

TRƯỜNG ĐẠI HỌC BÁCH KHOA HÀ NỘI
VIỆN TOÁN ỨNG DỤNG VÀ TIN HỌC



BÁO CÁO BÀI TẬP LỚN
MÔN GIẢI TÍCH SỐ

ĐỀ TÀI
KHAI TRIỂN KÌ DỊ CỦA MA TRẬN (SVD)

Giảng viên hướng dẫn: Hà Thị Ngọc Yến

Nhóm sinh viên thực hiện: Nhóm 10 - lớp 125001

HÀ NỘI, 6/2021

Các thành viên trong nhóm

- Nguyễn Thái Thịnh 20195921 (nhóm trưởng):
- Nguyễn Hữu An 20195836
- Phạm Văn Đại 20195848
- Vũ Thành Tân 20195867
- Lê Thanh Thảo 20195919
- Phạm Đình Thông 20195923

LỜI MỞ ĐẦU

Dù là ở thời đại nào, toán học luôn đóng một vai trò quan trọng trong đời sống của chúng ta. Trong các lĩnh vực quan trọng là khoa học, kỹ thuật, y học và tài chính... , ta luôn thấy toán học hiện hữu trong đó và là công cụ thiết yếu để có thể nghiên cứu sâu hơn về các vấn đề học thuật. Toán học ứng dụng, một nhánh toán học liên quan đến việc ứng dụng kiến thức toán học vào những lĩnh vực khác, thúc đẩy và sử dụng những phát minh toán học mới, từ đó đã dẫn đến việc phát triển lên những ngành toán hoàn toàn mới, chẳng hạn như thống kê và toán học tính toán. Toán học tính toán đưa ra và nghiên cứu những phương pháp giải các bài toán toán học mà con người thường không có khả năng giải số được. Giải tích số (Numerical Analysis) và rộng hơn là tính toán khoa học (scientific computing) đều nghiên cứu những chủ đề phi giải tích như khoa học toán học, đặc biệt là ma trận thuật toán và lý thuyết đồ thị.

Giải tích số được biết đến là một môn khoa học nghiên cứu cách giải gần đúng, chủ yếu là giải số, các phương trình, các bài toán xấp xỉ hàm số, và các bài toán tối ưu. Những vấn đề trong giải tích số đều là cơ sở và là các vấn đề cấp thiết không thể thiếu đối với sinh viên ngành Toán ứng dụng. Nhận thấy tính cần thiết của môn học này, dưới sự hướng dẫn tận tình của giảng viên bộ môn TS. Hà Thị Ngọc Yến, chúng em tiến hành tìm hiểu và nghiên cứu về chủ đề “Khai triển kỳ dị của ma trận (SVD) và ứng dụng”. Nội dung chính của bài báo cáo bao gồm:

Chương I: Giới thiệu chủ đề mà mục tiêu.

Chương II: Cơ sở lý thuyết.

Chương III: Khai triển kỳ dị của ma trận gồm định nghĩa, chứng minh tính chất và hệ quả.

Chương IV: Thuật toán chi tiết, các gói và chương trình chạy và hệ thống ví dụ.

Chương V: Ứng dụng SVDs trong thực tế.

Chương VI: Kết luận.

Mong rằng bài báo cáo này sẽ giúp các bạn hiểu hơn về SVD cách sử dụng, tính toán và ứng dụng. Với khả năng có hạn về mặt kiến thức và thời gian nên bài báo cáo của chúng em không tránh khỏi những sai sót. Nhóm em rất mong nhận được góp ý nhận xét của cô và các bạn để bài báo cáo được hoàn thiện hơn.

Chúng em xin chân thành cảm ơn giảng viên bộ môn TS. Hà Thị Ngọc Yến đã truyền đạt những nền tảng kiến thức về chủ đề cũng như tận tình hướng dẫn và giúp đỡ chúng em trong suốt quá trình hoàn thành bài báo cáo. Nhờ đó mà chúng em đã hoàn thành được bài báo cáo của mình tốt nhất có thể.

Mục lục

1	Mở đầu	5
1.1	Giới thiệu	5
1.2	Mục tiêu	5
2	Cơ sở lý thuyết	6
2.1	Bốn không gian con cơ bản của ma trận	6
2.1.1	Không gian cột (Column Space)	6
2.1.2	Không gian hàng (Row Space)	6
2.1.3	Không gian hạt nhân (Kernel)	6
2.1.4	Không gian hạt nhân trái (CoKernel)	6
2.2	Một số khái niệm đặc biệt về ma trận	7
2.2.1	Ma trận đối xứng (Symmetric Matrix)	7
2.2.2	Ma trận trực giao (Orthogonal matrix)	7
2.2.3	Trực chuẩn (Orthonormal)	7
2.3	Trị riêng, vector riêng và các vấn đề liên quan	8
2.3.1	Trị riêng (Eigenvalues), Vector riêng (Eigenvectors)	8
2.3.2	Ma trận nửa xác định dương (Positive semidefinite Matrix)	8
2.3.3	Chéo hóa trực giao	9
2.4	Chuẩn vector, chuẩn ma trận	9
2.4.1	Chuẩn vector (Vector Norm)	9
2.4.2	Chuẩn ma trận (Matrix Norm)	10
3	Khai triển kỳ dị của ma trận	11
3.1	Giới thiệu	11
3.2	Khai triển kỳ dị của ma trận (SVD)	11
3.3	Sự tồn tại SVDs	13
3.4	Giá trị kỳ dị, vector kỳ dị (Eigenvalues and eigenvectors)	14
3.5	Tính chất và Hệ quả của SVDs	16
3.5.1	Liên hệ với hạng của ma trận	16
3.5.2	Tìm chuẩn của ma trận	16
3.5.3	Liên hệ với nghịch đảo suy rộng của ma trận	16
3.5.4	Liên hệ với số điều kiện	16
3.5.5	Polar decomposition	16
3.5.6	Truncated SVD	18
4	Thuật toán và chương trình	19
4.1	Thuật toán	19
4.2	Chương trình	22
4.3	Hệ thống ví dụ	24
5	Ứng dụng	32
5.1	Ứng dụng trong xử lý ảnh	32
5.2	Ứng dụng tính số điều kiện	34
6	Kết luận	36

1 Mở đầu

1.1 Giới thiệu

Ta đã học một số phương pháp phân tách ma trận thành các ma trận thừa số:

- Phương pháp phân rã trị riêng
- Phương pháp phân hủy LU
- ...

Chúng được gọi chung là Matrix Decomposition (Matrix Factorization).

Khai triển kỳ dị của ma trận (Singular Value Decomposition) trong một số tài liệu còn được gọi là phân tích suy biến. Đây là một dạng khai triển rất đặc biệt của Matrix Decomposition, khi mà mọi ma trận không nhất thiết phải vuông, đều có thể phân tách thành tích của ba ma trận đặc biệt. Singular Value Decomposition (SVDs) được ứng dụng rộng rãi ở nhiều lĩnh vực khoa học công nghệ như xử lý dữ liệu trong thông tin và viễn thông, máy học, học sâu, dữ liệu lớn

1.2 Mục tiêu

Mục tiêu của bài thuyết trình và báo cáo môn học Giải tích số:

- Nắm vững một số kiến thức cơ bản về đại số tuyến tính.
- Nắm được khai triển kỳ dị SVDs, thực hành khai triển với các ma trận từ đơn giản đến phức tạp.
- Ứng dụng khai triển kỳ dị vào một số lĩnh vực trong toán học và tin học.

2 Cơ sở lý thuyết

2.1 Bốn không gian con cơ bản của ma trận

2.1.1 Không gian cột (Column Space)

Trong đại số tuyến tính, không gian cột (còn được gọi là ảnh, hoặc miền giá trị) của ma trận A , kí hiệu là $\text{Im}(A)$ là tập hợp tất cả các tổ hợp tuyến tính (span) của các vectơ cột của A . Đôi khi không gian cột được kí hiệu là $C(A)$

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

$$C(A) = \text{span}\{(1, 14, 4, 15)^T, (8, 11, 5, 10)^T, (13, 2, 16, 3)^T, (12, 7, 9, 6)^T\}$$

2.1.2 Không gian hàng (Row Space)

Đối với không gian hàng, ta định nghĩa theo cách tương tự không gian cột:

Định nghĩa: Không gian hàng của ma trận A , kí hiệu $\text{Im}(A^T)$ là tập hợp tất cả các tổ hợp tuyến tính (span) của các vectơ hàng của A . Đôi khi không gian hàng được kí hiệu là $C(A^T)$

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

$$C(A^T) = \text{span}\{(1, 8, 13, 12), (14, 11, 2, 7), (4, 5, 16, 9), (15, 10, 3, 6)\}$$

2.1.3 Không gian hạt nhân (Kernel)

Xét biến đổi tuyến tính được biểu diễn bởi ma trận A cỡ $m \times n$ với các hệ số trên một trường K (thường là thực hoặc phức), hạt nhân của A là tập các vector x sao cho $Ax = 0$, với 0 được hiểu là vector không. Số chiều của hạt nhân của A được gọi là số vô hiệu của A ($\text{Nullity}(A)$)
Kí hiệu:

$$N(A) = \text{Null}(A) = \text{Ker}(A) = \{x \in K^n | Ax = 0\}$$

2.1.4 Không gian hạt nhân trái (CoKernel)

Không gian hạt nhân trái, kí hiệu $\text{Ker}(A^T)$ của ma trận A gồm các vectơ x sao cho $x^T A = 0$. Không gian hạt nhân trái của A chính là hạt nhân của A^T , và là phần bù trực giao của không gian cột của A

Không gian hàng, không gian cột, không gian hạt nhân và không gian hạt nhân trái của A là bốn không gian con cơ bản liên quan tới ma trận A

$$\text{Ví dụ: } A = \begin{bmatrix} 1 & 2 & 3 \\ 4 & 5 & 6 \end{bmatrix}$$

$$\text{Im}(A) = \text{span}\{(1,4),(2,5),(3,6)\} = R^2, \text{Im}(A^T) = \text{span}\{(1,2,3),(4,5,6)\}$$

$$\text{Ker}(A) = \{x | Ax = 0\} = \text{span}\{(1, -2, 1)^T\}, \text{Ker}(A^T) = \{x | x^T A = 0\} = \{0\}$$

Định lý về hạng và số chiều (Rank-nullity theorem)

Hạng (rank) của ma trận A là số chiều của không gian vector được sinh bởi các vector cột của A , và cũng chính là số chiều của không gian vector sinh bởi các vector hàng A :

$$\text{rank}(A) = \dim(\text{Im}(A)) = \dim(\text{Im}(A^T))$$

Phát biểu định lý: Cho T là toán tử tuyến tính từ không gian R^n sang không gian R^m . Ta có: $\text{rank}(T) + \text{Nullity}(T) = \dim(V)$. Nếu A là ma trận của toán tử T thì $\text{Ker}(A) \perp \text{Im}(A^T) = R^n$, hơn nữa $\text{Ker}(A)$ và không gian hàng $\text{Im}(A^T)$ trực giao. Đồng thời $\text{Ker}(A^T) \perp \text{Im}(A) = R^m$ và $\text{Ker}(A^T)$ trực giao với không gian cột $\text{Im}(A)$

Hệ quả: Nếu A là ma trận cỡ $m \times n$ thì hạng của ma trận A luôn nhỏ hơn hoặc bằng $\min\{m, n\}$

2.2 Một số khái niệm đặc biệt về ma trận

2.2.1 Ma trận đối xứng (Symmetric Matrix)

Trong đại số tuyến tính, A là ma trận đối xứng nếu ta chuyển vị A mà vẫn thu được chính nó:

$$A = A^T$$

Mỗi phần tử của một ma trận đối xứng thì đối xứng qua đường chéo. Mở rộng ra, trong trường số phức, ma trận vuông A là ma trận Hermitian matrix nếu ta chuyển vị liên hợp A thì thu được chính nó: $A = A^* = \overline{A^T}$

2.2.2 Ma trận trực giao (Orthogonal matrix)

Ma trận trực giao P là ma trận vuông thực với các cột và hàng của nó là các vector trực chuẩn, nếu ta chuyển vị ma trận trực giao P thì thu được nghịch đảo của nó:

$$P^T = P^{-1}$$

Ta cũng mở rộng ra trong trường số phức, ma trận đơn nhất (unita matrix) là một ma trận vuông U , khi ta chuyển vị liên hợp U thì thu được nghịch đảo của chính nó:

$$U^* = U^{-1}$$

Tính chất đặc biệt của ma trận trực giao là nó bảo toàn về mặt hình học. Với ma trận trực giao P ta có $\langle Pu, Pv \rangle = \langle u, v \rangle$ với mọi u, v hay ma trận trực giao bảo toàn khoảng cách và góc.

2.2.3 Trực chuẩn (Orthonormal)

Hai vector trong không gian có tích vô hướng được gọi là trực chuẩn nếu chúng trực giao nhau (hay vuông góc) và đều là vector đơn vị. Một tập các vector trực chuẩn là khi chúng trực giao đôi một với nhau và tất cả đều là vector đơn vị. Một cơ sở gồm toàn các vector trực chuẩn thì gọi là cơ sở trực chuẩn.

Từ một không gian S nào đó, có một cơ sở của S chưa trực chuẩn. Ta có thể thu được một cơ sở trực chuẩn của S thông qua quá trình trực chuẩn hóa Gram - Schmidt. Quá trình này được trình bày dưới đây, hình ảnh được lấy từ cuốn giáo trình Đại số tuyến tính của giảng viên Bùi Xuân Diệu - Đại học Bách Khoa Hà Nội, xuất bản năm 2009

Định lý 5.23. Cho V là một không gian vectơ có tích vô hướng, $S = \{u_1, u_2, \dots, u_n\}$ là một họ vectơ độc lập tuyến tính của V . Ta có thể thay S bởi họ trực chuẩn $S' = \{v_1, v_2, \dots, v_n\}$, sao cho $\text{span}\{u_1, u_2, \dots, u_k\} = \text{span}\{v_1, v_2, \dots, v_k\}$ với mọi $k = 1, 2, \dots, n$.

Bước 1: Đặt $v_1 = \frac{u_1}{\|u_1\|}$

Bước 2: Đặt $\bar{v}_2 = u_2 + tv_1$ sao cho $\langle \bar{v}_2, v_1 \rangle = 0$ tức là $t = -\langle u_2, v_1 \rangle$. Sau đó chọn $v_2 = \frac{\bar{v}_2}{\|\bar{v}_2\|}$. Giả sử sau $k-1$ bước ta đã xây dựng được họ trực chuẩn $S'_{k-1} = \{v_1, v_2, \dots, v_{k-1}\}$ sao cho $\text{span}\{u_1, u_2, \dots, u_{k-1}\} = \text{span}\{v_1, v_2, \dots, v_{k-1}\}$. Ta thực hiện đến bước thứ k sau:

Bước k : Đặt $\bar{v}_k = u_k + t_1v_1 + \dots + t_{k-1}v_{k-1}$ sao cho $\langle \bar{v}_k, v_j \rangle = 0, j = 1, 2, \dots, k-1$ Tức là ta có $t_j = -\langle u_k, v_j \rangle, j = 1, 2, \dots, k-1$. Sau đó chọn $v_k = \frac{\bar{v}_k}{\|\bar{v}_k\|}$. Tiếp tục thực hiện đến khi $k = n$ ta thu được hệ trực chuẩn $S' = \{v_1, v_2, \dots, v_n\}$

2.3 Trị riêng, vector riêng và các vấn đề liên quan

2.3.1 Trị riêng (Eigenvalues), Vector riêng (Eigenvectors)

Trong đại số tuyến tính, một vector riêng hay vector đặc trưng của một biến đổi tuyến tính là một vector khác vector không mà khi biến đổi tuyến tính đó được áp lên nó, sẽ thu được một hệ số vô hướng nhân với chính vector đó. Hệ số vô hướng tương ứng, thường được ký hiệu là λ , được gọi là giá trị riêng. Về ý nghĩa hình học, một vector riêng tương ứng với một giá trị riêng thực khác 0 có cùng phương sau khi nó được kéo dãn ra bởi phép biến đổi và giá trị riêng là hệ số nhân. Nếu giá trị riêng là âm thì vector sẽ đổi chiều. Một cách dễ hiểu, trong một không gian vector đa chiều, phương của vector riêng không bị quay đi sau khi bị tác động bởi phép biến đổi

Định nghĩa: Giả thiết A là một biến đổi tuyến tính từ một không gian vector V trên trường F tới chính nó và x là một vector khác 0 trong V . x là một vector riêng của A khi ảnh của nó qua phép biến đổi $A(x)$ bằng một số vô hướng (λ) nhân với x :

$$Ax = \lambda x$$

Có sự tương ứng trực tiếp giữa các ma trận vuông cấp n và biến đổi tuyến tính tự đồng cấu từ một không gian vector n chiều tới chính nó, trên cơ sở bất kỳ. Vì vậy trong không gian vector hữu hạn chiều, việc định nghĩa giá trị riêng và vector riêng sử dụng ngôn ngữ của ma trận và ngôn ngữ biến đổi tuyến tính là tương đương. Để tìm trị riêng của ma trận vuông A cấp n , ta giải phương trình:

$$(A - \lambda I)x = 0$$

Muốn λ là trị riêng, điều kiện cần và đủ là hệ có nghiệm không tầm thường:

$$\det(A - \lambda I) = 0$$

Phương trình $\det(A - \lambda I) = 0$ là phương trình đặc trưng của ma trận vuông A , còn đa thức $\det(A - \lambda I)$ là đa thức đặc trưng của A

2.3.2 Ma trận nửa xác định dương (Positive semidefinite Matrix)

Ma trận A là nửa xác định dương nếu dạng toàn phương của nó không âm. Khi đó A có các trị riêng không âm. Như vậy, $A^T A$ là nửa xác định dương vì nó có dạng toàn phương không âm:

$$Q(x) = x^T (A^T A) x = (x^T A^T) Ax = (Ax)^T Ax \geq 0$$

Tương tự AA^T cũng là ma trận nửa xác định dương. Ta dễ dàng chỉ ra các trị riêng dương của AA^T và $A^T A$ như nhau và có cùng số giá trị là $r = \text{rank}(A)$

Ngoài ra ta có:

$$\text{Ker}(A^T A) = \text{Ker}(A)$$

Thật vậy giả sử $x \in \text{Ker}(A)$ thì $Ax = 0 \Rightarrow A^T Ax = 0$ hay $x \in \text{Ker}(A^T A)$. Ngược lại nếu $x \in \text{Ker}(A^T A)$ thì $A^T Ax = 0 \Rightarrow x^T A^T Ax = 0 \Rightarrow \|Ax\|^2 = 0 \Rightarrow Ax = 0$ hay $x \in \text{Ker}(A)$. Chứng minh tương tự, ta được $\text{Ker}(AA^T) = \text{Ker}(A^T)$

2.3.3 Chéo hóa trực giao

Cho ma trận vuông A , nếu tồn tại ma trận trực giao P sao cho $P^{-1}AP$ là ma trận đường chéo thì ta nói ma trận A chéo hoá trực giao được và P là ma trận làm chéo hoá trực giao A :

$$P^{-1}AP = \text{diag}(\lambda_1, \lambda_2, \dots, \lambda_r)$$

Trong đó λ_i là các trị riêng của ma trận A , $i = \overline{1, r}$ và r là hạng của ma trận A

Định lý: Điều kiện cần và đủ để ma trận A được là A đối xứng

Hệ quả: Các ma trận AA^T hay $A^T A$ chéo hóa trực giao được

Quy trình chéo hóa trực giao là sự kết hợp của quy trình Gram-Schmidt và chéo hóa. Các bước của quy trình chéo hóa trực giao như sau:

Bước 1: Từ một ma trận vuông $A_{n \times n}$ chéo hóa được, tiến hành giải phương trình đặc trưng $\det(A - \lambda I) = 0$ để tìm ra các n giá trị riêng (có thể trùng nhau) của A

Bước 2: Từ mỗi giá trị riêng phân biệt, giải hệ phương trình $(A - \lambda I)x = 0$ để tìm các vector riêng ứng với không gian riêng của từng giá trị riêng. Nếu không gian riêng ứng với giá trị riêng chỉ có 1 vector cơ sở, giả sử là v_i , ta tiến hành chuẩn hóa nó bằng cách đặt $v'_i = \frac{v_i}{\|v_i\|}$. Nếu không gian riêng ứng với giá trị riêng có nhiều hơn 1 vector cơ sở, ta áp dụng quy trình Gram-Schmidt lên tập vector cơ sở để thu được một cơ sở trực chuẩn của không gian riêng đó

Bước 3: Ma trận làm chéo hóa A là ma trận P có các cột là các vector riêng trực chuẩn của A . Ma trận đường chéo $P^{-1}AP = \text{diag}(\lambda_1, \lambda_2, \dots, \lambda_n)$ với các giá trị riêng được sắp xếp phụ thuộc vào thứ tự của các vector cột trong P

2.4 Chuẩn vector, chuẩn ma trận

2.4.1 Chuẩn vector (Vector Norm)

Định nghĩa chuẩn vector: Cho vector x trong không gian vector R^n , chuẩn của x kí hiệu là $\|x\|$ là một số không âm thỏa mãn các tính chất sau:

- 1, Xác định dương : $\|x\| \geq 0$; $\|x\| = 0 \Leftrightarrow x = 0$
- 2, Tuyến tính : $\|ax\| = |a| \cdot \|x\| \quad \forall a \in R$
- 3, Bất đẳng thức tam giác : $\|x + y\| \leq \|x\| + \|y\|$

Cho vector x trong không gian vector R^n . Ta định nghĩa chuẩn p (với $p \in R, p > 0$) của vector x như sau:

$$\|x\|_p = \left(\sum_{i=1}^n |x_i|^p \right)^{\frac{1}{p}}$$

Với $p = 1, 2$ và $+\infty$, ta lần lượt có các chuẩn tương ứng là chuẩn Manhattan, chuẩn Euclide và chuẩn cực đại.

2.4.2 Chuẩn ma trận (Matrix Norm)

Định nghĩa chuẩn ma trận: Chuẩn của ma trận A , ký hiệu $\|A\|$ là một số không âm và cũng thoả mãn các tính chất:

- 1, Xác định dương: $\|A\| \geq 0$; $\|A\| = 0 \Leftrightarrow A = 0$
- 2, Tuyến tính: $\|\alpha A\| = |\alpha| \cdot \|A\| \forall \alpha \in R$
- 3, Bất đẳng thức tam giác: $\|A + B\| \leq \|A\| + \|B\|$

Cho ma trận $A_{m \times n}$ trong không gian $R^{m \times n}$. Ta định nghĩa chuẩn p (với $p \in R, p > 0$) của ma trận A như sau:

$$\|A\|_p = \sup_{x \neq 0, x \in R^n} \frac{\|Ax\|_p}{\|x\|_p}$$

Một số chuẩn của ma trận:

- Chuẩn 1 (chuẩn cực đại cho cột):

$$\|A\|_1 = \max_{1 \leq j \leq n} \sum_{i=1}^m |a_{ij}|$$

- Chuẩn $+\infty$ (Chuẩn cực đại cho hàng):

$$\|A\|_\infty = \max_{1 \leq i \leq m} \sum_{j=1}^n |a_{ij}|$$

Ví dụ: $A = \begin{pmatrix} 4 & 2 & 3 \\ -1 & 3 & -2 \\ 0 & 1 & 5 \\ 2 & 4 & -3 \end{pmatrix}$

$$\|A\|_1 = \max_{1 \leq j \leq 3} \sum_{i=1}^4 |a_{ij}| = \max(7, 10, 13) = 13$$

$$\|A\|_\infty = \max_{1 \leq i \leq 4} \sum_{j=1}^3 |a_{ij}| = \max(9, 6, 6, 9) = 9$$

Ngoài ra, chuẩn Frobenius là chuẩn rất quan trọng mà sẽ được sử dụng trong bài báo cáo này
Định nghĩa: Chuẩn Frobenius của một ma trận $X \in R^{m \times n}$ là căn bậc hai của tổng các bình phương của các thành phần trong ma trận:

$$\|X\|_F = \sqrt{\sum_{i=1}^m \sum_{j=1}^n x_{ij}^2}$$

Chuẩn Frobenius của ma trận A cũng chính là giá trị kỳ dị lớn nhất σ_1 của A , là khái niệm ta sẽ nghiên cứu ở các chương sau

3 Khai triển kì dị của ma trận

3.1 Giới thiệu

Trong đại số tuyến tính, khai triển kỳ dị của ma trận (SVD) là một dạng của matrix factorization nhằm phân tích một ma trận thành tích của các ma trận số thực hoặc ma trận số phức. Đây là tổng quát hoá của phép phân tích riêng trong trường hợp ma trận không vuông khi nó biểu diễn ma trận gốc thành tích của 2 ma trận trực giao và một ma trận đường chéo.

SVD ban đầu được phát triển bởi các nhà hình học vi phân. Chứng minh đầu tiên về SVD đối với ma trận hình chữ nhật và ma trận phức vào năm 1936, họ đã xem nó như một sự tổng quát của phép biến đổi trực chính cho ma trận Hermitian

Năm 1907, Erhard Schmidt đã định nghĩa một tương tự của các singular value cho các toán tử tích phân. Và năm 1910 Émile Picard là người đầu tiên gọi các số σ_k là singular value (hoặc trong tiếng Pháp, valeurs singulières)

Các phương pháp thực tế để tính toán SVD có từ thập kỷ 1950, gần giống với thuật toán trị riêng Jacobi, sử dụng phép quay mặt phẳng hoặc phép quay Givens. Tuy nhiên, những phương pháp này đã được thay thế bằng phương pháp của Gene Golub xuất bản năm 1965, sử dụng các phép biến đổi hoặc phản xạ Householder.

Vào năm 1970, Golub đã công bố một biến thể của thuật toán Golub/Kahan mà vẫn được sử dụng nhiều nhất cho đến ngày nay

3.2 Khai triển kì dị của ma trận (SVD)

Định nghĩa: Khai triển kỳ dị của ma trận phức $A_{m \times n}$ là một dạng khai triển:

$$A = U \Sigma V^*$$

Ở đó U là ma trận vuông unita cỡ $m \times m$, Σ là ma trận đường chéo hình chữ nhật với các số thực không âm trên đường chéo, V là ma trận unita cỡ $n \times n$. Nếu ma trận A là ma trận thực, thì U và V là ma trận trực giao. Khi đó, khai triển SVDs sẽ là:

$$A = U \Sigma V^T \quad (1)$$

Các giá trị trên đường chéo của Σ : $\sigma_i = \Sigma_{ii}$ với $\sigma_1 \geq \sigma_2 \geq \dots \geq \sigma_r > 0$ được gọi là giá trị kỳ dị của A (Singular values). r là số giá trị khác 0 của σ và cũng chính bằng hạng của A . r vector cột đầu tiên của U và r vector cột đầu tiên V lần lượt được gọi là các vector kỳ dị trái (left-singular vectors) và các vector kỳ dị phải (right-singular vectors) của A

SVDs không phải là duy nhất, vì ta chỉ cần đổi dấu cả U và V thì (1) vẫn thỏa mãn. Tuy vậy người ta vẫn thường dùng “the SVD” thay vì “a SVD”, dễ dàng tính được có 2^{m+n-r} cách biểu diễn SVDs khác nhau. Hình ảnh ở trang tiếp theo là cách biểu diễn SVD của ma trận $A_{m \times n}$ trong hai trường hợp khác nhau:

$$\mathbf{A}_{m \times n} = \mathbf{U}_{m \times m} \times \Sigma_{m \times n} \times \mathbf{V}_{n \times n}^T$$

$(m < n)$

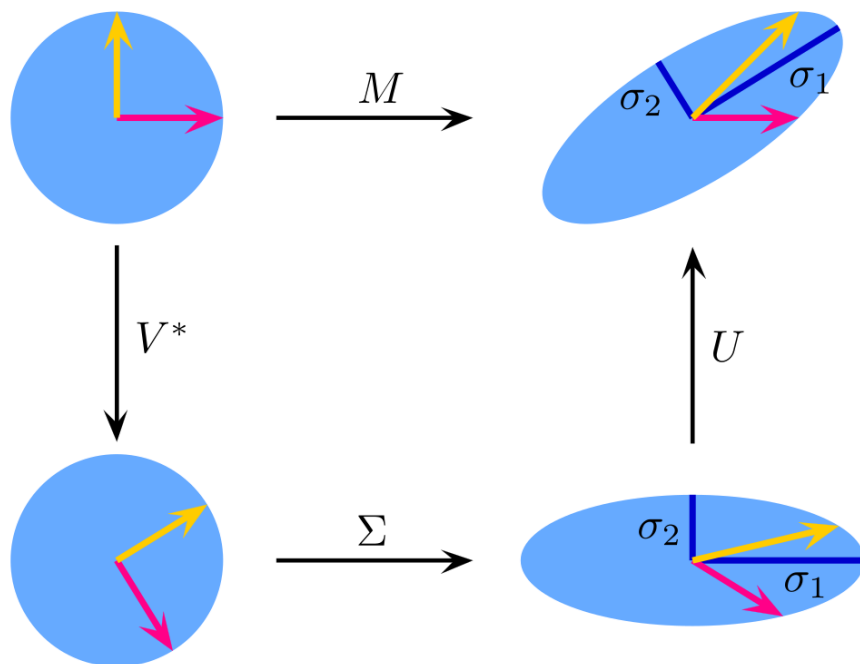
$$\mathbf{A}_{m \times n} = \mathbf{U}_{m \times m} \times \Sigma_{m \times n} \times \mathbf{V}_{n \times n}^T$$

$(m > n)$

Về mặt hình học, khai triển kỳ dị của ma trận sẽ lần lượt trải qua: phép xoay (rotation), phép nối rộng (scaling) và phép xoay ở cuối cùng. Nếu coi các dòng của ma trận A là các điểm dữ liệu và các chiều dữ liệu là các cột. Khi đó:

- Nhân các điểm dữ liệu với ma trận trực giao V^T chính là việc ta thực hiện một phép xoay và phép xoay này không làm thay đổi tích vô hướng của các vector.
- Phép nhân với ma trận đường chéo Σ sẽ làm co giãn độ lớn các chiều theo giá trị của các trị riêng trên đường chéo chính.
- Và cuối cùng, phép nhân với ma trận trực giao U lại thực hiện phép xoay lần nữa.

Các quá trình trên có thể được hình dung một cách trực quan như sau:



$$M = U \cdot \Sigma \cdot V^*$$

3.3 Sự tồn tại SVDs

Có nhiều cách chứng minh sự tồn tại của khai triển kỳ dị. Sau đây là cách chứng minh dựa vào định lý phổ của ma trận (Spectral theorem):

Chứng minh: Cho M là 1 ma trận phức $m \times n$. Vì M^*M là nửa xác định dương và đối xứng, theo định lý phổ, ở đây tồn tại 1 ma trận unita V cấp $n \times n$ sao cho:

$$V^*M^*MV = \begin{bmatrix} D & 0 \\ 0 & 0 \end{bmatrix} \quad (2)$$

Trong đó:

- D là ma trận đường chéo xác định dương, có kích thước $r \times r$, với r là số trị riêng khác 0 của M^*M ($r \leq \min(n, m)$)
- V ở đây là 1 ma trận mà cột thứ i với $1 < i \leq r$ là vector riêng của M^*M , ứng với trị riêng $D_i > 0$. Hơn nữa, cột thứ j của V , với $j > r$, là vector riêng của M^*M ứng với trị riêng $D_j = 0$

Có thể viết V dưới dạng $V = [V_1 \ V_2]$, trong đó các cột của V_1 và V_2 lần lượt chứa các vector riêng của M^*M tương ứng với các trị riêng khác 0 và bằng 0. Sử dụng cách viết này của V , phương trình (2) trở thành:

$$\begin{bmatrix} V_1^* \\ V_2^* \end{bmatrix} M^*M \begin{bmatrix} V_1 & V_2 \end{bmatrix} = \begin{bmatrix} V_1^*M^*MV_1 & V_1^*M^*MV_2 \\ V_2^*M^*MV_1 & V_2^*M^*MV_2 \end{bmatrix} = \begin{bmatrix} D & 0 \\ 0 & 0 \end{bmatrix}$$

Điều này chỉ ra : $V_1^*M^*MV_1 = D, V_2^*M^*MV_2 = 0$. Hơn nữa, phương trình thứ hai chỉ ra $MV_2 = 0$. Cuối cùng, do V là ma trận unita, với V_1 và V_2 , ta sẽ có:

- $V_1^*V_1 = I_1$
- $V_2^*V_2 = I_2$
- $V_1V_1^* + V_2V_2^* = I_{12}$

Trong đó các chỉ số dưới của các ma trận đơn vị được sử dụng để nói rằng chúng có các cấp khác nhau

Ta định nghĩa:

$$U_1 = MV_1D^{-\frac{1}{2}} \Rightarrow U_1D^{\frac{1}{2}}V_1^* = MV_1D^{-\frac{1}{2}}D^{\frac{1}{2}}V_1^* = M$$

Chú ý rằng $\{v_i\}_{i=1}^r$ là tập hợp các vector riêng của M^*M tương ứng với trị riêng dương. Hơn nữa:

$$\langle Mv_i, Mv_j \rangle = \langle v_i, M^T Mv_j \rangle = \lambda_j \langle v_i, v_j \rangle = 0$$

Nghĩa là $\{Mv_i\}_{i=1}^r$ là 1 tập các vector trực giao. Đồng thời:

$$\langle Mv_i, Mv_i \rangle = \langle v_i, M^T Mv_i \rangle = \lambda \langle v_i, v_i \rangle = \lambda \Rightarrow \|Mv_i\| = \sqrt{\lambda_i}$$

Hay $\{\lambda^{-\frac{1}{2}} Mv_i\}_{i=1}^r = u_i$ là tập các vector trực chuẩn. Đây gần như là kết quả mong muốn, ngoại trừ việc U_1 và V_1 nói chung là không đơn nhất (unita), vì chúng có thể không vuông. Số hàng của U_1 không nhỏ hơn số cột, vì kích thước của D không lớn hơn m hoặc n

Ngoài ra, vì:

$$U_1^* U_1 = D^{-\frac{1}{2}} V_1^* M^* M V_1 D^{-\frac{1}{2}} = D^{-\frac{1}{2}} D D^{-\frac{1}{2}} = I_1$$

Các cột của U_1 là trực chuẩn và có thể được mở rộng để thành cơ sở trực chuẩn. Ta có thể chọn U_2 sao cho $U = \begin{bmatrix} U_1 & U_2 \end{bmatrix}$ là đơn nhất

Với V_1 , ta đã có V_2 để làm nó đơn nhất. Bây giờ ta định nghĩa:

$$\Sigma = \begin{bmatrix} \begin{bmatrix} D^{\frac{1}{2}} & 0 \\ 0 & 0 \end{bmatrix} \\ 0 \end{bmatrix}$$

Trong đó các hàng 0 được thêm vào hoặc bỏ đi để số hàng 0 bằng với số cột của U_2 , và do đó kích thước của Σ là $m \times n$. Từ đó:

$$\begin{bmatrix} U_1 & U_2 \end{bmatrix} \begin{bmatrix} \begin{bmatrix} D^{\frac{1}{2}} & 0 \\ 0 & 0 \end{bmatrix} \\ 0 \end{bmatrix} \begin{bmatrix} V_1 & V_2 \end{bmatrix}^* = \begin{bmatrix} U_1 & U_2 \end{bmatrix} \begin{bmatrix} D^{\frac{1}{2}} V_1^* \\ 0 \end{bmatrix} = U_1 D^{\frac{1}{2}} V_1^* = M$$

Đó là kết quả mong muốn: $M = U \Sigma V^*$

(DPCM)

Chú ý rằng việc chứng minh có thể bắt đầu bằng chéo hóa MM^* thay vì M^*M (điều đó là do MM^* và M^*M có cùng các trị riêng khác 0). Ngoài ra, có nhiều cách chứng minh khác nữa như dựa vào chuẩn hay, tuy nhiên trong khuôn khổ bài báo cáo này sẽ không đề cập đến.

3.4 Giá trị kỳ dị, vector kỳ dị (Eigenvalues and eigenvectors)

Dễ dàng nhận thấy khai triển kỳ dị khác với phép nhân tích riêng ở chỗ nó áp dụng cho ma trận bất kỳ mà không yêu cầu ma trận đó phải vuông. Để thấy được mối liên hệ mật thiết giữa khai triển kỳ dị và phép phân tích riêng ta sẽ quay lại khai triển $A^T A$:

$$A^T A = (U \Sigma V^T)^T \cdot U \Sigma V^T = V \Sigma^T \cdot \Sigma V^T$$

$A^T A$ có một phân tích riêng là ma trận trực giao V và ma trận đường chéo $\Sigma^T \Sigma$. Tương tự AA^T có một phân tích riêng là ma trận trực giao U và ma trận đường chéo $\Sigma \Sigma^T$

Ta giả sử $A_{m \times n}$ có hạng bằng r , $r \leq \min\{m, n\}$. $A^T A$ là ma trận đối xứng nửa xác định dương nên:

$$\Sigma^T \Sigma = \text{diag}(\lambda_1, \dots, \lambda_n)$$

Với $\lambda_1 \geq \lambda_2 \geq \dots \geq \lambda_r > 0$ và $\lambda_{r+1} = \dots = \lambda_n = 0$. Khi đó ma trận chữ nhật $\Sigma = \text{diag}(\sigma_1, \dots, \sigma_r)$ với $\sigma_i = \sqrt{\lambda_i}$ là các giá trị kỳ dị của A , hay σ_i^2 là các giá trị riêng của $A^T A$, còn các vector kỳ dị phải v_i là các vector riêng của $A^T A$ ứng với $\lambda_i (i = \overline{1, r})$

Một cách tương tự ta cũng có thể xác định được các vector u_i từ AA^T . Tuy nhiên chúng ta thường không làm như vậy. Có một cách đơn giản hơn để xác định u_i :

$$\begin{aligned} \text{Do } AV &= U \Sigma \\ \Rightarrow Av_1 &= \sigma_1 u_1 \\ &\dots \\ Av_r &= \sigma_r u_r \\ Av_{r+1} &= 0 \end{aligned}$$

Ta rút ra được $u_1 = \frac{Av_1}{\sigma_1}; u_2 = \frac{Av_2}{\sigma_2}; \dots; u_r = \frac{Av_r}{\sigma_r}$

Ta có thể bổ sung các vector u_{r+1} đến u_m bằng các vector riêng trực chuẩn ứng với trị riêng $\lambda = 0$. Hay chính là các vector cơ sở trực chuẩn của $\text{Ker}(AA^T)$ tức $\text{Ker}(A^T)$

Ví dụ 1

$$A = \begin{bmatrix} 2 & 2 \\ 1 & 1 \end{bmatrix}$$

$$\text{Ta có: } A^T A = \begin{bmatrix} 2 & 1 \\ 2 & 1 \end{bmatrix} \begin{bmatrix} 2 & 2 \\ 1 & 1 \end{bmatrix} = \begin{bmatrix} 5 & 5 \\ 5 & 5 \end{bmatrix}$$

$$\det(A^T A - \lambda I) = \begin{vmatrix} 5 - \lambda & 5 \\ 5 & 5 - \lambda \end{vmatrix} = 0 \Rightarrow \begin{cases} \lambda = \lambda_1 = 10 \\ \lambda = \lambda_2 = 0 \end{cases}$$

\Rightarrow Giá trị kỳ dị $\sigma_1 = \sqrt{10}$

Để tìm v_1, v_2 ta xét phương trình đặc trưng:

$$(A^T A - \lambda_1 I)x = 0 \Leftrightarrow \begin{bmatrix} -5 & 5 \\ 5 & -5 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \end{bmatrix} \Leftrightarrow [x_1 \ x_2]^T = \alpha [1 \ 1]^T$$

$$(A^T A - \lambda_2 I)x = 0 \Leftrightarrow \begin{bmatrix} 5 & 5 \\ 5 & 5 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \end{bmatrix} \Leftrightarrow [x_1 \ x_2]^T = \beta [1 \ -1]^T$$

Trực chuẩn hoá Gram-Schmidt $[1 \ 1]^T$ và $[1 \ -1]^T$ ta được

$$v_1 = \left[\frac{1}{\sqrt{2}} \ \frac{1}{\sqrt{2}} \right]^T \text{ và } v_2 = \left[\frac{1}{\sqrt{2}} \ \frac{-1}{\sqrt{2}} \right]^T \Rightarrow V = \begin{bmatrix} \frac{1}{\sqrt{2}} & \frac{1}{\sqrt{2}} \\ \frac{1}{\sqrt{2}} & \frac{-1}{\sqrt{2}} \end{bmatrix}$$

Để tính u_1 ta lấy:

$$u_1 = \frac{Av_1}{\sigma_1} = \begin{bmatrix} \frac{2}{\sqrt{5}} \\ \frac{1}{\sqrt{5}} \end{bmatrix}$$

Để bổ sung u_2 ta có thể tìm vector riêng có độ dài đơn vị ứng với trị riêng $= 0$ của AA^T tức là tìm cơ sở trực chuẩn của $\text{Ker}(A^T)$. Và như vậy ta được:

$$_2 = \begin{bmatrix} \frac{1}{\sqrt{5}} \\ \frac{-2}{\sqrt{5}} \end{bmatrix} \Rightarrow U = \begin{bmatrix} \frac{2}{\sqrt{5}} & \frac{1}{\sqrt{5}} \\ \frac{1}{\sqrt{5}} & \frac{-2}{\sqrt{5}} \end{bmatrix}$$

Vậy khai triển SVDs của A là:

$$\begin{bmatrix} 2 & 2 \\ 1 & 1 \end{bmatrix} = \begin{bmatrix} \frac{2}{\sqrt{5}} & \frac{1}{\sqrt{5}} \\ \frac{1}{\sqrt{5}} & \frac{-2}{\sqrt{5}} \end{bmatrix} \begin{bmatrix} \sqrt{10} & 0 \\ 0 & 0 \end{bmatrix} \begin{bmatrix} \frac{1}{\sqrt{2}} & \frac{1}{\sqrt{2}} \\ \frac{1}{\sqrt{2}} & \frac{-1}{\sqrt{2}} \end{bmatrix}$$

Ví dụ 2

$$A = \begin{bmatrix} 1 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix}. \text{ Kết quả:}$$

$$\begin{bmatrix} 1 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix} = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix} \begin{bmatrix} \sqrt{2} & 0 & 0 \\ 0 & 1 & 0 \end{bmatrix} \begin{bmatrix} \frac{1}{\sqrt{2}} & \frac{1}{\sqrt{2}} & 0 \\ 0 & 0 & 1 \\ \frac{-1}{\sqrt{2}} & \frac{1}{\sqrt{2}} & 0 \end{bmatrix}$$

Ví dụ 3

$$A = \begin{bmatrix} 1 & 1 \\ 1 & 0 \\ 0 & 1 \end{bmatrix}. \text{ Kết quả:}$$

$$\begin{bmatrix} 1 & 1 \\ 1 & 0 \\ 0 & 1 \end{bmatrix} = \begin{bmatrix} \frac{2}{\sqrt{6}} & 0 & \frac{-1}{\sqrt{3}} \\ \frac{1}{\sqrt{6}} & \frac{-1}{\sqrt{2}} & \frac{1}{\sqrt{3}} \\ \frac{1}{\sqrt{6}} & \frac{1}{\sqrt{2}} & \frac{1}{\sqrt{3}} \end{bmatrix} \begin{bmatrix} \sqrt{3} & 0 \\ 0 & 1 \\ 0 & 0 \end{bmatrix} \begin{bmatrix} \frac{1}{\sqrt{2}} & \frac{1}{\sqrt{2}} \\ \frac{-1}{\sqrt{2}} & \frac{1}{\sqrt{2}} \end{bmatrix}$$

3.5 Tính chất và Hệ quả của SVDs

3.5.1 Liên hệ với hạng của ma trận

Số giá trị kỳ dị dương của ma trận A là hạng của ma trận A

$$\text{Ví dụ: } A = \begin{bmatrix} 1 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix}: \sigma_1 = \sqrt{2}, \sigma_2 = 1 \Rightarrow \text{rank}(A) = 2$$

3.5.2 Tìm chuẩn của ma trận

Giá trị kỳ dị lớn nhất của ma trận A chính là chuẩn Frobenius của A :

$$\sigma_1 = \|A\|_F$$

3.5.3 Liên hệ với nghịch đảo suy rộng của ma trận

Khai triển kỳ dị có thể được sử dụng để tính toán nghịch đảo suy rộng (giả nghịch đảo). Thật vậy với $M = U\Sigma V^T$ thì $M^+ = V\Sigma^+U^T$ trong đó Σ^+ được hình thành bằng cách thay thế các giá trị trên đường chéo khác không của Σ bằng nghịch đảo của chúng. Nghịch đảo suy rộng được vận dụng để giải các bài toán bình phương tối thiểu tuyến tính.

3.5.4 Liên hệ với số điều kiện

Số điều kiện của ma trận A khả nghịch là:

$$\text{cond}(A) = \|A\| \cdot \|A^{-1}\|$$

Người ta đã chứng minh được $\text{cond}(A) = \frac{\sigma_1(A)}{\sigma_r(A)}$ ở đó $\sigma_1(A)$ và $\sigma_r(A)$ lần lượt là giá trị kỳ dị lớn nhất và nhỏ nhất của A

Trong thực tế $\text{cond}(A)$ càng nhỏ thì ma trận A là điều kiện tốt. Trong trường hợp A là ma trận điều kiện xấu, ta có thể cắt bỏ các giá trị σ nhỏ và giữ lại giá trị σ lớn, khi đó ma trận được xác định trên không gian chiếu tương ứng với các σ lớn sẽ tốt hơn.

3.5.5 Polar decomposition

Khai triển polar của ma trận vuông thực hoặc phức A là dạng khai triển:

$$A = QS$$

Ở đó Q là ma trận unita, S là ma trận Hermitian nửa xác dương, cả 2 ma trận đều cùng kích thước. Thật vậy, từ khai triển kỳ dị của ma trận: $A = U\Sigma V^* = UV^* \cdot V\Sigma V^*$ đặt $Q = UV^*$; $S = V\Sigma V^*$ dễ dàng thấy $A = QS$ là triển khai polar cần tìm.

Ví dụ Polar decomposition: Ta đã có: $A = \begin{bmatrix} 2 & 2 \\ 1 & 1 \end{bmatrix} = \begin{bmatrix} \frac{2}{\sqrt{5}} & \frac{1}{\sqrt{5}} \\ \frac{1}{\sqrt{5}} & \frac{-2}{\sqrt{5}} \end{bmatrix} \begin{bmatrix} \sqrt{10} & 0 \\ 0 & 0 \end{bmatrix} \begin{bmatrix} \frac{1}{\sqrt{2}} & \frac{1}{\sqrt{2}} \\ \frac{1}{\sqrt{2}} & \frac{-1}{\sqrt{2}} \end{bmatrix}$

$$\text{Đặt } Q = UV^* = \begin{bmatrix} \frac{3}{\sqrt{10}} & \frac{1}{\sqrt{10}} \\ \frac{-1}{\sqrt{10}} & \frac{3}{\sqrt{10}} \end{bmatrix}$$

$$\text{Và } S = V\Sigma V^* = \begin{bmatrix} \frac{5}{\sqrt{10}} & \frac{5}{\sqrt{10}} \\ \frac{5}{\sqrt{10}} & \frac{5}{\sqrt{10}} \end{bmatrix}$$

$$\Rightarrow A = QS = \begin{bmatrix} \frac{3}{\sqrt{10}} & \frac{1}{\sqrt{10}} \\ \frac{-1}{\sqrt{10}} & \frac{3}{\sqrt{10}} \end{bmatrix} \begin{bmatrix} \frac{5}{\sqrt{10}} & \frac{5}{\sqrt{10}} \\ \frac{5}{\sqrt{10}} & \frac{5}{\sqrt{10}} \end{bmatrix} \text{ là khai triển polar của } A$$

3.5.6 Truncated SVD

Một biểu diễn tương đương khác của A dưới dạng tổng của các ma trận rank 1:

$$A = \sigma_1 u_1 v_1^T + \sigma_2 u_2 v_2^T + \dots + \sigma_r u_r v_r^T$$

Với $r = \text{rank}(A)$. Rõ ràng trong cách biểu diễn này, ma trận A chỉ phụ thuộc vào r cột đầu tiên của U , V và r giá trị khác 0 trên đường chéo chính của ma trận Σ . Nếu A có rank nhỏ hơn rất nhiều so với số hàng và số cột $r \ll m, n$ ta sẽ được lợi về mặt lưu trữ

Ví dụ minh họa với $m = 4, n = 6, r = 2$:

$$A = U_r \Sigma_r (V_r)^T = \sigma_1 u_1 v_1^T + \sigma_2 u_2 v_2^T$$

Rõ ràng nếu ta lưu cả ma trận A , ta phải dùng 24 đơn vị lưu trữ (A có kích thước 4×6). Nhưng nếu ta chỉ lưu khai triển kì dị của A , chỉ cần dùng 22 đơn vị lưu trữ. Nếu $r \ll m, n$ thì lợi thế về lưu trữ càng lớn

Chú ý rằng trong ma trận Σ , các giá trị trên đường chéo là không âm và giảm dần $\sigma_1 \geq \sigma_2 \geq \dots \geq \sigma_r \geq 0 = 0 = \dots = 0$. Thông thường, chỉ một lượng nhỏ các giá trị σ_i mang giá trị lớn, các giá trị còn lại thường nhỏ và gần 0. Khi đó ta có thể xấp xỉ ma trận A bằng tổng của $k < r$ ma trận có rank 1

$$A \approx A_k = U_k \Sigma_k (V_k)^T = \sigma_1 u_1 v_1^T + \dots + \sigma_k u_k v_k^T$$

Dưới đây là một định lý thú vị. Định lý này nói rằng sai số do cách xấp xỉ trên chính là căn bậc hai của tổng bình phương của các singular values mà ta bỏ qua ở phần cuối của Σ

$$\|A - A_k\|_F^2 = \sum_{i=k+1}^r \sigma_i^2$$

Ở đây $\|A\|_F = \sqrt{\sum_{i=1}^m \sum_{j=1}^n a_{ij}^2}$ là chuẩn Frobenius. Như vậy sai số do xấp xỉ càng nhỏ nếu phần

singular values bị truncated có giá trị càng nhỏ so với phần singular values được giữ lại. Đây là một định lý quan trọng giúp xác định việc xấp xỉ ma trận dựa trên lượng thông tin muốn giữ lại. Ví dụ, nếu ta muốn giữ lại ít nhất 90% lượng thông tin trong A :

- Trước hết ta tính $\sum_{j=1}^r \sigma_j^2$
- Sau đó chọn k nhỏ nhất sao cho: $\frac{\sum_{i=1}^k \sigma_i^2}{\sum_{j=1}^r \sigma_j^2} \geq 0.9$
- Khi k nhỏ, A_k có rank là k , ma trận có rank nhỏ. Vì vậy Truncated SVD còn được gọi là Low-rank approximation

4 Thuật toán và chương trình

4.1 Thuật toán

- Input: Ma trận $A_{m \times n}$
- Output: U, Σ, V^T
- Các bước thực hiện:

Bước 1: Nhập ma trận $A_{m \times n}$.

Bước 2: Tính hạng của ma trận A là r bằng hàm `matrix_rank(A)`.

Bước 3: So sánh m, và n, nếu $m \geq n$ ta đi tới bước 4, ngược lại tới bước 10.

Bước 4: Dùng hàm `eig(AAT)` trả về U là ma trận vuông trực giao cỡ $m \times m$ là ma trận m vector riêng trực chuẩn của AA^T và mảng w_i là các trị riêng λ_i không âm tương ứng. Mảng w_i sẽ gồm r trị riêng dương và m-r trị riêng bằng 0 (tính cả trị riêng kép). Ma trận U tại bước này chính là ma trận U cần tìm.

Bước 5: Sử dụng Bubble Sort sắp xếp mảng w_i các trị riêng theo thứ tự giảm dần đồng thời cũng sắp xếp các vector cột u_i của U tương ứng với trị riêng w_i .

Bước 6: Gán ma trận Σ là ma trận 0 cỡ $m \times n$ bằng hàm `np.zeros((m,n))`. Ta thay đổi các giá trị trên đường chéo thành $\sigma_i = \sqrt{w(i)}$ bằng hàm `Fill_Diagonal` (Σ, \sqrt{w}). Đây chính là ma trận Σ cần tính.

Bước 7: Gán $U = U^T$. Và tính ma trận $V_{r \times n}$, với r vector hàng của V được tính bằng công thức $v_i = \frac{u_i A}{\sigma_i}$ (σ_i là giá trị kỳ dị tương ứng đã được sắp xếp và được tính ở bước 5 và bước 6).

Bước 8: So sánh r và n, nếu $r < n$ là sai, ta sang bước 9. Còn nếu $r < n$ ta sử dụng hàm `null_space(A)` để tính ma trận hạt nhân $Ker(A) = B_{n \times (n-r)}$, rồi gán ma trận V bởi $V_{n \times r}$ gộp với ma trận $B_{(n-r) \times r}^T$ bằng hàm `concatenate ((V, BT), axis=0)`

Bước 9: Gán lại $U = U^T$ rồi xuất U, xuất Σ , xuất V.

Bước 10: Trong trường hợp $m < n$, ta thực hiện các bước tiếp theo một cách tương tự, đầu tiên ta dùng `eig(ATA)` trả về V là ma trận các vector riêng trực chuẩn của $A^T A$ và mảng w_i là mảng các trị riêng λ_i không âm tương ứng.

Bước 11: Sử dụng Bubble Sort sắp xếp mảng w_i các trị riêng theo thứ tự giảm dần đồng thời cũng sắp xếp các vector cột v_i của V tương ứng với trị riêng w_i .

Bước 12: Tương tự như bước 6, ta cũng gán ma trận Σ là ma trận 0 cỡ $m \times n$ bằng hàm `np.zeros((m,n))`. Ta thay đổi các giá trị trên đường chéo thành $\sigma_i = \sqrt{w(i)}$ bằng hàm `Fill_Diagonal` (Σ, \sqrt{w}). Đây chính là ma trận Σ cần tính.

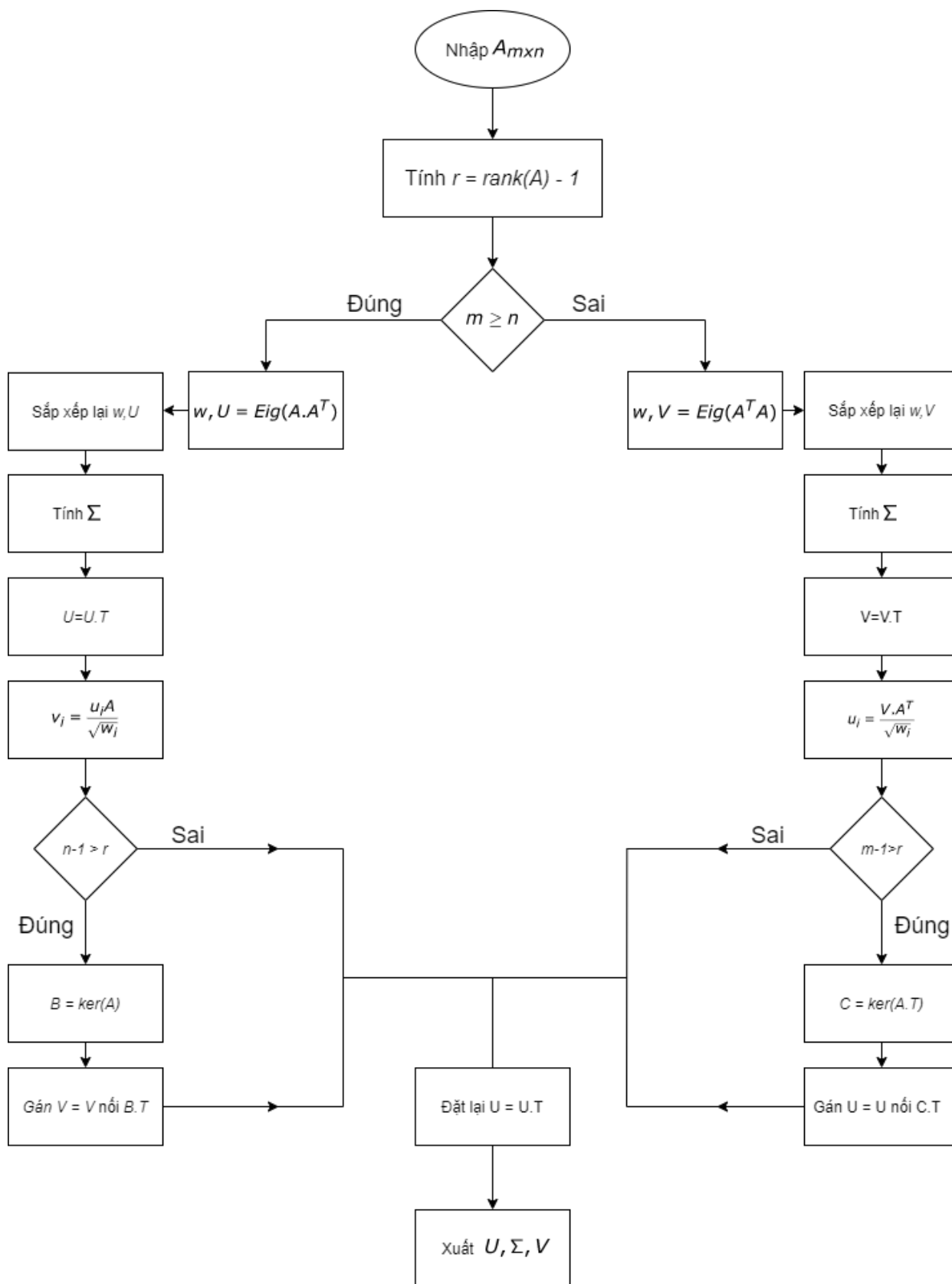
Bước 13: Gán $V = V^T$. Và tính ma trận $U_{r \times m}$, với r vector hàng của U được tính bằng công thức $u_i = \frac{v_i A^T}{\sigma_i}$ (σ_i là giá trị kỳ dị tương ứng đã được sắp xếp và được tính ở bước 11 và bước 12).

Bước 14: So sánh r và m, nếu $r < m$ là sai, ta về bước 10. Nếu $r < m$ là đúng ta sử dụng hàm `null_space(AT)` để tính ma trận hạt nhân trái $Ker(A^T) = C_{m \times (m-r)}$, rồi gán ma trận U bởi $U_{r \times m}$ gộp với ma trận $C_{(m-r) \times m}^T$ bằng hàm `concatenate ((U, CT), axis=0)`. Cuối cùng ta quay về bước 10.

Các gói được sử dụng:

- eig: Tìm trị riêng và vector riêng tương ứng.
- matrix_rank: hạng của ma trận.
- null_space: Không gian hạt nhân của ma trận.
- np.zeros(m,n): ma trận 0 cỡ $m \times n$
- Fill_diagonal(Σ, w): thêm các giá trị thực của W vào đường chéo chính của ma trận Σ
- concatenate ((a,b), axis=0) nối 2 ma trận theo hàng.
- concatenate ((a,b), axis=1) nối 2 ma trận theo cột.

Biểu diễn thuật toán bằng sơ đồ khối:



4.2 Chương trình

Sau đây là toàn bộ code của chương trình bằng python:

```
import numpy as np
from scipy.linalg import null_space
from numpy import linalg as LA
from scipy.linalg import svd
import math as mt

data=np.genfromtxt('input.txt', delimiter=' ')
A=np.array(data)

m=len(A)
n=len(A[0])

def SVD(A):
    # gan ma tran Sigma bang ma tran 0 co m.n
    sigma = np.zeros((m,n))
    v = np.eye(n)
    # r = rank ma tran -1
    r= np.linalg.matrix_rank(A) -1
    # neu A ko full rank thi s tim ker(A) v ker(A.T)
    if r != n-1:
        ns = null_space(A)
        ns =ns.T
        nd = null_space(A.T)
        nd = nd.T
    # m >=n ta tinh U truoc V sau
    if m>=n:
        # ta tim tri rieng w cua A.AT va tim vecto rieng tuong ung la u
        w,u = LA.eig(A@A.T)
        # sap xep tri rieng va vetor rieng
        for i in range(n-1):
            for j in range(i+1, n):
                if(w[j-1] < w[j]):
                    w[j], w[j-1] = w[j-1], w[j]
                    u[:,[j-1,j]] = u[:,[j,j-1]]
        U = u.T
        # ta tim r vector rieng dau tien ung voi U
        V= np.zeros((r+1,n))
        for i in range(r+1):
            V[i,:n]=(U[i,:m]@A)/(np.sqrt(w[i]))
            np.fill_diagonal(sigma, np.sqrt(w))
        #sau do tim n - r vector con lai
        if r != n-1:
            V = np.concatenate((V,ns), axis =0)

    # n < n ta tinh V truoc U sau tuong tu
    else:
        w,v = LA.eig(A.T@A)
        for i in range(n-1):
            for j in range(i+1, m):
                if(w[j-1] < w[j]):
                    w[j], w[j-1] = w[j-1], w[j]
```

```

        v[:,[j-1,j]] = v[:,[j,j-1]]

#    V =v
    U = np.zeros((r+1,m))
    for i in range(r+1):
        U[i,:m]=(v.T[i,:n]@A.T)/(np.sqrt(w[i]))
        np.fill_diagonal(sigma, np.sqrt(w))

    if r != n-1:
        U = np.concatenate((U,nd), axis = 0)

    V = v.T

    return U.T, sigma, V

```

```

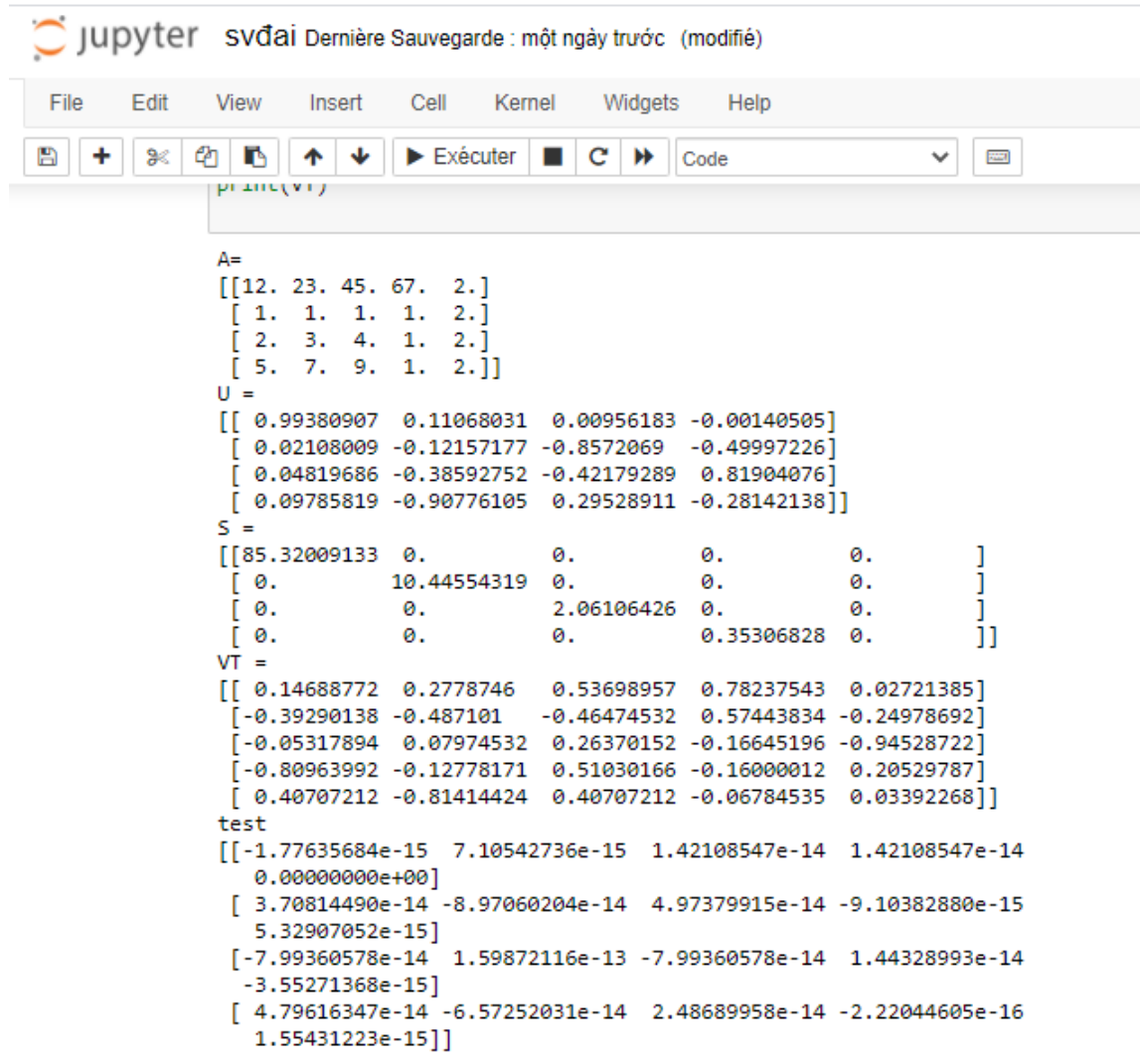
U,sigma,VT=SVD(A)
print("A=")
print(A)
print("U = ")
print(U)
print("S = ")
print(sigma)
print("VT = ")
print(VT)
print("test")
print(A- U@sigma@VT)
print("-----")
print('Using scipy')
U,S,VT = svd(A)
S=np.diag(S)
print("U=")
print(U)
print("S=")
print(S)
print("VT=")
print(VT)

```

4.3 Hệ thống ví dụ

Input là ma trận A nhập vào từ file input.txt và kết quả chạy như các ví dụ bên dưới, lưu ý VT chỉ là ký hiệu chứ không phải là chuyển vị của V, test là ma trận sai số của phép khai triển :

Ví dụ 1: Trường hợp full rank



```

jupyter svđai Dernière Sauvegarde : một ngày trước (modifié)
File Edit View Insert Cell Kernel Widgets Help
[+] [x] [print] [run] [stop] [refresh] [code]
print(VT)

A=
[[12. 23. 45. 67.  2.]
 [ 1.  1.  1.  1.  2.]
 [ 2.  3.  4.  1.  2.]
 [ 5.  7.  9.  1.  2.]]
U =
[[ 0.99380907  0.11068031  0.00956183 -0.00140505]
 [ 0.02108009 -0.12157177 -0.8572069  -0.49997226]
 [ 0.04819686 -0.38592752 -0.42179289  0.81904076]
 [ 0.09785819 -0.90776105  0.29528911 -0.28142138]]
S =
[[85.32009133  0.          0.          0.          0.          ]
 [ 0.          10.44554319  0.          0.          0.          ]
 [ 0.          0.          2.06106426  0.          0.          ]
 [ 0.          0.          0.          0.35306828  0.          ]]
VT =
[[ 0.14688772  0.2778746  0.53698957  0.78237543  0.02721385]
 [-0.39290138 -0.487101  -0.46474532  0.57443834 -0.24978692]
 [-0.05317894  0.07974532  0.26370152 -0.16645196 -0.94528722]
 [-0.80963992 -0.12778171  0.51030166 -0.16000012  0.20529787]
 [ 0.40707212 -0.81414424  0.40707212 -0.06784535  0.03392268]]
test
[[-1.77635684e-15  7.10542736e-15  1.42108547e-14  1.42108547e-14
  0.00000000e+00]
 [ 3.70814490e-14 -8.97060204e-14  4.97379915e-14 -9.10382880e-15
  5.32907052e-15]
 [-7.99360578e-14  1.59872116e-13 -7.99360578e-14  1.44328993e-14
 -3.55271368e-15]
 [ 4.79616347e-14 -6.57252031e-14  2.48689958e-14 -2.22044605e-16
  1.55431223e-15]]
-----

```


Ví dụ 2: Các trường hợp không full rank, chạy trên Jupyter

```
A=
[[ 1.  2.  3.  4.  5.]
 [10. 20. 30. 40. 50.]
 [12. 121. 1. 1. 1.]
 [ 4.  5. 67.  7.  8.]
 [ 3.  3.  3.  3.  3.]]

U =
[[ 2.87497716e-02 -6.72055331e-02 -6.71916493e-02 -6.56805267e-03
  -9.95037190e-01]
 [ 2.87497716e-01 -6.72055331e-01 -6.71916493e-01 -6.56805267e-02
   9.95037190e-02]
 [ 9.46720159e-01  3.05445586e-01  1.01018475e-01 -1.48054632e-02
   2.16579139e-17]
 [ 1.37799955e-01 -6.69287145e-01  7.29940585e-01 -1.58943858e-02
  -1.10741926e-17]
 [ 3.53641960e-02 -5.08208553e-02 -3.15518374e-02  9.97582626e-01
   2.59879612e-15]]

S =
[[1.25403684e+02 0.00000000e+00 0.00000000e+00 0.00000000e+00
  0.00000000e+00]
 [0.00000000e+00 8.48434817e+01 0.00000000e+00 0.00000000e+00
  0.00000000e+00]
 [0.00000000e+00 0.00000000e+00 4.58457066e+01 0.00000000e+00
  0.00000000e+00]
 [0.00000000e+00 0.00000000e+00 0.00000000e+00 2.16122970e+00
  0.00000000e+00]
 [0.00000000e+00 0.00000000e+00 0.00000000e+00 0.00000000e+00
  7.54487692e-07]]

VT =
[[ 1.18989019e-01  9.66125411e-01  1.51483513e-01  1.08707494e-01
   1.32961382e-01]
 [-7.01529786e-02  2.34366854e-01 -7.66735769e-01 -3.73429540e-01
  -4.61321336e-01]
 [-5.99626061e-02  4.81085345e-02  6.22813227e-01 -4.80513463e-01
  -6.12617786e-01]
 [ 9.66177476e-01 -9.48218220e-02 -3.56747795e-02  9.86421921e-02
  -2.15654666e-01]
 [ 2.09345991e-01 -1.91900492e-02  1.76941795e-16 -7.79813817e-01
   5.89657875e-01]]

test
[[ 1.57165097e-07 -1.44068379e-08 -2.66453526e-15 -5.85440030e-07
   4.42681679e-07]
 [-1.57165108e-08  1.44068935e-09 -3.55271368e-15  5.85439963e-08
  -4.42681767e-08]
 [ 0.00000000e+00 -1.42108547e-14  1.77635684e-15  4.99600361e-15
   3.55271368e-15]
 [ 8.88178420e-16 -2.66453526e-15 -1.42108547e-14  2.66453526e-15
  -1.77635684e-15]
 [ 0.00000000e+00 -8.88178420e-16 -4.44089210e-16  1.33226763e-15
   1.33226763e-15]]
```

```

A=
[[ 1.  2.  4.]
 [ 2.  4.  8.]
 [ 4.  8. 16.]
 [ 1.  1.  1.]
 [ 2.  2.  2.]
 [ 4.  5.  6.]
 [ 6.  7.  7.]]
U =
[[-1.79224627e-01 -1.22686835e-01 -3.66151238e-03  7.67287978e-01
 -1.74830250e-02 -1.86091852e-02 -3.49332655e-05]
 [-3.58449253e-01 -2.45373670e-01 -7.32302476e-03  3.86504451e-02
  1.16653973e-01 -6.95408544e-01 -1.05630180e-03]
 [-7.16898506e-01 -4.90747339e-01 -1.46460495e-02 -2.74544670e-01
 -1.61375923e-01  2.79819987e-01  3.79724104e-04]
 [-6.40207189e-02  1.20754439e-01 -4.16769971e-01 -5.07227507e-02
 -8.59360060e-02 -5.80278963e-02  8.94299980e-01]
 [-1.28041438e-01  2.41508878e-01 -8.33539941e-01 -1.01433531e-01
 -1.71871383e-01 -1.16059215e-01 -4.47464310e-01]
 [-3.39423014e-01  3.48797587e-01 -2.96952371e-02  5.07179626e-01
  8.59357544e-01  5.80292651e-01  1.25728090e-03]
 [-4.35600682e-01  6.99527571e-01  3.61041009e-01 -2.53589813e-01
 -4.29678772e-01 -2.90146326e-01 -6.28640451e-04]]
S =
[[25.34030431  0.      0.      ]
 [ 0.      4.98341503  0.      ]
 [ 0.      0.      0.18588182]
 [ 0.      0.      0.      ]
 [ 0.      0.      0.      ]
 [ 0.      0.      0.      ]
 [ 0.      0.      0.      ]]
VT =
[[-0.31787767 -0.49698921 -0.80743762]
 [ 0.72635019  0.4197134  -0.5442941 ]
 [-0.60940068  0.75950141 -0.22757069]]
test
[[ 9.99200722e-16  1.11022302e-15  1.77635684e-15]
 [-4.44089210e-16  0.00000000e+00  0.00000000e+00]
 [-2.66453526e-15 -1.77635684e-15 -3.55271368e-15]
 [-8.88178420e-16  4.44089210e-16 -4.44089210e-16]
 [-1.77635684e-15  1.11022302e-15 -8.88178420e-16]
 [ 4.88498131e-15 -6.21724894e-15  0.00000000e+00]
 [ 0.00000000e+00  7.10542736e-15  3.55271368e-15]]

```

```

A=
[[12. 12. 12. 12. 12. 12.]
 [36. 36. 36. 36. 36. 36.]
 [ 1.  2.  3. 54.  6.  6.]]
U =
[[-2.93093373e-01  1.18727734e-01  9.48683298e-01]
 [-8.79280118e-01  3.56183202e-01 -3.16227766e-01]
 [-3.75450061e-01 -9.26842625e-01  2.77555756e-17]]
S =
[[9.87257640e+01  0.00000000e+00  0.00000000e+00  0.00000000e+00
  0.00000000e+00  0.00000000e+00]
 [0.00000000e+00  4.35341649e+01  0.00000000e+00  0.00000000e+00
  0.00000000e+00  0.00000000e+00]
 [0.00000000e+00  0.00000000e+00  8.72385327e-07  0.00000000e+00
  0.00000000e+00  0.00000000e+00]]
VT =
[[-3.60054492e-01 -3.63857451e-01 -3.67660410e-01 -5.61611334e-01
 -3.79069288e-01 -3.79069288e-01]
 [ 3.05977741e-01  2.84687735e-01  2.63397730e-01 -8.22392568e-01
  1.99527712e-01  1.99527712e-01]
 [-7.80330721e-02  7.60464629e-01  1.39214076e-01  6.39444872e-02
 -4.42795060e-01 -4.42795060e-01]
 [-4.92230844e-01 -3.56002042e-01 -1.55314410e-01 -9.06480337e-02
  5.47097665e-01  5.47097665e-01]
 [-5.29805358e-01 -5.28607230e-01  4.57960223e-01 -7.06161466e-02
  3.35534255e-01  3.35534255e-01]
 [-2.30718159e-16 -2.47159554e-16  1.22580191e-16 -4.62447077e-17
 -7.07106781e-01  7.07106781e-01]]
test
[[ 6.45815277e-08 -6.29373757e-07 -1.15215993e-07 -5.29215693e-08
  3.66464898e-07  3.66464898e-07]
 [-2.15271783e-08  2.09791239e-07  3.84053251e-08  1.76405379e-08
 -1.22154951e-07 -1.22154951e-07]
 [ 3.55271368e-15 -1.24344979e-14  1.24344979e-14  0.00000000e+00
 -3.55271368e-15 -3.55271368e-15]]

```

```

A=
[[ 4.  4.  4.  4.  4.  4.  4.]
 [ 6.  6.  6.  6.  6.  6.  6.]
 [ 1.  1.  1.  1.  1.  1.  1.]
 [ 5. 55.  5.  5.  5.  5.  4.]]
U =
[[-1.12516513e-01  5.37798128e-01 -8.34629747e-01  3.88122692e-02]
 [-1.68774769e-01  8.06697192e-01  5.33744766e-01 -1.89397047e-01]
 [-2.81291282e-02  1.34449532e-01  1.36050392e-01  9.81133205e-01]
 [-9.78807368e-01 -2.04783144e-01 -5.55111512e-17 -9.71445147e-17]]
S =
[[5.73928768e+01 0.00000000e+00 0.00000000e+00 0.00000000e+00
 0.00000000e+00 0.00000000e+00 0.00000000e+00]
 [0.00000000e+00 1.55903075e+01 0.00000000e+00 0.00000000e+00
 0.00000000e+00 0.00000000e+00 0.00000000e+00]
 [0.00000000e+00 0.00000000e+00 0.00000000e+00 1.72046391e-07 0.00000000e+00
 0.00000000e+00 0.00000000e+00]
 [0.00000000e+00 0.00000000e+00 0.00000000e+00 1.42681088e-07
 0.00000000e+00 0.00000000e+00 0.00000000e+00]]
VT =
[[-1.11248660e-01 -9.63974139e-01 -1.11248660e-01 -1.11248660e-01
 -1.11248660e-01 -1.11248660e-01 -9.41941502e-02]
 [ 3.91391221e-01 -2.65373068e-01  3.91391221e-01  3.91391221e-01
  3.91391221e-01  3.91391221e-01  4.04526507e-01]
 [-9.13475039e-01  3.69593233e-03  1.81245623e-01  1.81245623e-01
  1.81245623e-01  1.81245623e-01  1.84796616e-01]
 [-7.01809137e-01 -8.56327862e-03  2.84634087e-01  2.84634087e-01
  2.84634087e-01  2.84634087e-01 -4.28163931e-01]
 [ 1.61356208e-01  1.37428204e-02  2.71072335e-01 -3.77770794e-01
 -3.77770794e-01 -3.77770794e-01  6.87141019e-01]
 [ 2.76486979e-17  2.54042429e-17 -1.62175287e-17 -2.11324865e-01
 -5.77350269e-01  7.88675135e-01 -3.52836756e-17]
 [-2.76486979e-17 -1.35750067e-17  3.75838292e-17 -7.88675135e-01
  5.77350269e-01  2.11324865e-01  4.28792511e-17]]
test
[[-1.27284022e-07  5.78133097e-10  2.44497298e-08  2.44497298e-08
  2.44497298e-08  2.44497298e-08  2.89069519e-08]
 [ 6.49181127e-08 -5.70812730e-10 -8.95180641e-09 -8.95180641e-09
 -8.95180641e-09 -8.95180641e-09 -2.85401001e-08]
 [ 1.19627357e-07  1.11225396e-09 -4.40880976e-08 -4.40880976e-08
 -4.40880976e-08 -4.40880976e-08  5.56127754e-08]
 [-1.95399252e-14 -2.84217094e-14 -8.88178420e-15 -1.24344979e-14
 -1.24344979e-14 -1.24344979e-14 -8.88178420e-15]]
-----

```

Ví dụ 3: Trường hợp không full rank, chạy trên Sublime text

```

A=
[[1. 1. 2. 2. 3. 3.]
 [2. 2. 4. 4. 6. 6.]
 [3. 3. 6. 6. 9. 9.]]
U =
[[-2.67261242e-01 -3.58568583e-01 -8.94427191e-01]
 [-5.34522484e-01 -7.17137166e-01  4.47213595e-01]
 [-8.01783726e-01  5.97614305e-01  1.11022302e-16]]
S =
[[1.97989899e+01 0.00000000e+00 0.00000000e+00 0.00000000e+00 0.00000000e+00
 0.00000000e+00 9.94162806e-08 0.00000000e+00 0.00000000e+00 0.00000000e+00]
 [0.00000000e+00 0.00000000e+00 0.00000000e+00 0.00000000e+00 0.00000000e+00
 0.00000000e+00 0.00000000e+00 0.00000000e+00 0.00000000e+00 0.00000000e+00]]
VT =
[[-0.18898224 -0.18898224 -0.37796447 -0.37796447 -0.56694671 -0.56694671]
 [-0.25354628 -0.25354628 -0.50709255 -0.50709255  0.42257713  0.42257713]
 [-0.98198051  0.03636965  0.0727393  0.0727393  0.10910895  0.10910895]
 [ 0.21618614 -0.31409772  0.73521049  0.30755702 -0.33127057 -0.33127057]
 [ 0.96489614 -0.0258987  0.04087702 -0.21837006 -0.09733523 -0.09733523]
 [ 0.  0.  0.  0. -0.70710678  0.70710678]]
test
[[-9.03830477e-09 -9.03830344e-09 -1.80766095e-08 -1.80766095e-08  1.50638422e-08  1.50638422e-08]
 [-1.80766095e-08 -1.80766095e-08 -3.61532191e-08 -3.61532191e-08  3.01276843e-08  3.01276843e-08]
 [ 1.50638413e-08  1.50638453e-08  3.01276826e-08  3.01276826e-08 -2.51064005e-08 -2.51064005e-08]]
-----

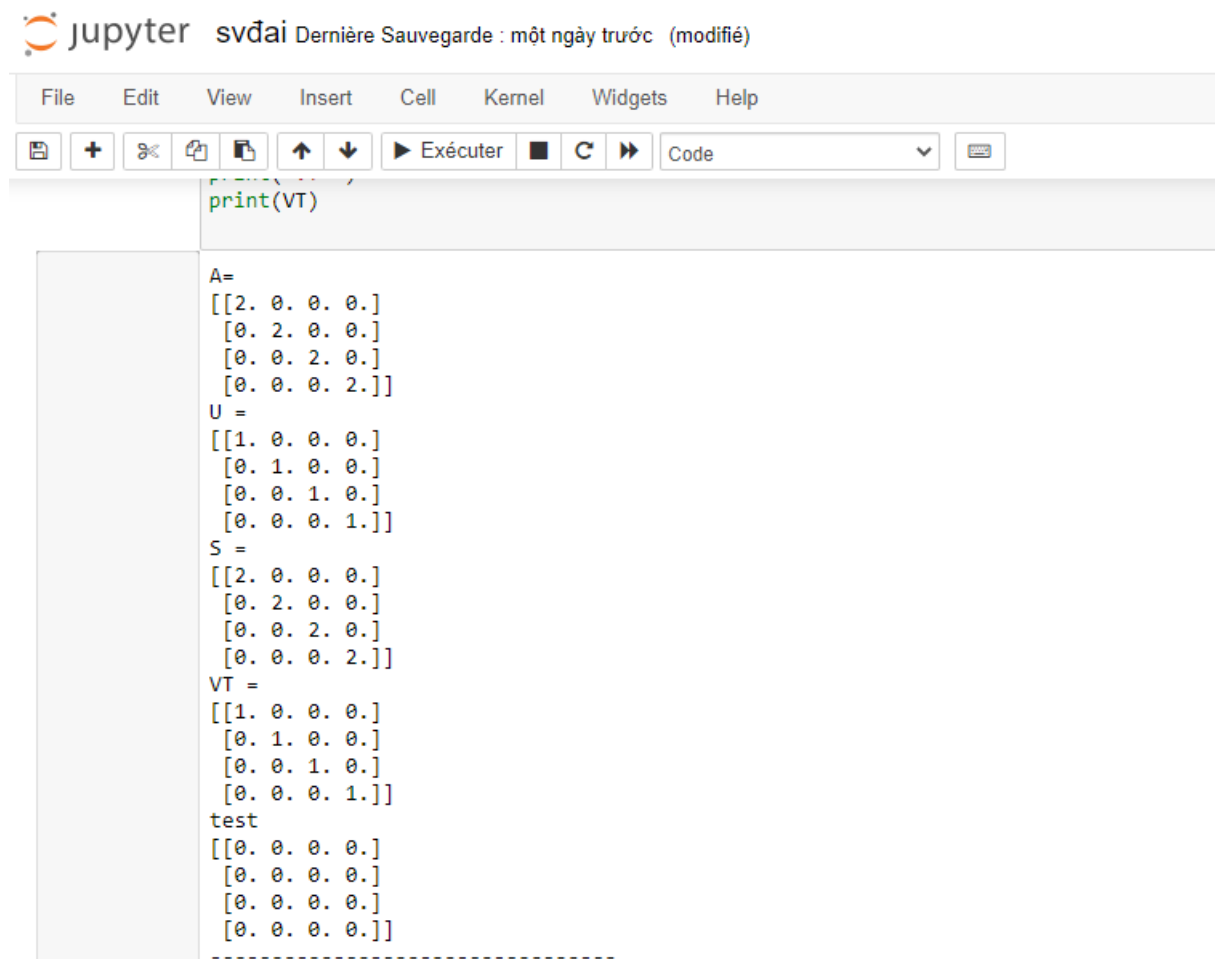
```

```

np.linalg.eigvals(sigma, np.sqrt(w))
A=
[[ 1.  2.  4.]
 [ 2.  4.  6.]
 [ 3.  6.  7.]
 [ 4.  8. 12.]]
U =
[[ 0.22833316  0.55485491 -0.70662319  0.40248927]
 [ 0.37719103  0.13314252 -0.24269439 -0.87360549]
 [ 0.48631124 -0.77685351 -0.3533116  0.20124463]
 [ 0.75438205  0.26628504  0.56298669  0.18524695]]
S =
[[1.98345994e+01 0.00000000e+00 0.00000000e+00]
 [0.00000000e+00 1.26042374e+00 0.00000000e+00]
 [0.00000000e+00 0.00000000e+00 1.18130204e-07]
 [0.00000000e+00 0.00000000e+00 0.00000000e+00]]
VT =
[[ 2.75235059e-01  5.50470118e-01  7.88180380e-01]
 [-3.52484982e-01 -7.04969963e-01  6.15444302e-01]
 [-8.94427191e-01  4.47213595e-01  1.66533454e-16]]
test
[[-7.46610038e-08  3.73305067e-08  1.77635684e-15]
 [-2.56428128e-08  1.28214084e-08 -8.88178420e-16]
 [-3.73305022e-08  1.86652525e-08 -1.77635684e-15]
 [ 5.94845346e-08 -2.97422691e-08  1.77635684e-15]]
-----
Using scipy

```

Ví dụ 4: Các trường hợp có trị riêng bội



The screenshot shows a Jupyter Notebook interface. At the top, the Jupyter logo is followed by the text "svđai Dernière Sauvegarde : một ngày trước (modifié)". Below this is a menu bar with options: File, Edit, View, Insert, Cell, Kernel, Widgets, and Help. Under the menu bar is a toolbar with icons for saving, adding cells, undo, redo, and running code. The main area contains a code cell with the following text:

```
print(VT)

A=
[[2. 0. 0. 0.]
 [0. 2. 0. 0.]
 [0. 0. 2. 0.]
 [0. 0. 0. 2.]]
U =
[[1. 0. 0. 0.]
 [0. 1. 0. 0.]
 [0. 0. 1. 0.]
 [0. 0. 0. 1.]]
S =
[[2. 0. 0. 0.]
 [0. 2. 0. 0.]
 [0. 0. 2. 0.]
 [0. 0. 0. 2.]]
VT =
[[1. 0. 0. 0.]
 [0. 1. 0. 0.]
 [0. 0. 1. 0.]
 [0. 0. 0. 1.]]
test
[[0. 0. 0. 0.]
 [0. 0. 0. 0.]
 [0. 0. 0. 0.]
 [0. 0. 0. 0.]]
```

jupyter svđai Dernière Sauvegarde : một ngày trước (modifié)

File Edit View Insert Cell Kernel Widgets Help

Exécuter Code

```
print(VT)
```

```
A=
[[10.  0.  0.  9.]
 [ 0. 15.  0.  0.]
 [ 0.  0. 15.  0.]
 [ 3.  0.  0.  5.]]
U =
[[ 0.92193508  0.          0.         -0.38734443]
 [ 0.          1.          0.          0.          ]
 [ 0.          0.          1.          0.          ]
 [ 0.38734443  0.          0.          0.92193508]]
S =
[[14.57774721  0.          0.          0.          ]
 [ 0.          15.          0.          0.          ]
 [ 0.          0.          15.          0.          ]
 [ 0.          0.          0.          1.57774721]]
VT =
[[ 0.71213912  0.          0.          0.70203837]
 [ 0.          1.          0.          0.          ]
 [ 0.          0.          1.          0.          ]
 [-0.70203837  0.          0.          0.71213912]]
test
[[-3.55271368e-15  0.00000000e+00  0.00000000e+00 -3.55271368e-15]
 [ 0.00000000e+00  0.00000000e+00  0.00000000e+00  0.00000000e+00]
 [ 0.00000000e+00  0.00000000e+00  0.00000000e+00  0.00000000e+00]
 [ 0.00000000e+00  0.00000000e+00  0.00000000e+00 -1.77635684e-15]]
```

5 Ứng dụng

Phân tích SVD là dạng phân tích có rất nhiều ứng dụng, trong khuôn khổ bản báo cáo này, một vài ứng dụng đơn giản và quan trọng nhất sẽ được nêu ra. Và một trong những ứng dụng ấn tượng nhất đó chính là sử dụng SVD trong hiệu chỉnh hình ảnh kĩ thuật số. Nhờ đó hình ảnh kĩ thuật số được truyền đi một cách hiệu quả bằng vệ tinh, internet. . . .

5.1 Ứng dụng trong xử lý ảnh

Mục đích của việc nén ảnh số là mã hoá các dữ liệu ảnh về một dạng thu gọn, tối thiểu hoá cả số bit dùng để biểu diễn ảnh lẫn các sai khác do quá trình nén gây ra. Tầm quan trọng của vấn đề nén ảnh có thể thấy rõ qua các số liệu cụ thể: với một bức ảnh trắng đen kích thước 512×512 pixels, mỗi pixel được biểu diễn bởi 8 bits (biểu diễn một trong 256 giá trị mức xám), cần khoảng 256 Kbytes dữ liệu, và ảnh màu cần gấp ba lần con số này. Với các dữ liệu video, cần 25 frames trên một giây, như vậy một đoạn video chỉ 30s phải cần đến 540MB dữ liệu, một con số quá lớn. Do đó vấn đề nén ảnh là hết sức cần thiết.

Nói chung, các phương pháp nén ảnh chủ yếu được phân thành 2 nhóm: nhóm không tổn hao và nhóm có tổn hao. Các phương pháp nén ảnh không tổn hao cho phép biểu diễn ảnh với chất lượng hoàn toàn ngang bằng với ảnh gốc. Các phương pháp này dựa trên các giải thuật nén được áp dụng cho tất cả các đối tượng dữ liệu nói chung chứ không chỉ riêng dữ liệu ảnh, ví dụ mã Huffman, mã số học, mã Golomb, ... Tuy nhiên, các phương pháp này không lợi dụng được những đặc tính riêng của dữ liệu ảnh và tỷ lệ nén rất thấp. Do đó, trong thực tế, các phương pháp nén có tổn hao là các phương pháp được sử dụng chủ yếu. Với các phương pháp này, luôn có sự đánh đổi giữa dung lượng ảnh với chất lượng ảnh.

Ý tưởng cơ sở của việc hiệu chỉnh ảnh là giảm số lượng thông tin truyền đi mà không làm mất đi những thông tin thực chất. Trong một bức ảnh kĩ thuật số, mỗi điểm ảnh được thể hiện bởi 3 giá trị màu Blue, Green, Red với các trị số từ 0 đến 255. Như vậy với một hình ảnh có độ lớn là 340×280 pixels thì chúng ta phải lưu trữ 3 ma trận (thể hiện màu sắc của các điểm) có cùng độ lớn là 340×280 tức là phải lưu trữ 285600 số. Tuy nhiên trong thực tế, khi truyền, lưu trữ thông tin ảnh chúng ta có thể không cần những hình ảnh, hoặc một số phần của hình ảnh đó có độ nét quá lớn. Sử dụng phân tích SVD chúng ta có thể loại bỏ rất nhiều thông tin không cần thiết đó. Ví dụ một hình ảnh 340×280 pixels được phân tích thành 3 ma trận A, B, C có cùng độ lớn 340×280 . Giả sử ma trận A có phân tích SVD là:

$$A = UV^T = \sigma_1 u_1 v_1^T + \dots + \sigma_r u_r v_r^T$$

Với giá trị $k < r$ bất kì ta có:

$$A = UV^T = \sigma_1 u_1 v_1^T + \dots + \sigma_k u_k v_k^T$$

A_k như đã chứng minh là xấp xỉ tốt nhất được xây dựng từ k giá trị kì dị đầu tiên của ma trận A. Ví dụ với $k = 20$ thì ma trận A_k thể hiện các dữ liệu của ma trận A tương ứng với 20 giá trị kì dị đầu tiên. Như vậy chúng ta chỉ cần lưu trữ 20 giá trị kì dị, 20 vec tơ u_i , 20 vec tơ v_i tương đương với $20 + 20 \times 280 + 20 \times 340 = 12420$ số. Tương tự với 2 ma trận B, C thì số lượng các số phải lưu trữ là $12420 \times 3 = 37260$ số. Rõ ràng phân tích SVD đã giúp giảm lượng thông tin cần lưu trữ một cách đáng kể.

Có thể hiểu đơn giản ứng dụng này là hiệu chỉnh độ nét của bức ảnh. Đây là phương pháp đã được ứng dụng trong thực tế như là: phương pháp thủy vân trên ảnh số. Trong ứng dụng này chúng ta sẽ hiệu chỉnh độ nét của một ảnh gốc theo tham số k tùy chọn:



Dòng lệnh trong chương trình python:

```
import matplotlib.pyplot as plt
import numpy as np
import scipy.linalg as ln
from PIL import Image
import urllib.request
from io import BytesIO

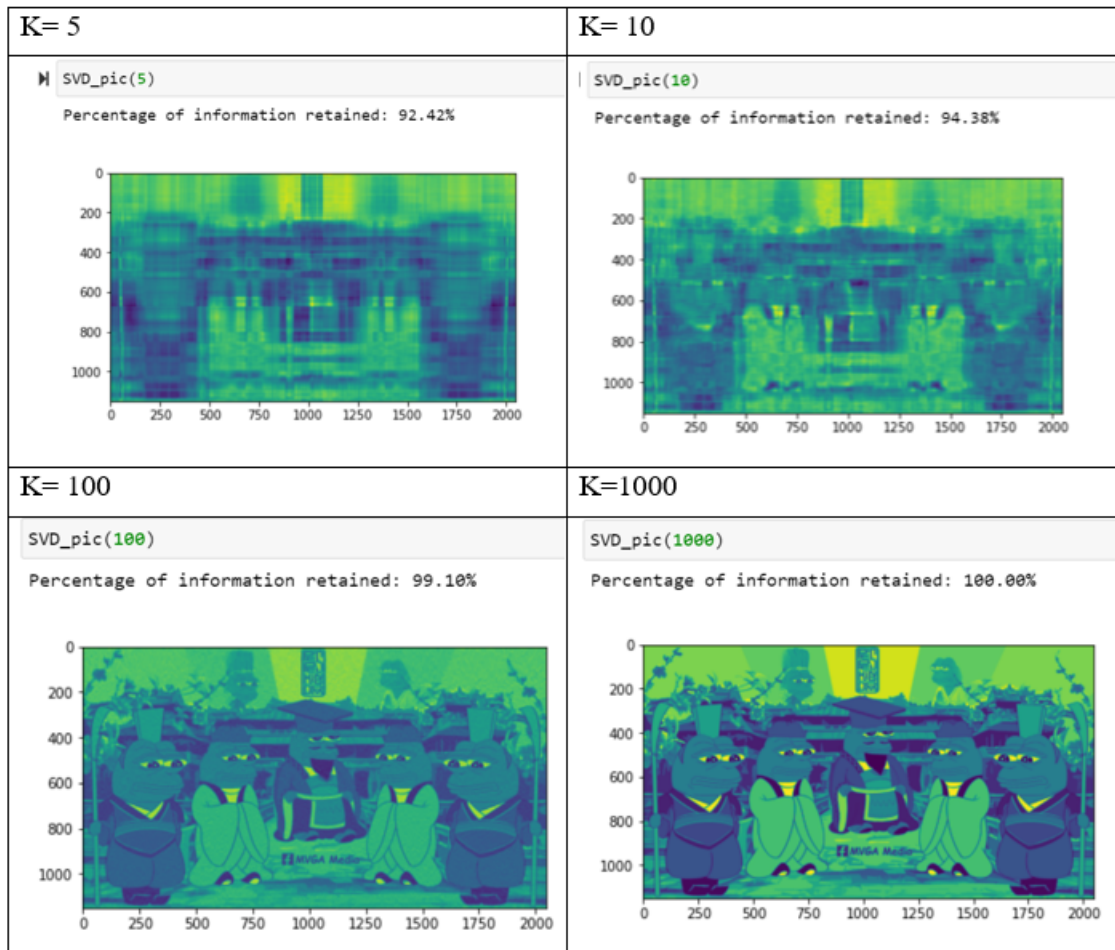
%matplotlib inline

url = str('https://bom.to/p7kEjxQ2VbSmM')
with urllib.request.urlopen(url) as url:
    f = BytesIO(url.read())

ig = np.array(Image.open(f))
print('Image shape: %s'%str(ig.shape))
# Convert to grey
ig = ig.dot([0.299, 0.5870, 0.114])
plt.imshow(ig)
def SVD_pic(n_evl):
    """
    n_evl: number of highest eigenvalues taken
    """
    #Take m, n shape
    m = ig.shape[0]
    n = ig.shape[1]
    #Singular Value Decomposition
    U, S, V = ln.svd(ig)
    #Get id position of n highest eigenvalues
    id_trunc = np.argsort(S)[::-1][:n_evl]
    #Extract matrix U_t, V_t, S_t
    U_t = U[np.ix_(np.arange(m), id_trunc)]
    V_t = V[np.ix_(id_trunc, np.arange(n))]
    S_diag = S[id_trunc]
    S_t = np.zeros((n_evl, n_evl))
    np.fill_diagonal(S_t, S_diag)
```

```
#Return picture
A = np.dot(U_t, S_t.dot(V_t))
#Norm Frobenius
fb = ln.norm(A-ig, 'fro')
prt_retain = (1-fb**2/np.sum(S**2))*100
plt.imshow(A)
print('Percentage of information retained: %.2f%% \n'%(prt_retain, '%'))
```

Với các giá trị khác nhau của tham số k, chương trình sẽ cho ra các ảnh hiệu chỉnh có độ nét khác nhau:



5.2 Ứng dụng tính số điều kiện

Do $A = U\Sigma V^T \Rightarrow A^T = V\Sigma^T U^T$

Ta cũng đã có $A^+ = V\Sigma^{-1}U^T$ (là ma trận nghịch ảnh suy rộng)

$A^T A = V\Sigma^T \Sigma V^T \Rightarrow (A^T A)^{-1} = V(\Sigma^T \Sigma)^{-1} V^T$

$\Rightarrow (A^T A)^{-1} A^T = V(\Sigma^T \Sigma)^{-1} V^T V \Sigma^T U^T = V(\Sigma^T \Sigma)^{-1} \Sigma^T U^T = A^+$

$\Rightarrow A^+ = (A^T A)^{-1} A^T$

Ta xét hệ phương trình: $Ax = y: \Rightarrow A^+ Ax = A^+ y \Rightarrow (A^T A)^{-1} A^T Ax = A^+ y$

$\Rightarrow x = A^+ y$. Tìm xấp xỉ nghiệm x của phương trình $Ax = y$ ($Ax = y$ vô nghiệm)

Chú ý: $A_{m \times n} (m > n): A_{m \times n}^+$. $\text{Rank}(A) = n$

Thuật toán cụ thể:

Input: Ma trận $A_{m \times n} (m > n)$; $\text{rank}(A) = n$; y

Output: Cond , A^+ , x

Bước 1: Tính U, Σ, V theo SVD

$$U, \Sigma, V = \text{Ln.svd}(A)$$

Bước 2: Xuất giá trị max, min của (σ)

omax = np.max(σ)

omin = np.min(σ)

Tính hệ số cond của ma trận: $\text{cond} = \frac{\text{omax}}{\text{omin}}$

Bước 3: Tính $A^+ = V\Sigma^{-1}U^T$

S là nghịch ảnh của Σ

$S = \text{Ln.inv}(\Sigma)$

Nối ma trận S với ma trận 0 đưa về dạng ma trận S^{-1} thành ma trận cỡ $n \times m$

$\Sigma^{-1} = \text{np.concatenate}((s, \text{np.zeros}((n, n-m))), \text{axis} = 1)$

$A^+ = \text{np.dot}(\text{np.dot}(V, \Sigma^{-1}), U^T)$

Bước 4: Tính $x = A^+y$

Code Python cho bài toán này:

```
import matplotlib.pyplot as plt
import numpy as np
import scipy.linalg as ln
from PIL import Image
import urllib.request

data=np.genfromtxt('hoiquy2.txt',delimiter=' ')
A = np.array(data)

m=len(A)
n=len(A[0])
AT=A.T
print(A)
#y1
#y = [1,1,1,10002]
#y =[1.7 ,1.2, 1.5,10003]
#y2
#y = [1120 ,1022 ,29 ,1004]
y = [1020.9, 1022.4, 29.2, 1004.8]

n_diag = min(m,n)
U, S_diag, V_T = ln.svd(A)
U_T = U.T
V = V_T.T
omax = np.max(S_diag)
omin = np.min(S_diag)
cond = omax/omin
print("gia tri so dieu kien cond la:")
print(cond)

S = np.zeros((n_diag, n_diag))
# gn S voi ma tran co duong cheo chinh =S_diag
np.fill_diagonal(S, S_diag)
S1 = ln.inv(S)

# Noi ma tran S1 voi ma tran 0 dua ve dang ma trang S_t thanh ma tran co n*m
```

```
S_t = np.concatenate((S1, np.zeros((n, m - n))), axis = 1)

# Tính ma trận Att theo công thức  $A^+ = (V.S_t)^{-1}.U_T$ 
#  $S_t = np.dot(np.dot(V, S_t), U_T)$ 
A_t = np.dot(np.dot(V, S_t), U_T)
print('Ma trận nghịch đảo suy rộng A+: \n %s \n' % A_t)

x = np.dot(A_t, y)
print(' nghiệm gần đúng của x là: \n %s' % x)
```

Cuối cùng là kết quả chạy code:

```
[[1000.  10.  10.]
 [ 12. 1000.  10.]
 [ 27.   7.  -5.]
 [  4. 100. 900.]]
giá trị số điều kiện cond là:
1.175765161536078
Ma trận nghịch đảo suy rộng A+:
[[ 9.99425793e-04 -9.09727478e-06  2.69181885e-05 -1.08541047e-05]
 [-1.21611029e-05  1.00116709e-03  7.07024191e-06 -1.09496763e-05]
 [-2.91313973e-06 -1.11152085e-04 -7.49565534e-06  1.11233686e-03]]
nghiệm gần đúng của x là:
[1.00089254 1.00038218 1.00084129]
```

6 Kết luận

Trên đây là toàn bộ phần báo cáo về khai triển kỳ dị của ma trận và ứng dụng. Những nội dung nổi bật mà bản báo cáo nêu ra được:

- Trong chương hai đã nhắc lại một số khái niệm cơ bản nhất về ma trận như các không gian con, trị riêng vector riêng, chuẩn. Đây là những nền tảng kiến thức quan trọng nhất để hiểu rõ hơn về SVDs
- Chương ba đã nêu định nghĩa, phát biểu chứng minh sự tồn tại khai triển kỳ dị cùng một số tính chất và hệ quả. Phần định nghĩa đã được nêu ở dạng tổng quát cho số phức và trong trường số thực, thoạt nhìn có sự khác nhau nhưng về bản chất chúng vẫn tương tự. Về phần chứng minh sự tồn tại khác có trên wiki và các tài liệu bằng tiếng Anh dành cho những bạn muốn tìm hiểu sâu hơn.
- Chương bốn đi vào thuật toán cụ thể, chương trình chạy cho thuật toán. Sau bài buổi thuyết trình tuần vừa rồi, nhóm đã tiếp thu những nhận xét của cô và gấp rút sửa chữa, bổ sung và hoàn thiện trong chưa đầy một tuần. Nhóm cũng đã chạy thử rất nhiều trường hợp từ ma trận full rank đến suy biến, tuy kết quả trả về chấp nhận được nhưng chắc chắn vẫn chưa phải chương trình tối ưu.
- Chương năm đã giới thiệu một số ứng dụng quan trọng nhất, cần lưu ý có rất nhiều lĩnh vực ứng dụng SVDs trong cuộc sống mà chúng ta đã, đang và sẽ khám phá trong tương lai.

Còn rất nhiều điều mà nhóm mình muốn chia sẻ với cô và các bạn. Bởi ngoài những ứng dụng hữu ích mà SVD mang lại, phương pháp này còn thể hiện vẻ đẹp và đặc biệt của toán học, thể hiện sự tổng quát hóa của phương pháp chéo hóa và chéo hóa trực giao. Bởi một ma trận không đơn thuần chỉ là những con số theo hàng hay theo cột, ẩn đằng sau đó là sự gặp nhau

và giao thoa của bốn không gian con cơ bản nhất, và nếu phải chọn một con số đại diện quan trọng nhất cho ma trận, câu trả lời chắc chắn sẽ là giá trị kỳ dị lớn nhất chứ không phải trị riêng hay con số nào khác. Mong rằng bản báo cáo này có thể giúp các bạn, một phần nào đó hiểu hơn về phương pháp vô cùng thú vị này, cũng như cung cấp thêm thông tin để có thể làm bài thi cuối kỳ một cách tốt nhất, đạt điểm cao nhất. Một lần nữa, nhóm em xin được gửi lời cảm ơn cô Yến đã giảng dạy, giúp đỡ chúng em trong suốt quá trình học tập môn học này. Hẹn gặp lại cô và các bạn vào các môn học tiếp theo.