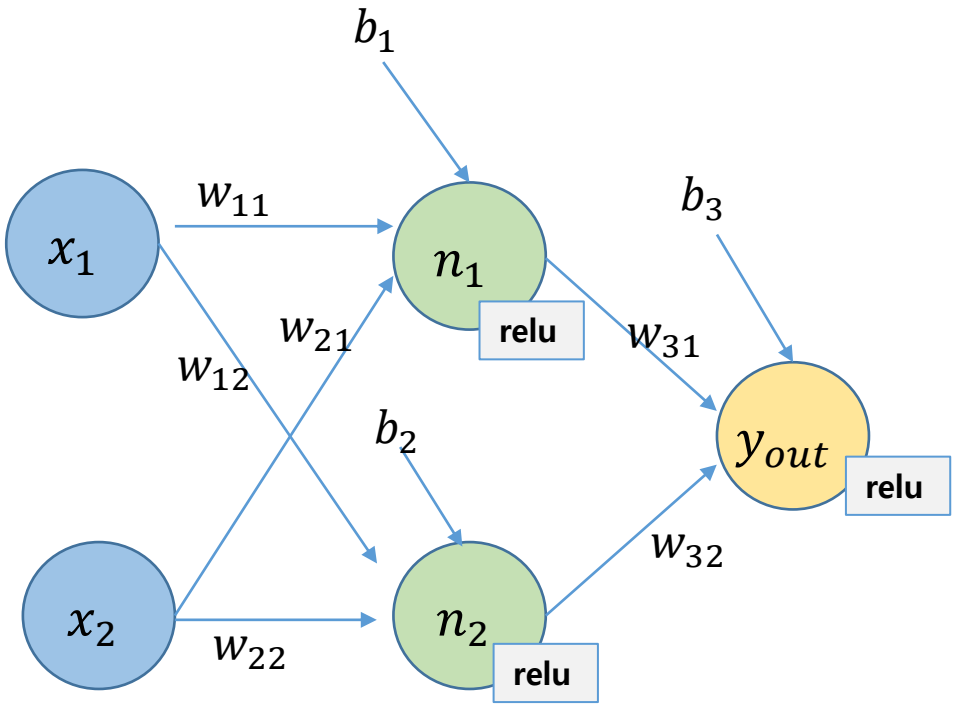
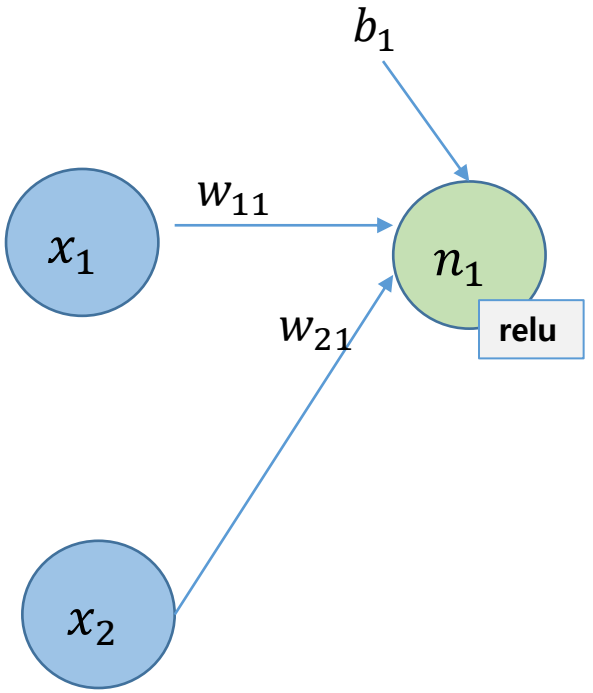


그림 7-4 퍼셉트론에서 다층 퍼셉트론으로

다중 퍼셉트론 (Multi Layer Perceptron: MLP)

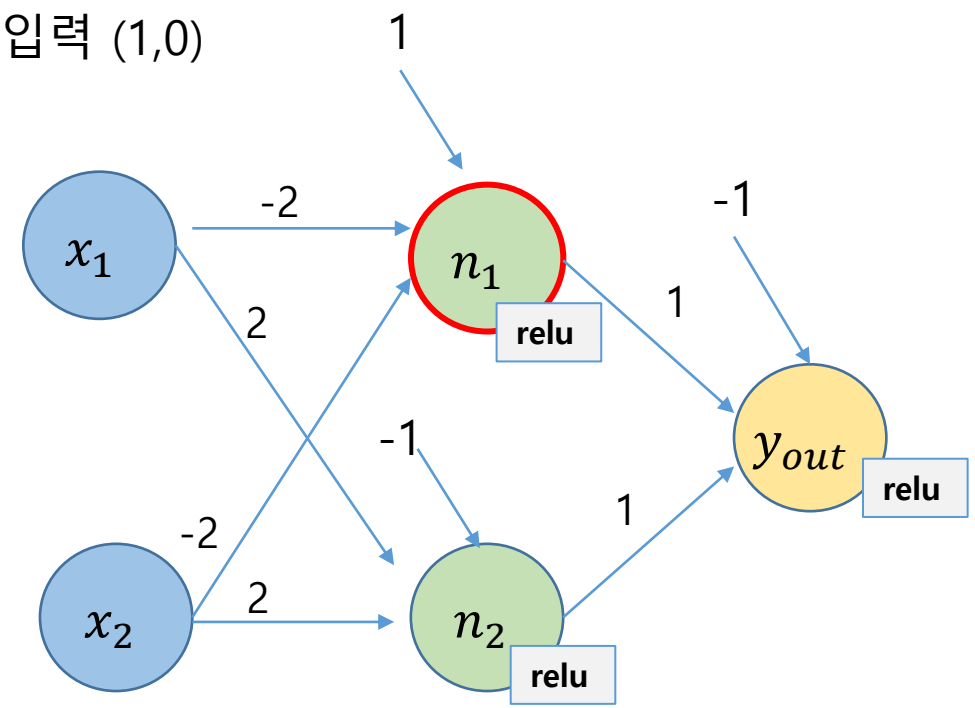
입력 (1,0)





다중 퍼셉트론 (Multi Layer Perceptron: MLP)

입력 (1,0)

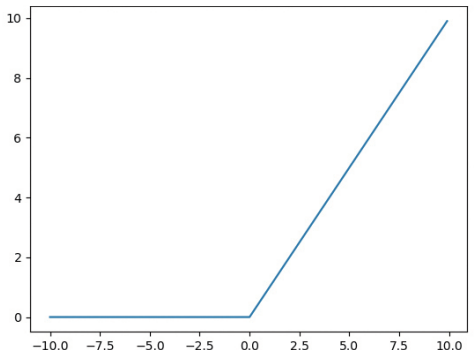


다음 다중 퍼셉트론의 출력은?

w_{12}

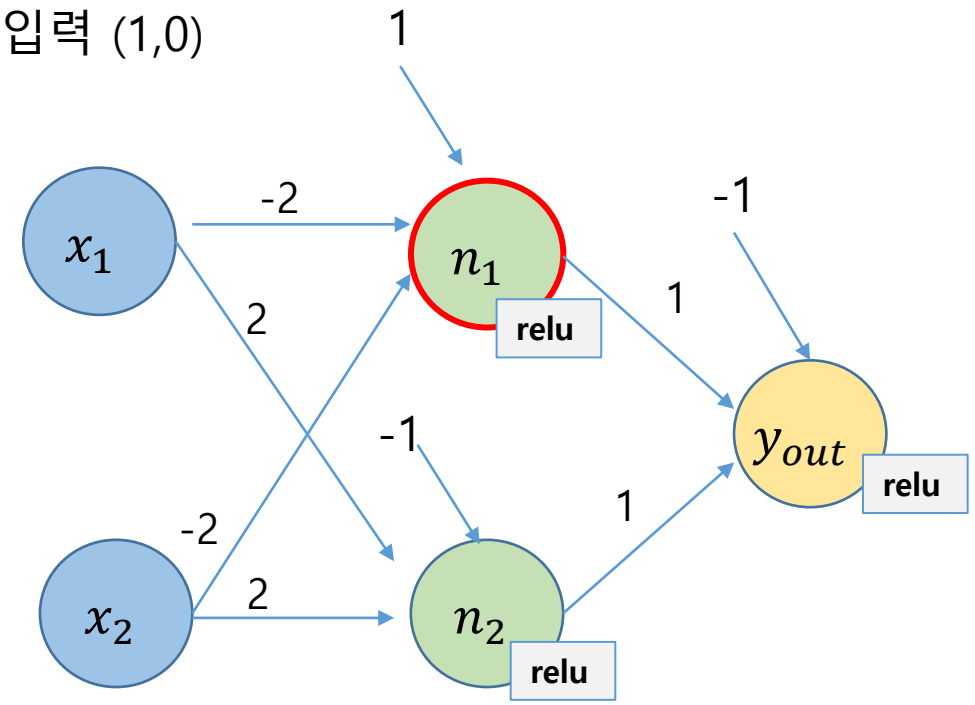
$$f(x) = \begin{cases} x & x > 0 \\ 0 & x \leq 0 \end{cases}$$

Relu



다중 퍼셉트론 (Multi Layer Perceptron: MLP)

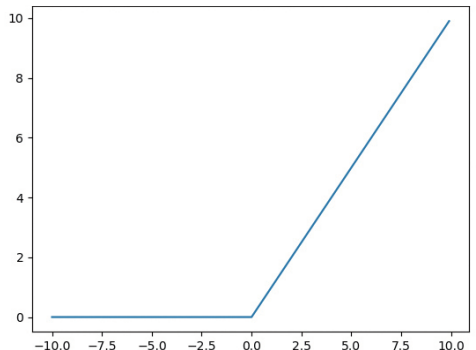
입력 (1,0)



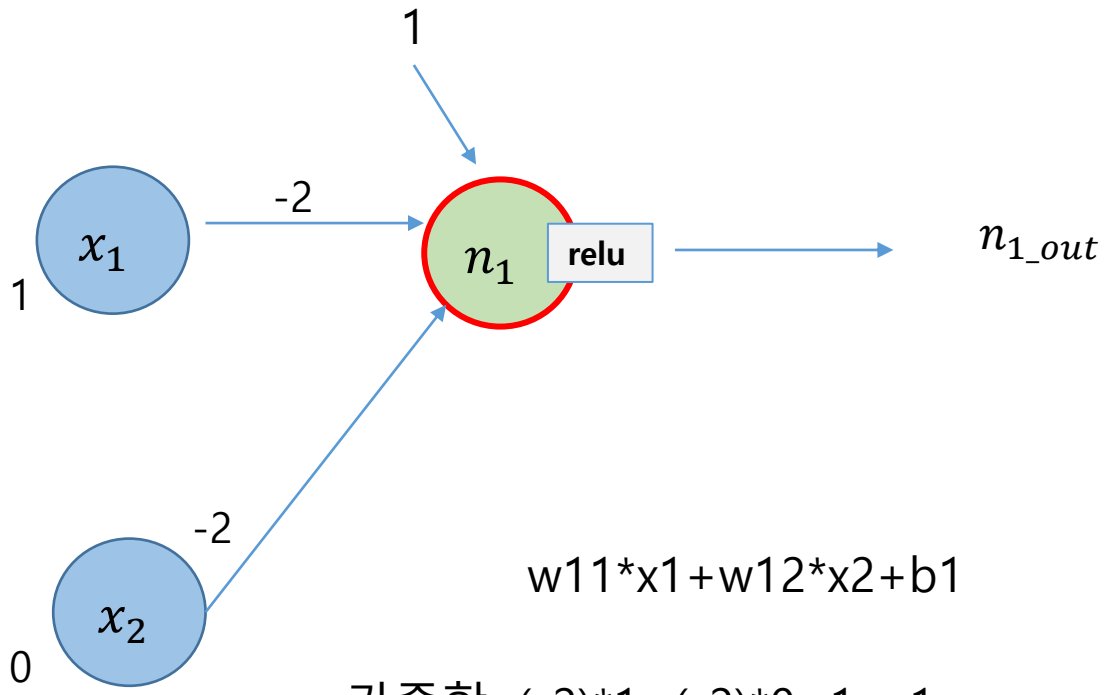
w_{12}

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

Relu



다음 다중 퍼셉트론의 출력은?



w11*x1+w12*x2+b1

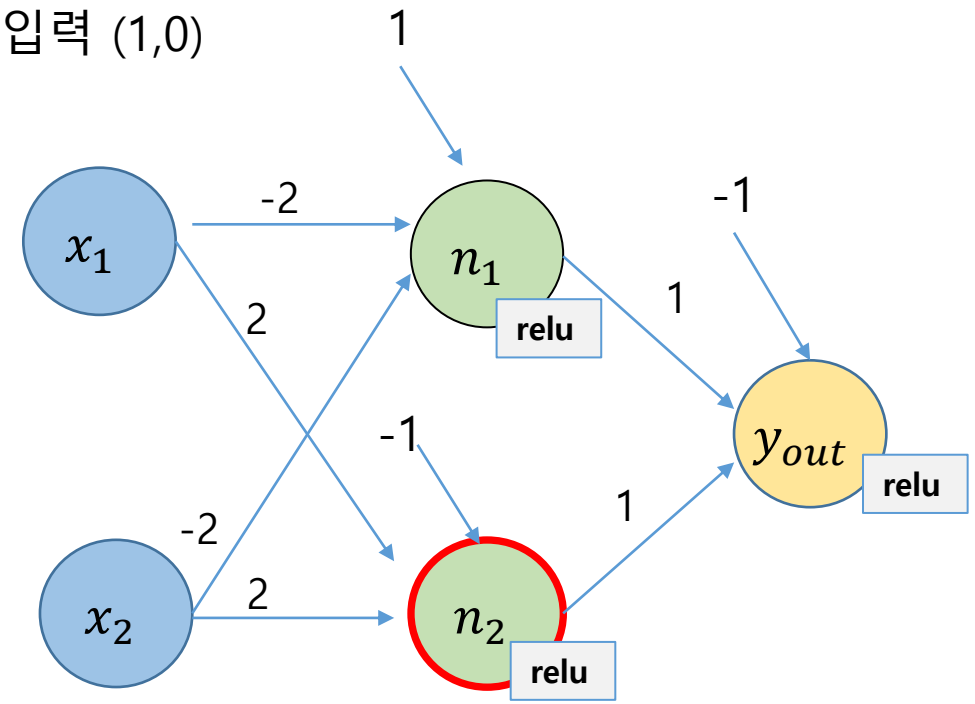
가중합: (-2)*1+(-2)*0+1=-1

Relu(-1)=0

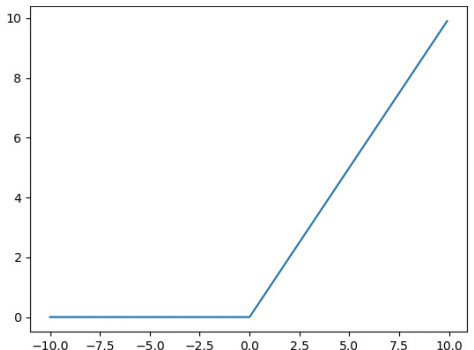
n1_out=0

다중 퍼셉트론 (Multi Layer Perceptron: MLP)

입력 (1,0)

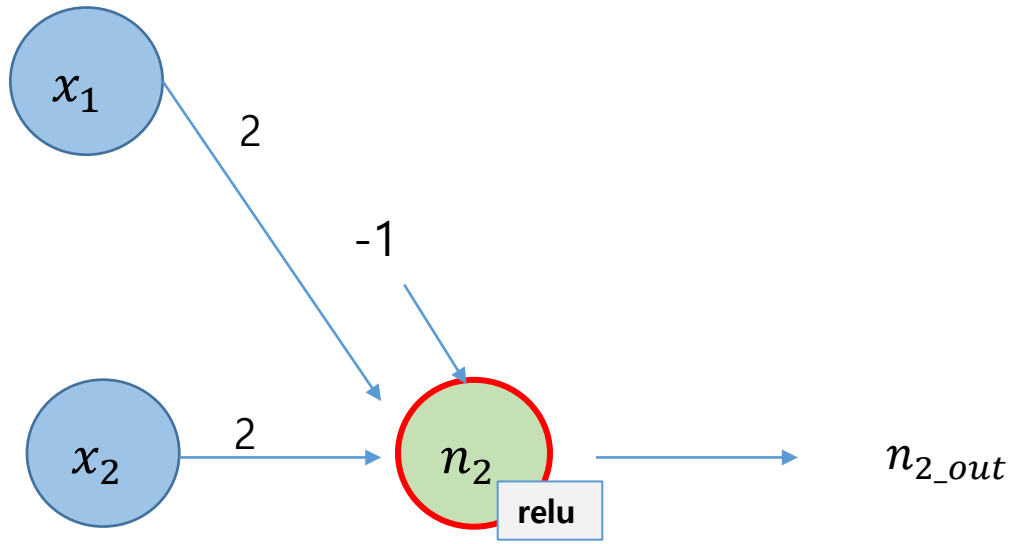


Relu



다음 다중 퍼셉트론의 출력은?

입력 (1,0)



w21*x1+w22*x2+b2

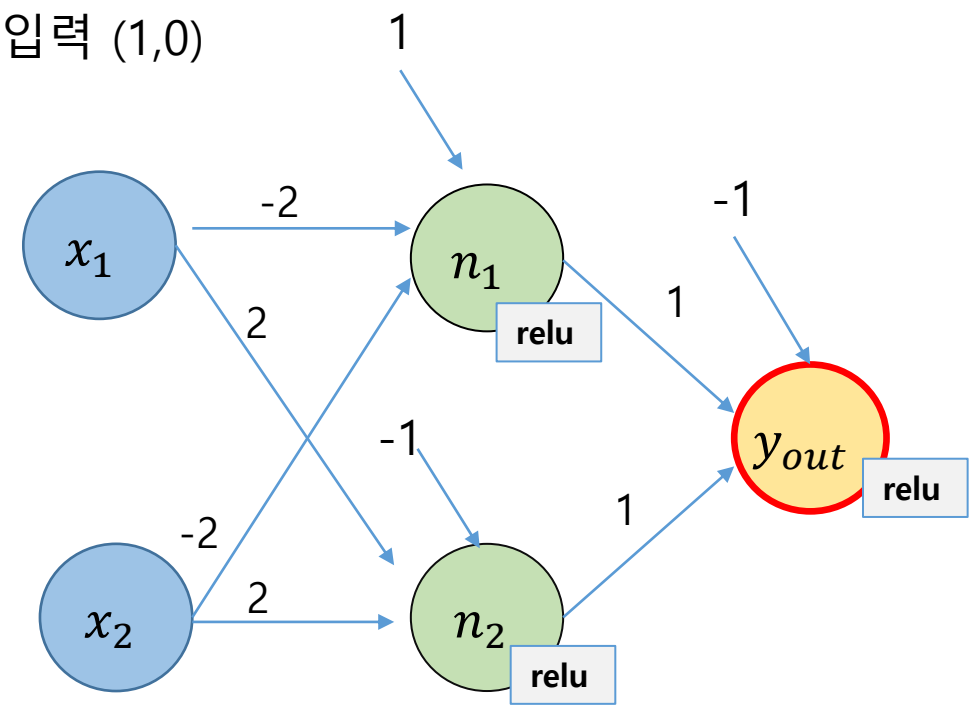
가중합: 1*(2)+2*(0)-1=1

Relu(1)=1

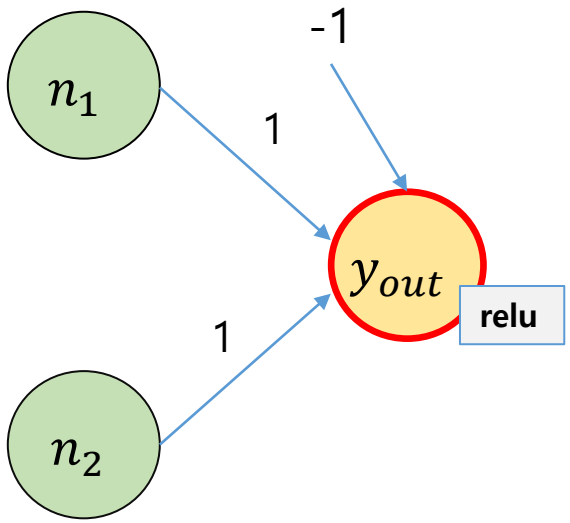
N2_out=1

다중 퍼셉트론 (Multi Layer Perceptron: MLP)

입력 (1,0)



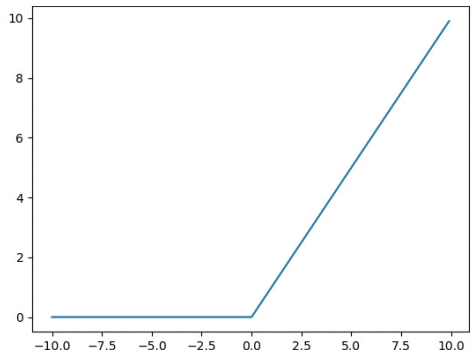
다음 다중 퍼셉트론의 출력은?



n1_out=0

n2_out=1

Relu



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.

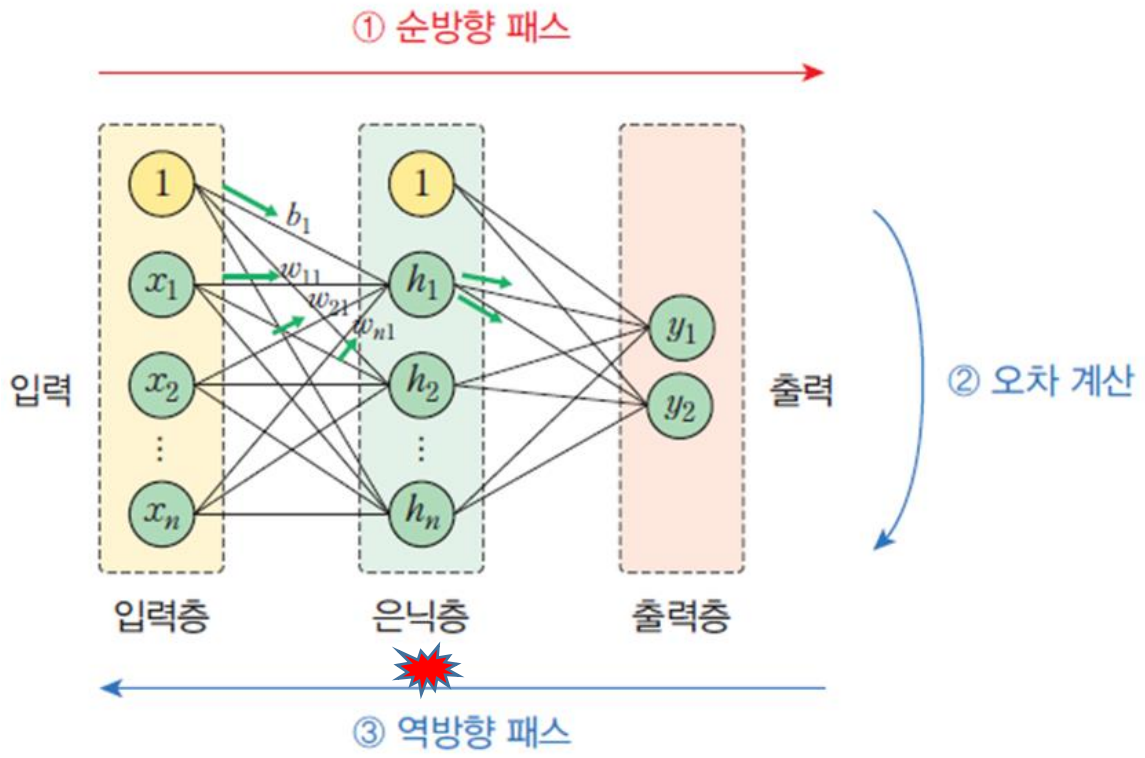


그림 6-5 순방향 패스

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

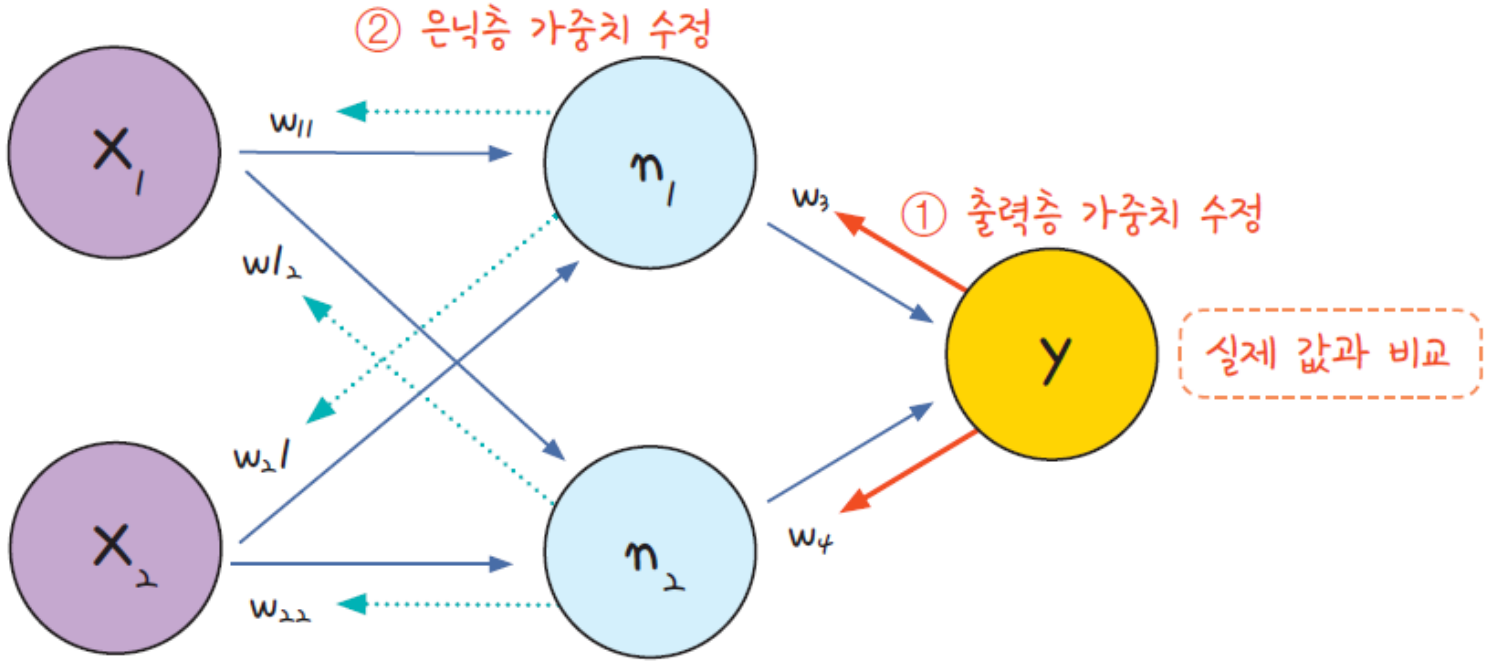
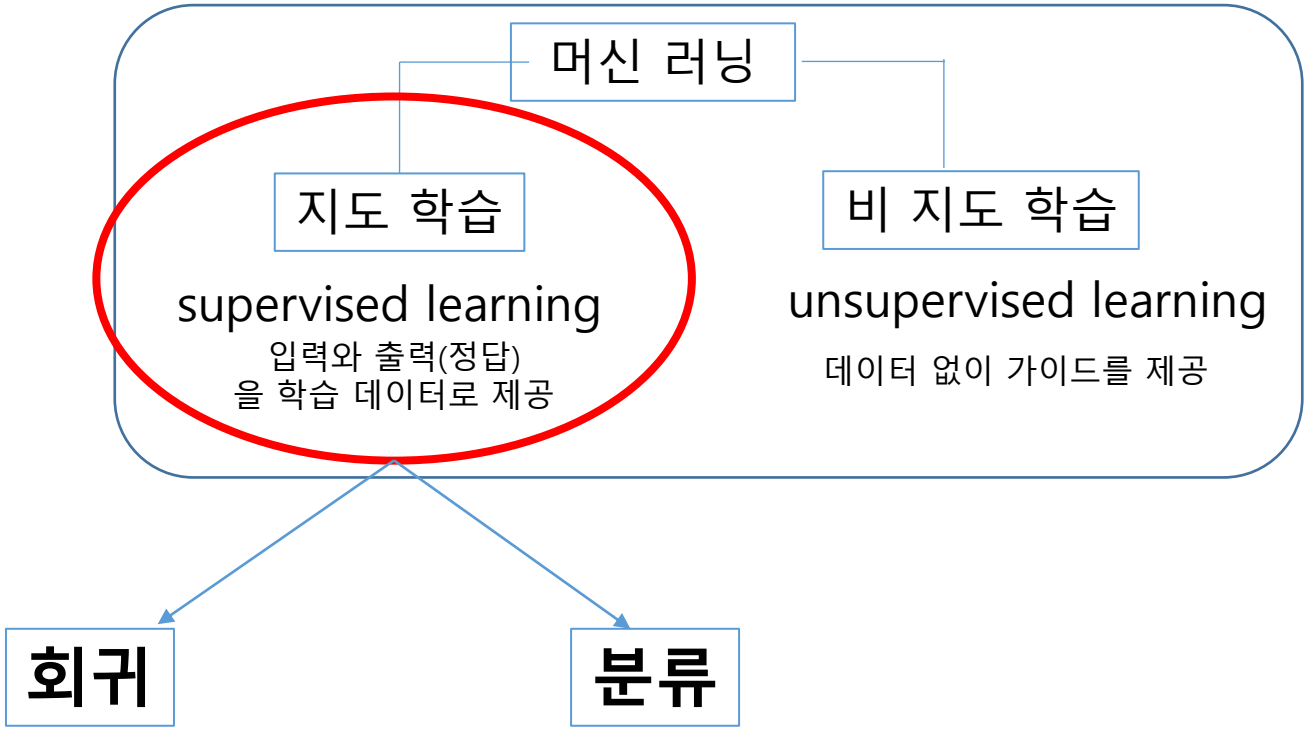
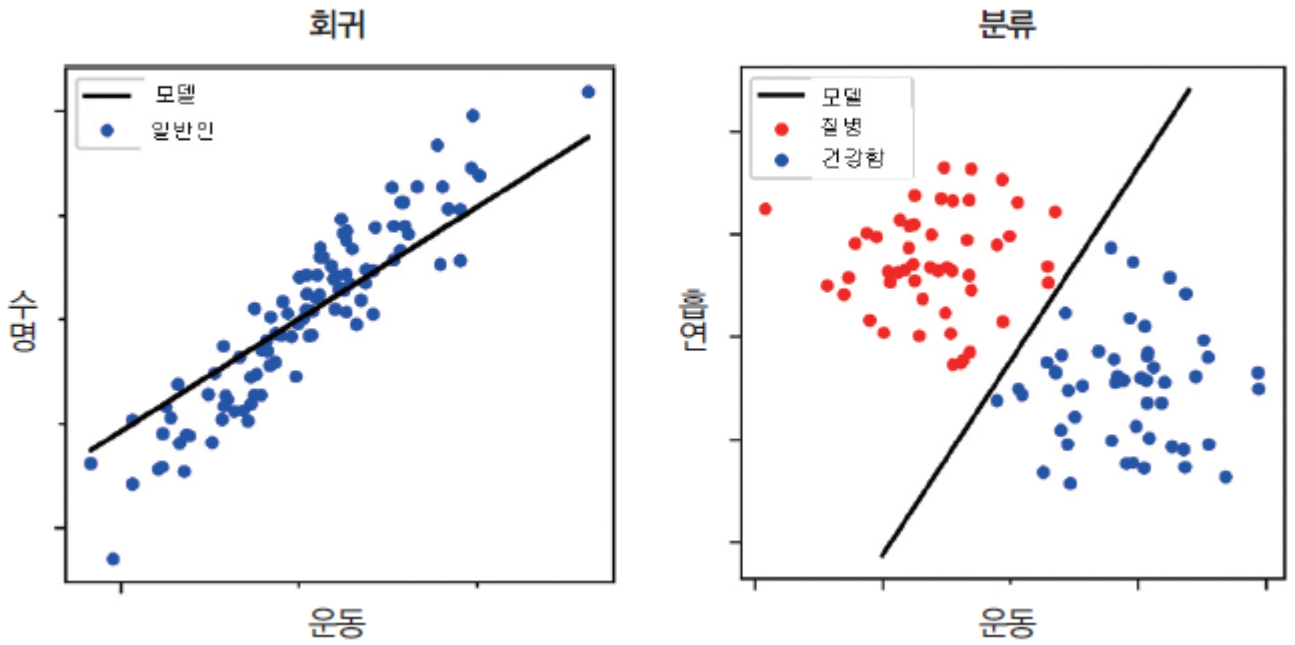


그림 8-2 다층 퍼셉트론에서의 오차 수정

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.

$$y = f(x)$$

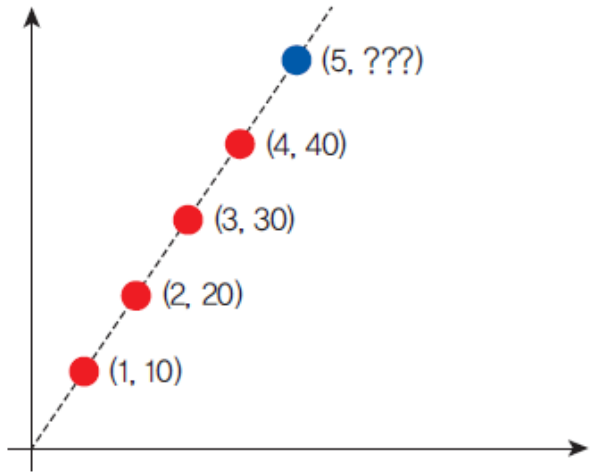
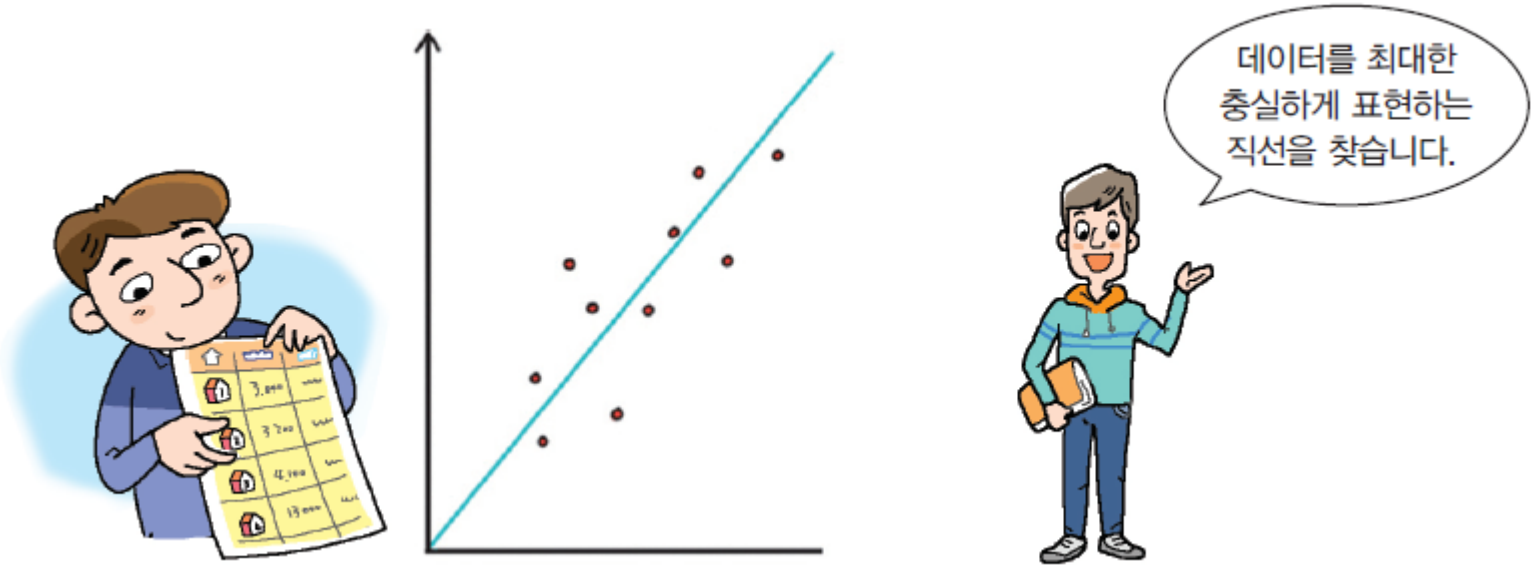


그림 3-7 회귀

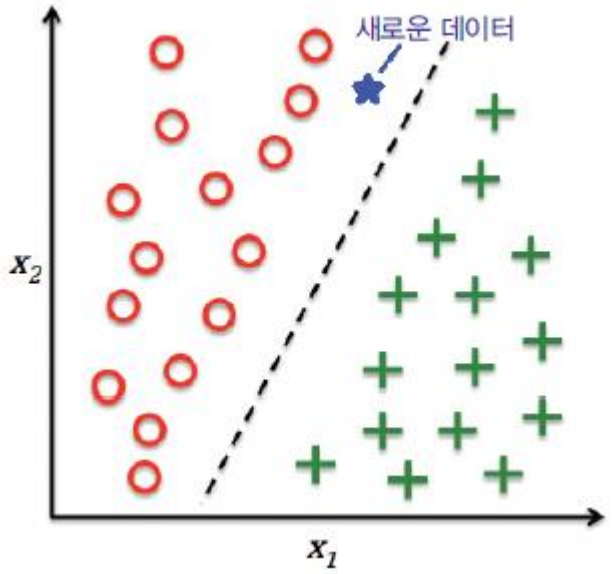
지도 학습 : 분류 vs. 회귀

- **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

- 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

수고 하셨습니다

jhmin@inhatc.ac.kr