**TRƯỜNG ĐẠI HỌC SƯ PHẠM KỸ THUẬT TP. HỒ CHÍ MINH**

**KHOA ĐÀO TẠO CHẤT LƯỢNG CAO**

**NGÀNH CÔNG NGHỆ THÔNG TIN**

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_



**BÁO CÁO ĐỒ ÁN 3**

**TÌM HIỂU THUẬT TOÁN MATRIX FACTORIZATION**

***GVHD: Lê Văn Vinh***

***SVTH: Võ Nguyễn Minh Hiền***

***MSSV: 17110134***

***SVTH: Nguyễn Thị Xuân Mai***

***MSSV: 17110184***

***SVTH: Tôn Nữ Như Quỳnh***

***MSSV: 17110214***

**TP. Hồ Chí Minh, tháng 12 năm 2020**

**LỜI CẢM ƠN**

Trong thời gian nghiên cứu Đồ án 3, nhóm thực hiện gặp không ít khó khăn về cách làm và kiến thức tuy nhiên nhóm đã nhận được sự giúp đỡ quý báu của thầy Lê Văn Vinh và các bạn, nhóm đã hoàn thành bài báo cáo này.

Tuy nhiên do thời gian có hạn, cũng như kinh nghiệm còn thiếu nên trong bài báo cáo này sẽ không tránh khỏi những thiếu sót, hạn chế nhất định. Những ý kiến nhận xét góp ý quý báu của thầy và các bạn là cơ sở để nhóm học hỏi và hoàn thiện kiến thức của mình. Nhóm rất mong nhận được nhận xét góp ý của thầy và các bạn.

Nhóm thực hiện xin chân thành cảm ơn!

**MỤC LỤC**

[Chương 1 Giới thiệu 1](#_Toc61781026)

[1.1 Lý do chọn đề tài 1](#_Toc61781027)

[1.2 Mục tiêu nghiên cứu 1](#_Toc61781028)

[1.3 Nhiệm vụ nghiên cứu 1](#_Toc61781029)

[Chương 2 Cơ sở lý thuyết 3](#_Toc61781030)

[2.1 Recommender System (RS) 3](#_Toc61781031)

[2.1.1 Giới thiệu 3](#_Toc61781032)

[2.1.2 Các khái niệm chính 4](#_Toc61781033)

[2.1.3 Thông tin phản hồi từ người dùng và hai dạng bài toán chính trong RS 5](#_Toc61781034)

[2.1.4 Các kỹ thuật chính trong RS 6](#_Toc61781035)

[2.1.4.1 Kỹ thuật k láng giềng dựa trên người dùng/mục tin (User/Item kNNs) 7](#_Toc61781036)

[2.1.4.2 Kỹ thuật Phân rã ma trận (Matrix factorization – MF) 9](#_Toc61781037)

[2.1.4.3 Các kỹ thuật không cá nhân hóa 9](#_Toc61781038)

[2.2 Content-based filtering 10](#_Toc61781039)

[2.3 Collaborative filtering 12](#_Toc61781040)

[2.3.1 Hai lĩnh vực trong Collaborative filtering 12](#_Toc61781041)

[2.3.1.1 Neighborhood method (Phương pháp vùng lân cận) 12](#_Toc61781042)

[2.3.1.2 Factors Model (Mô hình yếu tố tìm ẩn) 13](#_Toc61781043)

[2.4 Thuật toán Matrix factorization (MF) 14](#_Toc61781044)

[2.4.1 Giới thiệu 14](#_Toc61781045)

[2.4.2 Mô hình Matrix factorization cơ bản 17](#_Toc61781046)

[2.4.3 Stochastic gradient descent 18](#_Toc61781047)

[2.4.4 Phương pháp Alternating least squares 19](#_Toc61781048)

[2.4.5 Matrix factorization với bias 20](#_Toc61781049)

[2.4.6 Mức độ liên quan và nghiên cứu trước đó 21](#_Toc61781050)

[Chương 3 Cài đặt thuật toán 22](#_Toc61781051)

[3.1 Huấn luyện model 22](#_Toc61781052)

[3.1.1 Load dữ liệu và tìm hiểu dữ liệu 22](#_Toc61781053)

[3.1.2 Chuẩn bị dữ liệu 24](#_Toc61781054)

[3.1.3 Xây dựng và huấn luyện model 25](#_Toc61781055)

[3.1.4 Đánh giá model 28](#_Toc61781056)

[3.1.5 Sử dụng model 30](#_Toc61781057)

[3.1.6 Cải thiện model: 31](#_Toc61781058)

[3.1.7 Lưu model 33](#_Toc61781059)

[3.2 Áp dụng model vào ứng dụng 34](#_Toc61781060)

[3.2.1 Khởi tạo môi trường và load model đã huấn luyện 34](#_Toc61781061)

[3.2.2 Tạo 1 API để lấy phim gợi ý cho user: 35](#_Toc61781062)

[3.2.3 Kết quả 36](#_Toc61781063)

[Chương 4 Kết luận 38](#_Toc61781064)

[4.1 Kết quả 38](#_Toc61781065)

[4.2 Ưu điểm 38](#_Toc61781066)

[4.3 Nhược điểm 39](#_Toc61781067)

[TÀI LIỆU THAM KHẢO 40](#_Toc61781068)

**DANH MỤC CÁC HÌNH**

[Hình 2.1 Hệ thống gợi ý sản phẩm của Amazon 3](#_Toc61781738)

[Hình 2.2 Ma trận biểu diễn dữ liệu trong RS (user-item-rating matrix) 4](#_Toc61781739)

[Hình 2.3 Phương pháp k láng giềng 9](#_Toc61781740)

[Hình 2.4 Gợi ý sản phẩm thường được mua cùng nhau 10](#_Toc61781741)

[Hình 2.5 Minh hoạ Content-based Filtering 10](#_Toc61781742)

[Hình 2.6 Phương pháp tiếp cận vùng lân cận 13](#_Toc61781743)

[Hình 2.7 Mô hình tìm năng 14](#_Toc61781744)

[Hình 2.8 Phương trình (1) 15](#_Toc61781745)

[Hình 2.9 Phương trình (2) 16](#_Toc61781746)

[Hình 2.10 Phương trình (3) 16](#_Toc61781747)

[Hình 2.11 Phương trình (4) 16](#_Toc61781748)

[Hình 2.12 Phương trình xấp xỉ rating của user 17](#_Toc61781749)

[Hình 2.13 Phương trình giảm thiểu tối đa sự sai sót do chuẩn hóa 18](#_Toc61781750)

[Hình 2.14 Quy tắc cập nhật của SGD 19](#_Toc61781751)

[Hình 2.15 Phương trình ALS 19](#_Toc61781752)

[Hình 2.16 Phương trình maxtrix factorizstion với bias 20](#_Toc61781753)

[Hình 2.17 Giá trị trung bình của toàn bộ ratings 20](#_Toc61781754)

[Hình 2.18 Phương trình hàm mất mát 20](#_Toc61781755)

[Hình 3.1 Schema dữ liệu SQL 22](#_Toc61781756)

[Hình 3.2 Dữ liệu trong SQL 22](#_Toc61781757)

[Hình 3.3 Class MovieRating 23](#_Toc61781758)

[Hình 3.4 Hàm load dữ liệu từ SQL Server 23](#_Toc61781759)

[Hình 3.5 Gọi hàm load dữ liệu và lấy thông tin data trong main 24](#_Toc61781760)

[Hình 3.6 Tổng quát dữ liệu 24](#_Toc61781761)

[Hình 3.7 Hàm chuẩn bị dữ liệu 24](#_Toc61781762)

[Hình 3.8 Tổng quát tập train và test 25](#_Toc61781763)

[Hình 3.9 Hàm tạo pipeline chuyển đổi dữ liệu và train model 26](#_Toc61781764)

[Hình 3.10 Output của MapValueToKey() 27](#_Toc61781765)

[Hình 3.11 Hyperparamters của Matrix Factorization 27](#_Toc61781766)

[Hình 3.12 Output của BuildAndTrainModel() 28](#_Toc61781767)

[Hình 3.13 Hàm đánh giá model 29](#_Toc61781768)

[Hình 3.14 Đánh giá model 29](#_Toc61781769)

[Hình 3.15 Class MovieRatingPrediction 30](#_Toc61781770)

[Hình 3.16 Hàm dự đoán rating của user dành cho phim 31](#_Toc61781771)

[Hình 3.17 Dự đoán rating của user dành cho phim 31](#_Toc61781772)

[Hình 3.18 Hàm cải thiện model 33](#_Toc61781773)

[Hình 3.19 Output của Cross Validation 33](#_Toc61781774)

[Hình 3.20 Hàm lưu model 34](#_Toc61781775)

[Hình 3.21 Đăng ký PredictionEnginePool 34](#_Toc61781776)

[Hình 3.22 Constructor của MoviesController 35](#_Toc61781777)

[Hình 3.23 API gợi ý phim cho user 36](#_Toc61781778)

[Hình 3.24 Sử dụng API gợi ý phim 37](#_Toc61781779)

# Giới thiệu

## Lý do chọn đề tài

Sự phát triển của internet đã đưa chúng ta vào thế giới với một lượng lớn các phần tử thông tin như âm nhạc, phim ảnh, sách vở, trang web… với những đặc tính khác nhau. Kết quả của những thông tin khổng lồ đó, người ta cảm thấy rối rắm và một câu hỏi đặt ra “Cái nào là thích hợp với tôi hơn?” nảy sinh trong tư duy của họ. May thay, hệ thống gợi ý có thể chỉ ra các thông tin phù hợp trong số thông tin khổng lồ chưa có trật tự, nó sử dụng các kỹ thuật lọc để chọn ra những loại thông tin đặc trưng nhằm hiển thị các phần tử phù hợp với sở thích của người dùng. Theo cách này, hệ thống có tích hợp tính năng gợi ý sẽ thu hút được người dùng cả về sự hài lòng và tin cậy. Các hệ thống gợi ý tiêu biểu như Amazon, Netflix, IMDb, Youtube, Last.fm, MovieLens… đã tăng được số lượng khách truy cập nhờ vào tính năng hỗ trợ quyết định này của hệ thống. Vì những lí do trên nhóm đã quyết định chọn đề tài này làm “Đồ án 3 – Machine Learning”.

## Mục tiêu nghiên cứu

Mục tiêu của báo cáo này là để hiểu sâu hơn về các Matrix Factorization model cho các hệ thống gợi ý bằng cách trả lời các câu hỏi sau:

* Recommendation System (Hệ thống gợi ý) là gì? Collaborative Filtering hoạt động trên nguyên tắc nào?
* Thuật toán Matrix Factorization hoạt động như thế nào?
* Làm thế nào để huấn luyện một Matrix Factorization model và áp dụng nó vào việc gợi ý phim cho người dùng?

## Nhiệm vụ nghiên cứu

Recommendation system là một công cụ để đề xuất nội dung được cá nhân hóa cho người dùng dựa trên hành vi trước đó. Báo cáo này xem xét tác động của việc xem xét mặt hàng và khuynh hướng của người dùng trong việc phân tích nhân tố ma trận để thực hiện các hệ thống gợi ý. Nghiên cứu trước đây đã chỉ ra rằng khuynh hướng của người dùng có tác động đến khả năng dự đoán củas hệ thống gợi ý. Trong nghiên cứu này, hai cách triển khai khác nhau của Matrix Factorization bằng cách sử dụng stochastic gradient descent được áp dụng cho tập dữ liệu MovieLens 10M để trích dẫn các tính năng tiềm ẩn, một trong số đó xem xét khuynh hướng của người dùng và phim. Các thuật toán thực hiện tương tự khi xem xét khả năng dự đoán. Khi kiểm tra các tính năng được trích dẫn từ hai thuật toán, có một mối tương quan chặt chẽ giữa các tính năng được trích dẫn và thể loại phim. Chúng tôi cho thấy rằng mỗi tính năng tạo thành một danh mục phim riêng biệt trong đó mỗi phim được thể hiện như một tổ hợp các danh mục. Chúng tôi cũng cho biết cách sử dụng các tính năng để giới thiệu các bộ phim tương tự.

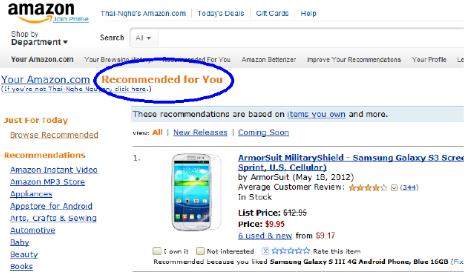
# Cơ sở lý thuyết

## Recommender System (RS)

### Giới thiệu

Hệ thống gợi ý (Recommender Systems – RS) là một dạng của hệ thống lọc thông tin (information filtering), nó được sử dụng để dự đoán sở thích (preferences) hay xếp hạng (rating) mà người dùng có thể dành cho một mục thông tin (item) nào đó mà họ chưa xem xét tới trong quá khứ (item có thể là bài hát, bộ phim, đoạn video clip, sách, bài báo…).

Ví dụ, trong hệ thống bán hàng trực tuyến (chẳng hạn như Amazon), nhằm tối ưu hóa khả năng mua sắm của khách hàng (user), người ta quan tâm đến việc những khách hàng nào đã “yêu thích” những sản phẩm (item) nào bằng cách dựa vào dữ liệu quá khứ của họ (dữ liệu này có thể là xếp hạng mà người dùng đã bình chọn trên sản phẩm, thời gian duyệt (browse) trên sản phẩm, số lần click chuột trên sản phẩm…) từ đó hệ thống sẽ dự đoán được người dùng có thể thích sản phẩm nào và đưa ra những gợi ý phù hợp cho họ.



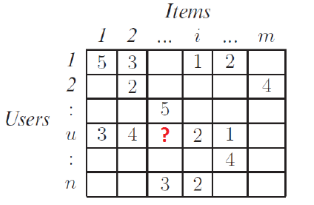
Hình . Hệ thống gợi ý sản phẩm của Amazon

Ngoài lĩnh vực thương mại điện tử như ví dụ trên, hiện tại RS cũng được ứng dụng khá thành công trong nhiều lĩnh vực khác như trong giải trí: gợi ý bài hát cho người nghe (ví dụ, hệ thống của LastFM – [www.last.fm](http://www.last.fm)), gợi ý phim ảnh (ví dụ, hệ thống của Netflix – [www.netflix.com](http://www.netflix.com)), gợi ý các video clip (ví dụ, hệ thống của Youtube – [www.youtube.com](http://www.youtube.com)); trong giao dục và đào tạo (gợi ý nguồn tài nguyên học tập như sách, bài báo, địa chỉ web… cho người học); trong các hệ thống trợ giảng thông minh (như dự đoán năng lực học tập của người học).

Hệ thống gợi ý không chỉ đơn thuần là một dạng Hệ thống thông tin mà nó còn là cả một lĩnh vực nghiên cứu hiện đang rất được các nhà khoa học quan tâm. Kể từ năm 2007 đến nay, hàng năm đều có hội thảo chuyên về hệ thống gợi ý của ACM (ACM RecSys) cũng như các tiểu bang dành riêng cho RS trong các hội nghị lớn khác như ACM KDD, ACM CIKM…

### Các khái niệm chính

Trong RS, thông thường người ta quan tâm đến ba thông tin chính là **người dùng** (user), **mục tin** (item, item có thể là sản phẩm, bộ phim, bài hát, bài báo… tùy hệ thống), và **phản hồi** (feedback) của người dùng trên mục tin đó (thường là các **xếp hạng/đánh giá** – rating biểu diễn mức độ thích/quan tâm của họ). Các thông tin này được biểu diễn thông qua một ma trận như hình dưới. Ở đó, mỗi dòng là một user, mỗi cột là một item, và mỗi ô là một giá trị phản hồi (ví dụ, xếp hạng) biểu diễn “mức độ thích” của user trên item tương ứng. Những ô trống là những item chưa được xếp hạng (lưu ý là mỗi user chỉ xếp hạng cho một vài item trong quá khứ, do vậy có rất nhiều ô trống trong ma trận này – còn gọi là ma trận thưa – sparse matrix).



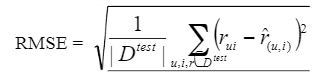
Hình . Ma trận biểu diễn dữ liệu trong RS (user-item-rating matrix)

Nhiệm vụ chính của RS là dựa vào các ô đã có giá trị trong ma trận trên (dữ liệu thu được từ quá khứ), thông qua mô hình đã được xây dựng, RS dự đoán các ô còn trống (của user hiện hành), sau đó sắp xếp kết quả dự đoán (ví dụ, từ cao xuống thấp) và chọn ra Top-N items theo thứ tự, từ đó gợi ý chúng cho người dùng.

Một cách hình thức:

* Gọi **U** là một tập hợp *n* người dùng (user), |**U| =** *n*, và *u* là một người dùng cụ thể nào đó (*u* € **U**).
* Gọi **I** là một tập hợp *m* mục thông tin (item), |**I**| = *m*, và *i* là một mục thông tin cụ thể nào đó (*i* € **I**).
* Gọi **R** là một tập hợp các giá trị phản hồi của người dùng (các phản hồi này thường là “sở thích” (preference) của người dùng) và € **R** (**R** ⸦ ℜ) là xếp hạng của người dùng *u* trên mục thông tin *i*.
* Gọi ⸦ **U** x **I** x **R** là tập dữ liệu huấn luyện.
* Gọi ⸦ **U** x **I** x **R** là tập dữ liệu kiểm thử.
* Gọi *r*: **U** x **I 🡪 R**.

Mục tiêu của RS là tìm một hàm : **U** x **I** 🡪 ℜ sao cho một hàm ξ(*r,* ) thỏa mãn một điều kiện nào đó. Ví dụ, nếu ξ là một hàm ước lượng độ chính xác thì nó cần phải được tối đa hóa, còn nếu ξ là hàm để đo độ lỗi như Root Mean Squared Error thì nó cần phải được tối thiểu.



### Thông tin phản hồi từ người dùng và hai dạng bài toán chính trong RS

Trong RS, giá trị phản hồi (feedback) của mỗi người dùng trên mục tin sẽ được ghi nhận lại để làm cơ sở cho việc dự đoán các giá trị kế tiếp. Tùy theo hệ thống mà giá trị này sẽ có ý nghĩa khác nhau, ví dụ nó có thể dùng để đo độ “phù hợp” hay “mức độ thích” (thường là các đánh giá trên các sản phẩm) trong các hệ thống thương mại điện tử hay “năng lực/kết quả thực hiện” của người dùng trong các hệ thống e-learning.

Giá trị có thể được xác định một các tường minh (explicit feedbacks) như thông qua việc đánh giá/xếp hạng (ví dụ, rating từ Star đến StarStarStarStarStar; hay like (1) và dislike (0), …) mà *u* đã bình chọn cho *i*, hoặc có thể được xác định một cách không tường minh (implicit feedbacks) thông qua số lần click chuột, thời gian mà *u* đã duyệt/xem *i*…

Có 2 dạng bài toán chính trong RS là *dự đoán xếp hạng (rating prediction)* của các hệ thống có phản hồi tường minh như đã trình bày ở trên và *dự đoán mục thông tin* (item prediction/recommendation) là việc xác định xác suất mà người dùng thích mục tin tương ứng.

### Các kỹ thuật chính trong RS

Hiện tại, trong RS có rất nhiều giải thuật được đề xuất, tuy nhiên có thể gom chúng vào trong các nhóm chính.

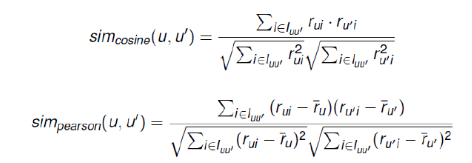
* Nhóm giải thuật lọc cộng tác (Collaborative Filtering): trong nhóm này, các giải thuật chủ yếu sử dụng:
  + Phương pháp láng giềng (Neighborhood-based còn gọi là Memory-based), trong đó hoặc là dựa trên dữ liệu quá khứ của người dùng “tương tự - similarity” (user-based approach), hoặc là dựa trên dữ liệu quá khứ của những item” tương tự” (item-based approach).
  + Dựa trên mô hình (Model-based): Nhóm này liên quan đến việc xây dựng các mô hình dự đoán dựa trên dữ liệu thu thập được trong quá khứ. Như mô hình Bayesian, các mô hình nhân tố tiềm ẩn (matrix factorization) là một điển hình.
* Nhóm giải thuật lọc trên nội dung (Content-based Filtering): Gợi ý các item dựa vào hồ sơ (profiles) của người dùng hoặc dựa vào nội dung/thuộc tính (attributes) của những item tương tự như item mà người dùng đã chọn trong quá khứ.
* Nhóm lai ghép: Kết hợp cả 2 cách trên.
* Nhóm kỹ thuật không cá nhân hóa (non-personalization).

Một trong những khuyết điểm của phương pháp lọc theo nội dung là khó khăn trong việc thu thập thông tin, trong khi phần lớn các mô hình dựa trên lọc cộng tác chỉ cần 3 thông tin (user id, item id, feedback) là có thể hoạt động tốt. Do vậy khuynh hướng hiện nay đa phần các nhà nghiên cứu thiên về nhóm lọc công tác.

#### Kỹ thuật k láng giềng dựa trên người dùng/mục tin (User/Item kNNs)

Có hai cách tiếp cận của lọc cộng tác theo mô hình K láng giếng là dựa trên người dùng (User\_kNN) – dự đoán dựa trên sự tương tự giữa các người dùng và dựa trên mục tin (Item\_kNN) – dự đoán dựa trên sự tương tự giữa các mục tin.

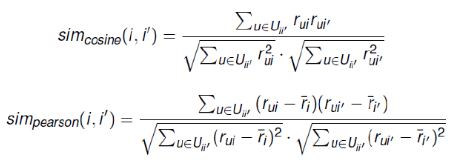
Kỹ thuật k láng giềng dựa trên người dùng (User\_kNN) xác định độ tương tự giữa hai người dùng thông qua việc so sánh các đánh giá của họ trên cùng sản phẩm, sau đó dự đoán xếp hạng trên sản phẩm i bởi người dùng u. Độ tương tự giữa người dùng u và người dùng u’ có thể được tính theo Cosine hoặc Pearson như sau:



Trong đó:

* và là đánh giá của người dùng u và u’ trên item i tương ứng.
* là tập các item được đánh giá bởi cả người dùng u và người dùng u’.’
* là giá trị đánh giá trung bình trên tất cả các item của người dùng u.
* là giá trị đánh giá trung bình trên tất cả các item người dùng u’.

Tương tự, kỹ thuật k láng giềng dựa trên mục tin (Item\_kNN) cũng xác định độ tương tự dựa trên các mục tin bằng phương pháp Cosine hoặc Pearson như sau:

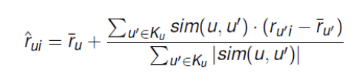


Trong đó:

* là tập các người dùng có đánh trên cả hai mục tin i và i’.
* là giá trị đánh giá trung bình của tất cả các người dùng trên item i.
* là giá trị đánh giá trung bình trên tất cả các người dùng trên item i’.

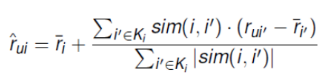
Sau khi tính toán độ tương tự giữa các người dùng hay giữa các sản phẩm, đánh giá của người dùng u trên sản phẩm I được dự đoán theo các công thức như bên dưới.

Với phương pháp User\_kNN, xếp hạng của người dùng u trên sản phẩm i được dự đoán qua công thức:



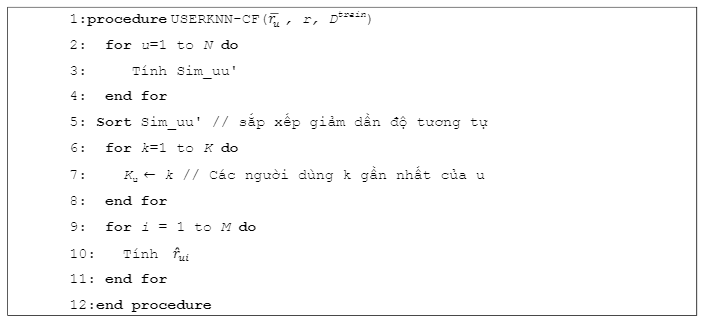
Trong đó *Sim(u, u’)* là độ tương tự giữa người dùng u và u’ được xác định bằng phương pháp Cosine hoặc Pearson như đã trình bày; là số người dùng có độ lân cận gần người dùng u (k láng giếng của u).

Với phương pháp Item\_kNN, xếp hạng của người dùng u trên sản phẩm I được dự đoán qua công thức:



Trong đó *Sim(i, i’)* là độ tương tự giữa mục tin i và i’ được xác định bằng phương pháp Cosine hoặc Pearson như đã trình bày; là số item có độ lân cận gần item i (k láng giếng của i).

Phương pháp User\_kNN để dự đoán đánh giá của người dùng u trên sản phẩm I được biểu diễn bằng ngôn ngữ giả như sau:



Hình . Phương pháp k láng giềng

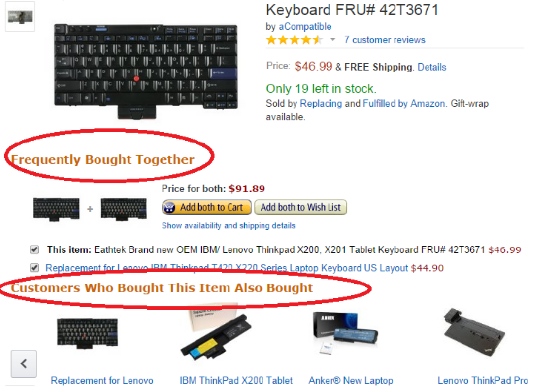
#### Kỹ thuật Phân rã ma trận (Matrix factorization – MF)

Được mô tả cụ thể ở mục 2.4 phần II.

#### Các kỹ thuật không cá nhân hóa

Trong nhóm kỹ thuật này, do chúng khá đơn giản, dễ cài đặt nên thường được các website/hệ thống tích hợp vào, gồm cả các website thương mại, website tin tức, hay giải trí. Chẳng hạn như trong các hệ thống bán hàng trực tuyế, người ta thường gợi ý các sản phẩm được xem/mua/bình luận… nhiều nhất; gợi ý các sản phẩm mới nhất; gợi ý các sản phẩm được mua/chọn cùng nhau. Một ví dụ khá điển hình là thông qua luật kết hợp (như Apriori), Amazon đã áp dụng khá thành công để tìm ra các sản phẩm hay được mua cùng nhau như hình minh họa.

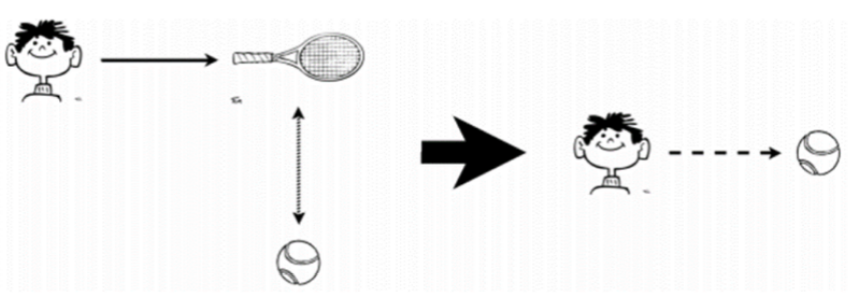
Tuy vậy, bất lợi của các phương pháp này là không cá nhân hóa cho từng người dùng, nghĩa là tất cả các user đều được gợi ý giống nhau khi chọn cùng sản phẩm.



Hình . Gợi ý sản phẩm thường được mua cùng nhau

## Content-based filtering

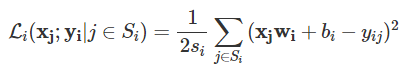
Content-based Filtering: Gợi ý các item dựa vào hồ sơ (profiles) của người dùng hoặc dựa vào nội dung/thuộc tính (attributes) của những item tương tự như item mà người dùng đã chọn trong quá khứ.



Hình . Minh hoạ Content-based Filtering

Đối với mỗi một item chúng ta sẽ tìm cách khởi tạo một item profile bằng cách thu thập các trường thông tin liên quan đến item. Mỗi một item profile được đại diện bởi một véc tơ đặc trưng **x**. Gọi  là tập hợp các sản phẩm mà khách hàng **i** đã rating và giá trị rating là véc tơ . Khi đó chúng ta cần tìm hệ số  là véc tơ cột chứa các hệ số hồi qui thể hiện mức độ yêu thích của khách hàng **i** đối với mỗi một chiều của sản phẩm.

Hàm loss function đối với những sản phẩm mà khách hàng **i** đã rating sẽ có dạng:



Trong đó  là một phần tử của véc tơ ,  là hệ số tự do trong phương trình hồi qui tuyến tính và si là số lượng sản phẩm mà khách hàng **i** đã đánh giá.

Nếu ta trích xuất ra ma trận con  gồm những sản phẩm mà khách hàng **i** đã rating. Mỗi dòng của ma trận là một véc tơ các đặc trưng tương ứng với một sản phẩm. Khi đó hàm loss function có thể được viết gọn lại thành:



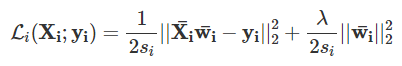
Với  là véc tơ cột gồm  phần từ bằng 1.

Để rút gọn hơn nữa hàm loss function ta biểu diễn nó đưới dạng phương trình của ma trận mở rộng:



Ở đây ma trận  là ma trận mở rộng của  bằng cách thêm một véc tơ cột bằng 1 ở cuối.   cũng là véc tơ mở rộng của khi thêm phần tử 1 ở cuối.

Đây là một phương trình hồi qui tuyến tính quen thuộc nên việc giải nó khá dễ. Trong một số trường hợp để giảm overfiting thì ta sẽ thêm thành phần kiểm soát (regularization term) theo norm chuẩn bậc 2 của  với trọng số là **λ** (thường có giá trị rất nhỏ). Hàm loss function với thành phần kiểm soát sẽ như sau:



Ưu điểm của phương pháp này là việc phân loại hoặc dự báo rating của các user sẽ độc lập nhau. Điểm rating của một khách hàng A lên sản phẩm P sẽ không bị phụ thuộc bởi những user khác mà chỉ phụ thuộc vào các đặc điểm liên quan đến sản phẩm P. Do đó chất lượng dự báo sẽ được tăng lên khi dữ liệu được thu thập về sản phẩm là những trường có quan trọng ảnh hưởng đến sở thích của khách hàng.

## Collaborative filtering

Collaborative filtering (CF) đưa ra dự đoán bằng cách xem xét mối quan hệ giữa user-item mà không cần phải có bất kỳ thông tin bổ sung nào về users hoặc items. Hai cách tiếp cận phổ biến khi triển khai CF recommender systems là neighborhood-based CF và các mô hình yếu tố tiềm ẩn.

Neighborhood-based methods đưa ra dự đoán bằng cách tính toán các điểm tương đồng giữa users hoặc items dựa trên mối quan hệ giữa user-item. Phương pháp hướng user xem xét những users khác có tương tác tương tự khi đưa ra dự đoán. Mặt khác, phương pháp item oriented approach sẽ tính toán độ giống nhau của item với item và đưa ra dự đoán bằng cách tính trọng số các items tương tự mà user đã xếp hạng trước đó.

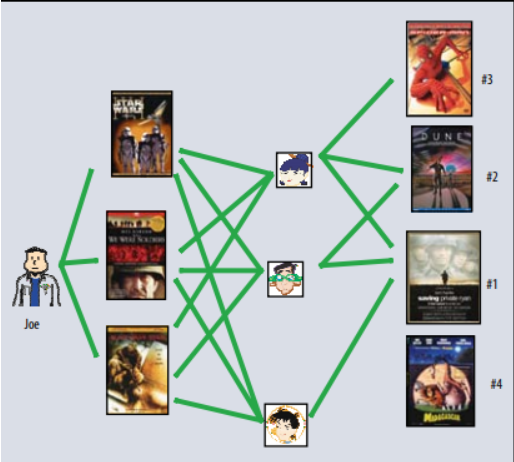
Các mô hình yếu tố tiềm ẩn cho các đề xuất phim sử dụng ma trận xếp hạng mục user để cố gắng mô tả đặc điểm của cả users và items theo một số features tiềm ẩn. Số lượng yếu tố thường có thể là 10-100 cho mỗi user và item. Các vectơ yếu tố cho user và item có thể được nhân với nhau để đưa ra xếp hạng dự đoán cho item. Những yếu tố này có thể được xem như là các tính năng do máy tính suy ra của một item hoặc user. Trong ngữ cảnh của phim, một feature có thể là một thể loại, nhóm tuổi mục tiêu, có thể là một thứ ít rõ ràng hơn như số lượng phát triển của nhân vật hoặc thậm chí hoàn toàn không thể giải thích được. Một phương pháp hệ số tiềm ẩn hiệu quả cho recommender systems được gọi là recommender systems mà chúng tôi sẽ tập trung vào trong báo cáo này.

### Hai lĩnh vực trong Collaborative filtering

Hai lĩnh vực chính của collaborative filtering là neighborhood method (phương pháp lân cận) và latent factors model (mô hình yếu tố tiềm ẩn).

#### Neighborhood method (Phương pháp vùng lân cận)

Phương pháp vùng lân cận tập trung vào việc tính toán các mối quan hệ giữa các items hoặc giữa users. Cách tiếp cận theo định hướng đánh giá sản phẩm dựa trên cách người dùng đánh giá của các “sản phẩm hàng xóm”. Sản phẩm hàng xóm là các sản phẩm khác có xu hướng nhận được đánh giá tương tự khi được đánh giá bởi cùng một người dùng. Ví dụ: Hãy xem phim Giải cứu Binh nhì Ryan. Những người phim “hàng xóm” của nó có thể bao gồm các bộ phim chiến tranh, phim của Spielberg và phim có Tom Hanks. Để dự đoán xếp hạng của một người dùng cụ thể cho Giải cứu Binh nhì Ryan, ta sẽ tìm kiếm những người hàng xóm gần nhất của phim mà người dùng này thực sự đã xếp hạng. Cách tiếp cận hướng người dùng xác định những người dùng cùng sở thích có thể bổ sung cho các đánh giá của nhau.



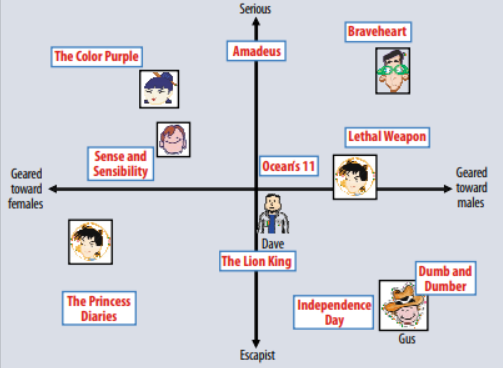
Hình . Phương pháp tiếp cận vùng lân cận

**Giải thích:**

Phương pháp tiếp cận vùng lân cận: Joe thích 3 bộ phim như ở trong ảnh. Hệ thống sẽ tìm ra 3 người dùng tương tự cũng thích 3 bộ phim ấy và những bộ phim khác họ thích. Cả 3 người đều thích Giải cứu Binh nhì Ryan, hệ thống sẽ gợi ý đầu tiên phim ấy cho Joe. 2 người thích Dune nên tiếp theo Joe sẽ được gợi ý Dune, và cứ như vậy...

#### Factors Model (Mô hình yếu tố tìm ẩn)

Mô hình yếu tố tiềm ẩn là một phương pháp thay thế, đưa ra các đánh giá bằng cách mô tả cả sản phẩm lẫn người dùng theo 20 đến 100 yếu tố được suy ra từ các mẫu đánh giá khác. Trong một khía cạnh nào đó, các yếu tố như vậy bao gồm một sự mô hình hóa bằng máy tính đối với các gen bài hát đã được đề cập ở trên. Đối với phim ảnh, các yếu tố được phát hiện có thể đo lường các yếu tố rõ ràng như thiên về hài kịch so với bi kịch; phim hành động hay định hướng cho trẻ em; các yếu tố ít được xác định rõ hơn như nhân vật có chiều sâu hay nhân vật độc đáo; hoặc các yếu tố khó giải thích được khác. Đối với người dùng, mỗi yếu tố quyết định mức độ người dùng thích phim như nào để cho điểm số cao trên hệ số phim tương ứng.



Hình . Mô hình tìm năng

**Giải thích:**

Minh họa của Mô hình tiềm năng: 1 đồ thị đã mô hình hóa cả người dùng lẫn phim bằng 2 yếu tố: hướng nam - hướng nữ và giả tưởng - thực tế.

Minh họa ý tưởng này cho một ví dụ đơn giản trong hai chiều. Hãy xem xét hai khía cạnh giả định được mô tả là hướng nữ giới - hướng nam giới và thực tế - giả tưởng. Hình này cho thấy một số bộ phim nổi tiếng và một số người dùng minh họa có thể thuộc vào hai chiều này. Đối với mô hình này, xếp hạng được dự đoán của người dùng cho một bộ phim, liên quan đến xếp hạng trung bình của phim, sẽ bằng sản phẩm dấu chấm của vị trí của phim và của người dùng trên biểu đồ. Ví dụ, Gus thích Dumb và Dumber, ghét The Color Purple, và để đánh giá Braveheart về mức trung bình. Lưu ý rằng một số phim như: Ocean 11 và người dùng như: Dave sẽ được mô tả là khá trung lập trên hai chiều này.

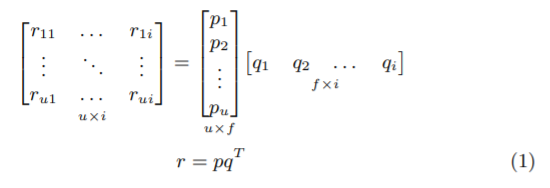
## Thuật toán Matrix factorization (MF)

### Giới thiệu

Matrix factorization (MF) thuộc nhóm Collaborative Filtering trong Recommendation System. Nó là một kỹ thuật tính toán latent features (tính chất ẩn) của một hệ thống dựa trên sự tương tác giữa user – item. Ví dụ trong hệ thống gợi ý các bộ phim, tính chất ẩn có thể là các thể loại phim như: hình sự, chính trị, hành động, hài, …; cũng có thể là một sự kết hợp nào đó của các thể loại này; hoặc cũng có thể là bất cứ điều gì mà chúng ta không thực sự cần đặt tên. Các tương tác với user – item này có thể được biểu diễn dưới dạng ma trận với users trên một trục và các items trên trục khác. Trong đó, mỗi item sẽ mang tính chất ẩn ở một mức độ nào đó tương ứng với các hệ số trong vector xx của nó, hệ số càng cao tương ứng với việc mang tính chất đó càng cao. Tương tự, mỗi user cũng sẽ có xu hướng thích những tính chất ẩn nào đó và được mô tả bởi các hệ số trong vector ww của nó. Hệ số cao tương ứng với việc user thích các bộ phim có tính chất ẩn đó. Giá trị của biểu thức xwxw sẽ cao nếu các thành phần tương ứng của xx và ww đều cao. Điều này nghĩa là item mang các tính chất ẩn mà user thích, vậy thì nên gợi ý item này cho user đó.

Trong hầu hết các tương tác của hệ thống gợi ý phim, tương tác chủ yếu là ratings của users đối với phim, nhưng cũng có thể là dữ liệu khác nhau, chẳng hạn như phản hồi ngầm, hiệu ứng tạm thời và / hoặc mức độ tin cậy. Rating matrix này thường rất thưa thớt trong các ứng dụng thế giới thực vì users thường chỉ đánh giá một phần nhỏ của tất cả các bộ phim trong hệ thống. MF đã được chứng minh là có thể đưa ra dự đoán rất tốt ngay cả trên các ma trận rất thưa thớt.

**Input của MF model sẽ là một ma trận chứa ratings của user đối với item, sau đó model sẽ làm giảm các kích thước của *rating* ma trận r bằng cách phân tích nhân tử thành tích của hai ma trận latent factor (thừa số tiềm ẩn), p cho users và q cho phim.**



Hình . Phương trình (1)

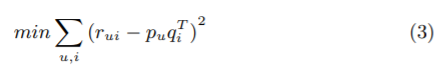
**f là số lượng tính năng được trích xuất, u là số lượng *user* và *i* số lượng *items*.**

**Mỗi hàng pu là một vectơ các đặc trưng cho user u và mỗi hàng qi là một vectơ của các tính năng cho một *item* i. Tích của các vectơ này tạo ra một ước tính của *rating* ban đầu.**



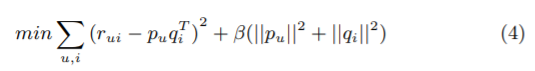
Hình . Phương trình (2)

**Có nhiều cách phân tích nhân tử một ma trận thành nhiều thành phần, được sử dụng trong nhiều lĩnh vực máy học và thống kê, nhưng hầu hết các phương pháp không hoạt động khi thiếu giá trị trong ma trận. Nếu nó có thể được thực hiện, không chỉ các giá trị quan sát được ước tính mà tất cả các giá trị còn thiếu sẽ được dự đoán. Một cách tiếp cận là đưa ra các giá trị còn thiếu, nhưng làm như vậy có thể làm sai lệch dữ liệu được quan sát do sự thưa thớt của ma trận ban đầu. Một cách khác là phân tích nhân tử bằng cách chỉ sử dụng xếp hạng và cố gắng giảm thiểu squared error.**



Hình . **Phương trình (3)**

**Tuy nhiên, điều này có thể dẫn đến việc trang bị quá nhiều dữ liệu training. Để tránh trang bị quá mức, một thuật ngữ chính quy được đưa vào squared error. Tác động của sự chính quy hóa được kiểm soát bởi β hằng số**



Hình . **Phương trình (4)**

**Nơi ||.|| biểu thị tiêu chuẩn frobenius. (4) có thể được tính gần đúng bằng cách sử dụng các thuật toán stochastic gradient descent và alternating least squares. Cách tiếp cận này đã được hiển thị rất hiệu quả và đồng thời có thể mở rộng trên các tập dữ liệu rất lớn. Một ví dụ là cuộc thi Giải thưởng Netflix nơi nó được sử dụng trong hai giải pháp đạt được cao nhất. Việc phân tích nhân tử có thể được thực hiện trước và ít bộ nhớ hơn được sử dụng khi đưa ra dự đoán (hai vectơ có kích thước nf) so với các cách tiếp cận neighborhood trong đó toàn bộ ma trận rating hoặc một tập hợp con của nó cần được lưu trong bộ nhớ.**

### Mô hình Matrix factorization cơ bản

Mô hình phân tích ma trận thành nhân tử (Matrix factorization) ánh xạ cả người dùng lẫn sản phẩm đến không gian yếu tố tiềm ẩn chung của không gian f, nhờ vậy mối quan hệ giữa người dùng - sản phẩm được mô hình hóa thành một phần tử thuộc không gian đấy. Thông qua đó, mỗi item i được liên kết với 1 vector q\_i ϵR^f*qi*​*ϵRf*. Và mỗi user được liên kết với 1 vector p\_u ϵR^f*pu*​*ϵRf*. Với mỗi item i xác định, phần tử q\_i*qi*​ đánh giá những yếu tố mà item đó sở hữu, dương hay âm. Đối với mỗi user u, phần tử p\_u*pu*​ đánh giá những mối quan tâm của người dùng, dương hay âm. Tích giữa q\_i*qi*​ và p\_u*pu*​ thể hiện mối quan hệ giữa người dùng u và sản phẩm i, q\_i p\_u*qi*​*pu*​ thể hiện đánh giá tổng thể của người dùng về item cụ thể. Điều này sẽ xấp xỉ rating của user dành cho sản phẩm, r\_ui*ru*​*i*, dẫn đến công thức:

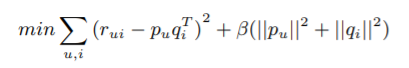


Hình . Phương trình xấp xỉ rating của user

Thách thức của thuật toán này chính là việc learning, tức là tính toán ánh xạ giữa item và người dùng để tạo thành vector q\_i*qi*​, p\_u*pu*​. Sau khi hệ gợi ý hoàn thành tính toán xong việc ánh xạ, chúng ta có thể dễ dàng ước lượng được đánh giá của người dùng bằng công thức trên.

Mô hình này gần giống với mô hình SVD (Singular Value Decomposition). Việc áp dụng SVD trong các chiến thuật lọc cộng tác đòi hỏi phân tích ma trận rating của user-item. Điều này thường làm tăng độ khó khăn bởi lượng lớn các giá trị bỏ trống do các ma trận rating của user-item thường khá thưa thớt. Các mô hình SVD thông thường thường không xác định khi tri thức về ma trận không đầy đủ. Hơn nữa, việc chỉ ghi lại những các mục đã biết, trong khi số lượng các mục này tương đối ít, sẽ chỉ càng gây thêm vấn đề overfitting.

Hệ thống nêu trên đòi hỏi vào việc gán giá trị để lấp đầy những giá trị còn thiếu và làm cho ma trận trở nên kín hơn. Tuy nhiên, việc gán giá trị có thể trở nên khá tốn kém khi số lượng dữ liệu gia tăng. Thêm vào đó, việc gán giá trị sai có thể khiến việc xem xét dữ liệu trở nên không đúng. Vì thế, những hệ thống gợi ý gần đây mô hình hóa trực tiếp từ quan sát rating, bỏ qua sự overfitting trong mô hình chuẩn hóa. Để học các vector yếu tố (q\_i*qi*​ và p\_u*pu*​), hệ thống giảm thiểu tối đa sự sai sót do chuẩn hóa trên tập các rating đã biết



Hình . Phương trình giảm thiểu tối đa sự sai sót do chuẩn hóa

Trong đó, K là tập hợp các cặp (u, i) mà r\_ui*ru*​*i* của chúng đã được xác định.

Hệ thống sẽ học mô hình bằng cách cố gắng phù hợp chúng với những rating đã quan sát được chúng trước đó. Tuy nhiên, mục tiêu ở đây là tổng quát hóa những rating trước đây theo một cách nào đó để có thể dự đoán được các rating sau này, hay các rating chưa biết.

### Stochastic gradient descent

Trong thuật toán này, tại 1 thời điểm, ta chỉ tính đạo hàm của hàm mất mát dựa trên *chỉ một* điểm dữ liệu xi rồi cập nhật θ dựa trên đạo hàm này. Việc này được thực hiện với từng điểm trên toàn bộ dữ liệu, sau đó lặp lại quá trình trên. Thuật toán rất đơn giản này trên thực tế lại làm việc rất hiệu quả.

Mỗi lần duyệt một lượt qua *tất cả* các điểm trên toàn bộ dữ liệu được gọi là một epoch. Với GD thông thường thì mỗi epoch ứng với 1 lần cập nhật θ, với SGD thì mỗi epoch ứng với NN lần cập nhật θ với NN là số điểm dữ liệu. Nhìn vào một mặt, việc cập nhật từng điểm một như thế này có thể làm giảm đi tốc độ thực hiện 1 epoch. Nhưng nhìn vào một mặt khác, SGD chỉ yêu cầu một lượng epoch rất nhỏ (thường là 10 cho lần đầu tiên, sau đó khi có dữ liệu mới thì chỉ cần chạy dưới một epoch là đã có nghiệm tốt). Vì vậy SGD phù hợp với các bài toán có lượng cơ sở dữ liệu lớn (chủ yếu là Deep Learning mà chúng ta sẽ thấy trong phần sau của blog) và các bài toán yêu cầu mô hình thay đổi liên tục, tức online learning.

**Thứ tự lựa chọn điểm dữ liệu**

Một điểm cần lưu ý đó là: sau mỗi epoch, chúng ta cần shuffle (xáo trộn) thứ tự của các dữ liệu để đảm bảo tính ngẫu nhiên. Việc này cũng ảnh hưởng tới hiệu năng của SGD.

Một cách toán học, quy tắc cập nhật của SGD là:



Hình . Quy tắc cập nhật của SGD

trong đó J(θ;xi;yi)J(θ;xi;yi) là hàm mất mát với chỉ 1 cặp điểm dữ liệu (input, label) là (xi,yixi,yi). **Chú ý:** chúng ta hoàn toàn có thể áp dụng các thuật toán tăng tốc GD như Momentum, AdaGrad… vào SGD.

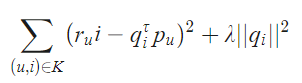
### Phương pháp Alternating least squares

ALS hay còn gọi là phương pháp bình phương tối thiểu thay phiên nhau. Gọi là thay phiên nhau bởi hàm mất mát trên có 2 biến khiến hàm không lồi (khó để tìm cực tiểu) chính vì vậy ta thay phiên cố định 1 biến rồi tối ưu hàm theo biến còn lại. Khi cố định 1 biến hàm trở thành hàm bậc 2 và có thể giải bằng phương pháp bình phương tối thiểu. Việc cố định 1 biến rồi tính toán lại biến còn lại được thực hiện lần lượt cho đến khi các điểm hội tụ lại tại điểm cực tiểu.

Tựu chung thì SGD dễ sử dụng và tốc độ nhanh hơn ALS tuy nhiên có 2 trường hợp mà ALS lại được ưu tiên hơn.

Trường hợp 1 là hệ thống có khả năng chạy các tiến trình 1 cách song song.

Ta thấy khi cố định 1 trong 2 cột công thức hàm cần tối ưu sẽ trở thành



Hình . Phương trình ALS

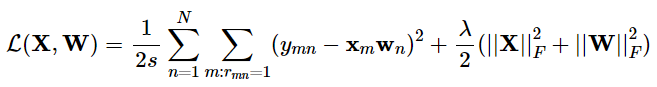
Từ đó bài toán có thể chuyển thành việc tối ưu từng cột 1 của q (hoặc p) cho nên việc sử dụng trong hệ thống song song sẽ dễ dàng hơn. Trong khi nếu SGD sử dụng trong hệ thống song song sẽ dễ gặp vấn đề bị ghi đè nếu kết quả cập nhật có liên quan đến 2 phép toán diễn ra song song.

Trường hợp thứ 2 là khi sử dụng với hệ thống tập trung vào các dữ liêu ẩn. Bởi không phải lúc nào tập huấn luyện cũng thưa thớt, nên việc lặp lại qua tất cả các điểm dữ liệu theo cách của SGD lại trở lên không thực tiễn. Trong trường hợp này sử dụng ALS cũng có hiệu quả cao hơn.

### Matrix factorization với bias

Một lợi thế của hướng tiếp cận Matrix Factorization cho Collaborative Filtering là khả năng linh hoạt của nó khi có thêm các điều kiện ràng buộc khác, các điều kiện này có thể liên quan đến quá trình xử lý dữ liệu hoặc đến từng ứng dụng cụ thể.

Giả sử ta chưa chuẩn hoá các giá trị *ratings* mà sử dụng trực tiếp giá trị ban đầu của chúng trong đẳng thức:



Hình . Phương trình maxtrix factorizstion với bias

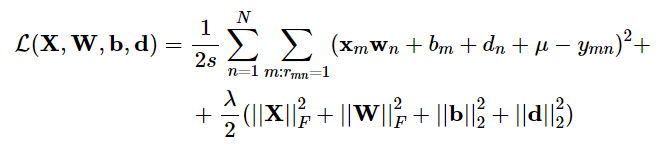
Việc chuẩn hoá cũng có thể được tích hợp trực tiếp vào trong hàm mất mát. Các *ratings* thực tế đều có những thiên lệch về *users* hoặc/và *items*. Có *user* dễ và khó tính, cũng có những *item* được *rated* cao hơn những *items* khác chỉ vì *user* thấy các *users* khác đã đánh giá *item* đó cao rồi. Vấn đề thiên lệch có thể được giải quyết bằng các biến gọi là *biases*, phụ thuộc vào mỗi *user* và *item* và có thể được tối ưu cùng với XX và WW. Khi đó, *ratings* của *user* nn lên *item* mm không chỉ được xấp xỉ bằng xmwnxmwn mà còn phụ thuộc vào các *biases* của *item* mm và *user* nn nữa. Ngoài ra, giá trị này cũng có thể phụ thuộc vào giá trị trung bình của toàn bộ *ratings* nữa:



Hình . Giá trị trung bình của toàn bộ ratings

với bm,dn,μbm,dn,μ lần lượt là bias của *item* mm, *user* nn, và giá trị trung bình của toàn bộ các *ratings* (là hằng số).

Lúc này, hàm mất mát có thể được thay đổi thành:



Hình . Phương trình hàm mất mát

### Mức độ liên quan và nghiên cứu trước đó

Các kỹ thuật Matrix factorization có thể được áp dụng cho một số hệ thống khác, không chỉ recommender systems. Việc hiểu các đặc điểm được trích dẫn là kết quả của quá trình phân tích nhân tử có thể cung cấp thông tin chi tiết sâu hơn về dữ liệu và cũng có thể sử dụng ngoài việc đưa ra dự đoán xếp hạng. Nghiên cứu trước đây đã chỉ ra rằng sử dụng bias có thể cải thiện hiệu suất dự đoán của mô hình matrix factorization. Báo cáo này sẽ so sánh mô hình này với một mô hình phân rã thông thường mà không sử dụng bias để hiểu chúng khác nhau như thế nào và những cải tiến nào có thể được thực hiện.

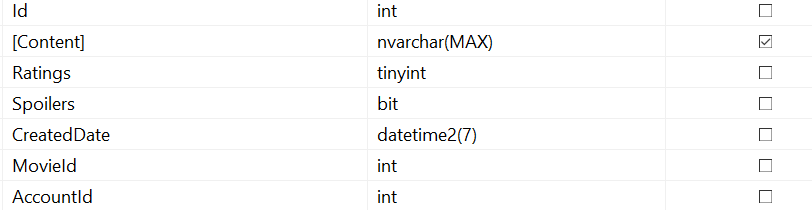
# Cài đặt thuật toán

## Huấn luyện model

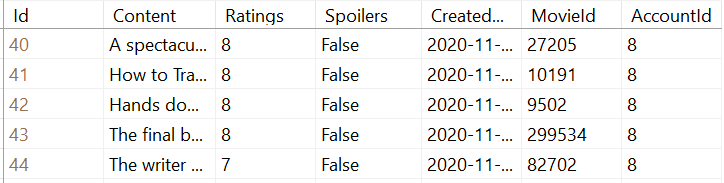
### Load dữ liệu và tìm hiểu dữ liệu

Dữ liệu của model được load từ bảng Reviews trong SQL, đây là bảng dùng để lưu các bài đánh giá của người dùng về phim. Trong Machine Learning, các cột được sử dụng để tạo dự đoán được gọi là Features và cột có dự đoán trả về được gọi là Label. Ở bài toán này, cột Ratings sẽ là Label và các cột còn lại sẽ là Features, tùy thuộc vào việc chúng ta chọn Features như thế nào để thuật toán trả về Label chính xác nhất.

Ở đây, cột [Content] (nội dung bài đánh giá) không thực sự quan trọng vì chúng ta đã có Ratings, cũng như Spoilers (đánh giá có tiết lộ tình tiết phim) và cột CreatedDate cũng như thế. Cho nên, chúng ta chỉ cần sử dụng cột AccountId (người viết bài đánh giá), MovieId (Id của bộ phim được đánh giá) và Ratings (điểm người dùng đánh giá phim: 0 – 10).

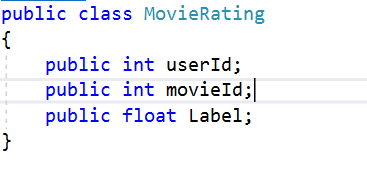


Hình . Schema dữ liệu SQL



Hình . Dữ liệu trong SQL

Trong ML.Net, khi load dữ liệu chúng ta cần chỉ rõ kiểu dữ liệu mà chúng ta sẽ load, nên chúng ta cần tạo một class gọi là MovieRating để chứa các cột dữ liệu load được từ database.



Hình . Class MovieRating

Tạo connection string và DatabaseSource, sử dụng DatabaseLoader của MLContext để load dữ liệu từ SQL Server.

public static IDataView LoadData(MLContext mlContext)

{

DatabaseLoader loader = mlContext.Data.CreateDatabaseLoader<MovieRating>();

string connectionString = @"Data Source=(Local);Database=MovieReviews;Integrated Security=True;Connect Timeout=30";

string sqlCommand = "SELECT AccountId as userId, MovieId as movieId, Ratings as Label FROM dbo.Reviews";

DatabaseSource dbSource = new DatabaseSource(SqlClientFactory.Instance, connectionString, sqlCommand);

return loader.Load(dbSource);

}

Hình . Hàm load dữ liệu từ SQL Server

static void Main(string[] args)

{

MLContext mlContext = new MLContext();

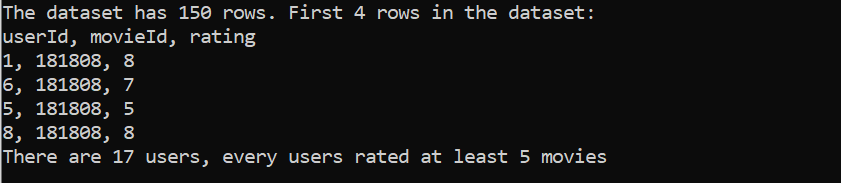
IDataView dataView = LoadData(mlContext);

GetDataInsights(dataView, mlContext);

}

Hình . Gọi hàm load dữ liệu và lấy thông tin data trong main

Dataset load được từ database có tổng cộng 150 dòng, có tổng cộng 17 user đã đánh giá các bộ phim, mỗi user đánh giá ít nhất 5 phim.



Hình . Tổng quát dữ liệu

### Chuẩn bị dữ liệu

Sử dụng TrainTestSplit của ML.Net để tách testFraction phần trăm dữ liệu khỏi dataset để làm tập Test, còn lại dùng cho tập Train. Dữ liệu tập Train được sử dụng để fit model. Dữ liệu Test được sử dụng để đưa ra dự đoán với model đã được đào tạo và đánh giá hiệu suất model.

public static (IDataView training, IDataView test) PrepareData (IDataView data, MLContext mlContext)

{

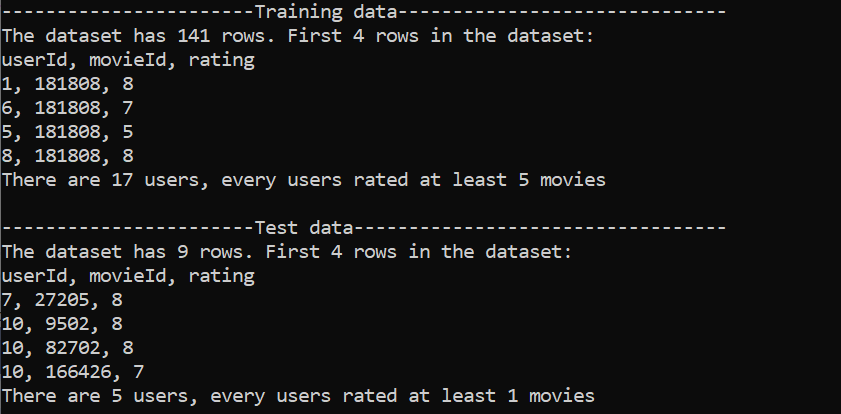
var split = mlContext.Data.TrainTestSplit(data, testFraction: 0.1, samplingKeyColumnName: null);

return (split.TrainSet, split.TestSet);

}

Hình . Hàm chuẩn bị dữ liệu

Đối với thuật toán Matrix Factorization, quá trình tách dữ liệu chúng ta cần để lại đầy đủ dữ liệu về profile của từng người dùng. Có thể thấy, sau khi tách dữ liệu ban đầu ra 2 tập train và test, tập train vẫn nắm giữ đủ các rating của tất cả user (17 user).



Hình . Tổng quát tập train và test

### Xây dựng và huấn luyện model

Có 2 khái niệm chính trong ML.Net: Transformers và Estimators. Các thuật toán Machine Learning yêu cầu dữ liệu ở một định dạng nhất định. Transformers được sử dụng để chuyển đổi dữ liệu dạng bảng sang định dạng tương thích. Còn Estimators là một dạng đặc biệt của Transformers, là một pipeline của sự chuyển đổi (Transformers chuẩn bị dữ liệu và Transformers đào tạo model theo thuật toán ML). Các tham số của Estimator hoặc Pipeline của Estimator được học khi Fit được gọi. Kết quả trả về của Fit là 1 Transformer.

public static ITransformer BuildAndTrainModel(MLContext mlContext, IDataView trainingDataView)

{

IEstimator<ITransformer> estimator = mlContext.Transforms.Conversion.MapValueToKey(outputColumnName: "userIdEncoded", inputColumnName: "userId")

.Append(mlContext.Transforms.Conversion.MapValueToKey(outputColumnName: "movieIdEncoded", inputColumnName: "movieId"));

var options = new MatrixFactorizationTrainer.Options

{

MatrixColumnIndexColumnName = "userIdEncoded",

MatrixRowIndexColumnName = "movieIdEncoded",

LabelColumnName = "Label",

NumberOfIterations = 50,

ApproximationRank = 100

};

var trainerEstimator = estimator.Append(mlContext.Recommendation().Trainers.MatrixFactorization(options));

Console.WriteLine("-----------------------Training the model------------------------------");

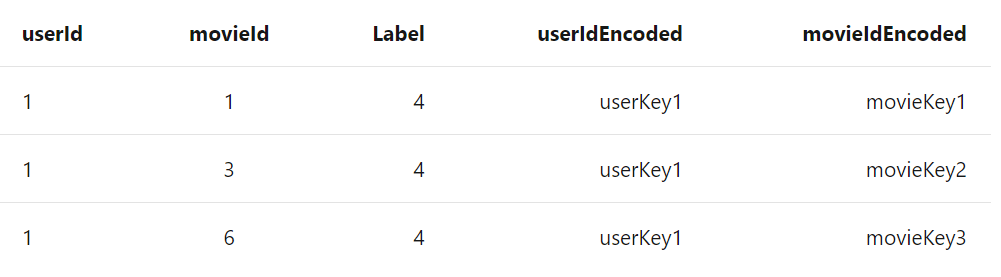
ITransformer model = trainerEstimator.Fit(trainingDataView);

return model;

}

Hình . Hàm tạo pipeline chuyển đổi dữ liệu và train model

Vì userId và movieId đại diện cho người dùng và phim, không phải giá trị thực, chúng ta sử dụng phương thức MapValueToKey() để chuyển đổi mỗi userId và mỗi movieId thành kiểu dữ liệu Feature (kiểu dữ liệu chấp nhận bởi các thuật toán recomendation system). MapValueToKey đọc cột userId và xây dựng 1 dictionary các giá trị ID duy nhất. Sau đó, nó tạo ra một cột đầu ra được gọi là userIdEncoded chứa mã hóa cho mỗi ID, tương tự như với movieId.



Hình . Output của MapValueToKey()

Khởi tạo giá trị các options (Hyperparamters) của Matrix Factorization như sau:

* MatrixColumnIndexColumnName là index cột của matrix sẽ giá trị user đã MapValueToKey().
* MatrixRowIndexColumnName là index hàng của matrix sẽ giá trị movie đã MapValueToKey().
* Cột Label.
* NumberOfIterations là số lần thực hiện train.
* ApproximationRank: nếu dữ liệu đầu vào có kích thước là m-x-n, thuật toán sẽ xây dựng hai ma trận xấp xỉ m\*k và k\*n trong đó k là ApproximationRank.

var options = new MatrixFactorizationTrainer.Options

{

MatrixColumnIndexColumnName = "userIdEncoded",

MatrixRowIndexColumnName = "movieIdEncoded",

LabelColumnName = "Label",

NumberOfIterations = 20,

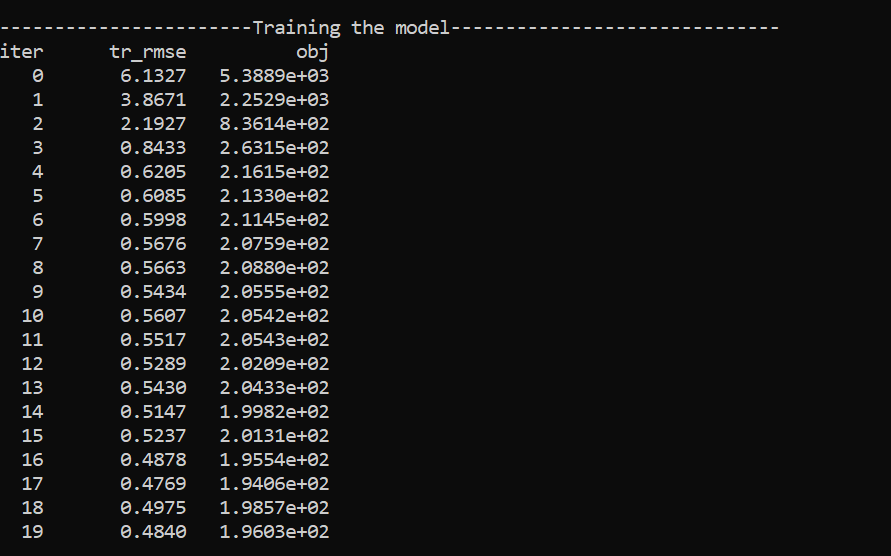
ApproximationRank = 100

};

var trainerEstimator = estimator.Append(mlContext.Recommendation().Trainers.MatrixFactorization(options));

Hình . Hyperparamters của Matrix Factorization

Thêm hàm BuildAndTrainModel() vào hàm main, chạy chương trình và ta sẽ có kết quả như sau:



Hình . Output của BuildAndTrainModel()

Trong đầu ra này, có 20 lần lặp. Trong mỗi lần lặp, số đo RMSE (Root Mean Square Error) giảm dần và dần hội tụ dần về 0. Root Mean Square Error (RMS hoặc RMSE) được sử dụng để đo sự khác biệt, sai số giữa các giá trị dự đoán của model và các giá trị quan sát được của tập dữ liệu thử nghiệm. Về mặt kỹ thuật, nó là căn bậc hai của trung bình các bình phương của các lỗi. Nó luôn luôn là một số dương và càng thấp thì model dự đoán càng tốt.

### Đánh giá model

Mỗi một loại model trong Machine Learning đều có các đại lượng riêng để đánh giá độ tốt của mode đó. Ví dụ: đối với bài toán phân loại (Classification), model được đánh giá bằng cách đo lường mức độ phù hợp của label dự đoán với label thực tế. Và đối với phân cụm (Clustering), đánh giá dựa trên mức độ gần của các điểm dữ liệu được phân nhóm với nhau và mức độ phân cách giữa các cụm.

Đối với các bài toán Regression và Recommendation đều dùng để dự đoán một con số. Đối với Recommendation, con số thường là giá trị xếp hạng (ví dụ từ 1 đến 5) hoặc đề xuất có / không (được biểu thị bằng 1 và 0 tương ứng). ML.Net cung cấp cho chúng ta các đại lượng đánh giá như sau: R-squared (R2), Absolute-loss hay còn gọi là Mean absolute error (MAE), Squared-loss hay còn gọi là Mean Squared Error (MSE) và RMS-loss hay còn gọi là Root Mean Squared Error (RMSE). Ở đây, chúng ta sẽ sử dụng hai phương thức đánh giá tiêu biểu là RSME và R-squared.

public static void EvaluateModel(MLContext mlContext, IDataView testDataView, ITransformer model)

{

var prediction = model.Transform(testDataView); // đầu vào là data gồm 5 cột: userId, movieId, label, userIdEncoded và movieIdEncoded, output: 5 cột + Score

var metrics = mlContext.Regression.Evaluate(prediction, labelColumnName: "Label", scoreColumnName: "Score");

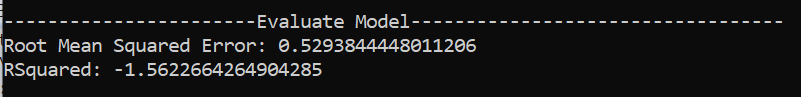
Console.WriteLine("Root Mean Squared Error: " + metrics.RootMeanSquaredError.ToString());

Console.WriteLine("RSquared: " + metrics.RSquared.ToString());

}

Hình . Hàm đánh giá model

R-squared hay còn gọi là R2 cho biết dữ liệu phù hợp với mô hình như thế nào và nó có giá trị từ -inf đến 1,00. Giá trị 1 có nghĩa là model khớp chính xác với dữ liệu 1 cách hoàn hảo. Trường hợp số âm do model có thể học kém nên điểm số bị âm. Giá trị 0 có nghĩa là dữ liệu là ngẫu nhiên hoặc không thể phù hợp với model. Chúng ta sẽ muốn điểm R-squared của model càng gần 1 càng tốt.

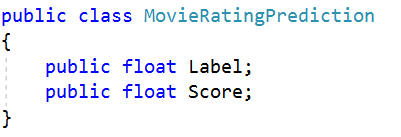


Hình . Đánh giá model

Xây dựng các model thành công là một quá trình lặp đi lặp lại. Model này có chất lượng ban đầu thấp hơn vì lượng dữ liệu đào tạo model còn ít. Nếu không hài lòng với chất lượng model, chúng ta có thể cố gắng cải thiện nó bằng cách cung cấp bộ dữ liệu đào tạo lớn hơn, chọn lại các Hyperparameters cho thuật toán hoặc chọn các thuật toán đào tạo khác nhau với các Hyperparameters khác nhau.

### Sử dụng model

Chúng ta sử dụng phương thức CreatePredictionEngine() để thiết lập công cụ dự đoán. Hai đối số kiểu đầu vào của phương thức là class dữ liệu đầu vào (MovieRating) và class giữ kết quả dự đoán (MovieRatingPrediction). Và khi công cụ dự đoán của chúng ta đã được thiết lập, chúng ta có thể chỉ cần gọi Predict (…) để đưa ra dự đoán độ yêu thích của người dùng lên một phim nào đó.



Hình . Class MovieRatingPrediction

public static void UseModelForSinglePrediction(MLContext mlContext, ITransformer model)

{

var predictionEngine = mlContext.Model.CreatePredictionEngine<MovieRating, MovieRatingPrediction>(model);

var testInput = new MovieRating { userId = 17, movieId = 10191 };

var movieRatingPrediction = predictionEngine.Predict(testInput);

Console.WriteLine($"User: {testInput.userId} is predicted to rate movie {testInput.movieId} {movieRatingPrediction.Score} stars");

if (Math.Round(movieRatingPrediction.Score, 1) > 7)

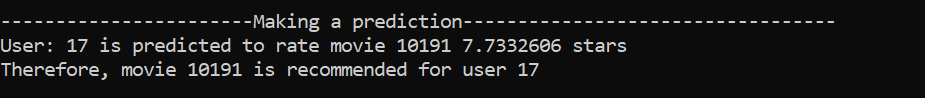
Console.WriteLine("Therefore, movie " + testInput.movieId + " is recommended for user " + testInput.userId);

else Console.WriteLine("Therefore, movie " + testInput.movieId + " is not recommended for user " + testInput.userId);

}

Hình . Hàm dự đoán rating của user dành cho phim

Bằng việc sử dụng model đã huấn luyện, chúng ta sẽ dự đoán mức độ yêu thích của user với id 17 với bộ phim với id 10191 là bao nhiêu. Lưu ý đây là phim mà user 17 chưa đánh giá, nếu dự đoán rating của user 17 cho phim lớn hơn 7 thì chúng ta sẽ gợi ý phim 10191 cho user 17. Bộ phim 10191 được các user khác đánh giá rất cao nên dựa theo mức độ yêu thích của user tương tự, chúng ta sẽ gợi ý phim cho user 17.



Hình . Dự đoán rating của user dành cho phim

### Cải thiện model:

Chúng ta có thể cải thiện model bằng cách thêm nhiều dữ liệu đào tạo, có đủ sample cho mỗi người dùng và id phim có thể giúp cải thiện chất lượng của model.

Nhưng vì hiện tại dữ liệu của chúng ta chỉ có 150 sample nên chúng ta sẽ sử dụng phương pháp Cross-Validation. Cross-validation là một kỹ thuật lấy mẫu để đánh giá mô hình học máy trong trường hợp dữ liệu không được dồi dào cho lắm. Cross validation đánh giá các model bằng cách chia ngẫu nhiên dữ liệu thành các tập con (thay vì trích xuất dữ liệu test từ tập dữ liệu train như chúng ta đã làm) và lấy một số nhóm làm dữ liệu train và một số nhóm làm dữ liệu test. Phương pháp này vượt trội hơn so với việc sử dụng TrainTestSplit ​​về chất lượng model.

public static ITransformer ImproveModel(MLContext mlContext, IDataView trainingDataView)

{

// Tạo pipeline

IEstimator<ITransformer> estimator = mlContext.Transforms.Conversion.MapValueToKey(outputColumnName: "userIdEncoded", inputColumnName: "userId")

.Append(mlContext.Transforms.Conversion.MapValueToKey(outputColumnName: "movieIdEncoded", inputColumnName: "movieId"));

var options = new MatrixFactorizationTrainer.Options

{

MatrixColumnIndexColumnName = "userIdEncoded",

MatrixRowIndexColumnName = "movieIdEncoded",

LabelColumnName = "Label",

NumberOfIterations = 20,

ApproximationRank = 100

};

var trainerEstimator = estimator.Append(mlContext.Recommendation().Trainers.MatrixFactorization(options));

// Chạy Cross Validation

var cvResults = mlContext.Regression.CrossValidate(trainingDataView, trainerEstimator, numberOfFolds: 5);

// Kết quả

Console.WriteLine("\nRMSE and R2 of 5 Cross Validation");

foreach (var fold in cvResults)

Console.WriteLine($"RMSE: {fold.Metrics.RootMeanSquaredError}, R2: {fold.Metrics.RSquared}");

// Lấy model tốt nhất

var bestFold = cvResults.OrderByDescending(fold => fold.Metrics.RSquared).FirstOrDefault();

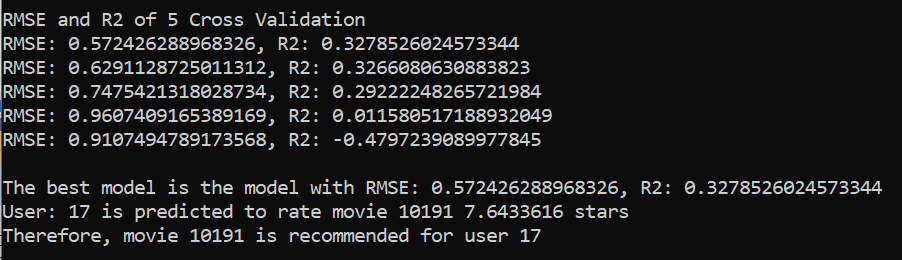
Console.WriteLine($"\nThe best model is the model with RMSE: {bestFold.Metrics.RootMeanSquaredError}, R2: {bestFold.Metrics.RSquared}");

return bestFold.Model;

}

Hình . Hàm cải thiện model

Tham số quan trọng trong kỹ thuật này là k (numberOfFolds), đại diện cho số nhóm mà dữ liệu sẽ được chia ra. Vì lý do đó, nó còn được gọi là K-fold Cross-validation. Trong hàm trên, k là 5. K-fold Cross-validation sẽ chia dataset thành 5 nhóm và ta sẽ có 5 lần chạy, với mỗi lần chạy chọn 1 nhóm làm test data, 4 nhóm còn lại sẽ là train data. Kết quả của Cross-validation đã cho chúng ta model có R2 cao hơn và RSME thấp hơn kết quả ở bước đánh giá model.



Hình . Output của Cross Validation

Ngoài cách sử dụng Cross-validation, chúng ta có thể thêm Feature cho dữ liệu mà hiện tại chỉ đang có 3 cột như là tuổi, giới tính, vị trí ... của user.

Cuối cùng, mặc dù ML.NET cung cấp các thuật toán đào tạo tốt, chúng ta có thể tinh chỉnh thêm hiệu suất của model bằng cách thay đổi các siêu tham số (Hyperparameters) của thuật toán. Đối với Matrix Factorization, chúng ta có thể thử nghiệm với các Hyperparameters như NumberOfIterations và ApproximationRank để xem liệu điều đó có mang lại kết quả tốt hơn hay không.

### Lưu model

Để sử dụng model để đưa ra dự đoán trong các ứng dụng người dùng cuối, trước tiên chúng ta phải lưu model.

public static void SaveModel(MLContext mlContext, DataViewSchema trainingDataViewSchema, ITransformer model)

{

var modelPath = Path.Combine(Environment.CurrentDirectory, "Data", "MovieRecommenderModel.zip");

mlContext.Model.Save(model, trainingDataViewSchema, modelPath);

}

Hình . Hàm lưu model

## Áp dụng model vào ứng dụng

Chúng ta sẽ áp dụng model đã được huấn luyện và cải thiện vào ứng dụng web ASP.NET Core Web API. Để sử dụng được model này, đầu tiên project của chúng ta sẽ cần cài đặt 3 package hỗ trợ Machine Learning của Microsoft đó là Microsoft.ML, Microsoft.ML.Recommender và Microsoft.Extensions.ML.

### Khởi tạo môi trường và load model đã huấn luyện

Như ở phần trước, để đưa ra một dự đoán duy nhất, chúng ta phải tạo PredictionEngine. PredictionEngine cần được khởi tạo phiên bản mới của nó ở mọi nơi cần thiết trong ứng dụng. Khi ứng dụng ngày càng phát triển, quá trình này có thể trở nên không thể quản lý được. Để cải thiện hiệu suất và an toàn luồng, chúng ta sử dụng PredictionEnginePool thay cho PredictionEngine, dịch vụ này tạo ra một ObjectPool của các đối tượng PredictionEngine để sử dụng trong toàn bộ ứng dụng web. Ở Startup.cs, chúng ta sẽ đăng ký một PredictionEnginePool để load model đã huấn luyện lên như sau.

Các model Machine Learning không ở trạng thái tĩnh. Khi có dữ liệu mới, mô hình được đào tạo lại và triển khai lại. Một cách để đưa phiên bản mới nhất của model vào ứng dụng của chúng ta là triển khai lại toàn bộ ứng dụng. Tuy nhiên, điều này đây là cách làm không thực tế vì phải làm cho ứng dụng ngừng hoạt động. PredictionEnginePool cung cấp cơ chế tải lại model đã cập nhật mà không cần gỡ ứng dụng xuống, chúng ta chỉ cần đặt watchForChanges thành true.

services.AddPredictionEnginePool<MovieRating, MovieRatingPrediction>().FromFile(Configuration["MLModelPath"], watchForChanges: true);

Hình . Đăng ký PredictionEnginePool

### Tạo 1 API để lấy phim gợi ý cho user:

MoviesController là nơi sẽ chứa API gợi ý của chúng ta, vì đã đăng ký một PredictionEnginePool nên chúng ta chỉ cần PredictionEnginePool vào constructor của MoviesController để sử dụng model đã huấn luyện cho việc gợi ý.

private readonly ApplicationDbContext \_context;

private readonly PredictionEnginePool<MovieRating, MovieRatingPrediction> \_model;

public MoviesController(ApplicationDbContext context, PredictionEnginePool<MovieRating, MovieRatingPrediction> model)

{

\_context = context;

\_model = model;

}

Hình . Constructor của MoviesController

Ở phần huấn luyện model, chúng ta chỉ thực hiện dự đoán rating của user lên một phim. Còn ở phần này, để lấy được gợi ý nhiều phim cho user, chúng ta sẽ thực hiện dự đoán đơn trên các phim mà user chưa thực hiện đánh giá trên hệ thống.

[HttpGet("Recommend/{userId}")]

public async Task<ActionResult<IEnumerable<MovieVM>>> RecommendMovies(int userId)

{

var reviews = await \_context.Reviews.ToListAsync();

// Lấy id của những phim mà user đã đánh giá

var ratedIds = reviews.Where(r => r.AccountId == userId).Select(r => r.MovieId).ToList();

// Sử dụng model để dự đoán rating của userId lên một số phim

var recommendedMovies = new List<Movie>();

MovieRatingPrediction prediction = null;

foreach (var movie in \_context.Movies.ToList())

{

movie.Reviews = null;

if (ratedIds.Contains(movie.Id)) continue;

prediction = \_model.Predict(new MovieRating

{

userId = userId,

movieId = movie.Id

});

if (prediction.Score >= 7) recommendedMovies.Add(movie);

// Chỉ recommend 6 phim

if (recommendedMovies.Count == 6) break;

}

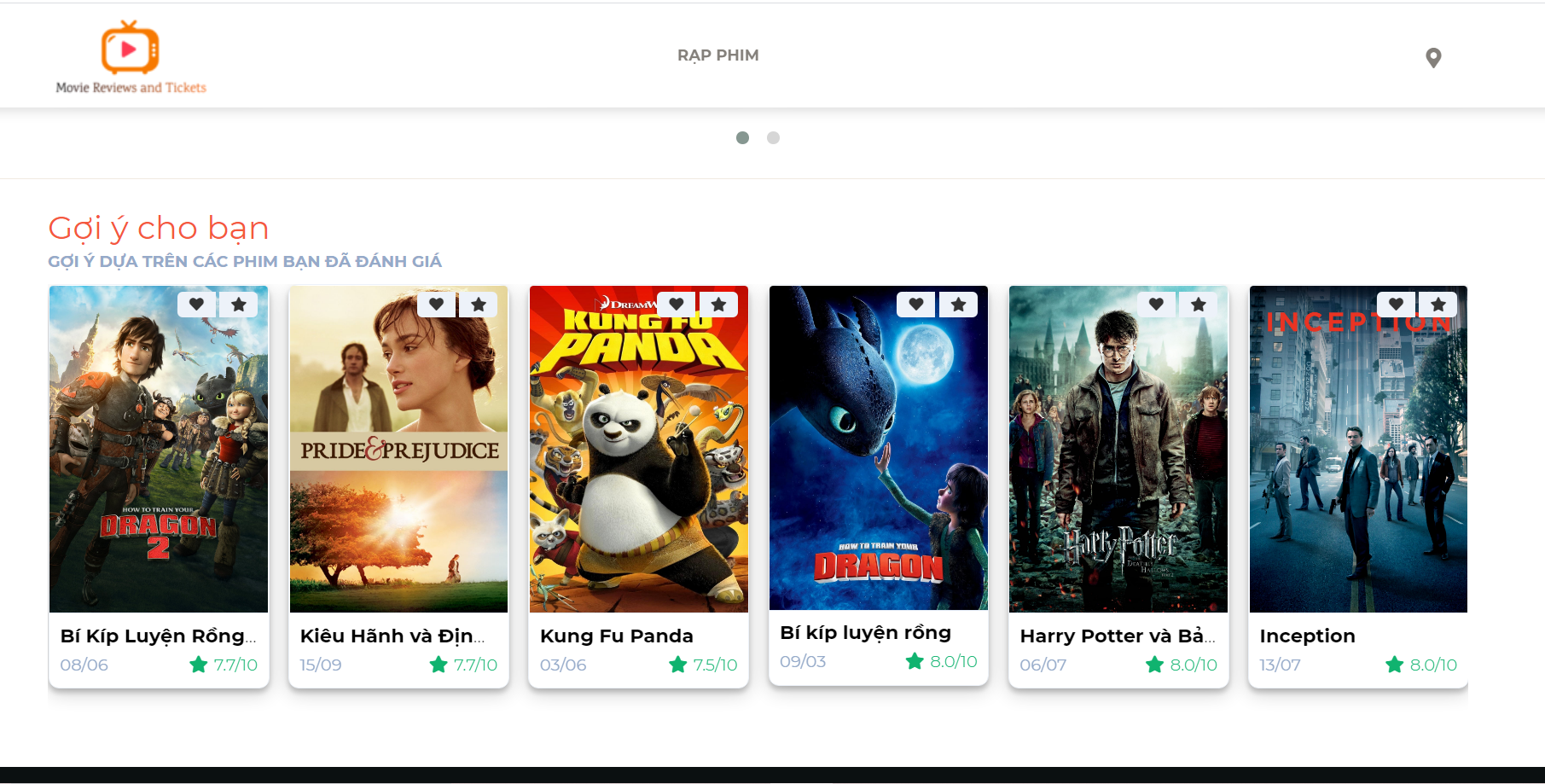
return recommendedMovies.Select(m => new MovieVM() { Movie = m, Ratings = AvgRatingsAsync(reviews.Where(r => r.MovieId == m.Id).ToList()) }).ToList();

}

Hình . API gợi ý phim cho user

### Kết quả

Sau khi tạo API, chúng ta chỉ việc để FrontEnd gọi API Recommend đi kèm với id của user đang đăng nhập, model sẽ đưa ra dự đoán những phim mà user có thể sẽ yêu thích (phim mà user có khả năng đánh giá cao hơn 7 điểm).



Hình . Sử dụng API gợi ý phim

# Kết luận

## Kết quả

* Hiểu được như thế nào là một Recommendation System, cũng như các cách tiếp cận của nó.
* Hiểu được cơ bản cách thức hoạt động của Matrix Factorization, làm sao model có thể đưa ra dự đoán đánh giá của user lên item cũng như input và output của thuật toán.
* Vận dụng kiến thức về Matrix Factorization, huấn luyện một model gợi ý phim dựa trên ratings của user bằng ML. Net và ASP .Net Core API. Source code: <https://github.com/TonNuNhuQuynh/Project3_17110134_17110184_17110214.git>

## Ưu điểm

* Matrix Factorization dựa trên nguyên tắc Collaborative Filtering của Recommendation System, nên nó dựa trên hành vi thực tế của người dùng, tức là thực tế khách quan. Đây là lợi thế lớn nhất - quan sát mọi người trong môi trường tự nhiên của họ và đưa ra dự đoán trực tiếp trên kết quả.
* Matrix Factorization giúp tiết kiệm thời gian tính toán trong khi đào tạo, ngay cả trên một tập dữ liệu nhiều chiều.
* Matrix Factorization giúp giảm độ phức tạp của dữ liệu. Trong khi các kỹ thuật chuyển đổi khác nhau trên tập dữ liệu nhiều chiều có thể dẫn đến các tập dữ liệu trở nên lớn và phức tạp hơn thì MF lại phân rã ma trận ban đầu thành các ma trận nhỏ hơn với các latent features.
* Trong các ứng dụng thế giới thực, users thường chỉ đánh giá một phần nhỏ của tất cả các bộ phim trong hệ thống. Matrix Factorization đã được chứng minh là có thể đưa ra dự đoán rất tốt ngay cả trên các ma trận rất thưa thớt.
* Dễ dàng huấn luyện và tích hợp vào ứng dụng để đưa ra gợi ý cho người dùng.

## Nhược điểm

* Vấn đề về item mới: khi có item mới được thêm vào, chưa có user nào đánh giá thì hệ thống sẽ không gợi ý item đó được.
* Hầu hết các thuật toán thuộc nhóm Collaborative Filtering đều gặp vấn đề về “Cold – Start”: model không thể học tiếp được dữ liệu mới, mà phải học lại từ đầu. Cho nên, khi có đánh giá mới từ user, model cần phải được huấn luyện lại.
* Phần cài đặt thuật toán chỉ mới sử dụng item id, user id và rating để dự đoán chứ chưa có các features khác như: thể loại phim, tuổi user, diễn viên ... để đưa ra dự đoán chính xác hơn.

# TÀI LIỆU THAM KHẢO

[1] Nguyen Thai-Nghe (2016). Hệ thống gợi ý: Kỹ thuật và ứng dụng, <https://www.researchgate.net/publication/310059523_He_thong_goi_y_Ky_thuat_va_ung_dung>

[2] phamdinhkhanh (2019). Collaborative và Content-based Filterting, <https://phamdinhkhanh.github.io/2019/11/04/Recommendation_Compound_Part1.html>

[3] Jakob Ivarsson, Mathias Lindgren (2016). Movie recommendations using matrix factorization: <https://www.diva-portal.org/smash/get/diva2:927190/FULLTEXT01.pdf?fbclid=IwAR1DTdL0AhMXj9QZq2fJRX-ErmzSWXDDjCEOqYglU-DInXrLR1Fk-FRil2A>

[4] Hoàng Đức Quân (2018). Công nghệ Matrix Factorization cho Hệ thống gợi ý: <https://viblo.asia/p/cong-nghe-matrix-factorization-cho-he-thong-goi-y-naQZRJe0Zvx>