

data - 숫자화 될 수 있는 정보

데이터의 종류

소리 데이터

이미지 영상 데이터

텍스트 데이터

숫자 데이터

직사각형의 형태로 숫자를 행과 열로 배열 - 이것이 그림이 될 수 있음 - 행열

숫자들을 왜 이렇게 일자로 펼쳐 놓을까 - 숫자의 나열은 벡터 모든 데이터는 벡터의 형태로 되어야 함 행열은 그래서 벡터가 아님 모든 데이터는 벡터의 상태여야 다루기가 쉬움

그 행열이 3층으로 되어 있으면 칼라 1층이면 흑백

영상은 2차원 칼라 이미지는 3차원 시간까지 있으면 4차원

어떻게 소리가 벡터화가 될까

우리의 소리는 wave form으로 이루어져있고 그 웨이브 폼 확대해보면 하나하나의 값들 그 값들을 벡터화 시킬 수 있음

텍스트는 어떻게 벡터화가 될까

0과 1로 벡터화 시킬 수 있음

모든 데이터는 벡터화 시킬 수 있음

제일 중요한 library중에 하나가 numpy: 벡터라이즈된 데이터들은 더하기 곱하기 같은 수학적 계산을 해야함 하지만 그냥 처리해버리면 list로만 처리되고 더하기 곱하기 안됨 그것을 도와주는게 numpy

array 리스트를 계산 가능한 행열 혹은 벡터로 바꿔주는 것 이 전에 import numpy 무조건 먼저 해줘야 함 import numpy as np 라고 해주면 numpy를 그 이후로 np로 바꿀 수 있음 뭐든 가능 n으로 바꿀수도 있음

X.shape 은 차원을 말해주는 것임

import 이제부터 해야함

기본적인 함수들은 그냥 사용했지만 중요한 함수들은 그 함수들이 포함되어 있는 library를 불러와야 함

numpy 가 제일 상위 개념

함수를 불러오는 법 `numpy.A.D.f(함수)` 이런식임 A안에 D안에 f라는 함수를 불러온다
(점)은 함수간의 포함 관계를 말해주는 것

```
import matplotlib.pyplot as plt
#똑같이 쓸 수 있는 것이 아닌 것은? 이런식 시험
from matplotlib import pyplot as plt
```

numpy를 왜 필요한가
list와 아주 비슷하지만 수학적으로 계산을 할 수 있기 때문
앞으로는 모두 numpy처리 해야함

matplotlib.pyplot 는 plot 하는 것과 관련된 것

empty도 numpy 제일 큰 library안에 있는 함수
`np.empty([2,3], dtype='int')` #이러한 갯수와 종류로 행열을 만들어주어라(랜덤)

`np.zeros([2,3])` #np library에 있는 zeros함수를 이용하는데 2 by 3 매트릭스를 만들어내라(0으로 채워진)

`np.array([[0, 0, 0],[0, 0, 0]])` # [0, 0, 0]는 그냥 벡터 우리가 이것을 2 by 3로 만들고 싶을 때는 [0, 0, 0] 추가 `[[0, 0, 0],[0, 0, 0]]`, 이것을 계산이 가능한 array로 만드는 방법이 `np.array()`

`np.ones([2,3])` 이렇게 하면 1., 찍힘 아마도 이러한 함수의 디폴트값이 float일 것임 그렇기 때문에 `dtype = int`로 해주면 점들이 없어짐

float도 종류가 여러가지가 있음 소수점 몇자리까지 해줄까 에 대한 정의가 float64 이런 것임 아주 정교한 일을 하고 싶을 때(아주 정교하지만 단점은 메모리를 많이 차지함)

아주 중요한 두 함수(`arange`, `linspace`)

`np.arange(5)` #5개의 index생성 for loop다음에 range썼던 것과 비슷 거기에 있었던 것들은 계산이 안되는 것 but 이 것들은 계산이 되는 것들

`np.arange(0,10)` #이런 식으로 range를 정해줄 수도 있음

`np.arange(0,10,2, dtype='float64')` #중요함 0부터 10까지 2의 간격으로 float~ :뒤에 것은 얼마나 정확도를 높이느냐

`np.linspace(0,10,6)` #0부터 10 포함해서 '6개' 로 나누어 준다

`X = np.array([[1, 2], [4, 5], [8, 9]])` #여기서 대괄호가 두개가 나오기때문에 2차원이다.. 3개면 3차원

Y.ndim #차원에 대해서

Y.shape

(2, 3, 2)

3x2(직사각형)이 2개가 있더라 제일 큰 것 부터 생각하면 됨 2개가 있다가 먼저

Y.dtype #타입에 대해서 알려줌

Y.astype(np.float64) 이렇게 바꿀 수 있음

np.zeros_like(Y) #형태를 유지한 채 모든 숫자를 0으로 바꿔주어라 Y*0 과 같음

유용하고도 중요한 것 plotting?

numpy 안 subpackage에 random 함수가 있음

data = np.random.normal(0, 1, 100)

만약 from numpy 를 썼다면 np쓸 필요 없음

from numpy import random.normal 이런 식

normal는 normal distribution의 약자(정규 분포 데이터를 만들어 주는 것 중 모양이 아니라 데이터!를 랜덤으로 만들어 주는 것 평균값과 표준편차 필요함) (0 - 평균, 1 - 표준편차, 100 - 데이터의 갯수)

plt.hist(data, bins=10)

plt.show()

#histogram bin이라는 옵션을 항상 달고 다니는데 바구니를 몇 개를 달고 다닐 것인가 x축위에 쌓여서 그래프를 그리는 것 그러한 그래프 y값에 해당하는 값은 항상 0을 포함한 자연수값이 나옴

저 값들을 다 합하면 100개 (시험!)

X = np.ones([2, 3, 4])

X

이렇게 행열을 만들어 낼 수 있음

이러한 행열을 reshape할 수 있음 몇 by 몇 by 몇이라는 shape을 바꿀 수 있음 but 그 안에 있는 element의 갯수의 총합은 같아야 바꿀 수 있음 -1은 알아서 숫자를 채워넣어라

Y = X.reshape(-1, 3, 2)

Y

np.allclose(X.reshape(-1, 3, 2), Y) #완전히 일치하느냐 하는 함수 allclose

random안에 있는 유용한 함수들 중에 방금 배웠던 normal함수 말고도 다른 것 많음 그 중 하나 randint

a = np.random.randint(0, 10, [2, 3]) #0부터 10까지 숫자 중에 뽑아서 2 by 3 행열을 만들어라

```
b = np.random.random([2,3]) #말그대로 랜덤하게 만들어내라  
np.savez("test", a, b) #파일을 저장해주는 함수 a와b라는 variable을 파일로 저장해주는 함수
```

```
!ls -al test* #저장이 잘 되었는지 확인해주는 함수
```

```
del a, b #메모리에서 variables 지우고 싶을 때
```

```
who는 어떤 variables들이 지금 available한지
```

```
npzfiles = np.load("test.npz") #저장한 값들을 불러오고  
npzfiles.files
```

```
npzfiles['arr_0'] #그 파일 이름에서 불러올 것 적으면 값이 불러져 나옴
```

```
data = np.loadtxt("regression.csv", delimiter=",", skiprows=1, dtype=('names':("X", "Y"), 'formats':('f','f'))) #csv는  
콤마로 나뉘어진 values들 이라는 파일 타입 엑셀로도 불러오기 할 수 있음 데이터를 파일로 주는 방법 중 하나
```

```
np.savetxt("regression_saved.csv", data, delimiter=",") #그 데이터를 주고싶거나 저장하고 싶을 때  
!ls -al regression_saved.csv
```

```
print(arr.size) #element의 총 갯수
```