BÀI 5: ĐỊNH THỨC, MA TRẬN VÀ ỨNG DỤNG

Mục tiêu:

- *Ôn luyện về các lệnh xử lý ma trận và định thức trong gói numpy.*
- *Một số ứng dụng của định thức và ma trận.*
- *Giới thiệu về sai số, bình phương cực tiểu (giải hệ có số phương trình nhiều hơn ẩn)*

Nội dung chính:

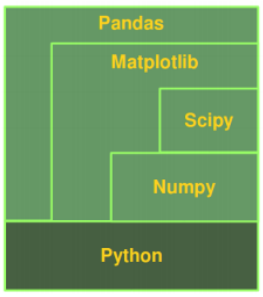
1. Sử dụng array và matrix trong numpy

*Dưới đây là tóm gọn một số lưu ý khi sử dụng kiểu **matrix** và **array** trong gói NumPy:*

1.1. Thông tin tóm gọn

- Numpy: là thư viện phần mềm hướng đến xây dựng kiểu dữ liệu, hàm xử lý/tính toán trên dữ liệu.
- Scipy: hướng đến các thuật toán toán học và các hàm thuận tiện trên nền tảng numpy.
- Sympy: hướng đến tính toán hình thức.
- Pandas: hướng đến phân tích chuỗi số ().

Một số lưu ý đặc biệt về dấu **[…]** và **(…)** khi sử dụng gói numpy:



Kí hiệu	Kí hiệu tương tự	Ý nghĩa
a(end)	a[-1]	Phần tử cuối của một vector
a(2,5)	a[1,4]	Phần tử của dòng 2, cột 5
a(2,:)	a[1] hoặc a[1,:]	Lấy toàn bộ dòng thứ 2
a(1:5,:)	a[0,5] hoặc a[:5] hoặc a[0:5,:]	Lấy 5 dòng đầu tiên của ma trận a
a(end-4:end.")	a[-5:]	5 phần tử dòng cuối cùng

Các dạng sao chép mảng dữ liệu:

TT	Lệnh sao chép	Minh họa
1	Tạo 1 tham chiếu	B = A
2	Tạo một bản sao chép hoàn chỉnh	B = np.copy(A) B = A.copy()
3	Tạo một phiên bản copy từ dãy có sẵn	np.copyto(B,A) B[:] = a

Minh họa:

```
>>> D = np.array([[1,2],[3,4]])
>>> np.copyto(E, D)
…… ← sinh viên ghi nhận kết quả: lỗi gì (nếu có)?
>>> E = np.array([[1,2],[3,5]])
>>> np.copyto(E, D)
…… ← sinh viên ghi nhận kết quả
>>> print(E)
…… ← sinh viên ghi nhận kết quả
```

1.2. Nên và không nên sử dụng kiểu dữ liệu numpy.matrix

Numpy hỗ trợ 2 kiểu dữ liệu để thể hiện ma trận, đó là numpy.array và numpy.matrix. Dưới đây là một số phân tích để hiểu hơn khi sử dụng kiểu dữ liệu numpy.matrix:

- *Nên sử dụng kiểu numpy.matrix:*

- Thuận tiện vì được hỗ trợ tất cả các hàm xử lý đại số tuyến tính.
- Hàm dựng uyển chuyển (tương thích với người sử dụng Matlab – một ngôn ngữ lập trình cũng rất mạnh):

Ví dụ: >>> B = np.matrix("[1 2 3; 4 5 6]")

- Có đầy đủ các hàm **.H**, **.I** và **.A** hỗ trợ tính toán/xử lý nâng cao về đại số.
- Dữ liệu chỉ có ở dạng 2D. Tuy nhiên, mỗi phần tử có thể là một đối tượng. Ví dụ:

```
>>> x = np.array([['a','b'],['c','d']])
>>> y = 'x'
>>> C = np.matrix([[x,y],[1, 2]] )
>>> print (C)
…….. # Sinh viên điền kết quả
……..
……..
……..
```

- *Không nên/thể sử dụng kiểu numpy.matrix:*

- Kiểu matrix chỉ mô tả ma trận 2 chiều, không mô tả ma trận nhiều hơn 2 chiều.

```
>>> A = np.matrix([[[1,2],[3,4]],[[5,6],[7,8]]])  # ← lỗi. Sinh viên ghi nhận lỗi
……..
>>> A = np.array([[[1,2],[3,4]],[[5,6],[7,8]]])   # sinh viên ghi nhận kết quả
>>> print(A)
……..
```

- Hiện tại, nhiều gói xử lý khác có thể trả về đối tượng **array** thay vì trả về **matrix** nên khó khăn để tương thích trong cùng một hệ thống.
- Một số phép toán chồng nạp chưa được nhất quán.
- Code có thể khó đọc với người chưa biết hoặc chưa thông thạo.

1.3. Đối tượng matrix từ các hàm trong gói numpy.matlib

Để tương thích với người sử dụng Matlab, sau khi khai báo:

>>> ***from numpy import matlib***

Chúng ta có thể sử dụng một số hàm sau để tạo đối tượng dạng matrix như sau:

empty Tạo ma trận rỗng	**zeros** Tạo ma trận toàn 0	**ones** Tạo ma trận toàn 1	**eye**
identity Tạo ma trận đơn vị	**repmat**	**Rand**	**Randn**

Sinh viên thực hành các lệnh sau:

>>> from numpy import matlib

>>> G = matlib.**identity**(5) # cú pháp: matlib.identity(***n***)

>>> print (G)

……..
……..
……..
……..
……..

>>> H = matlib.randn(3,2) # cú pháp: matlib.randn(****args***)

>>> print(H)

……..
……..
……..

>>> K = matlib.zeros([4,4]) # ← cú pháp: matlib.zeros(***shape***)

>>> print(K)

……..
……..
……..

2. Ứng dụng 2 –Liên phân số và số π

2.1. Liên phân số

Liên phân số, tiếng Anh gọi là Continued Fraction, là một dạng biểu diễn các số thực (hữu tỉ và vô tỉ) dưới dạng phân số nhiều tầng. Ví dụ đơn giản là:

$$\frac{9}{7} = 1 + \frac{1}{3 + \frac{1}{2}}$$

Từ giá trị trên, chúng ta có thể biểu diễn phân số theo cách "máy tính" có thể hiểu được là: $\frac{9}{7} = [1, 3, 2]$. Người ta chứng minh được rằng cách biểu diễn này sẽ duy nhất đối với mỗi số. Lưu ý, số $\frac{18}{14}$ cũng có biểu diễn tương tự số $\frac{9}{7}$. Giá trị phân số này là giá trị hữu hạn, ta gọi đó là liên phân số hữu hạn.

2.2. Liên phân số biểu diễn số π

Một ví dụ khác với số hữu tỉ: Theo lịch sử, trước khi có máy tính, số π được xem là số mang giá trị $\frac{22}{7}$. Khi biểu diễn liên phân số giá trị $\frac{22}{7} = 3 + \frac{1}{7}$. Tuy vậy, khi toán học phát triển hơn giá trị số $\pi = 3{,}14159265$ ... và giá trị mới của π tương ứng với phân số $\frac{355}{113}$, hoặc ở dạng liên phân số, đó là:

$$\pi = 3 + \cfrac{1}{7 + \cfrac{1}{15 + \cfrac{1}{1 + \cfrac{1}{292 + \cdots}}}}$$

Từ đó, số π có thể biểu diễn thành tập các giá trị của liên phân số là: $\pi = [3, 7, 15, 1, 292, \dots]$, cụ thể viết nhiều số hơn là: $\pi = [3, 7, 15, 1, 292, 1{,}1{,}1{,}2{,}1{,}3{,}1{,}14{,}2 \dots]$. Từ biểu diễn trên, dễ dàng chúng ta có thể tìm thấy các số hữu tỉ gần với số π (tất nhiên sẽ có sai số), như: $\frac{3}{1}, \frac{22}{7}, \frac{333}{106}, \frac{355}{113}$, ... Tổng quát hơn, công thức liên phân số là:

$$\frac{p_n}{q_n} = c_0 + \cfrac{1}{c_1 + \cfrac{1}{c_2 + \ddots + \cfrac{1}{c_n}}}$$

Với liên phân số, người ta chứng minh được rằng có một cách xác định p_n và q_n như sau:

$$\begin{pmatrix} c_0 & 1 \\ 1 & 0 \end{pmatrix}\begin{pmatrix} c_1 & 1 \\ 1 & 0 \end{pmatrix} \dots \begin{pmatrix} c_n & 1 \\ 1 & 0 \end{pmatrix} = \begin{pmatrix} p_n & p_{n-1} \\ q_n & q_{n-1} \end{pmatrix}, n = 0{,}1{,}2, \dots$$

Sinh viên thực hiện tính toán số π:

```
>>> import numpy as np
>>> c = [3, 7, 15, 1, 292, 1, 1, 1, 2, 1, 3, 1, 14, 2]
>>> M = np.mat([[1,2],[3,4]])  # khởi tạo ma trận 2x2
>>> for i in range(len(c)):
        ci = np.mat([[1,1],[1,0]])
        ci[0, 0] = c[i]
        if (i==0):
            M = ci
        else:
            M = M.dot(ci)
>>> print(M)
>>> print (M[0,0]/M[1,0])  # in số Pi theo dãy số  biểu diễn liên phân số.
…….. ← sinh viên viết giá trị số Pi tính được.
```

Ngày nay, kỹ thuật liên phân số được sử dụng lưu trữ cơ sở dữ liệu. Sinh viên sẽ được tiếp cận trong các chương trình tiếp theo.

3. Về phân tích hồi quy tuyến tính

[Mục này cung cấp thêm kiến thức cho sinh viên về dạng phương trình skiny, hệ phương trình có số phương trình nhiều hơn số ẩn]

3.1. Dẫn nhập: khái niệm về sai số

Bài toán minh họa: Khi lấy cây thước dài 200mm để đo chiều dài một cái bàn 800mm, nếu càng nhiều lượt đo thì số lượng "chiều dài" khác nhau sẽ tăng lên. Mặc dù chiều dài của cái bàn và độ chính xác cây thước được xem như là 1 con số không đổi. Điều đó được giải thích là do trong mỗi lượt đo đều có **những sai số (hiệu số giữa số đo và giá trị thực)** nhất định.

Lưu ý: các sai số gây ra ở đây là hoàn toàn mang tính chất ngẫu nhiên. Sinh viên cần phân biệt dữ liệu có "sai số" với dữ liệu bị "sai lầm". Khái niệm dữ liệu "sai lầm" trong đo đạc là khi:

- Sử dụng công cụ (như thước) bị sai (như thước bị cong, độ đo của cây thước không đúng).
- Người đo cố tình đọc sai giá trị đo.
- Thực hiện phương pháp đo sai, như: đo ẩu,...

Như vậy, nếu gọi chiều dài của bàn là x thì chúng ta chỉ có 1 biến cần tìm. Trong khi đó, mỗi lượt đo được xem như một phương trình để tìm x. Do đó, chúng ta không thể xác định x một cách đơn giản bằng việc giải phương trình vì trên thực tế mỗi lượt đo có khả năng có khác biệt giá trị với nhau. Tuy vậy, chiều dài của bàn sẽ nằm ở một khoảng giá trị "hợp lý" mà không thể vượt ra ngoài được. Ví dụ: chiếc bàn dài khoảng 80cm thì nó sẽ giao động ở một giá trị gần với 80cm chứ không thể vượt lên 3 hay 5 hay 10 mét!

Từ đó, mở rộng ra, chúng ta sẽ có một dạng hệ phương trình cần xác định nghiệm mà trong đó điều quan trọng là chúng ta tìm được sự liên hệ phụ thuộc giữa các biến chứ không chỉ dừng lại ở một giá trị.

3.2. Phương pháp bình phương cực tiểu (mô hình tuyến tính)

Xét câu chuyện khởi nghiệp: 03 bạn Lan – Linh – Loan trồng một loại cây giống chất lượng cao cung cấp xuất khẩu. Trong 5 lần đo chiều dài 1 cây mẫu, bạn Lan có số liệu đo như sau:

Ngày đo	Chiều cao (cm)
1	1.0
2	2.0
3	4.0
4	4.0
5	6.0

Vấn đề đặt ra là:

- Cây mẫu phát triển như thế nào trong 5 ngày qua (để so sánh với tiêu chuẩn và cây khác).
- Dự đoán chiều cao của cây giống trong ngày thứ 6.

Câu chuyện trên bắt đầu có tranh cãi khi:

- Bạn Linh cho rằng: cây đã phát triển theo phương trình $f(x) = 0.5 + x$.
- Bạn Loan cho rằng: cây đã phát triển theo phương trình $f(x) = 1.2x$.

Hỏi bạn nào đúng, bạn nào sai? Bạn hãy giúp 3 bạn trên tìm ra quy luật gần đúng nhất cho cây.

Với ví dụ trên, chúng ta sẽ nghiên cứu về phân tích hồi quy tuyến tính theo mô hình bình phương cực tiểu . Đây là một trong những phương pháp để tìm sự tương quan giữa một biến phụ thuộc Y với một hay nhiều biến độc lập X với mô hình sử dụng hàm tuyến tính (bậc 1). Các tham số của mô hình được ước lượng từ dữ liệu. Ở đó, người ta sẽ cố gắng tìm một "đường cong" gọi là "gần" nhất so với dữ liệu có được. "Đường cong" đó gọi là xấp xỉ cho dữ liệu và cũng để "dự báo" dữ liệu mới hoặc "lấp" những dữ liệu không tìm thấy. Và trong trường hợp này "đường cong" cần tìm chính là một đường thẳng (phương trình bậc 1) có giá trị tổng bình phương sai số nhỏ nhất tương ứng với dữ liệu đầu vào.

Xét qua sai số tổng bình phương của 2 bạn Linh và Loan trong bảng sau:

Mô hình của bạn Linh: $f(x) = 0.5 + x$				Mô hình của bạn Loan: $f(x) = 1.2x$			
Ngày x_i	y_i	$f(x_i)$	$[y_i - f(x_i)]^2$	*Ngày* x_i	y_i	$f(x_i)$	$[y_i - f(x_i)]^2$
1	1	1.5	$(-0.5)^2$	1	1	1.2	$(-0.2)^2$
2	2	2.5	$(-0.5)^2$	2	2	2.4	$(-0.4)^2$
3	4	3.5	$(+0.5)^2$	3	4	3.6	$(+0.4)^2$
4	4	4.5	$(-0.5)^2$	4	4	4.8	$(-0.8)^2$
5	6	5.5	$(+0.5)^2$	5	6	6.0	$(0.0)^2$
		Tổng: 1.25				Tổng: 1.00	

Nhìn vào bảng tính, chúng ta nhận thấy rằng bạn Loan có vẻ đúng hơn bạn Linh khi sai số nhỏ hơn (1.0 < 1.25). Tuy vậy, vấn đề đặt ra là sự tồn tại của một mô hình toán học cụ thể là mô hình tuyến tính (đường thẳng) đã được chứng minh mà có sai số nhỏ nhất hay không?

Dưới đây là **phương pháp hồi quy tuyến tính bằng bình phương cực tiểu**:

Định nghĩa:

Với tập gồm n bộ số $(x_1, y_1), (x_2, y_2), \dots, (x_n, y_n)$ cho trước. Hãy tìm một phương trình bậc 1:

$$f(x) = a_0 + a_1 x$$

để giá trị tổng bình phương các sai số là cực tiểu:

$$\min[y_1 - f(x_1)]^2 + [y_2 - f(x_2)]^2 + \cdots + [y_n - f(x_n)]^2$$

Phương pháp giải:

Có thể xem các giá trị y_i là những giá trị đo được, còn các giá trị $f(x_i)$ là những giá trị

Chúng ta có các biến đổi như sau:

$$y_1 = f(x_1) + [y_1 - f(x_1)]$$
$$y_2 = f(x_2) + [y_2 - f(x_2)]$$
$$\dots$$
$$y_n = f(x_n) + [y_n - f(x_n)]$$

Đặt $e_i = [y_i - f(x_i)]$ và thay $f(x_i) = (a_0 + a_1 x_i)$, chúng ta sẽ có hệ sau:

$$y_1 = (a_0 + a_1 x_1) + e_1$$
$$y_2 = (a_0 + a_1 x_2) + e_2$$
$$\dots$$
$$y_n = (a_0 + a_1 x_n) + e_n$$

Đặt:

$$Y = \begin{bmatrix} y_1 \\ y_2 \\ \dots \\ y_n \end{bmatrix}, X = \begin{bmatrix} 1 & x_1 \\ 1 & x_2 \\ \dots & \dots \\ 1 & x_n \end{bmatrix}, A = \begin{bmatrix} a_0 \\ a_1 \end{bmatrix}, E = \begin{bmatrix} e_1 \\ e_2 \\ \dots \\ e_n \end{bmatrix}$$

Khi đó, chúng ta có hệ sau:

$$Y = XA + E$$

Bằng phương pháp toán, người ta chứng minh được rằng, với hệ này, chúng ta sẽ được nghiệm

$$\boldsymbol{A = (X^T X)^{-1} X^T Y}$$

và giá trị tổng bình phương sai số là: $\boldsymbol{E^T E}$.

Lưu ý 1: việc chứng minh 2 công thức trên sinh viên sẽ được học trong các môn học khác. Sinh viên chỉ cần hiểu và áp dụng công thức để giải.

Áp dụng giải: Từ đó, sinh viên thực hiện việc giải hệ trên:

$$X = \begin{bmatrix} 1 & 1 \\ 1 & 2 \\ 1 & 3 \\ 1 & 4 \\ 1 & 5 \end{bmatrix}, Y = \begin{bmatrix} 1 \\ 2 \\ 4 \\ 4 \\ 6 \end{bmatrix}$$

Ta có:

$$A = (X^T X)^{-1} X^T Y = \left(\begin{bmatrix} 1 & 1 & 1 & 1 & 1 \\ 1 & 2 & 3 & 4 & 5 \end{bmatrix} \begin{bmatrix} 1 & 1 \\ 1 & 2 \\ 1 & 3 \\ 1 & 4 \\ 1 & 5 \end{bmatrix} \right)^{-1} \begin{bmatrix} 1 & 1 & 1 & 1 & 1 \\ 1 & 2 & 3 & 4 & 5 \end{bmatrix} \begin{bmatrix} 1 \\ 2 \\ 4 \\ 4 \\ 6 \end{bmatrix} = \begin{bmatrix} -0.2 \\ 1.2 \end{bmatrix} = \begin{bmatrix} a_0 \\ a_1 \end{bmatrix}$$

Như vậy, phương trình đường thẳng gần đúng nhất sẽ là: $\boldsymbol{y = -0.2 + 1.2x}$ và sai số là 0.8!

Sinh viên điền các lệnh Python tương ứng:

- Nhập X, Y và tính X^T

>>> X = np.array(……)

>>> Y = np.array(……)

>>> XT = ……

- Tính toán cơ bản các giá trị các giá trị $A1 = (X^T X)^{-1}$ và $A2 = X^T Y$

>>>……

>>>……

- Tính giá trị $A = A1.A2$ và sinh viên suy ra phương trình $f(x) = a_0 + a_1 x$ với $A = \begin{bmatrix} a_0 \\ a_1 \end{bmatrix}$

>>> A = A1.dot(A2)

……

Suy ra phương trình và tính các f_i

Để từ đó tính ra sai số bằng tổng của các bình phương sai số $(f_i - y_i)$.

BÀI TẬP CHƯƠNG 5

Bài 1) Một người đi xạ trị bệnh ung thư bằng việc bơm thuốc vào người. Khi vừa xạ trị xong, lượng thuốc tồn động trong người ở mức 10 đơn vị. Sau đó 1 giờ sau lượng thuốc còn lại là 8; lần lượt 2, 3, 4 giờ sau lượng thuốc còn lại là 7, 5 và 2.

Hãy viết các lệnh bằng Python để tìm phương trình tuyến tính bằng phương pháp bình phương cực tiểu với tập các cặp số (x, y) để tìm công thức giảm lượng thuốc đối với bệnh nhân theo thời gian:

$$(x, y) = \{(0, 10), (1, 8), (2, 7), (3, 5), (4, 2)\}$$

[*Kết quả tham khảo để kiểm chứng: Phương trình tuyến tính là:* $y = 12.1 - 1.9x$]

Hãy giải lại các BT lý thuyết sau đây (đã làm từ trước) và vẽ đồ thị tương ứng.

Bài 2) Cho mô hình $y = a_0 + a_1x + a_2x^2$. Tìm các tham số a_0, a_1 và a_2 để mô hình phù hợp với từng bộ dữ liệu sau.

a)

x	2	3	5	6
y	0	-10	-48	-76

b)

x	1	0	1	2
y	-2	-1	0	4

Bài 3) Cho mô hình $y = a + \frac{b}{x}$. Tìm các tham số a và b để mô hình phù hợp với bộ dữ liệu sau:

x	1	3	6
y	7	3	1

Bài 4) Cho mô hình $y = a + b\sqrt{x}$. Tìm các tham số a và b để mô hình phù hợp với bộ dữ liệu sau:

x	3	7	10
y	1.5	2.5	3