人工智慧概論與實務

Numpy

- 主要用於資料處理上
- 底層以C和Fortran實作，所以能快速地操作多重維度的資料陣列
- 具備平行處理的能力，可以將操作動作一次套用在大型陣列上
- 其它重要的資料科學套件（例如：Pandas），都是基於Numpy基礎上應用的

Numpy 和 Python List 的差別

Numpy array 如何使用?

```
In [1]: import numpy as np

        np.array([1,2,3])

Out[1]: array([1, 2, 3])
```

Numpy array 與List的共通點

```
In [5]: #list
        my_list = [1,2,3]
        print(my_list[0])

        1

In [6]: #numpy array
        my_array = np.array([1,2,3])
        print(my_array[0])

        1

In [7]: my_list[0] = -1
        my_array[0] = -1
        print(my_list)
        print(my_array)

        [-1, 2, 3]
        [-1  2  3]
```

所以這樣看起來numpy好像沒什麼特別?
為什麼大家都喜歡numpy

Numpy 的優勢

Numpy的核心優勢:就是快

用專業的語言描述的話，Numpy 喜歡用電腦記憶體中連續的一塊物理位址存儲資料，因為都是連號的嘛，找到前後的號，不用跑很遠，非常迅速。而 Python 的 List 並不是連續存儲的，它的資料是分散在不同的物理空間，連號的肯定比不連號的算起來更快。因為找他們的時間更少了


```
In [8]: import time

        t0 = time.time()
        # python list
        l = list(range(100))
        for _ in range(10000):
            for i in range(len(l)):
                l[i] += 1

        t1 = time.time()
        # numpy array
        a = np.array(l)
        for _ in range(10000):
            a += 1

        print("Python list spend {:.3f}s".format(t1-t0))
        print("Numpy array spend {:.3f}s".format(time.time()-t1))

        Python list spend 0.101s
        Numpy array spend 0.012s
```

Numpy 的優勢

Numpy的核心優勢:就是快

Numpy Array 和 Python List 在很多使用場景上是可以互換的，不過在大資料處理的場景下，而且你的資料類型又高度統一， 那麼 Numpy 絕對是你不二的人選，能提升的運算速度也是非常快。

Numpy 對於維度的問題

Numpy的核心優勢:除了快之外，對於多維度的資料處理

特別是在做機器學習，人工智慧的時候，十有八九，人工智慧的演算法裡面，就會出現多維資料的計算問題。可見多維資料在科學計算中的普遍性， 也可見 Numpy 真的是非常有價值的一個 Python 庫。

Numpy 創建多維度的數據資料

創建一個多維度的車輛數據資料
每筆資料包含百公里加速

```
In [10]: import numpy as np
         #四款車型的數據
         cars = np.array([5, 10, 12, 6])
         print("數據：", cars, "\n維度：", cars.ndim)
```

```
數據： [ 5 10 12  6]
維度： 1
```

Numpy 創建多維度的數據資料

In [10]:
```
import numpy as np
#四款車型的數據
cars = np.array([5, 10, 12, 6])
print("數據：", cars, "\n維度：", cars.ndim)
```

```
數據： [ 5 10 12  6]
維度： 1
```

In [15]:
```
cars = np.array([
[
    [5, 10, 12, 6],
    [5.1, 8.2, 11, 6.3],
    [4.4, 9.1, 10, 6.6]
],
[
    [6, 11, 13, 7],
    [6.1, 9.2, 12, 7.3],
    [5.4, 10.1, 11, 7.6]
],
])

print("總維度：", cars.ndim)
print("場地 1 數據：\n", cars[0], "\n場地 1 維度：", cars[0].ndim)
print("場地 2 數據：\n", cars[1], "\n場地 2 維度：", cars[1].ndim)
```

```
總維度： 3
場地 1 數據：
 [[ 5.  10.  12.   6. ]
 [ 5.1  8.2 11.   6.3]
 [ 4.4  9.1 10.   6.6]]
場地 1 維度： 2
場地 2 數據：
 [[ 6.  11.  13.   7. ]
 [ 6.1  9.2 12.   7.3]
 [ 5.4 10.1 11.   7.6]]
場地 2 維度： 2
```

car4
6
6.1
6.2

In [13]:
```
cars = np.array([
[5, 10, 12, 6],
[5.1, 8.1, 11, 6.1],
[5.2, 9.3, 11.5, 6.2]
])

print("數據：\n", cars, "\n維度：", cars.ndim)
```

```
數據：
 [[ 5.  10.  12.   6. ]
 [ 5.1  8.1 11.   6.1]
 [ 5.2  9.3 11.5  6.2]]
維度： 2
```

Numpy –數據新增

如何在原本基礎上的資料新增一筆新的資料

np.concatenate

```
In [16]: cars1 = np.array([5, 10, 12, 6])
         cars2 = np.array([5.2, 4.2])
         cars = np.concatenate([cars1, cars2])
         print(cars)
```

```
[ 5.  10.  12.   6.   5.2  4.2]
```

```
In [17]: test1 = np.array([5, 10, 12, 6])
         test2 = np.array([5.1, 8.2, 11, 6.3])

         # 首先我們要把它們都變成二維，下面這兩種方法都可以加維度
         test1 = np.expand_dims(test1, 0)
         test2 = test2[np.newaxis, :]

         print("test1加維度後 ", test1)
         print("test2加維度後 ", test2)

         # 然後在第一個維度上疊加
         all_tests = np.concatenate([test1, test2])
         print("疊加後\n", all_tests)
```

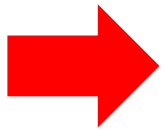
```
test1加維度後  [[ 5 10 12  6]]
test2加維度後  [[ 5.1  8.2 11.   6.3]]
疊加後
 [[ 5.  10.  12.   6. ]
 [ 5.1  8.2 11.   6.3]]
```

Numpy –觀察型態

除了使用np.ndim來觀察型態之外，有時候想了解資料的大小、規格
cars.size

In [19]:
```
cars = np.array([
[5, 10, 12, 6],
[5.1, 8.2, 11, 6.3],
[4.4, 9.1, 10, 6.6]
])

count = 0
for i in range(len(cars)):
    for j in range(len(cars[i])):
        count += 1
print("總共多少測試數據：", count)
```

總共多少測試數據： 12



In [21]:
```
cars = np.array([
[5, 10, 12, 6],
[5.1, 8.2, 11, 6.3],
[4.4, 9.1, 10, 6.6]
])

#count = 0
#for i in range(len(cars)):
#    for j in range(len(cars[i])):
#        count += 1
#print("總共多少測試數據：", count)
print("總共多少測試數據：", cars.size)
```

總共多少測試數據： 12

Numpy –觀察型態

我想知道所有維度的數量

```
In [23]: print("第一個維度：", cars.shape[0])
         print("第二個維度：", cars.shape[1])
         print("所有維度：", cars.shape)
```

```
第一個維度： 3
第二個維度： 4
所有維度： (3, 4)
```

```
[1]  import numpy as np

     np1 = np.array([1, 2, 3])

     np2 = np.array([3, 4, 5])
```

載入需要的函式庫
建立矩陣並給予初始值

```
[2]  print(np1)

     print(np2)

     print(np1 + np2)
```

印出矩陣
印出相加矩陣

```
[1 2 3]
[3 4 5]
[4 6 8]
```

Numpy

```
[3]  print(np1.ndim, np1.shape, np1.dtype)
```

```
1 (3,) int64
```

印出矩陣特性

Numpy

```
[6]  import numpy as np

     np3 = np.array([1, 2, 3, 4, 5, 6])

     print(np3)

     print(np3.ndim, np3.shape, np3.dtype)
```

```
[1 2 3 4 5 6]
1 (6,) int64
```

Numpy

```
[8]  np3 = np3.reshape([2, 3])

     print(np3)

     print(np3.ndim, np3.shape, np3.dtype)
```

```
[[1 2 3]
 [4 5 6]]
2 (2, 3) int64
```

矩陣維度轉換

Numpy

```
[9]  import numpy as np

     np1 = np.zeros([2, 3])
     np2 = np.ones([2, 3])

     print(np1)
     print(np2)
```

```
[[0. 0. 0.]
 [0. 0. 0.]]
[[1. 1. 1.]
 [1. 1. 1.]]
```

快速建立矩陣並給予初始值

Numpy

載入額外函式庫，用來畫圖及上傳圖檔

```
[1]  1 import numpy as np
     2 import matplotlib.pylab as plt
     3 from google.colab import files
```

```
[2]  1 uploaded = files.upload()
```

- **minion.jpg**(image/jpeg) - 29833 bytes, last modified: 2020/3/17 - 100% done

```
Saving minion.jpg to minion.jpg
```

Numpy

```
[3]   1 img = plt.imread("minion.jpg")
```

```
[5]   1 img.shape
```

```
(354, 630, 3)
```

將圖檔資料載入至img

查看特性

Numpy

```
[9]   1 print(img.ndim, img.shape, img.dtype)
```

```
3 (354, 630, 3) uint8
```

Numpy

```
[10]  1 print(img)
```

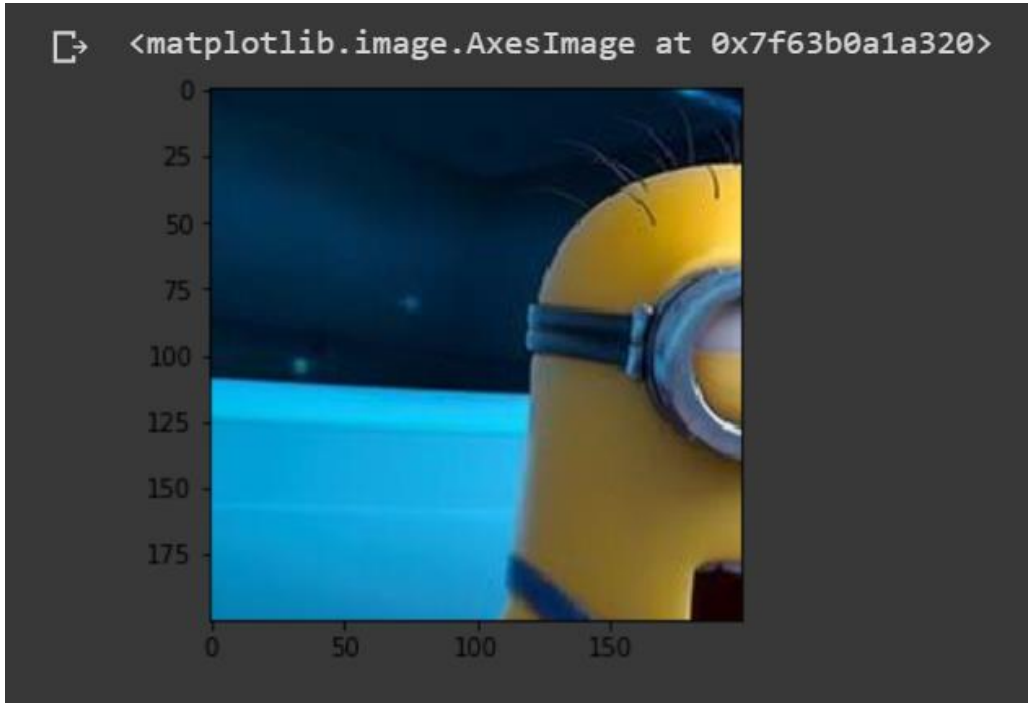
```
[[[  3  25  46]
  [  3  25  46]
  [  3  25  46]
  ...
  [ 44  95 142]
  [ 44  95 142]
  [ 44  95 142]]

 [[  3  25  46]
  [  3  25  46]
  [  3  25  46]
  ...
  [ 44  95 142]
  [ 44  95 142]
  [ 44  95 142]]
```

Numpy

```
[6]   1 plt.imshow(img)
```

```
<matplotlib.image.AxesImage at 0x7f63b28ab668>
```

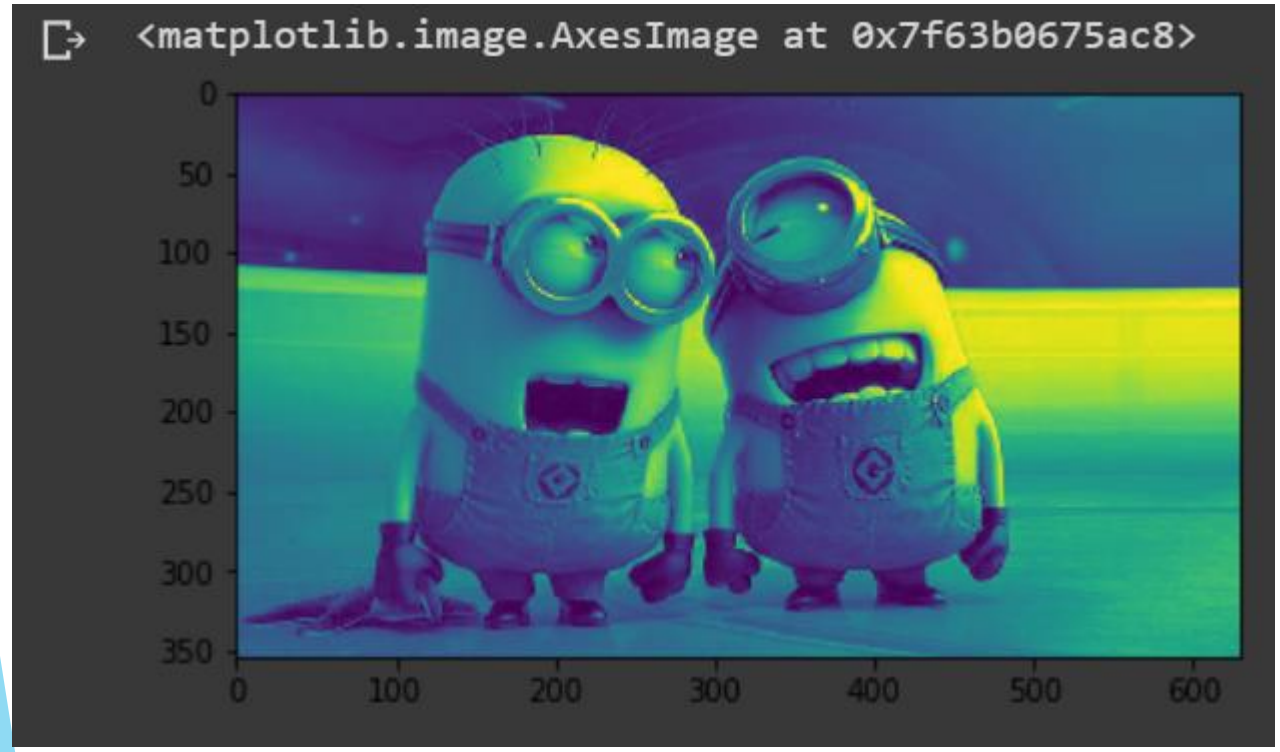
Numpy



利用簡單的運算概念，印出不同的效果

Numpy

利用簡單的運算概念，印出不同的效果



Numpy

```
[33]  1 import numpy as np
      2 import matplotlib.pylab as plt
      3 from google.colab import files
      4 import cv2
```

```
[34]  1 img = plt.imread("minion.jpg")
      2 img_cv = cv2.imread("minion.jpg")
```

```
[35]  1 img.shape
```

```
(354, 630, 3)
```

```
[37]  1 img_cv.shape
```

```
(354, 630, 3)
```

不同的套件載入同一張圖片，資料欄位的定義也有可能不同

Numpy

```
[29]  1 plt.imshow(img)
```

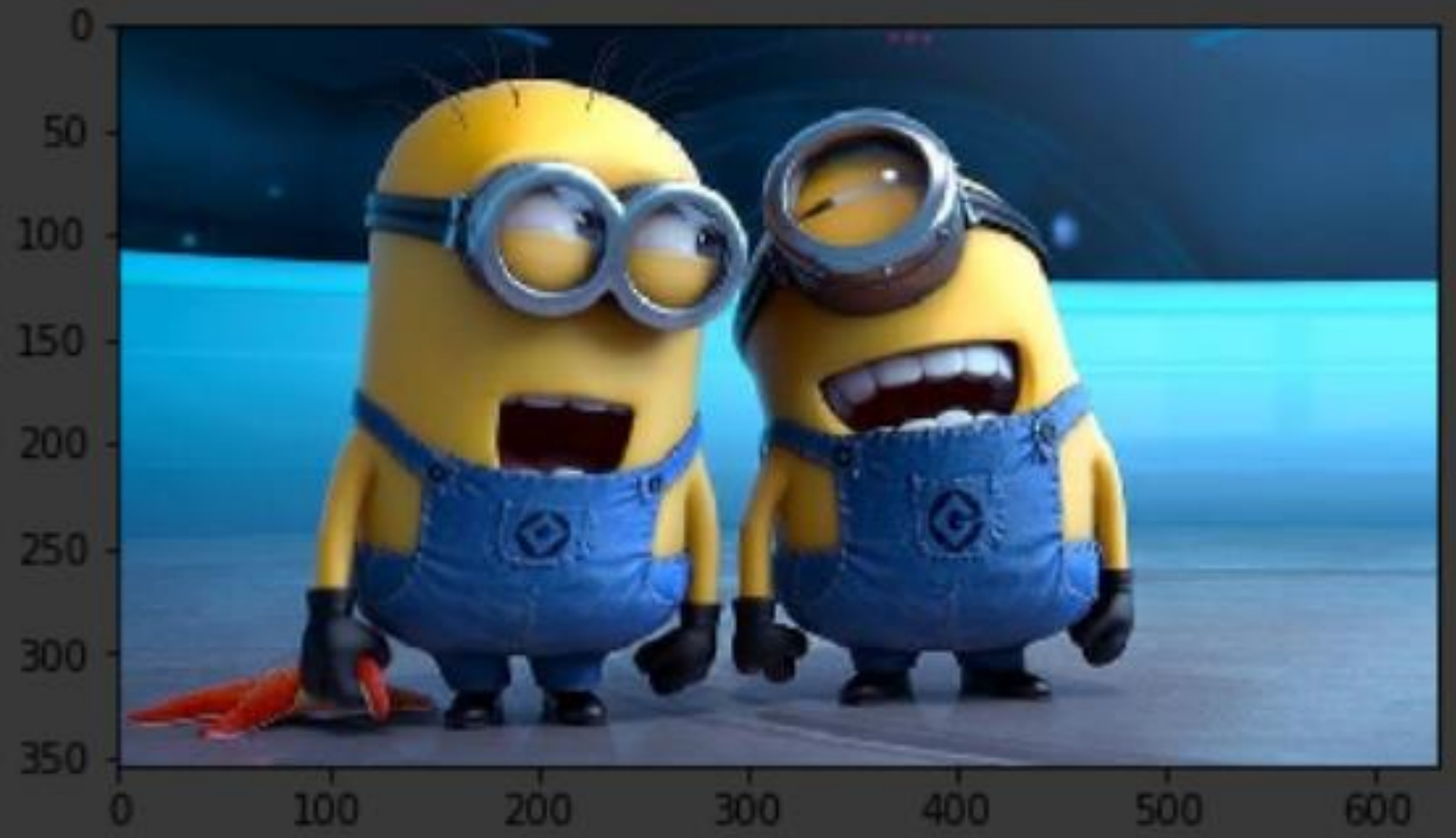
```
<matplotlib.image.AxesImage at 0x7f63b05e1550>
```


```
[40]  1 plt.imshow(img_cv)
```

```
<matplotlib.image.AxesImage at 0x7f63a73f9470>
```

```
[42]  1 plt.imshow(cv2.cvtColor(img_cv, cv2.COLOR_BGR2RGB))
```

```
<matplotlib.image.AxesImage at 0x7f63a7377cc0>
```

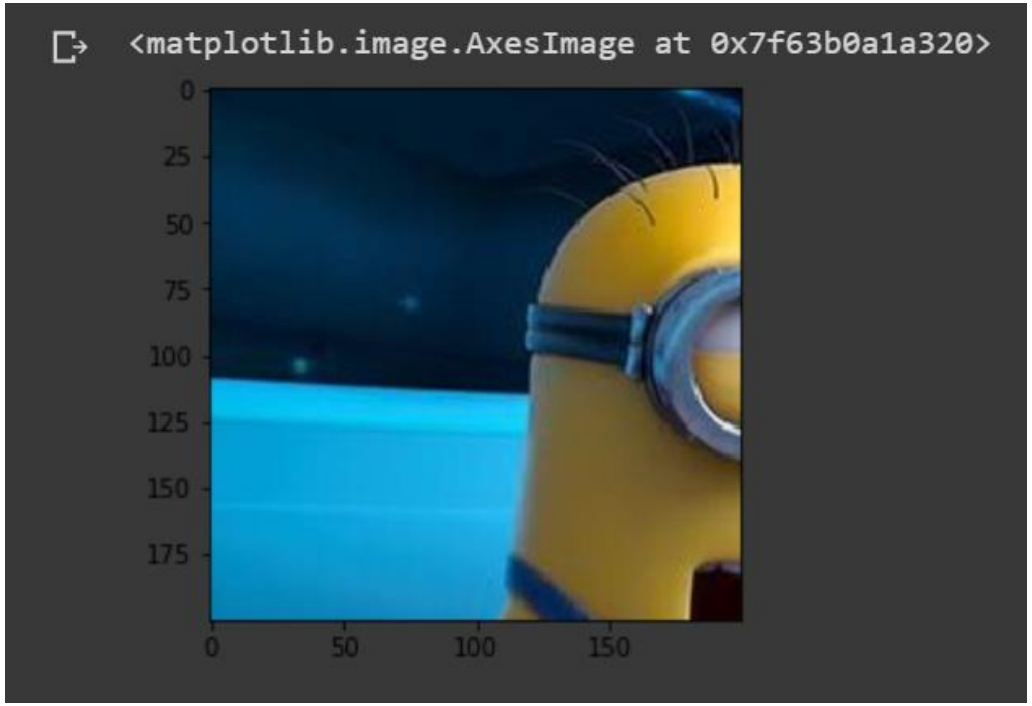


習題

- 利用載入的檔案，將色彩空間的資料互換，看看會產生什麼結果

習題

- 利用載入的檔案，加上一些簡單的矩陣運算，把圖片變成1/4大小




THANK
YOU!