ĐẠI HỌC QUỐC GIA THÀNH PHỐ HỒ CHÍ MINH
TRƯỜNG ĐẠI HỌC BÁCH KHOA THÀNH PHỐ HỒ CHÍ MINH

ĐẠI SỐ TUYẾN TÍNH

Báo cáo BTL

Phân tích SVD để nén dữ liệu

GVHD: TS Nguyễn Hữu Hiệp

Nhóm: 4 - L10

THÀNH PHỐ HỒ CHÍ MINH, THÁNG 5 2024

Thành viên

STT.	Họ và tên	MSSV	Tỷ lệ hoàn thành (%)
1	Hồ Đắc Minh Phương	2312738	0
2	Hứa Anh Tú	2313786	0
3	Huỳnh Công Đạt	2310639	0
4	Huỳnh Ngọc Hằng	2310902	0
5	Huỳnh Ngọc Thịnh	2313266	0
6	Kiều Chung Tú	2313787	0
7	Lê Chí Đại	2310621	0
8	Lê Ngọc Dương	2310589	0

Mục lục

1 LỜI MỞ ĐẦU

Trong thời đại của Công nghệ Thông tin và Truyền thông, việc xử lý, truyền tải và lưu trữ dữ liệu đã trở nên quan trọng hơn bao giờ hết. Với sự gia tăng không ngừng của lưu lượng dữ liệu, các kỹ thuật nén dữ liệu đóng một vai trò quan trọng trong việc giảm thiểu dung lượng lưu trữ và băng thông truyền dẫn cần thiết. Trong bối cảnh này, Phân rã Giá trị Riêng (Singular Value Decomposition - SVD) đã trở thành một công cụ hữu ích để nén dữ liệu một cách hiệu quả.

SVD là một kỹ thuật đại số tuyến tính phổ biến, cho phép phân tích và biểu diễn một ma trận dưới dạng tích của ba ma trận khác nhau. Khi áp dụng cho dữ liệu, SVD có khả năng tách biệt các thành phần quan trọng và không quan trọng của dữ liệu, từ đó cho phép loại bỏ các thành phần không quan trọng để đạt được nén dữ liệu. Điều này đặc biệt hữu ích trong các ứng dụng xử lý ảnh, xử lý tín hiệu và khám phá dữ liệu, nơi mà dữ liệu thường chứa một lượng lớn thông tin dư thừa hoặc nhiễu.

Trong bài báo này, chúng tôi sẽ khám phá cách sử dụng SVD để nén dữ liệu một cách hiệu quả. Chúng tôi sẽ trình bày các nguyên lý cơ bản của SVD và cách áp dụng nó cho các tập dữ liệu khác nhau. Ngoài ra, chúng tôi cũng sẽ đánh giá hiệu suất của kỹ thuật này so với các phương pháp nén dữ liệu khác và thảo luận về các ứng dụng tiềm năng của nó trong các lĩnh vực như xử lý ảnh, truyền thông đa phương tiện và khám phá dữ liệu.

2 CƠ SỞ LÝ THUYẾT, PHÂN TÍCH SVD

2.1 Cơ sở lý thuyết

2.1.1 Trị riêng và vecto riêng

Với một ma trận vuông A cấp n, giá trị vô hướng λ và vectors $xnx1 \neq 0$ gọi là trị riêng và vector riêng của ma trận A nếu thỏa mãn đẳng thức sau:

$$\boxed{Ax = \lambda x}$$

Từ đẳng thức trên ta suy ra được: $(A - \lambda I) \cdot x = 0$, ta thấy rằng nếu vector $\mathbf{x} = 0$ thì cũng thỏa mãn đẳng thức, thế nhưng việc chọn $\mathbf{x} = 0$ không mang lại ý nghĩa gì trong trường hợp này. Đẳng thức trên có dạng một hệ phương trình đặc trưng, để hệ có nghiệm khác 0 thì $det(A - \lambda I) = 0$. Từ đây ta có thể giải tìm ra trị riêng và vector riêng của ma trận A với $p(\lambda) = det(A - \lambda I)$ được gọi là đa thức đặc trưng của A.

$$\boxed{p(\lambda) = det(A - \lambda I)}$$

Bội đại số của λm là số bội của nghiệm trong đa thức đặc trưng $det(A - \lambda I) = 0$, KH: BĐS(λ_m).

Không gian con riêng của trị riêng λ_m là không gian nghiệm của $det(A - \lambda_m)x = 0$, KH: E_{λ_m} . Bội hình học của λ_m là số chiều của E_{λ_m}: BHH = $dim(E_{\lambda_m})$. **Tính chất của trị riêng và vector riêng:**

- Với các giá trị λo khác nhau thì ta có các vector riêng x ứng trị riêng đó.
- Tổng cộng ma trận vuông cấp n A có n trị riêng nhưng trong số đó có thể là số phức (ngay cả khi ma trận A là ma trận số thực) và trong số đó có các giá trị lập lại.
- Ma trận đối xứng thì tất cả các trị riêng đều là số thực.
- Tổng tất cả các trị riêng của A bằng với vết (Trace) của ma trận A, tức là bằng với tổng các phần tử trên đường chéo của A.
- Tích tất cả các trị riêng của A bằng với det (A)
- Tổng tất cả BĐS của các trị riêng bằng với cấp của A.

- Tổng tất cả các BHH của các trị riêng bằng với số vecto độc lập tuyến tính cực đại.
- Nếu λ_0 là trị riêng của A, thì λ_0^m là trị riêng của ma trận A^m, với $m \in \mathbb{N}$.
- Ma trận vuông A khả nghịch khi và chỉ khi không có trị riêng bằng 0. Nếu λ_0 là trị riêng của A, thì λ_0^-1 là trị riêng của ma trận A^-1.

2.1.2 Trực chuẩn và trực giao trong không gian Euclide

1. Trực giao: Trong không gian tích vô hướng, hai vecto $x, y \in \mathbb{V}$ được gọi là trực giao, $x \perp y$ nếu $(x, y) = 0$ Hệ vecto $x_1, x_2, \ldots, x_m \in \mathbb{V}$ được gọi là một hệ trực giao nếu chúng đôi một vuông góc với nhau, nghĩa là: $x_i \perp x_j; \forall i \neq jhay(x_i, x_j) = 0$ Một cơ sở mà là hệ trực giao được gọi là *cơ sở trực giao.*

2. Trực chuẩn: Hệ vecto $x_1, x_2, \ldots, x_m \in \mathbb{V}$ được gọi là một hệ trực giao và độ dài của mỗi vecto đều bằng 1. Tức là:

$$x_i \perp x_j; \forall i \neq j 1, 2, 3, ..., m$$
$$||x_i|| = 1; x_i = (1, m)$$

Gọi U= [u1, u2, ..., um] với $u_1, u_2, \ldots, u_m \in \mathbb{R}^m$ 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 (orthogonal matrix). Ma trận loại này không được gọi là ma trận trực chuẩn, không có định nghĩa cho ma trận trực chuẩn. Một tập hợp trực chuẩn tạo thành một cơ sở được gọi là *cơ sở trực chuẩn.*

- Trực chuẩn 1 cơ sở trực giao: Cho cơ sở trực giao $v_1, v_2, v_3, \ldots$ của không gian Euclide $\mathbb{R}^3$. Đặt

$$w_i = \frac{1}{\|v_i\|}, \quad (i = \overline{1, m})$$

Khi đó, hệ $W = \{w_1, w_2, w_3, \ldots\}$ là một cơ sở chuẩn trực của $\mathbb{R}^m$.

Họ chuẩn trực: họ vecto M gọi là trực chuẩn nếu M trực giao và $x \in \mathbb{M} : x = 1$

- Thuật toán Gram-Schmidt dùng để trực giao một họ vector:

  Cho $E = \{e_1, e_2, \ldots, e_m\}$ là họ độc lập tuyến tính của không gian vectơ V. Khi đó tồn tại một họ trực giao.

  $F = \{f_1, f_2, \ldots, f_m\}$ thỏa $\langle e_1, e_2, \ldots, e_m\rangle = \langle f_1, f_2, \ldots, f_m\rangle$.

  Thuật toán Gram-Schmidt để trực giao một họ vecto:

  - $f_1 = e_1$
  - $f_2 = e_2 - \frac{(e_3,f_2)}{(f_1,f_1)} f_1$
  - $f_3 = e_3 - \frac{(e_3,f_1)}{(f_1,f_1)} f_1 - \frac{(e_3,f_2)}{(f_2,f_2)} f_2$
  - $f_k = e_k - \frac{(e_k,f_1)}{(f_1,f_1)} f_1 - \frac{(e_k,f_2)}{(f_2,f_2)} f_2 - \ldots - \frac{(e_k,f_{k-1})}{(f_{k-1},f_{k-1})} f_{k-1}$

  Ta được $\{f_1, f_2, \ldots, f_m\}$ là cơ sở trực giao.

  Chuẩn hóa cơ sở trực giao suy ra *cơ sở trực chuẩn.*

  Một vài tính chất

  - $U^{-1} = U^T$: Nghịch đảo của một ma trận trực giao chính là chuyển vị của nó.
  - Nếu U là ma trận trực giao thì chuyển vị của nó U^T cũng là một ma trận trực giao.
  - Định thức của ma trận trực giao bằng 1 hoặc -1 do $\det(U) = \det(U^T)$ và $\det(U)\det(U^T) = \det(UU^T) = \det(I) = 1$.
  - Ma trận trực giao thể hiện cho phép xoay (rotate) một vector. Giả sử có hai vector $\vec{u}$ và $\vec{v}$ và ma trận trực giao U. Dùng ma trận này để xoay hai vector trên ta được $U\vec{u}$ và $U\vec{v}$. Tích vô hướng của hai vector mới là:

    $$(U\vec{u}) \cdot (U\vec{v}) = (U\vec{u})^T(U\vec{v}) \quad = (\vec{u}^T U^T)(U\vec{v}) = \vec{u}^T(U^T U)\vec{v} \quad = \vec{u}^T I \vec{v} = \vec{u}^T\vec{v} \quad = \vec{u} \cdot \vec{v}$$

    như vậy phép xoay không làm thay đổi tích vô hướng giữa hai vector.
  - Giả sử R là một ma trận con của ma trận trực giao U được tạo bởi r cột của U, ta sẽ có $U^T U = I$.

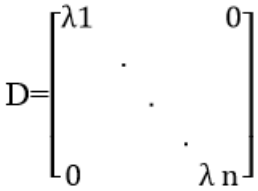
2.1.3 Chéo khóa ma trận

Cho một ma trận vuông cấp n A, ma trận A được gọi là ma trận chéo hóa được nếu tồn tại một ma trận P khả nghịch sao cho:

$$\boxed{P^{-1}AP = D}$$

Trong đó:

- D là ma trận mà các thành phần trên đường chéo chính d_{ij} là các trị riêng của ma trận, các thành phần còn lại bằng 0.



- P là ma trận khả nghịch được cho bởi

$$P = \Big[X_1, X_2, X_3, \ldots, X_n\Big]$$

Với X_k là các vector riêng của ma trận A:

> Không phải ma trận nào cũng chéo hóa được. Ma trận được gọi là chéo hóa được khi có BĐS bằng BHH hay tồn tại n vector độc lập tuyến tính.

- Ma trận chéo hóa trực giao:

Cho một ma trận vuông cấp n: A, nếu tồn tại một ma trận trực giao P sao cho: $P^{-1}AP = D$ thì ta gọi ma trận A là ma trận chéo hóa trực giao được. P được gọi là ma trận chéo hóa trực giao A. Ta thấy rằng P là một ma trận trực giao nên ta có thể suy ra tính chất $P^T = P^{-1}$ từ đó ta có công thức chéo hóa ma trận trực giao: $P^TAP = D$

$$\boxed{A = PDP^T}$$

Khi A là một ma trận chéo hóa trực giao được, ta có rằng:

$$A^T = (PDP^T)^T = (P^T)^T D^T P^T = PDP^T = A$$

Vậy nên bất kì một ma trận đối xứng nào cũng có thể chéo hóa trực giao được.

Trong thực tế, đối với các ma trận đối xứng, chúng ta có thể thực hiện chéo hóa trực giao để đơn giản hóa các dạng bậc hai, biến đổi từ tập dữ liệu không gian nhiều chiều sang ít chiều hơn để giảm kích thước tập dữ liệu.

Vậy đối với những ma trận không đối xứng hoặc không vuông, chúng ta có thể thực hiện giảm chiều của các dữ liệu này không?

SVD (Singular Value Decomposition) là một phương pháp phân tích ma trận bất thành 3 ma trận đơn giản, giúp giảm chiều của dữ liệu mà không làm mất đi các đặc trưng của dữ liệu. Phương pháp được hình thành từ công thức $U\Sigma V^T$ trong đó U và V là hai ma trận trực giao, Σ là ma trận đường chéo.[1]

2.2 Phân tích SVD

2.2.1 Giới thiệu về SVD

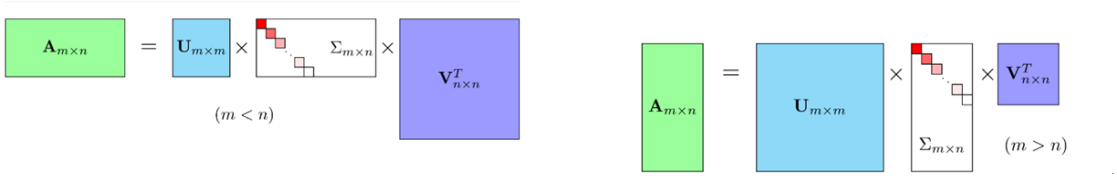
Phương pháp SVD (Singular Value Decomposition) của một ma trận $A_{m\times n}$ tức là phân tích ma trận đó thành tích của ba ma trận:

$A = U\Sigma V^T$

$$\boxed{A_{m\times n} = U_{m\times m}\Sigma_{m\times n}V^T_{n\times n}}\ [2]$$

Trong đó:

- $U_{m\times m}$ là ma trận trực giao, với các cột của $U_{m\times m}$ là các vector riêng của $A_{m\times n}A^T_{m\times n}$.
- $V_{n\times n}$ là ma trận trực giao, với các cột của $V_{n\times n}$ là các vector riêng của $A^T_{m\times n}A_{m\times n}$.
- $\Sigma_{m\times n}$ là ma trận chéo chữ nhật với các vị trí mà số hàng bằng số cột là các giá trị riêng không âm giảm dần của $A_{m\times n}$ $(\sigma_1 \geq \sigma_2 \geq \cdots \geq \sigma_r \geq 0)$ với r là hạng của A.



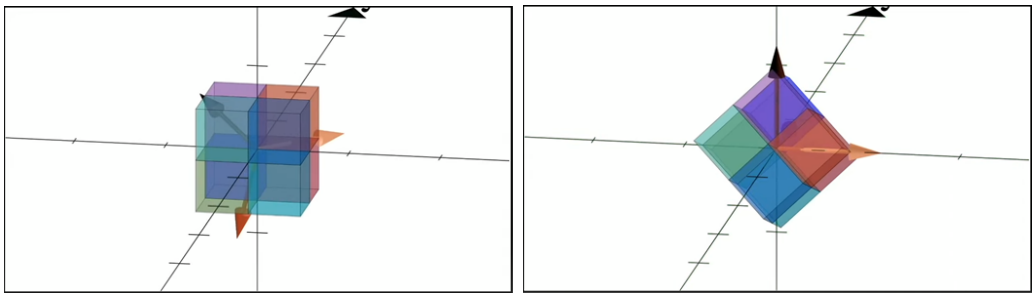
(a) SVD cho ma trận A khi $m < n$. (b) SVD cho ma trận A khi $m > n$

Hình 2.1: Các trường hợp SVD khi m, n không bằng nhau

Mô phỏng SVD: Ta biết rằng ma trận chữ nhật $A_{2\times 3}$ thực hiện một phép biến đổi tuyến tính phức tạp từ $\mathbb{R}_3$ sang $\mathbb{R}_2$. Nhưng khi sử dụng SVD, nó có thể được hiểu là sử dụng tuần tự 3 ma trận đơn giản.

- V^T: thực hiện phép quay sao cho cơ sở vector riêng (hệ cơ sở trực giao) bên phải trở về cơ sở tiêu chuẩn.

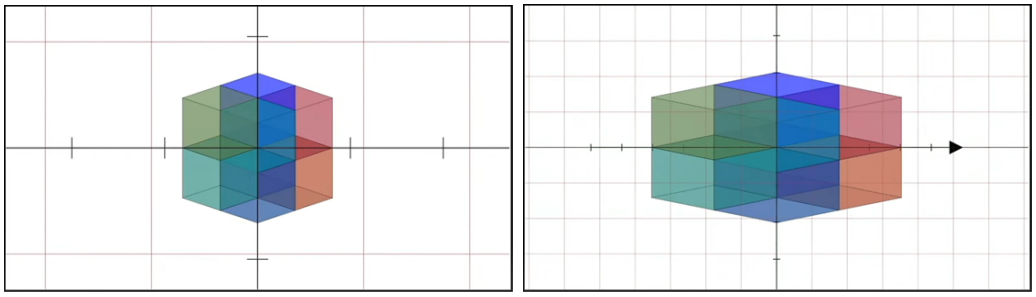
Hình 2.2: Mô phỏng sự biến đổi khi nhận với V^T



(a) Trước khi nhân (b) Sau khi nhân

- Σ: Loại bỏ các chiều, kéo dãn các các chiều còn lại dựa theo các giá trị riêng.

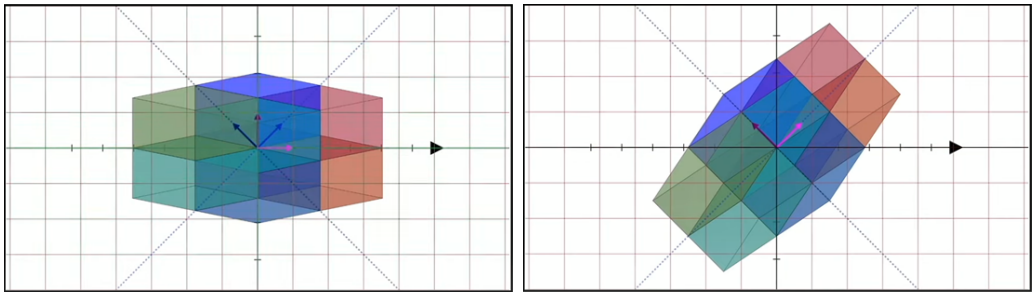
Hình 2.3: Mô phỏng sự biến đổi khi nhân với Σ



(a) Trước khi nhân (b) Sau khi nhân

- U: thực hiện phép quay sao cho cơ sơ tiêu chuẩn trở thành cơ sở vector riêng (hệ cơ sở trực giao) bên trái

Hình 2.4: Mô phỏng sự biến đổi khi nhân với U



(a) Trước khi nhân (b) Sau khi nhân

2.2.2 Thu gọn SVD

Giả sử r(A)=r, ta có thể thu gọn biểu thức tính SVD như sau:

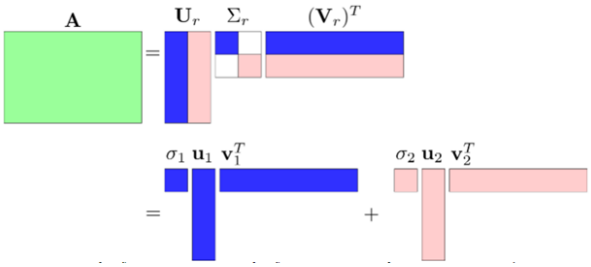
$$A = \sigma_1 u_1 v_1^T + \sigma_2 u_2 v_2^T + \cdots + \sigma_r u_r v_r^T$$

Với $u_i, v_i^T (1 \leq i \leq r)$ là các vector có hạng bằng 1.

Ta nhận thấy trong cách biểu diễn trên, A sẽ phụ thuộc vào r cột đầu tiên của U và V và r giá trị khác 0 trên đường chéo của Σ nên ta có cách biểu diễn gọn hơn và được gọi là Compact SVD:

$$A = U_r \Sigma_r (V_r)^T$$

Nếu ma trận A có hạng nhỏ hơn rất nhiều so với m, n thì ta sẽ được lợi rất nhiều về mặt lưu trữ.



Hình 2.5: Ví dụ minh họa khi $m = 4, n = 6, r = 2$.

2.2.3 Sai số SVD

Chú ý rằng trong ma trận Σ, các giá trị trên đường chéo là không âm và giảm dần $\sigma_1 \geq \sigma_2 \geq \cdots \geq \sigma_r \geq 0$. Thông thường, chỉ một lượng nhỏ các 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 hạng 1:

$$A \approx A_k = U_k \Sigma_k (V_k)^T \tag{2.1}$$

Sai số do cách xấp xỉ trên chính là căn bậc hai của tổng bình phương của các giá trị riêng mà ta đã bỏ qua ở phần cuối của Σ:

$$\|A - A_k\|_F^2 = \sum_{i=k+1}^{r} \sigma_i^2. \tag{2.2}$$

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.

2.2.4 Tính SVD thủ công

Ví dụ: Tìm SVD của A, $U\Sigma V^T$, với $A = \begin{bmatrix} 3 & 2 & 2 \\ 2 & 3 & -2 \end{bmatrix}$

1. Tìm singular value σ_i, bằng cách tìm các trị riêng dương của ma trận A^TA:

$$A^TA = \begin{bmatrix} 13 & 12 & 2 \\ 12 & 13 & -2 \\ 2 & -2 & 8 \end{bmatrix}$$

Khi đó thì đa thức đặc trưng của chúng ta:

$$P(\lambda) = det(A^TA - \lambda I) = -\lambda^3 + 34\lambda^2 - 225\lambda = \lambda(\lambda - 25)(\lambda - 9)$$

Với hai trị riêng dương là $\lambda_1 = 25$, $\lambda_2 = 9$ và $\lambda_3 = 0$ thì ta được 3 singular values lần lượt là $\sigma_1 = \sqrt{25} = 5$, $\sigma_2 = \sqrt{9} = 3$ và $\sigma_3 = 0$. Lưu ý ta xắp sếp σ_i giảm dần:

$$\sigma_1 \geq \sigma_2 \geq \cdots \geq \sigma_i \geq 0$$

Vậy thì ta đã có được ma trận $\Sigma = \begin{bmatrix} 5 & 0 & 0 \\ 0 & 3 & 0 \end{bmatrix}$

2. Tìm các vector riêng đơn vị của A^TA với các trị riêng đã tìm được:

- Với $\lambda = 25$, Ta có:

$$A^TA - 25I = \begin{bmatrix} -12 & 12 & 2 \\ 12 & -12 & -2 \\ 2 & -2 & -17 \end{bmatrix}$$

$$A^TA - 25I = 0 \Leftrightarrow \left[\begin{array}{ccc|c} -12 & 12 & 2 & 0 \\ 12 & -12 & -2 & 0 \\ 2 & -2 & -17 & 0 \end{array}\right] \rightarrow \left[\begin{array}{ccc|c} 1 & -1 & 0 & 0 \\ 0 & 0 & 1 & 0 \end{array}\right]$$

Vậy khi đó cơ sở của E_{25} là $\left\{ \begin{pmatrix} 1 \\ 1 \\ 0 \end{pmatrix} \right\}$.

Tuy nhiên ta cần vector đơn vị do đó sẽ chia cho $\sqrt{1^2 + 1^2} = \sqrt{2}$ cho ma trận cơ sở.

Vậy $v_1 = \left\{ \begin{pmatrix} 1/\sqrt{2} \\ 1/\sqrt{2} \\ 0 \end{pmatrix} \right\}$.

- Với $\lambda = 9$, Ta có:

$$A^TA - 9I = \begin{bmatrix} 4 & 12 & 2 \\ 12 & 4 & -2 \\ 2 & -2 & -1 \end{bmatrix}$$

$$A^TA - 9I = 0 \Leftrightarrow \left[\begin{array}{ccc|c} 4 & 12 & 2 & 0 \\ 12 & 4 & -2 & 0 \\ 2 & -2 & -1 & 0 \end{array}\right] \rightarrow \left[\begin{array}{ccc|c} 1 & -1 & -1/2 & 0 \\ 0 & 1 & 1/4 & 0 \end{array}\right]$$

Vậy khi đó cơ sở của E_9 là $\left\{ \begin{pmatrix} 1 \\ -1 \\ 4 \end{pmatrix} \right\}$. Tuy nhiên ta cần vector đơn vị do đó sẽ chia cho $\sqrt{1^2 + (-1)^2 + 4^2} = \sqrt{18}$ cho ma trận cơ sở. Vậy $v_2 =$

$$\left\{\begin{pmatrix}1/\sqrt{18}\\-1/\sqrt{18}\\4/\sqrt{18}\end{pmatrix}\right\}.$$

- Với $\lambda = 0$, Ta có:

$$A^TA - 0I = \begin{bmatrix}13 & 12 & 2\\12 & 13 & -2\\2 & -2 & 8\end{bmatrix}$$

$$A^TA = 0 \Leftrightarrow \left[\begin{array}{ccc|c}13 & 12 & 2 & 0\\12 & 13 & -2 & 0\\2 & -2 & 8 & 0\end{array}\right] \rightarrow \left[\begin{array}{ccc|c}1 & -1 & 4 & 0\\0 & 1 & -2 & 0\end{array}\right]$$

Vậy khi đó cơ sở của E_0 là $\left\{\begin{pmatrix}-2\\2\\1\end{pmatrix}\right\}$. Tuy nhiên ta cần vector đơn vị do đó sẽ chia cho $\sqrt{2^2+2^2+1^2} = 3$ cho ma trận cơ sở. Vậy $v_3 = \left\{\begin{pmatrix}-2/3\\2/3\\1/3\end{pmatrix}\right\}$.

- Cuối cùng thì ma trận V của ta là:

$$V = \begin{bmatrix}| & | & |\\v_1 & v_2 & v_3\\| & | & |\end{bmatrix} = \begin{bmatrix}1/\sqrt{2} & 1/\sqrt{18} & -2/3\\1/\sqrt{2} & -1/\sqrt{18} & 2/3\\0 & 4/\sqrt{18} & 1/3\end{bmatrix}$$

3. Ta có thể dễ dàng tìm ma trận trực giao U qua công thức sau:

$$\sigma_i u_i = Av_i$$

- $u_1 = \frac{1}{5}\begin{bmatrix}3 & 2 & 2\\2 & 3 & -2\end{bmatrix}\begin{bmatrix}1/\sqrt{2}\\1/\sqrt{2}\\0\end{bmatrix} = \begin{bmatrix}1/\sqrt{2}\\1/\sqrt{2}\end{bmatrix}$

- $u_2 = \frac{1}{3}\begin{bmatrix} 3 & 2 & 2 \\ 2 & 3 & -2 \end{bmatrix}\begin{bmatrix} 1/\sqrt{18} \\ -1/\sqrt{18} \\ 4/\sqrt{18} \end{bmatrix} = \begin{bmatrix} 1/\sqrt{2} \\ -1/\sqrt{2} \end{bmatrix}$
- Với $\sigma_3 = 0$ thì $u_3 = \begin{bmatrix} 0 \\ 0 \end{bmatrix}$
- Vậy ma trận U của ta là:

$$U = \begin{bmatrix} | & | \\ u_1 & u_2 \\ | & | \end{bmatrix} = \begin{bmatrix} 1/\sqrt{2} & 1/\sqrt{2} \\ 1/\sqrt{2} & -1/\sqrt{2} \end{bmatrix}$$

4. Cuối cùng thì ta cũng đã tìm được SVD của A:

$$\begin{aligned} A = U\Sigma V^T &= \begin{bmatrix} 1/\sqrt{2} & 1/\sqrt{2} \\ 1/\sqrt{2} & -1/\sqrt{2} \end{bmatrix}\begin{bmatrix} 5 & 0 & 0 \\ 0 & 3 & 0 \end{bmatrix}\begin{bmatrix} 1/\sqrt{2} & 1/\sqrt{18} & -2/3 \\ 1/\sqrt{2} & -1/\sqrt{18} & 2/3 \\ 0 & 4/\sqrt{18} & 1/3 \end{bmatrix}^T \\ &= \begin{bmatrix} 3 & 2 & 2 \\ 2 & 3 & -2 \end{bmatrix} \end{aligned}$$

3 SỬ DỤNG SVD ĐỂ GIẢM CHIỀU DỮ LIỆU

3.1 Khái niệm

Sử dụng Singular Value Decomposition (SVD) là một kỹ thuật quan trọng trong việc giảm chiều dữ liệu, giúp chúng ta xử lý dữ liệu lớn và trích xuất thông tin quan trọng. Dưới đây là một số bước chi tiết để sử dụng SVD để giảm chiều dữ liệu. [3]

3.2 Các bước dùng trong SVD để giảm chiều dữ liệu

1. *Chuẩn bị dữ liệu:* Bước đầu tiên là chuyển dữ liệu thành một ma trận trước khi thực hiện SVD, điều quan trọng là chuẩn bị dữ liệu một cách thích hợp. Điều này có thể bao gồm việc xử lý dữ liệu bị thiếu, chuẩn hóa dữ liệu để các biến có cùng phạm vi, hoặc thậm chí là áp dụng các kỹ thuật tiền xử lý như phân loại, biến đổi biến định lượng thành biến phân loại, vv. Mục tiêu là chuẩn bị dữ liệu sao cho nó phản ánh đúng bản chất của vấn đề và sẵn sàng cho việc phân tích tiếp theo.

2. *Phân tích SVD:* Sau khi dữ liệu đã được chuẩn bị, tiếp theo là thực hiện SVD trên ma trận dữ liệu Kết quả của SVD sẽ bao gồm ba ma trận: U, Σ, và V^T

3. *Lựa chọn số lượng thành phần:* Trước khi giảm chiều dữ liệu, dựa trên giá trị singular (các phần tử trên đường chéo của ma trận Σ), ta chọn số lượng chiều muốn giữ lại. Thông thường, sẽ giữ lại các chiều có giá trị singular lớn và loại bỏ các chiều có giá trị singular nhỏ. Điều này thường dựa trên một ngưỡng cố định hoặc tỷ lệ phần trăm tổng lượng thông tin mà bạn muốn giữ lại (ví dụ: 0,95 tổng lượng phương sai). Số lượng thành phần chính được chọn có thể ảnh hưởng đến hiệu suất và tốc độ tính toán của mô hình của bạn.

4. *Giảm chiều dữ liệu:* Sau khi đã chọn ra k giá trị đặc biệt từ ma trận Σ, ta giữ lại k cột tương ứng trong ma trận U và V^T. Điều này tạo ra các ma trận U và V^T, có kích thước $m \times k$ và $k \times n$ tương ứng. Tính tích các ma trận U, Σ và V^T mới, sau đó tạo một ma trận dữ liệu mới với số chiều đã chọn.

5. *Kiểm tra chất lượng giảm chiều (nếu cần):* Sau khi giảm chiều dữ liệu, kiểm tra phần trăm phương sai được giữ lại để đảm bảo rằng không mất quá nhiều thông tin quan trọng.

6. *Sử dụng dữ liệu giảm chiều cho mục đích tiếp theo:* Dữ liệu giảm chiều có thể được sử dụng tùy theo mục đích của người dùng.

4 CODE MATLAB

4.1 Code

Mã nguồn 4.1: MATLAB code

```
%% Read image to A
close all; clc;
file_path = './backkhoa.jpg';
A = im2double((imread(file_path)));

%% Split 3 color Channels
R = A(:, :, 1);
G = A(:, :, 2);
B = A(:, :, 3);

%% Performing SVD on 3 channels
[Ur, Sr, Vr] = svd(R,'econ');
[Ug, Sg, Vg] = svd(G,'econ');
[Ub, Sb, Vb] = svd(B,'econ');

%% Truncate some value to compress image
Re_images = [];
r = [5 20 100];
for i = 1:length(r) %truncation value in this array
    Re_R = Ur(:,1:r(i))*Sr(1:r(i),1:r(i))*Vr(:,1:r(i))';
    Re_G = Ug(:,1:r(i))*Sg(1:r(i),1:r(i))*Vg(:,1:r(i))';
    Re_B = Ub(:,1:r(i))*Sb(1:r(i),1:r(i))*Vb(:,1:r(i))';
    image = cat(3,Re_R, Re_G, Re_B);
    Re_images = cat(4, Re_images, image); % add the image to
       the Re_images array
end

%% Plot new reconstructed image
figure('Position',[50 50 800 800])
subplot(2,2,1)
imshow(A); title ('Original Image')
```

```
31
32 for i = 1:size(Re_images, 4)
33     subplot(2,2,i+1)
34     imshow(Re_images(:,:,:,i));
35     title (['Truncated Image, r = ', num2str(r(i))]);
36 end
```

4.2 Giải thích code

1. Đọc file từ đường dẫn `file_path` có hình ảnh và chuyển thành dạng số `double` để lưu trong ma trận `A`.

```
%% Read image to A
close all; clc;
file_path = './backkhoa.jpg';
A = im2double((imread(file_path)));
```

2. Tách ma trận `A` 3 chiều thành 3 ma trận 2 chiều lần lượt là `R, G, B` để lưu các giá trị màu của 3 kênh màu Red, Green và Blue.

```
%% Split 3 color Channels
R = A(:, :, 1);
G = A(:, :, 2);
B = A(:, :, 3);
```

3. Hàm `svd` được sử dụng để phân rã 3 ma trận `R,G,B` lần lượt thành các ma trận U, Σ, V tương ứng. Tùy chọn `'econ'` cho phép tính toán hiệu quả hơn bằng cách chỉ trả về các vector riêng tương ứng với giá trị riêng khác 0, tiết kiệm bộ nhớ và thời gian xử lý.

```
%% Performing SVD on 3 channels
[Ur, Sr, Vr] = svd(R,'econ');
[Ug, Sg, Vg] = svd(G,'econ');
[Ub, Sb, Vb] = svd(B,'econ');
```

4. Quá trình nén ảnh được thực hiện bằng cách giảm `r` giá trị riêng được giữ lại từ phân tích SVD. Với mỗi giá trị `r` trong mảng `[5 20 100]`, đoạn mã tạo ra một ảnh

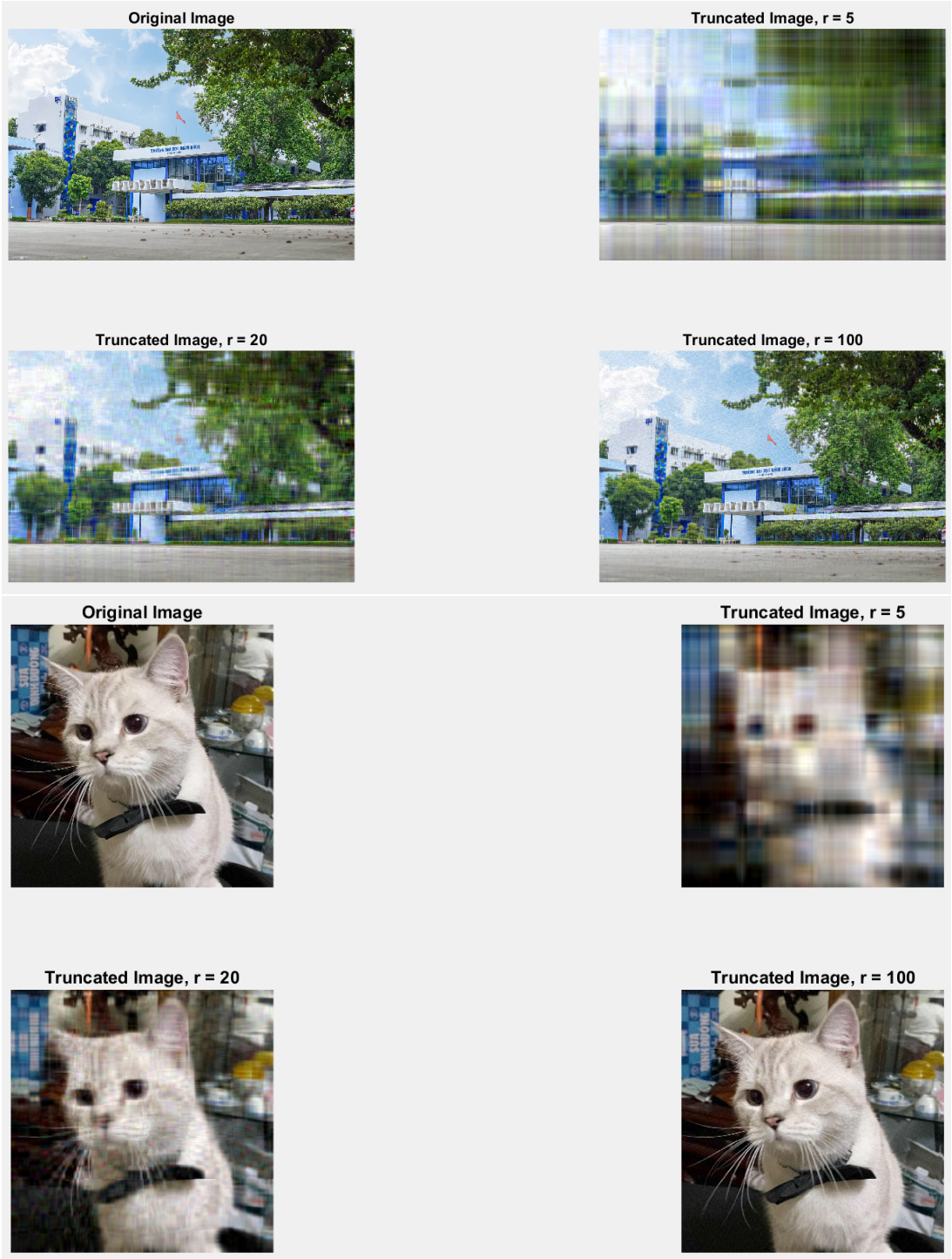
nén mới image từ các 3 kênh màu đã nén Re_R, Re_G, Re_B được tính toán dựa trên số lượng giá trị riêng giữ lại tương ứng. Các ảnh nén này được lưu vào mảng Re_images để so sánh chất lượng và kích thước tệp.

```
%% Truncate some value to compress image
Re_images = [];
r = [5 20 100];
for i = 1:length(r) %truncation value in this array
Re_R = Ur(:,1:r(i))*Sr(1:r(i),1:r(i))*Vr(:,1:r(i))';
Re_G = Ug(:,1:r(i))*Sg(1:r(i),1:r(i))*Vg(:,1:r(i))';
Re_B = Ub(:,1:r(i))*Sb(1:r(i),1:r(i))*Vb(:,1:r(i))';
image = cat(3,Re_R, Re_G, Re_B);
Re_images = cat(4, Re_images, image); % add the image to
   the Re_images array
end
```

5. Đoạn mã sau dùng để hiển thị ảnh gốc và các ảnh nén thu được từ quá trình cắt giảm giá trị riêng:

```
%% Plot new reconstructed image
figure('Position',[50 50 800 800])
subplot(2,2,1)
imshow(A); title ('Original Image')
for i = 1:size(Re_images, 4)
subplot(2,2,i+1)
imshow(Re_images(:,:,:,i));
title(['Truncated Image, r = ', num2str(r(i))]);
end
```

Đoạn mã tạo một cửa sổ hiển thị ảnh với kích thước 800x800 pixels. Ảnh gốc A được hiển thị trên ô đầu tiên. Các ảnh nén trong mảng Re_images được hiển thị trên các ô còn lại, với tiêu đề chỉ ra giá trị r tương ứng với mỗi ảnh.



Hình 4.1: Kết quả.

4.3 Tự hiện thực hoá hàm SVD

Mã nguồn 4.2: Hàm svd tự viết

```
A = [3 2 2 ; 2 3 -2];
ATA = A' * A;
[V1, D] = eig(ATA);
[sigma1, idx] = sort(sqrt(diag(D)), 'descend');
V = V1(:, idx);
Sigma = abs(diag(sigma1));
U = A * V/Sigma;
```

Dựa theo mục 2.2.4 bằng cách tính thủ công, thì ta hoàn toàn có thể viết lại hàm `svd()`.

- Tìm vector riêng và trị riêng của A:

```
A = [3 2 2 ; 2 3 -2];
ATA = A' * A;
[V1, D] = eig(ATA);
```

Đầu tiên ta tính ma trận A^TA và sử dụng lệnh `eig` để tìm vector riêng và trị riêng

- Tìm ma trận U và V từ các vector riêng và trị riêng vừa tính:

```
[sigma1, idx] = sort(sqrt(diag(D)), 'descend');
V = V1(:, idx);
Sigma = abs(diag(sigma1));
```

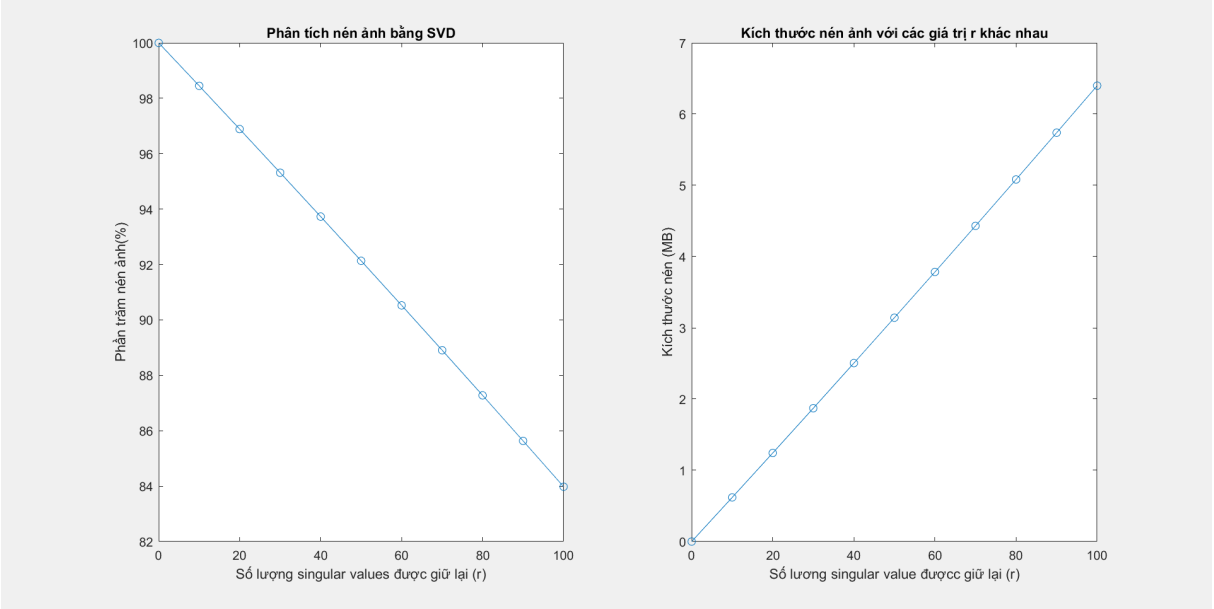
Tìm `Sigma` bằng cách lấy căn bậc 2 của các giá trị riêng trong ma trận `D` và sắp xếp theo thứ tự giảm dần. Sắp xếp lại các vecto riêng tương ứng với các giá trị riêng Viết `Sigma` thành ma trận chéo từ các giá trị `Sigma1` đã tìm được

- Tính U bằng cách sử dụng công thức $\sigma_i U = AV_i$

```
U = A * V/Sigma;
```

4.4 Phân tích sâu

4.4.1 Phân tích phần trăm ảnh bị nén



Hình 4.2: Phân tích độ nén và size của ảnh khi sử dụng SVD để nén ảnh

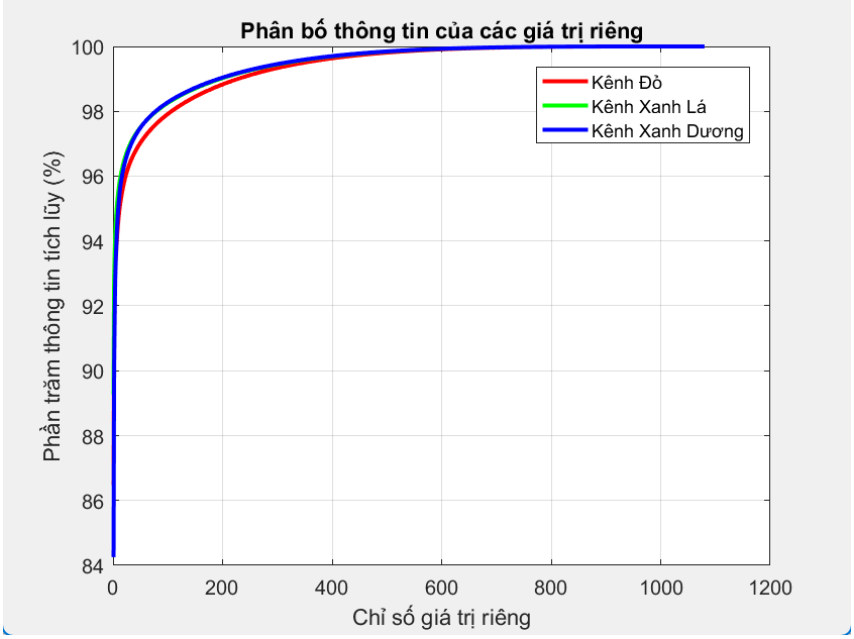
Đồ thị 4.2 tính toán độ nén của ảnh với các giá trị được giữ lại của singular values từ 0 đến 100. Ví dụ như đồ thị bên trái *Phân tích nén ảnh bằng SVD* ta thấy khi $r = 0$ tức là ta giữ lại 0 singular value (ma trận Σ rỗng), thì ta nén ảnh 100%. Và khi đó ảnh của ta có kích thước là 0MB (đồ thị phải). Tương tự với $r = 100$, khi giữ lại 100 hàng đầu của ma trận Σ thì ảnh nén ở mức 84% và size của ảnh khi đó là $\approx 6.3MB$.

4.4.2 Phân bố thông tin ứng với các singular values

Ta thấy, đường cong ở đồ thị 4.3 chỉ độ tập trung thông tin tăng nhanh ở các singular value đầu tiên.Điều đó có nghĩa là hầu hết thông tin quan trọng của ảnh đều được chứa trong những giá trị riêng lớn nhất. Nên là việc ta chọn giữ lại các singular values đầu tiên sẽ đảm bảo được các thông tin đặc trưng nhất của ảnh.

Ngoài ra, hình dạng của đường cong cũng cho thấy đặc điểm của ảnh. Đối với các ảnh đơn giản với ít chi tiết, đường cong sẽ tăng nhanh hơn. Ngược lại, với ảnh phức tạp có nhiều chi tiết, đường cong sẽ tăng chậm hơn vì thông tin được phân bố rải rác hơn giữa các giá trị riêng.

Bằng cách vẽ đường cong cho cả 3 kênh màu (đỏ, xanh lá, xanh dương), ta có thể so sánh phân bố năng lượng của các kênh màu và xác định kênh màu nào có thông tin tập



Hình 4.3: Phân tích độ tập trung thông tin của ảnh dựa trên các singular values

trung nhiều nhất hoặc phân bố đều nhất.

Tài liệu

[1] Đặng Văn Vinh, *Giáo trình Đại số tuyến tính.* NXB Đại học Quốc gia TP.HCM.

[2] C. Meyer, *Matrix analysis and applied linear algebra.* SIAM, 2001.

[3] J. Bisgard, *Analysis and Liner Algebra: The Singular Value Decomposition and Applications.* The American Mathematical Society, 2021.

Danh sách mã nguồn

Danh sách hình vẽ