

Python

環境 程式 安裝



2020 Deep Learning



STEP 1

<https://repo.continuum.io/archive/winzip/>

下載 Anaconda3-4.2.0-Windows-x86_64 的版本，並進行環境安裝

Anaconda3-4.3.0-Windows-x86_64.zip	418.8M	2017-01-31 10:50:25	e703a6
Anaconda2-4.2.0-Windows-x86.zip	322.0M	2016-09-28 12:01:22	b2bd3b
Anaconda2-4.2.0-Windows-x86_64.zip	378.8M	2016-09-28 12:01:06	987a36
Anaconda3-4.2.0-Windows-x86.zip	331.2M	2016-09-28 12:02:00	d6e415
<u>Anaconda3-4.2.0-Windows-x86_64.zip</u>	389.1M	2016-09-28 12:01:43	582e9f
Anaconda2-4.1.1-Windows-x86.zip	284.1M	2016-07-08 11:32:17	bb6273



STEP 2

開啟 Anaconda Prompt 依序輸入

<https://gettocode.com/2016/12/02/keras-on-theano-and-tensorflow-on-windows-and-linux/>

此網站內步驟(如圖)安裝即可成功

Link

- After Anaconda was installed open terminal and Install Theano.
- When you are asked about installing dependencies click 'y' for yes.

```
C:\>conda install theano
```

- To enable gcc compiler for Theano install following
- When you are asked about installing dependencies click 'y' for yes.

建議版本

pip install tensorflow==1.10.0
or pip install tensorflow==1.1.0

pip install keras==2.1.2
or pip install keras==2.2.2



STEP 3 - TEST

安裝完成後,可在Keras做mnist資料集測試,驗證程式安裝是否正確

依序輸入下列程式碼

```
>>> conda install git
```

```
>>> git clone https://github.com/fchollet/keras.git
```

```
>>> cd keras/examples/
```

```
>>> python mnist_mlp.py
```

大概跑五分鐘後會出現右圖,即安裝成功!!!

```
53120/60000 [=====>...] - ETA: 1s - loss: 0.0187 - acc: 0.9
53504/60000 [=====>...] - ETA: 1s - loss: 0.0186 - acc: 0.9
53888/60000 [=====>...] - ETA: 1s - loss: 0.0186 - acc: 0.9
54272/60000 [=====>...] - ETA: 1s - loss: 0.0186 - acc: 0.9
54656/60000 [=====>...] - ETA: 0s - loss: 0.0185 - acc: 0.9
55040/60000 [=====>...] - ETA: 0s - loss: 0.0185 - acc: 0.9
55424/60000 [=====>...] - ETA: 0s - loss: 0.0186 - acc: 0.9
55808/60000 [=====>...] - ETA: 0s - loss: 0.0185 - acc: 0.9
56192/60000 [=====>...] - ETA: 0s - loss: 0.0186 - acc: 0.9
56576/60000 [=====>...] - ETA: 0s - loss: 0.0188 - acc: 0.9
56960/60000 [=====>...] - ETA: 0s - loss: 0.0189 - acc: 0.9
57344/60000 [=====>...] - ETA: 0s - loss: 0.0189 - acc: 0.9
57728/60000 [=====>...] - ETA: 0s - loss: 0.0191 - acc: 0.9
58112/60000 [=====>...] - ETA: 0s - loss: 0.0190 - acc: 0.9
58496/60000 [=====>...] - ETA: 0s - loss: 0.0189 - acc: 0.9
58880/60000 [=====>...] - ETA: 0s - loss: 0.0188 - acc: 0.9
59264/60000 [=====>...] - ETA: 0s - loss: 0.0188 - acc: 0.9
59648/60000 [=====>...] - ETA: 0s - loss: 0.0189 - acc: 0.9
60000/60000 [=====] - 11s - loss: 0.0188 - acc: 0.9951
- val_loss: 0.1104 - val acc: 0.9828
Test loss: 0.110370836233
Test accuracy: 0.9828
```

成功跑出accuracy即代表安裝成功



STEP 4

開啟



Spyder
桌面應用程式

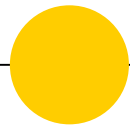
即可在裡面使用python做程式編寫,

寫好後按下Run



即可在右邊視窗中看到跑的結果

Python NumPy Tutorial



2020 Deep Learning

<http://cs231n.github.io/python-numpy-tutorial/>

Python versions

- ◎ Python現今支援兩種版本，分別是2.7版和3.5版。
- ◎ 由於Python 3.0引入了許多向後兼容的的語言，所以2.7版或是3.5版編寫的程式，無法在另一版本上運行。
- ◎ 後續介紹使用3.5版，可用指令確認版本：

```
>>> python --version
```



Python basic data types

numbers

```
y = 2.5
print(type(y)) # Prints "<class 'float'>"
print(y, y + 1, y * 2, y ** 2) # Prints "2.5 3.5 5.0 6.25"
```

booleans

```
t = True
f = False
print(type(t)) # Prints "<class 'bool'>"
print(t and f) # Logical AND; prints "False"
print(t or f)  # Logical OR; prints "True"
print(not t)   # Logical NOT; prints "False"
print(t != f)  # Logical XOR; prints "True"
```

strings

```
s = "hello"
print(s.capitalize()) # Capitalize a string; prints "Hello"
print(s.upper())      # Convert a string to uppercase; prints "HELLO"
print(s.rjust(7))     # Right-justify a string, padding with spaces; prints " hello"
print(s.center(7))    # Center a string, padding with spaces; prints " hello "
print(s.replace('l', '(ell)')) # Replace all instances of one substring with another;
                                # prints "he(ell)(ell)o"
print(' world '.strip()) # Strip leading and trailing whitespace; prints "world"
```




Python containers

用來持有、保存物件的群集 (Collection) 物件 Python 的資料容器可以讓我們將多個資料指派給一個物件像是 lists、dictionaries、sets 和 tuples

資料容器	建立方法	特性
list	使用中括號[]	富有彈性且能容納不同資料類型
dict	使用大括號{} 並搭配鍵值與資料	使用鍵值為資料作索引
set	使用大括號{ }	儲存唯一值
tuple	使用小括號()	不可以增減元素



Python containers **lists**

- 1) 一種極有彈性的資料容器，可以容納不同的資料類型

```
xs = [3, 1, 2]    # Create a List
print(xs, xs[2])  # Prints "[3, 1, 2] 2"
print(xs[-1])     # Negative indices count from the end of the list; prints "2"
xs[2] = 'foo'     # Lists can contain elements of different types
print(xs)         # Prints "[3, 1, 'foo']"
xs.append('bar')   # Add a new element to the end of the list
print(xs)         # Prints "[3, 1, 'foo', 'bar']"
x = xs.pop()      # Remove and return the last element of the list
print(x, xs)      # Prints "bar [3, 1, 'foo']"
```

Stack 為一個遵守**最後加入元素最先被取回**（後進先出，"last-in, first-out"）規則的資料結構。

`append()` 將一個項目放到堆疊的頂層。

`pop()` 且不給定索引值去取得堆疊最上面的項目



Python containers **lists**

- 2) Slicing 在使用冒號 (:) 進行元素選擇的時候需要注意，位於冒號前索引值的元素會被包含，但位於冒號後索引值的元素則**不會被包含**

```
nums = list(range(5))    # range is a built-in function that creates a list of integers
print(nums)              # Prints "[0, 1, 2, 3, 4]"
print(nums[2:4])         # Get a slice from index 2 to 4 (exclusive); prints "[2, 3]"
print(nums[2:])          # Get a slice from index 2 to the end; prints "[2, 3, 4]"
print(nums[:2])          # Get a slice from the start to index 2 (exclusive); prints "[0, 1]"
print(nums[:])           # Get a slice of the whole list; prints "[0, 1, 2, 3, 4]"
print(nums[:-1])         # Slice indices can be negative; prints "[0, 1, 2, 3]"
nums[2:4] = [8, 9]       # Assign a new sublist to a slice
print(nums)              # Prints "[0, 1, 8, 9, 4]"
```



Python containers **lists**

3) loops

```
animals = ['cat', 'dog', 'monkey']  
for animal in animals:  
    print(animal)  
# Prints "cat", "dog", "monkey", each on its own line.
```

```
animals = ['cat', 'dog', 'monkey']  
for idx, animal in enumerate(animals):  
    print('#%d: %s' % (idx + 1, animal))  
# Prints "#1: cat", "#2: dog", "#3: monkey", each on its own line
```

將數據轉成索引序列



Python containers **lists**

4) list comprehensions 用陳述式創建 list

```
nums = [0, 1, 2, 3, 4]
squares = []
for x in nums:
    squares.append(x ** 2)
print(squares)  # Prints [0, 1, 4, 9, 16]
```



```
nums = [0, 1, 2, 3, 4]
squares = [x ** 2 for x in nums]
print(squares)  # Prints [0, 1, 4, 9, 16]
```

```
nums = [0, 1, 2, 3, 4]
even_squares = [x ** 2 for x in nums if x % 2 == 0]
print(even_squares)  # Prints "[0, 4, 16]"
```

(contain conditions)



Python containers **dictionaries**

- 1) 主要的操作為藉由鍵(key)來儲存一個值並且可藉由該鍵來取出該值(value)。也可以使用 `del` 來刪除鍵值對。如果使用用過的鍵來儲存，該鍵所對應的較舊的值會被覆蓋。使用不存在的鍵來取出值會造成錯誤

```
d = {'cat': 'cute', 'dog': 'furry'} # Create a new dictionary with some data
print(d['cat']) # Get an entry from a dictionary; prints "cute"
print('cat' in d) # Check if a dictionary has a given key; prints "True"
d['fish'] = 'wet' # Set an entry in a dictionary
print(d['fish']) # Prints "wet"
# print(d['monkey']) # KeyError: 'monkey' not a key of d
print(d.get('monkey', 'N/A')) # Get an element with a default; prints "N/A"
print(d.get('fish', 'N/A')) # Get an element with a default; prints "wet"
del d['fish'] # Remove an element from a dictionary
print(d.get('fish', 'N/A')) # "fish" is no longer a key; prints "N/A"
```

確認一個鍵是否已存在於字典中

返回指定鍵的值，
如果值不在字典中返回默认值



Python containers **dictionaries**

2) loops

```
d = {'person': 2, 'cat': 4, 'spider': 8}
for animal in d:
    legs = d[animal]
    print('A %s has %d legs' % (animal, legs))
# Prints "A person has 2 legs", "A cat has 4 legs", "A spider has 8 legs"
```

```
d = {'person': 2, 'cat': 4, 'spider': 8}
for animal, legs in d.items():
    print('A %s has %d legs' % (animal, legs))
# Prints "A person has 2 legs", "A cat has 4 legs", "A spider has 8 legs"
```

鍵以及其對應的值可以藉由使用 `items()` 方法來同時取得



Python containers dictionaries

3) Dictionary comprehensions 透過鍵與值的陳述式來創建 dict

```
nums = [0, 1, 2, 3, 4]
even_num_to_square = {x: x ** 2 for x in nums if x % 2 == 0}
print(even_num_to_square) # Prints "{0: 0, 2: 4, 4: 16}"
```




Python containers **sets**

- 1) 無序、元素不重複的集合，不會包含重複的資料

```
animals = {'cat', 'dog'}  
print('cat' in animals)    # Check if an element is in a set; prints "True"  
print('fish' in animals)   # prints "False"  
animals.add('fish')        # Add an element to a set  
print('fish' in animals)   # Prints "True"  
print(len(animals))        # Number of elements in a set; prints "3"  
animals.add('cat')         # Adding an element that is already in the set does nothing  
print(len(animals))        # Prints "3"  
animals.remove('cat')      # Remove an element from a set  
print(len(animals))        # Prints "2"
```



Python containers **sets**

2) loops 無法使用索引來擷取特定元素

```
animals = {'cat', 'dog', 'fish'}  
for idx, animal in enumerate(animals):  
    print('#%d: %s' % (idx + 1, animal))  
# Prints "#1: fish", "#2: dog", "#3: cat"
```

3) set comprehension


```
from math import sqrt  
nums = {int(sqrt(x)) for x in range(30)}  
print(nums) # Prints "{0, 1, 2, 3, 4, 5}"
```



Python containers **tuples**

類似list，可當作字典的key，但元素不可增減

```
d = {(x, x + 1): x for x in range(10)} # Create a dictionary with tuple keys
t = (5, 6) # Create a tuple
print(type(t)) # Prints "<class 'tuple'>"
print(d[t]) # Prints "5"
print(d[(1, 2)]) # Prints "1"
```

 d - Dictionary (10 elements)

Key	Type	Size	
(0, 1)	int	1	0
(1, 2)	int	1	1
(2, 3)	int	1	2
(3, 4)	int	1	3
(4, 5)	int	1	4
(5, 6)	int	1	5
(6, 7)	int	1	6
(7, 8)	int	1	7
(8, 9)	int	1	8
(9, 10)	int	1	9



Python NumPy array

NumPy (Numerical Python)是Python語言的一個擴充程式庫，支援高階大量的維度陣列與矩陣運算
透過傳入 `list` 到 `numpy.array()` 創建陣列

```
import numpy as np

a = np.array([1, 2, 3]) # Create a rank 1 array
print(type(a))         # Prints "<class 'numpy.ndarray'>"
print(a.shape)         # Prints "(3,)"
print(a[0], a[1], a[2]) # Prints "1 2 3"
a[0] = 5               # Change an element of the array
print(a)               # Prints "[5, 2, 3]"

b = np.array([[1,2,3],[4,5,6]]) # Create a rank 2 array
print(b.shape)               # Prints "(2, 3)"
print(b[0, 0], b[0, 1], b[1, 0]) # Prints "1 2 4"
```



Python NumPy array

NumPy (Numerical Python)是Python語言的一個擴充程式庫，支援高階大量的維度陣列與矩陣運算
透過其他numpy函式創建陣列

```
import numpy as np

a = np.zeros((2,2)) # Create an array of all zeros
print(a)           # Prints "[[ 0.  0.]
                    #           [ 0.  0.]]"

b = np.ones((1,2)) # Create an array of all ones
print(b)           # Prints "[[ 1.  1.]]"

c = np.full((2,2), 7) # Create a constant array
print(c)            # Prints "[[ 7.  7.]
                    #           [ 7.  7.]]"

d = np.eye(2)       # Create a 2x2 identity matrix
print(d)            # Prints "[[ 1.  0.]
                    #           [ 0.  1.]]"

e = np.random.random((2,2)) # Create an array filled with random values
print(e)             # Might print "[[ 0.91940167  0.08143941]
                    #           [ 0.68744134  0.87236687]]"
```



Python NumPy

array math

```
import numpy as np

x = np.array([[1,2],[3,4]], dtype=np.float64)
y = np.array([[5,6],[7,8]], dtype=np.float64)

# Elementwise sum; both produce the array
# [[ 6.0  8.0]
# [10.0 12.0]]
print(x + y)
print(np.add(x, y))

# Elementwise difference; both produce the array
# [[-4.0 -4.0]
# [-4.0 -4.0]]
print(x - y)
print(np.subtract(x, y))
```

```
# Elementwise product; both produce the array
# [[ 5.0 12.0]
# [21.0 32.0]]
print(x * y)
print(np.multiply(x, y))

# Elementwise division; both produce the array
# [[ 0.2          0.33333333]
# [ 0.42857143  0.5         ]]
print(x / y)
print(np.divide(x, y))

# Elementwise square root; produces the array
# [[ 1.          1.41421356]
# [ 1.73205081  2.         ]]
print(np.sqrt(x))
```

** is elementwise multiplication*



Python NumPy array math

```
import numpy as np

x = np.array([[1,2],[3,4]])
y = np.array([[5,6],[7,8]])

v = np.array([9,10])
w = np.array([11, 12])

# Inner product of vectors; both produce 219
print(v.dot(w))
print(np.dot(v, w))

# Matrix / vector product; both produce the rank 1 array [29 67]
print(x.dot(v))
print(np.dot(x, v))

# Matrix / matrix product; both produce the rank 2 array
# [[19 22]
#  [43 50]]
print(x.dot(y))
print(np.dot(x, y))
```

dot() to compute inner products of vectors



Python NumPy broadcasting

執行broadcast的前提是兩個 ndarray 是 element-wise的運算

```
import numpy as np

# We will add the vector v to each row of the matrix x,
# storing the result in the matrix y
x = np.array([[1,2,3], [4,5,6], [7,8,9], [10, 11, 12]])
v = np.array([1, 0, 1])
y = x + v # Add v to each row of x using broadcasting
print(y) # Prints "[[ 2  2  4]
          #          [ 5  5  7]
          #          [ 8  8 10]
          #          [11 11 13]]"
```




Python SciPy

Image operations

(開源的Python演算法庫和數學工具包)

```
from scipy.misc import imread, imsave, imresize

# Read an JPEG image into a numpy array
img = imread('assets/cat.jpg')
print(img.dtype, img.shape) # Prints "uint8 (400, 248, 3)"

# We can tint the image by scaling each of the color channels
# by a different scalar constant. The image has shape (400, 248, 3);
# we multiply it by the array [1, 0.95, 0.9] of shape (3,);
# numpy broadcasting means that this leaves the red channel unchanged,
# and multiplies the green and blue channels by 0.95 and 0.9
# respectively.
img_tinted = img * [1, 0.95, 0.9] ➡ [R,G,B]

# Resize the tinted image to be 300 by 300 pixels.
img_tinted = imresize(img_tinted, (300, 300))

# Write the tinted image back to disk
imsave('assets/cat_tinted.jpg', img_tinted)
```



MATLAB files

使用「`scipy.io.loadmat`」和「`scipy.io.savemat`」，即可針對MATLAB file 讀寫。



Python Matplotlib

plotting

```
import numpy as np
import matplotlib.pyplot as plt

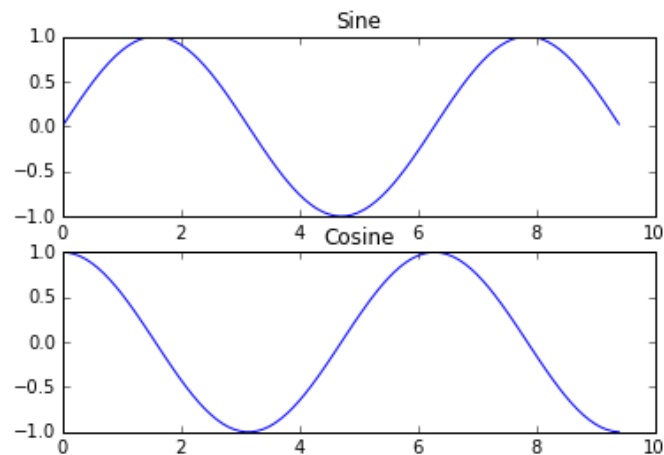
# Compute the x and y coordinates for points on sine and cosine curves
x = np.arange(0, 3 * np.pi, 0.1)
y_sin = np.sin(x)
y_cos = np.cos(x)

# Set up a subplot grid that has height 2 and width 1,
# and set the first such subplot as active.
plt.subplot(2, 1, 1)

# Make the first plot
plt.plot(x, y_sin)
plt.title('Sine')

# Set the second subplot as active, and make the second plot.
plt.subplot(2, 1, 2)
plt.plot(x, y_cos)
plt.title('Cosine')

# Show the figure.
plt.show()
```





Python Matplotlib

images

```
import numpy as np
from scipy.misc import imread, imresize
import matplotlib.pyplot as plt

img = imread('assets/cat.jpg')
img_tinted = img * [1, 0.95, 0.9]

# Show the original image
plt.subplot(1, 2, 1)
plt.imshow(img)

# Show the tinted image
plt.subplot(1, 2, 2)

# A slight gotcha with imshow is that it might give strange results
# if presented with data that is not uint8. To work around this, we
# explicitly cast the image to uint8 before displaying it.
plt.imshow(np.uint8(img_tinted))
plt.show()
```

