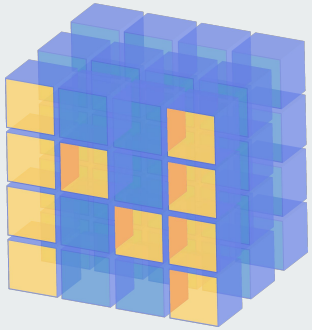
Python程式設計

科學運算的好幫手 - Numpy



官方網站: https://numpy.org/
教學網站: https://numpy.org/devdocs/user/quickstart.html

蘇維宗(Wei-Tsung Su)
suwt@au.ed.tw

歷史版本

版本	說明	日期	負責人
v1.0	初版	2020/05/17	蘇維宗

為何要使用Numpy?

對於經常進行數值分析、影像處理、或機器學習等科學計算的人應該都對於Matlab這套功能強大的軟體與程式語言並不陌生。

Numpy是Python中擁有與Matlab相似功能的套件，其特性主要包含

- 強大的多維度陣列(multi-dimensional array)的儲存與處理
- 可以整合C/C++與Fortran程式
- 具有線性代數、傅立葉轉換、與隨機變數等能力

為何不直接使用List?

藉由下面幾個矩陣運算來了解Numpy與List的不同

- 矩陣純量運算

$$2 * [1, 2, 3, 4] = [2, 4, 6, 8]$$

$$([1, 2, 3, 4])^2 = [1, 4, 9, 16]$$

- 矩陣乘積(dot product)

$$\begin{bmatrix} 1 & 2 \\ 3 & 4 \end{bmatrix} \cdot \begin{bmatrix} 1 & 2 \\ 3 & 4 \end{bmatrix} = \begin{bmatrix} 7 & 10 \\ 15 & 24 \end{bmatrix}$$

矩陣純量相乘

試著撰寫一個Python程式可以讓

```
>>> [1,2,3,4]
```

變成

```
>>> [2,4,6,8]
```

請完成下面的程式碼

```
1.  x = [1,2,3,4]
2.  # Let y = 2 * x
3.  print(y)
```

矩陣純量相乘(續)

我以為的...

```
1.  x = [1,2,3,4]
2.  # Let y = 2 * x
3.  y = 2 * x
4.  print(y)
5.  >>> [2,4,6,8]
```

我得到的...

```
1.  x = [1,2,3,4]
2.  # Let y = 2 * x
3.  y = 2 * x
4.  print(y)
5.  >>> [1,2,3,4,1,2,3,4]
```

矩陣純量平方

試著撰寫一個Python程式可以讓

```
>>> [1,2,3,4]
```

變成

```
>>> [1,4,9,16]
```

請完成下面的程式碼

```
1.  x = [1,2,3,4]
2.  # Let y = x^2
3.  print(y)
```

矩陣純量平方(續)

我以為的...

```
1.  x = [1,2,3,4]
2.  # Let y = x^2
3.  y = x ** 2
4.  print(y)
5.  >>> [1,4,9,16]
```

我得到的...

```
1.  x = [1,2,3,4]
2.  # Let y = x^2
3.  y = x ** 2
4.  >>> TypeError: ...
5.  print(y)
```

為何不直接使用List? (續)

試著撰寫一個Python程式可以進行矩陣乘積運算

$$\begin{bmatrix} 1 & 2 \\ 3 & 4 \end{bmatrix} \cdot \begin{bmatrix} 1 & 2 \\ 3 & 4 \end{bmatrix} = \begin{bmatrix} 7 & 10 \\ 15 & 24 \end{bmatrix}$$

結論: 直接用List來處理矩陣運算太不直覺了!!!

建立矩陣

https://numpy.org/devdocs/user/quickstart.html#array-creation

從List轉換

- List轉Numpy Array
  `numpy.array(List)`
- Numpy Array轉List
  `numpy.ndarray.tolist()`

範例程式

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

直接建立

- 建立初始值為0的矩陣
  numpy.zeros(shape)
- 建立初始值為1的矩陣
  numpy.ones(shape)
- 建立沒有初始值的矩陣
  numpy.empty(shape)
- ...

範例程式

```
1.  import numpy as np
2.  #建立2*3的矩陣，初始值為0
3.  X = np.zeros((2,3))
4.  print(X)
5.  #建立2*3的矩陣，初始值為1
6.  Y = np.ones((2,3)
7.  print(Y)
```

以內差法建立一維矩陣

- 從start到stop間隔step產生元素
  numpy.arange(start,stop,step)
- 從start到stop產生num個元素
  numpy.linspace(start,stop,num)

範例程式

```
1.  import numpy as np
2.  X = np.arange(0,10,2)
3.  print(X)
4.  Y = np.linspace(0,10,5)
5.  print(Y)
```

轉換維度

- 將多維矩陣轉成一維矩陣
  numpy.ndarray.ravel()
- 改變多維矩陣維度
  numpy.ndarray.reshape(shape)

範例程式

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

課堂練習

以Numpy撰寫程式進行矩陣運算

1. 在1到100之間產生一個1x400的一維矩陣
2. 將3所產生的矩陣，變形為2x200

基本運算

https://numpy.org/devdocs/user/quickstart.html#basic-operations

基本運算

Numpy的Array基本運算自帶有廣播能力(對矩陣內每個運算進行該運算)。

例如

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

課堂練習

以Numpy撰寫程式進行矩陣運算

1. 在1到100之間產生一個1x400的一維矩陣
2. 對1產生的一維矩陣所有元素進行以下函數運算

$$y = 2x^4 + 3x^2 - 5$$

索引、切割、與迭代

https://numpy.org/devdocs/user/quickstart.html#indexing-slicing-and-iterating

一維陣列索引(切割)

取得一維陣列的一部分(與List使用方式相同)

```
numpy.ndarray[start:stop:step]
```

範例程式

```
1.  import numpy as np
2.  X = np.array([1,2,3,4,5,6,7,8,9,10])
3.  Y = X[2:6:1]
4.  print(Y) # [3,4,5,6]
```

二維陣列索引(切割)

取得二維陣列的一部分

```
numpy.ndarray[start:stop:step,start:stop:step]
```

範例程式

```
1.  import numpy as np
2.  X = np.array([[1,2,3],[4,5,6],[7,8,9],[10,11,12]])
3.  Y = X[1:4,1:3]
4.  print(Y) # [[5,6],[8,9],[11,12]]
```

課堂練習

以Numpy撰寫程式取出框出的陣列

```
X =
[
    [ 1, 2, 3, 4, 5, 6],
    [ 7, 8, 9,10,11,12],
    [13,14,15,16,17,18],
    [19,20,21,22,23,24],
    [25,26,27,28,29,30],
    [31,32,33,34,35,36]
]
```

函數

https://numpy.org/devdocs/user/quickstart.html#universal-functions

常見的函數

Numpy支援了許多常見的函數，可對矩陣廣播運算

- 三角函數(`sin,cos,...`)
- 矩陣運算(`dot, inner, …`)
- 數學運算(`exp, sqrt, …`)
- …

二維矩陣相乘

二維矩陣相乘可以用dot()來完成，如下

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

Q & A

Computer History Museum, Mt. View, CA