Hướng dẫn Lab 10

Exercise 1

Exercise 1: Find the eigenvalues of

(a)

$$A = \begin{pmatrix} -1 & 3.5 & 14 \\ 0 & 5 & -26 \\ 0 & 0 & 2 \end{pmatrix}$$

(b)

$$B = \begin{pmatrix} -2 & 0 & 0 \\ 99 & 0 & 0 \\ 10 & -4.5 & 10 \end{pmatrix}$$

(c)

$$C = \begin{pmatrix} 5 & 5 & 0 & 2 \\ 0 & 2 & -3 & 6 \\ 0 & 0 & 3 & -2 \\ 0 & 0 & 0 & 5 \end{pmatrix}$$

(d)

$$D = \begin{pmatrix} 3 & 0 & 0 & 0 \\ 6 & 2 & 0 & 0 \\ 0 & 3 & 6 & 0 \\ 2 & 3 & 3 & -5 \end{pmatrix}$$

(e)

$$E = \begin{pmatrix} 3 & 0 & 0 & 0 & 0 \\ -5 & 1 & 0 & 0 & 0 \\ 3 & 8 & 0 & 0 & 0 \\ 0 & -7 & 2 & 1 & 0 \\ -4 & 1 & 9 & -2 & 3 \end{pmatrix}$$

Then, compute the determinant of A, B, C, D, E matrix based on the eigenvalues.

Hàm numpy.linalg.eig(A): tìm eigenvalues (giá trị riêng) và eigenvectors (vector riêng) của ma trận A.

==> Kết quả trả về của hàm là một tuple gồm 2 phần tử:

- eigenvalues của ma trận.
- eigenvectors của ma trận (mỗi eigenvector tương ứng với một eigenvalue).

Công thức tính det(A) dựa vào eigenvalues của ma trận A: det(A) = λ1 * λ2 * ... * λn

với λ1, λ2, ..., λn là các eigenvalue của ma trận A.

```
# Tìm eigenvalues và eigenvector của ma trận A
eigenvalues, v = np.linalg.eig(A)

# eigenvalues: mảng chứa các eigenvalues của ma trận A.

# v: ma trận mà mỗi cột là 1 eigenvector tương ứng với eigenvalue trong mảng eigenvalues.

# Có thể dùng hàm prod để tính det của ma trận dựa vào eigenvalues như sau:
det_A = np.prod(eigenvalues)
```

Exercise 2

Exercise 2: Let

$$A = \begin{pmatrix} -6 & 28 & 21 \\ 4 & -15 & -12 \\ 8 & a & 25 \end{pmatrix}$$

For each a in the set { 32, 31.9, 31.8, 32.1, 32.2}, compute the characteristic of A and the eigenvalues. In each case, create a graph of the characteristic polynomial $p(t) = det(A - \lambda I)$ for $0 \leq t \leq 3$. If possible, construct all graphs on one coordinate system.

- Tìm p(t) - đa thức đặc trưng của A ứng với từng giá trị **a** thuộc tập {32, 31.9, 31.8, 32.1, 32.2}.
- Vẽ đồ thị của p(t) với t thuộc [0, 3] ứng với mỗi giá trị của a.
- Tính giá trị riêng eigenvalues của ma trận A ứng với mỗi giá trị của a.

Hàm np.poly(): Tìm đa thức đặc trưng (characteristic polynomial) của ma trận vuông

Kết quả: mảng 1 chiều chứa hệ số của đa thức đặc trưng, xếp theo thứ tự giảm dần của bậc ($ax^k+bx^{k-1}+\ldots$)

Exercise 3

Exercise 3: Let $M \begin{bmatrix} -3 & -5 & -7 \\ -2 & 1 & 0 \\ 1 & 5 & 5 \end{bmatrix}$

(a) Use any appropriate software to find the eigenvalues of M

(b) For each eigenvalue λ found above, find the corresponding eigenvector **v** of **M** by using row reduction to solve $(M - \lambda I)v = 0$

(c) Construct a matrix **P** whose columns are the eigenvectors of **M**. Compute the product $\mathbf{D}=P^{-1}MP$ and confirm that **D** is diagonal. Compute the determinants of **D** and **M**, then confirm that they are equal.

(d) Construct another matrix **Q** whose columns are also the eigenvectors of **M** but this time place them in a different order than in **P** (for example, the first of columns of the eigenvectors of M is the second column of Q) and then compute $Q^{-1}MQ$ again. What has changed?

#3a,3b: Dùng hàm numpy.linalg.eig() tương tự Exercise 1

IGNORE #3c, #3d

Exercise 4

Exercise 4: Let $A = \begin{bmatrix} -2 & 2 & -3 \\ 2 & 1 & -6 \\ -1 & -2 & 0 \end{bmatrix}$

Use any appropriate software to find the eigenvalues and corresponding eigenvectors of A, A^T and A^{-1}. What do you observe ?

Tương tự Exercise 1: dùng hàm np.linalg.eig()

Exercise 5

Exercise 5: Let $A_1 = \begin{bmatrix} 4 & -5 \\ 2 & -3 \end{bmatrix}, A_2 = \begin{bmatrix} 0 & 2 \\ 0 & 1 \end{bmatrix}, A_3 = \begin{bmatrix} 2 & 3 \\ 1 & 4 \end{bmatrix} A_4 = \begin{bmatrix} 1 & 2 & -2 \\ -2 & 5 & -2 \\ -6 & 6 & -3 \end{bmatrix}$

$$A_5 = \begin{bmatrix} 1 & 2 & 3 & 4 \\ 5 & 6 & 7 & 8 \\ 9 & 10 & 11 & 12 \\ 13 & 14 & 15 & 16 \end{bmatrix}$$

Use any appropriate software to verify the matrices above are diagonalizable or not.

Một ma trận A (n×n) được gọi là có thể chéo hóa (**diagonalizable**) khi:

A = P * D * P^{-1}

với D là ma trận chéo (diagonal) n×n chứa các eigenvalues của A

và P là ma trận (n×n) matrix chứa các eigenvectors

Hàm np.diag(w): Tạo ma trận đường chéo từ **w**.

- Nếu **w là mảng 1 chiều**, hàm tạo ma trận đường chéo với các phần tử trên đường chéo là các phần tử của **w**.
- Nếu **w là mảng 2 chiều**, hàm trả về đường chéo của **w**.

Ví dụ:

```
w=np.array([1,2,3])
p=np.diag(w)
print(p)
```

```
[[1 0 0]
 [0 2 0]
 [0 0 3]]
```

```
w=np.array([[1,2,3],
            [4,5,6],
            [7,8,9]])
p=np.diag(w)
print(p)
```

```
[1 5 9]
```

Exercise 6

Exercise 6: Find the eigenvectors of the matrix $A = \begin{bmatrix} 1 & 2 & -2 \\ 0 & 3 & -2 \\ 0 & 0 & 1 \end{bmatrix}$ Using it, construct a matrix **P** that diagonalizes **A**. Compute $P^{-1}AP$

Tương tự Exercise 1 => Xác định P^(-1)*A*P

Vector riêng P của ma trận A là các vector mà khi ma trận A nhân với nó, kết quả thu được là một bản sao tỉ lệ của vector đó.

```
A=
 [[2 4 1]
 [2 0 3]
 [0 0 1]]
P=
 [[ 0.89442719 -0.70710678 -0.82055272]
 [ 0.4472136   0.70710678  0.06311944]
 [ 0.          0.          0.56807496]]
A*P=
 [[ 3.57770876  1.41421356 -0.82055272]
 [ 1.78885438 -1.41421356  0.06311944]
 [ 0.          0.          0.56807496]]
```

Exercise 7

Exercise 7: Find the singular values of the matrices

$$A_1 = \begin{bmatrix} 1 & 0 \\ 0 & -3 \end{bmatrix}, A_2 = \begin{bmatrix} -5 & 0 \\ 0 & 0 \end{bmatrix}, A_3 = \begin{bmatrix} \sqrt{6} & 1 \\ 0 & \sqrt{6} \end{bmatrix}, A_4 = \begin{bmatrix} \sqrt{3} & 2 \\ 0 & \sqrt{3} \end{bmatrix}$$

=> Tìm singular values của ma trận (Singular Value Decomposition - SVD):

Hàm np.linalg.svd(a, full_matrices=True, compute_uv=True, hermitian=False)

Trong đó:

a: ma trận đầu vào.

full_matrices: tham số tùy chọn (**mặc định là True**) => xác định kích thước của U và V.

full_matrices=True: kích thước của U và V bằng kích thước của ma trận đầu vào.

full_matrices=False: U và V có kích thước được thu gọn bằng số hàng (cho U) và số cột (cho V) tương ứng với số giá trị đặc biệt khác 0.

compute_uv: tham số tùy chọn (**mặc định là True**) xác định có cần tính U và V không.

compute_uv=True: hàm trả về ma trận U, mảng giá trị đặc biệt và ma trận chuyển vị V.

compute_uv=False: hàm chỉ trả về mảng giá trị đặc biệt.

hermitian: tham số tùy chọn (**mặc định là False**) xác định ma trận đầu vào có phải là Hermitian (đối xứng chéo) hay không.

hermitian=True: hàm tối ưu hóa tính toán cho ma trận đối xứng chéo.

Ví dụ: np.linalg.svd(A1, compute_uv=False): chỉ quan tâm đến giá trị đặc biệt, không cần tìm U, V.
=> Kết quả là mảng chứa các giá trị đặc biệt của ma trận A1.

Singular Value Decomposition - SVD của ma trận là phép phân tích ma trận thành 3 phần: **ma trận U, ma trận Sigma và ma trận V**:

- **Sigma** là ma trận chéo, trong đó các phần tử trên đường chéo chính được gọi là giá trị đặc biệt. Các giá trị đặc biệt là các số không âm và thể hiện mức độ quan trọng của các thành phần trong ma trận ban đầu. Các giá trị đặc biệt được sắp xếp theo thứ tự giảm dần.
- **U và V** là ma trận chứa các vector cột gọi là vector đặc biệt. Các vector đặc biệt tương ứng với các giá trị đặc biệt, nó cung cấp thông tin về hướng và độ lớn của các thành phần quan trọng trong ma trận ban đầu.

SVD có rất nhiều ứng dụng quan trọng trong xử lý tín hiệu, thị giác máy tính, nén dữ liệu, phân tích dữ liệu và nhiều lĩnh vực khác. SVD còn giúp thu giảm số chiều của dữ liệu, tìm kiếm các thành phần quan trọng và khám phá cấu trúc ẩn trong dữ liệu.

Ngoài ra, SVD cũng có vai trò quan trọng trong việc giải phương trình tuyến tính, tính toán ma trận nghịch đảo và giải quyết các vấn đề liên quan đến ma trận. **SVD** cho phép giải phương trình tuyến tính hiệu quả và ổn định, đặc biệt là trong trường hợp ma trận hệ số là ma trận không vuông hoặc không có nghịch đảo.

Ví dụ áp dụng SVD để giải hệ pttt:

```
A = np.array([[1, 2],
              [3, 4],
              [5, 6]])
b = np.array([7, 8, 9])
# Tính SVD của ma trận A
U, Sigma, Vt = np.linalg.svd(A)

# Tính nghịch đảo của Sigma
Sigma_inv = np.zeros_like(A.T)  # Tạo ma trận có kích thước và dữ liệu giống với A.T (không vuông)
Sigma_inv[:A.shape[1], :A.shape[1]] = np.diag(1 / Sigma)

# Tìm nghiệm x:  x = Vt.T * Sigma_inv * U.T * b
x = np.matmul(np.matmul(np.matmul(Vt.T , Sigma_inv) , U.T) , b)
print("Solution x:",x)
```

Exercise 8

Exercise 8: Compute SVD of each matrix below.

$$B_1 = \begin{bmatrix} -18 & 13 & -4 & 4 \\ 2 & 19 & -4 & 12 \\ -14 & 11 & -12 & 8 \\ -2 & 21 & 4 & 8 \end{bmatrix}, B_2 = \begin{bmatrix} 6 & -8 & -4 & 5 & -4 \\ 2 & 7 & -5 & -6 & 4 \\ 0 & -1 & -8 & 2 & 2 \\ -1 & -2 & 4 & 4 & -8 \end{bmatrix}$$

Tương tự Exercise 7

Exercise 9

Exercise 9: Write a program to compress image which an input image is given by user.

Tham khảo trang web sau:

https://www.geeksforgeeks.org/singular-value-decomposition-svd/?zarsrc=30&utm_source=zalo&utm_medium=zalo&utm_campaign=zalo

Exercise 10

Exercise 10: Write a program to implement Recommender System. It will be able to recommend an item x to an user y. Let $M_{n\times m}$ matrix present the relation of users and items

$$M_{n\times m} = \left(\begin{array}{c|ccc} & item1 & item2 & item3 \\ \text{user 1} & \infty & \infty & 3 \\ \text{user 2} & 2 & \infty & 5 \\ \text{user 3} & 5 & 6 & 7 \\ \text{user 4} & \infty & \infty & \infty \end{array}\right)$$

where ∞ means that user y has never rated item x before.

==> Cần đưa ra gợi ý một item chưa được đánh giá dựa vào ma trận biểu diễn mối quan hệ giữa người dùng và các item:

- Xây dựng hàm **recommend_item** xác định Item gợi ý ngẫu nhiên cho User (M, user):

  +Xác định dòng dữ liệu tương ứng cho user (*user_row*)

  +Xác định các cột (*item*) trên dòng *user_row* có giá trị chỉ định là User chưa từng đánh giá (ex: 0)

  +Nếu không có cột nào chưa đánh giá thì kết quả là "Không còn Item nào cần đánh giá"

  +Nếu có cột cần đánh giá thì kết quả là "Tên các Item cần đánh giá"

- Viết chương trình gọi hàm **recommend_item** ứng với từng user.

Hàm np.random.choice(a): chọn ngẫu nhiên 1 phần tử thuộc mảng a hoặc chọn số ngẫu nhiên trong khoảng np.arange(a)

Ví dụ:

```
a=np.array([1,3,5,7])
x=np.random.choice(a)
print(x)
b=10
y=np.random.choice(b)
print(y)
```

```
3
0
```

```
7
8
```