머신러닝&딥러닝-필요한 수학/통계학이론

선형 대수학의 기초인 스칼라, 벡터, 행렬

- 필요한 이유?
  - 신경망의 내부 작동을 이해하고 신경망을 집합적으로 구축하는 수학(선형 대수, 확률, 미적분학)에 익숙해져야 함
- 스칼라(scala)](https://)
  - 크기로만 설명되는 물리량
  - 스칼라는 숫자 값만으로 표현되는 양
  - 예) 부피, 밀도, 속도, 연령 등
  - 스칼라 변수는 일반 소문자(예: x, y, z)로 표시됨
- 벡터(vector)
  - 숫자의 배열
  - 숫자는 순서대로 배열되어 있으며 해당 순서의 색인을 통해 각 개별 숫자를 식별할 수 있음
  - 공간의 점을 식별하는 것으로 생각할 수 있으며, 각 요소는 서로 다른 축을 따라 좌표를 제공함
  - 크기와 방향을 모두 갖는 양을 나타내는 화살표
  - 화살표의 길이는 크기를 나타내고 방향은 방향을 나타냄
- 행렬(matrix)
  - 숫자의 2차원 배열

- 벡터가 0차에서 1차로 스칼라를 일반화하는 것처럼 행렬은 1차에서 2차로 벡터를 일반화함

```
1 #loading numpy
2 import numpy as np
3 # creating a row vector
4 row_vec = np.array([1, 2, 3])
5 print(row_vec)
6 #creating a column vector
7 column_vec = np.array([[1],
8                        [2],
9                        [3]])
10 print(column_vec)
```

```
[1 2 3]
[[1]
 [2]
 [3]]
```

```
1 # loading numpy
2 import numpy as np
3
4 # Creating a 3*4 matrix
5 matrix = np.array([[1, 2, 3 , 4],
6                    [5, 6, 7, 8],
7                    [9, 10, 11, 12]])
8 print(matrix)
```

```
[[ 1  2  3  4]
 [ 5  6  7  8]
 [ 9 10 11 12]]
```

```
1 # loading numpy
2 import numpy as np
3
4 tensor = np.array([
5   [[1,2,3],    [4,5,6],    [7,8,9]] ,
6   [[10,11,12], [13,14,15], [16,17,18]] ,
7   [[19,20,21], [22,23,24], [25,26,27]] ,
8   ])
9 print(tensor)
```

```
[[[ 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]]]
```

텐서(Tensor)란 무엇인가요?

정의

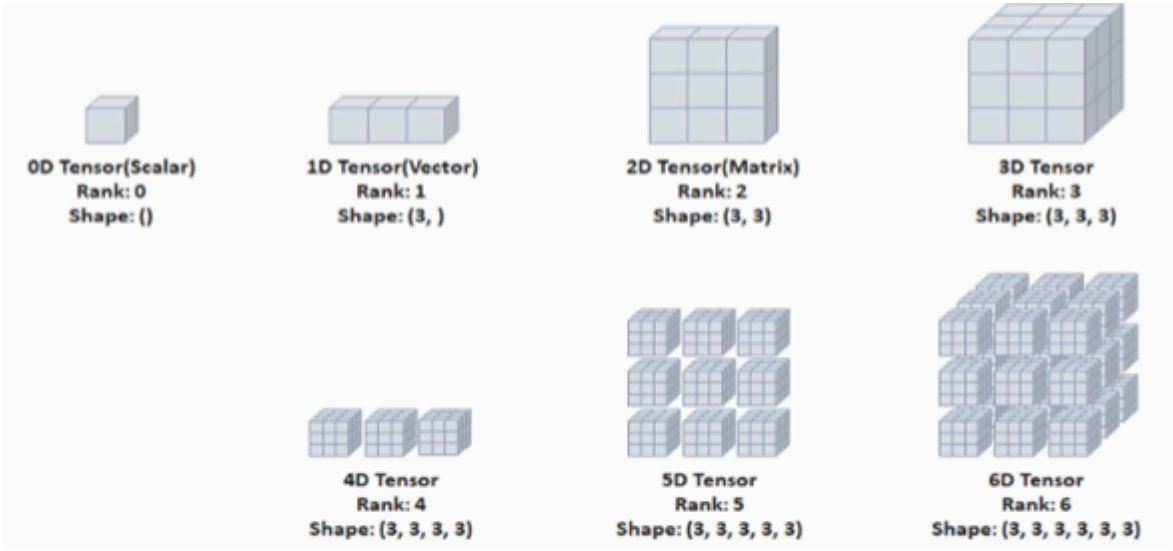
- 숫자들을 담는 커다란 상자 같은 것
- 머신러닝 시스템에서 다루는 모든 정보는 이 텐서 안에 저장됨 다차원의 배열을 통칭
- 딥러닝에서 텐서는 스칼라, 벡터, 행렬등의 다양한 모양의 숫자들을 모아놓은, 구조를 갖춘 덩어리를 지칭

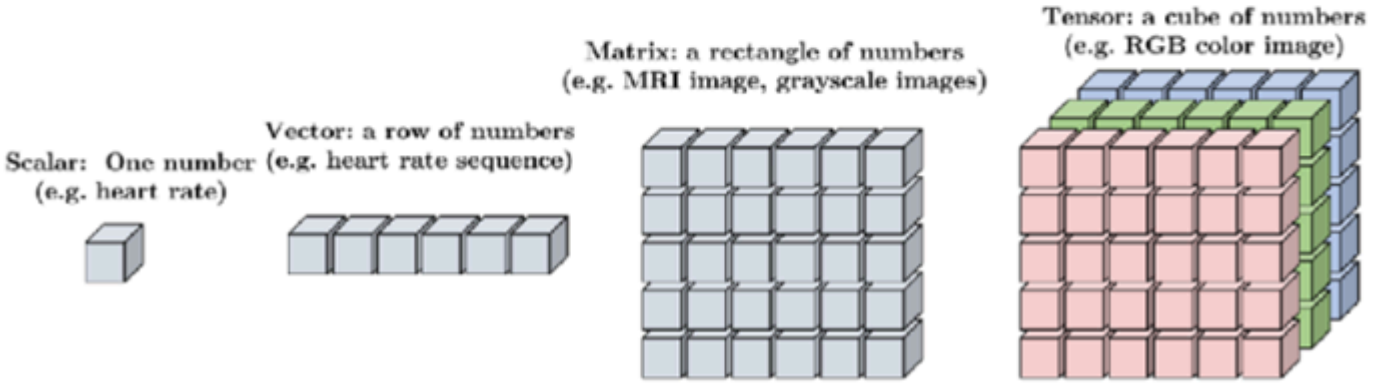
  - 딥러닝은 모델이라고 부르는 커다란 텐서들의 층를 통해 숫자들이 지나가면서 추론을 하기도 하고, 반대방향으로 거슬러 올라가면서 학습을 하기도 하는 과정을 지칭
  - 구글에서 만든 유명한 딥러닝 라이브러리 이름이 텐서플로우(TensorFlow)이기도 함

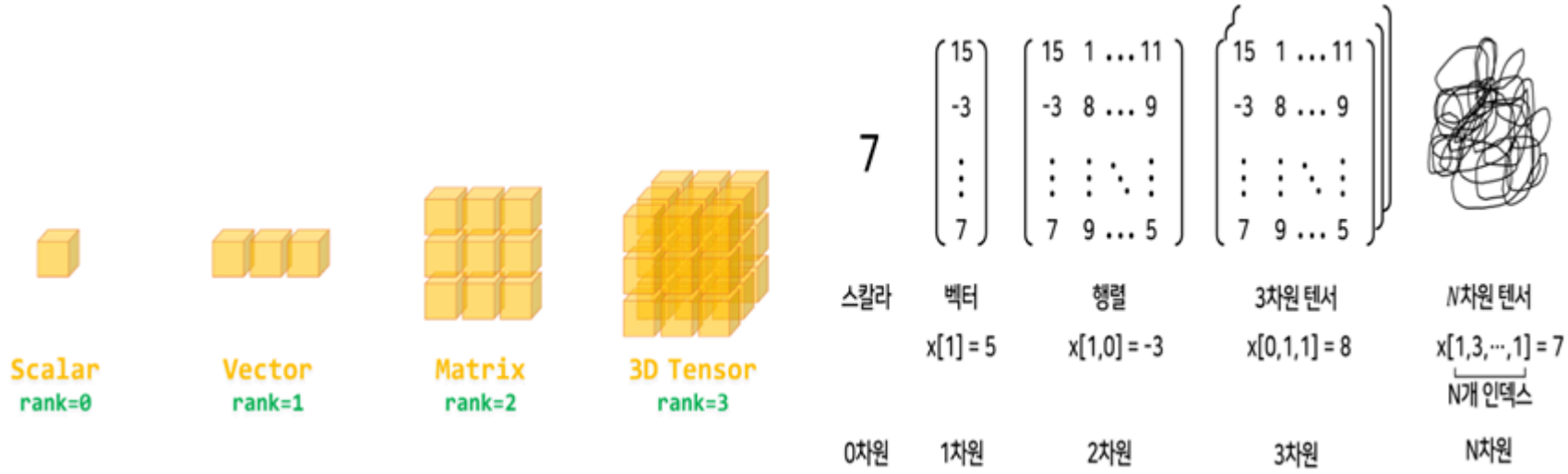
    - 물론 지금은 파이토치가 대세화 되었지만, 파이토치도 가장 핵심은 텐서임
- 텐서의 축의 갯수(rank)와 각 축의 길이(dimension)가 모양을 정의
    - 축이 없이 숫자 하나만 달랑 있는 것을 스칼라(Scalar)
    - 축이 1개이고 그 축 안에 n 개의 숫자가 나열된 것을 n차원 벡터 (Vector)
    - 2개의 축에 테이블 형식으로 숫자가 나열된 것을 n*m 행렬(Matrix)

텐서는 크게 세 가지 특성으로 정의

- 랭크(Rank) - 텐서의 축의 개수, 예를 들어, 벡터는 축이 하나뿐이니까 랭크가 1임
- 형태(Shape) - 각 축을 따라 차원의 수, 예를 들어, 정사각형 행렬은 (2, 2) 형태를 가짐
- 데이터 타입(Data Type)- 텐서 안에 담긴 데이터의 유형, float32, float64, int32 등


0D Tensor(Scalar)
Rank: 0
Shape: ()
1D Tensor(Vector)
Rank: 1
Shape: (3,)
2D Tensor(Matrix)
Rank: 2
Shape: (3, 3)
3D Tensor
Rank: 3
Shape: (3, 3, 3)
4D Tensor
Rank: 4
Shape: (3, 3, 3, 3)
5D Tensor
Rank: 5
Shape: (3, 3, 3, 3, 3)
6D Tensor
Rank: 6
Shape: (3, 3, 3, 3, 3, 3)


Scalar: One number
(e.g. heart rate)
Vector: a row of numbers
(e.g. heart rate sequence)
Matrix: a rectangle of numbers
(e.g. MRI image, grayscale images)
Tensor: a cube of numbers
(e.g. RGB color image)




```
#스칼라 또는 0D 텐서: 랭크가 0인 텐서, 즉, 숫자 하나로 이루어진 텐서를 의미
import numpy as np
zero_d_tensor = np.array(4)
zero_d_tensor
```

```
array(4)
```

```
#벡터 또는 1D 텐서: 랭크가 1인 텐서로, 숫자 여러 개가 일렬로 이어진 형태
one_d_tensor = np.array([1, 2, 3, 4])
one_d_tensor
```

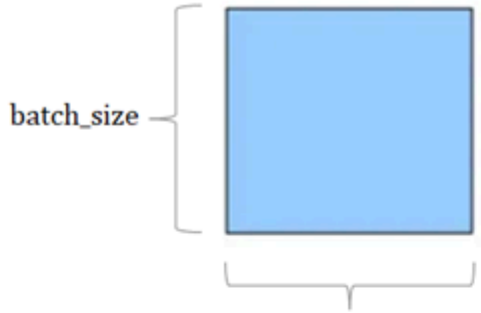
```
array([1, 2, 3, 4])
```

```
#행렬 또는 2D 텐서: 랭크가 2인 텐서로, 숫자가 행과 열로 이루어진 표 형태
two_d_tensor = np.array([[1, 2, 3, 4], [5, 6, 7, 8], [9, 10, 11, 12]])
two_d_tensor
```

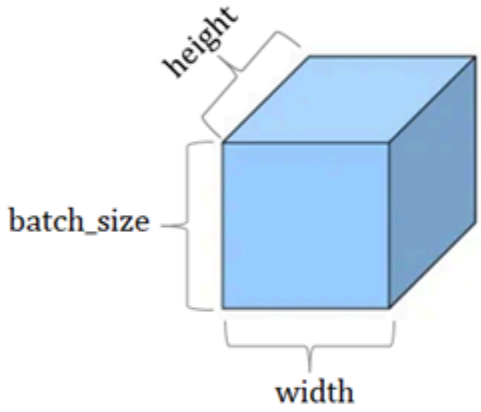
```
array([[ 1,  2,  3,  4],
       [ 5,  6,  7,  8],
       [ 9, 10, 11, 12]])
```

자주 쓰이는 텐서의 형태

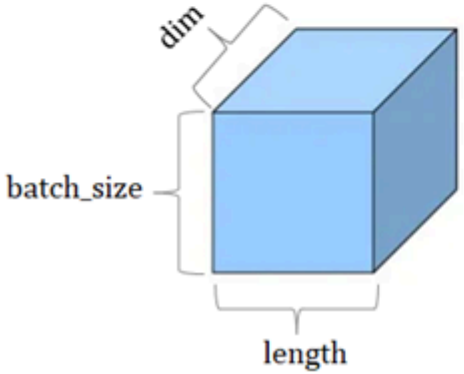
- 벡터 : 1차원 – (특성)
- 시퀀스 : 2차원 – (타임스텝, 특성)
- 이미지 : 3차원 – (높이, 너비, 채널)
- 동영상 : 4차원 – (프레임, 높이, 너비, 채널)



2D Tensor(Typical Simple Setting)



3D Tensor(Typical Computer Vision)** - 비전 분야에서의 3차원 텐서



3D Tensor(Typical Natural Language Processing)** - NLP 분야에서의 3차원 텐서

```
#아래와 같이 4개의 문장으로 구성된 전체 훈련 데이터가 있을때
#[['나는', '사과를', '좋아해'], ['나는', '바나나를', '좋아해'], ['나는', '사과를', '싫어해'], ['나는', '바나나를', '싫어해']]

```

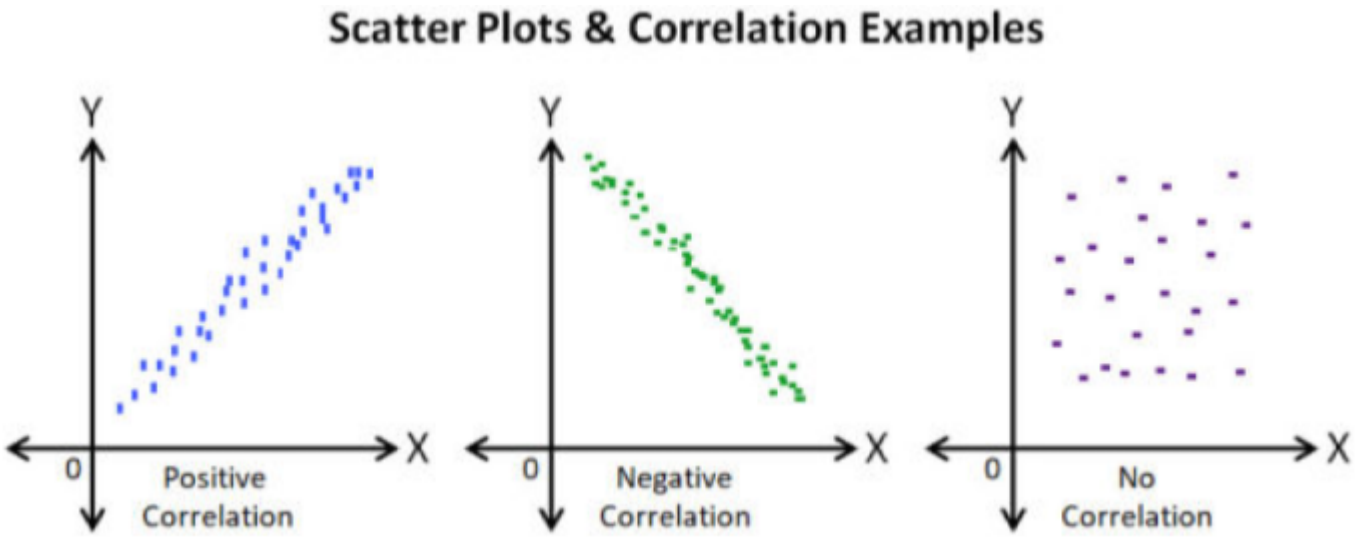
```
#컴퓨터는 아직 이 상태로는 '나는 사과를 좋아해'가 단어가 1개인지 3개인지 이해하지 못함
#우선 컴퓨터의 입력으로 사용하기 위해서는 단어별로 나눠주어야 함
#'나는' = [0.1, 0.2, 0.9]
#'사과를' = [0.3, 0.5, 0.1]
#'바나나를' = [0.3, 0.5, 0.2]
#'좋아해' = [0.7, 0.6, 0.5]

#위 기준을 따라서 훈련 데이터를 재구성하면 아래와 같음
#[[[0.1, 0.2, 0.9], [0.3, 0.5, 0.1], [0.7, 0.6, 0.5]],
# [[0.1, 0.2, 0.9], [0.3, 0.5, 0.2], [0.7, 0.6, 0.5]],
# [[0.1, 0.2, 0.9], [0.3, 0.5, 0.1], [0.5, 0.6, 0.7]],
# [[0.1, 0.2, 0.9], [0.3, 0.5, 0.2], [0.5, 0.6, 0.7]]]

#훈련 데이터는 4 × 3 × 3의 크기를 가지는 3D 텐서
#이제 batch size를 2로 하면~~~

#첫번째 배치 #1
[[[0.1, 0.2, 0.9], [0.3, 0.5, 0.1], [0.7, 0.6, 0.5]],
 [[0.1, 0.2, 0.9], [0.3, 0.5, 0.2], [0.7, 0.6, 0.5]]]

#두번째 배치 #2
[[[0.1, 0.2, 0.9], [0.3, 0.5, 0.1], [0.5, 0.6, 0.7]],
 [[0.1, 0.2, 0.9], [0.3, 0.5, 0.2], [0.5, 0.6, 0.7]]]

#컴퓨터는 배치 단위로 가져가서 연산을 수행
#현재 각 배치의 텐서의 크기는 (2 × 3 × 3)
#이는 (batch size, 문장 길이, 단어 벡터의 차원)의 크기임
```

```
[[[0.1, 0.2, 0.9], [0.3, 0.5, 0.1], [0.5, 0.6, 0.7]],
 [[0.1, 0.2, 0.9], [0.3, 0.5, 0.2], [0.5, 0.6, 0.7]]]
```

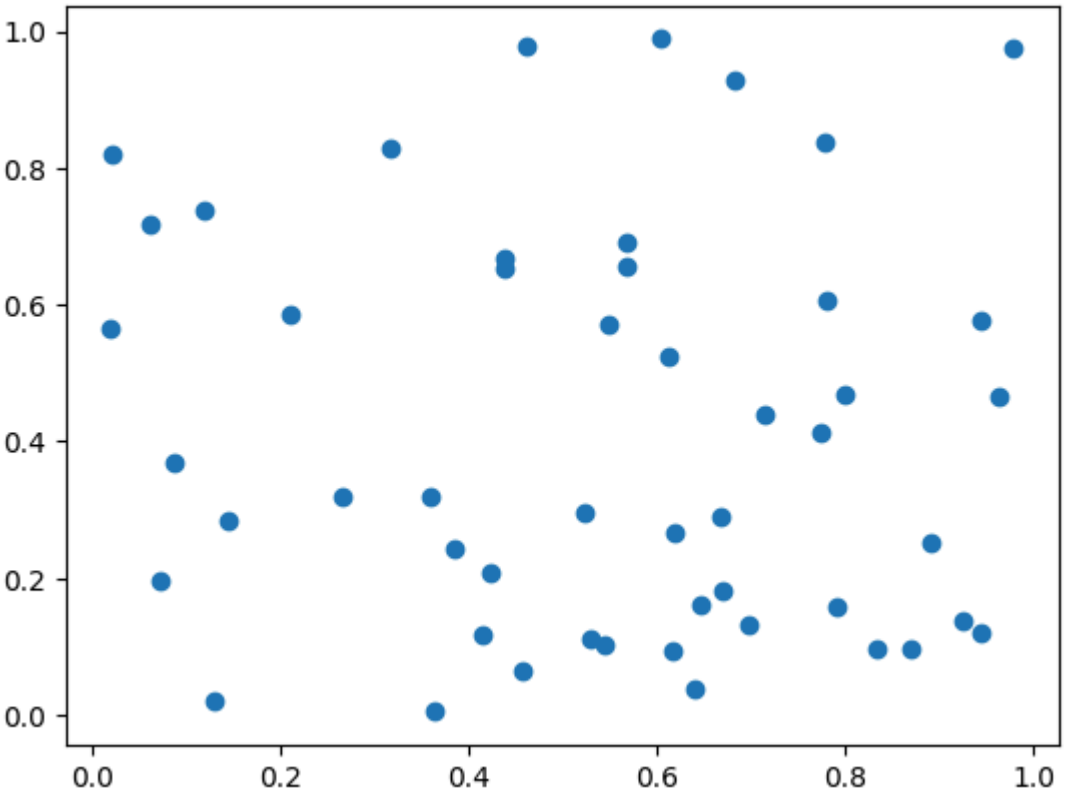
산점도(Scatter Plot)

1 코딩을 시작하거나 AI로 코드를 생성하세요.

- 변수가 2가지 이상일 때 두 변수의 연관성을 보여줌
- 각 자료의 변수 A의 값을 X값, 변수 B값을 Y값으로 정하여 점으로 표시
- 선형성(Linearity) - 두 변수의 연관성을 뜻하며, 양의 선형 관계, 음의 선형 관계가 있음




```
1 import matplotlib.pyplot as plt
2 import numpy as np
3 np.random.seed(0)
4 n = 50
5 x = np.random.rand(n)
6 y = np.random.rand(n)
7 plt.scatter(x, y)
8 plt.show()
```



분산과 표준편차

- 주어진 값들이 얼마나 흩어져 있는가를 나타내는 지표
    - 얼마나 흩어져 있느냐만 중요할 뿐 특정값과 가까운지 아닌지는(표적에서는 중앙에 맞았는지 안맞았는지) 분산/표준편차와는 무관함

흩어져 있음
분산 / 표준편차 ↑

모여 있음
분산 / 표준편차 ↓

- 집합 B를 보면 평균인 5를 중심으로 모여있고 집합 A는 B에 비해서 흩어져있는 걸 알 수 있음 --> 분산은 A가 더 큼

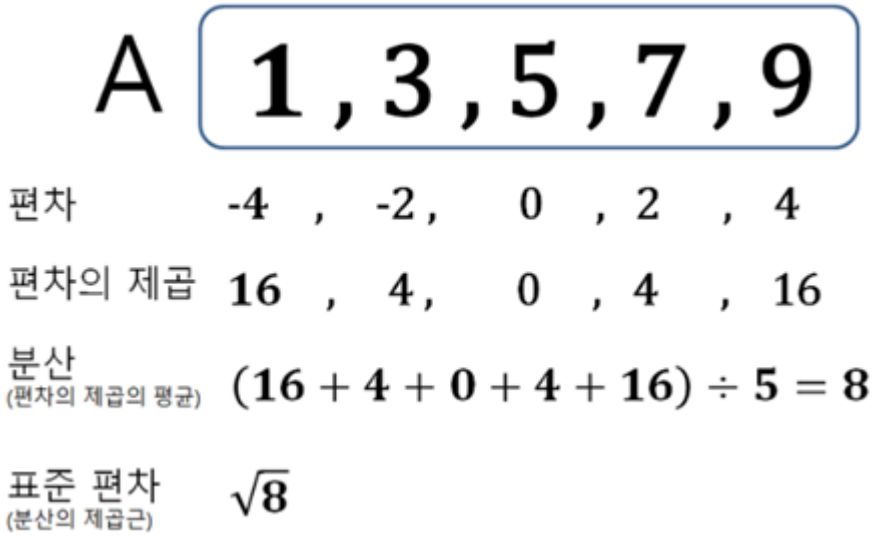
A 1,3,5,7,9 B 3,4,5,6,7

- 편차
  - 해당값에서 평균을 뺀 값, 즉 해당값이 평균에서 얼마나 떨어져있나를 보여줌
- 분산
  - 전체적으로 값들이 얼마나 떨어져있나를 나타내기 위해서 이 편차들의 평균을 구해보고 싶은데 그냥 구하면 + - 가 있으므로 0이 됨

- 따라서 부호에 따른 상쇄를 없애기위해 각 편차에 제곱을 해준뒤 그 값들의 평균을 구해줌

- 표준편차

  - 분산에 루트를 씌어준 값
  - 부호를 없애주기 위해서 제곱을 해준것을 보완하는 개념도 있고 실제로 분산값을 쓰다보면 값자체가 너무 커지기 때문에 표준편차라는 값을 만들었음



편차를 구하지 않고 분산을 구하기

$$V(X) = E(X^2) - \{E(X)\}^2$$

분산 = 제곱의 평균 − 평균의 제곱

A $1, 3, 5, 7, 9$

제곱 $1, 9, 25, 49, 81$

제곱의 평균 $(1 + 9 + 25 + 49 + 81) \div 5 = 33$

평균의 제곱 $\{(1 + 3 + 5 + 7 + 9) \div 5\}^2 = 25$

분산 $33 - 25 = 8$

A x_1, x_2, x_3, x_4, x_5

편차 $x_1 - m$ $x_2 - m$ $x_3 - m$ $x_4 - m$ $x_5 - m$

편차의 제곱 $(x_1-m)^2$ $(x_2-m)^2$ $(x_3-m)^2$ $(x_4-m)^2$ $(x_5-m)^2$

분산
(편차의 제곱의 평균)

$$\{(x_1-m)^2+(x_2-m)^2 + (x_3-m)^2 + (x_4-m)^2 + (x_5-m)^2\} \div 5$$

$$= (x_1^2 + x_2^2+x_3^2+x_4^2+x_5^2) \div 5$$ 제곱의 평균 $E(X^2)$

$$- 2m(x_1 + x_2 + x_3 + x_4 + x_5) \div 5 + 5m^2 \div 5$$

$(x_1 + x_2 + x_3 + x_4 + x_5) = 5m$

m^2 = 평균의 제곱 $\{E(X)\}^2$

zip() 함수

- 여러 개의 리스트의 동일한 위치에 있는 요소들을 묶어서 새로운 리스트를 생성

> - zip 함수는 파이썬의 내장 함수로, 여러 개의 반복 가능한(iterable) 객체를 인자로 받아서 동일한 인덱스의 요소를 튜플 형태로 묶어주는 역할을 함
> - '압축한다'라는 의미처럼, 두 개의 리스트를 하나로 묶어줌
> - 여러 개의 리스트의 동일한 위치에 있는 요소들을 묶어서 새로운 리스트를 만들어냄

```
# zip 함수 사용 예시
list1 = [1, 2, 3]
list2 = ['a', 'b', 'c']
zipped = zip(list1, list2)
print(list(zipped))  # [(1, 'a'), (2, 'b'), (3, 'c')]

for number, upper, lower in zip("12345", "ABCDE", "abcde"):
    print(number, upper, lower)
```

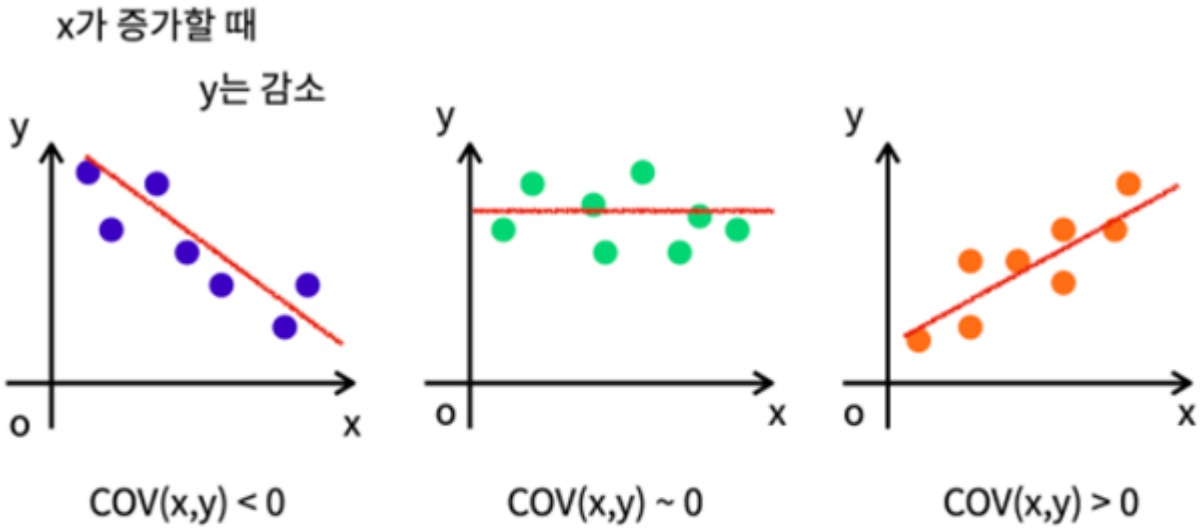
```
[(1, 'a'), (2, 'b'), (3, 'c')]
1 A a
2 B b
3 C c
4 D d
5 E e
```

공분산

- X라는 변수가 변화할 때 Y라는 변수가 어떤 방향성을 가지고 변화하는지를 나타내는 척도

  - 어떤 스칼라(scalar)인 두 확률변수 X, Y가 있을 때, 두 변수 사이에 어떤 상관관계가 있는지를 살펴보기 위해 공분산을 이용
  - X가 증가할 때 Y가 증가하는지 아니면 감소하는지를 나타냄

    - 공분산이 0보다 크다는 것은, 확률변수 X의 값이 커질 때, Y의 값도 커지는 경향이 있음을 의미하며
    - 공분산이 0보다 작으면 확률변수 X의 값이 커질 때, Y의 값이 작아지는 경향이 있다는 의미임
    - 만약, 공분산이 0이라면, 두 변수 사이에는 아무런 상관관계가 없다(uncorrelated)는 뜻임



변수의 선형 관계에 따른 공분산의 범위

$$E[(X-\mu_X)(Y-\mu_Y)]$$

- 공분산 Cov(X,Y)은 X의 편차인 X-E[X]와 Y의 편차인 Y-E[Y]를 곱한 값 들의 기대값임

- 기댓값의 성질(선형성, 분배 및 결합법칙 적용 가능)을 이용하여 전개하면 마지막 식과 같이 변형될 수도 있음

$$
\begin{aligned}
Cov(X, Y) &= E[(X - E[X])(Y - E[Y])] \\
&= E[XY - XE[Y] - YE[X] + E[X]E[Y]] \\
&= E[XY] - E[X]E[Y] - E[Y]E[X] + E[X]E[Y] \\
\therefore Cov(X, Y) &= E[XY] - E[X]E[Y]
\end{aligned}
$$

공분산 예

- A반의 학생들의 키를 X, 몸무게를 Y라고 하면, 일반적으로 두 변수 사이에는 양의 상관관계가 있을 가능성이 높음
  - X, Y는 이산 확률 변수
  - X의 기댓값 E(X)는 (1/5)*(175+165+163+170+161) = 166.8
  - Y의 기댓값 E(Y)는 (1/5)*(70+53+50+65+47) = 57

구 분	재원	지혜	민주	원준	윤지	기댓값, E()
키 (X)	175	165	163	170	161	166.8
몸무게 (Y)	70	53	50	65	47	57
X-E(X)	8.2	-1.8	-3.8	3.2	-5.8	-
Y-E(Y)	13	-4	-7	8	-10	-
(X-E(X))(Y-E(Y))	106.6	7.2	26.6	25.6	58	44.8

계산 결과, 공분산이 0보다 크기 때문에 이 경우에는 키가 클수록 몸무게가 많이 나가는 상관관계가 있다고 볼 수 있음

공분산 행렬(covariance matrix)

- 기하학적 의미
  - 행렬 = 선형변환, 벡터 공간을 다른 벡터 공간으로 mapping
- 데이터 구조적 의미
  - 각 feature의 변동이 얼마나 닮았는지 비교 ⇒ 분산 계산
  - 개수의 크기가 커지면 값이 커지므로 이를 방지하기 위해 n으로 나눔
  - 모든 변수에 대하여 분산과 공분산 값을 나타내는 정사각 행렬
  - 주 대각선 성분은 자기 자신의 분산 값을 나타냄
  - 주 대각선 이외의 성분은 가능한 두 변수의 공분산 값을 나타냄
  - [df.cov()] 또는 [np.cov()]를 사용하여 구할 수 있음

```
import numpy as np

h = np.array([170, 188, 165, 176, 160, 181, 178])
w = np.array([65, 82, 58, 68, 50, 71, 70])

cov_hw = np.cov(h, w, ddof = 1)

# ddof = 1이면 공분산을 계산할 때 (n-1)로 계산하라는 의미.
# 표본을 통해 모집단의 통계량을 추정할 때, 편향되지 않은 값을 얻기위해 보통 (n-1)로 계산한다.
# ddof = 0이면 단순 통계치(n)
# 기본값은 1이다.
print(cov_hw)
```

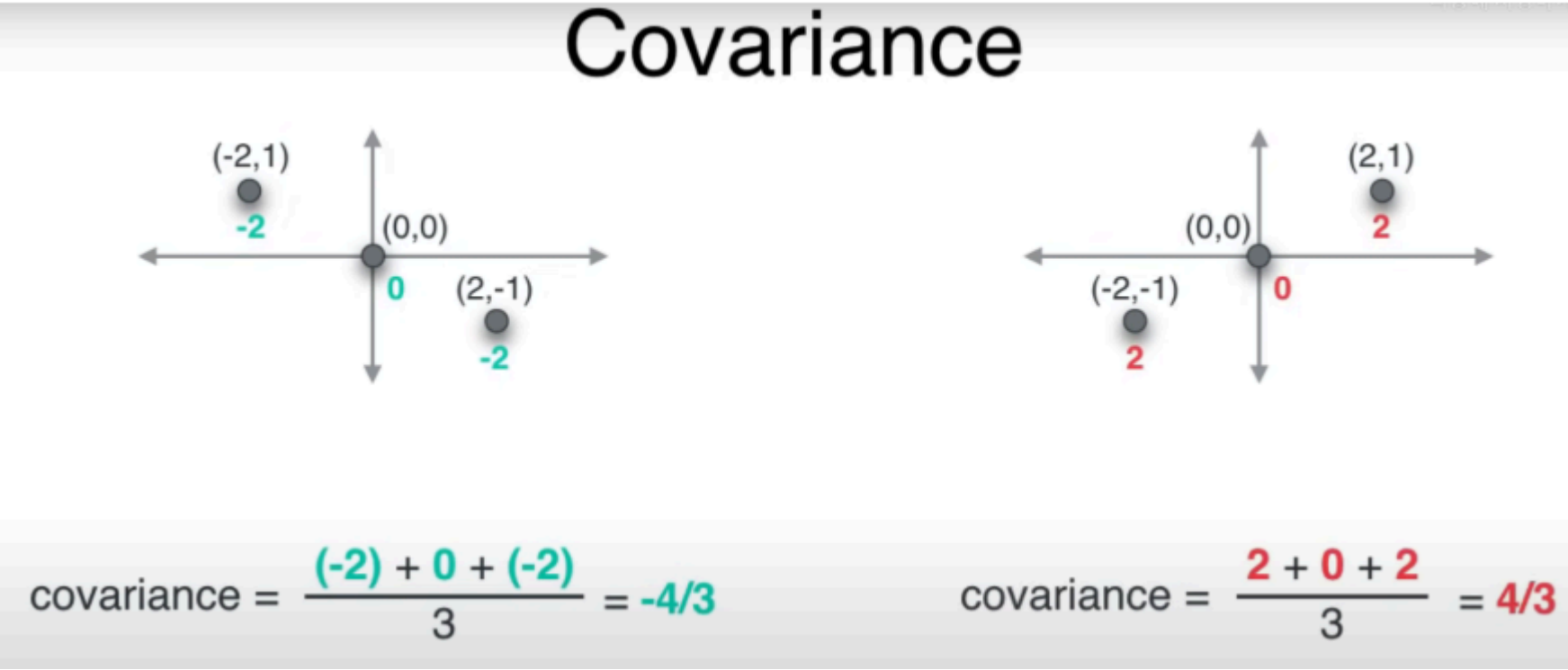
```
#여기서 대각선 방향의 값은 키와 몸무게 각각의 분산이 된다.
#만약, 키와 몸무게가 서로 독립적인 관계라면 공분산의 값은 0이 된다. (역은 성립하지 않음!)
#그런데 공분산은 두 변수 상관관계의 강한 정도를 의미하지는 않는다.
#공분산의 값이 변수 Scale에 따라 달라질 수 있기 때문이다.
#따라서 이를 보완하여 Scale을 맞춰준 것이 상관계수 (Correlation Coefficient)다.

import numpy as np
import pandas as pd

hw = pd.DataFrame(np.array([[170, 188, 165, 176, 160, 181, 178],
                            [65, 82, 58, 68, 50, 71, 70]]), index = ['키', '체중']).T
print('키 평균=', np.mean([170, 188, 165, 176, 160, 181, 178]))
print('체중 평균=', np.mean([65, 82, 58, 68, 50, 71, 70]))

#분산이란?
#데이터가 흩어진 정도로, 각 값들의 평균과의 차이(=편차)의 제곱의 평균이다.
print('키 분산=', np.var([170, 188, 165, 176, 160, 181, 178]))
print('체중 분산=', np.var([65, 82, 58, 68, 50, 71, 70]))

#표준편차는 분산에 루트를 씌운 것이다. 왜?
#분산 계산할 때 제곱을 하기 때문에, 데이터의 스케일에 따라서 같은 퍼짐이어도 분산 값이 크게 차이가 날 수도 있기 때문이다.
#이 스케일의 차이를 줄여주기 위해서 루트를 씌워서 보려는 것이다.
print('키 표준편차=', np.std([170, 188, 165, 176, 160, 181, 178]))
```

```
print('체중 표준편차=', np.std([65, 82, 58, 68, 50, 71, 70]))

cov_hw = hw.cov()
#키와 체중의 공분산이 96.5, 키와 키의 공분산이 03.00이라고 하면 된다. (키와 키의 공분산이라 하면 그냥 키의 분산이다)
print(cov_hw)
```

```
키 평균= 174.0
체중 평균= 66.28571428571429
키 분산= 79.71428571428571
체중 분산= 88.77551020408164
키 표준편차= 8.928285709714139
체중 표준편차= 9.422075684480657
        키          체중
키    93.0   96.500000
체중   96.5  103.571429
```

$$\mathrm{Cov}(x,y) = \begin{bmatrix} \sigma_x^2 & \mathrm{Cov}(x,y) \\ \mathrm{Cov}(x,y) & \sigma_y^2 \end{bmatrix}$$


x축 방향으로 퍼진 정도
x, y축 방향으로
함께 퍼진 정도
3
2
2
4
x, y축 방향으로
함께 퍼진 정도
y축 방향으로 퍼진 정도


Covariance
(-2,1)
-2
(0,0)
0
(2,-1)
-2
(2,1)
2
(0,0)
0
(-2,-1)
2
covariance = (-2) + 0 + (-2) / 3 = -4/3
covariance = 2 + 0 + 2 / 3 = 4/3

공분산 개념 알아보기

예제 데이터) 영화 분석 결과

데이터) 각 영화 별 싸움 횟수와 키스 횟수

영화 제목	the number of Kick	The number os Kiss	Type
냉정과열정사이	3	104	Romance
바람과함께사라지다	2	100	Romance
아름다운 여인	1	81	Romance
인정사정볼것없다	101	10	Action
놈놈놈	99	5	Action
화랑	98	2	Action
뜨거운 가슴으로	25	87	?

```
%matplotlib inline
import numpy as np
import matplotlib.pyplot as plt
# 0번째 열 : 킥 횟수, 1번째 열 : 키스 횟수
dataset = np.array([
    [3, 104],
    [2, 100],
    [1, 81],
    [101, 10],
    [99, 5],
    [98, 2],
])
labels = np.array(['Romance','Romance','Romance'
          'Action','Action','Action'])

inX = np.array([25, 87]) # 새로운 데이터

```

```
print(dataset)
print(labels)
```

```
[[  3 104]
 [  2 100]
 [  1  81]
 [101  10]
 [ 99   5]
 [ 98   2]]
['Romance' 'Romance' 'Romance' 'Action' 'Action' 'Action']
```

```
def desclibing(dataset, inX):
    plt.title("The Category of Movie")
    plt.scatter(dataset[:3,0],dataset[:3,1],label='Romance',
                c='g')
    plt.scatter(dataset[3:,0],dataset[3:,1],label='Action',
                c='r')
    plt.scatter(inX[0],inX[1],label="new_data",
```