BÁO CÁO MÔN TOÁN CHO TRÍ TUỆ NHÂN TẠO

- ỨNG DỤNG KỸ THUẬT SVD CHO HỆ THỐNG GỢI Ý SẢN PHẨM CHO NGƯỜI DÙNG –

Đại học Sư phạm Kỹ thuật Thành phố Hồ Chí Minh

- Khoa CNTT -

Sinh viên thực hiện: Nguyễn Trung Hiếu

MSSV: 22110138

Email: hieuanthonydisward@gmail.com

MỤC LỤC

I) Lý do chọn đề tài 3
II. Lý thuyết 3
1.Khái niệm 3
1.1 Full SVD 3
1.2 Reduced SVD/ Compact SVD 4
1.3 Semi-Orthonormal Matrices(Ma trận bán trực chuẩn) 4
2. Khái niệm User-Based Collaborative Filtering 4
3. Ứng dụng SVD cho hệ thống gợi ý sản phẩm cho người dùng(User base) 7
4. Các phương pháp chọn giá trị k cho Compact SVD. 12
4.1 Phương pháp tích lũy(Explained Variance) 12
4.2 Phương pháp khuỷu tay (Elbow Method) 12
4.3 Phương pháp dựa trên kinh nghiệm(Heuristic) 13
III) Dữ liệu và mô hình SVD 14
1. Các cột thuộc tính 14
2. Hệ thống gợi ý sản phẩm cho người dùng sử dụng SVD(SVD Recommendation System User Base) 16
IV) Kết luận 23
TÀI LIỆU THAM KHẢO 25

I) Lý do chọn đề tài

Trên thực tế, các hệ thống gợi ý đóng vai trò quan trọng trong việc cá nhân hóa trải nghiệm người dùng, là một bước chiến lược marketing quan trọng để giới thiệu người dùng với các sản phẩm khác một cách nhanh chóng. Các nền tảng lớn như Netflix, Spotify, Amazon hay Shopee đều sử dụng hệ thống gợi ý để nâng cao sự tương tác và trải nghiệm của người dùng.

Tuy nhiên dữ liệu đánh giá thường rất thưa, tức là có rất nhiều sản phẩm chưa được người dùng đánh giá, nên sẽ làm cho các phương pháp truyền thống như Collaborative Filtering gặp khó khăn trong việc dự đoán và gợi ý.

Tuy vậy, kĩ thuật SVD là một kĩ thuật phân rã ma trận có rất nhiều ứng dụng. Một số ứng dụng chính của SVD có thể kể đến như nén ảnh, xử lý ngôn ngữ tự nhiên, phát hiện bất thường, đặc biệt là hệ thống gợi ý cho người dùng. SVD có thể được áp dụng cho hệ thống gợi ý sản phẩm, phim, bài hát hay khóa học cho từng người dùng khác nhau.

Đề tài sẽ tập trung nghiên cứu về ứng dụng của SVD với hệ thống gợi ý sản phẩm cho người dùng, từ cách xây dựng ma trận người dùng – sản phẩm, đến phân rã ma trận cho đến xác định những người dùng khác có cùng sở thích để rồi đưa ra gợi ý.

II. Lý thuyết

1.Khái niệm

SVD(Singular Value Decomposition) là kỹ thuật phân tách một ma trận thành tích của các ma trận nhỏ hơn.

1.1 Full SVD

Giả sử ta có ma trận A(kích thước mxn) thì sẽ được phân tách thành:

$$A = U * \Sigma * \mathrm{V}^{\mathrm{T}}$$

U: Ma trận trực chuẩn chứa các vecto kì dị bên trái(Kích thước m x m).

Σ : Ma trận đường chéo chứa các giá trị kì dị(Kích thước m x n).

V: Ma trận trực chuẩn chứa các giá trị kì dị bên phải(Kích thước n x n).

- Các cột của U và V đều trực giao với nhau và là vector đơn vị.

- Các vector trực giao với nhau nếu tích chấm của 2 vecto bất kì bằng không. Một vector là vector đơn vị nếu độ dài của chúng bằng 1.

- Chuyển vị của ma trận trực chuẩn thì là nghịch đảo của nó. Như công thức trên ta có $U^T = U^{-1}$ hay $U * U^T = U^T * U = I$ và $V^T * V = V * V^T = I$.

1.2 Reduced SVD/ Compact SVD

- Đây là dạng phân rã kì dị rút gọn, trong Reduced SVD, ta chỉ giữ lại các giá trị kỳ dị khác 0 và bỏ các cột dư thừa của U và V. Mục đích là giúp giảm kích thước và tối ưu hóa lưu trữ.

Công thức:

$$M = U_r * \Sigma_r * V_r^T$$

Σ_r : Là ma trận đường chéo kích thước rxr(r là rank của ma trận M).

U_r: có kích thước m x r(thay vì m x m như Full SVD).

V_r^T: có kích thước r x n(thay vì n x n).

1.3 Semi-Orthonormal Matrices(Ma trận bán trực chuẩn)

Trong Full SVD, U và V là ma trận trực chuẩn, tức là:

$U^T = U^{-1}$ hay $U * U^T = U^T * U = I$ và $V^T * V = V * V^T = I$.

Trong Reduced SVD, U_r và V_r không vuông nên chỉ thỏa mãn:

$$U_r^T * U_r = I, V_r^T * V_r = I.$$

Tuy nhiên $U_r * U_r^T \neq I$ và $V_r * V_r^T \neq I$.

Như vậy, U_r và V_r được gọi là Semi-Orthonormal Matrices(Ma trận bán trực chuẩn).

2. Khái niệm User-Based Collaborative Filtering

Là một kĩ thuật để dự đoán những sản phẩm mà người dùng có thể thích dựa vào điểm số được đánh giá bởi những người dùng khác có cùng sở thích.

Bước 1: Tìm độ tương đồng giữa các người dùng tới người dùng mục tiêu U.

Độ tương đồng giữa 2 người dùng a và b được tính theo công thức:

$$Sim(a,b) = \frac{\sum_p (r_{ap} - \overline{r_a})(r_{pb} - \overline{r_b})}{\sqrt{\sum (r_{ap} - \overline{r_a})^2} * \sqrt{\sum (r_{bp} - \overline{r_b})^2}}$$

r_{ap}: Điểm đánh giá của người dùng a cho sản phẩm p

r_{bp}: Điểm đánh giá của người dùng b cho sản phẩm p

$\overline{r_a}$: điểm đánh giá trung bình của người dùng a

$$\overline{r_a} = \frac{1}{|P_a|} * \sum_{p \in P_a} r_{ap}$$

P_a: Tập hợp các mục mà người dùng a đã đánh giá

$\overline{r_b}$: điểm đánh giá trung bình của người dùng b

$$\overline{r_b} = \frac{1}{|P_b|} * \sum_{p \in P_b} r_{bp}$$

p: Một sản phẩm chung mà 2 người dùng a và b đều đã đánh giá.

Tử số: tích của 2 phần đã đánh giá đã chuẩn hóa

Mẫu số: Tích của 2 độ lệch chuẩn

Giá trị Sim(a,b) nằm trong khoảng [-1,1]

- Gần 1: 2 người dùng có xu hướng đánh giá giống nhau.
- Gần -1: 2 người dùng có xu hướng đánh giá ngược nhau.
- Bằng 0: Không có mối quan hệ giữa 2 người dùng.

Bước 2: Xếp hạng các mặt hàng tương đồng giữa các người dùng

Người dùng mục tiêu có thể rất giống với một số người dùng và có thể không giống nhiều với những người khác. Nên cần xếp hạng những người dùng liên quan. Chúng ta có thể tính r_up như sau:

$$r_{up} = \bar{r_u} + \frac{\sum_{i \in users} sim(u,i) * r_{ip}}{\sum_{i \in users} |sim(u,i)|}$$

r_{up}: Điểm đánh giá dự đoán của người dùng u cho sản phẩm p

$\bar{r_u}$: Điểm đánh giá trung bình của người dùng u

Sim(u,i): Độ tương đồng giữa người dùng u và người dùng i

r_{ip}: Điểm đánh giá thực tế của người dùng i cho sản phẩm p

Tuy nhiên, đối với SVD Recommendation System cho đề tài này, chúng ta cần tập trung vào mối quan hệ tương thích giữa các người dùng để đưa ra gợi ý.

3. Ứng dụng SVD cho hệ thống gợi ý sản phẩm cho người dùng(User base)

Ý tưởng:

Sử dụng phương pháp Compact SVD để phân rã ma trận R(mxn) thành các ma trận:

$$R \approx U_r * \Sigma_r * V_r^T$$

U_r: Có kích thước m x k, là các vecto thể hiện cho user

Σ_r: Có kích thước k x k chứa các giá trị kì dị.

V_r^T:Có kích thước k x n, là các vecto thể hiện cho sản phẩm

- Lý do sử dụng Compact SVD: Giúp giảm chiều dữ liệu, chỉ giữ lại k thành phần quan trọng nhất để biểu diễn dữ liệu đánh giá

Ta xét một vector người dùng mới r(1 x n)

Đối với mục tiêu gợi ý dự đoán cho người dùng, ta cần tìm ma trận U mới, gọi là $U_{new} = r * \Sigma_r^{-1} * V_r$

U_{new} là một vecto mới, tiếp theo ta xét cos α là góc giữa vecto này với từng vecto trong U_r ban đầu bằng công thức:

$$\cos\alpha = \frac{U_{new} * U_i}{|U_{new}| * |U_i|}$$

Giá trị cos α nằm trong khoảng [-1,1]

- Gần 1: 2 người dùng có xu hướng đánh giá giống nhau(2 vecto gần như trùng nhau).
- Gần -1: 2 người dùng có xu hướng đánh giá ngược nhau.
- Bằng 0: Không có mối quan hệ giữa 2 người dùng.

Sau đó ta xây dựng một bảng xếp hạng để chọn ra những người dùng có mức độ tương đồng, dựa trên các rating lớn nhất của những người tương đồng nhất để đưa ra đề xuất cho người dùng mới.

Ví dụ: Cho một bảng người dùng cùng các sản phẩm của họ:

	Sản phẩm A	**Sản phẩm B**	**Sản phẩm C**	**Sản phẩm D**
User1	5	4	0	1
User2	4	0	0	1
User3	1	1	0	5
User4	0	0	5	4
User5	0	1	5	4

Như vậy ta có một ma trận đánh giá A(Người dùng x sản phẩm)

$$A = \begin{matrix} 5 & 4 & 0 & 1 \\ 4 & 0 & 0 & 1 \\ 1 & 1 & 0 & 5 \\ 0 & 0 & 5 & 4 \\ 0 & 1 & 5 & 4 \end{matrix}$$

```
#Ví dụ về SVD cho user-based
import numpy as np
# Tạo ma trận người dùng x sản phẩm
A = np.array([
    [5, 4, 0, 1],
    [4, 0, 0, 1],
    [1, 1, 0, 5],
    [0, 0, 5, 4],
```

```
    [0, 1, 5, 4]
])

# Thực hiện SVD đầy đủ
U, S, Vt = np.linalg.svd(A, full_matrices=False)

# Chọn k = 2 (rút gọn SVD)
k = 2
U_r = U[:, :k]  # Lấy 2 cột đầu của U
S_r = np.diag(S[:k])  # Chuyển S thành ma trận đường chéo (2x2)
V_r_T = Vt[:k, :]  # Lấy 2 hàng đầu của V^T

#In kết quả
print("Ma trận U_r (Người dùng):\n", np.round(U_r, 2))
print("\nMa trận Σ_r (Giá trị kỳ dị):\n", np.round(S_r, 2))
print("\nMa trận V_r^T (Sản phẩm):\n", np.round(V_r_T, 2))
```

```
Ma trận U_r (Người dùng):
 [[-0.28 -0.79]
 [-0.17 -0.44]
 [-0.41 -0.12]
 [-0.59  0.32]
 [-0.61  0.26]]

Ma trận Σ_r (Giá trị kỳ dị):
 [[10.03  0.  ]
 [ 0.    7.19]]

Ma trận V_r^T (Sản phẩm):
 [[-0.25 -0.22 -0.6  -0.73]
 [-0.81 -0.42  0.41  0.07]]
```

Áp dụng Compact SVD với k = 2, ta được:

$$U_R = \begin{matrix} -0.28 & -0.79 \\ -0.17 & -0.44 \\ -0.41 & -0.12 \\ -0.59 & 0.32 \\ -0.61 & 0.26 \end{matrix}$$

$$\Sigma_r = \begin{matrix} 10.03 & 0 \\ 0 & 7.19 \end{matrix}$$

$$V_r^T = \begin{matrix} -0.25 & -0.22 & -0.6 & -0.73 \\ -0.81 & -0.42 & 0.41 & 0.07 \end{matrix}$$

Giả sử chúng ta đang muốn đưa ra đề xuất cho một người dùng mới có đánh giá như sau:

$$r = \begin{matrix} 5 & 0 & 3 & 4 \end{matrix}$$

Ta sẽ tính vector biểu diễn của người dùng mới theo công thức:

$$U_{new} = r * \Sigma_r^{-1} * V_r$$

```
r = np.array([[5, 0, 3, 4]])
S_r_inv = np.linalg.inv(S_r)
U_new = r @ V_r_T.T @ S_r_inv
print("\nMa trận U mới (Người dùng):\n", np.round(U_new, 2))
```

```
Ma trận U mới (Người dùng):
 [[-0.6  -0.36]]
```

Ta tìm thấy $U_{new} = \begin{matrix} -0.6 & -0.36 \end{matrix}$

Cuối cùng, tính Cosine Similarity giữa U_{new} và U_r

$$\cos \alpha = \frac{U_{new} * U_i}{|U_{new}| * |U_i|}$$

```
# Tính cosine similarity
cosine_similarities = []
for i, U_i in enumerate(U_r):
    cos_alpha = np.dot(U_new, U_i).item() / (np.linalg.norm(U_new) * np.linalg.norm(U_i))

    cosine_similarities.append((i + 1, round(cos_alpha, 2)))  # Làm tròn 2 chữ số

# In kết quả
print("\nCosine Similarity với người dùng khác:")
for user, cos_sim in cosine_similarities:
    print(f"Người dùng {user}: {cos_sim}")
```

```
Cosine Similarity với người dùng khác:
Người dùng 1: 0.77
Người dùng 2: 0.79
Người dùng 3: 0.97
Người dùng 4: 0.51
Người dùng 5: 0.59
```

cos similarity của người dùng mới và người dùng 3 là 0.97, rất gần nhau. Có nghĩa là 2 vecto gần như trùng nhau(vì $\cos^{-1}(1) = 0\ radian$), từ đó ta có thể thấy sở thích của người dùng mới rất giống với người dùng thứ 3, ta có thể đưa ra đề xuất sản phẩm D cho người dùng mới, vì sản phẩm D được người dùng thứ 3 đánh giá rất cao.

4. Các phương pháp chọn giá trị k cho Compact SVD.

- Cách chọn giá trị k tối ưu trong Compact SVD rất quan trọng vì nó ảnh hưởng đến khả năng ghi lại thông tin của dữ liệu gốc sau khi giảm số chiều, có 3 phương pháp chính để chọn k.

4.1 Phương pháp tích lũy(Explained Variance)

Mục tiêu của phương pháp này là giữ lại càng nhiều phương sai của dữ liệu gốc càng tốt trong khi giảm số chiều. Hay nói cách khác là chọn k sao cho tổng phương sai tích lũy đạt một ngưỡng nhất định (thường là 90% hoặc 95%).

- Phương sai giải thích được thể hiện bằng explained variance ratio, tức là tỉ lệ phương sai dữ liệu được giữ lại khi ta chọn k thành phần đầu tiên.
- Khi tăng k, lượng thông tin được giữ lại cũng tăng, nhưng đến một mức nào đó, việc tăng k không còn mang lại lợi ích đáng kể.

Công thức tính phương sai tích lũy:

$$EVR_k = \sum_{i=1}^{k} \frac{\sigma_i^2}{\sum_{j=1}^{m} \sigma_i^2}$$

Trong đó:

σ_i^2: Giá trị kì dị thứ i

m: tổng số thành phần chính có thể có

EVR_k: Tổng phương sai tích lũy khi chọn k thành phần

Gọi k* là số thành phần tối ưu:

$$k* = \min\{k | EVR_k \geq \tau\}$$

τ: Ngưỡng phương sai(thường là 90% hoặc là 95%)

4.2 Phương pháp khuỷu tay (Elbow Method)

Mục tiêu: Chọn k tại điểm mà giá trị kì dị σ_i bắt đầu giảm chậm lại:

Độ suy giảm giá trị kì dị:

$$D_k = \sigma_k - \sigma_{k+1}$$

D_k: Là độ giảm của giá trị kỳ dị giữa thành phần k và k+1

$$k* = argmax_k(D_k - D_{k+1})$$

Nếu vẽ trên biểu đồ, điểm này là "khuỷu tay"

4.3 Phương pháp dựa trên kinh nghiệm(Heuristic)

Mục tiêu: chọn k dựa trên quy tắc thực nghiệm

Nếu số lượng đặc trưng nhỏ(<50)

$$k^* = \min(10, m)$$

Nếu số chiều nhỏ hơn 10, giữ nguyên số chiều.

Nếu lớn hơn 10, ta chọn k = 10

Nếu số lượng đặc trưng lớn(>50)

$$k^* = 0.2 * m$$

Thường thường, trong Recommendation System.

k thường nằm trong khoảng 10-100

Nếu số lượng người dùng lớn(>1000), chọn $k \approx 50 - 100$

Phương pháp	**Cách tiếp cận**	**Ưu điểm**	**Nhược điểm**
Explained Variance	Chọn k sao cho tổng phương sai vượt 90% - 95%	Giữ lại tối đa thông tin, tự động điều chỉnh	Có thể không có giá trị k rõ ràng
Elbow Method	Tìm điểm khuỷu tay trên đồ thị giá trị kì dị	Trực quan, dễ áp dụng	Không phải lúc nào cũng có điểm gây rõ ràng
Heuristic	Chọn k dựa vào quy tắc 10-100	Dễ áp dụng, nhanh chúng	Không tối ưu cho từng bộ dữ liệu

III) Dữ liệu và mô hình SVD

Link dataset: https://www.kaggle.com/datasets/grikomsn/amazon-cell-phones-reviews.

Dataset này nói về các đánh giá của khách hàng trên Amazon về điện thoại di động của các thương hiệu: ASUS,Apple,Google, Huawei, Motorola, Nokia, OnePlus, Samsung, Sony, Xiaomi.

Dataset gồm 2 file:

Items.csv: Gồm các mặt hàng từ các thương hiệu.(lấy ngày 28/09/2019).

Reviews.csv: Chứa đánh giá cho các sản phẩm trong items.csv. Đây là bộ dữ liệu chính để áp dụng thuật toán SVD(lấy ngày 26/12/2019). Chúng ta sẽ xử lý với bộ dữ liệu này.

1. Các cột thuộc tính

Các cột thuộc tính trong file reviews.csv:

Tên cột thuộc tính	Kiểu dữ liệu	Ý nghĩa
Asin	Kiểu chuỗi(string)	Mã định danh sản phẩm của Amazon
Name	Kiểu chuỗi(string)	Tên tài khoản đánh giá
Rating	Kiểu số nguyên(int)	số điểm đánh giá
Date	Kiểu Date(string)	Thời gian đánh giá
Verified	Kiểu chuỗi(string)	Đã được xác thực hay chưa
Title	Kiểu chuỗi(string)	Tiêu đề đánh giá
Body	Kiểu chuỗi(string)	Nội dung đánh giá

helpfulVotes	Kiểu số nguyên(int)	Số người đồng ý cho rằng đánh giá này hữu ích

Ngoài ra, các cột thuộc tính trong file items.csv:

Tên cột thuộc tính	**Kiểu dữ liệu**	**Ý nghĩa**
Asin	Kiểu chuỗi(string)	Mã định danh sản phẩm của Amazon
Brand	Kiểu chuỗi(string)	Thương hiệu của sản phẩm đó
Title	Kiểu chuỗi(string)	Tên sản phẩm
URL	Kiểu chuỗi(string)	Đường dẫn tới sản phẩm đó
Image	Kiểu chuỗi(string)	Ảnh sản phẩm
Rating	Kiểu số thực(float)	Đây là điểm đánh giá trung bình của sản phẩm
ReviewURL	Kiểu chuỗi(string)	Đường dẫn tới trang đánh giá sản phẩm
total review	Kiểu số nguyên(integer)	Tổng số đánh giá của sản phẩm đó
Price	Kiểu số thực(float)	Giá sản phẩm
Original Price	Kiểu số thực(float)	Giá gốc sản phẩm

2. Hệ thống gợi ý sản phẩm cho người dùng sử dụng SVD(SVD Recommendation System User Base)

Link full code: https://github.com/HueyAnthonyDisward/User-based-recommender-system.git

Đầu tiên, ta cần đọc dataset và lưu lại:

```
# Load dữ liệu
file_path = '20191226-reviews.csv'
df = pd.read_csv(file_path)
```

Đầu tiên, trúng ta cần lập một ma trận A, ma trận A là ma trận ban đầu được xây dựng bởi người dùng với những đánh giá của họ về sản phẩm, như vậy ta chỉ cần giữ lại 3 cột: asin(mã sản phẩm), user(tên người dùng) và rating(điểm đánh giá):

```
# Chỉ giữ lại các cột cần thiết
ratings = df[['name', 'asin', 'rating']].copy()
```

Có một vấn đề với dataset này, đó là có một người dùng sẽ đưa ra nhiều rating cho một sản phẩm, như vậy ta chuẩn hóa bằng cách lấy trung bình rating của chúng. Đồng thời để cho chắc chắn, ta chuyển toàn bộ cột rating về kiểu số và loại bỏ các giá trị thiếu:

```
# Chuyển rating về kiểu số và loại bỏ giá trị NaN
ratings.loc[:, 'rating'] = pd.to_numeric(ratings['rating'], errors='coerce')
ratings = ratings.dropna(subset=['rating'])


# Tính trung bình rating của mỗi người dùng trên mỗi sản phẩm
ratings = ratings.groupby(['name', 'asin'], as_index=False).agg({'rating': 'mean'})
```

Để có thể đưa ra gợi ý, ta chỉ xử lý đối với các người dùng đánh giá từ 3 sản phẩm trở lên, ta tiến hành lọc:

```
user_counts = ratings['name'].value_counts()
valid_users = user_counts[user_counts >= 3].index
ratings = ratings[ratings['name'].isin(valid_users)]
```

Sau đó, ta chuyển dữ liệu thành ma trận user-item, đó cũng là ma trận A ban đầu:

```
# Chuyển dữ liệu thành ma trận user-item
rating_matrix = ratings.pivot(index='name', columns='asin',
values='rating').fillna(0)
```

Sau đó, ta chọn k = 50(Dựa trên cách chọn k Heuristic), sử dụng SVD để bắt đầu phân rã, thu được 3 ma trận U_r, Σ_r, V_r^T:

```
# Áp dụng SVD với k = 10
k = min(50, rating_matrix.shape[1])
svd = TruncatedSVD(n_components=k)
U_r = svd.fit_transform(rating_matrix)  # Ma trận U_r
Sigma_r = np.diag(svd.singular_values_)  # Ma trận Σ_r
V_r_T = svd.components_  # Ma trận V_r^T
```

Sau đó, ta chọn một người dùng bất kì trong dataset làm người dùng cần được gợi ý, ở đây chúng ta chọn người dùng số 20:

```
test_user_index = 20
r = rating_matrix.iloc[test_user_index].values.reshape(1, -1)
```

Dựa trên lý thuyết, ta tính U_{new}:

```
# Tính toán vector U_new cho người dùng mới
Sigma_r_inv = np.linalg.pinv(Sigma_r)  # Ma trận nghịch đảo của Σ_r
U_new = np.dot(np.dot(r, V_r_T.T), Sigma_r_inv)  # U_new = r * Σ_r^(-1) * V_r
```

Để cho chắc chắn, ta kiểm tra kích thước của các ma trận:

```
print(f"Size của rating_matrix: {rating_matrix.shape}")
print(f"Size của U_r: {U_r.shape}")
print(f"Size của Sigma_r: {Sigma_r.shape}")
print(f"Size của V_r_T: {V_r_T.shape}")
print(f"Size của r (test user): {r.shape}")
print(f"Size của Sigma_r_inv: {Sigma_r_inv.shape}")
print(f"Size của U_new: {U_new.shape}")
```

```
Size của rating_matrix: (1640, 639)
Size của U_r: (1640, 50)
Size của Sigma_r: (50, 50)
Size của V_r_T: (50, 639)
Size của r (test user): (1, 639)
Size của Sigma_r_inv: (50, 50)
Size của U_new: (1, 50)
```

Ma trận A ban đầu có kích thước 1640 x 639

Ta chọn k bằng 50, tức là:

U_r: Phải có kích thước 1640 x 50.

Σ_r: Phải có kích thước 50 x 50.

V_r^T:Phải có kích thước 50 x 639.

Đối với vecto người dùng cần gợi ý, thì cần phải có kích thước: 1 x 639.

Đối với ma trận nghịch đảo của Σ_r, cần có kích thước 50 x 50

Đối với ma trận U_{new}, cần có kích thước 1 x 50

Kiểm tra so với kích thước ma trận trong chương trình, kết quả hoàn toàn trùng khớp.

Sau khi có vector U_{new} rồi, ta tiến hành tính hệ sốcosine của U_{new} với các vector U_i trong U_r, vì Cosine có giá trị lớn nhất là 1, nên ta lấy 10 người dùng cố hệ số cosine cao nhất(gần 1 nhất), đó là những người dùng có thể có nhiều điểm tương đồng với người dùng cần gợi ý nhất:

```
# Tính cosine similarity giữa U_new và tất cả người dùng khác
similarities = cosine_similarity(U_new, U_r)[0]


# Lấy 10 người dùng có hệ số cosine cao nhất
similar_users = np.argsort(similarities)[::-1][0:10]
```

Kết quả trả về:

```
10 người dùng gần nhất với User mới:
1. Abby - Cosine Similarity: 0.9834
2. Jodi - Cosine Similarity: 0.8441
3. Homer - Cosine Similarity: 0.8419
4. ben - Cosine Similarity: 0.8265
5. nathan - Cosine Similarity: 0.8265
6. DW - Cosine Similarity: 0.8215
7. Wilson - Cosine Similarity: 0.8184
8. tim - Cosine Similarity: 0.8167
9. Gloria - Cosine Similarity: 0.8104
10. Tommy - Cosine Similarity: 0.7970
```

Sau đó, vì sở thích của những người này có độ tương đồng khá cao đối với người dùng cần gợi ý, nên ta sẽ đưa ra đề xuất đối với người dùng mới với những sản phẩm được các người dùng trên đánh giá cao, nhưng có một vấn đề, đó là có thể sản phẩm duy nhất của một trong số người dùng trên có rating khá thấp, hơn nữa những người dùng này có thể có chung sản phẩm được đánh giá cao, nếu không xử lý thì danh sách đề xuất sẽ rất ít nên chúng ta cần đặt ra một số điều kiện:

- Đề xuất những sản phẩm mà người dùng mới chưa đánh giá.
- Không đưa ra đề xuất cùng một sản phẩm.
- Chỉ đề xuất những sản phẩm có rating > 3.

```
# Tìm 10 sản phẩm cao nhất từ từng người dùng tương đồng nhưng chưa được test_user đánh giá và có rating > 3
recommended_products = []
seen_products = set()


for user in similar_user_ids:
    user_top_products = ratings[(ratings['name'] == user) & (ratings['rating'] > 3)].sort_values(by='rating', ascending=False)
    for _, row in user_top_products.iterrows():
        if row['asin'] not in seen_products and row['asin'] not in test_user_rated_products:
            recommended_products.append(row)
            seen_products.add(row['asin'])
        if len(recommended_products) >= 10:
            break
    if len(recommended_products) >= 10:
        break
```

Ta có danh sách sản phẩm được đề xuất:

```
10 sản phẩm được gợi ý:
            asin  rating
23953  B01NB1KG8U     4.0
19733  B0134TVNSQ     5.0
19734  B01BG1MW5C     5.0
57877  B078YXQ38Z     5.0
57875  B019S654DO     4.0
50195  B07R57VRJ7     5.0
60325  B00WF988BW     5.0
60326  B01N9TM7KG     5.0
60329  B071VGLCRX     4.0
60330  B07VYP6VLS     4.0
```

Để cho chắc chắn, ta cần kiểm tra lại các sản phẩm gợi ý có trùng với các sản phẩm người dùng đã đánh giá không:

```
# Lấy danh sách sản phẩm mà người dùng mới đã đánh giá
test_user_rated_products = set(rating_matrix.iloc[test_user_index][rating_matrix.iloc[test_user_index] > 0].index)


# Lấy danh sách sản phẩm gợi ý
recommended_products_list = set(recommended_products_df['asin'])


# Đếm số lượng sản phẩm gợi ý trùng với sản phẩm đã đánh giá
overlap = recommended_products_list.intersection(test_user_rated_products)
print(f"Số sản phẩm gợi ý trùng với sản phẩm đã đánh giá: {len(overlap)} / {len(recommended_products_list)}")
print("Các sản phẩm trùng:", overlap)
```

```
Số sản phẩm gợi ý trùng với sản phẩm đã đánh giá: 0 / 10
Các sản phẩm trùng: set()
```

Như vậy, không có sản phẩm có gợi ý trùng với các sản phẩm mà người dùng đã đánh giá.

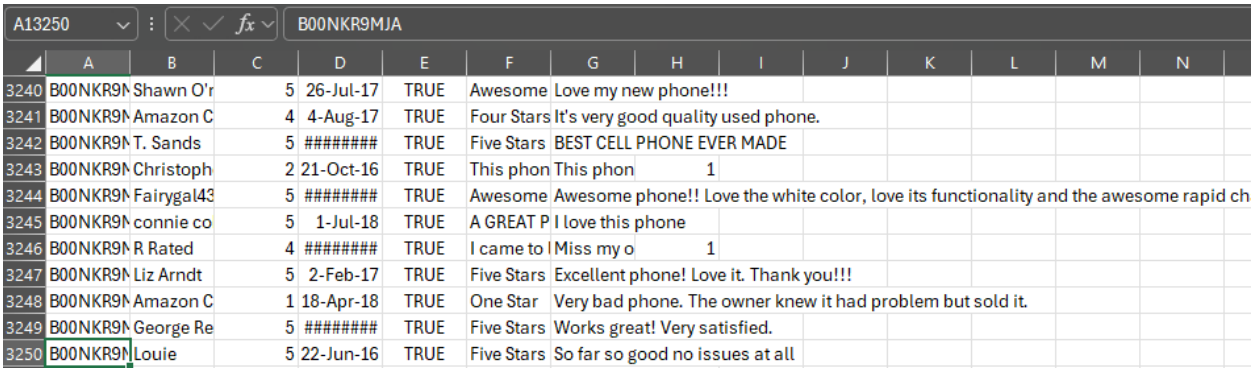
Tiếp theo, ta cần thử với một người dùng mới hoàn toàn không có trong dataset:

```
# Thêm người dùng mới với 5 sản phẩm ngẫu nhiên và rating ngẫu nhiên từ 1-5
new_user_name = 'Huey'
all_products = rating_matrix.columns.tolist()
new_user_products = random.sample(all_products, 5)
new_user_ratings = [random.randint(1, 5) for _ in range(5)]
```

```
10 người dùng gần nhất với người dùng Huey:
1. Louie - Cosine Similarity: 0.9261
2. Rebekah - Cosine Similarity: 0.9232
3. Kristi - Cosine Similarity: 0.9203
4. shannon - Cosine Similarity: 0.8987
5. #NAME? - Cosine Similarity: 0.8850
6. LD - Cosine Similarity: 0.8737
7. BC - Cosine Similarity: 0.8548
8. Whitney - Cosine Similarity: 0.8323
9. B - Cosine Similarity: 0.7466
10. J.E. - Cosine Similarity: 0.7005

10 sản phẩm được gợi ý:
            asin  rating
6454   B00NKR9MJA     5.0
8192   B01JAWWVXW     5.0
8194   B07ND4ZN2X     5.0
8191   B00JYR6GGM     4.0
6147   B072N3GKSM     4.0
11098  B00O15MWOM     5.0
11100  B07W14HFQP     5.0
0      B01M01YX15     5.0
2      B0788F8DKC     5.0
6183   B00JEHJMG8     5.0
Số sản phẩm gợi ý trùng với sản phẩm đã đánh giá: 0 / 10
Các sản phẩm trùng: set()
```

Kiểm tra thử, người dùng Louie – Người dùng có độ tương thích cao nhất với người dùng Huey



	A	B	C	D	E	F	G	H	I	J	K	L	M	N
3240	B00NKR9N	Shawn O'r	5	26-Jul-17	TRUE	Awesome	Love my new phone!!!							
3241	B00NKR9N	Amazon C	4	4-Aug-17	TRUE	Four Stars	It's very good quality used phone.							
3242	B00NKR9N	T. Sands	5	########	TRUE	Five Stars	BEST CELL PHONE EVER MADE							
3243	B00NKR9N	Christoph	2	21-Oct-16	TRUE	This phon	This phon	1						
3244	B00NKR9N	Fairygal43	5	########	TRUE	Awesome	Awesome phone!! Love the white color, love its functionality and the awesome rapid ch							
3245	B00NKR9N	connie co	5	1-Jul-18	TRUE	A GREAT P	I love this phone							
3246	B00NKR9N	R Rated	4	########	TRUE	I came to l	Miss my o	1						
3247	B00NKR9N	Liz Arndt	5	2-Feb-17	TRUE	Five Stars	Excellent phone! Love it. Thank you!!!							
3248	B00NKR9N	Amazon C	1	18-Apr-18	TRUE	One Star	Very bad phone. The owner knew it had problem but sold it.							
3249	B00NKR9N	George Re	5	########	TRUE	Five Stars	Works great! Very satisfied.							
3250	B00NKR9N	Louie	5	22-Jun-16	TRUE	Five Stars	So far so good no issues at all							

Có đánh giá 5 sao cho sản phẩm B00NKR9MJA, như vậy hệ thống đã đưa ra đề xuất đúng.

IV) Kết luận

SVD (Phân rã giá trị suy biến) là một kỹ thuật quan trọng trong xử lý dữ liệu, đặc biệt là trong Machine Learning và Data Science. Trong đề tài nghiên cứu lần này chúng ta đã tìm hiểu về cách áp dụng SVD vào hệ thống gợi ý người dùng.

SVD có nhiều ưu điểm trong hệ thống gợi ý:

- **Giảm nhiễu:** Loại bọ dữ liệu không quan trọng, giúp gợi ý chính xác.
- **Hiểu được yếu tố tiềm ẩn:** Giúp tìm ra các đặc điểm ẩn quan trọng để đưa ra gợi ý.
- **Gợi ý:** Có thể đưa ra gợi ý cho người dùng đối với các sản phẩm mà họ có thể thích.
- **Tối ưu hóa hiệu suất:** Khi giảm chiều dữ liệu thì thuật toán chạy nhanh hơn.

Trên thực tế, có những thương hiệu lớn sử dụng SVD để áp dụng cho hệ thống của họ, trên Kaggle cũng có những dataset liên quan:

Netflix: Netflix sử dụng SVD để phân tích đánh giá của người dùng và gợi ý phim phù hợp với sở thích cá nhân.

Spotify: Tương tự thì spotify cũng dùng SVD dựa trên lịch sử nghe nhạc để đề xuất bài hát.

Amazon: Amazon phân tích đánh giá sản phẩm và đề xuất những sản phẩm liên quan dựa trên mức độ yêu thích của người dùng.

SVD là một kỹ thuật mạnh mẽ trong gợi ý, giúp cải hiện độ chính xác và tối ưu hóa hiệu suất, giúp phát hiện yếu tố ẩn trong dữ liệu để từ đó gợi ý cho từng người dùng.

Đề tài nghiên cứu về lý thuyết cơ bản của SVD cùng với cách áp dụng cho hệ thống gợi ý người dùng(user base), thực hành trên một tập dataset để có thể áp dụng các bước để từ đó tìm ra quan hệ của người dùng mới với các người dùng khác rồi đưa ra đề xuất dựa trên sở thích của các người dùng liên quan.

Phương pháp SVD vừa giúp tăng độ chính xác và giảm đáng kể thời gian tính toán nhờ khả năng nén dữ liệu và tối ưu hóa xử lý. Tuy nhiên vẫn còn một số thách thức như lựa chọn số lượng thành phần đặc trưng phù hợp hay xử lý dữ liệu thưa cần được quan tâm để nâng cao hiệu suất.

SVD là một công cụ mạnh mẽ và linh hoạt trong hệ thống gợi ý, trong tương lai có thể kết hợp SVD với các phương pháp khác(Mô hình mạng nơ ron hoặc kỹ thuật tối ưu hóa) để nâng cao hiệu suất.

TÀI LIỆU THAM KHẢO

[1] Jason Brownlee, *Using Singular Value Decomposition to Build a Recommender System*, Machine Learning Mastery, 2023. Available: https://machinelearningmastery.com/using-singular-value-decomposition-to-build-a-recommender-system/

[2] GeeksforGeeks, *SVD in Recommendation Systems*, 2023. Available: https://www.geeksforgeeks.org/svd-in-recommendation-systems/

[3] GeeksforGeeks, *User-Based Collaborative Filtering*, 2023. Available: https://www.geeksforgeeks.org/user-based-collaborative-filtering/