Python 快速上手 : 模組介紹 numpy

2020/09/30

Modules

import module

We don’t have to start from the very beginning every time.

- Build-in module
- Used-defined module
- Using the specified version module
- Using the module form github (e.g. developer version)

The different way to import module (1/2)

Take the build-in os module as example, we are going to invoke the os.path.abspath function to get the absolute path.

```
>>> import os # import whole os module
>>> os.path.abspath('..')
```

```
>>> from os import * # import whole os module
>>> os.path.abspath('..')
```

The different way to import module (2/2)

Take the build-in os module as example, we are going to invoke the os.path.abspath function to get the absolute path.

```
>>> from os import path # import whole path submodule
>>> path.abspath('..')
```

```
>>> from os.path import abspath # import abspath function
>>> abspath('..')
```

Numpy

Numpy 資料型態

- Numpy 是很常被拿來處理多維資料的 Python library，透過基本資料型態 array，可以用來存放多維資料。
- e.g. 希望將 list 中的值全部加 2

```
>>> A = [1, 2, 3]
>>> new_A = [ x + 2 for x in A]
>>> print(new_A)
[3, 4, 5]
```

```
>>> import numpy as np
>>> A = [1, 2, 3]
>>> array_A = np.array(A)
>>> array_A = array_A + 2
>>> print(array_A)
array([3, 4, 5])
```

矩陣基本屬性 – 變數資料型態

- 我們可以透過 type 來檢查資料型態是否屬於 numpy array。

```
type(a) # <class ‘numpy.ndarray’>
```

- 有時候在 function 裡面的時候需要先檢查操作對象是否為 ndarray，可以透過isinstance 這個 build-in function。

```
isinstance(a, np.ndarray) # True
isinstance([[1, 2, 3], [4, 5, 6]], np.ndarray) # False
```

矩陣基本屬性 – 矩陣值資料型態

- numpy.ndarray 建立時可指定型態

```
>>> a = np.arange(10, dyype='int64')
```

- 矩陣裡的每一個值我們稱為 scalars，我們可以透過 dtype 檢查，通常在處理圖片或是神經網路的權重時會有影響。

```
>>> a.dtype # dtype('int64')
```

矩陣基本屬性 – 矩陣維度資訊

- 做矩陣運算之前，因為維度的特性，我們必須要先確認矩陣的一些屬性。
- e.g. 矩陣乘法要第一個矩陣的 column 個數 = 第二個矩陣的 row 個數才成立

$$(3, 2) \bullet (2, 1) = (3, 1)$$

矩陣基本屬性 – 矩陣維度資訊

- 透過 shape 我們可以得知每一維度的個數，進一步推算 row 與 column 個數

```
a.shape # (2, 3) => 2 rows, 3 columns
```

- 如果單純要知道矩陣的維度可以透過 ndim

```
a.ndim # 2 => 2 dimensions
```

矩陣類型

在矩陣操作的過程中有時候會用到特殊的矩陣：

- 空矩陣，可透過 np.zeros 或是 np.ones 宣告一個 0 或 1 的矩陣
- 對角矩陣，可透過 np.eye 來宣告一個對角線為 1 其餘為 0 的矩陣

```
>>> np.zeros((3,3))

>>> np.ones((3,3))

>>> np.eye(3)
```

axis 的概念

axis 指運算的方向。在二維矩陣裡有兩個方向：

沿列 (axis=0)，或是**沿行(axis=1)**的方向。

axis=0 

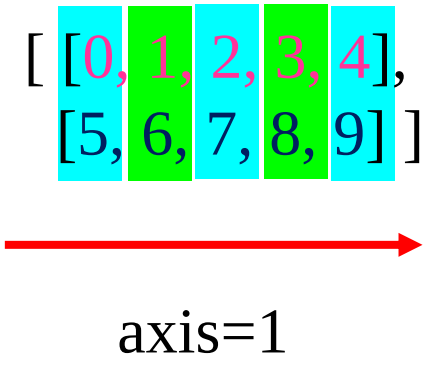
[[0, 1, 2, 3, 4],
[5, 6, 7, 8, 9]]

記得，類似矩陣的東西都是「先列後行」

axis 的概念

axis 指運算的方向。在二維矩陣裡有兩個方向：

沿列 (axis=0)，或是**沿行(axis=1)**的方向。

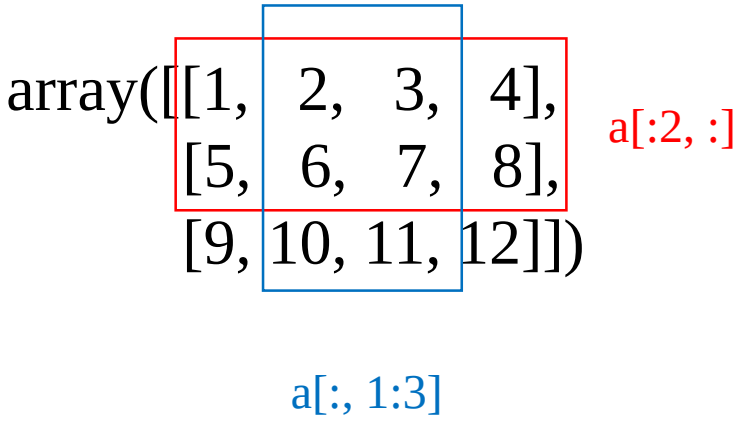


記得，類似矩陣的東西都是「先列後行」

矩陣索引 - slicing

- ndarray 跟 list 取出片段的方式一樣，可以透過 index 檢視部分矩陣資訊

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



矩陣索引 - integer indexing

- 前面提到的 slicing 通常用於觀察整個 row 或是 column 的連續片段，但是有時候我們只是想看某些 scalars，可以透過每個維度的 index 來觀察。

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

>>> a[[0, 1, 2], [0, 1, 0]]
array([1, 4, 5])
```

矩陣索引 – Boolean indexing

- integer indexing 必須給定確切的 index，但是索引時最常用的情境是給定條件下，找出符合的 scalars。
- ndarray 根據給定的條件會回傳一個 Boolean matrix，我們就可以根據

  boolean matrix 當作 mask 取出符合條件的 scalars。

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

矩陣索引

矩陣變形 – np.expand_dims

- 為了符合某些 function 的 input 需求，或是在處理圖片的過程中，我們有時候需要在不改變原始值的情況下改變維度。
- e.g. 灰階圖像 (grayscale) 是二維資料要轉換成與 RGB 圖像同維度的格式

  shape(200, 200) -> shape(200, 200, 1)

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

Numpy operation

矩陣四則運算 – Arithmetic operation

• 矩陣的四則運算透過 Numpy 內部機制，實做 element-wise。

```
a = np.array([[1, 2], [3, 4]], dtype=np.float64)
b = np.array([[5, 6], [7, 8]], dtype=np.float64)

print(a+b)
print(a-b)
print(a*b)
print(a/b)
```

基本線性代數

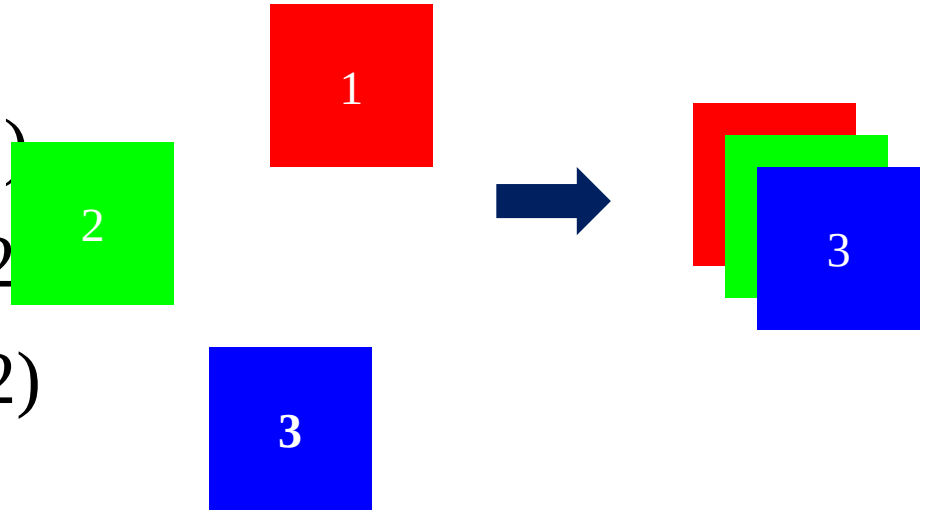
- 轉置矩陣： m*n 矩陣在向量空間上轉置為 n*m
- 逆矩陣：n*n 矩陣 A 存在一個 n*n 矩陣B，使得 AB = BA = I

```
>>> a = np.array([[0, 1], [2, 3]])
>>> print(a.T)

>>> inverse = np.linalg.inv(a)
>>> print(inverse)
>>> print(np.dot(a, inverse))
```

矩陣堆疊 - stack

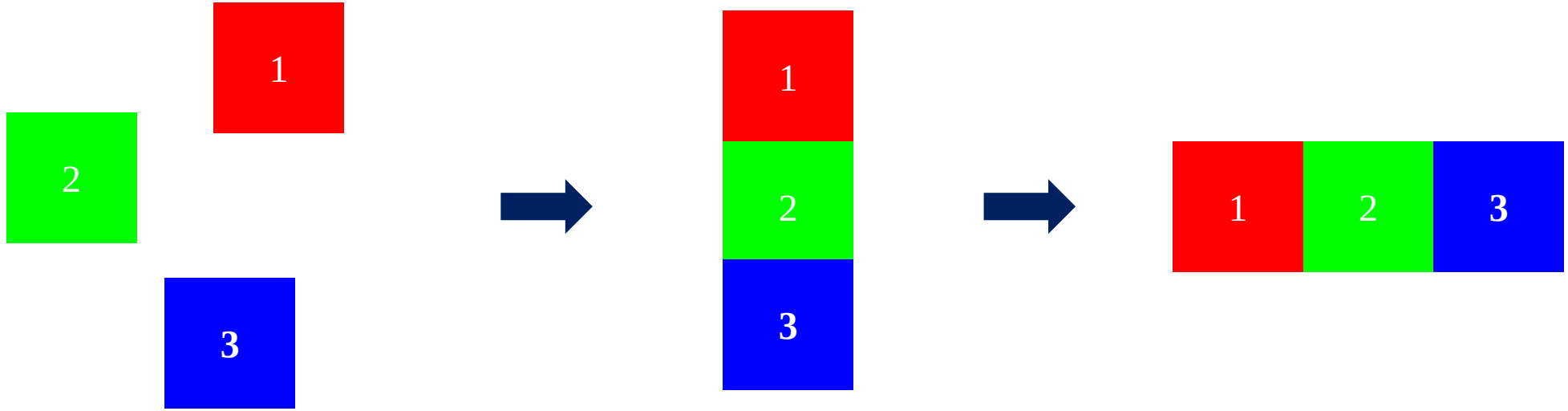
- 在實做深度學習網路或是圖片的時候，我們有時候需要對多個 array 做合併 np.stack 可以根據為度將多個 array 合併。

```
r = np.array([[0, 1], [2, 3]]) # shape = (2, 2)
g = np.array([[4, 5], [6, 7]]) # shape = (2, 2
b = np.array([[8, 9], [1, 3]]) # shape = (2, 2)

np.stack([r, g, b], axis = 0)
```



矩陣堆疊 – vstack & hstack

- 有時候我們只是希望把矩陣水平或垂直的合併
  - np.vstack((x, y))：vertical，垂直合併
  - np.hstack((x, y))：horizontal，水平合併



(2, 2) * 3

vstack
(6, 2)

hstack
(2, 6)