ĐẠI HỌC QUỐC GIA THÀNH PHỐ HỒ CHÍ MINH
TRƯỜNG ĐẠI HỌC BÁCH KHOA
KHOA KHOA HỌC - KỸ THUẬT MÁY TÍNH

Cơ sở tính toán cho Khoa Học Máy Tính

Bài tập lớn

Community Structure Identification

Giảng viên hướng dẫn: **Nguyễn An Khương**
Nguyễn Tiến Thịnh

Sinh viên thực hiện:

Lâm Phùng Phước Vinh	2270093
Lê Phương Nam	2170545
Ngô Quốc Khánh	1812593
Nguyễn Quốc Anh	2270074

Ho Chi Minh, 10-2022

Mục lục

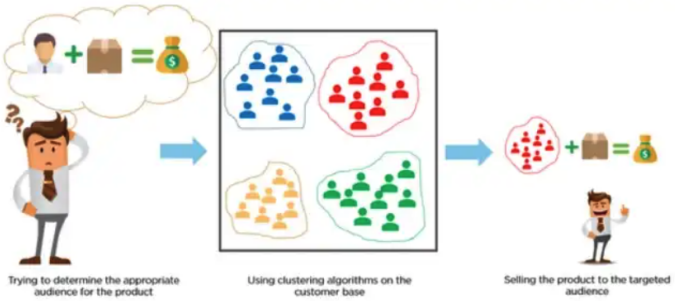
1 Giới thiệu

1.1 Mô tả bài toán

Có nhiều các cộng đồng ở trong nhiều lĩnh vực khác nhau ở quanh chúng ta, việc phát hiện ra các cộng đồng để phục vụ cho khác mục đích như phân cụm, phân nhóm là hết sức quan trọng.

Một số bài toán về phân nhóm, phân cụm bằng các giải thuật cổ điển đang đối mặt với nhiều vấn đề, hiệu năng giảm, hay các giải thuật cũ đều là giải thuật thuộc về lớp **NP và NP-hard**. Hơn thế nữa là lượng dữ liệu sinh ra ngày càng nhiều, các hệ thống cũ đối diện với khả năng không còn đáp ứng được với các thách thức mới.

Ví dụ: Bài toán **Segmentation with Clustering (phân đoạn thị trường)** là bài toán kinh điển cho thị trường mua bán. Các nhà doanh nghiệp hoạch định được lượng user hiện tại thuộc về các phân khúc nào, để từ đó đưa ra các hành động cụ thể, các chương trình bán hàng khác nhau cho mỗi nhóm đối tượng khác nhau, từ đó nhằm tối ưu hoá doanh thu, giảm chi phí và tăng trải nghiệm của khách hàng.



Hình 1: *Bài toán phân loại khách hàng*

Trong nội dung bài báo cáo, nhóm sẽ trình bày về bài toán **Market Segmentation with Clustering**.

2 Dẫn nhập

2.1 Mô tả bài toán

Bài toàn phân cụm dữ liệu là bài toán phân chia tập dữ liệu ban đầu thành các nhóm nhỏ hơn nhằm mục đích tóm tắt hoặc nâng cao hiểu biết về tệp dữ liệu ban đầu. Phân cụm dữ liệu có một lịch sử lâu dài và đã có một số lượng lớn các kỹ thuật phân cụm dữ liệu được phát triển và áp dụng trong nhiều lĩnh vực khác nhau.

Trong thực tế, dữ liệu hiện nay đang có xu hướng càng ngày càng trở nên phức tạp hơn và có số lượng chiều (đặc trưng của dữ liệu) nhiều hơn so với dữ liệu thông thường và nó đặt ra những thách thức mới như:

- Trong khi số lượng chiều của dữ liệu tăng lên thì chỉ có một số ít chiều dữ liệu có ý nghĩa trong việc phân cụm, việc dữ liệu tồn tại các chiều dư thừa hoặc không có nhiều ý nghĩa

đối với dữ liệu ảnh hưởng đến độ chính xác trong việc phân tích dữ liệu ban đầu

- Dữ liệu ngày càng thưa thớt vì các điểm dữ liệu có khả năng nằm ở các chiều khác nhau, dẫn đến việc xác định khoảng cách giữa các điểm dữ liệu lúc này sẽ không còn phản ánh đúng được mối quan hệ giữa 2 điểm dữ liệu
- Khó khăn trong việc tìm ra giải pháp và xây dựng model phù hợp với dữ liệu quá nhiều chiều
- Khó khăn trong việc mô tả dữ liệu một cách trực quan (data visualization).
- Overfitting model.

Việc giảm chiều dữ liệu là rất quan trọng trong bài toán phân cụm, không chỉ giúp giảm mức độ phức tạp, thời gian xử lí của giải thuật mà còn cung cấp cho người dùng một bức tranh rõ ràng hơn về dữ liệu cần được quan tâm
Có nhiều giải pháp cũng như cách tiếp cận để giải quyết thách thức về số lượng lớn chiều của dữ liệu đầu vào, ở bài tập lớn này nhóm đề xuất cách tiếp cận giải quyết vấn đề dựa trên PCA (principal component analysis) và SVD (Singular value decomposition)
Về mặt ý tưởng :

- PCA: Cùng một tập dữ liệu, với góc nhìn khác nhau ta sẽ thu được kết quả khác nhau, PCA là một thuật toán tìm ra một không gian mới (ít chiều hơn so với không gian ban đầu) sao cho phương sai của dữ liệu là lớn nhất trên mỗi trục
- SVD: là một dạng của matrix factorixation, phân tích ma trận bất kì ban đầu thành tích của 3 ma trận: trong đó 2 ma trận ngoài cùng bên trái và bên phải lần lượt là các ma trận trực giao và ma trận S là ma trận đường chéo không vuông với các giá trị trên đường chéo là các trị riêng của ma trận gốc

2.2 Mô tả tập dữ liệu

Như vậy bài toán Market Segmentation là bài toán community detection để tìm được/ phân loại các khách hàng trong tập dữ liệu ban đầu những khách hàng có 1 hoặc một vài đặc điểm trong hành vi mua hàng như nhau để gom lại thành một nhóm
Input: tập dữ liệu của khách hàng tại một trung tâm thương mại:

- CustomerID: mã định danh của khách hàng tại trung tâm thương mại
- Gendor: Giới tính : Male/Female
- Age: độ tuổi của khách hàng
- Annual Income: thu nhập hàng năm của khách hàng (ngàn USD)
- Spending Score: điểm được cho bởi trung tâm thương mại dựa trên hành vi và thói quen mua hàng của khách hàng

Output: Danh sách k nhóm đã được nhận dạng và được đánh số từ 1,2,...,k. Tương ứng lần lượt với từng nhóm sẽ là tập hợp các khách hàng có trong nhóm đó
Mô hình hóa bài toán: Với dữ liệu đầu vào như trên ta sẽ mô hình hóa bài toán như sau: mỗi customerID sẽ tương ứng với một node, các thuộc tính còn lại của dữ liệu đầu vào tương ứng lần lượt là các chiều trong không gian, các chiều này dùng để xác định vị trí của node dựa trên dữ liệu đầu vào, khoảng cách giữa các node sẽ chính là euclidean distance

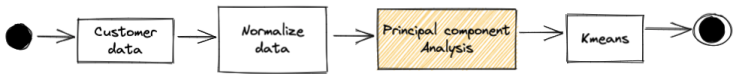
3 Cách tiếp cận

Thông thường, trong các bài toán về phân cụm, phân nhóm khách hàng, nguồn dữ liệu sẽ là các bản ghi về thông tin cá nhân, thông tin giao dịch của khách hàng. Vì thế trong hầu hết các trường hợp, một số thông tin của khách hàng có thể không cần thiết để phục vụ bài toán. Ví dụ như thông tin về *CustomerID, hay số căn cước công dân,....*

Chính vì thế việc có một giải thuật đủ tốt, hiệu quả, để thu giảm số chiều dữ liệu sao cho hiệu quả, đồng thời vẫn giữ lại được các đặc trưng của tập dữ liệu mà không bị mất mát quá nhiều.

Các bài toán về giảm chiều dữ liệu có thể giải quyết hiệu quả với các giải thuật như **Princical Component Analysis(PCA)** hoặc **Singular Value Decomposition(SVD)** rất hiệu quả trong bài toán này.

Dữ liệu sau khi đã được thu giảm về số chiều nhỏ hơn, ta có thể áp dụng một số giải thuật về phân cụm khá hiệu quả như **KMeans** để giải quyết bài toán.



Hình 2: *Luồng dữ liệu với bài toán phân đoạn khách hàng.*

3.1 Giải thuật PCA

3.1.1 Khái niệm

Định nghĩa Principal Component Analysis (PCA) là gì?
Principal Component Analysis (PCA) là Phân tích thành phần chính (PCA) - một thuật ngữ thuộc nhóm Technology Terms - Công nghệ thông tin.

Phép phân tích thành phần chính (PCA) là một kỹ thuật được sử dụng để xác định một số lượng nhỏ của các biến không tương quan được gọi là thành phần chủ yếu từ một tập lớn của dữ liệu. Kỹ thuật này được sử dụng rộng rãi để nhấn mạnh sự thay đổi và nắm bắt được mô hình mạnh mẽ trong một tập dữ liệu. Phát minh bởi Karl Pearson vào năm 1901, phân tích thành phần chính là một công cụ được sử dụng trong mô hình dự báo và phân tích dữ liệu thăm dò. phép phân tích thành phần chính được coi là một phương pháp thống kê hữu ích và được sử dụng trong các lĩnh vực như nén hình ảnh, nhận diện khuôn mặt, khoa học thần kinh và đồ họa máy tính.

Giải thích ý nghĩa
Phép phân tích thành phần chính giúp dễ dàng hơn làm cho dữ liệu để khám phá và hình dung. Đây là một kỹ thuật phi tham số đơn giản để trích xuất thông tin từ các tập dữ liệu phức tạp và khó hiểu. Phép phân tích thành phần chính là tập trung vào số lượng chênh lệch tối đa với số lượng ít nhất các thành phần chính. Một trong những lợi thế riêng biệt gắn liền với phép phân tích thành phần chính là một khi mô hình được tìm thấy trong các dữ liệu có liên quan, nén dữ liệu cũng được hỗ trợ. Một làm cho việc sử dụng phép phân tích thành phần chính để loại bỏ các số biến hoặc khi có quá nhiều yếu tố dự báo so với số quan sát hoặc để tránh đa cộng tuyến. Nó có liên quan chặt chẽ với phân tích tương quan kinh điển và làm cho việc sử dụng biến

đổi trực giao để chuyển đổi các thiết lập của các quan sát có chứa các biến tương quan vào một tập hợp các giá trị được gọi là thành phần chủ yếu. Số lượng các thành phần chủ yếu được sử dụng trong phép phân tích thành phần chính là nhỏ hơn hoặc bằng với số ít các quan sát. phép phân tích thành phần chính là nhạy cảm với các tỷ lệ tương đối của các biến ban đầu được sử dụng.

PCA là một phương pháp biến đổi làm giảm số lượng các biến tương quan thành một tập biến nhỏ sao cho các biến mới được tạo là tổ hợp tuyến tính của các biến cũ không tương quan. Ví dụ ta có 100 biến ban đầu tương quan tuyến tính với nhau, sau đó ta dùng phương pháp PCA để xoay không gian cũ sang không gian mới chỉ còn lại 5 biến không tương quan tuyến tính. Dữ liệu vẫn nhận được lượng thông tin nhiều nhất từ nhóm biến ban đầu.

3.1.2 Đặc tính PCA

Một số đặc điểm của PCA được đề cập như sau:

- Giúp giảm kích thước dữ liệu
- Giúp trực quan hóa khi dữ liệu có quá nhiều kích thước thông tin.
- Do dữ liệu gốc có số chiều lớn (nhiều biến) nên PCA giúp chúng ta xoay trục tọa độ để xây dựng trục tọa độ mới đảm bảo tính biến thiên của dữ liệu và giữ được nhiều thông tin nhất mà không ảnh hưởng đến chất lượng. số lượng các mô hình dự đoán. (Tối đa hóa độ biến thiên).
- Bởi vì PCA giúp tạo ra một hệ tọa độ mới, theo nghĩa toán học, PCA giúp chúng ta xây dựng các biến nhân tố mới là tổ hợp tuyến tính của các biến ban đầu.
- Trong không gian mới có thể giúp chúng ta khám phá những thông tin có giá trị mới khi trong chiều thông tin cũ thông tin có giá trị này bị che giấu.

3.1.3 Mô hình PCA

Xét một không gian (dữ liệu) gồm k biến, k biến này được biểu diễn qua j thành phần chính sao cho (j $<$ k). Hãy xem xét thành phần chính đầu tiên của biểu mẫu:

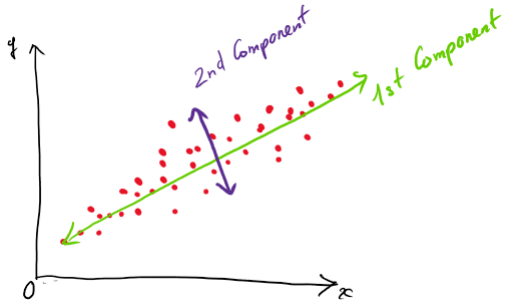
$$PC_1 = a_1X_1 + a_2X_3 + a_4X_5 + ...a_kX_k$$

Thành phần chính đầu tiên chứa hầu hết thông tin từ k biến ban đầu (được hình thành dưới dạng tổ hợp tuyến tính của các biến ban đầu) và bây giờ chúng ta tiếp tục xem xét thành phần chính thứ hai được biểu diễn tuyến tính từ k biến ban đầu tuy nhiên thành phần chính thứ 2 không được trực giao với thành phần chính ban đầu hoặc (thành phần chính thứ 2 không tương quan tuyến tính với thành phần chính đầu tiên). Về lý thuyết, chúng ta có thể xây dựng nhiều thành phần chính từ nhiều biến ban đầu. Tuy nhiên, chúng ta cần tìm trục không gian sao cho một vài thành phần có thể biểu diễn hầu hết thông tin từ các biến ban đầu.

3.1.4 Ví dụ PCA

Ví dụ dữ liệu của chính ta có N features thì sau khi áp dụng PCA sẽ còn K features chính mà thôi (K$<$N). Và tất nhiên K features này tất nhiên không được chọn ngẫu nhiên.

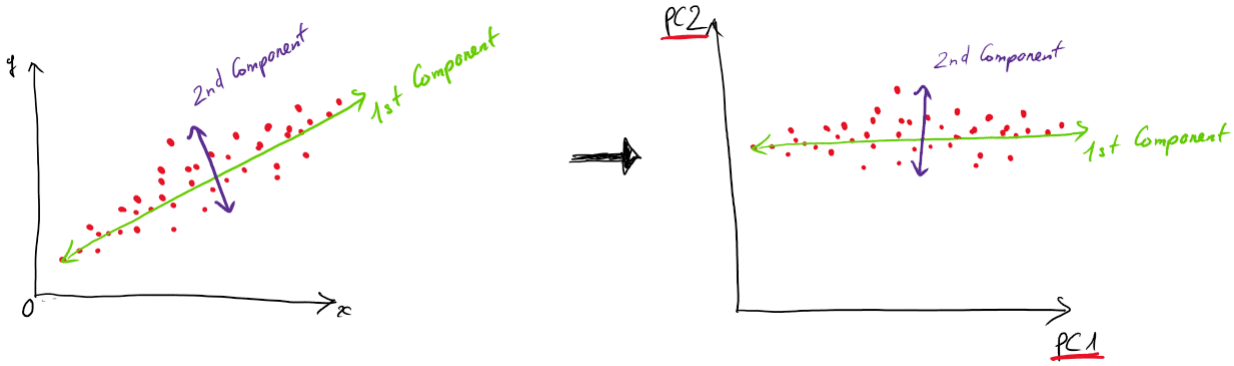
Các components ở đây ta nói thực chất là các vectors độc lập tuyến tính được chọn sao cho khi chiếu các điểm dữ liệu lên vector đó thì các điểm dữ liệu có sự variance lớn nhất (biến động nhiều nhất, phương sai lớn nhất).



Hình 3: *PCA*

Ví dụ như hình trên, chúng ta chọn 2 vector component theo thứ tự: 1st Comp sẽ có mức độ variance lớn nhất, ta chọn trước, sau đó đến 2nd Comp.... và cứ thế. Khi làm thực tế chúng ta sẽ cần xác định hoặc thử sai xem sẽ chọn bao nhiêu components là hợp lý và mang lại kết quả tốt nhất.

Xét một cách nhìn khác thì PCA cũng là một bài toán chuyển hệ tọa độ như hình dưới:



Hình 4: *PCA*

Cách đơn giản nhất để giảm chiều dữ liệu từ D về $K < D$ là chỉ giữ lại K phần tử *quan trọng nhất*. Tuy nhiên, việc làm này chắc chắn chưa phải tốt nhất vì chúng ta chưa biết xác định thành phần nào là quan trọng hơn. Hoặc trong trường hợp xấu nhất, lượng thông tin mà mỗi thành phần mang là như nhau, bỏ đi thành phần nào cũng dẫn đến việc mất một lượng thông tin lớn.

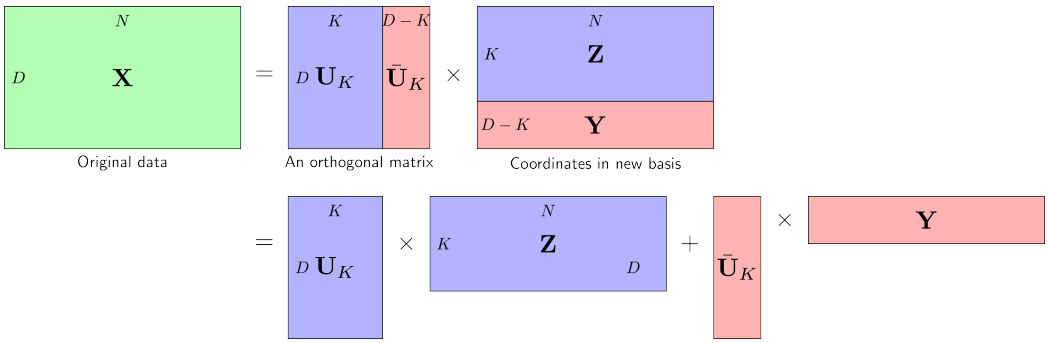
Tuy nhiên, nếu chúng ta có thể biểu diễn các vector dữ liệu ban đầu trong một hệ cơ sở mới mà trong hệ cơ sở mới đó, *tầm quan trọng* giữa các thành phần là khác nhau rõ rệt, thì chúng ta có thể bỏ qua những thành phần ít quan trọng nhất.

Lấy một ví dụ về việc có hai camera đặt dùng để chụp một con người, một camera đặt phía trước người và một camera đặt trên đầu. Rõ ràng là hình ảnh thu được từ camera đặt phía

trước người mang nhiều thông tin hơn so với hình ảnh nhìn từ phía trên đầu. Vì vậy, bức ảnh chụp từ phía trên đầu có thể được bỏ qua mà không có quá nhiều thông tin về hình dáng của người đó bị mất.

PCA chính là phương pháp đi tìm một hệ cơ sở mới sao cho thông tin của dữ liệu chủ yếu tập trung ở một vài toạ độ, phần còn lại chỉ mang một lượng nhỏ thông tin. Và để cho đơn giản trong tính toán, PCA sẽ tìm một hệ trực chuẩn để làm cơ sở mới.

Giả sử hệ cơ sở trực chuẩn mới là U và chúng ta muốn giữ lại K toạ độ trong hệ cơ sở mới này. Không mất tính tổng quát, giả sử đó là K thành phần đầu tiên. Quan sát hình dưới đây:



Hình 5: *Ý tưởng chính của PCA: Tìm một hệ trực chuẩn mới sao cho trong hệ này, các thành phần quan trọng nhất nằm trong K thành phần đầu tiên.*

Quan sát hình vẽ trên với cơ sở mới $U = [U_K, \bar{U}_K]$ là một hệ trực chuẩn với U_K là ma trận con tạo bởi K cột đầu tiên của U. Với cơ sở mới này, ma trận dữ liệu có thể được viết thành:

$$X = U_K Z + \bar{U}_K Y \qquad (1)$$

Từ đây ta cũng suy ra:

$$\begin{bmatrix} Z \\ Y \end{bmatrix} = \begin{bmatrix} U_K^T \\ \bar{U}_K^T \end{bmatrix} X \Rightarrow \begin{matrix} Z = U_K^T \\ Y = \bar{U}_K^T \end{matrix} \qquad (2)$$

Mục đích của PCA là đi tìm ma trận trực giao U sao cho phần lớn thông tin được giữ lại ở phần màu xanh $U_K Z$ và phần màu đỏ $\bar{U}_K Y$ sẽ được lược bỏ và thay bằng một ma trận không phụ thuộc vào từng điểm dữ liệu. Nói cách khác, ta sẽ xấp xỉ Y bởi một ma trận có toàn bộ các cột là như nhau. Chú ý rằng các cột này có thể phụ thuộc vào dữ liệu training nhưng không phụ thuộc vào dữ liệu test, các bạn sẽ thấy rõ hơn khi lập trình mà tôi sẽ trình bày trong bài tiếp theo. Gọi mỗi cột đó là b và có thể coi nó là bias, khi đó, ta sẽ xấp xỉ:

$$Y \approx b1^T$$

Trong đó $1^T \in \mathbb{R}^{1\times N}$ là vector hàng có toàn bộ các phần tử bằng 1. Giả sử đã tìm được U, ta cần tìm b thoả mãn:

$$b = argmin_b ||Y - b1^T||_F^2 = argmin_b ||\bar{U}_K^T X - b1^T||_F^2$$

Giải phương trình đạo hàm theo b của hàm mục tiêu bằng 0:

$$(b1^T - \bar{U}_K^T X)1 = 0 \Rightarrow Nb = \bar{U}_K^T X 1 \Rightarrow b = \bar{U}_K^T \bar{x}$$

Như vậy, **việc tính toán sẽ thuận tiện hơn nhiều nếu vector kỳ vọng** $\bar{x} = 0$. Việc này có thể đạt được nếu ngay từ đầu, chúng ta trừ mỗi vector dữ liệu đi vector kỳ vọng của toàn bộ dữ liệu. Đây chính là các bước đầu tiên của PCA.

Với giá trị b tìm được này, dữ liệu ban đầu sẽ được xấp xỉ với:

$$X \approx \tilde{X} = U_K Z + \bar{U}_K \bar{U}_K^T \bar{x} 1^T \qquad (3)$$

Kết hợp (1), (2), (3) ta định nghĩa hàm mất mát chính như sau:

$$J = \frac{1}{N}||X - \tilde{X}||_F^2 = \frac{1}{N}||\bar{U}_K \bar{U}_K^T X - \bar{U}_K \bar{U}_K^T \bar{x} 1^T||_F^2 \qquad (4)$$

Chú ý rằng, nếu các cột của một ma trận V tạo thành một hệ trực chuẩn thì với một ma trận W bất kỳ, ta luôn có:

$$||VW||_F^2 = trace(W^T V^T V W) = trace(W^T W) = ||W||_F^2$$

Vì vậy hàm mất mát trong (4) có thể viết lại thành:

$$\begin{aligned} J &= \frac{1}{N}||\bar{U}_K^T (X - \bar{x}1)^T||_F^2 = \frac{1}{N}||\bar{U}_K^T \hat{X}||_F^2 \\ &= \frac{1}{N}||\hat{X}^T \bar{U}_K||_F^2 = \frac{1}{N}\sum_{i=K+1}^{D} ||\hat{X}^T u_i||_2^2 \\ &= \frac{1}{N}\sum_{i=K+1}^{D} u_i^T \hat{X}\hat{X}^T u_i \\ &= \sum_{i=K+1}^{D} u_i^T S u_i \qquad (5) \end{aligned}$$

Với $\hat{X} = X - \bar{x}1^T$ là dữ liệu đã chuẩn hoá và với S là ma trận hiệp phương sai của dữ liệu. Ta gọi ma trận này $\hat{X}$ là *zero-corrected* data hoặc *dữ liệu đã được chuẩn hoá*. Có thể nhận thấy $\hat{X}_n = X_n - \bar{x}$.

Công việc còn lại là tìm các u_i để mất mát là nhỏ nhất. Trước hết, chúng ta có một nhận xét thú vị. Nhắc lại định nghĩa ma trận hiệp phương sai $S = \frac{1}{N}\hat{X}^T\hat{X}$. Với ma trận U trực giao bất kỳ, thay $K = 0$ vào (5) ta có:

$$\begin{aligned} L &= \sum_{i=1}^{D} u_i^T S u_i = \frac{1}{N}||\hat{X}^T U||_F^2 \\ &= \frac{1}{N} trace(\hat{X}^T U U^T \hat{X}) \qquad (6) \\ &= \frac{1}{N} trace(\hat{X}^T \hat{X}) \qquad (7) \\ &= \frac{1}{N} trace(\hat{X}\hat{X}^T) \qquad (8) \\ &= trace(S) = \sum_{i=1}^{D} \lambda_i \qquad (9) \end{aligned}$$

Với $\lambda_1 \geq \lambda_2 \geq ... \geq \lambda_D \geq 0$ là các trị riêng của ma trận nửa xác định dương S. Chú ý rằng các trị riêng này là thực và không âm.

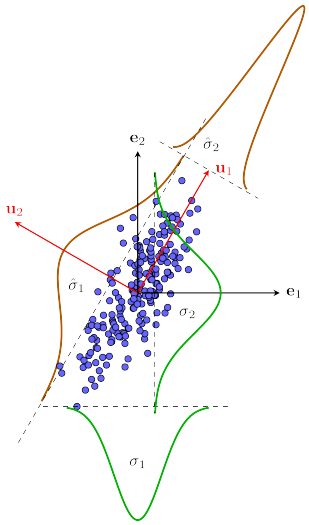
Như vậy L không phụ thuộc vào cách chọn ma trận trực giao U và bằng tổng các phần tử trên đường chéo của S. Nói cách khác, L chính là tổng của các phương sai theo từng thành phần của dữ liệu ban đầu.

Vì vậy, việc tối thiểu hàm mất mát J được cho bởi (7) tương đương với việc tối đa:

$$F = L - J = \sum_{i=1}^{K} \mathrm{u}_i S \mathrm{u}_i^T$$

Định lý 1: F đạt giá trị lớn nhất bằng $\sum_{i=1}^{K} \lambda_i$ khi u_i là các vector riêng có norm 2 bằng 1 ứng với các trị riêng này. Tất nhiên, chúng ta không quên điều kiện trực giao giữa các u_i.

Chú ý rằng $\lambda_i, i = 1, ..., K$ chính là K trị riêng lớn nhất của ma trận hiệp phương sai S. Trị riêng lớn nhất λ_1 của ma trận này còn được gọi là *Thành phần chính thứ nhất* (First Principal Component), trị riêng thứ hai λ_2 còn được gọi là *Thành phần chính thứ hai*, etc. Chính vì vậy, phương pháp này có tên gọi là *Phân tích thành phần chính* - Principal Component Analysis. Ta chỉ giữ lại K thành phần chính của dữ liệu khi muốn giảm số chiều dữ liệu. Để có cái nhìn trực quan hơn, chúng ta cùng theo dõi Hình dưới đây:



Hình 6: *PCA dưới góc nhìn Thống kê. PCA có thể được coi là phương pháp đi tìm một hệ cơ sở trực chuẩn đóng vai trò một phép xoay, sao cho trong hệ cơ sở mới này, phương sai theo một số chiều nào đó là rất nhỏ, và ta có thể bỏ qua.*

Trong không gian ban đầu với các vector cơ sở màu đen e_1, e_2, phương sai theo mỗi chiều dữ liệu đều lớn. Trong không gian mới với các vector cơ sở màu đỏ u_1, u_2, phương sai theo chiều thứ hai $\hat{\sigma}_2$ rất nhỏ so với $\hat{\sigma}_1$. Điều này nghĩa là khi chiếu dữ liệu lên u_2 ta được các điểm rất gần nhau và gần với kỳ vọng theo chiều đó. Trong trường hợp này, kỳ vọng theo mọi chiều bằng 0 nên ta có thể thay thế toạ độ theo chiều u_2 bằng 0. Rõ ràng là nếu dữ liệu có phương sai càng nhỏ theo một chiều nào đó thì khi xấp xỉ chiều đó bằng một hằng số, sai số xảy ra càng nhỏ. PCA thực chất là đi tìm một phép xoay tương ứng với một ma trận trực giao sao cho trong hệ toạ độ mới,

tồn tại các chiều có phương sai nhỏ mà ta có thể bỏ qua; ta chỉ cần giữ lại các chiều/thành phần khác quan trọng hơn. Như đã chứng minh ở trên, tổng phương sai theo mọi chiều trong hệ cơ sở nào cũng là như nhau và bằng tổng các trị riêng của ma trận hiệp phương sai. Vì vậy, PCA còn được coi là phương pháp giảm số chiều dữ liệu mà giữ được tổng phương sai còn lại là lớn nhất.

Tôi sẽ bỏ qua phần chứng minh của Định lý 1. Tuy nhiên, cũng nêu một vài ý để bạn đọc có thể hình dung:

Khi $K = 1$. Ta cần giải bài toán:

$$\max_{u_1} \quad u_1^T S u_1$$
$$s.t. : ||u_1||_2 = 1$$

Như đã đề cập ở phía trên, hàm mục tiêu đạt giá trị lớn nhất bằng λ_1 khi u_1 là một vector riêng của ma trận hiệp phương sai S tương ứng với trị riêng λ_1. Vậy định lý đúng với $K = 1$.

Giả sử u_1 đã là vector riêng ứng với trị riêng lớn nhất của S thế thì nghiệm u_2 của bài toán tối ưu:

$$\max_{u_2} \quad u_2^T S u_2$$
$$s.t. ||u_2||_2 = 1 \qquad (10)$$
$$u_2^T u_1 = 0$$

là một vector riêng của S ứng với trị riêng lớn thứ hai $_2$ của nó. Chú ý rằng $_2$ có thể bằng $_1$ nếu không gian riêng ứng với $_1$ có số rank lớn hơn 1.

Nhận định này có thể được chứng minh bằng phương pháp nhân tử Lagrange. Thật vậy, Lagrangian của bài toán (10) là:

$$\mathcal{L}(u_2, \mathcal{V}_1, \mathcal{V}_2) = u_2^T S u_2 + \mathcal{V}_1 u_1^T u_2 + \mathcal{V}_2(1 - u_2^T u_2)$$

Ta cần giải hệ phương trình đạo hàm của $\mathcal{L}$ theo từng biến bằng 0:

$$\frac{\partial \mathcal{L}}{\partial u_2} = 2Su_2 + \mathcal{V}_1 u_1 - 2\mathcal{V}_2 u_2 = 0 \qquad (11)$$
$$\frac{\partial \mathcal{L}}{\partial \mathcal{V}_1} = u_1^T u_2 = 0 \qquad (12)$$
$$\frac{\partial \mathcal{L}}{\partial \mathcal{V}_2} = 1 - u_2^T u_2 = 0 \qquad (13)$$

Nhân cả hai vế của (11) với u_1^T vào bên trái ta có:

$$2u_1^T S u_2 + \mathcal{V}_1 = 0$$

Vì $Su_1 = \lambda_1 u_1 \Rightarrow u_1^T S u_2 = \lambda_1 u_1^T u_2 = 0$. Từ đó suy ra $\mathcal{V}_1 = 0$ và (11) lúc này tương đương với:

$$Su_2 = \mathcal{V}_2 u_2 \Rightarrow u_2^T S u_2 = \mathcal{V}_2$$

Vậy u_2 là một vector riêng của S ứng với $\mathcal{V}_2$. Và để hàm mục tiêu đạt giá trị lớn nhất, $\mathcal{V}_2$ cần càng lớn càng tốt. Điều này dẫn đến $\mathcal{V}_2$ phải là trị riêng thứ hai của S.

Lập luận tương tự, ta có thể chứng minh được: Nếu $\mathrm{u}_i, i = 1, 2, ..., k-1$ là các vector riêng ứng với trị riêng lớn thứ i của ma trận nửa xác định dương S, hơn nữa, $k-1$ vector riêng này tạo thành một hệ trực chuẩn, thế thì:

$$\begin{aligned} \underset{\mathrm{u}_k}{max} \quad & \mathrm{u}_k^T S \mathrm{u}_k \\ s.t. \quad & \mathrm{u}_k^T \mathrm{u}_k = 1; \\ & \mathrm{u}_k^T \mathrm{u}_i = 1, i = 1, ..., k-1 \end{aligned}$$

bằng đúng với trị riêng tiếp theo λ_k tại u_k là vector riêng ứng với trị riêng này.

3.1.5 Các bước thực hiện PCA

Từ các suy luận phía trên, ta có thể tóm tắt lại các bước trong PCA như sau:

1. Tính vector kỳ vọng của toàn bộ dữ liệu:

$$\bar{\mathrm{x}} = \frac{1}{N} \sum_{n=1}^{N} X_n$$

2. Trừ mỗi điểm dữ liệu đi vector kỳ vọng của toàn bộ dữ liệu:

$$\hat{X}_n = X_n - \bar{\mathrm{x}}$$

3. Tính ma trận hiệp phương sai:

$$S = \frac{1}{N} \hat{X} \hat{X}^T$$

4. Tính các trị riêng và vector riêng có norm bằng 1 của ma trận này, sắp xếp chúng theo thứ tự giảm dần của trị riêng.

5. Chọn K vector riêng ứng với K trị riêng lớn nhất để xây dựng ma trận U_K có các cột tạo thành một hệ trực giao. K vectors này, còn được gọi là các thành phần chính, tạo thành một không gian con gần với phân bố của dữ liệu ban đầu đã chuẩn hoá.

6. Chiếu dữ liệu ban đầu đã chuẩn hoá $\hat{X}$ xuống không gian con tìm được.

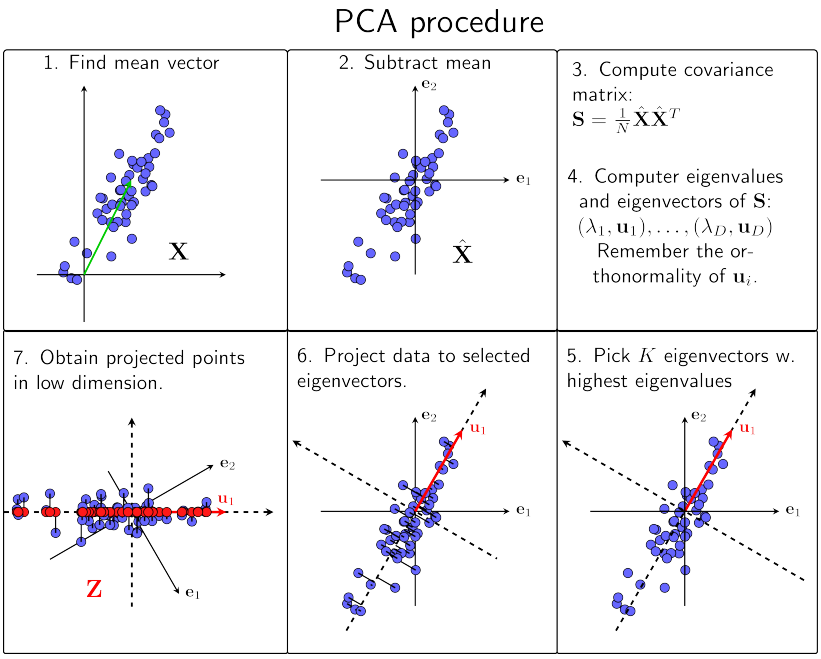
7. Dữ liệu mới chính là toạ độ của các điểm dữ liệu trên không gian mới

$$Z = U_K^T \hat{X}$$

Dữ liệu ban đầu có thể tính được xấp xỉ theo dữ liệu mới như sau:

$$X \approx U_K Z + \bar{\mathrm{x}}$$

Các bước thực hiện PCA có thể được xem trong Hình dưới đây:



Hình 7: *Các bước thực hiện PCA.*

3.2 Giải thuật SVD

3.2.1 Giới thiệu

Nhắc lại bài toán "chéo hóa" đã từng học ở môn Đại số tuyến tính: Bài toán chéo hoá ma trận. Bài toán đó nói rằng: Một ma trận vuông $A \in \mathbb{R}^n$ được gọi là chéo hoá được (diagonalizable) nếu tồn tại ma trận đường chéo D và ma trận khả nghịch P sao cho:

$$A = PDP^{-1} \tag{1}$$

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:

$$AP = PD \tag{2}$$

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:

$$Ap_i = Pd_i = d_{ii}p_i \tag{3}$$

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. Trong đó biểu thức (3) chỉ ra rằng mỗi phần tử d_{ii} phải là một trị riêng (eigenvalue) của A và mỗi vector cột pi 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 (2) còn được gọi là Eigen Decomposition.

Một điểm quan trọng là cách phân tích như (2) 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 thoả mãn (2) thì kP, D cũng thoả 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 dễ 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.

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. Dưới đây, sẽ là phát biểu SVD cũng như các tính chất và ứng dụng điển hình của nó.

3.2.2 Nhắc lại về đại số tuyến tính

3.2.2.a Trị riêng và vector riêng

Cho một ma trận vuông $A \in \mathbb{R}^n$, nếu số vô hướng λ và vector $x \neq 0 \in \mathbb{R}^n$ thõa 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 Axác 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 $\|u\|_2^2 \geq 0$. Từ đó suy ra λ là một số không âm.

3.2.2.b Hệ trực giao và trực chuẩn

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

$$u_i^T u_j = 1$$

với i = j

$$u_i^T u_j = 0$$

với các trường hợp còn lại

Gọi $U = [u_1, u_2, \ldots, u_m]$ với $u_1, u_2, \ldots, u_m \in \mathbb{R}^m$ là trực chuẩn, thế thì từ trên có thể suy ra ngay:

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

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 được gọi là ma trận trực chuẩn, không có định nghĩa cho ma trận trực chuẩn.
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 (determinant) của ma trận trực giao bằng 1 hoặc 1. Điều này có thể suy ra từ việc $det(U) = det(UT)$ và $det(U)det(UT) = det(I) = 1$.

4.Ma trận trực giao thể hiện cho phép xoay (rotate) một vector. Giả sử có hai vector $x, y \in \mathbb{R}^m$ và ma trận trực giao $U \in \mathbb{R}^{m \times m}$. Dùng ma trận này để xoay hai vector trên ta được Ux, Uy. Tích vô hướng của hai vector mới là:

$$(Ux)^T(Uy) = x^T U^T U y = x^T y$$

như vậy phép xoay không làm thay đổi tích vô hướng giữa hai vector.
5.Giả sử $\hat{U} \in \mathbb{R}^{m \times r}, r \leq m$ là môt ma trận con của ma trận trực giao U được tạo bởi r cột của U, ta sẽ có $\hat{U}^T \hat{U} = I_r$. Việc này có thể được suy ra từ biểu thức phía trên.

3.2.3 Singular Value Decomposition (SVD)

Vì trong mục này cần nắm vững chiều của mỗi ma trận nên tôi sẽ thay đổi ký hiệu một chút để chúng ta dễ hình dung. Ta sẽ ký hiệu một ma trận cùng với số chiều của nó, ví dụ $A^{m \times n}$ nghĩa là ma trận$A \in \mathbb{R}^{m \times n}$.

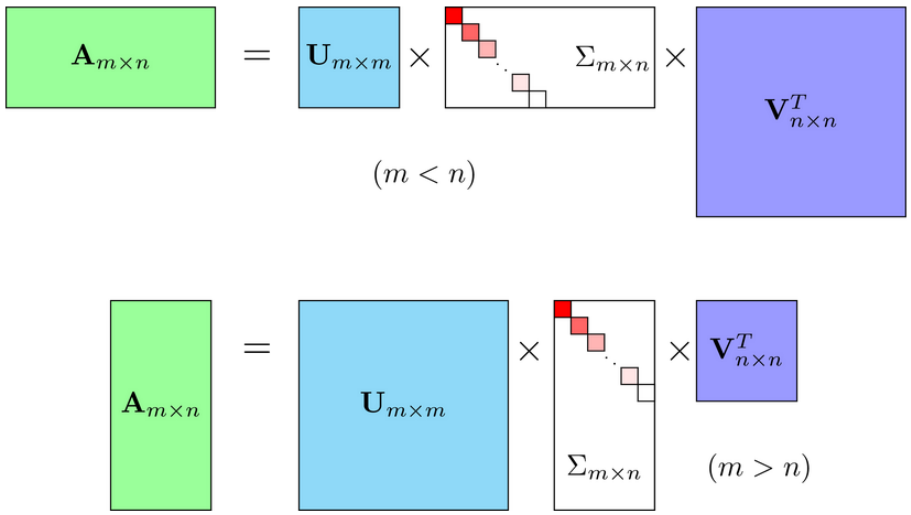
3.2.3.a Phát biểu SVD

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

$$A_{m \times n} = U_{m \times m} \Sigma_{m \times n} (V_{n \times n})^T \tag{4}$$

Trong đó, U, V là các ma trận trực giao, Σ là ma trận đường chéo không vuông với các phần tử trên đường chéo$\sigma_1 \geq \sigma_2 \geq ... \geq \sigma_r \geq 0 = 0 = ... = 0$ và r là rank của ma trận 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 A.

Chú ý rằng cách biểu diễn (4) không là duy nhất vì ta chỉ cần đổi dấu của cả U và V thì (4) vẫn thoả mãn. Do vậy, người ta vẫn thường dùng 'the SVD' thay vì 'a SVD'.
Hình dưới đây mô tả SVD của ma trận Am×n trong hai trường hợp:$m < n$ và $m > n$. Trường hợp $m = n$ có thể xếp vào một trong hai trường hợp trên.



Hình 8: *SVD cho ma trận A khi: m<n (hình trên), và m>n (hình dưới). Σ là một ma trận đường chéo với các phần tử trên đó giảm dần và không âm. Màu đỏ càng đậm thể hiện giá trị càng cao. Các ô màu trắng trên ma trận này thể hiện giá trị 0.*

3.2.3.b Compact SVD

Viết lại biểu thức (4) dưới dạng tổng của các ma trận rank 1:

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

với chú ý rằng mỗi $u_1 v_1^T, 1 \leq i \leq r$ là một ma trận có rank bằng 1.
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 của ma trận Σ. Vì vậy ta có một cách phân tích gọn hơn và gọi là compact SVD:

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

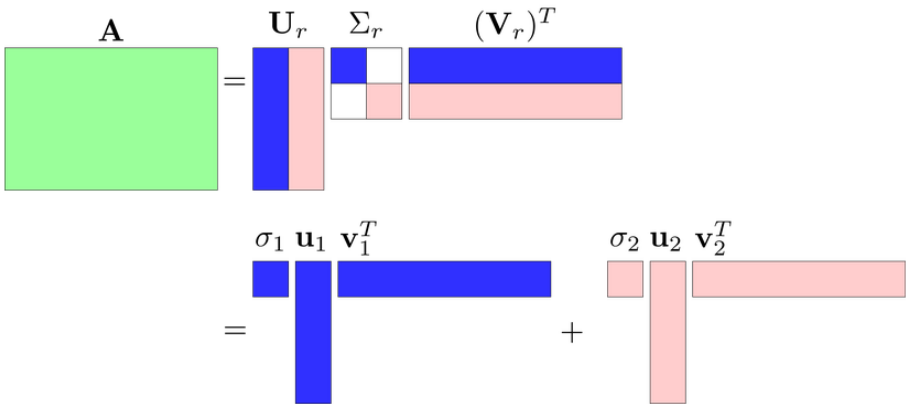
Với U_r, V_r lần lượt là ma trận được tạo bởi r cột đầu tiên của U và V. Σ_r là ma trận con được tạo bởi r hàng đầu tiên và r cột đầu tiên của . Nếu ma trận A có rank nhỏ hơn rất nhiều so với số hàng và số cột $r \ll m, n$, ta sẽ được lợi nhiều về việc lưu trữ.
Dưới đây là ví dụ minh hoạ với $m = 4,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

3.2.3.c Truncated SVD

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

$$A \approx A_k = U_k \Sigma_k (V_k)^T = \sigma_1 u_1 v_1^T + \sigma_2 u_2 v_2^T + ... + \sigma_k u_k v_k^T \quad (5)$$

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 đã bỏ qua ở phần cuối của Σ. Ở đây sai số được định nghĩa là Frobineous norm của hiệu hai ma trận:



Hình 9: *Biểu diễn SVD dạng thu gọn và biểu diễn ma trận dưới dạng tổng các ma trận có rank bằng 1.*

Định lý:

$$\|A - A_k\|_F^2 = \sum_{i=k+1}^{r} \sigma_i^2 \tag{6}$$

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

$$\begin{aligned}
\|\mathbf{A} - \mathbf{A}_k\|_F^2 &= \left\| \sum_{i=k+1}^{r} \sigma_i \mathbf{u}_i \mathbf{v}_i^T \right\|_F^2 \\
&= \text{trace}\left\{ \left(\sum_{i=k+1}^{r} \sigma_i \mathbf{u}_i \mathbf{v}_i^T \right) \left(\sum_{j=k+1}^{r} \sigma_j \mathbf{u}_j \mathbf{v}_j^T \right)^T \right\} \\
&= \text{trace}\left\{ \sum_{i=k+1}^{r} \sum_{j=k+1}^{r} \sigma_i \sigma_j \mathbf{u}_i \mathbf{v}_i^T \mathbf{v}_j \mathbf{u}_j^T \right\} \\
&= \text{trace}\left\{ \sum_{i=k+1}^{r} \sigma_i^2 \mathbf{u}_i \mathbf{u}_i^T \right\} \\
&= \text{trace}\left\{ \sum_{i=k+1}^{r} \sigma_i^2 \mathbf{u}_i^T \mathbf{u}_i \right\} \\
&= \text{trace}\left\{ \sum_{i=k+1}^{r} \sigma_i^2 \right\} \\
&= \sum_{i=k+1}^{r} \sigma_i^2
\end{aligned}$$

Thay $k = 0$ ta sẽ có:

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

Từ đó:

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

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% 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.

3.2.3.d Best Rank k Approximation

Người ta chứng minh được rằng (Singular Value Decomposition - Princeton) Ak chính là nghiệm của bài toán tối ưu:

$$min_B \|A - B\|_F$$

$$s.t. rank(B) = k \tag{7}$$

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

$$min_B \|A - B\|_2$$

$$s.t. rank(B) = k \tag{8}$$

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

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

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.
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.

3.3 K means

3.3.1 Tổng quan về K-means

Thuật toán k-means sử dụng phương pháp tạo và cập nhật trung tâm để phân nhóm các điểm dữ liệu cho trước vào các nhóm khác nhau. Đầu tiên chúng sẽ tạo ra các điểm trung tâm ngẫu nhiên. Sau đó gán mỗi điểm trong tập dữ liệu vào trung tâm gần nó nhất. Sau đó chúng sẽ cập nhật lại trung tâm và tiếp tục lặp lại các bước đã kể trên. Điều kiện dừng của thuật toán: Khi các trung tâm không thay đổi trong 2 vòng lặp kế tiếp nhau. Tuy nhiên, việc đạt được 1 kết quả hoàn hảo là rất khó và rất tốn thời gian, vậy nên thường người ta sẽ cho dừng thuật toán khi đạt được 1 kết quả gần đúng và chấp nhận được

3.3.2 Nội dung thuật toán

Input :N đối tượng cần được phân cụm $\{x_1, x_2...x_n\}$ và số lượng cụm k
Thuật toán K-means có thể được chia thành các bước như sau:

3.3.3 Tạo các trung tâm ngẫu nhiên

Tùy ý chọn K node đầu tiên làm tủng tâm ban đầu của cụm

$$C^0 = (m_1^0, m_2^0, m_3^0, ...m_k^0) \tag{9}$$

3.3.4 Gán các điểm dữ liệu vào các cụm

Với mỗi điểm dữ liệu, ta sẽ tính khoảng cách của nó tới các trung tâm (bằng Khoảng cách Euclid). Ta sẽ gán chúng vào trung tâm gần nhất. Tập hợp các điểm được gán vào cùng 1 trung tâm sẽ tạo thành cụm

$$d(x_i, m_i) = \sqrt{(\sum_{i=1}^{n}(x_{i1} - m_{j1})^2)} \tag{10}$$

Với i = 1,2,...,N và j = 1,2,..,N
d chính là khoảng cách giữa data i và nhóm j

3.3.5 Cập nhật trung tâm

Với mỗi cụm đã tìm được ở bước 2, trung tâm mới sẽ là trung bình cộng của các điểm dữ liệu trong cụm đó.

$$m_i = \frac{1}{N_i} * \sum_{j=1}^{N_i}(x_{ij}) \tag{11}$$

Trong đó Ni là số lượng data node có trong cụm đó
Thuật toán sẽ lặp lại các bước trên cho tới khi đạt được kết quả chấp nhận được (Square error

criterion).

$$E = \sum_{i=1}^{k} \sum_{j=1}^{n_i} |x_{ij} - m_i|^2 \tag{12}$$

Trong đó:

- x là data node j nằm trong cụm i
- m_i là trung tâm của cụm i
- n_i là số lượng data node nằm trong cụm i

4 Thực nghiệm

Trong nội dung bài báo cáo này, nhóm sẽ sử dụng tập dữ liệu trên Kaggle về thông tin của khách hàng. Tập dữ liệu có thể được tìm thấy tại Kaggle (Mall Customer Segmentation Data).

4.1 Khám phá dữ liệu

Chúng ta sẽ sử dụng thư viện Pandas để đọc file dữ liệu, sau đó mô tả dữ liệu của 5 dòng đầu tiên.

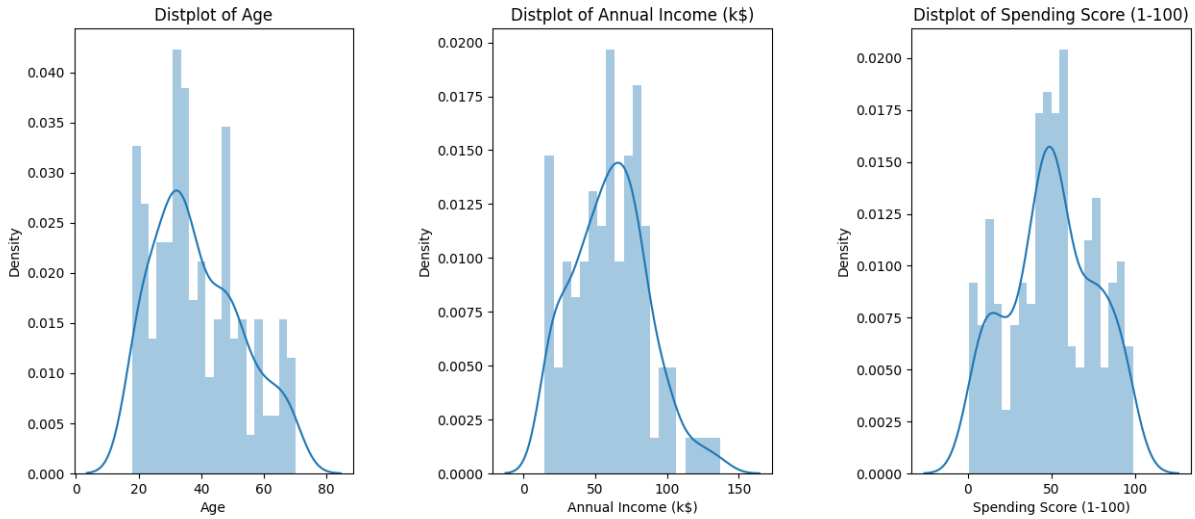
```
RangeIndex: 200 entries, 0 to 199
Data columns (total 5 columns):
 #   Column                  Non-Null Count  Dtype
---  ------                  --------------  -----
 0   CustomerID              200 non-null    int64
 1   Gender                  200 non-null    object
 2   Age                     200 non-null    int64
 3   Annual Income (k$)      200 non-null    int64
 4   Spending Score (1-100)  200 non-null    int64
dtypes: int64(4), object(1)
memory usage: 7.9+ KB
```

Tập dữ liệu gồm 5 cột: CustomerID, Gender, Age, Annual Income và Spending Score. Tổng số hàng trong tập dữ liệu là 200.

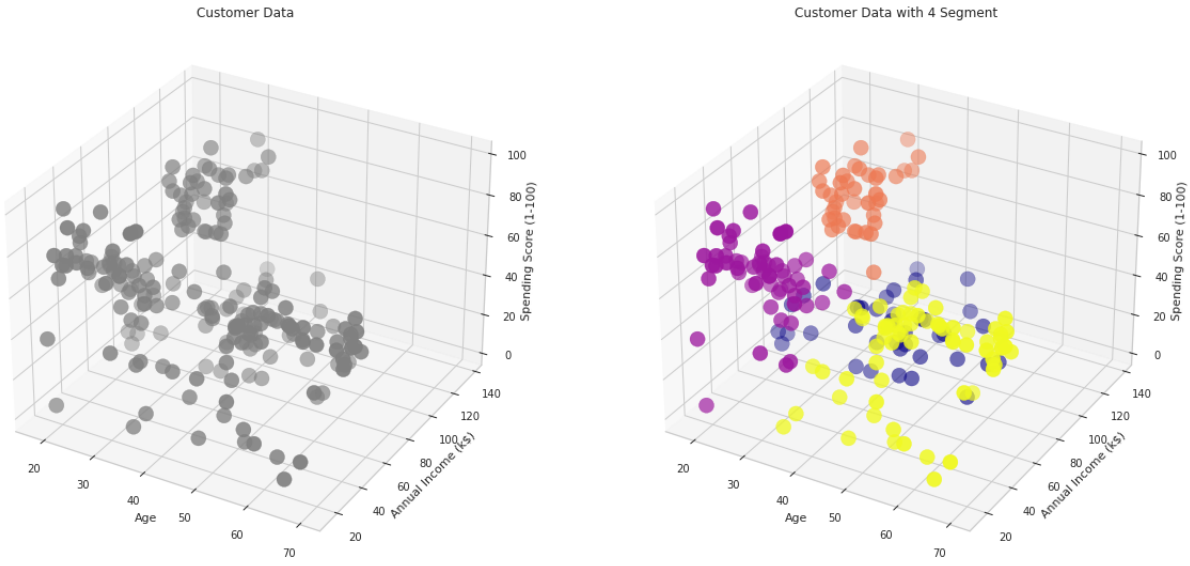
Từ biểu đồ Historical data ta có thể thấy một số thông tin:

- Độ tuổi trải dài từ 20 tới 70 tuổi
- Thu nhập giao động từ 10 đến 140 ngàn $ trên một năm.
- Spending Score từ 0 tới 100.

Mô phỏng tập dữ liệu trên không gian 3 chiều:



Hình 10: *Biểu đồ Historical Data của 3 cột: Age, Income và Spending Score*



Hình 11: *Biểu đồ tập dữ liệu trên không gian nhiều chiều.*

4.2 Áp dụng giải thuật PCA để tìm ra số lượng chiều cần giữ lại cho bài toán

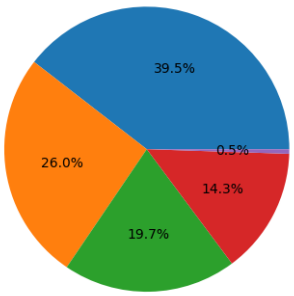
Biểu đồ mô phỏng tập dữ liệu ở trên ta chỉ mô phỏng 3 chiều của tập dữ liệu, may mắn là tập dữ liệu của chúng ta chỉ có 5 chiều dữ liệu. Hãy thử tưởng tượng xem nếu chúng ta phải xử lý bài toán liên quan đến dữ liệu nhiều chiều, tập dữ liệu lớn. Vì vậy việc giảm chiều dữ liệu để có thể dễ dàng mô phỏng trên các đồ thị mà con người có thể nhìn thấy được là một việc hết sức

cần thiết.
Với ý tưởng đó, chúng ta sẽ sử dụng giải thuật PCA để phân tích thành phần dữ liệu của tập dữ liệu. Từ đó xác định xem hệ cơ sở mới bằng cách áp dụng PCA.
Thư viện **scikit-learn** của Python cung cấp cho chúng ta các mô hình PCA có sẵn, chúng ta chỉ cần truyền dữ liệu vào mô hình. Đây là quá trình thực hiện.

```
# Apply principal Component Analysis
pca = PCA()

# fit PCA
pca.fit(data_scaled)
# get PCA features
features = range(pca.n_components_)

# data PCA
data_pca = pca.transform(data_scaled)
variance_ratio = pca.explained_variance_ratio_
```



Hình 12: *Số lượng các đặc trưng khi áp dụng PCA.*

Từ đồ thị trên có thể thấy, từ tập dữ liệu ban đầu nếu như chuyển nó về hệ cơ sở mới 2 chiều thì có thể giữ được **66%** đặc trưng của dữ liệu, nếu giữ lại 3 chiều thì số lượng đặc trưng có thể giữ lại lên tới **86%**.

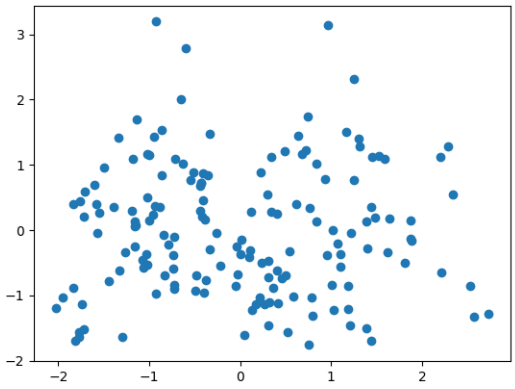
Với tập dữ liệu khi thực nghiệp của chúng ta khá nhỏ, và để trực quan hoá chúng ta sẽ thử giữ lại 2 chiều dữ liệu. Đây là quá trình thực hiện.

```
new_pca = PCA(n_components=2, svd_solver='full')
# fit new pca model
new_pca.fit(data_scaled)

# Transform new PCA
data_scaled_new = new_pca.transform(data_scaled)
print(data_scaled_new.shape)
```

4.3 Bài toán phân cụm cho tập dữ liệu trong hệ cơ sở mới sử dụng KMeans.

KMeans là thuật toán khá đơn giản nhưng lại hiệu quả với các bài toán phân cụm dữ liệu. Tuy nhiên với các bài toán có lượng dữ liệu lớn thì KMeans lại có tốc độ khá chậm. Vì thế việc giảm



Hình 13: *Mô phỏng tập dữ liệu mới trên không gian 2 chiều.*

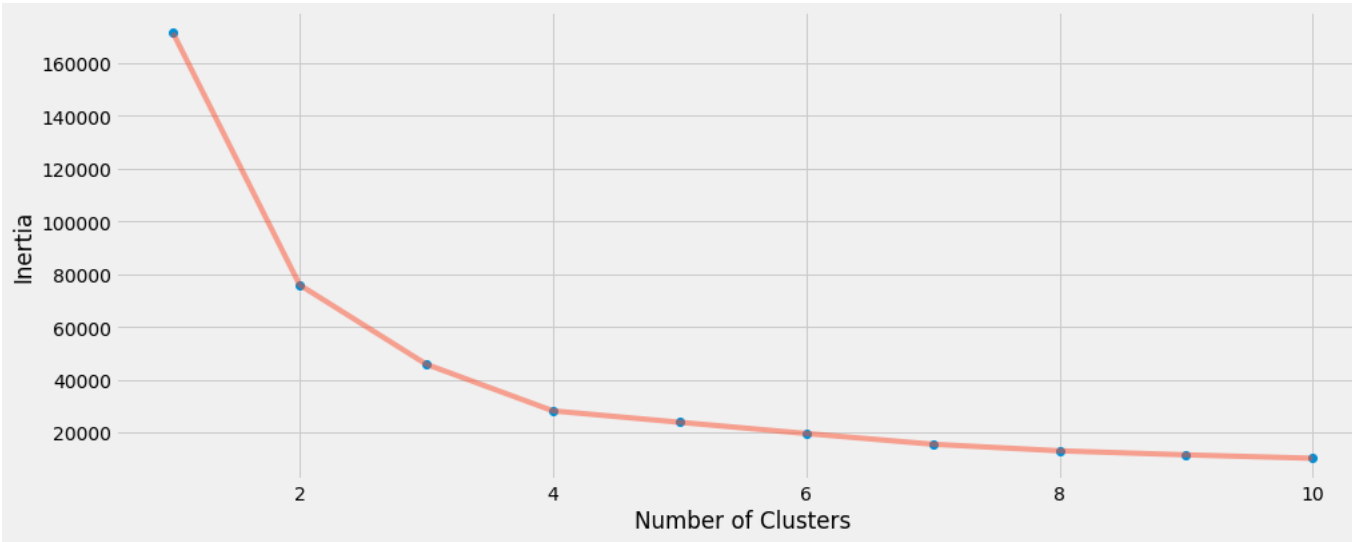
chiều dữ liệu bằng PCA là một cách thức hay để tận dụng giải thuật KMeans.

Giải thuật Kmeans trong bài thực nghiệm này sẽ gồm 2 bước:

- Xác định số luọng cụm cần chia cho tập dữ liệu
- Thực hiện việc phân cụm

4.3.1 Xác định cụm dữ liệu cần chia

Chọn cụm dữ liệu cần chia bằng *Squared Distance between Centroids and data points.*



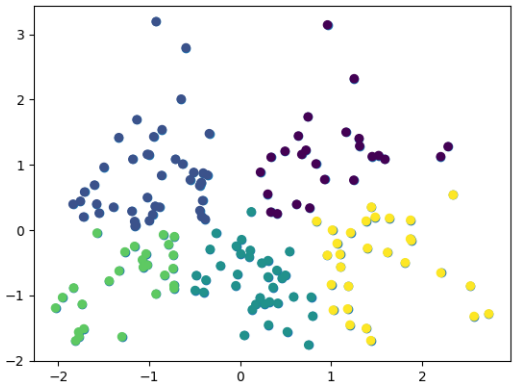
Hình 14: *Xác định số cụm dữ liệu cần chia.*

Có thể dễ dàng thấy, số lượng cụm chúng ta nên chọn trong tập dữ liệu là 5.

4.3.2 Phân cụm

```
# K-Means Clustering
kmeans = KMeans(n_clusters=5, init='k-means++', random_state=0)
```

Dữ liệu sau khi được phân cụm.

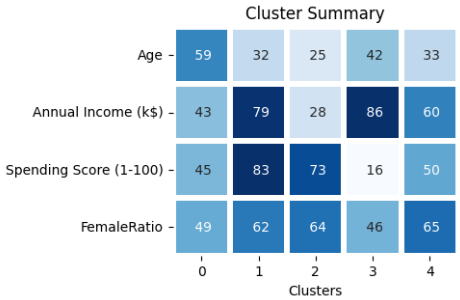


Hình 15: *Dữ liệu sau khi đã được phân cụm.*

Sau khi đã lựa chọn số lượng cụm cần thiết để phân loại, chúng ta sẽ đi đánh giá mô hình, áp dụng quy trình phân loại như hình dưới đây:

Hình 16: *Quá trình đánh giá mô hình*

Đánh giá mô hình:



Hình 17: *Biểu đồ Heatmap trên tập Train*

5 Kết luận

Bằng việc kết hợp giữa kỹ thuật giảm chiều số liệu sử dụng **Principal Component Analysis** kết hợp với thuật toán phân cụm **KMeans**, ta có thể giảm thiểu được số chiều, giữ được một lượng lớn đặc trưng của dữ liệu, từ đó ta có thể khai phá được các đặc trưng ẩn (hidden features) của dữ liệu, giúp cho các bước huấn luyện và đánh giá phía sau trở nên hiệu quả và chính xác hơn.

6 Tài liệu tham khảo

Tài liệu

[1] Youguo Li, Haiyan Wu (2012)."A Clustering Method Based on K-Means Algorithm" International Conference on Solid State Devices and Materials Science, Physics Procedia 25 (2012) 1104 – 1109

[2] Richard A. Johnson, Dean W. Wichern. "Applied Multivariate Statistical Analysis (Classic Version)".

[3] Tim Roughgarden Gregory Valiant. "The Modern Algorithmic Toolbox" .The Singular Value Decomposition (SVD) and Low-Rank Matrix Approximations.

[4] UCLA Mathematics. "The Moore-Penrose Pseudoinverse".

[5] Princeton University. "Singular Value Decomposition".

[6] Đặng Tuấn Vũ, "PCA - Principal Components Analysis", 12 tháng 03, 2017; https://rpubs.com/vudt1993/257891

[7] Nguyễn Chiến Thắng, "Principal Component Analysis (PCA) – tuyệt chiêu giảm chiều dữ liệu", 22 tháng 04; 2021, https://miai.vn/2021/04/22/principal-component-analysis-pca-tuyet-chieu-giam-chieu-du-lieu/

[8] "Principal Component Analysis (PCA)", https://filegi.com/tech-term/principal-component-analysis-pca-10147/

[9] Nhóm nghiên cứu vnquants, "Giới thiệu về phương pháp Principal Component Analysis (PCA) và một số ứng dụng trong tài chính (Phần I)", 25 tháng 12, 2013; https://tuanvanle.wordpress.com/2013/12/25/gioi-thieu-ve-phuong-phap-principal-component-analysis-pca-va-mot-so-ung-dung-trong-tai-chinh-phan-i/