Trường Đại học Bách Khoa Hà Nội

Viện Công nghệ Thông Tin và Truyền Thông

Đồ án Tốt nghiệp Đại học

Tìm hiểu về các thuật toán gợi ý

Trần Thị Hải Hà

Hà Nội, 05/2019

Trường Đại học Bách Khoa Hà Nội

Viện Công nghệ Thông Tin và Truyền Thông

Đồ án Tốt nghiệp Đại học

Tìm hiểu về các thuật toán gợi ý

Sinh viên thực hiện Trần Thị Hải Hà

Người hướng dẫn ThS. Ngô Văn Linh

Hà Nội, 05/2019

Lời cam kết

Họ và tên sinh viên: . . . . . .

Điện thoại liên lạc: . . . . . Email: . . . . . .

Lớp: . . . . . Hệ đào tạo: . . . . . .

Tôi – *Trần Thị Hải Hà* – cam kết Đồ án Tốt nghiệp (ĐATN) là công trình nghiên cứu của bản thân tôi dưới sự hướng dẫn của *ThS. Ngô Văn Linh*. Các kết quả nêu trong ĐATN là trung thực, là thành quả của riêng tôi, không sao chép theo bất kỳ công trình nào khác. Tất cả những tham khảo trong ĐATN – bao gồm hình ảnh, bảng biểu, số liệu, và các câu từ trích dẫn – đều được ghi rõ ràng và đầy đủ nguồn gốc trong danh mục tài liệu tham khảo. Tôi xin hoàn toàn chịu trách nhiệm với dù chỉ một sao chép vi phạm quy chế của nhà trường.

Hà Nội, ngày tháng năm

Tác giả ĐATN

Họ và tên sinh viên

Lời cảm ơn

Đầu tiên, em xin được gửi lời cảm ơn chân thành đến các thầy giáo, cô giáo của trường Đại học Bách Khoa Hà Nội, đặc biệt là các thầy giáo, cô giáo của Viện Công nghệ thông tin và truyền thông. Trong suốt những năm học tập tại trường, các thầy, các cô đã dạy cho em vô vàn những kiến thức vô cùng quý báu và bổ ích. Đây sẽ là hành trang tuyệt vời giúp em tự tin trước ngưỡng cửa tương lai đầy thử thách. Đồng thời, em cũng xin được gửi lời cảm ơn đặc biệt đến thầy giáo, ThS.Ngô Văn Linh, người đã luôn tích cực hướng dẫn và giúp đỡ em rất nhiều để em có thể hoàn thành đồ án tốt nghiệp này. Một lần nữa, em xin trân trọng cảm ơn thầy!

Em cũng xin được gửi lời cảm ơn đến gia đình và bạn bè, những người đã giúp đỡ em trong suốt thời gian qua! Cảm ơn mọi người rất nhiều!

Tóm tắt

Recommendation Systems, hệ gợi ý, là hệ thống có khả năng dự đoán sở thích, thói quen của người dùng dựa trên hành vi của người dùng trong quá khứ. Hệ gợi ý là một bài toán rất thực tế và có tính ứng dụng cao. Hiện nay, bài toán này đang ngày càng thu hút được sự quan tâm của các lập trình viên theo đuổi Artificial intelligence (Trí tuệ nhân tạo) nói chung và các lập trình viên theo đuổi Recommendation Systems nói riêng. Vì cảm thấy đây là một bài toán rất thú vị và bổ ích nên em đã lựa chọn đề tài "Tìm hiểu về các thuật toán gợi ý" để thực hiện đồ án lần này.

Trong đồ án, em sẽ giới thiệu và trình bày sơ lược về hệ gợi ý. Từ đó, trình bày một số thuật toán đã và đang được sử dụng trong hệ gợi ý, bao gồm: Content-based, Neighborhood-based và Matrix Factorization. Sau đó, cài đặt thử nghiệm các phương pháp này để so sánh và đánh giá độ chính xác và tính tin cậy của từng phương pháp.

Do kiến thức của em còn hạn chế nên báo cáo không tránh khỏi còn thiếu sót. Em mong nhận được những góp ý của các thầy cô để báo cáo thêm hoàn thiện.

Mục lục

Danh mục hình vẽ

Danh mục bảng

Danh mục các từ viết tắt

STT	Tên viết tắt	Tên đầy đủ
1	RS	Recommender system (Recommendation System)
2	CB	Content-based
3	CF	Collaborative Filtering
4	NBCF	Neighborhood-based Collaborative Filtering
5	uuCF	User-user Collaborative Filtering
6	iiCF	Item-item Collaborative Filtering
7	MF	Matrix Factorization
8	RMSE	Root Mean Squared Error
9	MLs	Tập dữ liệu MovieLens
10	1M(0.1)	Tập MovieLens 1M với tỉ lệ test/train là 0.1
11	1M(0.3)	Tập MovieLens 1M với tỉ lệ test/train là 0.3

Danh mục thuật ngữ

STT	Thuật ngữ	Ý nghĩa
1	Artificial Intelligence	Trí tuệ nhân tạo
2	Machine Learning	Học máy
3	Recommender system(s) Recommendation System(s)	Hệ gợi ý
4	Content-based Filtering	Phương pháp gợi ý dựa trên nội dung
5	Collaborative Filtering	Phương pháp gợi ý dựa trên lọc cộng tác
6	Neighborhood-based Collaborative Filtering	Phương pháp gợi ý dựa trên hàng xóm gần nhất
7	User-user Collaborative Filtering	Phương pháp gợi ý dựa trên sự tương đồng của các users (người dùng)
8	Item-item Collaborative Filtering	Phương pháp gợi ý dựa trên sự tương đồng của các items
9	Matrix Factorization	Phương pháp gợi ý bằng kỹ thuật phân rã ma trận
10	Utility Matrix	Ma trận biểu diễn các giá trị ratings của các users cho các items
11	Normalized Utility Matrix	Ma trận Utility đã được chuẩn hóa

12	Cosine Similarity	Độ đo khoảng cách giữa các vector bằng hàm cosine
13	Pearson Corelation (Pearson)	Độ đo khoảng cách giữa các vector bằng hàm pearson
14	Similarity Matrix	Ma trận khoảng cách
15	Gradient Descent	Phương pháp tối ưu hàm mất mát
16	Regression	Bài toán hồi quy
17	Classification	Bài toán phân lớp
18	Linear Regression	Mô hình hồi quy tuyến tính
19	Root Mean Squared Error	Sai số toàn phương trung bình
20	Precision	Độ chính xác
21	Recall	Độ hồi tưởng
22	View-enhanced eALS	Phương pháp gợi ý dựa trên mô hình hóa sự kiện
23	Neural Matrix Factorization	Phương pháp gợi ý kết hợp phân tích ma trận và mạng nơ-ron
24	Co-rating Model	Phương pháp gợi ý kết hợp phân tích phản hồi ẩn và phản hồi tường minh

Chương 1 Giới thiệu đề tài

1.1 Đặt vấn đề

Xã hội ngày càng phát triển, chất lượng cuộc sống của chúng ta cũng ngày càng được nâng cao. Giờ đây, thay vì phải đến tận cửa hàng để mua sắm, chúng ta hoàn toàn có thể đặt những món đồ mà mình yêu thích dù ở bất kì đâu, chỉ cần có Internet. Những món đồ đó có thể là đồ ăn, đồ uống, đồ gia dụng, quần áo, dịch vụ giải trí, etc.

Khi sử dụng những dịch vụ như vậy, bạn sẽ thấy rất nhiều hệ thống có mục gợi ý các sản phẩm, ví dụ như:

- Gợi ý phim trên Netflix
- Gợi ý video trên Youtube
- Gợi ý kết bạn, new feed, quảng cáo,… trên Facebook
- Gợi ý "Frequently bought together" trên Amazon

Hình 1 Ví dụ về gợi ý phim trên Netflix

(Nguồn: https://images.techhive.com/images/article/2013/08/mylist-100050912-large.jpg)

Những thuật toán đằng sau những tính năng đó chính là những thuật toán Machine Learning, có tên gọi chung là Recommender Systems hay Recommendation Systems, tức

là hệ gợi ý. Những hệ gợi ý này sẽ giúp tăng trải nghiệm người dùng, biến khách hàng ảo thành khách hàng thật, từ đó làm tăng hiệu quả sử dụng của hệ thống.

Tuy nhiên, lượng dữ liệu trên Internet rất lớn, người dùng thì lại vô cùng đa dạng, thông tin phản hồi tường minh của người dùng thì khan hiếm và khó thu thập làm bài toán gợi ý ngày càng trở nên phức tạp hơn. Chính vì vậy, bài toán hệ gợi ý, đã và đang là một thách thức thú vị và ngày càng nhận được nhiều sự quan tâm của cả các nhà nghiên cứu và doanh nghiệp.

1.2 Định hướng và giải pháp

Ý tưởng của hệ gợi ý là mô hình hóa hành động của người dùng dựa trên lịch sử tương tác (như xem, click, bình luận, đánh giá,…) của họ từ đó đưa ra dự đoán phù hợp với người dùng. Cụ thể, hệ gợi ý thường sử dụng hai chiến lược chính, đó là: content-based filtering (lọc dựa trên nội dung) và collaborative filtering (lọc cộng tác) [7].

Content-based filtering: Gợi ý các item dựa vào hồ sơ (profiles) của user (người dùng) hoặc dựa vào nội dung/thuộc tính (attributes) của những items tương tự như những items mà user đã chọn trong quá khứ [1][2]. Phương pháp này cần thông tin tường minh và khó thu thập.

Collaborative filtering: Gợi ý các items dựa trên sự tương quan (similarity) giữa các users và/hoặc items [1][2]. Mục đích của phương pháp này phân tích sự tương đồng của users và sự tương đồng của items, nhằm phát hiện ra những tương tác user-item tiềm năng [6]. Phương pháp này sử dụng phản hồi trong quá khứ của người dùng về sản phẩm, có thể là phản hồi tường minh (như đánh giá, bình luận, …) hoặc phản hồi ẩn (như click, xem,…).

Trong thực tế, dữ liệu tường minh của người dùng khó thu thập và cần khá nhiều thời gian. Vì vậy, chiến lược lọc cộng tác đang ngày càng được nghiên cứu và ứng dụng nhiều hơn trong các dự án thực tế.

Trong đồ án này, em tìm hiểu về 3 mô hình, là: Content-based Filtering, Neighborhood-based Filtering (NBCF) và Matrix Factorixzation (MF) để hiểu hơn về đặc điểm của các thuật toán. Đồng thời, cài đặt thử nghiệm và so sánh, đánh giá các phương pháp về hiệu quả và tính tin cậy của từng thuật toán.

1.3 Bố cục đồ án

Phần còn lại của báo cáo đồ án tốt nghiệp này được tổ chức như sau.

Chương 2 trình bày các khái niệm tổng quan về hệ gợi ý, bao gồm khái niệm và thành phần của hệ gợi ý. Sau đó, trình bày các phương pháp gợi ý và phân tích thuật toán của từng phương pháp. Trong đồ án này, em tìm hiểu về 3 phương pháp là Content-based Filtering, Neighborhood-based Collaborative Filtering và Matrix Factorization Collaborative Filtering.

Tiếp theo, Chương 3 sẽ trình bày về phương pháp cài đặt và đánh giá các thuật toán. Trong phần này, em sẽ mô tả tập dữ liệu Movielens được sử dụng để nghiên cứu và phương pháp đánh giá từng thuật toán dựa trên tập dữ liệu MovieLens. Sau đó, trình bày kết quả thu được sau khi cài đặt thử nghiệm.

Cuối cùng, trong Chương 4, em sẽ trình bày kết luận đánh giá những kết quả thu được của quá trình nghiên cứu. Từ đó đưa ra hướng phát triển tiếp theo cho đồ án.

Chương 2 Tổng quan về hệ gợi ý

2.1 Hệ gợi ý

2.1.1 Khái niệm

Hệ thống gợi ý (Recommeder System – RS) là một dạng của hệ thống lọc thông tin (Information Filtering), thường được sử dụng để dự đoán sở thích (preferences) hoặc đánh giá (rating) của mỗi người dùng (user) đối với một mục thông tin (item) nào đó mà họ chưa xem xét tới trong quá khứ.

Hệ gợi ý có 2 thực thể chính là **users** và **items**. Trong đó, **users** là người dùng; **items** là sản phẩm, ví dụ như các bài hát, sách, video, bộ phim,… hoặc cũng có thể là các users khác (trong gợi ý kết bạn).

Mục đích chính của hệ gợi ý là dự đoán mức độ quan tâm của mỗi user với các item. Từ đó, có chiến lược gợi ý phù hợp với mỗi user. Điều này sẽ giúp cải thiện trải nghiệm của người dùng trên hệ thống, từ đó biến khách hàng tiềm năng trở thành khách hàng thật, làm tăng hiệu năng hoạt động và doanh thu của hệ thống.

2.1.2 Ma trận Utility Matrix và bài toán gợi ý

Trong các hệ gợi ý, có 2 thực thể chính là **users** và **items**. Mỗi user sẽ có mức độ quan tâm (*degree of preference*) tới mỗi item là khác nhau. Mức độ quan tâm này, nếu đã biết trước, sẽ được gán cho một giá trị ứng với mỗi cặp user-item. Giả sử rằng mức độ quan tâm được đo bằng giá trị user rate cho item, tạm gọi giá trị này là **rating**. Khi đó, tập hợp tất cả các **ratings**, bao gồm cả những giá trị chưa biết cần được dự đoán, sẽ tạo nên một ma trận gọi là ma trận **utility matrix** [1].

Ví dụ:

	A	B	C	D	E	F
Mưa nửa đêm	5	5	0	0	1	?
Cỏ úa	5	?	?	0	?	?
Vùng lá me bay	?	4	1	?	?	1
Con cò bé bé	1	1	4	4	4	?
Em yêu trường em	1	0	5	?	?	?

Hình 2 Ví dụ về ma trận Utility Matrix

(Nguồn https://machinelearningcoban.com/assets/23_contentbasedrecommendersys)

Hình 2 là một ví dụ về utility matrix. Trong đó, có 6 users là A, B, C, D, E, F và 5 bài hát. Giá trị trong các ô màu xanh là mức độ yêu thích của mỗi user tương ứng với mỗi bài hát theo *ratings* từ 0 (không thích) đến 5 (yêu thích). Còn các ô màu hồng, chứa dấu '?', thể hiện các bài hát chưa được người dùng đáng giá. Công việc của một RS chính là dự đoán các giá trị khuyết thiếu này, từ đó gợi ý cho người dùng những items có ratings dự đoán là tốt nhất. Vì vậy, RS, đôi khi cũng được coi là bài toán *Matrix Completion* (*Hoàn thiện ma trận*) [1].

Các thành phần cơ bản của Bài toán gợi ý bao gồm:

Đầu vào: Các tập dữ liệu về users, items và một số feedback của các users cho các items. Ví dụ, trong bài toán gợi ý phim thì chúng ta cần có danh sách users; danh sách items là các bộ phim và feedback của user về các bộ phim, có thể là số lượt click, số lượt xem, nội dung bình luận hoặc số sao đánh giá,…

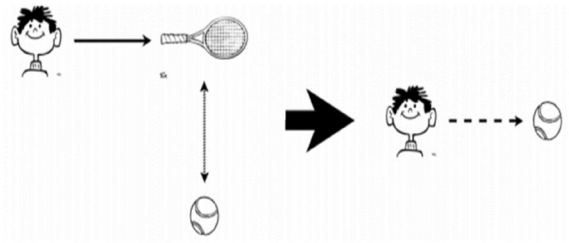
Đầu ra: Các giá trị dự đoán đánh giá của mỗi user về các items mà user đó chưa từng xét tới trong quá khứ. Sau đó, xếp hạng items theo đánh giá để đưa ra tập các items phù hợp để gợi ý cho mỗi user. Mức độ quan tâm của người dùng đối với những gợi ý đó càng cao thì chứng tỏ hệ thống gợi ý càng hiệu quả. Vì vậy, mục tiêu của bài toán này là gợi ý ra những items phù hợp nhất với mỗi user.

Phạm vi áp dụng: Phạm vi áp dụng của bài toán gợi ý vô cùng đa dạng. Điển hình là trong thương mại điện tử, tin tức, giải trí trực tuyến. Ví dụ như gợi ý kết bạn trên mạng xã hội, gợi ý sản phẩm trong thương mại điện tử, gợi ý phim, video, bài hát,… trong giải trí trực tuyến, gợi ý tin tức,…

2.2 Một số phương pháp của hệ gợi ý

2.2.1 Content-based Filtering

Content-based Filtering: Gợi ý các item dựa vào hồ sơ (profiles) của user (người dùng) hoặc dựa vào nội dung/thuộc tính (attributes) của những items tương tự như item mà user đã chọn trong quá khứ [1][2].



Hình 3 Mô tả ý tưởng của Content-based filtering

(Nguồn https://viblo.asia/p/gioi-thieu-ve-he-thong-goi-y-recommender-systems-hoac-recommendation-systems-maGK78yOZj2)

Ví dụ trong Hình 3, nếu có một người đã mua một cây vợt, thì khả năng cao là người đó sẽ cần mua thêm một quả bóng. Vì vậy, hệ thống sẽ gợi ý cho người dùng mua thêm quả bóng. Tương tự như vậy, khi bạn đang xem tập 1 của bộ phim hoạt hình Doraemon, hệ thống sẽ dự đoán bạn sẽ muốn xem tiếp tập 2, khi đó, hệ thống sẽ gợi ý cho bạn tập 2 của bộ phim đó.

Ý tưởng chính của phương pháp này là:

- Biểu diễn mỗi item dưới dạng một feature vector (vec-tơ đặc trưng). Sau đó, sử dụng các vector đó để tính toán độ tương đồng giữa các items.
- Gợi ý cho mỗi user các items có độ tương đồng cao với item mà user đã lựa chọn trong quá khứ.

Mô tả thuật toán:

1. **Xây dựng Item Profile**

   Đầu tiên, chúng ta cần xây dựng *profile* (hồ sơ) cho mỗi item. Profile này được biểu diễn dưới dạng toán học là một "feature vector" n chiều. Trong những trường hợp đơn giản, fearute vector có thể được trích xuất trực tiếp từ *item* [1].

Ví dụ, để xây dựng feature vector cho các items là các bộ phim, chúng ta có thể xét các thuộc tính, như: tên phim, tên series, thể loại, đạo diễn, năm sản xuất, diễn viên chính, mô tả tóm tắt, số lượt xem, điểm đánh giá trung bình, số lượt bình luận,…

Giả sử chúng ta sẽ đơn giản hóa các bài hát thành các feature vector 2 chiều, trong đó: một chiều là mức độ Bolero, một chiều là mức độ Thiếu nhi của bài hát. Với x_1, x_2, x_3, x_4, x_5 lần lượt là các fearute vector tương ứng của mỗi bài hát, chúng ta sẽ có biểu diễn của các vector như trong Hình 4:

	A	B	C	D	E	F	item's feature vectors
Mưa nửa đêm	5	5	0	0	1	?	$\mathbf{x}_1 = [0.99, 0.02]$
Cỏ úa	5	?	?	0	?	?	$\mathbf{x}_2 = [0.91, 0.11]$
Vùng lá me bay	?	4	1	?	?	1	$\mathbf{x}_3 = [0.95, 0.05]$
Con cò bé bé	1	1	4	4	4	?	$\mathbf{x}_4 = [0.01, 0.99]$
Em yêu trường em	1	0	5	?	?	?	$\mathbf{x}_5 = [0.03, 0.98]$
User's models	θ_1	θ_2	θ_3	θ_4	θ_5	θ_6	← **need to optimize**

Hình 4 Ví dụ về feature vector

(Nguồn https://machinelearningcoban.com/2017/05/17/contentbasedrecommendersys/)

Sau khi có feature vector của từng item, chúng ta sẽ đi tìm mô hình θ_i cho mỗi user, để dự đoán mức độ yêu thích của user đó với mỗi item.

2. Xây dựng hàm mất mát

Sau khi có tập các vec-tơ đặc trưng, chúng ta cần lựa chọn một mô hình để tính toán độ quan tâm của user với mỗi item. Đây có thể là bài toán Regression, nhằm tìm kiếm rating thuộc một khoản gía trị hoặc bài toán Classification, nhằm gán nhãn "quan tâm" hoặc "không quan tâm" của user với mỗi item, tùy thuộc vào từng ứng dụng [1].

Trong đồ án này, em đã sử dụng mô hình Linear Regression, mô hình tuyến tính, để thử nghiệm phương pháp này.

Xét bài toán với (i) N là số users, (ii) M là số items, (iii) Y là ma trận utility matrix, (iv) R là ma trận *rated or not* thể hiện việc một *user* đã đánh giá một *item* hay chưa. Cụ thể, rij = 1 nếu *item* thứ i đã được đánh giá bởi *user* thứ j, ngược lại rij = 0 nếu *item* thứ i chưa được đánh giá bởi *user* thứ j.

Giả sử, ta có thể tìm được một mô hình (vector cột w_i và tham số bias b_n), khi đó, ta có thể tính được mức độ quan tâm của mỗi user với mỗi item bằng một hàm tuyến tính:

$$\boldsymbol{y_{mn} = x_m w_n + b_n} \tag{2.1}$$

trong đó, x_m là vec-tơ đặc trưng của item thứ m.

Xét một *user* thứ n bất kỳ, nếu ta coi tập dữ liệu huấn luyện là tập hợp các thành phần đã được *điền* của y_n, ta có thể xây dựng hàm mất mát tương tự như sau:

$$\boldsymbol{\mathcal{L}_n = \frac{1}{2} \sum_{m\,:\,r_{mn}=1} (x_m w_n + b_n - y_{mn})^2 + \frac{\lambda}{2} \|w_n\|_2^2} \tag{2.2}$$

trong đó, thành phần thứ hai gồm $\|\circ\|_2^2$ và λ là một tham số dương.

Trong thực nghiệm, trung bình cộng của *lỗi* thường được dùng, và hàm mất mát $\mathcal{L}_n$ được viết lại thành:

$$\boldsymbol{\mathcal{L}_n = \frac{1}{2s_n} \sum_{m\,:\,r_{mn}=1} (x_m w_n + b_n - y_{mn})^2 + \frac{\lambda}{s_n} \|w_n\|_2^2} \tag{2.3}$$

,trong đó s_n là số lượng các *items* mà *user* thứ n đã đánh giá. Nói cách khác, s_n là tổng các phần tử trên cột thứ n của ma trận *rated or not* R:

$$\boldsymbol{s_n = \sum_{m=1}^{M} r_{mn}} \tag{2.4}$$

Vì hàm mục tiêu chỉ phụ thuộc vào các *items* đã được đánh giá, nên ta có thể rút gọn nó bằng cách đặt $\hat{y}_n$ là *sub vector* của y_n, được xây dựng bằng cách trích xuất các thành phần *khác dấu "?"* ở cột thứ n, tức lấy ra các thành phần đã được đánh giá bởi *user* thứ n trong ma trận Y. Đồng thời, đặt $\hat{X}_n$ là *sub matrix* của ma trận X, được tạo bằng cách trích xuất các hàng tương ứng với các *items* đã được *rated* bởi *user* thứ n. Khi đó, biểu thức hàm mất mát của mô hình cho *user* thứ n được viết gọn lại thành:

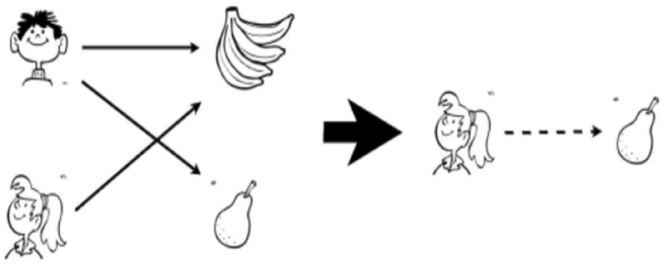
$$\boldsymbol{\mathcal{L}_n = \frac{1}{2s_n} \left\|\hat{X}_n w_n + b_n e_n - \hat{y}_n\right\|_2^2 + \frac{\lambda}{2s_n} \|w_n\|_2^2} \tag{2.5}$$

trong đó, e_n là vector cột chứa s_n phần tử 1.

Cuối cùng, sử dụng Gradient Descent để tìm được w_n và b_n. Trong đồ án này, em đã sử dụng class Ridge của thư viện sklearn.linear_model để thử nghiệm.

2.2.2 Neighborhood-based Collaborative Filtering (NBCF)

Neighborhood-based Collaborative Filtering: Gợi ý các items dựa trên sự tương quan (similarity) giữa các users và/hoặc items [1][2]. Có thể hiểu rằng đây là cách gợi ý các items tới một user dựa trên những users có hành vi tương tự.



Hình 5 Mô tả ý tưởng của NBCF

(Nguồn https://viblo.asia/p/gioi-thieu-ve-he-thong-goi-y-recommender-systems-hoac-recommendation-systems-maGK78yOZj2)

Như ví dụ, có một cậu bé thích chuối và lê, đồng thời, có một cô bé có sở thích tương tự như cậu bé. Khi đó, khả năng cao là cô bé cũng sẽ thích lê giống như cậu bé, vì vậy hệ thống sẽ gợi ý lê cho cô bé. Tương tự, nếu trên một hệ thống xem phim, có user A và B đều cùng xem một số bộ phim. Khi user A đã xem một bộ phim mà user B chưa xem, rất có khả năng là user B cũng sẽ muốn xem bộ phim ấy, do đó, hệ thống sẽ gợi ý bộ phim ấy cho user B.

Phương pháp này được chia thành 2 hướng nhỏ hơn, đó là User-User Collaborative Filtering (uuCF) và Item-Item Collaborative Filtering (iiCF).

uuCF : Tư tưởng của hướng tiếp cận này là tìm ra những nhóm users tương tự nhau. Từ đó, dự đoán mức độ yêu thích của một user dựa trên những users khác cùng nhóm.

Hướng tiếp cận này được thực hiện như sau:

- Biểu diễn mỗi user bằng một feature vector (vec-tơ đặc trưng) được xây dựng từ những feedback trong quá khứ của user với các items. Sau đó, sử dụng những vector này để tính toán độ tương đồng giữa các users.
- Để tính toán độ yêu thích của một user U cho một item I, ta sẽ lựa chọn ra k users đã từng đánh giá I và có độ tương đồng với user U là cao nhất. Sau đó, dựa vào những feedback của k user đó với item I để tính toán ra kết quả.

- Cuối cùng, lựa chọn những items được dự đoán là được user U yêu thích nhất để gợi ý cho U.

iiCF: Tương tự như uuCF, phương pháp này sẽ tìm ra những nhóm items tương tự nhau. Sau đó, dự đoán mức độ yêu thích của user với item dựa trên độ yêu thích của user đó với các items khác cùng loại.

Hướng tiếp cận này được thực hiện như sau:

- Biểu diễn mỗi item bằng một feature vector (vec-tơ đặc trưng). Sau đó, tính toán độ tương đồng giữa các items bằng cách tính khoảng cách giữa các vector tương ứng.
- Để tính toán độ yêu thích của một user U cho một item I, ta sẽ lựa chọn ra k items đã từng được U đánh giá và có độ tương đồng với I là cao nhất. Sau đó, dựa vào những feedback của U với k items đó để tính toán ra kết quả.
- Cuối cùng, lựa chọn những items được dự đoán là phù hợp nhất với user U để gợi ý.

Trong thực tế, vì số lượng items thường nhỏ hơn nhiều so với số lượng users nên phương pháp tính toán độ tương đồng của iiCF sẽ nhanh hơn so với uuCF.

2.2.2.1 User – User Collaborative Filtering

1. Chuẩn hóa ma trận Utility Matrix

Dữ liệu duy nhất chúng ta sử dụng cho phương pháp này là ma trận utility matrix:

	U_0	U_1	U_2	U_3	U_4	U_5	U_6
i_0	5	5	2	0	1	?	?
i_1	4	?	?	0	?	2	?
i_2	?	4	1	?	?	1	1
i_3	2	2	3	4	4	?	4
i_4	2	0	4	?	?	?	5

Hình 6 Ma trận Utility Matrix ban đầu

(Nguồn https://machinelearningcoban.com/2017/05/24/collaborativefiltering/)

Để có thể sử dụng ma trận này vào việc tính toán, chúng ta cần thay những dấu '?' bởi một giá trị. Đơn giản nhất thì chúng ta có thể thay vào đó giá trị '0'. Hoặc dùng một giá trị khác là '2.5' – giá trị trung bình giữa 0 và 5. Tuy nhiên, những cách này không thực sự tốt vì những giá trị này sẽ hạn chế với những users dễ tính hoặc khó tính. Với những

users dễ tính, thích tương ứng với 5 sao, không thích sẽ ít sao hơn, như 2 hoặc 3 sao. Khi đó, '2.5' sẽ làm những đánh giá không thích sẽ trở thành negative. Ngược lại, với những user khó tính, họ thậm chí chỉ đánh 3 sao khi thích và dưới 3 khi không thích. Vì vậy, hợp lý hơn cả, chúng ta sẽ sử dụng giá trị trung bình cộng ratings của mỗi user [1].

Cụ thể, chúng ta sẽ xử lý như sau:

	U_0	U_1	U_2	U_3	U_4	U_5	U_6
i_0	5	5	2	0	1	?	?
i_1	4	?	?	0	?	2	?
i_2	?	4	1	?	?	1	1
i_3	2	2	3	4	4	?	4
i_4	2	0	4	?	?	?	5
θ_j	3.2	2.75	2.5	1.33	2.5	1.5	3.33

Hình 7 Ma trận Utility gốc và trung bình cộng ratings của các users

(Nguồn https://machinelearningcoban.com/2017/05/24/collaborativefiltering/)

Như trong Hình 7, chúng ta có một ví dụ về trung bình cộng ratings của mỗi users. Tuy nhiên, thay vì trực tiếp sử dụng các giá trị này thay cho các dấu '?' tương ứng với mỗi user. Chúng ta sẽ trừ ratings của mỗi user cho giá trị trung bình ratings tương ứng của user đó và thay dấu '?' bằng giá trị 0. Mục đích của cách xử lý này là [1]:

- Phân loại ratings thành 2 loại: giá trị âm (user không thích item) và giá trị dương (user thích item). Các giá trị bằng 0, tương ứng với những item chưa được đánh giá.
- Số chiều của utility matrix thường rất lớn, trong khi lượng ratings biết trước thường rất nhỏ so với kích thước của toàn bộ ma trận. Nếu thay dấu '?' bằng '0' thì chúng ta có thể sử dụng sparse matrix, tức ma trận chỉ lưu các giá trị khác 0 và vị trí của giá trị đó. Như vậy, việc lưu trữ sẽ tối ưu hơn.

Ma trận sau khi chuẩn hóa được gọi là normalized utility matrix:

	U_0	U_1	U_2	U_3	U_4	U_5	U_6
i_0	1.75	2.25	-0.5	-1.33	-1.5	0	0
i_1	0.75	0	0	-1.33	0	0.5	0
i_2	?	1.2	-1.5	0	0	-0.5	-2.33
i_3	-1.25	-0.75	0.5	2.67	1.5	0	0.67
i_4	-1.25	-2.75	1.5	0	0	0	1.67

Hình 8 Ma trận Normalized Utility Matrix

(Nguồn https://machinelearningcoban.com/2017/05/24/collaborativefiltering/)

2. Similarity Function

Sau khi chuẩn hóa ma trận utility, chúng ta cần tính toán độ tương đồng giữa các users.

Xét tiếp ví dụ (Hình 8)

Trong đó, có 7 users lần lượt là u_0, u_1, u_2, u_3, u_4, u_5, u_6 và 5 items lầm lượt là i_1, i_2, i_3, i_4, i_5. Có thể quan sát thấy u_0, u_1 đều thích i_0 và không thích i_3, i_4lắm, còn các users khác thì ngược lại.

Đặt mức độ giống nhau giữa hai user u_i, u_j là sim(u_i, u_j). Khi đó, một similarity function tốt, cần đảm bảo:

$$sim(u_0, u_1) \geq sim(u_0, u_i) \ \forall \ i > 1$$

Một số similarity function thường được sử dụng là Cosine Similarity và Pearson Corelation.

$$\mathbf{cosine\_similarity}(\boldsymbol{u_i}, \boldsymbol{u_j}) = \mathbf{cos}(\boldsymbol{u_i}, \boldsymbol{u_j}) = \frac{\boldsymbol{u_i u_j}}{\|\boldsymbol{u_i}\|\|\boldsymbol{u_j}\|} \tag{2.6}$$

$$\mathbf{pearson}(\boldsymbol{u_i}, \boldsymbol{u_j}) = \frac{\boldsymbol{E}[\boldsymbol{u_i u_j}] - \boldsymbol{E}[\boldsymbol{u_i}]\boldsymbol{E}[\boldsymbol{u_j}]}{\sqrt{\boldsymbol{E}[\boldsymbol{u_i^2}] - \boldsymbol{E}[\boldsymbol{u_i}]^2}\sqrt{\boldsymbol{E}[\boldsymbol{u_j^2}] - \boldsymbol{E}[\boldsymbol{u_j}]^2}} \tag{2.7}$$

Áp dụng similarity function để tính độ tương đồng giữa các users, ta sẽ thu được ma trận similarity matrix.

	U_0	U_1	U_2	U_3	U_4	U_5	U_6
U_0	1	0.83	-0.58	-0.79	-0.82	0.2	-0.38
U_1	0.83	1	-0.87	-0.4	-0.55	-0.23	-0.71
U_2	-0.58	-0.87	1	0.27	0.32	0.47	0.96
U_3	-0.79	-0.4	0.27	1	0.87	-0.29	0.18
U_4	-0.82	-0.55	0.32	0.87	1	0	0.16
U_5	0.2	-0.23	0.47	-0.29	0	1	0.56
U_6	-0.38	-0.71	0.96	0.18	0.16	0.56	1

Hình 9 Ma trận Similariry Matrix của các users

(Nguồn https://machinelearningcoban.com/2017/05/24/collaborativefiltering/)

Hình 9 là ví dụ sử dụng hàm khoảng cách cosine similarity.

3. Rating Prediction

Chúng ta sẽ dự đoán ratings của một user với mỗi item dựa trên k users gần nhất (neighbor users), tương tự như phương pháp K-nearest neighbors (KNN).

Công thức phổ biến thường được sử dụng để dự đoán rating của user u cho item i là:

$$\widehat{\boldsymbol{y}}_{\boldsymbol{i,u}} = \frac{\sum_{\boldsymbol{u_j} \in \boldsymbol{N(u,i)}} \overline{\boldsymbol{y}}_{\boldsymbol{i,u_j}} \boldsymbol{sim(u, u_j)}}{\sum_{\boldsymbol{u_j} \in \boldsymbol{N(u,i)}} | \boldsymbol{sim(u, u_j)} |} \quad (2.8)$$

Trong đó, N(u, i) là tập k users gần nhất (có độ tương đồng cao nhất) với u và đã từng đánh giá i.

Ví dụ dự đoán normalized rating của user u_1 cho item i_1với k = 2 là số users gần nhất.

- Bước 1: Xác định các users đã rated cho i_1, đó là u_0, u_3, u_5
- Bước 2: Lấy simalaries của u_1 với u_0, u_3, u_5. Kết quả lần lượt là: $\{u_0, u_3, u_5\} : \{0.83, -0.4, -0.23\}$.

Với k = 2. Chọn 2 giá trị lớn nhất là 0.83 và -0.23, tương ứng với các users u_0, u_5. Hai users này có normalized ratings với i_1 là: $\{u_0, u_5\} : \{0.75, 0.5\}$

- Bước 3: Tính normalized ratings theo công thức (**2.8**):

$$y_{i_1,u_1} = \frac{0.83 * 0.75 + (-0.23) * 0.5}{0.83 + |-0.23|} \approx 0.48$$

Tiến hành tính toán để dự đoán cho tất cả các trường hợp missing ratings (chưa có dự đoán), ta sẽ thu được ma trận normalized utility matrix như ví dụ:

	U_0	U_1	U_2	U_3	U_4	U_5	U_6
i_0	1.75	2.25	-0.5	-1.33	-1.5	0.18	-0.63
i_1	0.75	0.48	-0.17	-1.33	-1.33	0.5	0.05
i_2	0.91	1.25	-1.5	-1.84	-1.78	-0.5	-2.33
i_3	-1.25	-0.75	0.5	2.67	1.5	0.59	0.67
i_4	-1.25	-2.75	1.5	1.57	1.56	1.59	1.67

Hình 10 Ma trận Normalized Utility Matrix hoàn thiện

(Nguồn https://machinelearningcoban.com/2017/05/24/collaborativefiltering/)

Cuối cùng, cộng lại các giá trị ratings với ratings trung bình (ở bước chuẩn hóa) theo từng cột. Chúng ta sẽ thu được ma trận hoàn thiện.

2.2.2.2 Item – Item Collaborative Filtering

Trong phương pháp này, thay vì tính similarity giữa các users như trong uuCF, chúng ta sẽ tính similarity giữa các items, rồi gợi ý cho users những items gần giống với item mà user đó đã thích. Lợi ích của phương pháp này là [1]:

- Vì số lượng items thường rất nhỏ so với số lượng users nên kích thước ma trận similarity matrix sẽ nhỏ hơn rất nhiều, giúp tối ưu hơn cả về mặt lưu trữ và tốc độ tính toán.
- Thường thì mỗi item được đánh giá bởi rất nhiều users, và con số này thường lớn hơn nhiều so với số items mà mỗi user đánh giá. Vì vậy, số giá trị đã biết trong một vector biểu diễn item sẽ lớn hơn nhiều so với một vector biểu diễn user. Trong trường hợp có thêm một số dữ liệu về ratings, giá trị trung bình ratings của iiCF sẽ ít thay đổi hơn so với uuCF, vì vậy sẽ ít phải cập nhật ma trận similarity matrix hơn.

Về mặt tính toán, iiCF có thể thực hiện theo uuCF bằng cách chuyển vị ma trận utility matrix, coi như items đánh giá users [1][5]. Sau khi tính được kết quả, chúng ta lại thực hiện chuyển vị một lần nữa để thu được kết quả cuối cùng.

Về mặt thuật toán, ta sẽ tiến hành iiCF như sau:

Đầu tiên là chuẩn hoán ma trận utility matrix, thay vì tính trung bình cộng ratings của các users, chúng ta sẽ tính trung bình cộng ratings của các items.

	U_0	U_1	U_2	U_3	U_4	U_5	U_6		
i_0	5	5	2	0	1	?	?	→	2.6
i_1	4	?	?	0	?	2	?	→	2
i_2	?	4	1	?	?	1	1	→	1.75
i_3	2	2	3	4	4	?	4	→	3.17
i_4	2	0	4	?	?	?	5	→	2.75

Hình 11 Ma trận Utility gốc và trung bình cộng ratings của các items

(Nguồn https://machinelearningcoban.com/2017/05/24/collaborativefiltering/)

Sau đó thực hiện chuẩn hóa bằng cách trừ các giá trị ratings đã biết của mỗi item cho trung bình cộng ratings tương ứng của item đó, đồng thời thay các ratings chưa biết bằng 0. Từ đó, thu được ma trận normalized utility matrix

	U_0	U_1	U_2	U_3	U_4	U_5	U_6
i_0	2.4	2.4	-6	-2.6	-1.6	0	0
i_1	2	0	0	-2	0	0	0
i_2	0	2.25	-0.75	0	0	-0.75	-0.75
i_3	-1.17	-1.17	-0.17	0.83	0.83	0	0.83
i_4	-0.75	-2.75	1.25	0	0	0	2.25

Hình 12 Ma trận Normalized Utiliy Matrix chuẩn hóa theo items

(Nguồn https://machinelearningcoban.com/2017/05/24/collaborativefiltering/)

Tiếp theo, tính similarity matrix bằng similarity function cho các items (tương tự như uuCF):

	i_0	i_1	i_2	i_3	i_4
i_0	1	0.77	0.49	-0.89	-0.52
i_1	0.77	1	0	-0.64	-0.14
i_2	0.49	0	1	-0.55	-0.88
i_3	-0.89	-0.64	-0.55	1	0.68
i_4	-0.52	-0.14	-0.88	0.68	1

Hình 13 Ma trận Similarity Matrix giữa các item

(Nguồn https://machinelearningcoban.com/2017/05/24/collaborativefiltering/)

Cuối cùng, sử dụng các ma trận similarity matrix và normalized utility matrix để dự đoán ra ratings của các users với mỗi items (tương tự như uuCF).

Ví dụ, dự đoán rating của user U cho item I, ta thực hiện:

- Bước 1: Tìm tập N(i, u) các items mà U đã đánh giá.
- Bước 2: Lấy similarites của I với các items trong tập N(u, i). Chọn ra k items gần nhất (có similarity cao nhất) với I.
- Bước 3: Tính rating dự đoán theo công thức

$$\hat{y}_{i,u} = \frac{\sum_{i_j \in N(i,u)} \bar{y}_{u,i_j} sim(i, i_j)}{\sum_{i_j \in N(i,u)} | sim(i, i_j) |} \quad (2.9)$$

Sau khi dự đoán hết các ratings chưa biết, chúng ta sẽ thu được ma trận normalized utility matrix đầy đủ:

	U_0	U_1	U_2	U_3	U_4	U_5	U_6
i_0	2.4	2.4	-6	-2.6	-1.6	-0.29	-1.52
i_1	2	2.4	-0.6	-2	-1.25	0	-2.25
i_2	2.4	2.25	-0.75	-2.6	-1.2	-0.75	-0.75
i_3	-1.17	-1.17	-0.17	0.83	0.83	0.34	0.83
i_4	-0.75	-2.75	1.25	1.03	1.16	0.65	2.25

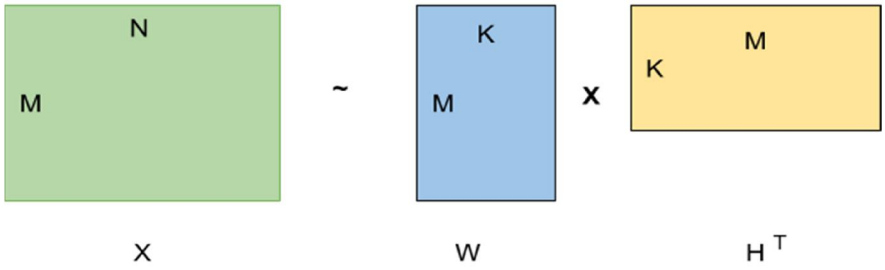
Hình 14 Ma trận Normalized Utility Matrix đầy đủ ratings

(Nguồn https://machinelearningcoban.com/2017/05/24/collaborativefiltering/)

2.2.3 Matrix Factorization (MF)

Matrix Factorization: Là một hướng tiếp cận khác của Collaborative Filtering, còn gọi là Matrix Decomposition, nghĩa là gợi ý bằng kỹ thuật Phân rã ma trận.

Kỹ thuật phân rã ma trận là phương pháp chia một ma trận lớn X thành hai ma trận có kích thước nhỏ hơn là W và H, sao cho ta có thể xây dựng lại X từ hai ma trận nhỏ hơn này càng chính xác càng tốt, nghĩa là X ~ WH^T [4].



Hình 15 Mô tả kỹ thuật phân rã ma trận

Ý tưởng chính của Matrix Factorization cho RS là đặt items và users vào trong cùng một không gian thuộc tính ẩn. Trong đó, $W \in R^{|U|*K}$ là một ma trận mà mỗi dòng u là một vector bao gồm K nhân tố tiềm ẩn (latent factors) mô tả user u và $H \in R^{|I|*K}$ là một ma trận mà mỗi dòng i là một vector bao gồm K nhân tố tiềm ẩn mô tả cho item i.

Áp dụng phương pháp này vào bài toán gợi ý, chúng ta có **x** là một vector của *item profile*. Mục tiêu của chúng ta là tìm một vector **w** tương ứng với mỗi user sao cho ratings đã biết của user đó cho item (**y**) xấp xỉ với:

$$\boldsymbol{y} \approx \boldsymbol{xw} \tag{2.10}$$

Mở rộng với Y là utility matrix, giả sử đã được điền hết giá trị, ta có:

$$Y \approx \begin{bmatrix} \mathbf{x}_1\mathbf{w}_1 & \mathbf{x}_1\mathbf{w}_2 & \dots & \mathbf{x}_1\mathbf{w}_N \\ \mathbf{x}_2\mathbf{w}_1 & \mathbf{x}_2\mathbf{w}_2 & \dots & \mathbf{x}_2\mathbf{w}_N \\ \dots & \dots & \ddots & \dots \\ \mathbf{x}_M\mathbf{w}_1 & \mathbf{x}_M\mathbf{w}_2 & \dots & \mathbf{x}_M\mathbf{w}_N \end{bmatrix} = \begin{bmatrix} \mathbf{x}_1 \\ \mathbf{x}_2 \\ \dots \\ \mathbf{x}_M \end{bmatrix} \begin{bmatrix} \mathbf{w}_1 & \mathbf{w}_2 & \dots & \mathbf{w}_N \end{bmatrix} = \mathbf{XW}$$

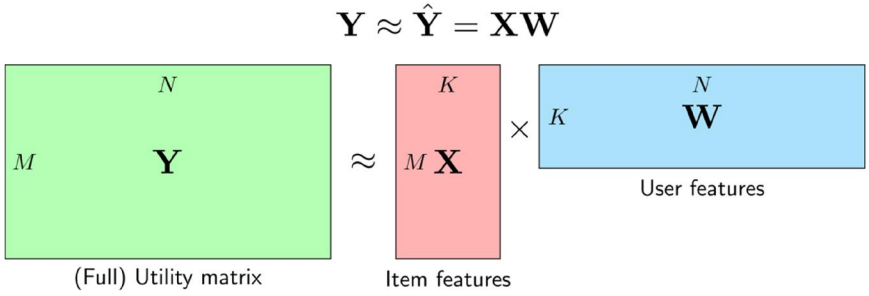
với M, N lần lượt là số users và số items.

Lưu ý, do X là được xây dựng dựa trên thông tin mô tả item và quá trình xây dựng này độc lập với quá trình đi tìm hệ số phù hợp cho mỗi user [1]. Nên việc xây dựng *item profile*

đóng vai trò quan trọng và có thể ảnh hưởng trực tiếp đến hiệu năng của mô hình. Thêm nữa, việc xây dựng mô hình riêng lẻ cho mỗi user dẫn đến kết quả chưa thực sự tốt vì không khai thác được đặc điểm giống nhau giữa các user.

Giả sử rằng ta không cần xây dựng trước các item profile mà ma trận này có thể huấn luyện đồng thời với ma trận trọng số, hay nói các khác bài toán này là bài toán tối ưu các ma trận X và W, trong đó X là ma trận của toàn bộ các item profiles, mỗi hàng tương ứng với 1 item. Còn W là ma trận của toàn bộ user models (các mô hình của users), mỗi cột tương ứng với một user. Chúng ta sẽ cố gắng xấp xỉ utility matrix $Y \in R^{M*N}$ bằng tích của hai ma trận con là $X \in R^{M*K}$ và $W \in R^{K*N}$.

Trong đó, K được chọn thường nhỏ hơn rất nhiều so với M và N, và cả hai ma trận X và W đều phải có bậc (rank) không được vượt quá K.



Hình 16 Utility matrix Y được phân tích thành tích của hai ma trận low-rank X và W

(Nguồn https://machinelearningcoban.com/2017/05/31/matrixfactorization/)

Cụ thể, quy trình xây dựng và tối ưu hàm mất mát như sau:

Đầu tiên, chúng ta sẽ xét hàm mất mát không có cả bias và biến tối ưu cho X và W:

$$\mathcal{L}_{(X,W)} = \frac{1}{2s} \sum_{n=1}^{N} \sum_{m:\, r_{mn}} (y_{mn} - x_m w_n)^2 + \frac{\lambda}{2} (\|X\|_F^2 + \|W\|_F^2) \qquad (2.11)$$

Trong đó, $r_{mn} = 1$ nếu item thứ *m* đã được đánh giá bởi user thứ *n*, $\|\circ\|_F^2$ là căn bậc hai của tổng bình phương tất cả các phần tử của ma trận, **s** là toàn bộ số ratings đã có. Thành phần thứ nhất chính là trung bình sai số của mô hình. Thành phần thứ hai trong hàm mất mát phía, có tách dụng giúp tránh overfitting.

Lưu ý, tương tự như NBCF, các giá trị ratings được sử dụng là các giá trị đã chuẩn hóa, bằng cách trừ đi trung bình cộng các giá trị ratings đã biết trong cùng một hàng (với iiCF)

và trong cùng một cột (với uuCF) – mục 1 phần 2.2.2.1. Trong một số trường hợp, ta có thể không cần chuẩn hóa ma trận utility matrix nhưng những trường hợp đó sẽ phải dùng các kĩ thuật khác để giải quyết tính cá nhân trong các ratings.

Tiếp theo, chúng ta sẽ tối ưu X và W bằng cách cố định từng ma trận và tối ưu ma trận còn lại cho tới khi hội tụ.

Khi cố định X, việc tối ưu W chính là bài toán tối ưu của Content-based Filtering:

$$\mathcal{L}_{(W)} = \frac{1}{2s} \sum_{n=1}^{N} \sum_{m:\, r_{mn}} (y_{mn} - x_m w_n)^2 + \frac{\lambda}{2} \|W\|_F^2 \tag{2.12}$$

Ngược lại, khi cố định W, việc tối ưu X được đưa về tối ưu hàm:

$$\mathcal{L}_{(X)} = \frac{1}{2s} \sum_{n=1}^{N} \sum_{m:\, r_{mn}} (y_{mn} - x_m w_n)^2 + \frac{\lambda}{2} \|X\|_F^2 \tag{2.13}$$

Hai bài toán này sẽ được tối ưu bằng Gradient Descent.

Chúng ta có thể thấy rằng, bài toán tối ưu W có thể được tách thành N bài toán nhỏ (N là số lượng users), mỗi bài toán tương ứng với việc đi tối ưu một cột của ma trận W.

$$\mathcal{L}_{(W_n)} = \frac{1}{2s} \sum_{m\,:\, r_{mn}} (y_{mn} - x_m w_n)^2 + \frac{\lambda}{2} \|w_n\|_2^2 \tag{2.14}$$

Vì biểu thức $(y_{mn} - x_m w_n)^2$ chỉ phụ thuộc vào các items đã được user đang xét đánh giá, nên ta có thể đơn giản nó bằng cách sử dụng sub matrix, là matrix chỉ chứa các giá trị ratings đã biết. Khi đó, hàm mất mát có dạng:

$$\mathcal{L}_{(W_n)} = \frac{1}{2s} \left\|\hat{y}_{mn} - \hat{X}_n w_n\right\|^2 + \frac{\lambda}{2} \|w_n\|_2^2 \tag{2.15}$$

Và đạo hàm của nó là:

$$\frac{\partial \mathcal{L}_{(W_n)}}{\partial w_n} = -\frac{1}{s} \hat{X}_n^T (\hat{y}_n - \hat{X}_n w_n) + \lambda w_n \tag{2.16}$$

Vậy công thức cập nhật cho các cột của W là:

$$\boldsymbol{w_n} = \boldsymbol{w_n} - \boldsymbol{\eta}\left(\frac{\mathbf{1}}{\boldsymbol{s}}\widehat{\boldsymbol{X}}_{\boldsymbol{n}}^{\boldsymbol{T}}(\widehat{\boldsymbol{y}}_{\boldsymbol{n}} - \widehat{\boldsymbol{X}}_{\boldsymbol{n}}\boldsymbol{w_n}) + \boldsymbol{\lambda w_n}\right) \tag{2.17}$$

Tương tự với X, mỗi hàng tương ứng với một item sẽ được tìm bằng cách tối ưu:

$$\boldsymbol{\mathcal{L}}_{(x_m)} = \frac{\mathbf{1}}{\mathbf{2}\boldsymbol{s}}\sum_{\boldsymbol{n}:\boldsymbol{r}_{\boldsymbol{mn}}}(\widehat{\boldsymbol{y}}_{\boldsymbol{mn}} - \boldsymbol{x_m w_n})^2 + \frac{\boldsymbol{\lambda}}{\mathbf{2}}\|\boldsymbol{x_m}\|_2^2 \tag{2.18}$$

Đặt W_m là ma trận được tạo bằng các cột của W tương ứng với các users đã đánh giá items đó và sử dụng submatrix Y tương ứng là y_m. Công thức trên sẽ trở thành:

$$\boldsymbol{\mathcal{L}}_{(x_m)} = \frac{\mathbf{1}}{\mathbf{2}\boldsymbol{s}}\left\|\widehat{\boldsymbol{y}}^{\boldsymbol{m}} - \boldsymbol{x_m}\widehat{\boldsymbol{W}}_{\boldsymbol{n}}\right\|_2^2 + \frac{\boldsymbol{\lambda}}{\mathbf{2}}\|\boldsymbol{x_m}\|_2^2 \tag{2.19}$$

Vậy công thức cập nhật cho mỗi hàng của X là:

$$\boldsymbol{x_n} = \boldsymbol{x_n} - \boldsymbol{\eta}\left(\frac{\mathbf{1}}{\boldsymbol{s}}\left(\widehat{\boldsymbol{y}}^{\boldsymbol{m}} - \boldsymbol{x_m}\widehat{\boldsymbol{W}}_{\boldsymbol{m}}\right)\widehat{\boldsymbol{W}}_{\boldsymbol{m}}^{\boldsymbol{T}} + \boldsymbol{\lambda w x_m}\right) \tag{2.19}$$

Sau khi cố định X, tính W và ngược lại, cố định W và tính X cho đến khi các ma trận này hội tụ, ta sẽ thu được ma trận X và W cần tìm. Từ đó, dự đoán các giá trị ratings chưa biết.

Ngoài phương pháp trên, để tăng độ chính xác của thuật toán này, ta sẽ thêm xét hàm mất mát với bias và hệ số tối ưu cho X và W.

Như trong NBCF, chúng ta có bước chuẩn hóa ma trận để tránh sự thiên lệch do sự khó tính hay dễ tính khác nhau giữa các users. Với MF, ta có thể chuẩn không chuẩn hóa mà sử dụng trực tiếp các giá trị ratings ban đầu, bằng cách tối ưu các biases cùng lúc với X và W.

Trong trường hợp này, ratings của user m cho item n được xác định bởi công thức:

$$\boldsymbol{y_{mn}} \approx \boldsymbol{x_m y_n} + \boldsymbol{b_m} + \boldsymbol{d_n} + \boldsymbol{\mu} \tag{2.20}$$

với b_m , d_n, μ lần lượt là bias của item m, user n và ratings trung bình của toàn bộ các ratings.

Và hàm mất mát trong trường hợp này có dạng:

$$\begin{aligned}\boldsymbol{\mathcal{L}}(\boldsymbol{X},\boldsymbol{W},\boldsymbol{b},\boldsymbol{d}) = \frac{\mathbf{1}}{\mathbf{2}\boldsymbol{s}}\sum_{\boldsymbol{n}=\mathbf{1}}^{\boldsymbol{N}}\sum_{\boldsymbol{m}:\boldsymbol{r}_{\boldsymbol{mn}}}(\boldsymbol{x_m w_n} + \boldsymbol{b_m} + \boldsymbol{d_n} + \boldsymbol{\mu} - \boldsymbol{y_{mn}})^2 \\ + \frac{\boldsymbol{\lambda}}{\mathbf{2}}\left(\|\boldsymbol{X}\|_{\boldsymbol{F}}^2 + \|\boldsymbol{W}\|_{\boldsymbol{F}}^2 + \|\boldsymbol{b}\|_{\boldsymbol{F}}^2 + \|\boldsymbol{d}\|_{\boldsymbol{F}}^2\right)\end{aligned} \tag{2.21}$$

Tiến hành cố định X, b và tối ưu W,d và ngược lại, cố định W, d và tối ưu X, b, theo công thức:

$$\boldsymbol{w_n = w_n - \eta \frac{1}{s} X_n (X_n^T w_n + b_n + d_n - y_{mn}) + \lambda w_n} \quad (2.22)$$

$$\boldsymbol{b_n = b_n - \eta \frac{1}{s} 1^T (X_n^T w_n + b_n + d_n - y_{mn})} \quad (2.23)$$

$$\boldsymbol{x_m = x_m - \eta \frac{1}{s} W_m (W_m^T x_m + b_n + d_n - y_{mn}) + \lambda x_m} \quad (2.24)$$

$$\boldsymbol{d_m = d_m - \eta \frac{1}{s} 1^T (W_m^T x_m + b_n + d_n - y_{mn})} \quad (2.25)$$

Cuối cùng, ta sẽ thu được các ma trận X, b, W, d, từ đó dự đoán các ratings chưa biết.

Chương 3 Các phương pháp thử nghiệm

3.1 Tập dữ liệu

3.1.1 Mô tả tập dữ liệu

GroupLens Reasearch đã thu thập và tạo ra các bộ dữ liệu có sẵn từ trang web MovieLens (http://movielens.org). Các tập dữ liệu này có tên là MovieLens (MLs). Có nhiều tập MLs khác nhau, tùy thuộc vào kích thước của tập hợp. Thống kê về các bộ dữ liệu MLs được sử dụng được trình bày trong Bảng 1 dưới đây:

Bảng 1 Bảng thống kê mô tả các tập dữ liệu MovieLens

Chú thích: (*) Tổng số ratings trong cả tập train và tập test.

	Dataset	Users	Movies	Interactions	Sparsity
1	100K	943	1682	100000(*)	93.7%
2	1M	6040	3883	1000209	95.81%

Thành phần của các tập dữ liệu bao gồm các thông tin về: users, movies, ratings, genres và tags, etc. Trong đồ án này, để đánh giá các mô hình NBCF và MF, em chỉ sử dụng tập ratings. Với mô hình Content-based, em sử dụng các tập dữ liệu về users, movies và genres.

Tập **ratings** bao gồm các trường: userID, itemID, rating, timestamp.

	userId	movieId	rating	timestamp
0	1	2	3.5	1112486027
1	1	29	3.5	1112484676
2	1	32	3.5	1112484819
3	1	47	3.5	1112484727
4	1	50	3.5	1112484580

Hình 17 Minh họa cấu trúc tập dữ liệu ratings của Movielens

Với tập **users** bao gồm các trường: ID, age, sex, job, zipcode. Trong đồ án này chưa khai thác được mối liên hệ giữa những thông tin này và sở thích của mỗi user nên ta chỉ sử dụng ID để phân biệt các users khác nhau.

Với tập**movies**, ta chỉ quan tâm ID của mỗi movie và thông tin thể loại của mỗi movie.

Tập **genres** thì bao gồm 3 trường: movieID, Tilte và Genres. Cụ thể thì các bộ phim trong bộ MLs được chia thành 19 thể loại khác nhau.

3.1.2 Tiền xử lý tập dữ liệu

Để giảm kích thước của bộ dữ liệu (số lượng users, items) và tính thưa (mật độ tương tác giữa những users và các items) bước tiền xử lý đã loại bỏ đi những user có số lần mua hàng nhỏ hơn 2 lần, và chỉ giữ lại những item đã được các user còn lại tương tác. Đồng thời, đồng bộ định dạng của các tập dữ liệu để tiện cài đặt và thử nghiệm các mô hình.

Khác với tập MLs 100K đã có sẵn tập train và tập test, tập MLs 1M không được chia sẵn. Để tiến hành đánh giá các mô hình với tập 1M, tiến hành chia dữ liệu thành 2 phần test/train theo các tỉ lệ là lần lượt 1/10, 3/10 để tiến hành đánh giá. Tập chia theo tỉ lệ 1/10 được gọi là tập 1M(0.1), tương tự, tập còn lại là 1M(0.3)

3.2 Cài đặt mô hình

Để cài đặt các mô hình, đồ án sử dụng ngôn ngữ Python 3.6.3 trên nền tảng Anaconda.

Thông tin cấu hình phần cứng của thiết bị thử nghiệm.

Bảng 2 Cấu hình phần cứng

	Thông tin
Hệ điều hành	Ubuntu 16.04 LTS
CPU	Intel Core i5-8400 CPU @ 2.80GHZ x 6
RAM	8 GB

3.3 Phương pháp đánh giá và độ đo sử dụng

3.3.1 Mục tiêu đánh giá

- Phân tích ảnh hưởng của các tham số trên từng mô hình.
- So sánh hiệu quả của các mô hình với bộ tham số tốt nhất.

3.3.2 Phương pháp đánh giá

Để đáng giá khả năng gợi ý của các mô hình, trong đồ án này, em sử dụng phương pháp Hold – out bằng cách chia tập dữ liệu thành 2 tập train/test. Điều này đã được viết trong phần tiền xử lý (mục 3.1.2).

3.3.3 Đánh giá độ tin cậy của giải thuật

Root Mean Squared Error (RMSE) là độ đo phổ biến, thường được sử dụng trong các hệ gợi ý để đánh giá độ tin cậy của giải thuật.

RMSE được tính bằng công thức:

$$\mathbf{RMSE} = \sqrt{\frac{\mathbf{1}}{|\mathbf{D^{test}}|} \sum_{\mathbf{u,i,r} \in \mathbf{D^{test}}} (\mathbf{r_{ui}} - \mathbf{\hat{r}_{ui}})^2} \quad (3.1)$$

RMSE càng nhỏ thì độ tin cậy của giải thuật càng cao.

3.3.4 Đánh giá hiệu quả của hệ gợi ý

Để đánh giá hiệu quả của việc gợi ý, người ta thường sử dụng các độ đo liên quan đến "sự phù hợp" của kết quả trả về. Gợi ý được xem là phù hợp khi user có các phản hồi (như click, xem, mua,…) với các items trong danh sách items được gợi ý cho user đó.

Trong đồ án này, các độ đo được sử dụng là Precision và Recall.

Precision là tỷ lệ giữa số lượng các gợi ý phù hợp trên tổng số các gợi ý đã tạo ra [3].

$$\mathbf{Precision} = \frac{\mathbf{Số\ lượng\ gợi\ ý\ phù\ hợp}}{\mathbf{Số\ lượng\ gợi\ ý\ tạo\ ra}} \quad (3.2)$$

Recall là tỷ lệ giữa số lượng các gợi ý phù hợp và số lượng các mục tin mà người dùng đã chọn lựa (xem, nghe, mua, đọc,..) [3]. Recall được sử dụng để đo khả năng hệ thống tìm được những mục dữ liệu phù hợp so với những gì mà người dùng cần.

$$\textbf{Recall} = \frac{\textbf{Số lượng gợi ý phù hợp}}{\textbf{Số lượng sản phẩm được mua bởi người dùng}} \quad (3.3)$$

Ở trong trường hợp này, "số lượng sản phẩm được mua bởi người dùng" là số lượng phim mà người dùng đã xem.

3.4 Các tham số và kịch bản

3.4.1 Tham số

Mô hình Content-based:

- *Tham số* λ: Trọng số regularization của hàm mất mát để tránh overfitting.

- *Tham số top* ***K*** *items gợi ý*: Số lượng items được lựa chọn để gợi ý cho người dùng. Tham số này được sử dụng để đánh giá Precision và Recall.

Mô hình NBCF:

- *top* ***N*** *neighbors gần nhất*: Là số lượng users có độ tương đồng cao nhất được lựa chọn để dự đoán giá trị rating trong mô hình uuCF, và là số lượng items có độ tương đồng cao nhất được lựa chọn với mô hình iiCF.

- ***similarity function***: Hàm khoảng cách được dùng để tính toán khoảng cách giữa các users (trong mô hình uuCF) và items (trong mô hình iiCF). Hai độ đo khoảng cách được sử dụng trong đồ án này là cosine và pearson.

- *top* ***K*** *items gợi ý*: Số lượng item được lựa chọn để gợi ý cho mỗi user. Tham số này sử dụng cho độ đo Precision và Recall.

Mô hình MF:

- *Kích thước ẩn* ***n_factors*** : Thể hiện số chiều ẩn giữa các users và items.

- *Tham số* λ: Trọng số regularization của hàm mất mát để tránh overfitting.

- *Tham số* ***learning_rate***: Trọng số Gradient Descent, sử dụng để điều chỉnh tốc độ học.

- *Số vòng lặp* ***n_epochs***: Số lần lặp để huấn luyện.

- *Top* ***K*** *items gợi ý*: Số lượng item được lựa chọn để gợi ý cho mỗi user và đánh giá. Tham số này sử dụng để đánh giá độ đo Precision và Recall.

3.4.2 Kịch bản

Bảng 3 Mô tả kịch bản tham số đánh giá RMSE các mô hình

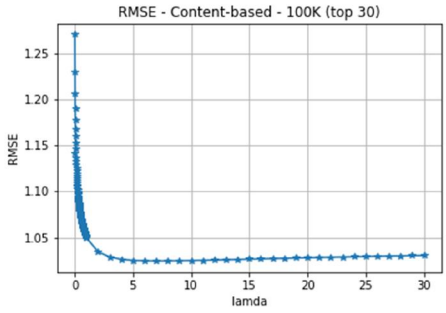
Mô hình	Tập dữ liệu	Tham số
Content-based	100K	$\lambda \in \{0.1, 0.2, \ldots, 0.9\} \cup \{1, 2, 3, \ldots, 30\}$
	1M(0.1) 1M(0.3)	$\lambda \in \{0.1, 0.2, \ldots, 0.9\} \cup \{1, 2, 3, \ldots, 100\} \cup \{200, 300, 400, 500\}$
uuCF	100K 1M(0.1) 1M(0.3)	$N \in \{1, 5, 10\ 15, 20, 25\} \cup \{30, 40, 50, \ldots, 100\}$
		similarity_function $\in$ {cosine_similarity, pearson}
MF	100K	n_factors $\in \{2, 5, 10\}$
		$\lambda \in \{0.01, 0.1, 0.5, 1\}$
		learning_rate $\in \{0.1, 0.5, 0.75, 1, 2\}$
	1M(0.1) 1M(0.3)	n_factors $\in \{2, 5, 10, 20, 30, 40, 50\}$
		$\lambda \in \{0.01, 0.1, 0.5, 1\}$
		learning_rate $\in \{0.1, 0.5, 0.75, 1, 2\}$

Để đánh giá độ đo Precision và Recall, tất cả các mô hình đều sử dụng kịch bản chung cho tất cả các tập dữ liệu là: $K \in \{10, 20, 30, \ldots, 500\}$

3.5 Ảnh hưởng của các tham số đến mô hình Content-based

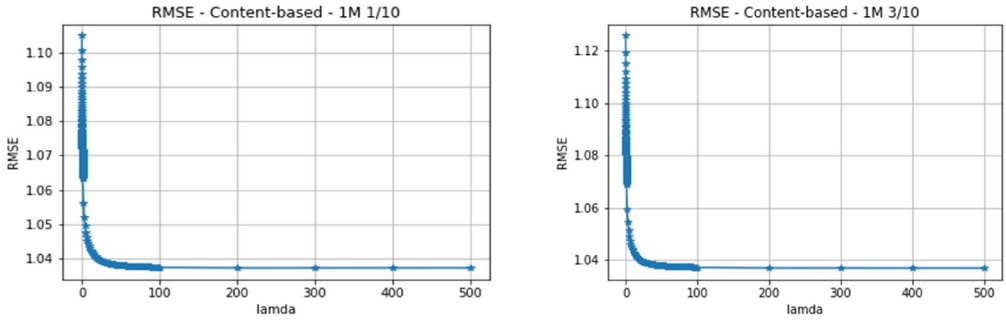
3.5.1 Ảnh hưởng của tham số λ

Để phân tích ảnh hưởng của tham số λ đến mô hình Content-based, ta thiết lập tham số λ tăng dần từ 0.1 đến 30 cho tập MLs 100K và từ 0.1 đến 500 cho tập MLs 1M (bao gồm 1M(0.1) và 1M(0.3)).



Hình 18 Ảnh hưởng của λ đến Content-based với tập MLs 100K

Quan sát Hình 18 và Hình 19, ta có thể thấy được ảnh hưởng của tham số λ đến mô hình Content-based với các tập dữ liệu, thông qua độ đo RMSE. Với tập 100K, khi λ bắt đầu tăng từ 0.1 đến khoảng 1 thì RMSE giảm mạnh và tiến rất gần về giá trị 1.00. Khi λ tiếp tục tăng từ 1 đến 30 thì RMSE tăng không đáng kể.



Hình 19 Ảnh hưởng của λ đến Content-based với tập MLs 1M(0.1) và 1M(0/3)

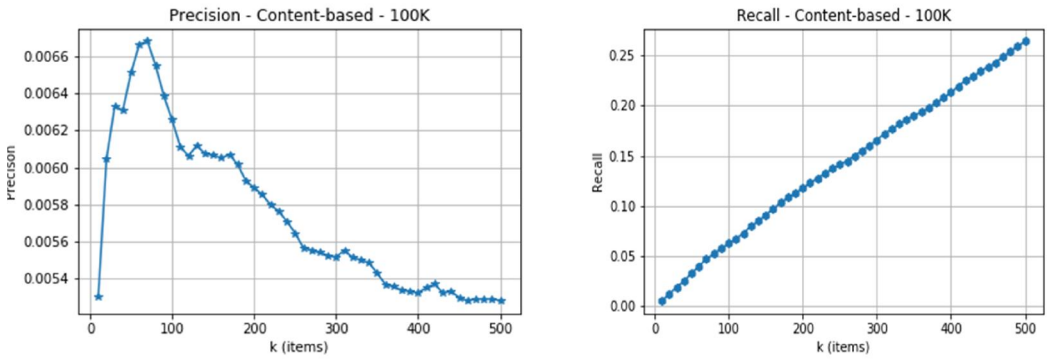
Với hai tập 1M, khi λ tăng từ 0.1 đến khoảng 10, thì RMSE giảm nhanh. Sau đó, khi λ tiếp tục tăng đến 100 thì RMSE không tăng nữa và các giá trị thu được gần như là không thay đổi.

Kết quả thu được chứng tỏ, λ được lựa chọn là giá trị tương đối nhỏ so với kích thước của tập dữ liệu.

Theo kết quả thử nghiệm đo được, kết quả tốt nhất thu được trên tập 100K khi λ = 7 và trên tập 1M là λ = 98. Những giá trị này sẽ được sử dụng để đánh giá ở mục tiếp theo (3.5.2).

3.5.2 Ảnh hưởng của top K items gợi ý

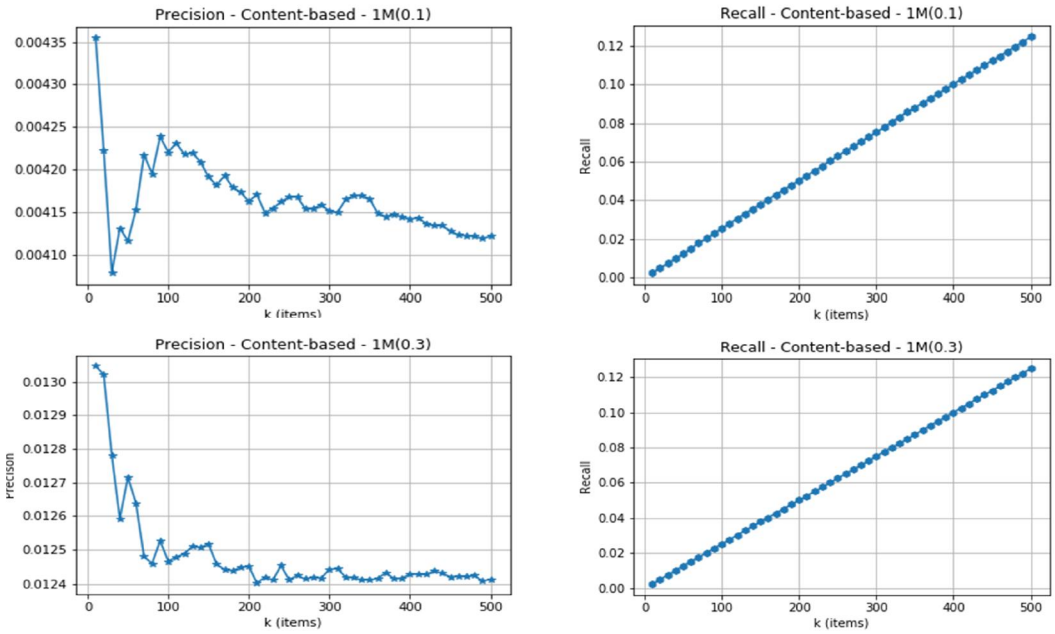
Để phân tích ảnh hưởng của top K items gợi ý đến các độ đo Precision và Recall ta thử nghiệm mô hình với tham số tốt nhất thu được ở phần trước. Với tập MLs 100K, ta sử dụng tham số λ = 7 và sử dụng λ =98 cho hai tập MLs 1M. Tiến hành đánh giá với K tăng dần từ 10 đến 500, bước nhảy là 10.



Hình 20 Ảnh hưởng của top K items gợi ý đến Content-based với tập MLs 100K

Quan sát Hình 20, ta thấy được, khi K tăng dần, Recall cũng tăng lên đồng thời. Còn độ đo Precision, như trong hình, khi K tăng đến từ 10 đến 70 thì Precision tăng dần. Sau đó, khi K tiếp tục tăng thì Precision giảm dần.

Khi K tăng lên, số lượng items gợi ý tăng thì số lượng items phù hợp cũng tăng theo. Ban đầu khi K nhỏ, Precision sẽ lớn dần do số lượng sản phẩm phù hợp tăng. Nhưng khi K lớn dần, số lượng items phù hợp thì không thay đổi nhiều do kích thước dữ liệu test có hạn thì Precision sẽ giảm dần.



Hình 21 Ảnh hưởng của top K items đến Content-based với tập MovieLens 1M

Hình 21 biểu diễn Precision và Recall thu được khi đánh giá phương pháp Content-based với tập 1M(0.1) và 1M(0.3). Quan sát thấy, tương tự như trên tập 100K, Recall trong trường hợp này cũng tăng dần khi K tăng. Precision thì khác. Với tập 1M(0.1), khi K tăng tử khoảng 10 đến 30 thì Precision giảm nhanh. Khi K tiếp tục tăng từ 30 đến 90 thì Precision cũng tăng theo. Sau đó, K tiếp tục tăng thì Precision giảm dần. Tập 1M(0.3), thì Precision hầu như có xu hướng giảm dần khi K tăng.

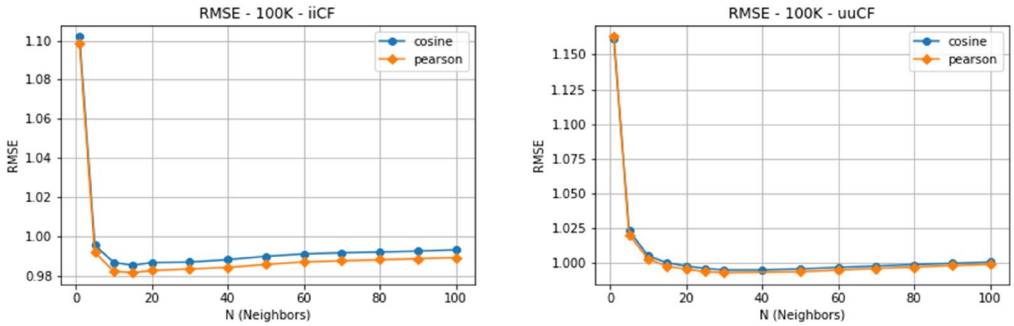
Theo kết quả thử nghiệm đo được, kết quả tốt nhất thu được trên tập 100K khi K =70, Precision $\approx$ 0.00668. Trên tập 1M(0.1), là khi K = 10, Precision $\approx$ 0.004354 và trên tập 1M(0.3) là 0.013046 khi K = 10.

3.6 Ảnh hưởng của các tham số đến mô hình NBCF

3.6.1 Ảnh hưởng của top N neighbors gần nhất và similarity functions

Để phân tích ảnh hưởng của top N neighbors gần nhất và similarity function trong phương pháp NBCF (bao gồm cả iiCF và uuCF), ta thử nghiệm mô hình trên hai tập MLs 100K và 1M(0.1) với bộ tham số:

- N tăng dần từ 10 đến 100: N ∈ {1, 5, 10 15, 20, 25} ∪ {30, 40 …, 90, 100}
- similarity_function = {cosine_similarity, pearson}



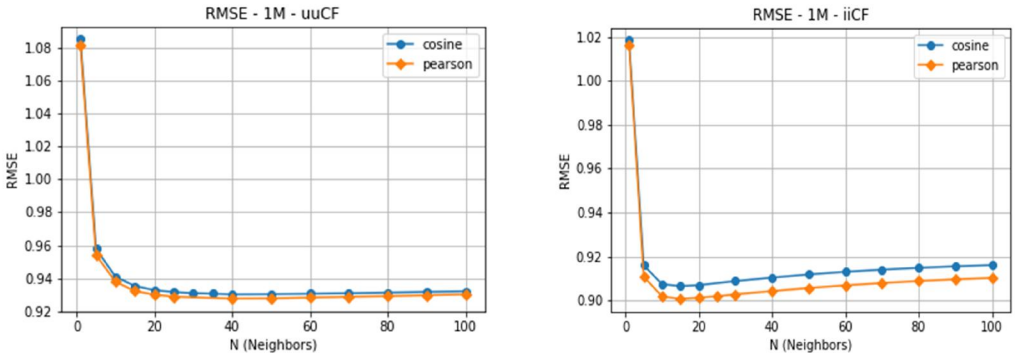
Hình 22 Ảnh hưởng của top N neighbors đến NBCF trên tậpMLs 100K

Từ Hình 22 và Hình 23, ta có thể quan sát được ảnh hưởng của tham số N neighbors gần nhất và similarity function đến mô hình NBCF, trên 2 tập dữ liệu 100K và 1M(0.1), thông qua tham số RMSE.

Đầu tiên, kết quả tạo ra khi sử dụng hai similarry function không chênh lệch nhiều với cả 2 mô hình (iiCF và uuCF). Đặc biệt, với iiCF, hai đường biểu diễn cho cosine và pearson gần như trùng khớp với nhau. Điều này chứng tỏ, hai hàm số này đem lại kết quả tương đối giống nhau trong việc xếp hạng các neighbors gần nhất.

Về ảnh hưởng của N neighbors gần nhất. Với cả 2 mô hình, ta thấy RMSE giảm nhanh khi N tăng từ 1 đế 5, sau đó tiếp tục giảm nhẹ khi N đến khoảng 20. Khi N lớn hơn 20, RMSE tăng chậm khi N tăng dần đến 100.

Tiếp tục, ta sẽ xét ảnh hưởng của các tham số trên tập MLs 1M(0.1).



Hình 23 Ảnh hưởng của top N neighbors đến NBCF trên tập MLs 1M(0.1)

Hình 23 biểu diễn ảnh hưởng của tham số N và similarity function đến mô hình NBCF trên tập MLs 1M(0.1). Tương tự như tập 100K, trên cả hai mô hình, RMSE thay đổi tương tự nhau khi sử dụng hàm cosine hoặc pearson.

Với mô hình uuCF, trên tập 1M, RMSE giảm khi N tăng từ 10 đến 20. Sau đó, khi N tiếp tục tăng thì RMSE cũng tăng dần.

Kết quả thu được chứng tỏ, sai số RMSE lớn (độ tin cậy của hệ thống sẽ thấp) khi số neighbors được lựa chọn quá nhỏ. Khi số lượng N neighbors gần nhất tăng lên thì kết quả thu được sẽ tốt hơn. Kết quả sẽ tốt cho tới khi giá trị này chỉ đăt một giá trị nhất định vì nếu xét lượng neighbors gần nhất là quá lớn thì khả năng những neighbor đó có thể khác rất nhiều so với đối tượng đang xét, khi đó độ chính xác sẽ giảm. Giá trị N hợp lý là giá trị nhỏ so với số lượng users của tập dữ liệu. Nếu N tăng lên càng lớn hơn giá trị đó thì sai số sẽ tiếp tục tăng.

Theo kết quả thử nghiệm đo được:

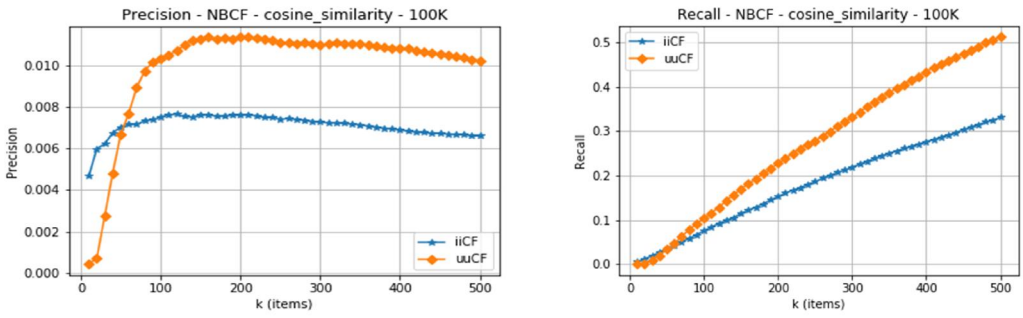
- Với mô hình iiCF, với cả hai simalrity functions , trên tập 100K, kết quả tốt nhất khi N = 30, trên tập 1M, kết quả tốt nhất khi N = 40.
- Với mô hình uuCF, với cả hai simalrity functions, trên tập 100K, kết quả tốt nhất khi N = 10, trên tập 1M, kết quả tốt nhất khi N = 20.

Những kết quả này sẽ được sử dụng để thử nghiệm với tham số K items ở phần tiếp theo.

3.6.2 Ảnh hưởng của top K items gợi ý

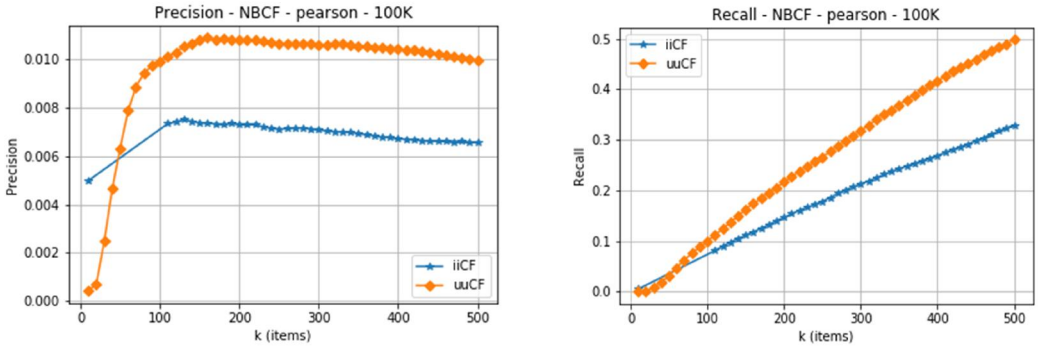
Để đánh giá độ chính xác của mô hình, ta tiến hành kiểm thử tập kết quả gợi ý với những tham số tốt nhất thu được ở phần 3.6.1. Cụ thể, ta sẽ thử nghiệm với tập K items tăng từ 10 đến 500, vơi bước nhảy là 10.

Với tập MovieLens 100K, ta sử dụng bộ tham số, với λ = 30 và similarity function là hàm cosine. Kết quả thu được thể hiện trên Hình 24.



Hình 24 Ảnh hưởng của top K items gợi ý đến NBCF với tập 100K

Quan sát Hình 24, ta thấy được, khi K tăng dần, Recall cũng tăng theo. Còn với Precision, khi K tăng từ 10 đến khoảng 100, thì Presision tăng lên rõ rệt. Sau đó, khi K tăng từ 100 đến 200, Precision tăng chậm lại. Cuối cùng, khi K tăng từ 200 đến 500 thì Precision có xu hướng giảm dần.



Hình 25 Ảnh hưởng của top K items gợi ý đến NBCF với tập 1M(0.1)

Hình 25 biểu diễn ảnh hưởng của top K items đến NBCF với hai độ đo là Precision và Recall trên tập 1M(0.1). Quan sát, ta thấy được, tương tự như tập 100K, khi K tăng dần, Recall cũng tăng theo. Còn với Precision, khi K tăng từ 10 đến khoảng 100, thì Presision tăng lên rõ rệt. Sau đó, khi K tăng từ 100 đến 170, Precision tăng chậm lại. Cuối cùng, khi K tăng từ 170 đến 500 thì Precision có xu hướng giảm dần.

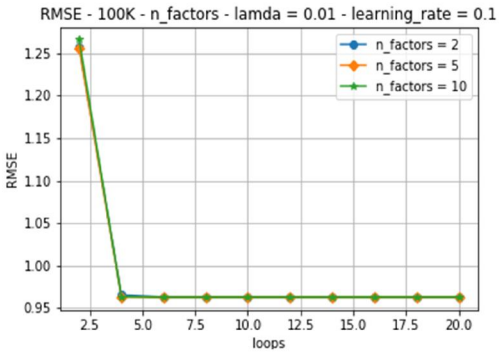
Theo kết quả đo được, trên tập 100K, mô hình iiCF cho kết quả tốt nhất với K = 120 (Precision ≈ 0.0076528) và mô hình uuCF là K = 160 (Precision ≈ 0.01136). Trên tập 1M(0.1), kết quả lần lượt là cho 2 mô hình iiCF và uuCF là K = 10 (Precision ≈ 0.01276), K = 30 (Precision ≈ 0.01053).

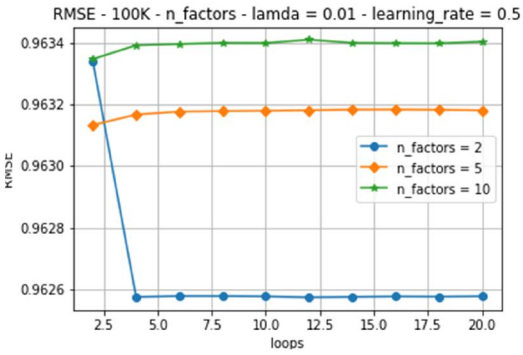
3.7 Ảnh hưởng của các tham số đến mô hình MF

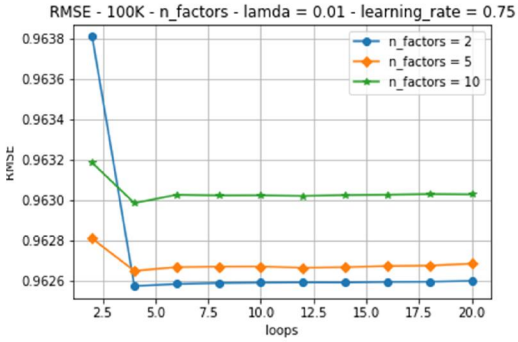
3.7.1 Ảnh hưởng của tham số ẩn n_factors và số vòng lặp

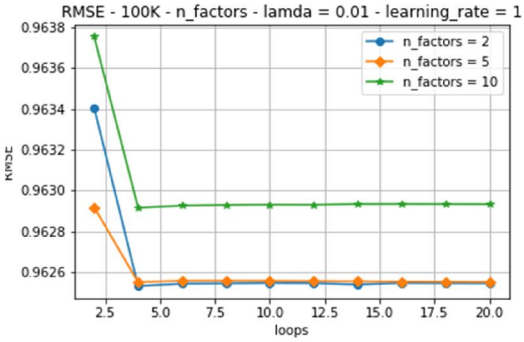
Để phân tích ảnh hưởng của tham số ẩn n_factors và số vòng lặp đến mô hình MF, ta tiến hành thử nghiệm lặp 20 lần, bước nhảy 2, trên tập 100K với bộ tham số:

- $\lambda \in \{0.01, 0.1, 0.5, 1\}$
- learning_rate $\in \{0.1, 0.5, 0.75, 1, 2\}$
- n_factors $\in \{2, 5, 10\}$

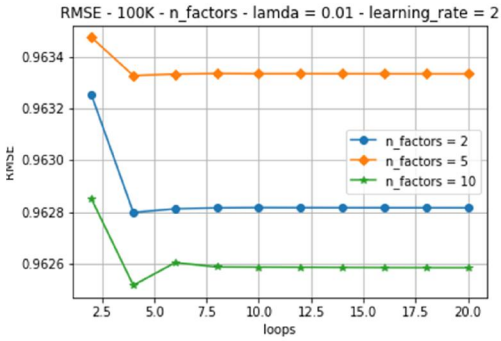

RMSE - 100K - n_factors - lamda = 0.01 - learning_rate = 0.1
n_factors = 2
n_factors = 5
n_factors = 10
RMSE
loops

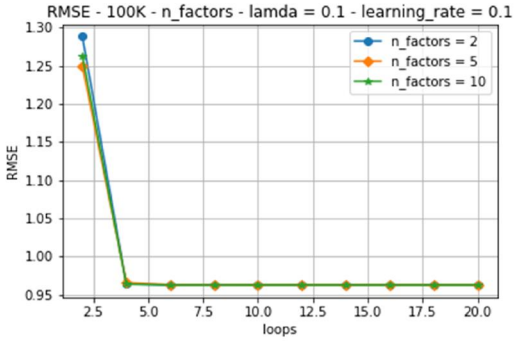

RMSE - 100K - n_factors - lamda = 0.01 - learning_rate = 0.5
n_factors = 2
n_factors = 5
n_factors = 10
RMSE
loops

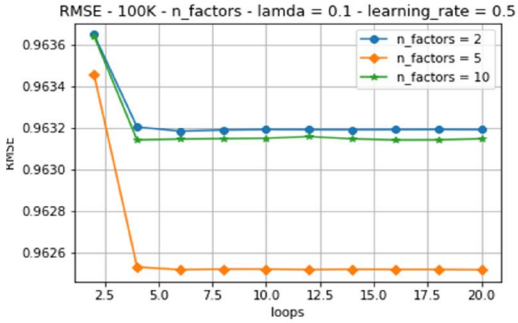

RMSE - 100K - n_factors - lamda = 0.01 - learning_rate = 0.75
n_factors = 2
n_factors = 5
n_factors = 10
RMSE
loops

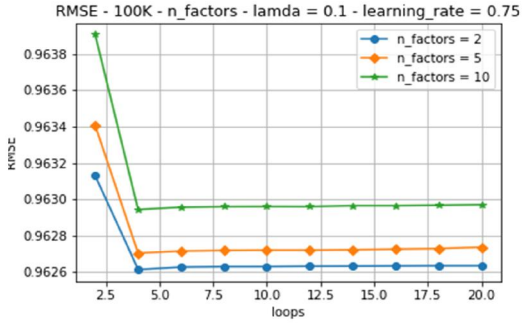

RMSE - 100K - n_factors - lamda = 0.01 - learning_rate = 1
n_factors = 2
n_factors = 5
n_factors = 10
RMSE
loops

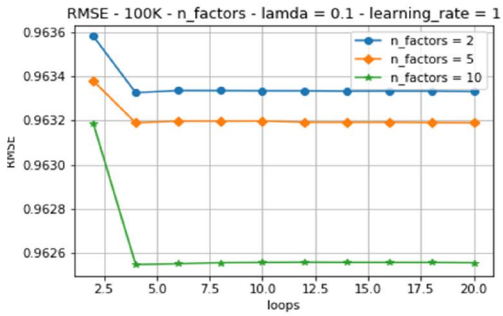
-

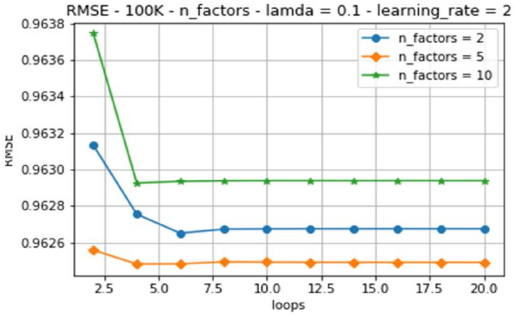

RMSE - 100K - n_factors - lamda = 0.01 - learning_rate = 2
n_factors = 2
n_factors = 5
n_factors = 10
RMSE
loops

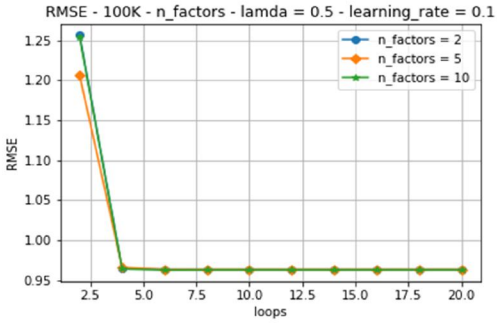

RMSE - 100K - n_factors - lamda = 0.1 - learning_rate = 0.1
n_factors = 2
n_factors = 5
n_factors = 10
RMSE
loops

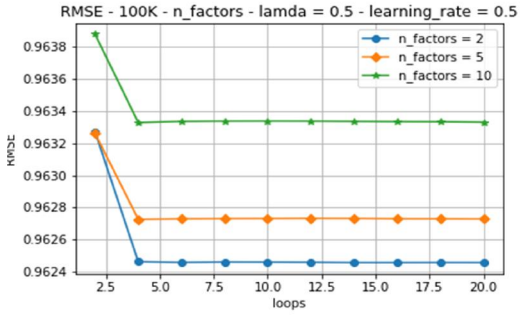

RMSE - 100K - n_factors - lamda = 0.1 - learning_rate = 0.5
n_factors = 2
n_factors = 5
n_factors = 10
RMSE
loops

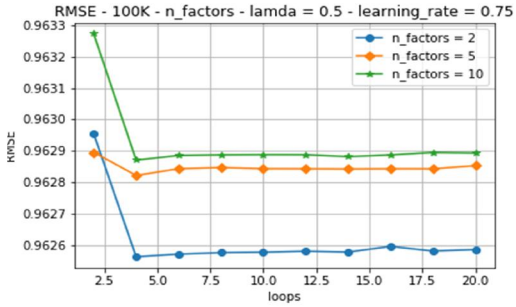

RMSE - 100K - n_factors - lamda = 0.1 - learning_rate = 0.75
n_factors = 2
n_factors = 5
n_factors = 10
RMSE
loops

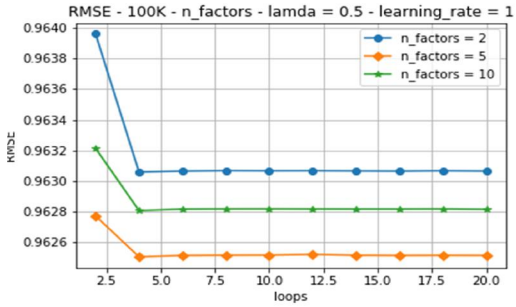

RMSE - 100K - n_factors - lamda = 0.1 - learning_rate = 1
n_factors = 2
n_factors = 5
n_factors = 10
RMSE
loops

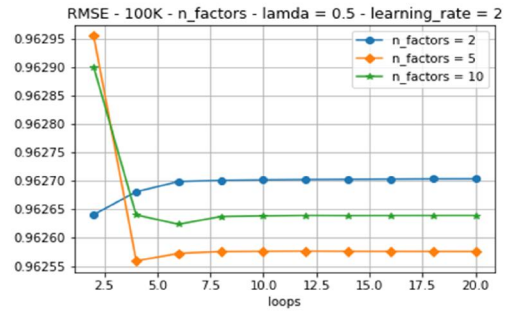

RMSE - 100K - n_factors - lamda = 0.1 - learning_rate = 2
n_factors = 2
n_factors = 5
n_factors = 10
RMSE
loops

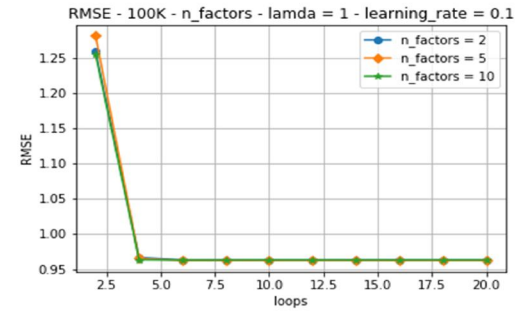

RMSE - 100K - n_factors - lamda = 0.5 - learning_rate = 0.1
n_factors = 2
n_factors = 5
n_factors = 10
RMSE
loops

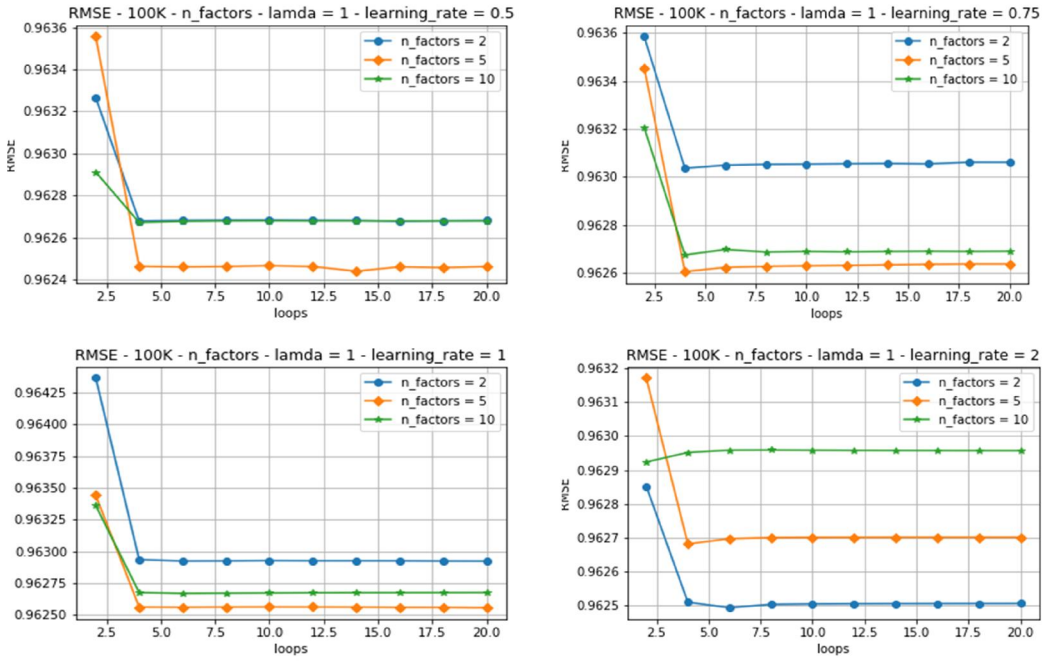

RMSE - 100K - n_factors - lamda = 0.5 - learning_rate = 0.5
n_factors = 2
n_factors = 5
n_factors = 10
RMSE
loops


RMSE - 100K - n_factors - lamda = 0.5 - learning_rate = 0.75
n_factors = 2
n_factors = 5
n_factors = 10
RMSE
loops


RMSE - 100K - n_factors - lamda = 0.5 - learning_rate = 1
n_factors = 2
n_factors = 5
n_factors = 10
RMSE
loops


RMSE - 100K - n_factors - lamda = 0.5 - learning_rate = 2
n_factors = 2
n_factors = 5
n_factors = 10
RMSE
loops


RMSE - 100K - n_factors - lamda = 1 - learning_rate = 0.1
n_factors = 2
n_factors = 5
n_factors = 10
RMSE
loops



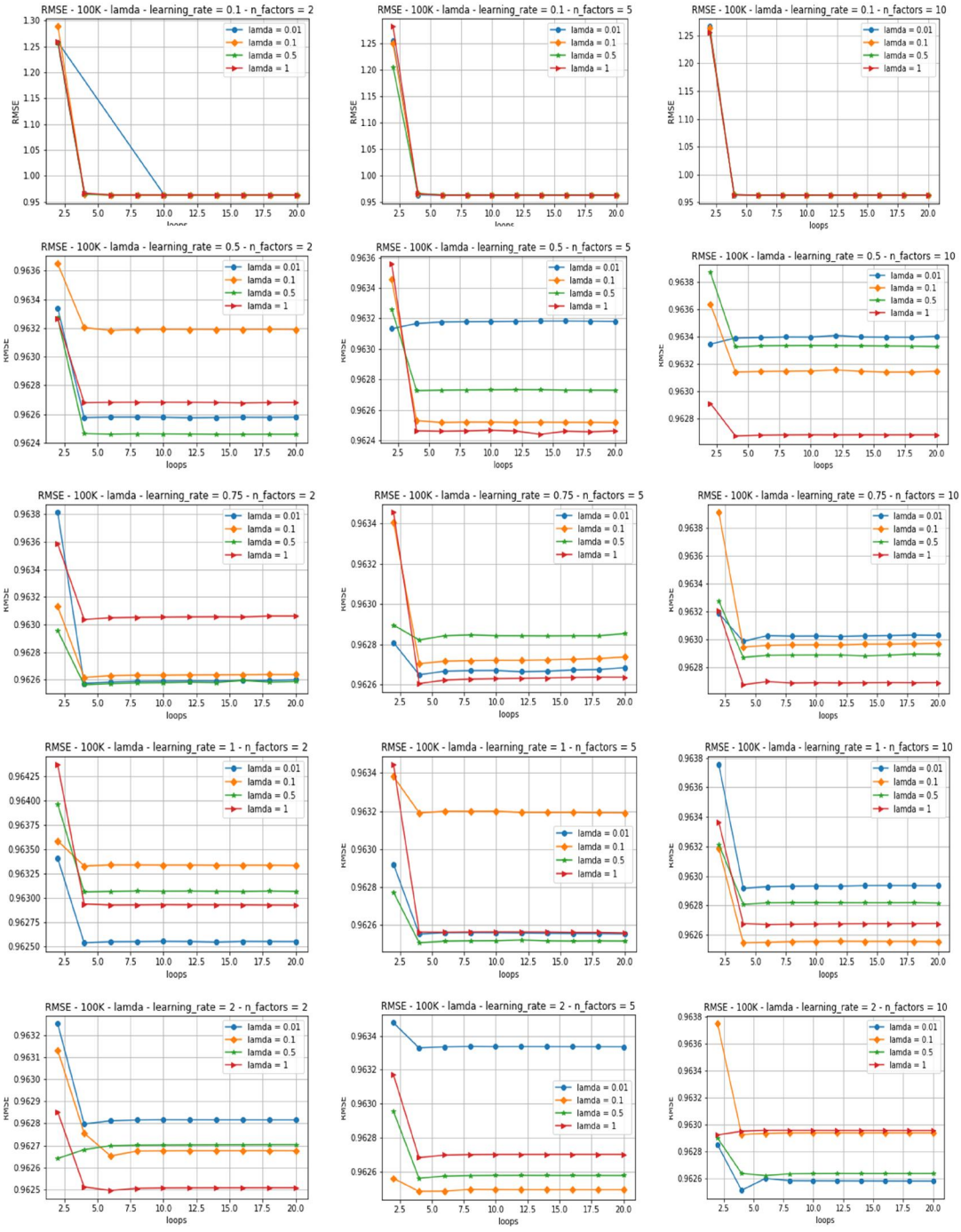
Hình 26 Ảnh hưởng của tham số ẩn n_factors đến MF

Quan sát hình 26, ta thấy, khi số vòng lặp nhỏ hơn 5, sai số với cả 3 trường hợp của n_factors đều cho kết quả RMSE lớn. Khi số vòng lặp tăng lên lớn hơn 5 thì RMSE gần như không thay đổi. Và trong tất cả các trường hợp, n_factors = 2 (đường màu xanh dương) cho kết quả tốt hơn cả khi 11/20 trường hợp giá trị n_factors này cho kết quả RMSE nhỏ nhất. Do tập 100K khá nhỏ, gồm 943 users và 1628 movies nên n_factors rất nhỏ so sẽ cho kết quả tốt nhất.

3.7.2 Ảnh hưởng của tham số λ và số vòng lặp

Để phân tích ảnh hưởng của tham số λ và số vòng lặp đến mô hình MF, ta tiến hành thử nghiệm trên tập 100K với 20 vòng lặp, bước nhảy 2, cũng với bộ tham số:

- $\lambda \in \{0.01, 0.1, 0.5, 1\}$
- learning_rate $\in \{0.1, 0.5, 0.75, 1, 2\}$
- n_factors $\in \{2, 5, 10\}$



Hình 27 Ảnh hưởng của λ đến mô hình MF

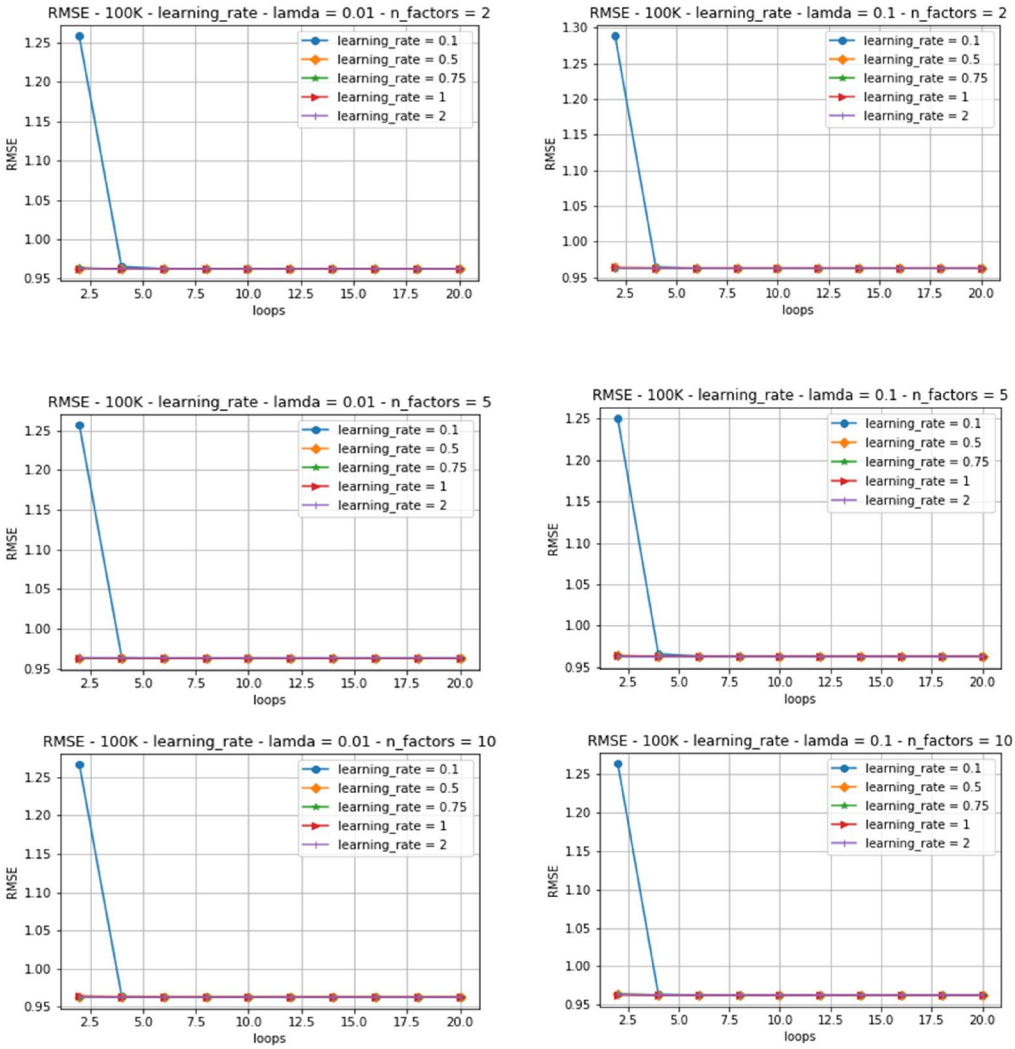
Quan sát Hình 27, ta nhận thấy, khi số vòng lặp rất nhỏ, dưới 3 thì RMSE đều rất lớn. Khi số vòng lặp tăng lên lớn hơn 5 thì RMSE gần như không đổi.

Trong 15 trường hợp, có 7 trường hợp λ = 0.01 (đường màu xanh dương) cho kết quả xấu nhất và có 6 trường hợp λ = 1 (đường màu đỏ) cho kết quả RMSE thấp nhất. Vậy có thể lựa chọn λ = 1 là tham số tốt nhất tìm được.

3.7.3 Ảnh hưởng của tham số learning_rate và số vòng lặp

Để phân tích ảnh hưởng của tham số learning_rate và số vòng lặp đến mô hình MF, ta tiến hành thử nghiệm trên tập 100K với 20 vòng lặp, bước nhảy 2, sử dụng bộ tham số:

- λ ∈ {0.01, 0.1}
- learning_rate ∈{0.1, 0.5, 0.75, 1, 2}
- n_factors ∈ {2, 5, 10}

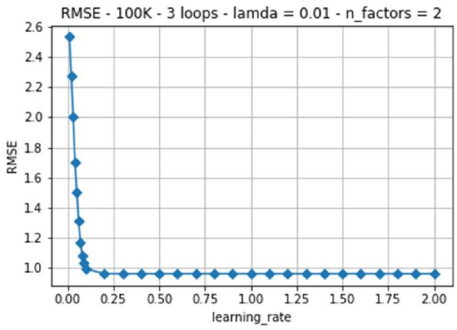


Hình 28 Ảnh hưởng của learning_rate đến MF

Nếu chia Hình 28 thành hai cột, mỗi cột 3 ảnh, ta thấy rằng, cột thứ nhất, với λ = 0.01 và cột thứ hai với λ = 0.1, tương đối giống nhau về các giá trị RMSE thu được. Điều này cho thấy, tham số learning_rate độc lập so với các tham số còn lại. Learning_rate là tham số điều chỉnh tốc độ học (hay tốc độ hội tụ) của quá trình huấn luyện. Nếu lựa chon learning_rate quá nhỏ, quá trình học sẽ rất chậm. Ngược lại, nếu lựa chọn learning_rate quá lớn thì có thể mô hình bỏ qua trường hợp đem lại kết quả tốt nhất (Hình 29).

Với learning_rate = 0.1, ta thấy kết quả RMSE lớn với khoảng 5 vòng lặp đầu tiên. Còn các giá trị còn lại đều cho kết quả như nhau. Như vậy, tham số learning_rate tốt không được quá nhỏ, phải lớn hơn 0.1 khi sử dụng tập MLs 100K.

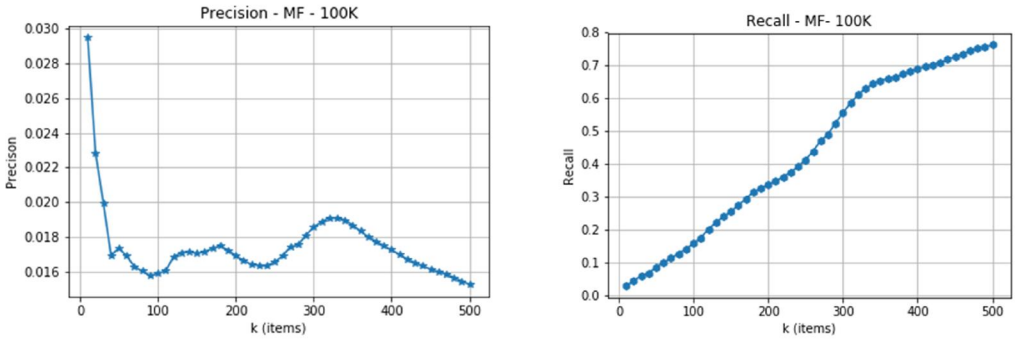
Dưới đây là biểu đồ thể hiện sự thay đổi RMSE, khi sử dụng giá trị learning_rate tăng từ 0.01 đến 2.



Hình 29 Sự ảnh hưởng của learning_rate với MF khi sử dụng 3 vòng lặp

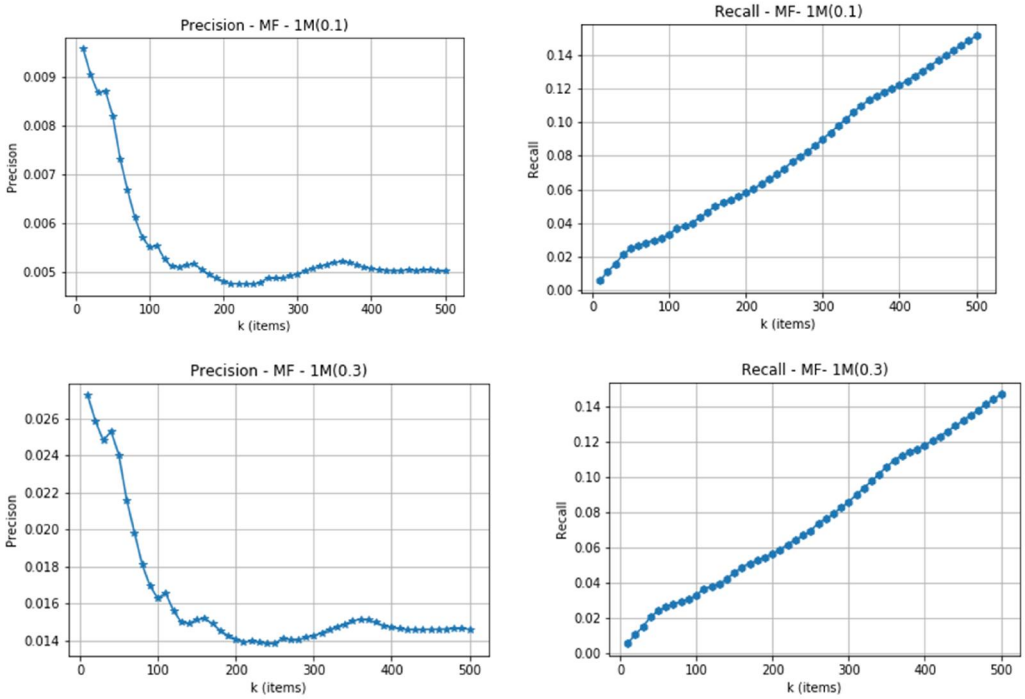
3.7.4 Ảnh hưởng của top K items gợi ý

Để đánh giá độ chính xác của mô hình, ta tiến hành kiểm thử tập kết quả gợi ý với những tham số tốt nhất thu được ở các mục trên. Cụ thể, ta sẽ tiến hành thử nghiệm với tập K items tăng từ 10 đến 500, vơi bước nhảy là 10, trên cả ba tập dữ liệu MovieLens 100K và 1M(0.1) và 1M(0.3).



Hình 30 Ảnh hưởng của top K items gợi ý đến MF trên tập 100K

Quan sát Hình 30, ta thấy được, khi K tăng dần, Recall cũng tăng dần. Còn với Precision, khi K tăng từ 10 đến khoảng 40, thì Presision giảm nhanh. Sau đó, khi K tăng từ 40 đến 250, Precision tăng giảm không ổn định. Khi K tiếp tục tăng lên trên 250, Precision tăng ổn định hơn đến khi K lên đến khoảng 340. Sau đó, thì pricision giảm.



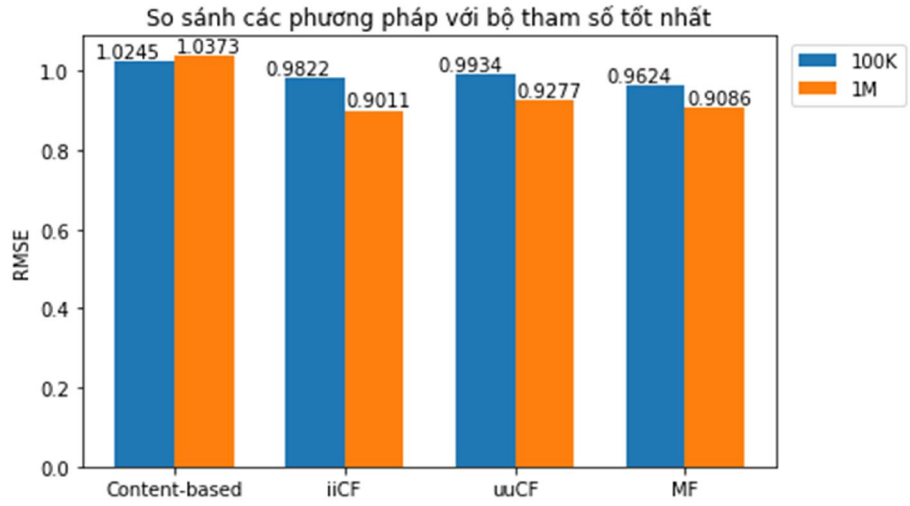
Hình 31 Ảnh hưởng của top K đến mô hình MF trên tập 1M

Hình 31 biểu diễn ảnh hưởng của top K items đến mô hình MF với hai độ đo là Precision và Recall trên tập 1M với cách chia test/train theo tỉ lệ lần lượt là 0.1 và 0.3. Quan sát, ta thấy được, tương tự như tập 100K, khi K tăng dần, Recall cũng tăng theo. Còn với Precision, khi K tăng từ 10 đến khoảng 100, sau đó giảm tiếp đến khi K tăng lên khoảng 200. Precision tăng lên khi K tăng từ khoảng 200 đến 360. Cuối cùng, khi K tiếp tục tăng thì Precision có xu hướng giảm dần.

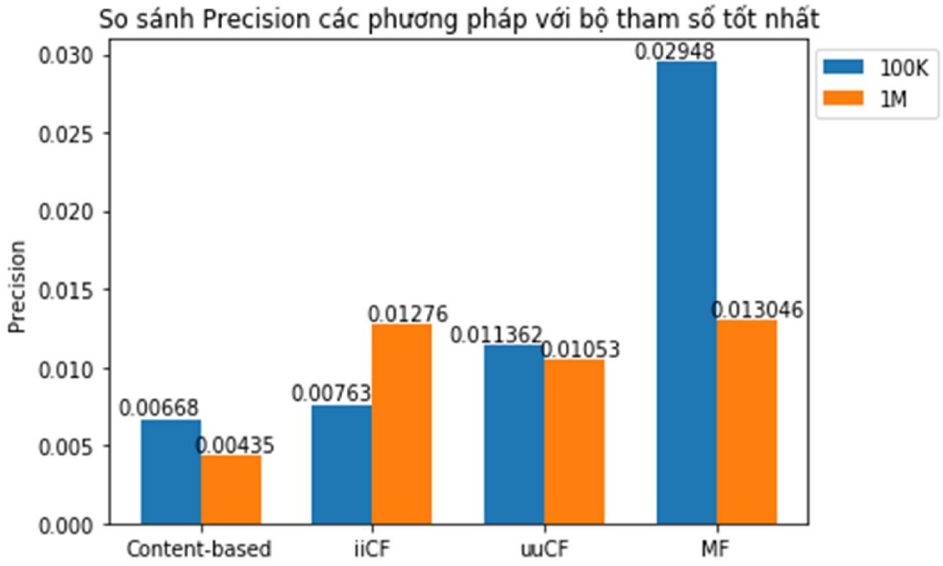
Theo kết quả đo được, trên tập 100K, mô hình MF cho kết quả tốt nhất khi K = 10, Precision ≈ 0.0294803. Trên tập 1M, kết quả lần lượt là cho hai tập 1M(0.1) và 1M(0.3) là Precision ≈ 0.0095695 (K = 10) và P ≈ 0.0273509 (K = 10).

3.8 So sánh các phương pháp với bộ trọng số tốt nhất

Với các kết quả thu được từ các mục 3.6, 3.7, 3.8. Ta lựa chọn bộ tham số tốt nhất cho mỗi mô hình trên các tập dữ liệu MovieLens 100K và 1M(0.1) để so sánh hiệu quả của từng mô hình. Kết quả biểu biễn trong Hình 32 và Hình 33dưới đây.



Hình 32 Biểu đồ so sánh các phương pháp với bộ tham số tốt nhất



Hình 33 Biểu đồ so sánh Precision của các phương pháp

Quan sát Hình 32, ta thấy, mô hình Content-based cho kết quả tồi nhất với RMSE trên cả hai tập dữ liệu đều lớn hơn 1. Hạn chế này là do mô hình thử nghiệm chỉ sử dụng thông tin về thể loại phim để xây dựng feature vector cho các items. Để thuật toán Content-based

hiệu quả hơn, cần sử dụng tập dữ liệu cung cấp nhiều dữ liệu tường minh về items và feedback của users cho các items.

Các mô hình NBCF và MF thì thu được kết quả tốt hơn với RMSE vào khoảng từ 0.901 đến 0.994. Xét tập MLs 100K, mô hình MF cho kết quả tốt nhất với RMSE ≈ 0.9624. Với tập dữ liệu MLs 1M(0.1), mô hình iiCF cho kết quả tốt nhất với RMSE ≈ 0.9011, xếp ngay sau đó là mô hình MF với kết quả RMSE ≈ 0.9086. Tuy nhiên, Precision của MF lại cho kết quả cao hơn cả so với cả 3 mô hình còn lại. Độ đo RMSE là để xác định độ tin cậy của thuật toán, nhưng nếu thuật toán chính xác cho phần ratings thấp tốt hơn phần ratings cao, làm cho Precision không thực sự tốt thì thuật toán đó cũng không được đánh giá cao. Vì mục đích sau cùng của hệ gợi ý đưa ra được danh sách items mà user quan tâm. Hơn nữa, với tập dữ liệu lớn, tốc độ của thuật toán MF nhanh hơn rất nhiều. Vì vậy, có thể kết luận rằng thuật toán cho kết quả tốt nhất sau quá trình thử nghiệm ghi nhận được là mô hình MF.

Chương 4 Kết luận và hướng phát triển

4.1 Kết luận

Đã có rất nhiều công trình nghiên cứu và thư viện ra đời nhằm giải quyết bài toán gợi ý. Sau khi tiến hành thử nghiệm, em nhận thấy kết quả em thu được không chênh lệch nhiều so với kết quả các công trình trước đó.

Trong quá trình thực hiện đồ án, em đã thực hiện:

- Tìm hiểu về hệ gợi ý và bài toán gợi ý.
- Tìm hiểu và thử nghiệm các phương pháp được sử dụng để giải quyết bài toán gợi ý, bao gồm: Content-based Filtering, Neighborhoods-based (gồm User-user Collaborative Filtering và Item-item Collaborative Filtering) và Matrix Factorization Collaborative Filtering, trên 2 tập dữ liệu MovieLens 100K và 1M. Thu thập số liệu từ quá trình thử nghiệm để phân tích ảnh hưởng của các tham số đối với từng mô hình và lựa chọn bộ tham số tốt nhất cho từng mô hình.
- So sánh kết quả thu được khi thử nghiệm các mô hình. Mỗi mô hình đều có những ưu, nhược điểm khác nhau và có khả năng dự đoán kết quả khác nhau. Trong đồ án này, kết quả thu được cho thấy mô hình MF là tốt nhất.

Đồng thời, em cũng chưa thực hiện được:

- Thử nghiệm bài toán với các bộ dữ liệu lớn hơn và có nhiều dữ liệu về content-based hơn.
- Tiếp cận bài toán với các phương pháp mới như
- Ứng dụng kết quả thu được cho một bài toán thực tế cụ thể.

Tại thời điểm thực hiện đồ án, em đang theo đuổi con đường trở thành một lập trình viên Web. Nhưng vì nhận thấy sự hữu ích và tính ứng dụng cao của hệ gợi ý, nên em đã chọn đề tài này để thực hiện đồ án. Nhờ vậy, em đã thu được rất nhiều kiến thức, bao gồm các khái niệm về hệ gợi ý, cách xây dựng mô hình học máy, kinh nghiệm thử nghiệm và đánh giá, lựa chọn tham số cho các mô hình.

4.2 Hướng phát triển

Hướng phát triển tiếp theo của đồ án sẽ nhằm mục đích hạn chế các vấn đề còn tồn đọng trong đồ án hiện tại:

- Tiếp cận bài toán gợi ý các mô hình mới hơn như: View-enhanced eALS, Neural Matrix Factorization, Co-rating Model, etc.
- Mỗi phương pháp đều có những ưu và nhược điểm riêng. Vì vậy, ta có thể kết hợp các mô hình với nhau để thu được kết quả tối ưu hơn.

Sau đó, em mong muốn có thể áp dụng những kinh nghiệm và kiến thức thu được để áp dụng vào bài toán thực tế cụ thể, như mảng thương mại điện tử hoặc tin tức, giải trí trực tuyến. Áp dụng những kiến thức thu được trong đồ án này, cùng kiến thức về Web của bản thân để có thể tạo ra những trang Web đem lại trải nghiệm chân thật hơn tới người dùng.

Tài liệu tham khảo

[1] Nguyễn Hữu Tiệp, Machine Learning cơ bản, 2018

[2] Francesco Ricci, Lior Rokack, Bracha Shapira, Paul B. Kantor, *Recommender systems handbook*, Springer, 2015.

[3] Guy Shani and Asela Gunawardana, Evaluating recommendation systems, *Recommender Systems Handbook*, Springer, 2011

[4] Korea, Y., Bell, R., & Volinsky, Matrix Factorization techniques for recommender systems, IEEE Computer Society Press, 2009

[5] Ekstrand, Michael D., John T. Riedl, and Joseph A. Konstan, *Collaborative filtering recommender systems*, Foundations and Trends® in Human-Computer Interaction 4.2 (2011).

[6] Badrul Sarwar, George Karypis, Joseph Konstan, and John Riedl, *Itembased collaborative filtering recommendation algorithms*, ACM, 2001.

[7] Alexander Felfering, Michael Jeran, Gerald Ninaus, Florian Reinfrank, Stefan Reiterer and Martin Stettinger, Basic Approaches in Recommendation System, *Recommendation Systems in Software Engineering*, 2013.