TRƯỜNG ĐẠI HỌC BÁCH KHOA HÀ NỘI

VIỆN TOÁN ỨNG DỤNG VÀ TIN HỌC

BÁO CÁO MÔN HỌC

GIẢI TÍCH SỐ

<u>Đề tài:</u>

Khai triển kỳ dị của ma trận (SVD) và Ứng dụng

Giảng viên: TS. Hà Thị Ngọc Yến

Nhóm 11

Bùi Đức Tài	20185475
Đào Thị Thu Hà	20185444
Đặng Thị Ánh	20185433
Vũ Minh Nguyệt	20185469
Nguyễn Văn Hiệp	20185448

Hà Nội, 6-2021

MỤC LỤC

Lời mở đầu

Phương pháp *phân tích suy biến* (singular value decomposition) được viết tắt là SVD là một trong những phương pháp thuộc nhóm matrix factorization được phát triển lần đầu bởi những nhà hình học vi phân. Phương pháp này sẽ tìm ra một lớp các ma trận xấp xỉ tốt nhất với một ma trận cho trước dựa trên khoảng cách norm Frobenios giữa 2 ma trận. Người ta đã chứng minh được rằng ma trận xấp xỉ tốt nhất được biểu diễn dưới dạng tích của 3 ma trận rất đặc biệt bao gồm 2 *ma trận trực giao* (orthogonal matrix) và 1 *ma trận đường chéo* (diagonal matrix).

Hiện nay phân tích SVD của ma trận xuất hiện rất nhiều trong các ứng dụng thực tế như về tín hiệu số, tính các giá trị xấp xỉ trong kĩ thuật, công nghệ thông tin, và được ứng dụng trong các công cụ tìm kiếm trên các website. Tại đây nhóm chúng em thực hiện đề tài này nhằm mục đích đưa đến cho người đọc những kiến thức cơ bản nhất về khai triển SVD và tạo một cái nhìn tổng quan về cách khai triển cũng như một số tính chất, hệ quả quan trọng liên quan đến dạng khai triển này. Cùng với cơ sở lí thuyết kết hợp với ngôn ngữ lập trình Python nhóm em cũng đưa ra chi tiết một số ứng dụng của SVD như xử lý nén hình ảnh, phân tích quang phổ,….

Nội dung báo cáo gồm 3 phần:

Chương 1: Kiến thức cơ bản

Hệ thống lại một số khái niệm của đại số tuyến tính: hạng của ma trận, trị riêng và vecto riêng, hệ trực giao và trực chuẩn, chuẩn vecto, chuẩn ma trận.

Chương 2: Khai triển giá trị kì dị

Trình bày về khai triển giá trị kì dị (Singular Value Decomposition) của ma trận.

Chương 3: Ứng dụng

Các ứng dụng của SVD và thực hành trên ngôn ngữ Python.

CHƯƠNG 1: KIẾN THỨC CƠ BẢN

1.1. Hạng của ma trận

Cho A là ma trận thực cỡ m$\times n$. Giả sử $m \geq n$.

Định nghĩa 1.1. *Tập ảnh của ma trận A* được *định nghĩa bởi*:

$$\text{Im}A = \{y \in R^m, \exists x \in R^n: y = Ax\}$$

Nếu $A = (a_1 \dots a_n)$ (với $a_1, \dots, a_n$ là các cột của A) thì

$$\text{Im}A = span\{a_1, \dots, a_n\},$$

Với $span\{a_1, \dots, a_n\}$ là không gian con của R^m sinh bởi n vector $a_1, a_2, \dots, a_n$.

Định nghĩa 1.2. Tập không điểm của ma trận A được định nghĩa bởi :

$$\text{Ker}A = \{x \in R^n: Ax = 0\}$$

Định nghĩa 1.3. Hạng của một ma trận A được xác định bởi số chiều của Im(A) tức là :

$$rank(A) = \dim(\text{ImA})$$

Chú ý

1. Hạng của một ma trận A được có thể được định nghĩa bởi số lớn nhất các cột (hay hàng của A) độc lập tuyến tính .
2. Nếu $A \in R^{m\text{x}n}$, thì dim(KerA) + $rank(A)) = n$

1.2. Trị riêng và vecto riêng

Cho một ma trận vuông A ∈ R^{nxn}, nếu số vô hướng λ và vector x ≠ 0 ∈ R^n thoả mãn:

$$Ax = \lambda x$$

Thì λ được gọi là một trị riêng của A và x được gọi là vector riêng tương ứng với trị riêng đó.

Một vài tính chất:

1. Nếu x là một vector riêng của A ứng λ với thì kx, k≠0 cũng là vector riêng ứng với trị riêng đó.

2. Mọi ma trận vuông bậc n đều có n trị riêng (kể cả lặp) và có thể là các số phức.
3. Với ma trận đối xứng xác định riêng, tất cả các trị riêng đều là các số thực.
4. Với *ma trận xác định dương*, tất cả các trị riêng của nó đều là các số thực dương. Với *ma trận đối xứng nửa xác định dương*, tất cả các trị riêng của nó đều là các số thực không âm.

1.3. Hệ trực giao và trực chuẩn

Một hệ cơ sở $u_1, u_2,\ldots,u_m \in R^m$ được gọi là trực giao nếu mỗi vector đều khác 0 và tích 2 vector bất kỳ bằng 0.

$$u_i \neq 0;\ \ u_i u_j = 0\ \forall\ 1 \leq i \neq j \leq m$$

Một hệ cơ sở $u_1, u_2,\ldots,u_m \in R^m$ được gọi là trực chuẩn nếu nó là một hệ trực giao và độ dài mỗi vector bằng 1.

Gọi $U = [u_1, u_2,\ldots,u_m]$ với $u_1, u_2,\ldots,u_m \in R^m$ là trực chuẩn, ta có:

$$UU^T = U^TU = I$$

Trong đó I là ma trận đơn vị bậc m. Ta gọi U là *ma trận trực giao*.

Một vài tính chất:

1. $U^{-1} = U^T$: nghịch đảo của một ma trận trực giao chính là chuyển vị của nó.
2. Nếu U là ma trận trực giao thì chuyển vị của nó U^T cũng là một ma trận trực giao.

3. Định thức của ma trận trực giao bằng 1 hoặc -1.

4. Dưới dạng biến đổi tuyến tính, một ma trận trực giao bảo toàn các yếu tố hình học: tích vô hướng, độ dài, góc,… Ma trận trực giao thể hiện cho phép xoay một vector.

5. Giả sử $U^* \in R^{mxr}$, r < m là một ma trận trực giao của U được tạo bởi r cột của U ta sẽ có $U^{*T}U^* = I_r$

1.4. Chuẩn Vector

Định nghĩa 1.4. Chuẩn vectơ trên R^n là một hàm $f: R^n \to R$ thỏa mãn các tính chất sau:

$$(i) f(x) \geq 0, \forall x \in R^n$$

$$f(x) = 0 <=> x = 0$$

$$(ii) f(x + y) \leq f(x) + f(y), \forall x, y \in R^n$$

$$(iii) f(\alpha x) = |\alpha|\, f(x), \forall \alpha \in R, \forall x \in R^n$$

Kí hiệu: $f(x) = \|x\|$.

Cho Vector $x = (x_1, x_2, \dots, x_n)^T$, các chuẩn vector thông dụng là :

Chuẩn 2 (p = 2)

$$\|x\|_2 == (|x_1|^2 + \cdots + |x_n|^2)^{\frac{1}{2}} = (x^T x)^{\frac{1}{2}}$$

Bổ đề: Nếu $Q \in R^n$ là ma trận trực giao và $x \in R^n$ thì

$$\| Qx \|_2^2 = \| x \|_2^2$$

Chứng minh. Ta có theo định nghĩa chuẩn 2 của vector, ta có

$$\| Qx \|_2^2 = \langle Qx, Qx \rangle = (Qx)^T (Qx)$$

$$= x^T Q^T Q x \ (do\ tính\ chất\ của\ ma\ trận\ chuyển\ vị)$$

$$= x^T x \ (do\ Q\ là\ ma\ trận\ trực\ giao\ nên\ Q^T Q = I)$$

$$= \| x \|_2^2$$

1.5. Chuẩn ma trận

Định nghĩa 1.5. Chuẩn ma trận trên R^{mxn} là hàm số f: $R^{mxn} \longrightarrow R$ thỏa mãn các tính chất sau:

$$(i) f(A) \geq 0, \forall A \in R^{mxn}$$

$$f(A) = 0 <=> A = 0$$

$$(ii) f(A + B) \leq f(A) + f(B), \forall A, B \in R^{mxn}$$

$$(iii) f(\alpha A) = |\alpha|\ f(A), \forall \alpha \in R, \forall A \in R^{mxn}$$

$Kí\ hiệu: f(A) = \|A\|$

Cho ma trận $(A) = (a_{ij})_{mx} = \begin{pmatrix} a_{11} & a_{12} & \dots & a_{1n} \\ a_{11} & a_{22} & \dots & a_{2n} \\ \vdots & \vdots & \ddots & \vdots \\ a_{m1} & a_{m2} & \dots & a_{mn} \end{pmatrix}$,các chuẩn ma trận thông dụng là:

Chuẩn toán tử

Nếu ma trận A thuộc không gian vector R^{mxn} thì chuẩn của A ứng với chuẩn p của vector là:

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

Chuẩn 2

$$||A||_2 = sup_{x\neq 0}\frac{||A||_2}{||x||_2}, \forall x \in R^n$$

Chuẩn F (Frobeneous)

$$\|A\|_F = \left(\sum_{i=1}^{m} \sum_{j=1}^{n} a_{\mathrm{ij}}^2 \right)^{\frac{1}{2}}$$

Định lý 1.1. Nếu $A \in R^{mxn}$, thì tồn tại một vector đơn vị Z (theo chuẩn 2) trong R^n sao cho $A^T A\mathrm{Z} = \mu^2\mathrm{Z}$, ở đây $\mu = \|A\|_2$

Chứng minh :

Xét hàm $g(x)= \frac{1}{2}\frac{||A||_2^2}{||x||_2^2} = \frac{1}{2}\frac{x^T A^x Ax}{x^T x}$

Giả sử $\mathrm{Z} \in R^n$ là một vector đơn vị sao cho $\|A\mathrm{Z}\|_2 = \|A\|_2$. Vì z là điểm mà hàm $g(x)$ đạt giá trị cực đại, suy ra$\nabla g(\mathrm{Z}) = 0$, ở đây ∇g là gradient của g. Một phép lấy vi phân chỉ ra rằng với mọi $i = 1; n$ ta có :

$$\frac{\partial g(z)}{\partial z_i} = \frac{\left[(Z^T Z) \sum_{j=1}^{n} (A^T A)_{ij} z_j - (Z^T A^T) z_j\right]}{(Z^T)^2}$$

Với ký hiệu $A^T A\mathrm{Z} = ((z^T A^T A\mathrm{Z})\mathrm{Z}$. Định lý được suy ra bởi việc đặt $\mu = \|A\mathrm{Z}\|_2$

Định lý suy ra rằng:

$$\|A\|_2 = \sqrt{\lambda_{max}(A^T A)}.$$

1.6. Tính bất biến của chuẩn hai

Chuẩn hai của ma trận bất biến với phép nhân trái và phải với các ma trận trực giao. Tính chất này được thể hiện ở định lý sau đây:

Định lý 1.2. Cho $A \in R^{mxn}$, nếu hai ma trận $Q \in R^{mxm}$ và $Z \in R^{nxn}$ là các ma trận trực giao thì:

$$\|QAZ\|_2 = \|A\|_2$$

Chứng minh.

Theo định nghĩa của chuẩn 2 ta có:

$$||A||_2^2 = \left(sup_{x\neq 0}\frac{||QAx||_2}{||x||_2}\right)^2$$

Do Q là ma trận trực giao nên ta có:

$$\left(sup_{x\neq 0}\frac{||QAx||_2}{||x||_2}\right)^2 = \left(sup_{x\neq 0}\frac{||Ax||_2}{||x||_2}\right)^2 = ||A||_2^2$$

CHƯƠNG 2: KHAI TRIỂN KỲ DỊ CỦA MA TRẬN (SVD)

2.1. Giới thiệu

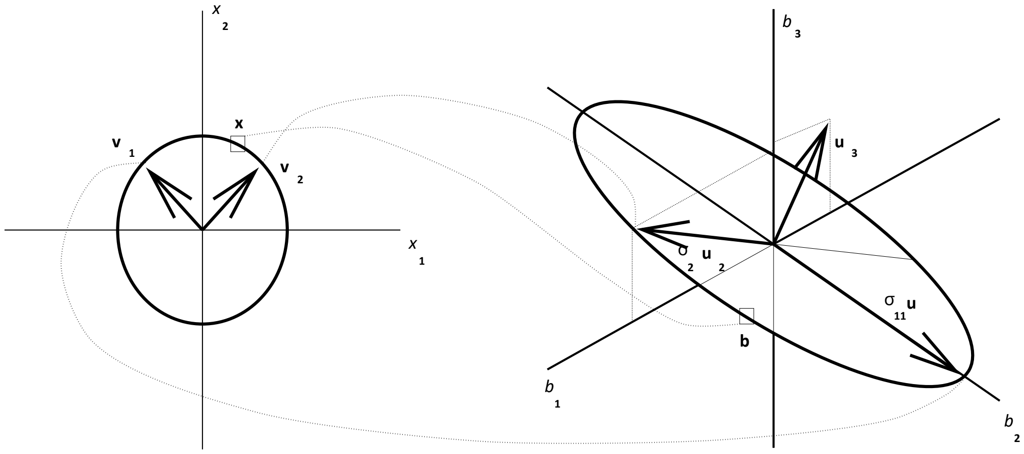
2.1.1.Giới thiệu chung

Phương pháp *phân tích suy biến* (singular value decomposition) được viết tắt là SVD là một trong những phương pháp thuộc nhóm matrix factorization được phát triển lần đầu bởi những nhà hình học vi phân. Ban đầu mục đích của phương pháp này là tìm ra một phép xoay không gian sao cho tích vô hướng của các vector không thay đổi. Từ mối liên hệ này khái niệm về ma trận trực giao đã hình thành để tạo ra các phép xoay đặc biệt. Phương pháp SVD đã được phát triển dựa trên những tính chất của ma trận trực giao và ma trận đường chéo để tìm ra một ma trận xấp xỉ với ma trận gốc. Phương pháp này sau đó đã được ứng dụng rộng rãi trong các lĩnh vực như hình học vi phân, hồi qui tuyến tính, xử lý hình ảnh, clustering, các thuật toán nèn và giảm chiều dữ liệu, và đặc biệt đặc biệt hiệu quả trong các bài toán recommendation mà ta sẽ trình bày ở cuối bài viết.

2.1.2.Mục tiêu

Phương pháp SVD sẽ tìm ra một lớp các ma trận xấp xỉ tốt nhất với một ma trận cho trước dựa trên khoảng cách norm Frobenios giữa 2 ma trận. Người ta đã chứng minh được rằng ma trận xấp xỉ tốt nhất được biểu diễn dưới dạng tích của 3 ma trận rất đặc biệt bao gồm 2 *ma trận trực giao* (orthogonal matrix) và 1 *ma trận đường chéo* (diagonal matrix). Quá trình nhân ma trận thực chất là quá trình biến đổi các điểm dữ liệu của ma trận gốc thông qua những phép xoay trục (rotation) và phép thay đổi độ lớn (scaling) và từ đó tạo ra những điểm dữ liệu mới trong không gian mới. Điều đặc biệt của ma trận đường chéo đó là các phần tử của nó chính là những giá trị riêng của ma trận gốc. Những điểm dữ liệu trong không gian mới có thể giữ được 100% thông tin ban đầu hoặc chỉ giữ một phần lớn thông tin của dữ liệu ban đầu thông qua các phép truncate SVD. Bằng cách sắp xếp các trị riêng theo thứ tự giảm dần trên đường chéo chính thuật toán SVD có thể thu được ma trận xấp xỉ tốt nhất mà vẫn đảm bảo giảm được hạng của ma trận sau biến đổi và kích thước các ma trận nhân tử nằm trong giới hạn cho phép. Do đó nó tiết kiệm được thời gian và chi phí tính toán và đồng thời cũng tìm ra được một giá trị dự báo cho ma trận gốc với mức độ chính xác cao.

2.2. Định nghĩa về về giá trị kỳ dị, các vector kỳ dị trái và phải



$$S = \{x \in R^n \mid \|x\| \le 1\} \to A \in R^{m\times n} \to AS = \{Ax \mid x \in S\}$$

- Có 1 ánh xạ tuyến tính từ không gian $R^{m\times n}$ vào $R^{m\times n}$ ta được ma trận cỡ mxn. A có thể coi như 1 phép biến hình cầu từ trong không n chiều thành 1 hình khác trong không gian m chiều.
- Vector $(v_1, v_2) \to (u_1, u_2, u_3)$. Trong đó các $\vec{u}$ là các vector đơn vị và các $\vec{\sigma_i}$ chỉ về độ lớn
- Kết hợp của phép co dãn và phép quay các bán trục cũng bị co dãn, trục dài ra, trục co lại. Từ hình tròn trở thành 1 hình elip

Giả sử $A \in R^{m\times n}, m > n, rankA = n$. Tức A là 1 ma trận gầy, số hàng lớn hơn số cột

- Xét trên hình, Giá trị kỳ dị của ma trận A là các giá trị σ_i đang làm co dãn các bán trục hay độ dài của các vector bán trục chính các giá trị kỳ dị.
- Các $\vec{u}$ sinh ra các trục của elip được gọi là các vector kỳ dị trái.
- Các $\vec{v_i} \to \vec{u_i}\vec{\sigma_i}$ thì $\vec{v_i}$ gọi là vector kỳ dị phải ứng với giá trị kỳ dị $\vec{\sigma_i}$ hay $\vec{v_i}$ là các vector trực chuẩn thoả mãn $A\vec{v_i} = \vec{u_i}\vec{\sigma_i}$

2.3. Định nghĩa khai triển kỳ dị của ma trận chữ nhật đứng hạng đủ

2.3.1.Định nghĩa khai triển kỳ dị của ma trận hạng đủ

Đặt
$$U = [u_1, u_2, ..., u_n]$$
$$\sum = diag(\sigma_1, \sigma_2, ..., \sigma_n)$$
$$V = [v_1, v_2, ..., v_n]$$

Với $\sum$ là ma trận đường chéo, gồm các phần tử trên đường chéo là các giá trị kỳ dị.

Khi đó :

$$Av_i = \sigma_i u_i, \forall i = \overline{1,n} \Leftrightarrow AV = U\sum \Leftrightarrow A = U\sum V^T$$
$$\Leftrightarrow A = \sigma_1 u_1 v_1^T + \sigma_2 u_2 v_2^T + ... + \sigma_n u_n v_n^T$$

Cụ thể:

$$A[v_1 \quad v_2 \quad ... \quad v_n] = [Av_1 \quad Av_2 \quad ... \quad Av_v]$$
$$= [\sigma_1 u_1 \, \sigma_2 u_2 \quad ... \quad \sigma_n u_n] = [u_1 \quad u_2 \quad ... \quad u_n]\begin{bmatrix} \sigma_1 & & & \\ & \sigma_2 & & \\ & & \ddots & \\ & & & \sigma_n \end{bmatrix}$$

Khi có 1 ma trận đường chéo, ta nhân vào bên trái, ta được các hàng tương ứng nhân với các phần trử trên đường chéo tương ứng.

$A = U\sum V^T$ là một khai triển kì dị

Dễ thấy:

- A: $R^{m\times n} \to R^{m\times n}$, số chiều m < số chiều n
- Hệ vector $v_1, v_2, ..., v_n$ trực chuẩn cũng là ma trận cơ sở của $R^{m\times n}$
- $V = [v_1, v_2, ..., v_n]$ là một ma trận trực giao
- U cũng là một ma trận không vuông các cột cũng tạo thành một ma trận trực chuẩn cỡ mxn

⇨ Tích trên sẽ là một ma trận A cỡ mxn. Đây là cách viết thứ nhất

Cách viết thứ 2:

$$A = \begin{bmatrix} \sigma_1 u_1 \, \sigma_2 u_2 & \dots & \sigma_n u_n \end{bmatrix} \begin{bmatrix} v_1^T \\ v_2^T \\ \vdots \\ v_n^T \end{bmatrix}$$

2.3.2.Trường hợp hạng không đủ

- Trường hợp: $A \in R^{m \times n}, m > n, rank = r < n$
- Các giá trị kỳ dị: $\sigma_1 \geq \sigma_2 \geq \dots \geq \sigma_r \geq 0$
- Các vector kỳ dị trái: $u_1, u_2, \dots, u_r$
- Các vector kỳ dị phải: $v_1, v_2, \dots, v_r$

Khai triển kỳ dị của ma trận:

$$U = [u_1, u_2, \dots, u_r] \in R^{m \times n},$$

$$\sum = diag(\sigma_1, \sigma_2, \dots, \sigma_r) \in R^{m \times n},$$

$$V = [v_1, v_2, \dots, v_r]$$

$$A = U \sum V^T = \sigma_1 u_1 v_1^T + \sigma_2 u_2 v_2^T + \dots + \sigma_n u_n v_n^T$$

Trong trường hợp vẫn là ma trận gầy, hạng không đủ. Không gian ảnh chỉ có n chiều, lúc này cái elip sẽ có n trục và có n giá trị kì dị, tương tự cũng sẽ có n giá trị kỳ dị trái phải.

Bản thân các vector vi là các vector trực chuẩn với nhau.

$$v_j^T v_j = \begin{cases} 1 & i = j \\ 0 & i \neq j \end{cases}$$

$$AV = U \sum (V^T V) = (U \sum V^T) V$$

Bổ sung $\{v_1, \dots, v_r\} \overrightarrow{bs} \{v_1, \dots, v_r, v_{r+1}, \dots, v_n\}$

$$Av_i = \begin{cases} \sigma_i u_i & \\ 0 & i > r \end{cases}$$

2.4. Xác định giá trị kỳ dị và các vector kỳ dị

2.4.1. Phương pháp

Giả sử tồn tại: $A^T A = \left(U\Sigma V^T\right)^T\left(U\Sigma V^T\right) = V\Sigma^2 V^T$ (*)

Phân tích:

- ❖ U là ma trận cơ sở có các cột là vector trong 1 hệ trực chuẩn
  - ⇨ $U^T U$ là một ma trận đơn vị
- ❖ Đồng thời $V\Sigma^2 V^T$ là dạng chéo hoá trực giao ma trận đối xứng, Σ đã mở rộng để có thể thực hiện phép nhân bằng cách bổ sung các phần băng 0
  - ⇨ Đây là một con đường để tìm giá trị kỳ dị và vector kỳ dị

Từ (*):

- Thấy được v_i là các vector riêng ưng với các giá trị riêng khác 0 của $A^T A$
- σ_i^2 là các giá trị riêng của $A^T A$
- $\{v_i\}_{i=\overline{1,r}}$ là cơ sở trực chuẩn của không gian vector ker $A^\perp$
- $\|A\| = \sigma_1$

2.4.2. Thực hiện

❖ Các bước thực hiện

Bước 1: Tính A^T, $A^T A$ và AA^T

Bước 2: Xác định trị riêng của ma trận $A^T A$ =>Các giá trị kì dị ma trận A
Với mỗi giá trị riêng λ_i của ma trận ta tìm được giá trị kì dị của ma trận

$$\sigma_i = \sqrt{\lambda_i} \text{ với i = min(m,n)}$$

Bước 3: Xây dựng ma trận Σ , tùy thuộc vào vào kích thức của A mà Σ sẽ có 2 dạng khác nhau

Bước 4: Tính giá trị ma trận V => V^T
Ứng với mỗi trị riêng của $A^T A$ ta tìm được một vector riêng v_i, từ đó ta được:

$$V = [v_1 v_2 \ldots v_n] \Rightarrow V^T$$

Bước 5: Tính giá trị ma trận U.

- Cách 1:

  Ứng với mỗi trị riêng của AA^T ta tìm được một vector riêng u_i, từ đó ta được:

  $U = [u_1 u_2 \dots u_m]$

- Cách 2:

  Với các σ_i (i : 1 → r) là các giá trị kì dị của ma trận A.

  Đặt $u_i = \frac{1}{\sigma_i} A v_i$ (i = 1, …, r).

  Từ đó ta xây dựng được ma trận:

  - $U = [\, u_1 \dots u_m \,]$ (m > r).

Bước 6: Từ đó ta có :

$$A = U.\Sigma.V^T$$

❖ Ví dụ:

$A = \begin{bmatrix} 4 & 4 \\ -3 & 3 \end{bmatrix}$, $A^T = \begin{bmatrix} 4 & -3 \\ 4 & 3 \end{bmatrix}$

$A^T A = \begin{bmatrix} 4 & -3 \\ 4 & 3 \end{bmatrix} \begin{bmatrix} 4 & 4 \\ -3 & 3 \end{bmatrix} = \begin{bmatrix} 25 & 7 \\ 7 & 25 \end{bmatrix}$

$A A^T = \begin{bmatrix} 4 & 4 \\ -3 & 3 \end{bmatrix} \begin{bmatrix} 4 & -3 \\ 4 & 3 \end{bmatrix} = \begin{bmatrix} 32 & 0 \\ 0 & 18 \end{bmatrix}$

⇨ Giá trị riêng $\lambda_1 = 32$, $\lambda_2 = 18$

⇨ Giá trị kỳ dị $\sigma_1 = \sqrt{32}$, $\sigma_2 = \sqrt{18}$

Vector riêng của ma trận $A^T A$ là $\begin{bmatrix} 1 \\ 1 \end{bmatrix}$ và $\begin{bmatrix} 1 \\ -1 \end{bmatrix}$

⇨ $v_1 = \begin{bmatrix} 1/\sqrt{2} \\ 1/\sqrt{2} \end{bmatrix}$, $v_2 = \begin{bmatrix} 1/\sqrt{2} \\ -1/\sqrt{2} \end{bmatrix}$

⇨ $V^T = \begin{bmatrix} 1/\sqrt{2} & 1/\sqrt{2} \\ 1/\sqrt{2} & -1/\sqrt{2} \end{bmatrix}$

Tương tự, ta tìm được U = $\begin{bmatrix} 1 & 0 \\ 0 & -1 \end{bmatrix}$

⇨ $\Sigma = \begin{bmatrix} 4\sqrt{2} & 0 \\ 0 & 3\sqrt{2} \end{bmatrix}$

Cuối cùng, ta được:

$$A = U\Sigma V^T = \begin{bmatrix} 1 & 0 \\ 0 & -1 \end{bmatrix} \begin{bmatrix} 4\sqrt{2} & 0 \\ 0 & 3\sqrt{2} \end{bmatrix} \begin{bmatrix} 1/\sqrt{2} & 1/\sqrt{2} \\ 1/\sqrt{2} & -1/\sqrt{2} \end{bmatrix}$$

❖ Code:

```
import numpy as np
from numpy import linalg as LA
from scipy.linalg import svd
# Input A
data = np.genfromtxt('input1.txt', delimiter=' ')
A = np.array(data)
m = len(A)
n = len(A[0])
AT = A.T
if n > m:
    res = np.dot(AT, A)
    w, v = LA.eig(res)
        i = 0
    while (i < len(w) - 1):
        j = i + 1
        while (j < len(w)):
            if w[i] < w[j]:
                w[[i, j]] = w[[j, i]]
                v[:, [i, j]] = v[:, [j, i]]
            j = j + 1
        i = i + 1
else:
    res = np.dot(A, AT)
    w, u = LA.eig(res)
    i = 0
```

```
    while (i < len(w) - 1):
        j = i + 1
        while (j < len(w)):
            if w[i] < w[j]:
                w[[i, j]] = w[[j, i]]
                u[:, [i, j]] = u[:, [j, i]]
            j = j + 1
        i = i + 1

# Middle Singular Vectors
s = [] Khai báo mảng ( ma trận ) kỳ dị
for i in range(n):
    temp = [0] * m
    s.append(temp)
for i in range(min(n, m)): w
    s[i][i] = np.sqrt(w[i])
s = np.array(s)
if n > m:
    u = []
    for i in range(m):
        t = []
        for j in range(n):
            t.append(v[j][i])
        t = (1 / s[i][i]) * (A.dot(t))
        u.append(t)
    u = np.array(u)
    u = u.T
else:
    v = []
    for i in range(n):
        t = []
        for j in range(m):
            t.append(u[j][i])
        t = (1 / s[i][i]) * (AT.dot(t))
        v.append(t)
    v = np.array(v)
    v = v.T
```

❖ Input vào Output:

Đầu Vào :

```
4 4
-3 3
```

Đâu Ra :

```
A=
[[ 4.  4.]
 [-3.  3.]]
U =
[[1. 0.]
 [0. 1.]]
S =
[[5.65685425 0.        ]
 [0.         4.24264069]]
V =
[[ 0.70710678 -0.70710678]
 [ 0.70710678  0.70710678]]
```

2.5. Nghịch đảo suy rộng

Giả sử A là ma trận khác 0 với khai triển kỳ dị là $A=\left(U\Sigma V^T\right)$. Khi đó $A^{\dagger}=V\Sigma^{-1}U^T$ được gọi là ma trận nghịch đảo suy rộng của A

Trường hợp m>n, rankA=n

$$A^{\dagger}=\left(A^TA\right)^{-1}A^T$$

$$x=A^{\dagger}y\in X_{ls}=\left\{z\,|\,\|Az-y\|=\min_{\omega}\|A\omega-y\|\right\}$$

2.6. Số điều kiện của ma trận (Pseudo-inverse or Moore-penrose inverse)

Giả sử A là ma trận khả nghịch

y = Ax

$$y+\delta y=A(x+\delta x)$$

$$\delta x = A^{-1}\delta y \rightarrow \|\delta x\| \le \|A^{-1}\|\|\delta y\|$$

$$\|y\| \le \|A\|\|x\| \Rightarrow \frac{1}{\|x\|} \le \frac{\|A\|}{\|y\|} \Rightarrow \frac{\|\delta x\|}{\|x\|} \le \|A\|\|A^{-1}\|\frac{\|\delta y\|}{\|y\|}$$

$$condA = \|A\|\|A^{-1}\| = \frac{\sigma_{\max}}{\sigma_{\min}}$$

Một số tính chất:

i) $cond(A) \ge 1;$

ii) Nếu A là ma trận trực giao (tức là $A' = A^{-1}$) thì $con(A) = 1;$

iii) Với mọi $c \neq 0 \in R$: ta đều có $cond(cA) = cond(A)$;

iv) Nếu $D = diag(d_i)_1^n$ thì $cond(D) = \dfrac{\max|d_i|}{\min|d_i|}$.

CHƯƠNG 3: ỨNG DỤNG

3.1. Phân tích suy biến của một ma trận

Trong python, việc phân tích suy biến của một ma trận được thực hiện dễ dàng thông qua việc sử dụng hàm SVD của package scipy như sau:

```
import scipy.linalg as ln
import numpy as np
m, n = 2, 3
n_diag = min(m, n)
#Tạo một ma trận ngẫu nhiên cỡ mxn
A = np.random.rand(m, n)
U, S_diag, V = ln.svd(A)

#Tạo một ma trận đường chéo S
S = np.zeros((n_diag, n_diag))
np.fill_diagonal(S, S_diag)
if m > n:
    S = np.concatenate((S, np.zeros((1, n))), axis = 0)
elif m < n:
    S = np.concatenate((S, np.zeros((m, 1))), axis = 1)

print('Ma tran A: \n %s \n'%A)
print('Ma tran truc giao U: \n %s \n'%U)
print('Check Frobenius U^TU-I: \n %s \n'%ln.norm(np.dot(U.T,U)-
np.eye(m, m), 'fro'))
print('Ma tran truc giao V: \n %s \n'%V)
print('Check Frobenius V^TV-I: \n %s \n'%ln.norm(np.dot(V.T,V)-
np.eye(n, n), 'fro'))
print('Ma tran duong cheo S: \n %s \n'%S_diag)
print('Ma tran S: \n %s \n'%S)
print('Check Frobenius U.S.V - A: \n %s \n'%ln.norm(np.dot(U, S.dot(V))-
A,'fro'))
```

Kết quả thu được:

```
Ma tran A:
 [[0.55635564 0.09099476 0.68099651]
 [0.30590558 0.7678004  0.52385038]]

Ma tran truc giao U:
 [[-0.6535162  -0.75691253]
 [-0.75691253  0.6535162 ]]

Check Frobenius U^TU-I:
 4.965249222956005e-16

Ma tran truc giao V:
 [[-0.49039095 -0.52787749 -0.69344219]
 [-0.42864846  0.83888409 -0.33546084]
 [-0.75879985 -0.13273597  0.63765504]]

Check Frobenius V^TV-I:
 3.3693213942615752e-16

Ma tran duong cheo S:
 [1.21358517 0.51603663]

Ma tran S:
 [[1.21358517 0.         0.        ]
 [0.         0.51603663 0.        ]]

Check Frobenius U.S.V - A:
 2.391654643720554e-16
```

3.2. Giải phương trình tuyến tính

Kiến thức áp dụng:

Dựa trên những kiến thức về ma trận nghịch đảo Pseudo - Inverse. Ma trận giả nghịch đảo có thể được giải nhờ phép phân tích suy biến của A. Nếu ma trận A có biểu diễn phân tích suy biến dạng:

$$A=U\Sigma V^{T\backslash}$$

Khi đó ma trận giả nghịch đảo của A sẽ là:

$$A\dagger=V\Sigma^{-1} U^{T}$$

Ta thấy ma trận giả nghịch đảo vẫn có tính chất như ma trận nghịch đảo đó là A†A=I. Bên dưới ta sẽ sử dụng SVD để giải phương trình hồi qui tuyến tính.

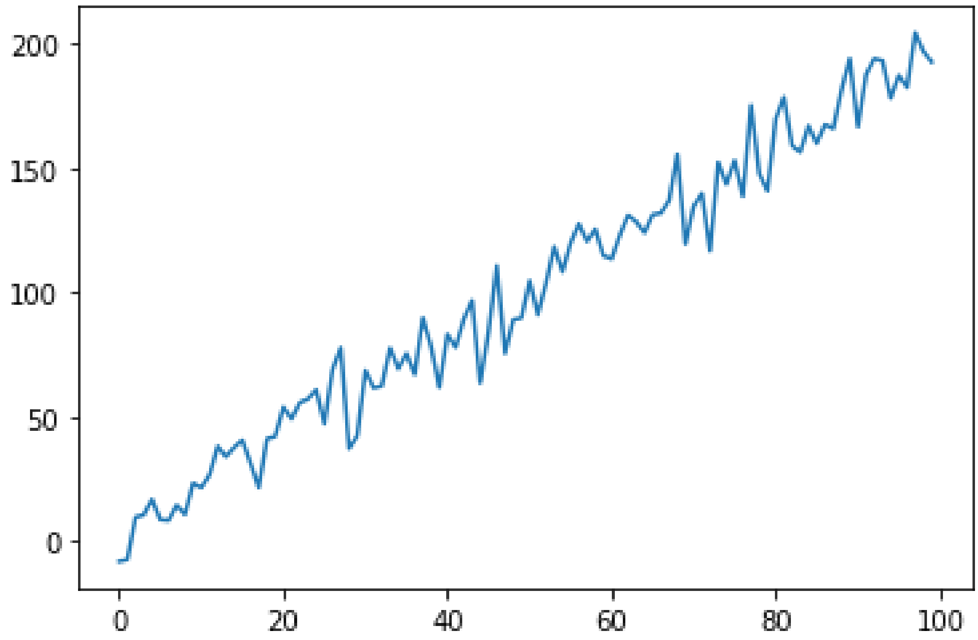
Code:

In[1]:

```
import numpy as np
import matplotlib.pyplot as plt
%matplotlib inline

n = 100
x = np.arange(n)
```

```
X = np.concatenate((x.reshape(n, 1), np.ones((n, 1))), axis = 1)
y = np.random.randn(n)*10 + 2*x
plt.plot(x, y)
```

Out[1]:



In [2]:

```
# Tính ma trận A, b
A = np.dot(X.T, X)
b = np.dot(X.T, y)

# Phân tích SVD
U, S_diag, V = np.linalg.svd(A)
S = np.zeros((S_diag.shape[0], S_diag.shape[0]))
np.fill_diagonal(S, S_diag)
S_inv = np.linalg.inv(S)

# Pseudo - inverse matrix
A_pse =  np.dot(V.T, np.dot(S_inv, U))

# Hệ số ước tính
w_svd = np.dot(A_pse, b)
print('w Giá trị nhận được từ SVD: %s' %str(w_svd))
```

Out[2]:

```
w Giá trị nhận được từ SVD: [1.94927789 3.62551228]
```

In [3]:

```
#@title
from sklearn import linear_model
```

```python
ln_reg = linear_model.LinearRegression(fit_intercept = False)
ln_reg.fit(X, y)
print('w calculated from sklearn: %s' %str(ln_reg.coef_))
```

Out[3]:

```
w calculated from sklearn: [1.94927789 3.62551228]
```

3.3. Nén ảnh

❖ Lý thuyết: Áp dụng phương pháp Truncate SVD

Giả định A là ma trận xác định dương, khi đó mọi trị riêng không ấm

Trên ma trận đường chéo của Σ_{mn} lấy ra t dòng và cột ứng với top t các giá trị riêng $\sigma_1 > \sigma_2 > ... > \sigma_t > 0$ lớn nhất của A và U,V^T

Phần còn lại sẽ bị loại bỏ, như vậy ta sẽ thu được ma trận xấp xỉ của A là:

$$\hat{A} = U_t \Sigma_t V_t^T$$

Hoặc ta có thể biểu diễn:

$$A = \sum_{t=1}^{t} u_i \sigma_i v_i$$

Khi đó ta tính được khoảng cách norm Frobenius giữa A và $\hat{A}$ chính bằng tổng bình phương của các trị riêng còn lại từ $\sigma_{t+1} -> \sigma_n$:

$$\begin{aligned}
||\mathbf{A} - \hat{\mathbf{A}}||_F^2 &= (\mathbf{A} - \hat{\mathbf{A}})^\mathbf{T}(\mathbf{A} - \hat{\mathbf{A}}) \\
&= (\sum_{i=t+1}^{n} \mathbf{u_i}\sigma_i\mathbf{v_i})^\mathbf{T}(\sum_{i=t+1}^{n} \mathbf{u_i}\sigma_i\mathbf{v_i}) \\
&= (\sum_{i=t+1}^{n} \sigma_i\mathbf{v_i^T}\mathbf{u_i^T})(\sum_{i=t+1}^{n} \sigma_i\mathbf{u_i}\mathbf{v_i}) \\
&= \sum_{i=t+1}^{n}\sum_{j=t+1}^{n} \sigma_i\sigma_j\mathbf{v_i^T}\mathbf{u_i^T}\mathbf{u_j}\mathbf{v_j} \\
&= \sum_{i=t+1}^{n} \sigma_i^2\mathbf{v_i^T}\mathbf{u_i^T}\mathbf{u_i}\mathbf{v_i} \\
&= \sum_{i=t+1}^{n} \sigma_i^2\mathbf{v_i^T}\mathbf{v_i} \\
&= \sum_{i=t+1}^{n} \sigma_i^2
\end{aligned}$$

⇨ Như vậy ta đã lý giải lý do chọn ra top t trị riêng có giá trị lớn nhất. Khi đó sai số của 2 ma trận sẽ là nhỏ nhất vì bằng tổng bình phương của các giá trị riêng còn lại. Phương pháp này còn cho ta biết tỷ lệ phần trăm lượng thông tin lưu giữ trong ma trận xấp xỉ thông qua công thức:

$$\frac{\sum_{i=1}^{t}\sigma_i^2}{\sum_{j=1}^{n}\sigma_j^2}$$

Từ đây ta tìm ra ma trận xấp xỉ tốt nhất với ma trận gốc nhưng có hạng xác định trước nó và nhỏ hơn hạng của ma trận gốc. Mức độ xấp xỉ đo lường thông qua chuẩn F của hiệu 2 ma trận:

$$\arg\min_{rank(X)=k} \|\mathbf{A}-\mathbf{X}\|_{\mathbf{F}}^{2}$$

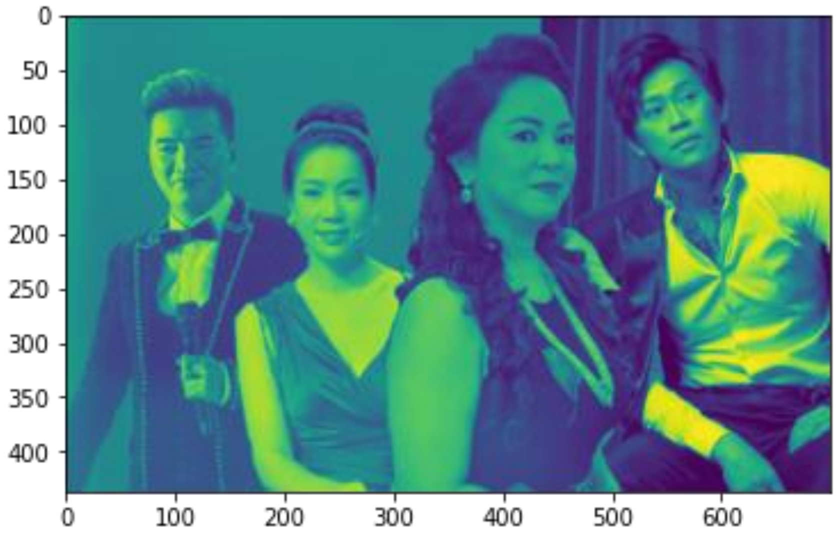
❖ Code:

In[1]:

```
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://cdn.baogiaothong.vn/upload/images/2021-
2/article_img/2021-05-17/img-bgt-2021-phuong-hang-1-1621248851-
width800height500.jpeg')
with urllib.request.urlopen(url) as url:
    f = BytesIO(url.read())

ig = np.array(Image.open(f))
print('Hinh anh: %s'%str(ig.shape))
# Chuyển sang màu xám
ig = ig.dot([0.299, 0.5870, 0.114])
plt.imshow(ig)
```

Out[1]:

```
Hinh anh: (437, 700, 3)
<matplotlib.image.AxesImage at 0x7f2323ad0d90>
```



In[2]:

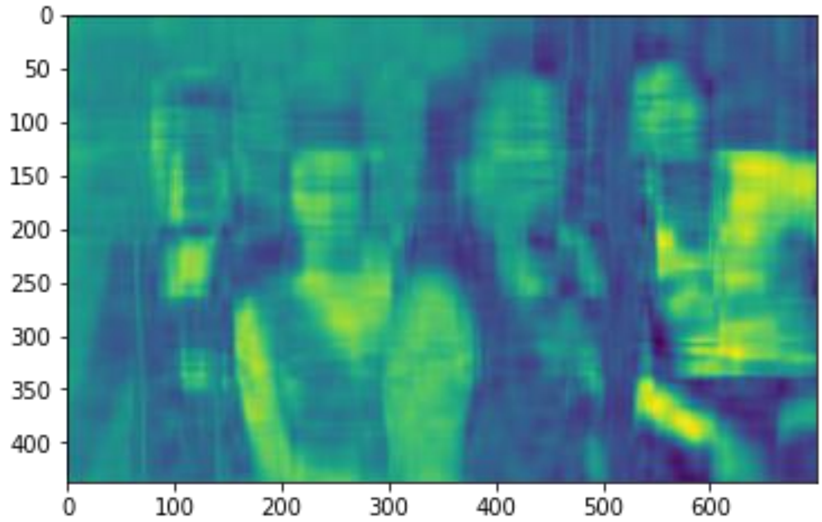
```
def SVD_pic(n_evl):
    """
    n_evl: number of highest eigenvalues taken
    """
    #Xác định kích thước
    m = ig.shape[0]
    n = ig.shape[1]
    #Singular Value Decomposition
    U, S, V = ln.svd(ig)
    #Xác định vị trí id của n giá trị riêng cao nhất
    id_trunc = np.argsort(S)[::-1][:n_evl]
    #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('Tỉ lệ thông tin ảnh được giữ lại : %.2f%s \n'%(prt_retain, '%')
)
```

In[3]:

```
SVD_pic(10)
```

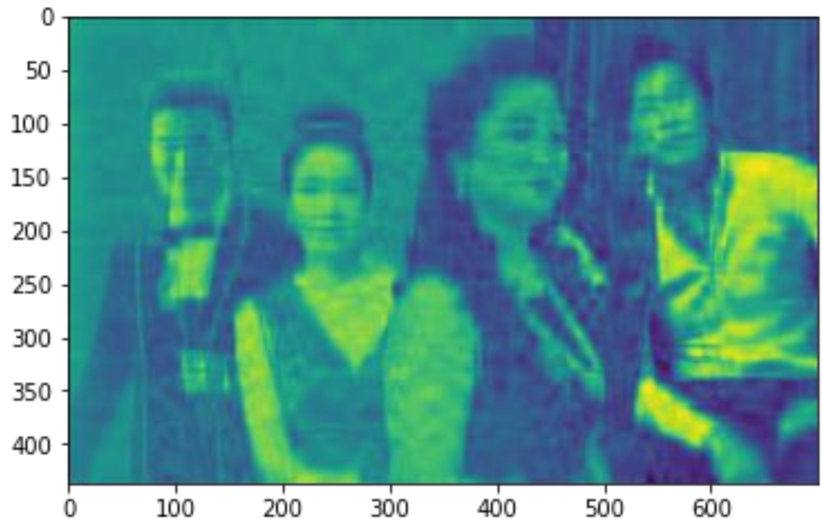
Out[3]:

```
Tỉ lệ thông tin ảnh được giữ lại : 96.74%
```



In[4]:
```
SVD_pic(20)
```

Out[4]:
```
Tỉ lệ thông tin ảnh được giữ lại : 98.55%
```



Ngoài ra SVD còn có rất nhiều ứng dụng hữu ích khác như phân tích quang phổ, nén hình ảnh bằng âm thanh, tất cả có trong file code nhóm em đã thực hiện có đi kèm trong Tệp báo cáo

Danh mục tài liệu tham khảo

[1] Dương Quốc Việt ,Nguyễn Cảnh Lương, *Đại Số* ,NXB Khoa học và Kỹ thuật

[2] Lê Trọng Vinh,*Giáo Trình Giải Tích Số*,NXB Khoa học và Kỹ thuật

[3] Website: Academia.vn/PhysicsMathemathics

[4] Website: MathWorks.com