vncoder.vn

Bài 21: Đại số tuyến tính

5-6 phút

NumPy chứa mô-đun numpy.linalg cung cấp tất cả các hàm cần thiết cho đại số tuyến tính. Một số chức năng quan trọng trong mô-đun như sau :

1 . numpy.dot() :

Hàm này trả về tích số chấm của hai mảng. Đối với vectơ 2-D, nó tương đương với phép nhân ma trận. Đối với mảng 1-D, nó là tích bên trong của các vectơ. Đối với mảng N chiều, nó là một tích tổng trên trục cuối cùng của a và trục cuối cùng thứ hai của b.

Ví dụ :

```
import numpy.matlib
import numpy as np

a = np.array([[1,2],[3,4]])
b = np.array([[11,12],[13,14]])
np.dot(a,b)
```

Kết quả :

Lưu ý rằng kết quả được tính là -

```
[[1*11+2*13, 1*12+2*14],[3*11+4*13, 3*12+4*14]]
```

2. numpy.vdot()

Hàm này trả về tích số chấm của hai vectơ. Nếu đối số đầu tiên phức tạp, thì liên hợp của nó được sử dụng để tính toán. Nếu id đối số là mảng nhiều chiều, nó sẽ được làm phẳng.

Ví dụ :

```
import numpy as np
a = np.array([[1,2],[3,4]])
b = np.array([[11,12],[13,14]])
print np.vdot(a,b)
```

Kết quả :

Kết quả được tính như sau : 1*11 + 2*12 + 3*13 + 4*14 = 130

3. numpy.inner()

Hàm này trả về tích bên trong của vectơ cho mảng 1-D. Đối với các kích thước cao hơn, nó trả về tích tổng trên các trục cuối cùng.

Ví dụ 1:

```
import numpy as np
print
np.inner(np.array([1,2,3]),np.array([0,1,0]))
# Equates to 1*0+2*1+3*0
```

Kết quả :

Đối với mảng nhiều chiều :

Ví dụ 2 :

```
# Multi-dimensional array example
import numpy as np
a = np.array([[1,2], [3,4]])

print 'Array a:'
```

```
print a
b = np.array([[11, 12], [13, 14]])

print 'Array b:'
print b

print 'Inner product:'
print np.inner(a,b)
```

Kết quả :

```
Array a:
[[1 2]
[3 4]]

Array b:
[[11 12]
[13 14]]

Inner product:
[[35 41]
[81 95]]
```

Kết quả được tính như sau :

```
1*11+2*12, 1*13+2*14
3*11+4*12, 3*13+4*14
```

4. numpy.matmul()

Hàm numpy.matmul () trả về tích ma trận của hai mảng. Trong khi nó trả về một sản phẩm bình thường cho mảng 2-D, nếu kích thước của một trong hai đối số là> 2, nó được coi là một chồng ma trận nằm trong hai chỉ mục cuối cùng và được phát sóng tương ứng.

Mặt khác, nếu một trong hai đối số là mảng 1-D, thì nó được thăng cấp thành ma trận bằng cách thêm 1 vào thứ nguyên của nó, thứ nguyên này sẽ bị xóa sau khi nhân.

Ví dụ 1 :

```
# For 2-D array, it is matrix multiplication
import numpy.matlib
import numpy as np

a = [[1,0],[0,1]]
b = [[4,1],[2,2]]
print np.matmul(a,b)
```

Kết quả :

Ví dụ 2 :

```
# 2-D mixed with 1-D
import numpy.matlib
import numpy as np

a = [[1,0],[0,1]]
b = [1,2]
print np.matmul(a,b)
print np.matmul(b,a)
```

Kết quả :

Ví dụ 3 :

```
# one array having dimensions > 2
import numpy.matlib
import numpy as np

a = np.arange(8).reshape(2,2,2)
b = np.arange(4).reshape(2,2)
```

```
print np.matmul(a,b)
```

Kết quả :

```
[[[2   3]
   [6   11]]
  [[10  19]
   [14  27]]]
```

5. Determinant (Định thức) :

Định thức là một giá trị rất hữu ích trong đại số tuyến tính. Nó được tính toán từ các phần tử đường chéo của một ma trận vuông. Đối với ma trận 2x2, nó chỉ đơn giản là phép trừ tích của phần tử trên cùng bên trái và dưới cùng bên phải với tích của hai phần tử còn lại.

Nói cách khác, đối với ma trận [[a, b], [c, d]], định thức được tính là ‘ad-bc’. Ma trận vuông lớn hơn được coi là hợp của ma trận 2x2.

Hàm numpy.linalg.det () tính toán định thức của ma trận đầu vào.

Ví dụ 1 :

```
import numpy as np
a = np.array([[1,2], [3,4]])
print np.linalg.det(a)
```

Kết quả :

Ví dụ 2 :

```
import numpy as np

b = np.array([[6,1,1], [4, -2, 5], [2,8,7]])
print b
```

```
print np.linalg.det(b)
print 6*(-2*7 - 5*8) - 1*(4*7 - 5*2) + 1*(4*8 -
-2*2)
```

Kết quả ;

```
[[ 6 1 1]
 [ 4 -2 5]
 [ 2 8 7]]

-306.0

-306
```

6. numpy.linalg.solve()

Hàm numpy.linalg.solve () cung cấp nghiệm của phương trình tuyến tính ở dạng ma trận.

Xét các phương trình tuyến tính sau:

x + y + z = 6

2y + 5z = -4

2x + 5y - z = 27

Nếu ba ma trận này được gọi là A, X và B, phương trình sẽ trở thành:

7. numpy.linalg.inv()

Sử dụng hàm numpy.linalg.inv () để tính nghịch đảo của ma trận. Nghịch đảo của ma trận sao cho nếu nó được nhân với ma trận ban đầu, nó sẽ được kết quả là ma trận đơn vị.

Ví dụ 1 :

```
import numpy as np
```

```
x = np.array([[1,2],[3,4]])
y = np.linalg.inv(x)
print x
print y
print np.dot(x,y)
```

Kết quả :

```
[[1 2]
 [3 4]]
[[-2.   1. ]
 [ 1.5 -0.5]]
[[  1.00000000e+00   1.11022302e-16]
 [  0.00000000e+00   1.00000000e+00]]
```

Ví dụ 2 :

```
import numpy as np
a = np.array([[1,1,1],[0,2,5],[2,5,-1]])

print 'Array a:”
print a
ainv = np.linalg.inv(a)

print 'Inverse of a:'
print ainv

print 'Matrix B is:'
b = np.array([[6],[-4],[27]])
print b

print 'Compute A-1B:'
x = np.linalg.solve(a,b)
```

```
print x
# this is the solution to linear equations x =
5, y = 3, z = -2
```

Kết quả :

```
Array a:
[[ 1 1 1]
 [ 0 2 5]
 [ 2 5 -1]]

Inverse of a:
[[ 1.28571429 -0.28571429 -0.14285714]
 [-0.47619048 0.14285714 0.23809524]
 [ 0.19047619 0.14285714 -0.0952381 ]]

Matrix B is:
[[ 6]
 [-4]
 [27]]

Compute A-1B:
[[ 5.]
 [ 3.]
 [-2.]]
```

Kết quả tương tự bằng cách sử dụng hàm -