

Xây dựng hệ thống gợi ý sản phẩm sử dụng tập dữ liệu Amazon Reviews' 23

Thành viên nhóm:

Nguyễn Hoàng Phúc - 22110400

Phạm Trung Kỳ - 22110361

Mục lục

- 1. Tóm tắt
- 2. Giới thiệu
- 3. Dữ liệu
 - 3.1 Dữ liệu sử dụng trong dự án
 - 3.2 Dữ liệu cho mô hình SVD
 - 3.3 Tiền xử lý dữ liệu
- 4. Phương pháp
 - 4.1 Ý tưởng chung
 - 4.2 Matrix Factorization
 - 4.2.1 SGD (Stochastic Gradient Descent)
 - 4.2.2 ALS (Alternating Least Squares)
 - 4.2.3 SVD (Singular Value Decomposition)
 - 4.3 Locality Sensitive Hashing (LSH)
 - 4.4 Neural Collaborative Filtering (NCF)
- 5. Thực nghiệm, kết quả và thảo luận
 - 5.1 Thiết lập thí nghiệm
 - A. Quy trình chung
 - B. ALS (Alternating Least Squares)
 - C. SGD (Stochastic Gradient Descent)
 - D. NCF (Neural Collaborative Filtering)
 - E. SVD (Singular Value Decomposition)
 - F. LSH (Locality-Sensitive Hashing)
 - 5.2 Kết quả
- 6. Kết luận
- 7. Phụ lục
 - A. Chuẩn hóa dữ liệu theo item mean
 - B. Công thức cập nhật trong SGD
 - C. Công thức cập nhật trong ALS (Alternating Least Squares)
- 8. Đóng góp
- 9. Tham khảo

1. Tóm tắt

Hệ thống gợi ý sản phẩm (recommender systems) là thành phần then chốt cho các nền tảng thương mại điện tử, giúp cá nhân hóa trải nghiệm người dùng và tăng tỷ lệ chuyển đổi mua hàng. Dự án này xây dựng một hệ thống gợi ý sản phẩm sử dụng dữ liệu đánh giá từ **Amazon Reviews 2023**, tập con **Arts, Crafts & Sewing** (tổng ~1.7M đánh giá). Nhóm thử nghiệm các phương pháp cổ điển và hiện đại: **Matrix Factorization** (SVD, ALS tối ưu bằng alternating least squares), tối ưu trực tiếp bằng **SGD, Neural Collaborative Filtering (NCF)** để học mối quan hệ phi tuyến giữa user & item, và **LSH** để tăng tốc tìm kiếm sản phẩm tương tự. Dữ liệu được tiền xử lý (loại bỏ bản ghi thiếu, đánh chỉ số User/Item, chuẩn hóa rating, xử lý văn bản cho mô hình text-aware). Hiệu năng được đánh giá bằng MAE, RMSE cho dự đoán rating và P@K, NDCG@K cho xếp hạng gợi ý. Kết quả thực nghiệm cho thấy ALS cho MAE/RMSE tốt hơn SVD; mô hình NCF và ALS cho kết quả tốt tương tự nhau. Báo cáo mô tả chi tiết thiết kế thí nghiệm, siêu tham số, kết quả, phân tích quá khớp, các bước tối ưu và đề xuất cải tiến trong tương lai.

2. Giới thiệu

Bài toán. Với tập đánh giá $\mathcal{D} = (u, i, r_{ui})$ (User ID, Item ID, Rating), mục tiêu là sẽ dự đoán rating cho các cặp (u, i) chưa có và sinh ra danh sách top-K sản phẩm cho mỗi user sao cho phù hợp với sở thích.

Tại sao quan trọng. Gợi ý cá nhân hóa giúp tăng trải nghiệm, tăng thời gian tương tác, và doanh thu. Bài toán thách thức do dữ liệu rất thưa (sparse), số lượng user/item lớn, và tính động của sở thích.

Input / Output.

- Input: ma trận đánh giá $R \in \mathbb{R}^{|U| \times |I|}$ (sparse), metadata (title, main category, description,...), review text.
- Output: (1) predicted rating \hat{r}_{ui} cho cặp chưa xuất hiện, (2) danh sách các item top-K gợi ý cho user

3. Dữ liệu

Trong dự án này, nhóm sử dụng **Amazon Reviews Dataset (2023)** được công bố bởi McAuley Lab (UCSD). Đây là một trong những tập dữ liệu gợi ý sản phẩm quy mô lớn nhất hiện nay, với hơn 571,54 triệu đánh giá được thu thập từ nhiều danh mục hàng hóa khác nhau. Mỗi bản ghi (record) bao gồm thông tin đa dạng như:

- **User Reviews:** điểm đánh giá (rating), nội dung bình luận (review text), số lượt bình chọn "helpful", v.v.

- **Item Metadata:** thông tin mô tả, giá, hình ảnh sản phẩm, danh mục.
- **Links:** dữ liệu quan hệ giữa sản phẩm và người dùng (ví dụ: "bought together", "also viewed").

Tập dữ liệu đã được nhóm tác giả chia sẵn thành ba phần train, validation, và test dựa trên **Absolute-Timestamp Splitting**. Cụ thể, mỗi người dùng có một chuỗi tương tác theo thời gian, và các mốc thời gian t_1 và t_2 được định nghĩa như sau:

- **Training part:** các tương tác có timestamp trong khoảng $(-\infty, t_1)$
- **Validation part:** các tương tác trong khoảng $[t_1, t_2]$
- **Testing part:** các tương tác trong khoảng $[t_2, +\infty)$

Với tập dữ liệu năm 2023, các giá trị được cố định là $t_1 = 1628643414042$ và $t_2 = 1658002729837$. Cách chia này đảm bảo rằng dữ liệu mô phỏng thực tế, mô hình chỉ được huấn luyện trên dữ liệu quá khứ và đánh giá trên tương tác xảy ra sau đó, giúp kết quả phản ánh đúng tính chất dự đoán theo thời gian.

3.1. Dữ liệu sử dụng trong dự án

Nhóm chọn danh mục "**Arts, Crafts and Sewing**" làm tập chính để huấn luyện, với tổng cộng khoảng **1,8 triệu lượt đánh giá**, bao gồm:

- **197.3K người dùng (users)**
- **90.0K sản phẩm (items)**
- **Tỷ lệ chia tập:** 1.4M (train) / 192.8K (validation) / 210.8K (test)

Mỗi dòng dữ liệu bao gồm các trường chính sau:

Trường	Mô tả
user_id	Mã định danh người dùng
parent_asin	Mã sản phẩm
rating	Điểm đánh giá (1-5)
timestamp	Thời điểm đánh giá
history	Lịch sử các sản phẩm đã đánh giá trước đó

Ví dụ về một số dòng dữ liệu trong tập train:

user_id	parent_asin	rating	timestamp	history
AFKZENTNBQ7A7V7UX...	0913212148	5.0	1441260318000	NULL
AFKZENTNBQ7A7V7UX...	B0BGXTQLL1	5.0	1523092443644	0913212148
AFKZENTNBQ7A7V7UX...	B0BV5ZGRRM	5.0	1564347303691	0913212148 B0BGXT...

Trong quá trình xử lý, nhóm chỉ sử dụng ba trường `user_idx`, `item_idx`, và `rating` cho các mô hình gợi ý chính (ALS, SGD, NCF, LSH). Riêng với **SVD**, do đặc thù thuật toán yêu cầu ma trận đánh giá đầy đủ, nhóm chuyển đổi dữ liệu sang dạng ma trận $R_{m \times n}$ (người dùng × sản phẩm).

3.2. Dữ liệu cho mô hình SVD

Ma trận đầy đủ được tạo thành để sử dụng cho thuật toán SVD là khá lớn và vì hạn chế phần cứng, nhóm sử dụng danh mục nhỏ hơn là “**All_Beauty**”, có kích thước khiêm tốn hơn nhiều:

- **253 người dùng**
- **356 sản phẩm**
- Dữ liệu được chia theo tỷ lệ 70% - 10% – 20% cho train – validation – test tương ứng.

Danh mục này phù hợp để kiểm thử các thuật toán phân rã ma trận (Matrix Factorization) như **SVD** mà không gặp giới hạn về bộ nhớ hoặc thời gian huấn luyện.

3.3 Tiền xử lý dữ liệu

Trong dữ liệu gốc, `user_id` và `parent_asin` là kiểu chuỗi không tiện cho việc đưa vào mô hình, `rating` cũng chưa được chuẩn hóa dẫn đến việc huấn luyện lâu hơn, giá trị rating từ 0 đến 5 tiềm ẩn bias, khó khăn trong việc xử lý vấn đề cold-start.

Mục tiêu là chuẩn hóa rating và tạo chỉ số user/item nhất quán giữa các split để phục vụ việc huấn luyện mô hình, xử lý vấn đề cold-start khi user/item mới xuất hiện từ tập valid hoặc test không có trong tập train.

Đọc dữ liệu

- Dữ liệu gồm ba split `train/valid/test` (giữ nguyên như mục 3.1).
- Dữ liệu được đọc bằng PySpark để xử lý hiệu quả các tập dữ liệu lớn.
- Ta xử lý riêng biệt user và item, nhưng chung quy trình.

Tạo mapping user/item

- Lấy tập hợp tất cả user/item xuất hiện trong các split, loại bỏ lặp lại, ta được tập chứa các user/item id.
- Gán `userIndex` và `itemIndex` (unique integer) cho mỗi user/item, nhằm đảm bảo mapping nhất quán giữa các split và các lần chạy.
- Lưu mapping ra file để tái sử dụng cho mô hình.

Chuẩn hóa rating theo item (item-mean normalization)

- Tính `item_mean = avg(rating)` trên train; tính `global_mean` trên train để dùng khi item mới xuất hiện trong valid/test.

- Với mỗi split:

- Gán index user/item theo mapping.
- Gán `item_mean` cho mỗi record; nếu item chưa xuất hiện trong train → dùng `global_mean`.
- Sinh cột chuẩn hóa: `rating_norm = rating - item_mean`.

Lưu dữ liệu tiền xử lý

- Lưu `item_means` và DataFrame đã chuẩn hóa (train/valid/test) để dùng trực tiếp cho mô hình.

Một số kết quả của quá trình xử lý dữ liệu

- User mapping

<code>user_id</code>	<code>userIndex</code>
AE22236AFRRSMQIKG...	0
AE222H3FGXWLHRFUM...	1
AE224QIIILW6WVFAE...	2

- Rating norm

<code>itemIndex</code>	<code>parent_asin</code>	<code>userIndex</code>	<code>user_id</code>	<code>rating</code>	<code>rating_norm</code>
89574	B0C7CXW1JB	193523	AHXOCRWNKC552ESFF...	4.0	-0.7849872773536
21220	B00N9RIBIO	41477	AEV2AM5IEUVKSKEJ...	5.0	0.55714285714285
89056	B0C4KB3GMD	193989	AHXY3L4R6MASA7JH4...	1.0	-3.0491803278688
75509	B09HBS73N9	183561	AHR7OPTZYKDBAAJQW...	5.0	0.49910114342077
77422	B09PFQ3NJW	970	AE2OBGULEEQ6SQPV6...	5.0	0.49910114342077

4. Phương pháp

4.1 Ý tưởng chung

Mỗi user u và item i được mô tả bằng vector đặc trưng $p_u, q_i \in \mathbb{R}^k$. Giá trị phù hợp (compatibility) được tính bằng dot product: $\hat{r}_{ui} = p_u^\top q_i$. Với NCF dùng mạng phi tuyến $f(p_u, q_i)$.

Các phương pháp được sử dụng ở đây sẽ học các vector đặc trưng từ dữ liệu đánh giá của những người dùng với những item một cách tự động.

4.2 Matrix Factorization

Matrix Factorization là cách biểu diễn ma trận user-item **dưới dạng tích của hai ma trận nhỏ hơn**:

$$R \approx PQ^\top$$

- R là ma trận rating gốc (user \times item).
- P là ma trận **user latent factors** (mỗi user thành một vector đặc trưng).
- Q là ma trận **item latent factors** (mỗi item thành một vector đặc trưng).

Các thuật toán MF sẽ cố gắng tìm P và Q sao cho **tích PQ^\top dự đoán rating gần đúng với rating thật**.

4.2.1. SGD (Stochastic Gradient Descent)

- Ý tưởng: **cập nhật từng user và item từng bước** dựa trên sai số dự đoán.
- Sai số dự đoán cho user u với item i :

$$e_{ui} = r_{ui} - p_u^\top q_i$$

- Cập nhật vector latent với learning rate η :

$$\begin{aligned} p_u &\leftarrow p_u + \eta(e_{ui}q_i - \lambda p_u) \\ q_i &\leftarrow q_i + \eta(e_{ui}p_u - \lambda q_i) \end{aligned}$$

- Giải thích:
 - $e_{ui}q_i$ và $e_{ui}p_u$ giúp giảm sai số dự đoán.
 - λp_u và λq_i là regularization để tránh overfitting.

4.2.2. ALS (Alternating Least Squares)

- Ý tưởng là thay phiên nhau tối ưu P rồi Q^{**} .
- Khi Q cố định, mỗi vector p_u được tính bằng công thức **least squares**:

$$p_u = (Q_u^\top Q_u + \lambda I)^{-1} Q_u^\top r_u$$

- Trong đó:
 - Q_u là ma trận chứa các q_i mà user u đã đánh giá.
 - r_u là vector rating quan sát của user u .
- Tương tự, khi P cố định, cập nhật Q .
- Ưu điểm là dễ song song hóa, xử lý ma trận thưa tốt.

4.2.3. SVD (Singular Value Decomposition)

- Ý tưởng là tách ma trận rating $R \in \mathbb{R}^{m \times n}$ thành ba ma trận:

$$R \approx U\Sigma V^\top$$

- $U \in \mathbb{R}^{m \times k}$: đặc trưng người dùng
- $\Sigma \in \mathbb{R}^{k \times k}$: giá trị kỳ dị
- $V \in \mathbb{R}^{n \times k}$: đặc trưng sản phẩm
- **Truncated SVD**: chỉ giữ $k \ll \min(m, n)$ thành phần chính \rightarrow giảm nhiễu, giảm chiều, tăng tốc.
- Kết quả ta được:

$$\hat{R} = U\Sigma V^\top$$

với \hat{r}_{ui} là rating dự đoán; các hàng của U, V là embedding của user và item.

4.3 Locality Sensitive Hashing (LSH)

Ý tưởng:

- LSH là kỹ thuật để tìm các vector gần nhau trong không gian lớn nhanh hơn so với tính toán tất cả cặp.
- Cách làm: ánh xạ mỗi vector vào một "bucket" sao cho các vector gần nhau thường rơi vào cùng bucket.
- Ứng dụng:
 - Tìm **nearest neighbors xấp xỉ** (approximate nearest neighbors).
 - Dùng cho item-based KNN hoặc tạo candidates trước khi xếp hạng bằng model chính.
 - Gợi ý top sản phẩm liên quan.

Cách hoạt động:

1. Chia không gian embedding bằng cách sinh ra một hoặc nhiều tập các mặt phẳng ngẫu nhiên (hyperplane).
2. Với mỗi vector:
 - Kiểm tra nó nằm phía nào của mỗi mặt phẳng (ví dụ: >0 hay <0).
 - Chuyển mỗi vector thành chuỗi nhị phân (0/1) \rightarrow **hash code**.
3. Các vector có hash code giống nhau sẽ vào cùng bucket.
4. Khi vector mới vào sẽ được hash thành hashcode và thực hiện KNN với các vector trong bucket tương ứng.

4.4 Neural Collaborative Filtering (NCF)

Ý tưởng của việc sử dụng mạng neural network là để học mối quan hệ phi tuyến giữa các cặp vector embedding user u và item i trong việc dự đoán rating \hat{r}_{ui} .

Input:

- Mỗi user u được ánh xạ thành vector embedding p_u
- Mỗi item i được ánh xạ thành vector embedding p_i
- Các vector embedding này được khởi tạo ngẫu nhiên và sẽ được cải thiện dần khi mô hình học.

Kết hợp embedding:

- Ghép vector user và item thành một vector chung:

$$x = [p_u, p_i]$$

- Vector này đại diện cho tương tác giữa user và item.

MLP (Multi-Layer Perceptron):

- Dữ liệu đi qua các lớp phi tuyến liên tiếp (Linear + Activation + Dropout/BatchNorm)
- MLP học cách ánh xạ từ vector embedding sang rating dự đoán, khám phá các tương tác phi tuyến phức tạp hơn giữa user và item.

Output:

- Lớp cuối cùng trả về **rating dự đoán** \hat{r}_{ui}
- Sử dụng **MSE loss** khi training để mạng học dự đoán rating thật với AdamW optimizer.

5. Thực nghiệm, kết quả và thảo luận

5.1 Thiết lập thí nghiệm

A. Quy trình chung

- **Split:** train/valid/test giữ nguyên theo mục 3.1.
- **Metrics chính:**

- Regression: **MAE, RMSE:**

$$\text{MAE} = \frac{1}{N} \sum |r_{ui} - \hat{r}_{ui}|,$$

$$\text{RMSE} = \sqrt{\frac{1}{N} \sum (r_{ui} - \hat{r}_{ui})^2}.$$

- Ranking: **Precision@K, NDCG@K:**

$$\text{DCG}_p = \sum_{i=1}^p \frac{\text{rel}_i}{\log_2(i+1)},$$

$$\text{NDCG}_p = \frac{\text{DCG}_p}{\text{IDCG}_p}.$$

- **Pipeline experiment:**

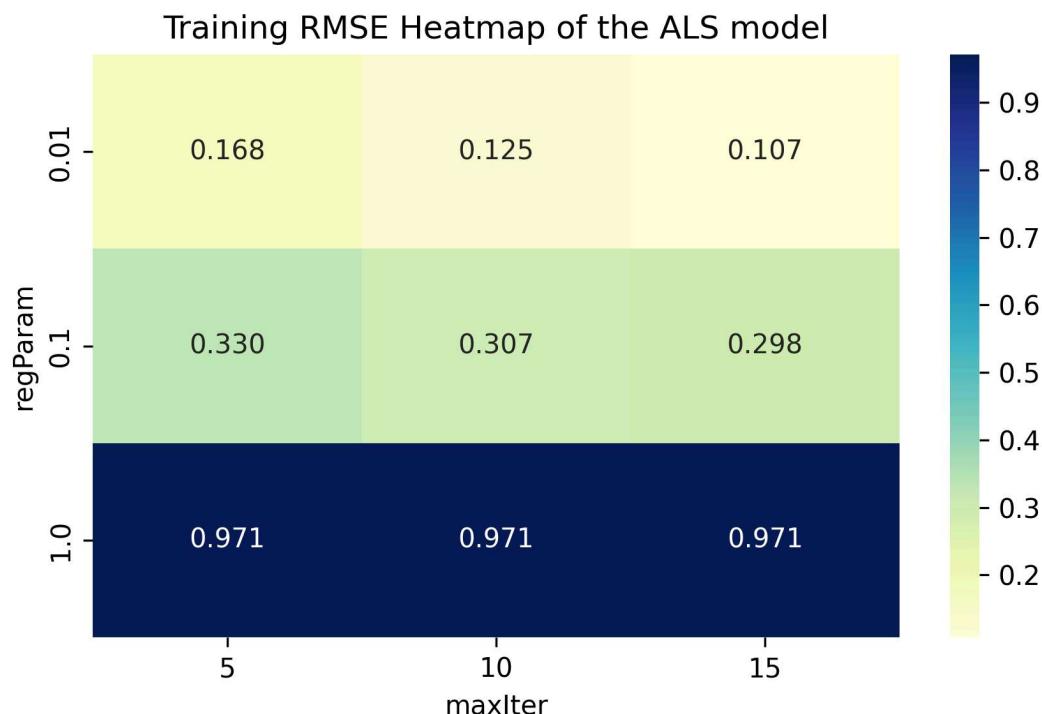
1. Load dữ liệu đã tiền xử lý ở mục 3.3.

2. Huấn luyện mô hình trên train, validation để chọn hyperparameter/early stopping, và cuối cùng đánh giá trên tập test.
3. Lưu checkpoint, lịch sử training, kết quả experiment.

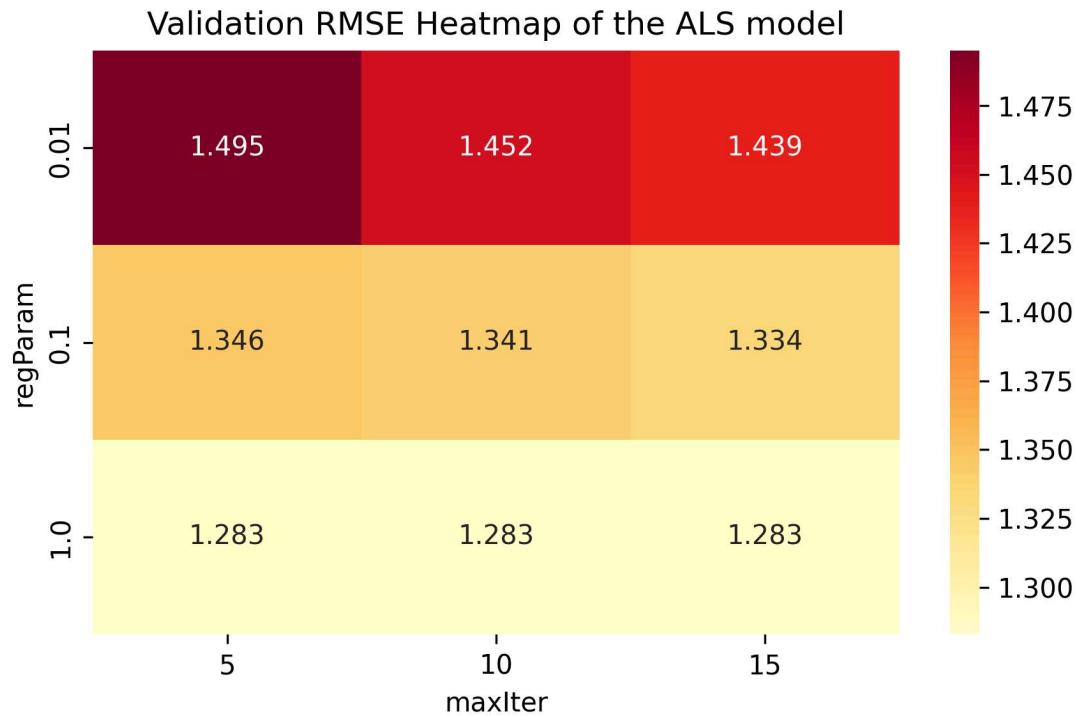
B. ALS (Alternating Least Squares)

Nhóm sử dụng **ALS** trong Spark để huấn luyện mô hình phân rã ma trận, với mục tiêu tối ưu các siêu tham số `rank`, `regParam`, và `maxIter`. Quá trình tìm kiếm được thực hiện bằng **grid search** trên các giá trị: `rank = 10`, `regParam ∈ {0.01, 0.1, 1.0}`, và `maxIter ∈ {5, 10, 15}`. Mỗi tổ hợp được huấn luyện trên tập train và đánh giá trên tập validation bằng các độ đo RMSE và MAE.

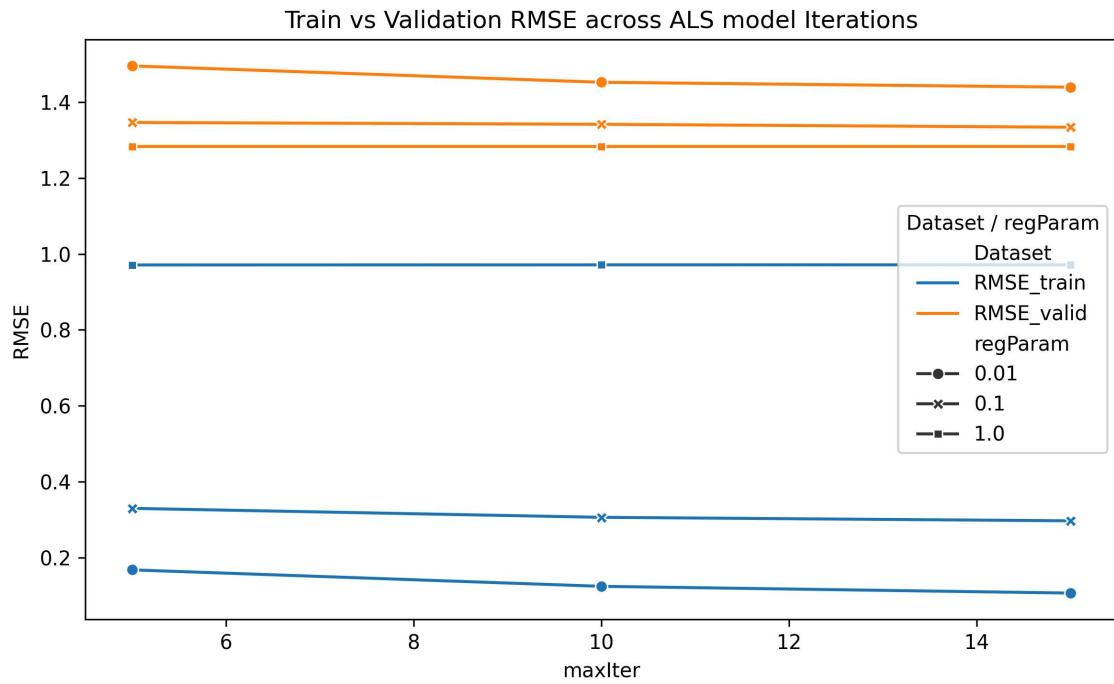
Kết quả cho thấy cấu hình **rank = 10, regParam = 1.0, maxIter = 10–15** đạt hiệu năng tốt nhất, với **RMSE=1.1078** và **MAE=0.7635** cân bằng giữa train và validation, cho thấy việc tăng giá trị regularization giúp giảm hiện tượng overfitting rõ rệt so với các giá trị nhỏ hơn. Mô hình cũng đạt **0.9821 NDCG@10** và **0.7952 Precision@10** trên tập test



Hình 1. Phân bố **RMSE** trên tập huấn luyện của mô hình ALS theo các giá trị siêu tham số. Biểu đồ heatmap thể hiện sai số RMSE tương ứng với từng cặp (`maxIter`, `hệ số regularization λ`), cho thấy mức độ hội tụ và khả năng khớp dữ liệu huấn luyện của mô hình.

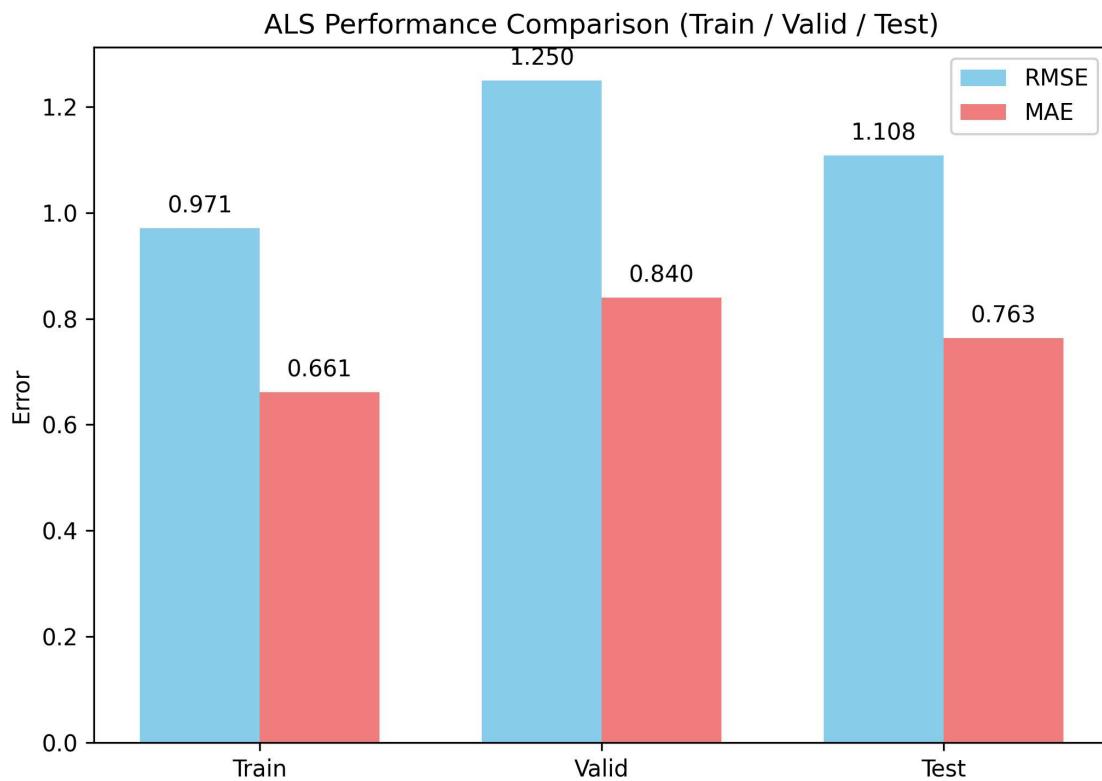


Hình 2. Phân bố RMSE trên tập kiểm định của mô hình ALS theo các giá trị siêu tham số. Biểu đồ cho thấy ảnh hưởng của regularization và số vòng lặp đến khả năng tổng quát hóa của mô hình; λ lớn giúp giảm sai số kiểm định và hạn chế overfitting.



Hình 3. So sánh RMSE giữa tập huấn luyện và tập kiểm định theo số vòng lặp `maxIter` và các giá trị regularization λ . Biểu đồ lineplot cho thấy xu hướng hội tụ: RMSE tập huấn luyện giảm rất nhẹ khi tăng số vòng

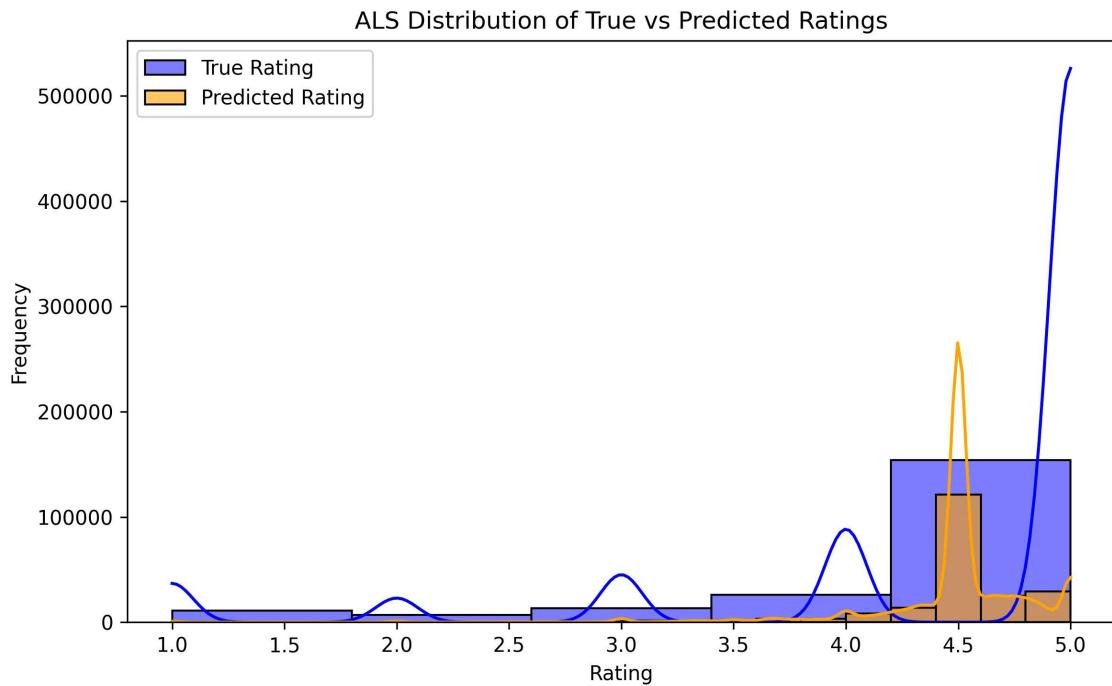
lặp, trong khi RMSE tập kiểm định ổn định hơn với regularization cao giảm mạnh overfitting, thể hiện khả năng tổng quát hóa của mô hình.



Hình 4. So sánh RMSE và MAE trên các tập Train, Validation và Test.

Biểu đồ barplot tổng hợp giúp đánh giá hiệu năng chung của mô hình trên tất cả các tập dữ liệu, đồng thời quan sát sự cân bằng giữa huấn luyện và kiểm định, cũng như khả năng dự đoán trên tập Test.

Khi regParam nhỏ (0.01–0.1), mô hình đạt sai số huấn luyện rất thấp nhưng sai số kiểm định cao, cho thấy hiện tượng overfitting rõ rệt. Khi tăng regParam lên 1.0, sai số huấn luyện tăng nhẹ trong khi sai số kiểm định giảm và ổn định hơn, chứng tỏ mô hình tổng quát hóa tốt hơn. Ngoài ra, việc tăng maxIter giúp mô hình hội tụ tốt hơn ở các giá trị nhỏ, nhưng ảnh hưởng không đáng kể khi regParam cao. Nhìn chung, cấu hình rank = 10, regParam = 1.0, maxIter = 10–15 cho kết quả cân bằng và đáng tin cậy nhất.



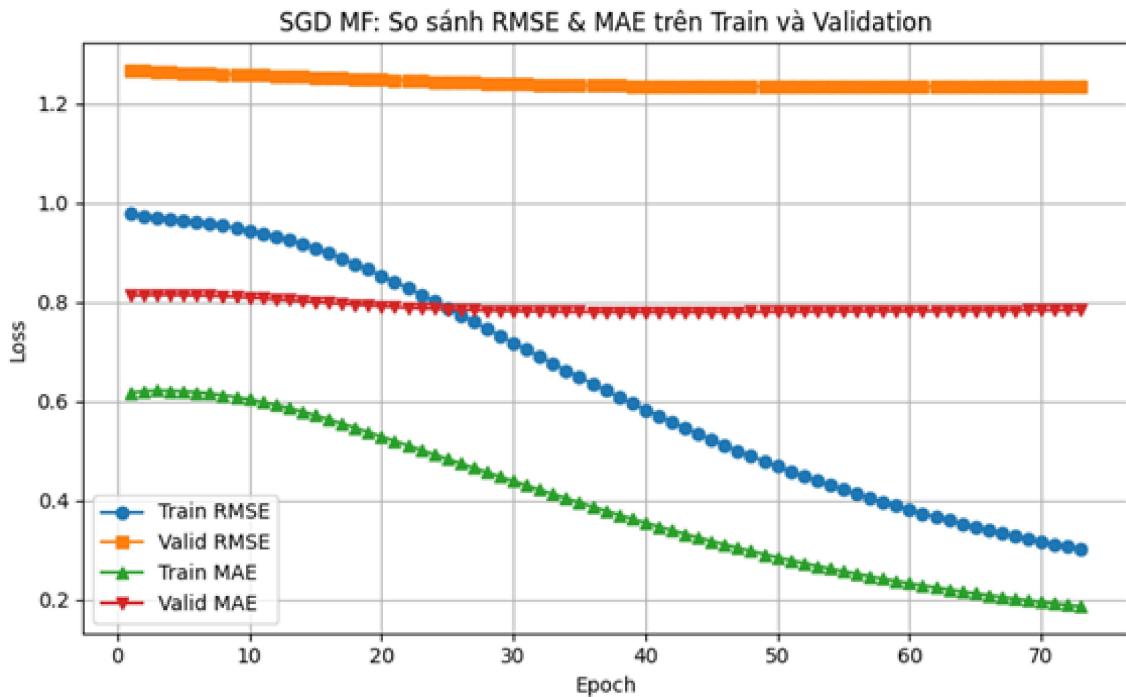
Hình 5. Phân phối điểm đánh giá (rating distribution) trong tập dữ liệu.

Biểu đồ histogram thể hiện tần suất xuất hiện của từng mức rating từ 1 đến 5.

Phân phối điểm đánh giá bị lệch phải, trong đó rating 4 và 5 chiếm tỷ lệ lớn nhất, tiếp đến là rating 1, 3 và 2 thấp hơn nhiều. Mô hình dự đoán tập trung chủ yếu trong khoảng 4.5–5, đạt đỉnh ở 4.5, cho thấy mô hình có xu hướng định giá các sản phẩm ở mức cao, phản ánh dữ liệu huấn luyện bị thiên lệch về các đánh giá tích cực. Điều này nhấn mạnh tầm quan trọng của việc sử dụng regularization hoặc normalization để giảm sai lệch trong dự đoán và cải thiện khả năng tổng quát hóa.

C. SGD (Stochastic Gradient Descent)

Trong quá trình huấn luyện mô hình phân rã ma trận bằng thuật toán SGD, nhóm thử nghiệm hai bộ siêu tham số để tìm cấu hình tối ưu. Ở bộ đầu tiên $K = 30, lr = 0.007, \lambda = 0.02$, validation RMSE gần như không đổi, cho thấy hiện tượng underfitting do số chiều ẩn và tốc độ học còn thấp. Khi tăng lên $K = 50, lr = 0.01, \lambda = 0.05$, mô hình cải thiện rõ rệt, đạt MAE = 0.704, RMSE = 1.0948, Precision@K = 0.8878 và NDCG@K = 0.9819. Do validation loss đã bão hòa, nhóm dừng việc tinh chỉnh thêm vì mô hình đã đạt mức tổng quát hóa tốt, tránh overfitting và tối thiểu hóa chi phí huấn luyện không cần thiết.

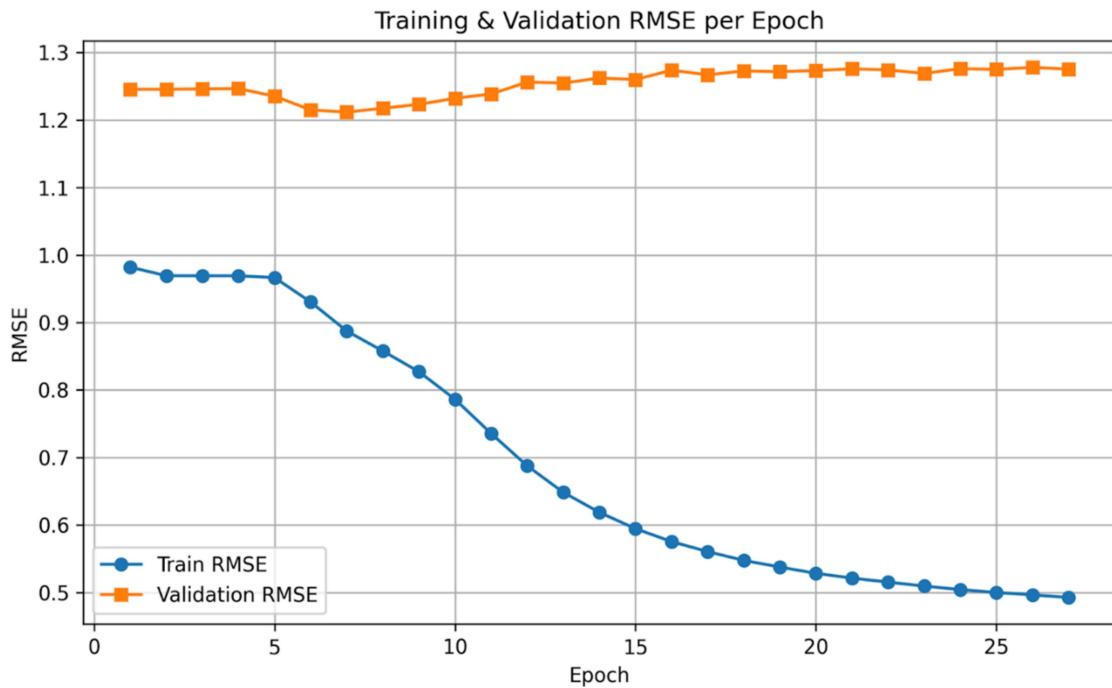


Hình 6. Biểu đồ huấn luyện mô hình SGD – train loss và validation loss theo số epoch.

Về RMSE thì biểu đồ cho thấy train loss giảm nhanh, trong khi validation loss hầu như không giảm đáng kể. Điều này cho thấy dữ liệu tương đối đơn giản, khiến mô hình hội tụ sớm và có overfitting rõ rệt. Tuy nhiên, khoảng cách nhỏ giữa RMSE train và validation loss cũng gợi ý rằng mô hình đã gần đạt giới hạn hiệu năng với cấu hình siêu tham số hiện tại. Với MAE loss thì ngoài việc có giá trị thấp hơn RMSE thì cũng cho một hình dáng đồng nhất.

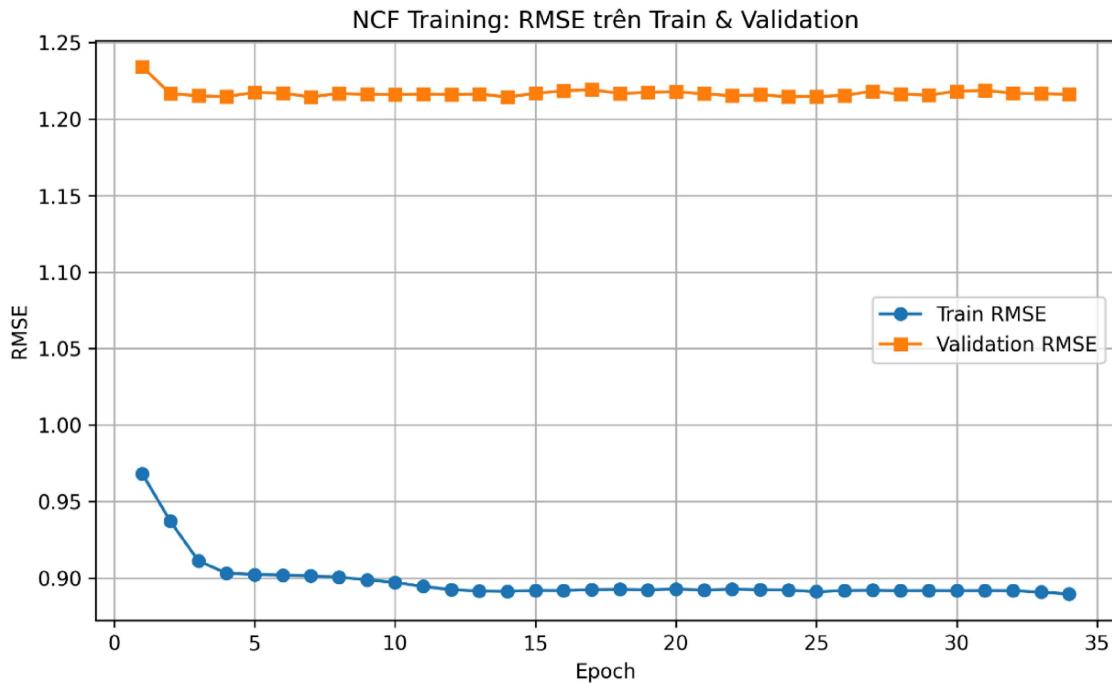
D. NCF (Neural Collaborative Filtering)

Với learning rate = 0.0001, mô hình cho thấy RMSE trên tập huấn luyện giảm mạnh và nhanh chóng đạt mức rất thấp, trong khi RMSE trên tập validation lại tăng dần theo epoch. Hiện tượng này thể hiện rõ ràng tình trạng overfitting: mô hình học thuộc dữ liệu huấn luyện nhưng không khái quát được cho dữ liệu mới.



Hình 7. Biểu đồ RMSE trên tập huấn luyện và kiểm định của mô hình NCF $lr = 0.0001$.

Ngược lại, khi learning rate = 0.001, cả train RMSE và validation RMSE đều ổn định hơn. Mô hình hội tụ nhanh ở giai đoạn đầu, RMSE trên tập huấn luyện giảm đều đến mức hợp lý, còn RMSE trên tập validation giữ ổn định quanh mức thấp (~1.21), chứng tỏ khả năng tổng quát tốt hơn.



Hình 8. Biểu đồ RMSE và MAE trên tập huấn luyện và kiểm định của mô hình NCF $lr = 0.001$.

Do mạng NCF đã được thiết kế khá sâu và phức tạp (3 tầng ẩn [128, 64, 32] kèm BatchNorm và Dropout), năng lực biểu diễn của mô hình đã rất mạnh. Việc thử thêm các hyperparameter khác (như tăng số tầng, thay đổi dropout hay LR khác) nhiều khả năng chỉ làm mất ổn định huấn luyện hoặc tăng overfitting, mà không cải thiện đáng kể hiệu suất. Vì vậy LR = 0.001 giúp mô hình cân bằng giữa tốc độ học và khả năng khái quát, trong khi LR = 0.0001 khiến mô hình học quá kỹ dữ liệu train và mất tính tổng quát, kết quả cuối cùng trên tập test đạt MAE = 0.7382, RMSE = 1.0878, Precision@K = 0.7952, và NDCG@K = 0.982

D. SVD (Singular Value Decomposition)

Trong thí nghiệm này, nhóm áp dụng phương pháp phân rã ma trận SVD thuần túy, không điều chỉnh các siêu tham số như số chiều tiềm ẩn hay hệ số regularization. Mục tiêu nhằm đánh giá khả năng nén thông tin và tái tạo ma trận đánh giá dựa trên các thành phần đặc trưng chính (singular values). Trên tập dữ liệu có quy mô nhỏ hơn đáng kể so với bộ sử dụng cho NCF và SGD, SVD đạt **MAE = 0.7694, RMSE = 1.0634, Precision@10 = 0.2064** và **NDCG@K = 0.9651**. Kết quả này cho thấy dù chỉ áp dụng phân rã cơ bản, mô hình vẫn duy trì độ chính xác dự đoán tương đối ổn định. RMSE và MAE ở mức thấp phản ánh khả năng tái tạo tốt, trong khi NDCG@K cao chứng tỏ SVD vẫn sắp xếp sản phẩm phù hợp với sở thích người dùng. Dù Precision@K còn hạn chế và dữ liệu nhỏ làm giảm tính tổng quát, kết quả này vẫn minh chứng cho vai trò nền tảng của SVD trong các hệ thống gợi ý. Việc không tinh chỉnh thêm siêu tham số là hợp lý vì mục tiêu chính là kiểm chứng khả năng biểu diễn tiềm ẩn của mô hình, không phải tối ưu hóa hiệu suất tuyệt đối.

F. LSH (Locality-Sensitive Hashing)

Để tăng tốc việc tìm kiếm các item tương tự dựa trên vector embedding (user hoặc item), nhóm áp dụng **BucketedRandomProjectionLSH** với `bucketLength = 2.0` và `numHashTables = 3`. Tập dữ liệu gồm ~79k item vector 10 chiều, kết hợp metadata như `title`, `main_category` và `store`.

Mô hình được huấn luyện trên toàn bộ tập vector, sau đó thực hiện truy vấn top-5 items gần nhất cho các vector user và item. Kết quả cho thấy LSH trả về các vector rất gần với truy vấn (Euclidean distance gần 0) trong thời gian trung bình ~0.91–1.08 giây, phù hợp với các ứng dụng gợi ý trực tuyến.

Cấu hình hiện tại cân bằng tốt giữa độ chính xác và tốc độ truy vấn, đồng thời giúp tránh tính toán brute-force trên toàn bộ tập dữ liệu lớn.

5.2 Kết quả

Bảng 5.2 tổng hợp hiệu năng của các mô hình gợi ý sản phẩm trên tập dữ liệu *Arts_Crafts_and_Sewing*, bao gồm MAE, RMSE, Precision@10, NDCG@10 và **Overall Score** — chỉ số tổng hợp đánh giá toàn diện khả năng dự đoán và xếp hạng của mô hình.

Mô hình	MAE	RMSE	Precision@10	NDCG@10	Overall Score
SVD	0.7694	1.0634	0.2064	0.9651	0.664
SGD	0.7040	1.0948	0.8878	0.9819	0.882
ALS	0.7635	1.1078	0.7952	0.9821	0.855
NCF	0.7382	1.0878	0.7952	0.9820	0.861

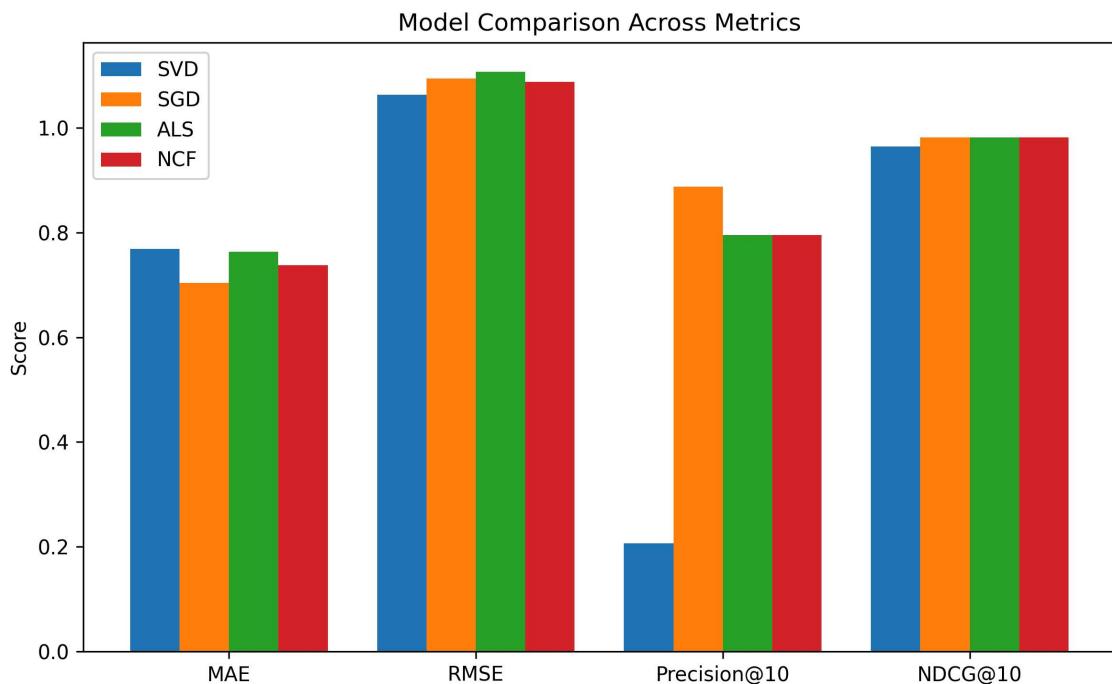
Công thức tính Overall Score:

$$\text{Overall Score} = 0.4 \times \text{RMSE norm} + 0.4 \times \text{Precision}@10 + 0.2 \times \text{NDCG}@10$$

$$\text{RMSE norm} = 1 - \frac{\text{RMSE} - \min(\text{RMSE})}{\max(\text{RMSE}) - \min(\text{RMSE})}$$

Trong đó, RMSEnorm được đảo lại để giá trị RMSE thấp tương ứng với score cao hơn. Overall Score kết hợp cả khả năng dự đoán chính xác rating và khả năng xếp hạng top-K items, cho phép so sánh tổng thể giữa các mô hình.

Từ bảng trên, có thể thấy **SGD đạt Overall Score cao nhất (0.882)**, nhờ kết hợp tốt giữa độ chính xác rating (MAE thấp) và khả năng xếp hạng top-K (Precision@10 cao), trong khi **SVD** có Overall Score thấp nhất (0.664) do Precision@10 kém. **ALS** và **NCF** thể hiện hiệu năng ổn định, với Overall Score lần lượt là 0.855 và 0.861.



Hình 9. So sánh hiệu năng giữa các mô hình gợi ý sản phẩm.

Biểu đồ thể hiện giá trị NDCG@10 và Precision@10 của các mô hình SVD, SGD, ALS, và NCF trên tập dữ liệu Arts_Crafts_and_Sewing.

Mô hình SGD đạt Precision@10 cao nhất và NDCG@10 gần cao nhất, cho thấy khả năng xếp hạng top-K items chính xác hơn so với các mô hình khác. ALS và NCF có hiệu năng ổn định ở cả hai thước đo, với Precision@10 và NDCG@10 gần như bằng nhau, cho thấy cả hai mô hình đều mạnh trong việc duy trì ranking tổng thể. Trong khi đó, SVD có Precision@10 thấp nhất mặc dù NDCG@10 vẫn tương đối cao, chỉ ra rằng mô hình này bị ảnh hưởng rất nhiều bởi nhiễu do ma trận thưa, khó phân biệt các item hàng đầu nhưng vẫn giữ được ranking chung tương đối.

6. Kết luận

Nhóm đã triển khai và so sánh bốn mô hình gợi ý sản phẩm trên tập dữ liệu *Amazon Arts_Crafts_and_Sewing*, bao gồm SVD, SGD, ALS và NCF. Dựa trên chỉ số tổng hợp **Overall Score**, mô hình **SGD** đạt kết quả cao nhất (**0.882**), thể hiện khả năng cân bằng tốt giữa độ chính xác dự đoán (MAE thấp nhất) và khả năng xếp hạng top-K (Precision@10 cao). **ALS** và **NCF** đạt hiệu năng tương đối ổn định với Overall Score lần lượt là **0.855** và **0.861**, cho thấy hai mô hình này vẫn duy trì khả năng xếp hạng tốt dù độ sai số dự đoán cao hơn. Trong khi đó, **SVD** cho thấy hạn chế với Overall Score chỉ **0.664**, chủ yếu do Precision@10 thấp, phản ánh nhược điểm của phương pháp tuyến tính trong việc học mối quan hệ phức tạp giữa người dùng và sản phẩm.

Hướng phát triển tiếp theo của đề tài tập trung vào việc nâng cao chất lượng gợi ý và khả năng ứng dụng trong thực tế. Trước hết, nhóm dự kiến tối ưu hóa các vector embedding của người dùng và sản phẩm, đồng thời tích hợp thêm các đặc trưng nội dung hoặc thông tin meta-data để cải thiện khả năng biểu diễn của mô hình. Bên cạnh đó, hệ thống có thể được mở rộng thành một pipeline hai giai đoạn, trong đó bước đầu sử dụng các phương pháp như LSH hoặc KNN để tạo tập ứng viên (candidate generation), sau đó áp dụng NCF hoặc các mô hình học sâu khác cho bước xếp hạng (ranking), nhằm tăng hiệu quả gợi ý top-K. Cuối cùng, nếu có dữ liệu người dùng thực tế, nhóm sẽ triển khai A/B testing để đánh giá khách quan hiệu quả của mô hình trong môi trường ứng dụng thật.

7. Phụ lục

A. Chuẩn hóa dữ liệu theo item mean

Gọi:

- r_{ui} là rating gốc của user u cho item i ,
- I_u, U_i là tập các user và item tương ứng,
- μ_i là rating trung bình của item i trên tập train:

$$\mu_i = \frac{1}{|U_i|} \sum_{u \in U_i} r_{ui}.$$

Sau đó, rating được chuẩn hóa thành:

$$r^{\text{norm}} * ui = r * ui - \mu_i.$$

- Nếu item mới xuất hiện trong tập validation hoặc test mà không có giá trị trung bình từ train, nhóm sử dụng **global mean** μ_{global} :

$$r^{\text{norm}} * ui = \begin{cases} r * ui - \mu_i, & \text{nếu } i \in \text{train items}, \\ r_{ui} - \mu_{\text{global}}, & \text{nếu } i \notin \text{train items}. \end{cases}$$

B. Công thức cập nhật trong SGD

Ta tối thiểu hóa hàm mất mát (cho toàn bộ dữ liệu quan sát được):

$$\mathcal{L} = \sum_{(u,i) \in \mathcal{D}} \frac{1}{2} (r_{ui} - p_u^\top q_i)^2 + \frac{\lambda}{2} (\|p_u\|^2 + \|q_i\|^2)$$

đặt sai số (prediction error) cho một cặp (u, i) là

$$e_{ui} = r_{ui} - p_u^\top q_i.$$

Với chuẩn viết có hệ số $1/2$ phía trước bình phương, gradient theo p_u và q_i của thành phần mất mát tại cặp (u, i) là:

$$\begin{aligned} \frac{\partial}{\partial p_u} \left(\frac{1}{2} e_{ui}^2 + \frac{\lambda}{2} \|p_u\|^2 \right) &= -e_{ui} q_i + \lambda p_u, \\ \frac{\partial}{\partial q_i} \left(\frac{1}{2} e_{ui}^2 + \frac{\lambda}{2} \|q_i\|^2 \right) &= -e_{ui} p_u + \lambda q_i. \end{aligned}$$

Áp dụng quy tắc cập nhật SGD với learning rate (η) (cập nhật theo hướng ngược gradient):

$$\begin{aligned} p_u &\leftarrow p_u - \eta (-e_{ui} q_i + \lambda p_u) = p_u + \eta (e_{ui} q_i - \lambda p_u), \\ q_i &\leftarrow q_i - \eta (-e_{ui} p_u + \lambda q_i) = q_i + \eta (e_{ui} p_u - \lambda q_i). \end{aligned}$$

Ghi chú về các dạng khác nhau của công thức: Một số hàm mất mát không có hệ số $1/2$ (ví dụ $(r - \hat{r})^2 + \lambda |\cdot|^2$), thì gradient sẽ chứa nhân tố 2 và cập nhật sẽ có thêm hệ số 2 . Người thực nghiệm thường gộp hệ số 2 vào η (hoặc dùng biểu thức có $1/2$ như trên) để có dạng cập nhật đơn giản như trình bày (nhưng cuối cùng về mục tiêu tối ưu vẫn như nhau).

C. Công thức cập nhật trong ALS (Alternating Least Squares)

Ý tưởng là tách bài toán tối ưu đồng thời theo P và Q thành hai bước lặp: cố định Q rồi tối ưu P (từng hàng p_u một), sau đó cố định P và tối ưu Q . Với định nghĩa mất mát toàn cục

$$\mathcal{L} = \sum_{(u,i) \in \mathcal{D}} (r_{ui} - p_u^\top q_i)^2 + \lambda \left(\sum_u \|p_u\|^2 + \sum_i \|q_i\|^2 \right),$$

xét một user cụ thể u . Gọi I_u là tập các item mà user u đã đánh giá; xây hai đối tượng:

- $r_u \in \mathbb{R}^{|I_u|}$: vector giá trị rating quan sát (thứ tự tương ứng với I_u);
- $Q_u \in \mathbb{R}^{|I_u| \times K}$: ma trận có các hàng là q_i^\top với $i \in I_u$.

Khi cố định mọi q_i , bài toán đối với p_u là:

$$\min_{p_u} \|r_u - Q_u p_u\|^2 + \lambda \|p_u\|^2.$$

Đây là bài toán least squares đã có công thức giải, công thức có dạng:

$$(Q_u^\top Q_u + \lambda I_K) p_u = Q_u^\top r_u.$$

$$p_u = (Q_u^\top Q_u + \lambda I)^{-1} Q_u^\top r_u,$$

Trong thực thi, $Q_u^\top Q_u = \sum_{i \in I_u} q_i q_i^\top$ (ma trận $K \times K$) và $Q_u^\top r_u = \sum_{i \in I_u} r_{ui}, q_i$ (vector kích thước K).

Tương tự, khi cố định P ta có:

$$q_i = (P_i^\top P_i + \lambda I)^{-1} P_i^\top r_i$$

với P_i ma trận gồm các p_u^\top cho user đã đánh giá item i , và r_i là vector rating quan sát cho item i .

8. Đóng góp

Tất cả các thành viên trong nhóm đều cố gắng hết sức để hoàn thành dự án một cách công bằng, đảm bảo mỗi người đều đóng góp vào các phần công việc quan trọng.

- **Nguyễn Hoàng Phúc (10/10)**: chịu trách nhiệm tiền xử lý dữ liệu, triển khai thuật toán ALS và LSH và đánh giá, đồng thời biên tập và soạn thảo báo cáo.
- **Phạm Trung Kỳ (10/10)**: triển khai các thuật toán SVD, SGD và NCF, thực hiện các thí nghiệm, đánh giá, đồng thời tham gia soạn thảo báo cáo.

9. Tham khảo

Deisenroth, M. P., Faisal, A. A., & Ong, C. S. (2020). *Mathematics for Machine Learning* (pp. 111–134). Cambridge University Press.

GeeksforGeeks. (2025, September 30). *Stochastic Gradient Descent*.
<https://www.geeksforgeeks.org/machine-learning/ml-stochastic-gradient-descent-sgd/>

CodeSignal. (2025). *Implementing the Alternating Least Squares Algorithm*.
<https://codesignal.com/learn/courses/diving-deep-into-collaborative-filtering-techniques-with-als/lessons/implementing-the-alternating-least-squares-algorithm>

Pinecone. (2025). *Locality Sensitive Hashing (LSH): The Illustrated Guide.*

<https://www.pinecone.io/learn/series/faiss/locality-sensitive-hashing/>

GeeksforGeeks. (2025, July 23). *Overfitting and Regularization in ML.*

<https://www.geeksforgeeks.org/machine-learning/overfitting-and-regularization-in-ml/>

Apache Spark. (2025). *Machine Learning Library (MLlib) Guide.*

<https://spark.apache.org/docs/latest/ml-guide.html>