vncoder.vn

Bài 17: Sort, Search & Counting

6-8 phút

Nhiều chức năng liên quan đến sắp xếp có sẵn trong NumPy. Các hàm sắp xếp này thực hiện các thuật toán sắp xếp khác nhau, mỗi thuật toán được đặc trưng bởi tốc độ thực thi, hiệu suất trong trường hợp xấu nhất, không gian hoạt động cần thiết và tính ổn định của các thuật toán. Bảng sau đây cho thấy sự so sánh của ba thuật toán sắp xếp.

Kind	Speed	Worst case	Work space	Stable
‘quicksort’	1	O(n^2)	0	Không
‘mergesort’	2	O(n*log(n))	~n/2	Có
‘heapsort’	3	O(n*log(n))	0	Không

1. numpy.sort() :

Hàm sort () trả về một bản sao đã được sắp xếp của mảng đầu vào

```
numpy.sort(a, axis, kind, order)
```

Trong đó :

1. a : Mảng được sắp xếp
2. axis : Trục dọc theo mảng sẽ được sắp xếp. Nếu không có, mảng sẽ được làm phẳng, sắp xếp trên trục cuối cùng
3. kind : Mặc định là quicksort

4. order : Nếu mảng chứa các trường, thứ tự các trường sẽ được sắp xếp

Ví dụ :

```
import numpy as np
a = np.array([[3,7],[9,1]])

print 'Our array is:'
print a
print '\n'

print 'Applying sort() function:'
print np.sort(a)
print '\n'

print 'Sort along axis 0:'
print np.sort(a, axis = 0)
print '\n'

# Order parameter in sort function
dt = np.dtype([('name', 'S10'),('age', int)])
a = np.array([("raju",21),("anil",25),("ravi",
17), ("amar",27)], dtype = dt)

print 'Our array is:'
print a
print '\n'

print 'Order by name:'
print np.sort(a, order = 'name')
```

Kết quả :

```
Our array is:
[[3 7]
 [9 1]]

Applying sort() function:
[[3 7]
 [1 9]]

Sort along axis 0:
[[3 1]
 [9 7]]

Our array is:
[('raju', 21) ('anil', 25) ('ravi', 17)
('amar', 27)]

Order by name:
[('amar', 27) ('anil', 25) ('raju', 21)
('ravi', 17)]
```

2. numpy.argsort() :

Hàm numpy.argsort () thực hiện sắp xếp gián tiếp trên mảng đầu vào, dọc theo trục đã cho và sử dụng kiểu sắp xếp cụ thể để trả về mảng chỉ số dữ liệu. Mảng này được sử dụng để xây dựng mảng đã sắp xếp

Ví dụ :

```
import numpy as np
x = np.array([3, 1, 2])

print 'Our array is:'
print x
```

```
print '\n'

print 'Applying argsort() to x:'
y = np.argsort(x)
print y
print '\n'

print 'Reconstruct original array in sorted
order:'
print x[y]
print '\n'

print 'Reconstruct the original array using
loop:'
for i in y:
   print x[i],
```

Kết quả :

```
Our array is:
[3 1 2]

Applying argsort() to x:
[1 2 0]

Reconstruct original array in sorted order:
[1 2 3]

Reconstruct the original array using loop:
1 2 3
```

3. numpy.lexsort()

hàm thực hiện sắp xếp gián tiếp bằng cách sử dụng một chuỗi

khóa. Các khóa có thể được xem như một cột trong bảng tính. Hàm trả về một mảng chỉ số, sử dụng dữ liệu đã sắp xếp để lấy dữ liệu. Lưu ý rằng khóa cuối cùng sẽ là khóa chính

Ví dụ :

```
import numpy as np

nm = ('raju','anil','ravi','amar')
dv = ('f.y.', 's.y.', 's.y.', 'f.y.')
ind = np.lexsort((dv,nm))

print 'Applying lexsort() function:'
print ind
print '\n'

print 'Use this index to get sorted data:'
print [nm[i] + ", " + dv[i] for i in ind]
```

Kết quả :

```
Applying lexsort() function:
[3 1 0 2]

Use this index to get sorted data:
['amar, f.y.', 'anil, s.y.', 'raju, f.y.',
'ravi, s.y.']
```

NumPy có một số hàm để tìm kiếm bên trong một mảng. Có sẵn các hàm tìm min, max cũng như các phần tử thỏa mãn một điều kiện nhất định.

4. numpy.argmax() and numpy.argmin()

Hai hàm này trả về chỉ số của các phần tử cực đại và cực tiểu tương ứng dọc theo trục đã cho.

Ví dụ :

```
import numpy as np
a = np.array([[30,40,70],[80,20,10],
[50,90,60]])

print 'Our array is:'
print a
print '\n'

print 'Applying argmax() function:'
print np.argmax(a)
print '\n'

print 'Index of maximum number in flattened
array'
print a.flatten()
print '\n'

print 'Array containing indices of maximum
along axis 0:'
maxindex = np.argmax(a, axis = 0)
print maxindex
print '\n'

print 'Array containing indices of maximum
along axis 1:'
maxindex = np.argmax(a, axis = 1)
print maxindex
print '\n'

print 'Applying argmin() function:'
```

```
minindex = np.argmin(a)
print minindex
print '\n'

print 'Flattened array:'
print a.flatten()[minindex]
print '\n'

print 'Flattened array along axis 0:'
minindex = np.argmin(a, axis = 0)
print minindex
print '\n'

print 'Flattened array along axis 1:'
minindex = np.argmin(a, axis = 1)
print minindex
```

Kết quả :

```
Our array is:
[[30 40 70]
 [80 20 10]
 [50 90 60]]

Applying argmax() function:
7

Index of maximum number in flattened array
[30 40 70 80 20 10 50 90 60]

Array containing indices of maximum along axis
0:
[1 2 0]
```

```
Array containing indices of maximum along axis
1:
[2 0 1]

Applying argmin() function:
5

Flattened array:
10

Flattened array along axis 0:
[0 1 1]

Flattened array along axis 1:
[0 2 0]
```

5. numpy.nonzero() :

Hàm numpy.nonzero () trả về chỉ số của các phần tử khác 0 trong mảng đầu vào.

Ví dụ :

```
import numpy as np
a = np.array([[30,40,0],[0,20,10],[50,0,60]])

print 'Our array is:'
print a
print '\n'

print 'Applying nonzero() function:'
print np.nonzero (a)
```

Kết quả :

```
Our array is:
[[30 40 0]
 [ 0 20 10]
 [50 0 60]]

Applying nonzero() function:
(array([0, 0, 1, 1, 2, 2]), array([0, 1, 1, 2,
0, 2]))
```

6. numpy.where()

Hàm where () trả về chỉ số của các phần tử trong mảng đầu vào thỏa mãn điều kiện đã cho.

Ví dụ :

```
import numpy as np
x = np.arange(9.).reshape(3, 3)

print 'Our array is:'
print x

print 'Indices of elements > 3'
y = np.where(x > 3)
print y

print 'Use these indices to get elements
satisfying the condition'
print x[y]
```

Kết quả :

```
Our array is:
[[ 0. 1. 2.]
```

```
 [ 3. 4. 5.]
 [ 6. 7. 8.]]

Indices of elements > 3
(array([1, 1, 2, 2, 2]), array([1, 2, 0, 1,
2]))

Use these indices to get elements satisfying
the condition
[ 4. 5. 6. 7. 8.]
```

7. numpy.extract()

Hàm extract () trả về các phần tử thỏa mãn bất kỳ điều kiện nào.

```
import numpy as np
x = np.arange(9.).reshape(3, 3)

print 'Our array is:'
print x

# define a condition
condition = np.mod(x,2) == 0

print 'Element-wise value of condition'
print condition

print 'Extract elements using condition'
print np.extract(condition, x)
```

Kết quả :

```
Our array is:
```

```
[[ 0. 1. 2.]
 [ 3. 4. 5.]
 [ 6. 7. 8.]]

Element-wise value of condition
[[ True False True]
 [False True False]
 [ True False True]]

Extract elements using condition
[ 0. 2. 4. 6. 8.]
```