NumPy 覚え書き

緑川章一

1　NumPy とは？

Python で数値計算行うためのライブラリである。NumPy は【Numerical Python】を縮めて作られた言葉である。読み方は、/ńʌmpaɪ/ (NUM-py) 、または、/ńʌmpi/ (NUM-pee) である。

2　インポートとバージョンの確認

```
>>> import numpy as np
>>> print(np.__version__)
1.19.1
```

3　数の表現

3.1　無理数

平方根は、sqrt()。

【例 1】

```
>>> from numpy import *
>>> sqrt(2)
1.4142135623730951
>>> pi
3.141592653589793
>>> e
2.718281828459045
```

【例 2】

```
>>> import numpy as np
>>> np.sqrt(2)
1.4142135623730951
>>> np.pi
3.141592653589793
>>> np.e
2.718281828459045
```

3.2 複素数

Physon では、複素数が定義されている。虚数単位は、i ではなく j で表す。

```
>>> a=1+2j
>>> b=3-1j
>>> a+b
(4+1j)
>>> a*b
(5+5j)
```

ついでに、$e^{\pi i}$ を計算する。

```
>>> np.e**(np.pi*1j)
(-1+1.2246467991473532e-16j)
```

4 ベクトル：すべては配列 array で

ベクトルは 1 次元配列で表す。
二つのベクトル、

$$\boldsymbol{v}_1 = \begin{pmatrix} 1 \\ 2 \\ 3 \end{pmatrix}, \qquad \boldsymbol{v}_1 = \begin{pmatrix} 5 \\ 3 \\ 1 \end{pmatrix},$$

を array を用いて表現すると、

```
>>> v1 = np.array([1,2,3])
>>> v2 = np.array([5,3,1])
>>> print("v1 =", v1)
v1 = [1 2 3]
>>> print("v2 =", v2)
v2 = [5 3 1]
```

ベクトルの定数倍

```
>>> print("2 v1 =", 2*v1)
2 v1 = [2 4 6]
```

ベクトルの足し算、内積 (dot product)、外積 (cross product) は、

```
>>> print("v1+v2 =", v1+v2)
v1+v2 = [6 5 4]
>>> print("v1・v2 =", np.dot(v1,v2))
v1・v2 = 14
>>> print("v1× v2 =", np.cross(v1,v2))
v1× v2 = [-7 14 -7]
```

一方、$*$ は、各成分の積を表す。

$$\boldsymbol{v}_1 * \boldsymbol{v}_2 = \begin{pmatrix} 1 \\ 2 \\ 3 \end{pmatrix} * \begin{pmatrix} 5 \\ 3 \\ 1 \end{pmatrix} = \begin{pmatrix} 1 \times 5 \\ 2 \times 3 \\ 3 \times 1 \end{pmatrix} = \begin{pmatrix} 5 \\ 6 \\ 3 \end{pmatrix}$$

```
>>> print("v1*v2 =", v1*v2)
v1*v2 = [5 6 3]
```

2 次元ベクトルの場合にも、内積 (dot product) だけでなく、外積 (cross product) も定義されている。

```
>>> v1=np.array([2,1])
>>> v2=np.array([-1,1])
>>> print("v1・v2 =", np.dot(v1,v2))
v1・v2 = -1
>>> print("v1× v2 =", np.cross(v1, v2))
v1× v2 = 3
```

2 次元の場合、

$$\boldsymbol{a} = \begin{pmatrix} a_1 \\ a_2 \end{pmatrix}, \qquad \boldsymbol{b} = \begin{pmatrix} b_1 \\ b_2 \end{pmatrix}$$

とすると、外積は、

$$\boldsymbol{a} \times \boldsymbol{b} = a_1 b_2 - a_2 b_1$$

である。

5　行列とその演算

5.1　行列の表現

例えば、2 行 2 列の行列

$$M_1 = \begin{pmatrix} 1 & 2 \\ 3 & 4 \end{pmatrix}$$

は、配列を用いて次のように表現される。

```
>>> M1 = np.array([[1,2],[3,4]])
>>> print("M1 = \n", M1)
M1 =
 [[1 2]
 [3 4]]
```

3 行 3 列の行列、例えば、

$$M = \begin{pmatrix} 1 & 2 & 3 \\ 4 & 5 & 6 \\ 7 & 8 & 9 \end{pmatrix}$$

の場合には、

```
>>> M = np.array([[1,2,3],[4,5,6],[7,8,9]])
>>> print("M = \n", M)
M =
 [[1 2 3]
 [4 5 6]
 [7 8 9]]
```

5.2　行列の積

行列 $A = [a_{ij}]$ と $B = [b_{ij}]$ の積が $C = [c_{ij}]$ となるとき、

$$C = AB$$

と表す。ただし、

$$c_{ij} = \sum_k a_{ik} b_{kj}$$

である。この演算は、dot(A,B) で表す。
【例】　パウリのスピン行列

$$s_x = \begin{pmatrix} 0 & 1 \\ 1 & 0 \end{pmatrix}, \quad s_y = \begin{pmatrix} 0 & -i \\ i & 0 \end{pmatrix}, \quad s_z = \begin{pmatrix} 1 & 0 \\ 0 & -1 \end{pmatrix}$$

の積の計算を行う。今後は、スクリプトファイルを作成して実行する。

プログラム

```
import numpy as np

s_x = np.array([[0,1],[1,0]])
s_y = np.array([[0,-1j],[1j,0]])
s_z = np.array([[1,0],[0,-1]])

print("s_x s_y = \n", np.dot(s_x, s_y))
print("s_y s_z = \n", np.dot(s_y, s_z))
print("s_z s_x = \n", np.dot(s_z, s_x))
```

実行結果

```
s_x s_y =
 [[0.+1.j 0.+0.j]
 [0.+0.j 0.-1.j]]
s_y s_z =
 [[0.+0.j 0.+1.j]
 [0.+1.j 0.+0.j]]
s_z s_x =
 [[ 0  1]
 [-1  0]]
```

多少分かりにくいが、$s_x s_y = is_z, \quad s_y s_z = is_x, \quad s_z s_x = is_y$ となっていることが分かる。

当然のことながら、行列の積は、掛け算の順序により結果が異なる。すなわち、一般には、

$$AB \neq BA$$

である。

【例】$s_x s_y + s_y s_x = 0, \quad s_y s_z + s_z s_y = 0, \quad s_z s_x + s_x s_z = 0$ を確かめてみよう。

前述のプログラムに続けて、

プログラム

```
print("s_x s_y + s_y s_x= \n", np.dot(s_x, s_y)+np.dot(s_y, s_x))
print("s_y s_z + s_z s_y= \n", np.dot(s_y, s_z)+np.dot(s_z, s_y))
print("s_z s_x + s_x s_z= \n", np.dot(s_z, s_x)+np.dot(s_x, s_z))
```

と書いて実行すると、

結果

```
s_x s_y + s_y s_x=
 [[0.+0.j 0.+0.j]
 [0.+0.j 0.+0.j]]
s_y s_z + s_z s_y=
 [[0.+0.j 0.+0.j]
 [0.+0.j 0.+0.j]]
s_z s_x + s_x s_z=
 [[0 0]
 [0 0]]
```

を得る。

一方、$+$ は各成分毎の和を、$*$ 積は各成分毎の積を表す。

$$A+B = \begin{pmatrix} a_{11} & a_{12} & \cdots & a_{1n} \\ a_{21} & a_{22} & \cdots & a_{2n} \\ \vdots & \vdots & \ddots & \vdots \\ a_{n1} & a_{n2} & \cdots & a_{nn} \end{pmatrix} + \begin{pmatrix} b_{11} & b_{12} & \cdots & b_{1n} \\ b_{21} & b_{22} & \cdots & b_{2n} \\ \vdots & \vdots & \ddots & \vdots \\ b_{n1} & b_{n2} & \cdots & b_{nn} \end{pmatrix} = \begin{pmatrix} a_{11}+b_{11} & a_{12}+b_{12} & \cdots & a_{1n}+b_{1n} \\ a_{21}+b_{21} & a_{22}+b_{22} & \cdots & a_{2n}+b_{21} \\ \vdots & \vdots & \ddots & \vdots \\ a_{n1}+b_{n1} & a_{n2}+b_{n1} & \cdots & a_{nn}+b_{nn} \end{pmatrix}$$

$$A*B = \begin{pmatrix} a_{11} & a_{12} & \cdots & a_{1n} \\ a_{21} & a_{22} & \cdots & a_{2n} \\ \vdots & \vdots & \ddots & \vdots \\ a_{n1} & a_{n2} & \cdots & a_{nn} \end{pmatrix} * \begin{pmatrix} b_{11} & b_{12} & \cdots & b_{1n} \\ b_{21} & b_{22} & \cdots & b_{2n} \\ \vdots & \vdots & \ddots & \vdots \\ b_{n1} & b_{n2} & \cdots & b_{nn} \end{pmatrix} = \begin{pmatrix} a_{11}b_{11} & a_{12}b_{12} & \cdots & a_{1n}b_{1n} \\ a_{21}b_{21} & a_{22}b_{22} & \cdots & a_{2n}b_{21} \\ \vdots & \vdots & \ddots & \vdots \\ a_{n1}b_{n1} & a_{n2}b_{n1} & \cdots & a_{nn}b_{nn} \end{pmatrix}$$

【例】

$$A = \begin{pmatrix} 0 & 2 & 1 \\ 4 & -1 & 3 \\ 7 & 6 & 5 \end{pmatrix}, \quad B = \begin{pmatrix} 1 & 2 & 3 \\ 4 & 5 & 6 \\ 7 & 8 & 9 \end{pmatrix} \quad \text{の場合}$$

プログラム

```
import numpy as np

A = np.array([[0,2,1],[4,-1,3],[7,6,5]])
B = np.array([[1,2,3],[4,5,6],[7,8,9]])

print("A+B =\n", A+B)
print("A*B =\n", A*B)
```

結果

```
A+B =
 [[ 1  4  4]
 [ 8  4  9]
 [14 14 14]]
A*B =
 [[ 0  4  3]
 [16 -5 18]
 [49 48 45]]
```

5.3　行列とベクトルの積

行列 $A = [a_{ij}]$ とベクトル $\boldsymbol{v} = [v_i]$ との積、$A\boldsymbol{v}$

$$A\boldsymbol{v} = \begin{pmatrix} a_{11} & a_{12} & \cdots & a_{1n} \\ a_{21} & a_{22} & \cdots & a_{2n} \\ \vdots & \vdots & \ddots & \vdots \\ a_{n1} & a_{n2} & \cdots & a_{nn} \end{pmatrix} \begin{pmatrix} v_1 \\ v_2 \\ \vdots \\ v_n \end{pmatrix} = \begin{pmatrix} a_{11}v_1 + a_{12}v_2 + \cdots + a_{1n}v_n \\ a_{21}v_1 + a_{22}v_2 + \cdots + a_{2n}v_n \\ \vdots \\ a_{n1}v_1 + a_{n2}v_2 + \cdots + a_{nn}v_n \end{pmatrix}$$

もまた、dot(A, v) で与えられる。

一方、dot(v, A) は、

$$\begin{aligned} \boldsymbol{v}^T A &= \begin{pmatrix} v_1, & , v_2, & \cdots, & v_n \end{pmatrix} \begin{pmatrix} a_{11} & a_{12} & \cdots & a_{1n} \\ a_{21} & a_{22} & \cdots & a_{2n} \\ \vdots & \vdots & \ddots & \vdots \\ a_{n1} & a_{n2} & \cdots & a_{nn} \end{pmatrix} \\ &= \begin{pmatrix} v_1a_{11} + v_2a_{21} + \cdots + v_na_{n1}, & v_1a_{12} + v_2a_{22} + \cdots + v_na_{n2}, & \cdots, & v_1a_{1n} + v_2a_{2n} + \cdots + v_na_{nn} \end{pmatrix} \end{aligned}$$

である。

また、$+$ は各成分の和を、$*$ 積は各成分の積を表す。

$$A + \boldsymbol{v} = \begin{pmatrix} a_{11} & a_{12} & \cdots & a_{1n} \\ a_{21} & a_{22} & \cdots & a_{2n} \\ \vdots & \vdots & \ddots & \vdots \\ a_{n1} & a_{n2} & \cdots & a_{nn} \end{pmatrix} + \begin{pmatrix} v_1 \\ v_2 \\ \vdots \\ v_n \end{pmatrix} = \begin{pmatrix} a_{11} + v_1 & a_{12} + v_2 & \cdots & a_{1n} + v_n \\ a_{21} + v_1 & a_{22} + v_2 & \cdots & a_{2n} + v_n \\ \vdots & \vdots & \ddots & \vdots \\ a_{n1} + v_1 & a_{n2} + v_2 & \cdots & a_{nn} + v_n \end{pmatrix}$$

$$A * \boldsymbol{v} = \begin{pmatrix} a_{11} & a_{12} & \cdots & a_{1n} \\ a_{21} & a_{22} & \cdots & a_{2n} \\ \vdots & \vdots & \ddots & \vdots \\ a_{n1} & a_{n2} & \cdots & a_{nn} \end{pmatrix} * \begin{pmatrix} v_1 \\ v_2 \\ \vdots \\ v_n \end{pmatrix} = \begin{pmatrix} a_{11}v_1 & a_{12}v_2 & \cdots & a_{1n}v_n \\ a_{21}v_1 & a_{22}v_2 & \cdots & a_{2n}v_n \\ \vdots & \vdots & \ddots & \vdots \\ a_{n1}v_1 & a_{n2}v_2 & \cdots & a_{nn}v_n \end{pmatrix}$$

【例】

$$A\boldsymbol{v} = \begin{pmatrix} 1 & 2 & 3 \\ 4 & 5 & 6 \\ 7 & 8 & 9 \end{pmatrix} \begin{pmatrix} 1 \\ 2 \\ -1 \end{pmatrix} = \begin{pmatrix} 2 \\ 8 \\ 14 \end{pmatrix}$$

$$\boldsymbol{v}^T A = \begin{pmatrix} 1, & 2, & -1 \end{pmatrix} \begin{pmatrix} 1 & 2 & 3 \\ 4 & 5 & 6 \\ 7 & 8 & 9 \end{pmatrix} = \begin{pmatrix} 2, & 4, & 6 \end{pmatrix}$$

$$A + \boldsymbol{v} = \begin{pmatrix} 1 & 2 & 3 \\ 4 & 5 & 6 \\ 7 & 8 & 9 \end{pmatrix} + \begin{pmatrix} 1 \\ 2 \\ -1 \end{pmatrix} = \begin{pmatrix} 2 & 4 & 2 \\ 5 & 7 & 5 \\ 8 & 10 & 8 \end{pmatrix}$$

$$A * \boldsymbol{v} = \begin{pmatrix} 1 & 2 & 3 \\ 4 & 5 & 6 \\ 7 & 8 & 9 \end{pmatrix} * \begin{pmatrix} 1 \\ 2 \\ -1 \end{pmatrix} = \begin{pmatrix} 1 & 4 & -3 \\ 4 & 10 & -6 \\ 7 & 16 & -9 \end{pmatrix}$$

プログラム

```
import numpy as np
A = np.array([[1,2,3],[4,5,6],[7,8,9]])
v = np.array([1,2,-1])
print("A v =\n", np.dot(A, v))
print("v^T A  =\n", np.dot(v, A))
print("A+v =\n", A+v)
print("A*v =\n", A*v)
```

結果

```
A v =
 [ 2  8 14]
v^T A  =
 [2 4 6]
A+v =
 [[ 2  4  2]
 [ 5  7  5]
 [ 8 10  8]]
A*v =
 [[ 1  4 -3]
 [ 4 10 -6]
 [ 7 16 -9]]
```

6　行や列の取り出し

プログラム

```
import numpy as np

A = np.array([[2, 3, 5],[2,4,6],[4,1,3],[1,3,5]])

print("A= \n", A, "\n")

print("i 行目の取り出し")

i=1
print("A[",i,"] = ", A[i], "\n")

print("i 列目の取り出し")
i=2
print("A[:,",i,"] = ", A[:, 2])
```

結果

```
A=
 [[2 3 5]
 [2 4 6]
 [4 1 3]
 [1 3 5]]

i 行目の取り出し
A[ 1 ] =  [2 4 6]

i 列目の取り出し
A[:, 2 ] =  [5 6 3 5]
```

7　numpy.array を pandas.Series，DataFrame に変換

プログラム

```
import numpy as np

A = np.array([[2, 3, 5],[2,4,6],[4,1,3],[1,3,5]])

from pandas import Series,DataFrame

A_df = DataFrame(A)

print("A_df =")
print(A_df,"\n")

#i 行目の抽出
i=2
print("A_df.loc[",i,"] = ")
print(A_df.loc[i],"\n")

#i 列目の抽出
print("A_df[",i,"] =")
print(A_df[i])
```

結果

```
A_df =
   0  1  2
0  2  3  5
1  2  4  6
2  4  1  3
3  1  3  5

A_df.loc[ 2 ] =
0    4
1    1
2    3
Name: 2, dtype: int32

A_df[ 2 ] =
0    5
1    6
2    3
3    5
Name: 2, dtype: int32
```