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

BÀI TẬP LỚN ĐẠI SỐ TUYẾN TÍNH

PHÂN TÍCH SVD KHỬ NHIỄU ÂM THANH

Nhóm 5-L13

Giảng viên hướng dẫn: Nguyễn Hữu Hiệp
Sinh viên thực hiện: Trần Minh Nhật - 2014008

Ho Chi Minh City, 2022

Mục lục

1 CƠ SỞ LÍ THUYẾT CỦA PHÂN TÍCH SVD

1.1 Giới thiệu

Trong Đại số tuyến tính ta thường gặp các dạng bài toán chéo hóa ma trận: Với một ma trận $A \in \mathbb{R}^{n \times n}$ được gọi là *chéo hóa* được (diagonalizable) nếu tồn tại ma trận đường chéo **D** và ma trận khả nghịch **P** sao cho:

$$A = PDP^{-1} \ (1)$$

Số lượng phần tử khác 0 của ma trận đường chéo **D** chính là rank của ma trận **A**.
Nhân cả hai vế của (1) với **P** ta có:

$$AP = PD \ (2)$$

Gọi p_i, d_i lần lượt là cột thứ i của ma trận **P** và **D**. Vì mỗi một cột của vế trái và vế phải của (2) phải bằng nhau, ta sẽ có:

$$A_{p_i} = Pd_i = d_{ii}p_i \ (3)$$

với d_{ii} là phần tử thứ i của p_i

Dấu bằng thứ hai xảy ra vì **D** là ma trận đường chéo, tức d_i chỉ có thành phần d_{ii} là khác 0. Biểu (3) chỉ ra rằng mỗi phần tử d_{ii} phải trị riêng (eigenvalue) của **A** và mỗi vector cột p_i, phải là một vector riêng (eigenvector) của **A** ứng với trị riêng d_{ii}

Cách phân tích một ma trận vuông thành nhân tử như (1) còn được gọi là Eigen Decomposition.

Một điểm quan trọng là cách phân tích như (1) chỉ được áp dụng với ma trận vuông và không phải lúc nào cũng tồn tại. Nó chỉ tồn tại nếu ma trận **A** có n vector riêng độc lập tuyến tính, vì nếu không thì không tồn tại ma trận **P** khả nghịch. Thêm nữa, cách phân tích này cũng không phải là duy nhất vì nếu **P, D** thỏa mãn (1) thì **kP, D** cũng thỏa mãn với k là một số thực khác 0 bất kỳ

Việc phân tích một ma trận ra thành tích của nhiều ma trận đặc biệt khác (Matrix Factorization hoặc Matrix Decomposition) mang lại nhiều ích lợi quan trọng mà các bạn sẽ thấy: giảm số chiều dữ liệu, nén dữ liệu, tìm hiểu các đặc tính của dữ liệu, giải các hệ phương trình tuyến tính, clustering, và nhiều ứng dụng khác. Recommendation System cũng là một trong rất nhiều ứng dụng của Matrix Factorization.

Một trong những phương pháp Matrix Factorization rất đẹp của Đại số tuyến tính. Phương pháp đó có tên là Singular Value Decomposition (SVD). Mọi ma trận, không nhất thiết là vuông, đều có thể được phân tích thành tích của ba ma trận đặc biệt.

1.2 Kiến thức liên quan đại số tuyến tính

1.2.1 Trị riêng và vector riêng

Cho một ma trận vuông $A \in \mathbb{R}^{n\times n}$, nếu số vô hướng λ và vector $x \neq 0 \in \mathbb{R}^n$ thỏa 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 với λ thì $kx, k \neq 0$ 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, 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 nửa xác định dương, tất cả các trị riêng của nó đều là các số thực không âm.
Tính chất cuối cùng có thể được suy ra từ định nghĩa của ma trận (nửa) xác định dương. Thật vậy, gọi $u \neq 0$ là vector riêng ứng với một trị riêng λ của ma trận **A** xác định dương, ta có:

$$Au = \lambda u \Rightarrow u^T Au = \lambda u^T u = \lambda ||u||_2^2$$

Vì **A** là nửa xác định dương nên với mọi $u \neq 0 : u^T Au \geq 0$; $u \neq 0$ nên $||u||_2^2 > 0$. Từ đó suy ra λ là một số không âm

1.2.2 Hệ trực giao và trực chuẩn

Một hệ cơ sở $u_1, u_2, ..., u_m \in \mathbb{R}^m$ được gọi là trực giao (orthogonal) nếu mỗi vector là khác 0 và tích của hai vector khác nhau bất kỳ bằng 0:

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

Một hệ cơ sở $u_1, u_2, ..., u_m \in \mathbb{R}^m$ được gọi là trực chuẩn (orthonormal) nếu nó là một hệ trực giao và độ dài Euclidean (norm 2) của mỗi vector bằng 1:

$$u_i^T u_j = \begin{cases} 1 \;\; if \;\; i = j \\ 0 \;\; otherwise \end{cases}$$

Gọi $U = [u_1, u_2, ..., u_m]$ với $u_1, u_2, ..., u_m \in \mathbb{R}^m$ là trực chuẩn, từ (4) có thể suy ra:

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

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 dược gọi là ma trận trực chuẩn, không có định nghĩa cho ma trận trực chuẩn
Một vài tính chất:
1. $\mathbf{U}^{-1} = \mathbf{U}^T$: 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 $\mathbf{U}^T$ cũng là một ma trận trực giao.
3. Định thức (determinant) của ma trận trực giao bằng 1 hoặc -1. Điều này có thể suy ra từ việc $\det(\mathbf{U}) = \det(\mathbf{U}^T)$ và $\det(\mathbf{U})\det(\mathbf{U}^T) = \det(\mathbf{I}) = 1$
4. Ma trận trực giao thể hiện cho phép xoay (rotate) một vector. Giả sử có hai vector $\mathbf{x}, \mathbf{y} \in \mathbb{R}^m$ và ma trận trực giao $\mathbf{U} \in \mathbb{R}^{m\times m}$. Dùng ma trận này để xoay hai vector trên ta được Ux, Uy. Tích vô hướng của hai vector mới là:

$$(\mathbf{Ux})^T(\mathbf{Uy}) = \mathbf{x}^T\mathbf{U}^T\mathbf{Uy} = \mathbf{x}^T\mathbf{y}$$

như vậy phép xoay không làm thay đổi tích vô hướng giữa hai vector.
5. Giả sử $\hat{\mathbf{U}} \in \mathbb{R}^{m\times r}, r < m$ là môt ma trận con của ma trận trực giao **U** được tạo bởi r cột của **U**, ta sẽ có $\hat{\mathbf{U}}^T\hat{\mathbf{U}} = \mathbf{I}_r$. Việc này có thể suy ra từ (4)

1.3 Singular Value Decomposition

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, khử nhiễu âm thanh, clustering, các thuật toán nén và giảm chiều dữ liệu, và đặc biệt đặc biệt hiệu quả trong các bài toán recommendation.

Một số thông tin chung nhất về thuật toán SVD: 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.

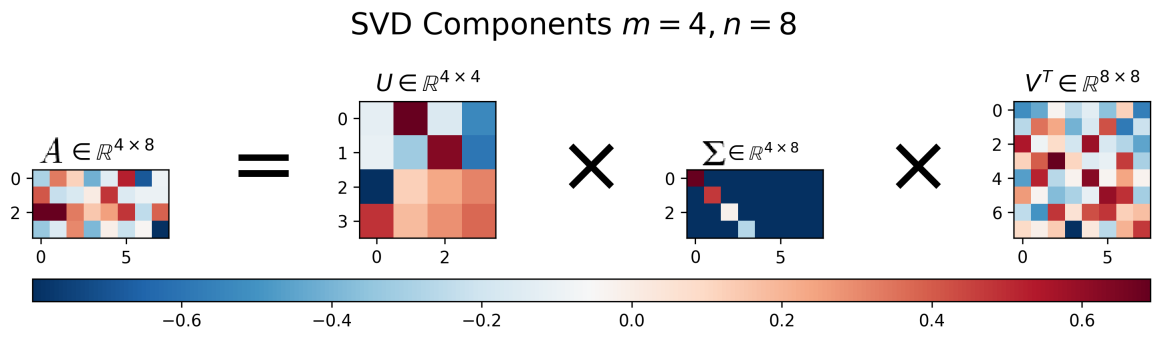
1.3.1 Phát biểu SVD

Một ma trận $\mathbf{A}_{m\times n}$ bất kỳ đều có thể phân tích thành dạng:

$$\mathbf{A}_{m\times n} = \mathbf{U}_{m\times m}\mathbf{\Sigma}_{m\times n}(\mathbf{V}_{n\times n})^T \quad (5)$$

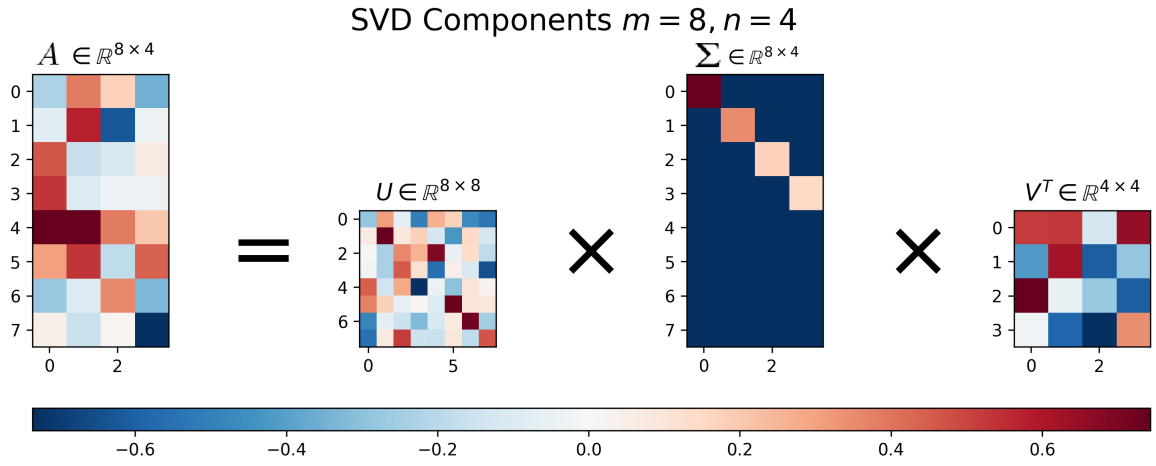
Trong đó, $\mathbf{U}, \mathbf{V}$ là các *ma trận trực giao*, $\mathbf{\Sigma}$ là ma trận đường chéo không vuông với các phần tử trên đường chéo $\sigma_1 \geq \sigma_2 \geq \cdots \geq \sigma_r \geq 0 = 0 = \cdots = 0$ và r là rank của ma trận $\mathbf{A}$. Lưu ý rằng mặc dù Σ không phải ma trận vuông, ta vẫn có thể coi nó là ma trận chéo nếu các thành phần khác không của nó chỉ nằm ở vị trí *đường chéo*, tức tại các vị trí có chỉ số hàng và chỉ số cột là như nhau.
Số lượng các phần tử khác 0 trong Σ chính là rank của ma trận $\mathbf{A}$.

Mô tả SVD của ma trận A trong trường hợp m<n (m=4 và n=8)



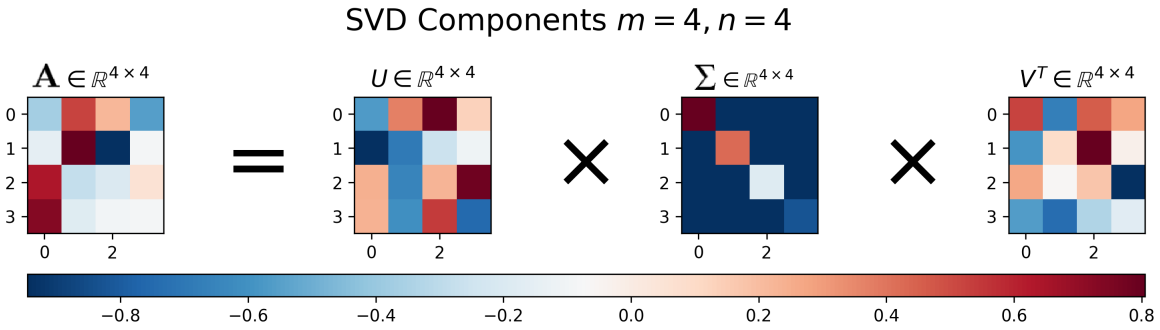
Hình 1: *Biểu diễn SVD của ma trận A khi m<n*

Mô tả SVD của ma trận A trong trường hợp m>n (m=8 và n=4)



Hình 2: *Biểu diễn SVD của ma trận A khi m>n*

Mô tả SVD của ma trận A trong trường hợp m=n (m=n=4)



Hình 3: *Biểu diễn SVD của ma trận A khi m=n*

1.3.2 Nguồn gốc của SVD

Từ công thức (5) ta có:

$$\mathbf{A}\mathbf{A}^T = \mathbf{U}\mathbf{\Sigma}\mathbf{V}^T(\mathbf{U}\mathbf{\Sigma}\mathbf{V}^T)^T = \mathbf{U}\mathbf{\Sigma}\mathbf{V}^T\mathbf{V}\mathbf{\Sigma}^T\mathbf{U}^T = \mathbf{U}\mathbf{\Sigma}\mathbf{\Sigma}^T\mathbf{U}^T = \mathbf{U}\mathbf{\Sigma}\mathbf{\Sigma}^T\mathbf{U}^{-1} \qquad (6)$$

Dấu bằng cuối cùng xảy ra vì $\mathbf{V}^T\mathbf{V} = \mathbf{I}$ do $\mathbf{V}$ là một ma trận trực giao.
Quan sát thấy rằng $\Sigma\Sigma^T$ là một ma trận đường chéo với các phần tử trên đường chéo là $\sigma_1^2, \sigma_2^2, \ldots$. Vậy (6) chính là Eigen Decomposition của $\mathbf{A}\mathbf{A}^T$. Thêm nữa, $\sigma_1^2, \sigma_2^2, \ldots$ chính là các trị riêng của $\mathbf{A}\mathbf{A}^T$. Ma trận $\mathbf{A}\mathbf{A}^T$ luôn là ma trận nửa xác định dương nên các trị riêng của nó là không âm. Các σ_i là căn bậc hai của các trị riêng của $\mathbf{A}\mathbf{A}^T$ còn được gọi là *singular values* của $\mathbf{A}$. Cái tên Singular Value Decomposition xuất phát từ đây.
Cũng theo đó, mỗi cột của $\mathbf{U}$ chính là một vector riêng của $\mathbf{A}\mathbf{A}^T$. Ta gọi mỗi cột này là *left-singular vectors* của $\mathbf{A}$. Tương tự như thế, $\mathbf{A}^T\mathbf{A} = \mathbf{V}\Sigma^T\Sigma\mathbf{V}^T$ và các cột của $\mathbf{V}$ còn được gọi là các *right-singular vectors* của $\mathbf{A}$

1.4 Các phép giảm chiều SVD

Thông thường việc phân tích suy biến một ma trận có kích thước lớn sẽ rất lâu vì đòi hỏi phải giải phương trình đặc trưng để tìm ra các giá trị đặc trưng, từ đó suy ra ma trận đường chéo $\mathbf{\Sigma}$. Từ phương trình phân tích riêng $\mathbf{A^T A = V \Sigma^T \Sigma V^T}$ suy ra $\mathbf{A^T AV = V \Sigma^T \Sigma}$, sử dụng phương trình ứng với từng cột cả 2 vế trái và phải để tính ra được các vector riêng ứng với mỗi trị riêng và suy ra được ma trận $\mathbb{V}$. Cách tìm ma trận $\mathbb{U}$ cũng được suy ra tương tự từ phương trình phân tích riêng $\mathbf{A^T = U \Sigma \Sigma^T U^T}$. Quá trình này phải trải qua nhiều bước và khi kích thước ma trận lớn, chi phí thời gian và lưu trữ sẽ rất lớn. Vì vậy các dạng giảm chiều SVD sẽ rút gọn quá trình tính toán.

1.4.1 Compact SVD

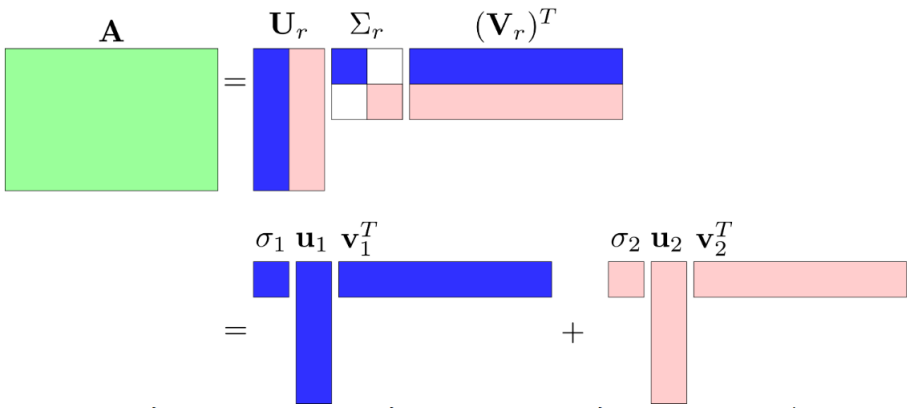
Viết lại biểu thức (5) dưới dạng tổng của các ma trận rank 1:

$$A = \sigma_1 u_1 v_1^T + \sigma_2 u_2 v_2^2 + ... + \sigma_r u_r v_r^T$$

với chú ý rằng mỗi $u_i v_i^T, 1 \leq i \leq r$ là một ma trận có rank bằng 1
Rõ ràng trong cách biểu diễn này, ma trận $\mathbf{A}$ chỉ phụ thuộc vào r cột đầu tiên của $\mathbf{U}, \mathbf{V}$ và r giá trị khác 0 trên đường chéo của ma trận Σ. Vì vậy ta có một cách phân tích *gọn* hơn và gọi là *compact SVD*:

$$\mathbf{A} = \mathbf{U}_r \Sigma_r (\mathbf{V}_r)^T$$

Với $\mathbf{U}_r, \mathbf{V}_r$ lần lượt là ma trận được tạo bởi r cột đầu tiên của $\mathbf{U}$ và $\mathbf{V}$. Σ_r là ma trận con được tạo bởi r hàng đầu tiên và r cột đầu tiên của Σ. Nếu ma trận $\mathbf{A}$ có rank nhỏ hơn rất nhiều so với số hàng và số cột $r \ll m, n$, ta sẽ được lợi nhiều về việc lưu trữ.
Dưới đây là ví dụ minh hoạ với $m = 4, n = 6, r = 2$



Hình 4: *Biểu diễn SVD dạng thu gọn và biểu diễn ma trận dạng tổng các ma trận có rank bằng 1.*

1.4.2 Truncated SVD

Chú ý rằng trong ma trận Σ, các giá trị trên đường chéo là không âm và giảm dần $\sigma_1 \geq \sigma_2 \geq \ldots, \geq \sigma_r \geq 0 = 0 = \cdots = 0$. 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 $\mathbf{A}$ bằng tổng của $k < r$ ma trận có rank 1:

$$\mathbf{A} \approx \mathbf{A}_k = \mathbf{U}_k \Sigma_k (\mathbf{V}_k)^T = \sigma_1 \mathbf{u}_1 \mathbf{v}_1^T + \sigma_2 \mathbf{u}_2 \mathbf{v}_2^2 + \cdots + \sigma_k \mathbf{u}_k \mathbf{v} k^T \quad (7)$$

Định lý dưới đây nói rằng sai số do cách xấp xỉ trên chính là căn bậc hai của tổng bình phương của các singular values mà ta đã bỏ qua ở phần cuối của Σ. Ở đây sai số được định nghĩa là Frobineous

norm của hiệu hai ma trận:
Định lí

$$||\mathbf{A} - \mathbf{A}_k||_F^2 = \sum_{i=k+1}^{r} \sigma_i^2 \quad (8)$$

Chứng minh
Sử dụng tính chất $||\mathbf{X}||_F^2 = \text{trace}(\mathbf{X}\mathbf{X}^T)$ và $\text{trace}(\mathbf{X}\mathbf{Y}) = \text{trace}(\mathbf{Y}\mathbf{X})$ với mọi ma trận $\mathbf{X}, \mathbf{Y}$ ta có:

$||\mathbf{A} - \mathbf{A}_k||_F^2 = ||\sum_{i=k+1}^{r} \sigma_i \mathbf{u}_i \mathbf{v}_i^T||_F^2 (9)$ $= \text{trace}\left\{\left(\sum_{i=k+1}^{r} \sigma_i \mathbf{u}_i \mathbf{v}_i^T\right)\left(\sum_{j=k+1}^{r} \sigma_j \mathbf{u}_j \mathbf{v}_j^T\right)^T\right\} (10)$
$= \text{trace}\left\{\sum_{i=k+1}^{r} \sum_{j=k+1}^{r} \sigma_i \sigma_j \mathbf{u}_i \mathbf{v}_i^T \mathbf{v}_j \mathbf{u}_j^T\right\} (11)$ $= \text{trace}\left\{\sum_{i=k+1}^{r} \sigma_i^2 \mathbf{u}_i \mathbf{u}_i^T\right\} (12)$
$= \text{trace}\left\{\sum_{i=k+1}^{r} \sigma_i^2 \mathbf{u}_i^T \mathbf{u}_i\right\} (13)$ $= \text{trace}\left\{\sum_{i=k+1}^{r} \sigma_i^2\right\} (14)$ $= \sum_{i=k+1}^{r} \sigma_i^2 (15)$

Dấu bằng ở (12) là vì $\mathbf{V}$ là ma trận trực giao (xem (4))
Dấu bằng ở (13) là vì hàm trace có tính chất giao hoán.
Dấu bằng ở (15) là vì biểu thức trong dấu ngoặc của (14) là một số vô hướng.

Thay $k = 0$ ta sẽ có:

$$||\mathbf{A}||_F^2 = \sum_{i=1}^{r} \sigma_i^2 \quad (16)$$

Từ đó:

$$\frac{||\mathbf{A} - \mathbf{A}_k||_F^2}{||\mathbf{A}||_F^2} = \frac{\sum_{i=k+1}^{r} \sigma_i^2}{\sum_{j=1}^{r} \sigma_j^2} \quad (17)$$

Như vậy, **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% lương thông tin trong $\mathbf{A}$, trước hết ta tính $\sum_{j=1}^{r} \sigma_j^2$, sau đó chọn k là số 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, ma trận $\mathbf{A}_k$ có rank là k, là một ma trận có rank nhỏ. Vì vậy, Truncated SVD còn được coi là một phương pháp *Low-rank approximation*

1.4.3 Best Rank k Approximation

Người ta chứng minh được rằng (Singular Value Decomposition - Princeton) $\mathbf{A}_k$ chính là nghiệm của bài toán tối ưu:

$$\min_{\mathbf{B}} ||\mathbf{A} - \mathbf{B}||_F \text{ s.t.rank}(\mathbf{B}) = k \quad (17)$$

và như đã chứng minh ở trên $||\mathbf{A} - \mathbf{A}_k||_F^2 = \sum_{i=k+1}^{r} \sigma_i^2$
Nếu sử dụng norm 2 của ma trận thay vì Frobenius norm để đo sai số, $\mathbf{A}_k$ cũng là nghiệm của bài toán tối ưu:

$$\min_{\mathbf{B}} ||\mathbf{A} - \mathbf{B}||_2 \text{ s.t.rank}(\mathbf{B}) = k \quad (18)$$

và sai số: $||\mathbf{A} - \mathbf{A}_k||_2^2 = \sigma_{k+1}^2$. Định nghĩa của norm 2 của một ma trận là:

$$||\mathbf{A}||_2 = \max_{||\mathbf{x}||_2=1} ||\mathbf{A}\mathbf{x}||_2$$

Đây là lý do căn bậc hai của tổng bình phương của các phần tử của một ma trận không được gọi là norm 2 như đối với vector.

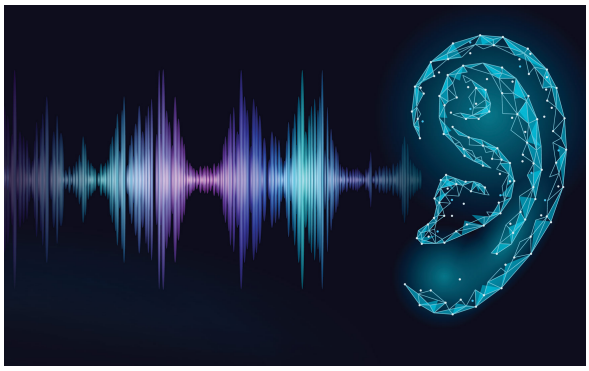
$$||\mathbf{A}||_2 = \sigma_1$$

Frobenius norm và norm 2 là hai norms được sử dụng nhiều nhất trong ma trận. Như vậy, xét trên cả hai norm này, Truncated SVD đều cho xấp xỉ tốt nhất. Vì vậy Truncated SVD còn được gọi là Best low-rank Approximation.

2 ỨNG DỤNG SVD VÀO KHỬ NHIỄU ÂM THANH

2.1 Đặt vấn đề

Với cuộc sống của chúng ta hiện nay, thì rất quen thuộc chính là những âm thanh xung quanh với từng hoàn cảnh khác nhau như: tiếng của giảng viên trong buổi giảng bài trên lớp; âm thanh trong buổi hòa nhạc; tiếng nói phát ra trong cuộc trò chuyện,...



Chúng ta thường ghi lại video hoặc những âm thanh cần thiết đến khi cần thì dùng lại, hiện nay thì điện thoại thông minh và các thiết bị ghi âm đã phát triển và có thể phục vụ nhu cầu đó của chúng ta. Tuy nhiên, có một điều khó có thể tránh khỏi đó là những tạp âm (tiếng gió, tiếng ồn, tiếng chó sủa,...) khiến cho chất lượng âm thanh giảm gây khó chịu cho người nghe hoặc tạp âm quá nhiều khiến chúng ta không thể nghe được nội dung chính cần nghe lại
Vậy vấn đề đặt ra là làm thế nào để loại bỏ những tạp âm đó hay còn gọi là khử nhiễu giúp nâng cao chất lượng âm thanh tốt hơn có thể

2.2 Khử nhiễu âm thanh dùng SVD

- Hiện nay chúng ta có khá nhiều phần mềm và ứng dụng hỗ trợ việc xử lí âm thanh, đặc biệt là tính năng khử nhiễu. Có thể kể đến như AudioDroid, Ringtone Editor, WavePad Sound Editor,... Các phần mềm này thường được viết dựa trên mã nguồn xử lí nhiễu âm thanh của các ngôn ngữ lập trình như C/C++, Python, Matlab,...Trên phương diện báo cáo, nhóm tác giả sẽ thực hiện khử nhiễu âm thanh ứng dụng SVD của Matlab
- Về thuật toán: trong Matlab có rất nhiều phương pháp khác nhau, cũng như hỗ trợ các thuật toán để thực hiện khử nhiễu âm thanh như phân tích PCA, biến đổi Fourier hữu hạn, phân tích SVD,... Tuy nhiên việc khử nhiễu âm thanh bằng SVD có nhiều ưu điểm, thông qua việc biến đổi dữ liệu mới trong không gian mà vẫn đảm bảo được hạng của ma trận và kích thước các nhân tử của ma trận trong một giới hạn cho phép. Như vậy nhờ vào thuật toán này, chúng ta có thể xóa bỏ những vết nhiễu của âm thanh, từ đó giúp đầu ra của âm thanh chất lượng hơn

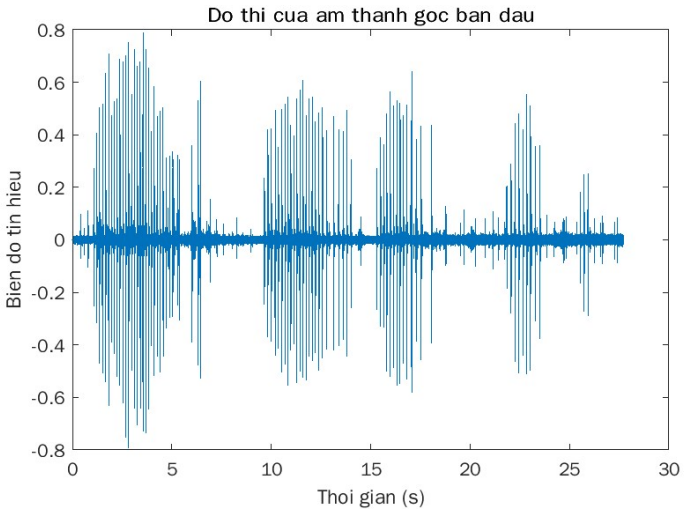
2.3 Quá trình khử nhiễu âm thanh bằng SVD dùng Matlab

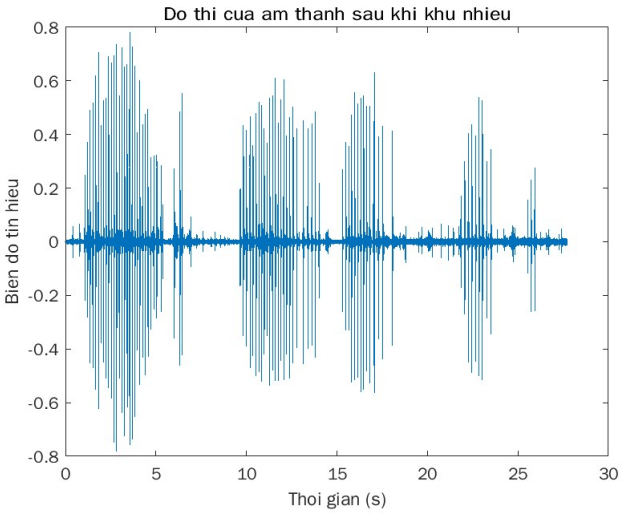
Các bước phân tích SVD cho một file âm thanh trên Matlab

1. Đọc file âm thanh (định dạng wav, mp3,...) vào Matlab sử dụng lệnh (audioread)
[Y,Hz] = audioread('input.wav');
Với Y là giá trị mẫu, Hz là tần số của mẫu

2. Chúng ta có thể nghe thử để kiểm tra file âm thanh vừa với đọc vào Matlab dùng lệnh (sound)
sound(Y,Hz);

3. Vẽ biểu đồ cho file âm thanh để trực quan hơn, dễ quan sát hơn dùng lệnh (plot)
t = 0:1/Hz:(length(Y)-1)/Hz;
figure('name','Do thi cua am thanh goc ban dau');
xlabel('Thoi gian (s)');
ylabel('Bien do tin hieu');
title('Do thi cua am thanh goc ban dau');
plot(t, Y);

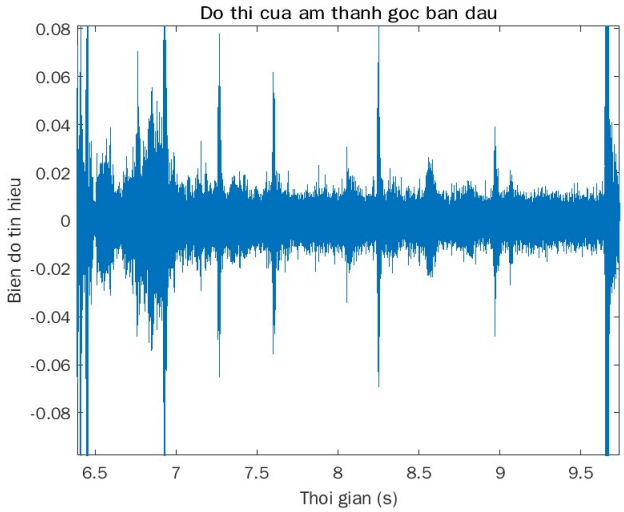


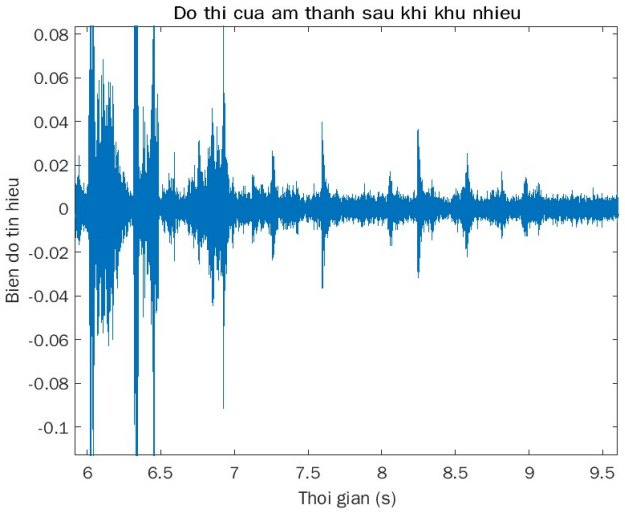


4. Tiến hành phân tích SVD để khử đoạn âm thanh bị nhiễu đó. Mã hóa thành một ma trận gồm các nhân tử để nhận dạng được, từ dó phân tích SVD để khử nhiễu
Chuyển âm thanh về dạng ma trận X với tỉ lệ nhỏ hơn để phân tích, vì kích thước của Y rất lớn (888582, 1) nên không thể tính toán trên ma trận này, ta dùng lệnh (reshape):
X = reshape(Y,[ ],822);

Phân tích SVD của ma trận X với lệnh (svd) do Matlab cung cấp sẵn
[U,S,V] = svd(X); .
Với U, V là các ma trận trực giao, S là ma trận đường chéo như đã phân tích trong phần cơ sở lí thuyết trước đó

5. Qua quá trình phân tích ma trận A thành SVD, chọn các điểm trong không gian chính là những đoạn bị tạp âm để xử lí. Sau khi xử lí hoàn tất, ta dùng câu lệnh "figure, subplot" để hiển thị đoạn âm thanh mà ta vừa xử lí, để đối chiếu so sánh.





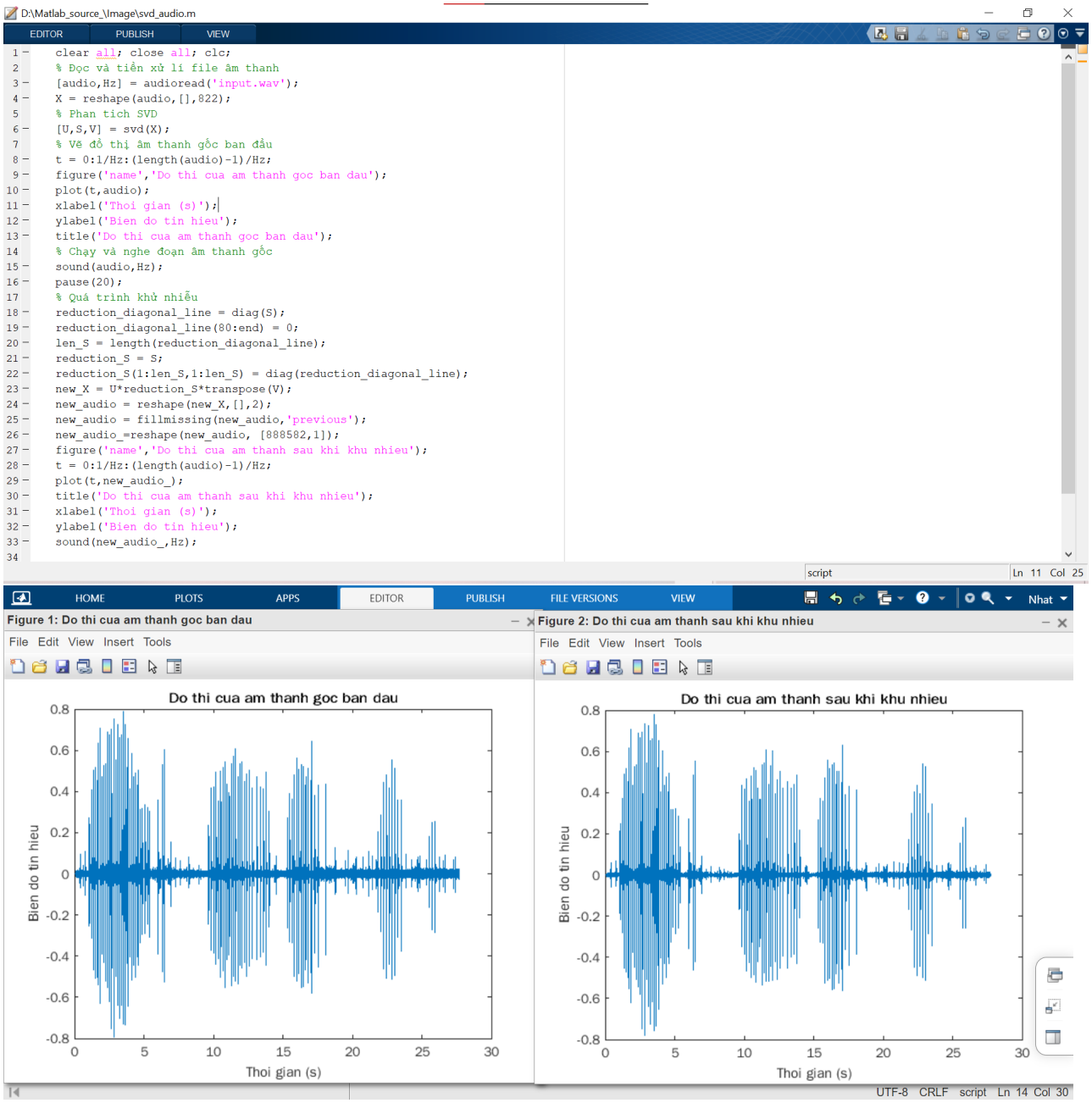
2.4 Mã nguồn Matlab

```
clear all; close all; clc;
% Doc va tien xu l file am thanh
[audio,Hz] = audioread('input.wav');
X = reshape(audio,[],822);
% Phan tich SVD
[U,S,V] = svd(X);
% Ve do thi am thanh goc ban dau
t = 0:1/Hz:(length(audio)-1)/Hz;
figure('name','Do thi cua am thanh goc ban dau');
plot(t,audio);
xlabel('Thoi gian (s)');
ylabel('Bien do tin hieu');
title('Do thi cua am thanh goc ban dau');
% Chay va nghe doan am thanh goc
sound(audio,Hz);
pause(20);

% Qua trinh khu nhieu
reduction_diagonal_line = diag(S);
reduction_diagonal_line(80:end) = 0;
len_S = length(reduction_diagonal_line);
reduction_S = S;
reduction_S(1:len_S,1:len_S) = diag(reduction_diagonal_line);
new_X = U*reduction_S*transpose(V);
new_audio = reshape(new_X,[],2);
new_audio = fillmissing(new_audio,'previous');
new_audio_=reshape(new_audio, [888582,1])
figure('name','Do thi cua am thanh sau khi khu nhieu');
t = 0:1/Hz:(length(audio)-1)/Hz;
plot(t,new_audio_);
title('Do thi cua am thanh sau khi khu nhieu');
xlabel('Thoi gian (s)');
ylabel('Bien do tin hieu');
sound(new_audio_,Hz);
```

Một số lệnh đã sử dụng

- clear all: Xóa các biến, dữ liệu trước đó
- close all: Đóng tất cả các cửa sổ, đồ thị đang mở.
- clc: Xóa màn hình Command Line
- audioread('file name'): Mở 1 file âm thanh cần khử nhiễu có sẵn trong máy tính.
- reshape: Thay đổi hình dạng của một mảng mà không thay đổi dữ liệu.
- [U,S,V] = svd (A): Phân tích SVD ma trận A.
- Sigmas= diag(S): Gán các giá trị trên đường chéo chính của ma trận S.
- plot: Vẽ đồ thị .
- for i=1:length(rank) approx_sigmas = sigmas: Loại bỏ các trị riêng thấp.
- approx_S = S: Gắn trị riêng đã xử lí vào ma trận S.
- xlabel, ylabel:Đặt tên trục x, trục y.
- sound: Lệnh phát âm thanh.
- Diag: Lấy các giá trị trên đường chéo.
- Reduction_diagonal_line(80:end) = 0: Khử các giá trị đường chéo từ hàng 80 trở đi về giá trị 0.
- Fillmissing: Điền vào các giá trị trống vào các giá trị đứng trước đó.



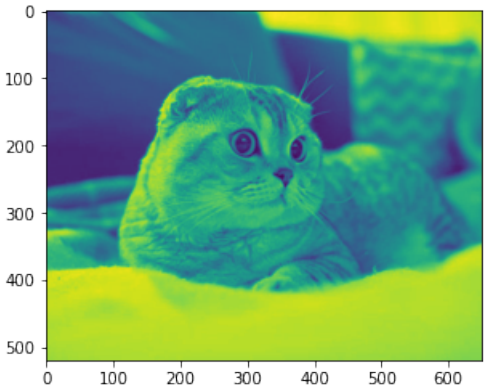
3 MỘT VÀI ỨNG DỤNG KHÁC CỦA SVD

3.1 Image Compression (Nén hình ảnh)

Giả sử ta có một tập dữ liệu nhiều ảnh có kích thước rất lớn. Khả năng lưu trữ của server là có hạn. Trong tình huống này ta có thể sau nén một bức ảnh bằng thuật toán truncate SVD để giảm kích thước của bộ ảnh vừa với dung lượng server mà thông tin của các bức ảnh vẫn giữ được một lượng lớn.
Để lưu ảnh với Truncated SVD, ta sẽ lưu các ma trận $\mathbf{U}_k \in \mathbb{R}^{m\times k}, \Sigma_k \in \mathbb{R}^{k\times k}, \mathbf{V}_k \in \mathbb{R}^{n\times k}$. Tổng số phần tử phải lưu là $k(m+n+1)$ với chú ý rằng ta chỉ cần lưu các giá trị trên đường chéo của $4k(m+n+1)$. Nếu so giá trị này với ảnh gốc có kích thước mn, mỗi giá trị là 1 số nguyên 1 byte, tỉ lệ nén là:

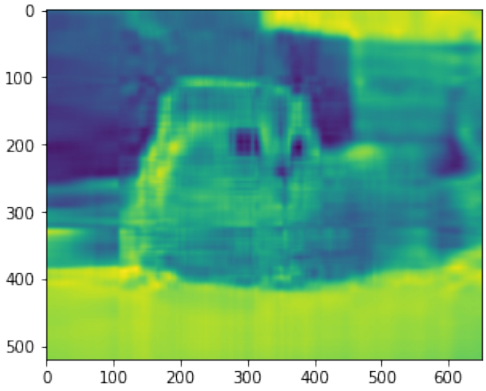
$$\frac{4k(m+n+1)}{mn}$$

Khi $k \ll m, n$, ta có tỉ lệ nén xấp xỉ 0.69, tức đã tiết kiệm được khoảng 30% bộ nhớ.



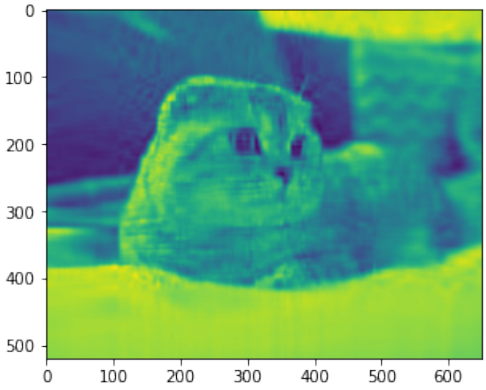
Hình 5: *Bức ảnh gốc khi chưa nén*

Nén theo phương pháp SVD sao cho chỉ lấy lần lượt 10 trị riêng lớn nhất để lưu trữ thông tin



Hình 6: *Nén ảnh với 10 trị riêng lớn nhất*

Nén theo phương pháp SVD sao cho chỉ lấy lần lượt 20 trị riêng lớn nhất để lưu trữ thông tin



Hình 7: *Nén ảnh với 10 trị riêng lớn nhất*

3.2 Áp dụng cho thuật toán recommendation system

Trong các thuật toán recommendation phương pháp SVD cũng tỏ ra hiệu quả hơn so với các phương pháp matrix factorization sử dụng gradient descent. Ý tưởng khi áp dụng vào recommendation system cũng tương tự như nén ảnh đó là căn cứ vào những cặp (user, item) đã được rating của ma trận tiện ích ta sẽ tìm ra một ma trận xấp xỉ tốt nhất với ma trận tiện ích. Sử dụng ma trận xấp xỉ để dự báo cho những cặp (user, item) chưa được rating.

3.3 Giảm chiều dữ liệu

Các ma trận A_k gần khít với A và có hạng bằng k nên ta có thể dùng SVD để giảm chiều dữ liệu.
Việc giảm chiều dữ liệu giúp ta có khả năng biểu diễn bộ dữ liệu đó một cách khá chính xác trên đồ thị. Giả sử ta có một tập dữ liệu 4 chiều và ta muốn biểu diễn tập dữ liệu này trên đồ thị thì ta có thể dùng SVD để giảm chiều dữ liệu về 3.
Việc giảm chiều dữ liệu nhưng vẫn giữ được đặc trưng của bộ dữ liệu còn giúp số lượng tham số cần tính toán là ít hơn nên tính toán nhanh hơn.
Ngoài các ứng dụng trên, SVD còn có các ứng dụng trong tối ưu cực trị rời rạc, lát cắt cực đại, K-means Clustering, Graph Partitioning, ... với một performance tính toán tương đối, rất thích hợp cho lượng dữ liệu lớn nhưng công dụng chính nhất vẫn là **giảm chiều dữ liệu**

Tài liệu

[1] CS168, *The Singular Value Decomposition (SVD) and Low-Rank Matrix Approximations*
http://theory.stanford.edu/~tim/s15/l/l9.pdf

[2] Kaggle, *Phương pháp phân tích suy biến.*
https://www.kaggle.com/code/phamdinhkhanh/singular-value-decomposition/notebook

[3] Machine learning cơ bản, *Bài 26: Singular Value Decomposition*
https://machinelearningcoban.com/2017/06/07/svd/

[4] Standford University, *Singular Value Decomposition*
https://www.youtube.com/watch?v=P5mlg91as1c

[5] Viblo, *[Handbook] Singular Values Decomposition và một số ứng dụng*
https://viblo.asia/p/handbook-singular-values-decomposition-va-mot-so-ung-dung-yMnKMOoml7P

[6] Mathwork, *QR decomposition: History and its Applications*
https://www.mathworks.com/help/audio/ug/spectral-descriptors.html?searchHighlight=noise%20reduction%20audio%20code&s_tid=srchtitle

[7] Youtube, *Lập trình Matlab xử lí âm thanh*
https://www.youtube.com/watch?v=xXk9nS6YdU8&feature=youtu.be

[8] Youtube, *MATLAB Tutorial for Beginners 43 - Audio Analysis Using MATLAB | Audio Analysis in MATLAB*
https://www.youtube.com/watch?v=SJRHv5vvlnU

[9] Vimach.net, *MATLAB cơ bản 9: Vecto (mảng) với file âm thanh*
https://vimach.net/threads/matlab-co-ban-9-vecto-mang-voi-file-am-thanh.160/