NumPy and Matplotlib

http://www.engr.ucsb.edu/~shell/che210d/numpy.pdf

Overview

NumPy (Numeric Python) เป็นโมดูลส่วนเสริมของ Python ที่มีฟังก์ชันเกี่ยวกับคณิตศาสตร์และการคำนวณต่างๆ มาให้ใช้งาน โดยทั่วไปจะเกี่ยวกับการจัดการข้อมูลชุด (Array) ขนาดใหญ่และเมทริกซ์

NumPy นี้ครบคลุมการคำนวณมากมายสามารถทำงานได้ใกล้เคียงกับ commercial software เช่น MatLab เลยทีเดียว

เนื่องจาก NumPy มีความสามารถมากในรายวิชานี้เราจะเรียนเกี่ยวกับ Vector and matrix mathematics

การ Import NumPy module

มีหลายวิธีในการ import NumPy เข้ามาใช้งาน โดยทั่วไปวิธีมาตรฐานได้แก่คำสั่ง

```
import numpy
```

อย่างไรก็ตามหากมีการเรียกใช้งาน NumPy บ่อยครั้ง แต่ละครั้งจะต้องพิมพ์ numpy.X ดังนั้นเพื่อความสะดวก เราจะย่อเวลาเรียกเป็น np เราจะเปลี่ยนการ import เป็น

```
import numpy as np
```

ทำให้เราเรียกใช้งานฟังก์ชันได้โดยเขียนเป็น np.X

Arrays

Arrays เป็นคุณสมบัติหลักของ NumPy มีลักษณะคล้ายกับ list ยกเว้น สมาชิกทุกตัวใน array จะต้องเป็นข้อมูลชนิดเดียวกัน โดยทั่วไปแล้วข้อมูลที่เก็บจะเป็นตัวเลขเช่น int หรือ float

Arrays มีความสามารถในการดำเนินการเกี่ยวกับข้อมูลที่เป็นตัวเลขจำนวนมากๆ ได้อย่างรวดเร็วและมีประสิทธิภาพมากกว่า list

เราจะนำ Array นี้มาสร้างเป็น Vectors และ Matrices

Vector and Matrix

- Vector เป็นลำดับของตัวเลขที่เขียนในรูป

$$u = \begin{pmatrix} u_1 \\ u_2 \\ \dots \\ u_n \end{pmatrix}$$

ตัวอย่างเช่น $\begin{pmatrix} 1 \\ -2 \end{pmatrix}, \begin{pmatrix} 0.6 \\ -4 \\ 9 \end{pmatrix}$

- Matrix เป็นแถวลำดับสี่เหลี่ยมของตัวเลขซึ่งเขียนในรูป

$$\mathrm{A} = \begin{pmatrix} a_{11} & \cdots & a_{1n} \\ \vdots & \ddots & \vdots \\ a_{m1} & \cdots & a_{mn} \end{pmatrix}$$

ตัวอย่างเช่น $\begin{pmatrix} 1 & 2 \\ 3 & 4 \end{pmatrix}$

การสร้าง array

เราจะสร้าง Vector และ Matrix ได้โดยการใช้ Array

Array สามารถถูกสร้างได้จาก list

```
import numpy as np

a = np.array([1,2,4,8],float)

print(a)

type(a)
```

```
[ 1.  2.  4.  8.]

<class 'numpy.ndarray'>
```

Array 1 มิติ เทียบได้กับ Vector นั่นเอง

การสร้างและเข้าถึงข้อมูลใน array

ในการสร้าง array จะรับค่า 2 ค่าได้แก่

- list ที่ต้องการเปลี่ยนเป็น array
- ชนิดของสมาชิกที่ต้องการสร้างเป็น array

สมาชิกของ array จะถูกเข้าถึง แบ่งและจัดการได้เช่นเดียวกับ list เช่น

```
import numpy as np
a = np.array([1,2,4,8], float)
print(a[3])                    8.0
a[0] = 5
print(a)                       [ 5.  2.  4.  8.]
```

การเข้าถึงข้อมูลใน array หลายมิติ

Array สามารถทำให้เป็นหลายมิติได้ ไม่เหมือนกับ list การสามารถเข้าถึงข้อมูลในแกนที่แตกต่างกันทำได้โดยใช้ comma ภายในปีกกา

Array 2 มิติ เทียบได้กับ Matrix นั่นเอง

```
import numpy as np

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

print(a)
```

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

```
print(a[0,0])
```

```
1.0
```

```
print(a[0,1])
```

```
2.0
```

การ Transpose

การ transpose array สามารถทำได้โดยใช้คำสั่ง transpose() ซึ่งจะเป็นการสร้าง array ใหม่

```
import numpy as np

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

print(a)

b = a.transpose()

print(b)
```

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

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

การทำ array ให้เหลือมิติเดียว

Array หลายมิติสามารถทำให้เหลือมิติเดียวได้โดยใช้ฟังก์ชัน flatten()

```
import numpy as np

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

print(a)

a = a.flatten()

print(a)
```

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

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

การสร้าง array แบบอื่น

การสร้าง array แบบอื่น

ใช้ฟังก์ชัน arange ซึ่งคล้ายกับฟังก์ชัน range แต่คืนค่าเป็น array

```
import numpy as np

a = np.arange(5,dtype=float)

print(a)
```

```
[ 0.  1.  2.  3.  4.]
```

```
b = np.arange(1,6,2, dtype=float)

print(b)
```

```
[ 1.  3.  5.]
```

Array mathematics

การดำเนินการทางคณิตศาสตร์ที่ทำกับ array โดยทั่วไปจะทำกับสมาชิกทีละตัว นั่นหมายความว่า array ควรจะมีขนาดเท่ากันระหว่างที่มีการบวก, ลบ หรือการดำเนินการอื่นๆ

```
import numpy as np
a = np.array([1,2,3],dtype=float)
b = np.array([5,1,8],dtype=float)
print(a+b)
print(a-b)
print(a*b)
print(a/b)
print(a%b)
print(b**a)
```

```
[ 6.  3. 11.]
[-4.  1. -5.]
[ 5.  2. 24.]
[ 0.2    2.     0.375]
[ 1.  0.  3.]
[  5.   1. 512.]
```

Array mathematics

สำหรับ array 2 มิติ การคูณจะเป็นการคูณตัวต่อตัว ไม่ใช่การคูณเมทริกซ์ สำหรับการคูณเมทริกซ์จะมีอธิบายภายหลัง

```
import numpy as np

a = np.array([[1,2],[3,4]],dtype=float)

b = np.array([[2,0],[1,3]],dtype=float)

print(a*b)
```

```
[[ 2.  0.]
 [ 3. 12.]]
```

Array mathematics

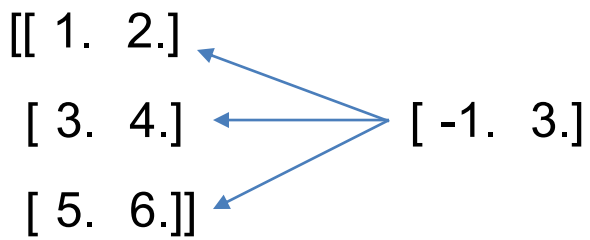
อย่างไรก็ตามหาก array ที่มิติไม่สอดคล้องกันจะถูก broadcast แทน นั้นหมายความว่า array ขนาดเล็กจะถูกทำซ้ำ ตัวอย่าง

```
import numpy as np

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

b = np.array([-1,3],dtype=float)

print(a+b)
```




```
[[ 0.  5.]
 [ 2.  7.]
 [ 4.  9.]]
```

- ซึ่งในที่นี้ array b ที่มีขนาด 1 มิตินั้นจะถูก broadcast ไปยัง array 2 มิติที่ขนาดสอดคล้องกับ a นั่นคือ b จะถูก ทำซ้ำในแต่ละ item ของ a ราวกับว่า ทำงานกับ [[-1. 3.], [-1. 3.], [-1. 3.]] อยู่

Array mathematics

นอกจากการดำเนินการพื้นฐานต่างๆ NumPy ยังมีฟังก์ชันทางคณิตศาสตร์ให้ใช้งานอีกมากมากซึ่งจะทำงานกับสมาชิกใน array ทีละตัว เช่น

abc, sign, sqrt, log, log10, exp, sin, cos, tan, arcsin, arccos, arctan, sinh, cosh, tanh, arcsinh, arccosh, arctanh, floor, ceil, rint, pi, e (np.pi, np.e)

```
import numpy as np
a = np.array([1,4,9],dtype=float)
print(np.sqrt(a))
```

```
[ 1.  2.  3.]
```

Basic array operation

มีหลายฟังก์ชันที่ทำงานกับทั้ง array

ข้อมูลใน array สามารถหาผลรวมหรือคูณกันทั้งหมดได้

```
import numpy as np

a = np.array([2,4,3], float)

print(a.sum())                                  9.0

print(a.prod())                                 24.0

c = np.array([[0,-2],[3,-1],[3,-5]], float)

print(c.sum())                                  -2
```

หรือใช้ np.sum(a) np.prod(a) แทนได้เช่นกัน

Basic array operation

มีฟังก์ชันสำหรับคำนวณค่าทางสถิติที่ทำกับชุดข้อมูลที่เก็บใน array เช่น mean, variance และ standard deviation โดยเรียกด้วย mean(), var() และ std() ตามลำดับ

```
import numpy as np

a = np.array([2,1,9], float)

print(a.mean())
```

4.0

```
print(a.var())
```

12.6666666667

```
print(a.std())
```

3.55902608401

Basic array operation

max() ฟังก์ชันสำหรับการหาค่ามากที่สุด

min() ฟังก์ชันสำหรับการหาค่าน้อยที่สุด

argmax() ฟังก์ชันสำหรับการหาตำแหน่งของค่าที่มากที่สุด

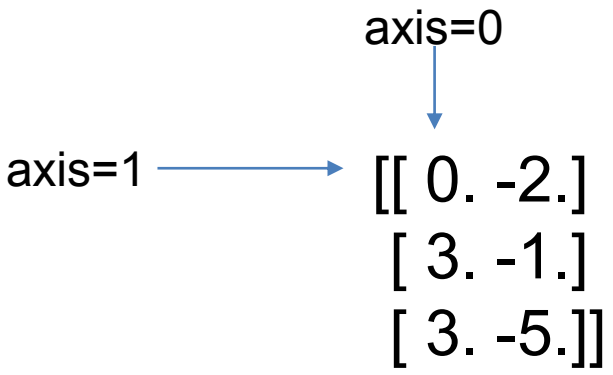
argmin() ฟังก์ชันสำหรับการหาตำแหน่งของค่าที่น้อยที่สุด

```
import numpy as np
a = np.array([2,1,9], float)
print(a.max())
print(a.min())
print(a.argmax())
print(a.argmin())
```

```
9.0
1.0
2
1
```

Basic array operation

สำหรับ array หลายมิติแต่ละฟังก์ชันที่อธิบายก่อนหน้าสามารถระบุแกนที่ต้องการทำการดำเนินการได้ ตัวอย่างเช่น




```
import numpy as np

a = np.array([[0,-2],[3,-1],[3,-5]], float)

print(a.mean(axis=0))
print(a.mean(axis=1))
print(a.min(axis=1))
print(a.max(axis=0))
```

```
[ 2.         -2.66666667]
[-1.  1. -1.]
[-2. -1. -5.]
[ 3. -1.]
```

การเรียงลำดับข้อมูลใน array 1 มิติ

หาต้องการให้ array 1 มิติเรียงลำดับใช้คำสั่ง sorted()

```
import numpy as np

a = np.array([6,4,8,2,9], float)

b = sorted(a)

print(b)
```

[2.0, 4.0, 6.0, 8.0, 9.0]

Vector and Matrix mathematics

NumPy มีฟังก์ชันหลายฟังก์ชันให้ใช้งานเกี่ยวกับการดำเนินการทางคณิตศาสตร์ที่เกี่ยวกับเวกเตอร์และเมทริกซ์ ได้แก่

- dot product (คูณเมทริกซ์)
- inner, outer, cross product
- determinant
- eigenvalues และ eigenvectors
- inverse ของเมทริกซ์

dot product

- ตัวอย่าง $a = \begin{bmatrix} 1 & 2 \\ -1 & 0 \\ 3 & 2 \end{bmatrix}, b = \begin{bmatrix} 1 & 5 & 2 \\ -2 & 0 & 1 \end{bmatrix}$

ต้องการหา $ab = ?$

$$ab = \begin{bmatrix} -3 & 5 & 4 \\ -1 & -5 & -2 \\ -1 & 15 & 8 \end{bmatrix}$$

ตัวอย่างการใช้งาน dot product

```
import numpy as np

a = np.array([[1,2],[-1,0],[3,2]])

b = np.array([[1,5,2],[-2,0,1]])

c = np.dot(a,b)

print(c)
```

```
[[-3  5  4]
 [-1 -5 -2]
 [-1 15  8]]
```

ตัวอย่าง inner, outer, cross product

```
import numpy as np

a = np.array([1,4,0],float)

b = np.array([2,2,1],float)

x = np.outer(a,b)

print(x)

y = np.inner(a,b)

print(y)

z = np.cross(a,b)

print(z)
```

```
[[ 2.  2.  1.]
 [ 8.  8.  4.]
 [ 0.  0.  0.]]

10.0

[ 4. -1. -6.]
```

Linear algebra function

นอกจากนี้ NumPy ยัง build-in ฟังก์ชันสำหรับแก้ปัญหาเกี่ยวกับ linear algebra ที่เกี่ยวกับเมทริกซ์ แต่ต้องเรียกผ่าน sub module linalg ซึ่ง ฟังก์ชันที่สามารถเรียกใช้ได้แก่

- det() สำหรับหา determinant
- inv() สำหรับหา inverse ของเมทริกซ์

ตัวอย่างการใช้งาน Linear algebra function

```
import numpy as np

a = np.array([[4,2,0],[9,3,7],[1,2,1]])

x = np.linalg.det(a)

print(x)
```

```
-48.0
```

```
y = np.linalg.inv(a)

print(y)
```

```
[[ 0.22916667  0.04166667 -0.29166667]
 [ 0.04166667 -0.08333333  0.58333333]
 [-0.3125      0.125       0.125     ]]
```

z = np.dot(a,y) คำถาม Z ควรมีค่าเป็นเท่าไร