Recommendation systems: Principles, methods and evaluation

# Abstract

Trong hệ thống Internet ngày nay, khi mà sự lựa chọn trở nên quá nhiều, chúng cần phải được lọc, sắp xếp thứ tự ưu tiên và các thông tin có liên quan phải được cung cấp một cách hiệu quả nhằm giảm bớt sự quá tải thông tin. Điều này đã tạo nên một vấn đề tiềm ẩn cho người sử dụng Internet.

Các hệ thống kiến nghị sẽ giải quyết được vấn đề này bằng cách tìm kiếm thông qua một lượng lớn các thông tin động và cung cấp cho người dùng những nội dung và dịch vụ mang tính cá nhân hóa.

Bài báo cáo này sẽ khảo sát các đặc điểm và tiềm năng của các kỹ thuật dự đoán khác nhau trong hệ thống kiến nghị.

# Introduction

Sự bùng nổ của lượng thông tin số và lượng người dùng Internet đã tạo ra một thử thách tiềm năng về tình trạng quá tải thông tin, điều nay gây cản trở sự tức thời mà Internet mang lại. Các hệ thống truy xuất thông tin chẳng hạn như Google, DevilFinder hay Altavista đã giải quyết một phần nào đó vấn đề này, nhưng tính cá nhân hóa và tối ưu hóa vẫn còn đang rất sơ sài. Điều này đã làm tăng nhu cầu trong việc sử dụng các hệ thống kiến nghị hơn bao giờ hết.

Hệ thống kiến nghị là các hệ thống lọc thông tin để đối phó với tình trạng quá tải thông tin bằng cách lọc phần thông tin quan trọng ra khỏi lượng lớn thông tin sẵn có, dựa trên sở thích hoặc là các hành vi quan sát được từ người dùng. Hệ thống kiến nghị có khả năng dự đoán liệu một người dùng cụ thể có quan tâm tới một nội dung nào đó hay không dựa trên hồ sơ lưu trữ của người dùng đó.

Hệ thống kiến nghị mang lại lợi ích cho cả nhà cung cấp lẫn người dùng sử dụng dịch vụ. Chúng giảm thiểu chi phí giao dịch trong việc tìm kiếm và lựa chọn các mặt hang ở môi trường mua sắm trực tuyến. Hệ thống kiến nghị còn chứng minh được khả năng cải thiện trong quá trình đưa ra quyết định. Trong các thư viện khoa học, hệ thống kiến nghị còn hỗ trợ người dùng bằng cách cho phép họ bằng những tìm kiếm vượt ra ngoài cả những danh mục thường dùng.

# Related work

Hệ thống kiến nghị được định nghĩa như một chiến thuật đưa ra quyết định cho người dùng dưới một môi trường thông tin phức tạp. Hệ thống kiến nghị được định nghĩa từ góc nhìn của thương mại điện tử như một công cụ giúp người dùng tìm kiếm thông qua bộ cơ sở dữ liệu liên quan tới sở thích và thói quen tiêu thụ của chính họ. Nó còn là một phương tiện để giúp đỡ và tăng cường tính tương tác của các mạng xạ hội bằng cách đồng thời sử dụng các khuyến nghị của người khác để đưa ra lựa chọn cuối cùng. Hệ thống kiến nghị xử lý các vấn đề quá tải về thông tin mà người dùng thường gặp phải bằng cách cung cấp cho họ những nội dung và dịch vụ mang tính cá nhân hóa.

Gần đây, các phương pháp tiếp cận khác nhau cho hệ thống khuyễn nghĩ đã được phát triển: lọc cộng tác, lọc sửa trên nội dung hoặc là lọc hỗn hợp

# Phase of recommendation process

## Information collection phase

Đây là bước để thu thập các thông tin liên quan của người dùng để tạo ra các hồ sơ hay mô hình của họ phục vụ cho các nhiệm vụ dự đoán. Bao gồm đặc tính, hành vi hoặc là nội dung của các nguồn tài nguyên mà người dùng đã và thường truy cập. Một tác tử khuyến nghĩ sẽ không thể hoạt động chính xác cho đến khi hồ sơ/mô hình về người dùng được xây dựng. Hệ thống cần phải được thu thập thông tin càng nhiều càng tốt từ người dùng và từ đó cung cấp những khuyến nghị có cơ sở từ trên nền tảng thông tin đó. Hệ thống khuyến nghị dựa vào các loại thông tin đầu vào khác nhau chẳng hạn như những phản hồi mang tính hài lòng rõ ràng từ người dùng, hoặc các loại thông tin ẩn dựa vào sự suy diễn thông qua quá trình quan sát người dùng. Các phản hồi hỗn hợp cũng có thể thu được thông qua sự kết hợp của cả 2 dạng phản hồi trên. Trong nền tảng E-learning, một hồ sơ người dùng là một tập hợp của các thông tin mang tính cá nhân của người đó, chẳng hạn như kĩ năng nhận thức, khả năng trí tuệ, phong cách học, sự thích thú và sự tương tác với hệ thống. Hồ sơ này sẽ được dùng để truy xuất các thông tin cần thiết để xây dựng mô hình của người dùng. Sự thành công của bất cứ một hệ thống khyến nghị nào cũng phụ thuật rất lớn vào khả năng thể hiện mô hình người dùng của chính nó.

## Explicit feedback

Hệ thống sẽ thường xuyên nhắc nhở thông qua giao diện người dùng để nhận đánh giá về sản phẩm của họ với mục đích cải thiện mô hình. Sự chính xác của sự kiến nghị phụ thuộc vào số lượng đánh giá của người dùng. Vấn đề còn lại là phương pháp này đòi hỏi sự hợp tác từ người dùng, và họ cũng thường không cung cấp đủ các thông tin cần thiết. Mặc dù điều này cần sự nỗ lực lớn từ người dùng, nhưng đó là một nguồn dữ liệu đáng tin cậy để từ đó đưa ra các khuyến nghị xác đáng.

## Implicit feedback

Hệ thống sẽ tự động lấy các số liệu của người dùng bằng cách giám sát nhiều hoạt động khác nhau của họ, chẳng hặn như lịch sử mua sắp, lịch sử chuyển hướng, và thời gian mà họ dành cho một trang web, những đường dẫn hay email có liên quan. Phương pháp này giảm thiếu gánh nặng cho người dùng bởi vì nó hoạt động độc lập dựa theo hành vi của họ. Bên cạnh đó, còn nhiều sự tranh cãi về dữ liệu mà phương pháp này mang lại. Một số ý kiến cho rằng nó chính xác khách quan hơn bởi vì nó không bị ảnh hưởng dựa theo phong trào về mặt xã hội và một số cách thức để kiếm rating khác.

## Hybrid feedback

Sức mạnh của cả EF và IF có thể tổng hợp lại thành hệ thống hỗn hợp để từ đó giảm thiểu những thiếu sót và có được một hệ thống với hiệu năng tốt nhất. Điều này có thể đạt được bằng cách sử dụng dữ liệu IF để kiểm tra thông số EF hoặc chỉ cho phép người dùng cung cấp phản hồi EF một khi họ đã thể hiện sự quan tâm đó thông qua IF.

## Learning phase

Trong bước này chúng ta sẽ áp dụng thuật toán học để lọc và khai thác các thuộc tính của người dùng có được từ bước trước đó.

## Prediction/recommendation phase

Bước này sẽ khuyến nghị hoặc dự đoán những loại hàng mà người dùng có thể thích thú hơn. Điều này có thể được thực hiện trực tiếp dựa trên bộ dữ liệu được thu thập trong giai đoạn thu thập thông tin.



# Recommendation filtering techniques

Việc sử dụng các kỹ thuật kiến nghị hiệu quả và chính xác là rất quan trọng đối với một hệ thống, nó sẽ cung cấp các kiến nghị tốt và tối ưu cho chính những người dùng đó. Điều này giải thích tầm quan trọng của việc hiểu các tính năng và tiềm năng của các kỹ thuật đề xuất khác nhau. Hình dưới cho thấy sự khác nhau cơ bản của các kỹ thuật lọc đề xuất.



## Collaborative recommendation systems

* Hệ thống gợi ý dựa theo lọc cộng tác (Collaborative recommendation systems): là phương pháp gợi ý được triển khai rộng rãi nhất và thành công nhất trong thực tế.
* Hệ thống theo lọc công tác phân tích và tổng hợp các điểm số đánh giá của các đối tượng, nhận ra sự tương đồng giữa những người sử dụng trên cơ sở các điểm số đánh giá của họ và tạo ra các gợi ý dựa trên sự so sánh này. Hồ sơ (profile) của người sử dụng điển hình trong hệ thống lọc cộng tác bao gồm một vector các đối tượng (item) và các điểm số đánh giá của chúng, với số chiều tăng lên liên tục khi người sử dụng tương tác với hệ thống theo thời gian.
* Một số hệ thống sử dụng phương pháp chiết khấu dựa trên thời gian (time-based discounting) để tính toán cho yếu tố “trượt” đối với sự quan tâm của người sử dụng. Trong một số trường hợp điểm số đánh giá (rating) có thể là nhị phân (thích/không thích) hoặc các giá trị số thực cho thấy mức độ ưu tiên.
* Thế mạnh lớn nhất của kỹ thuật gợi ý theo lọc cộng tác là chúng hoàn toàn độc lập với sự biểu diễn của các đối tượng đang được gợi ý, và do đó có thể làm việc tốt với các đối tượng phức tạp như âm thanh và phim. Schafer, Konstan & Riedl (1999) gọi lọc cộng tác là “tương quan giữa người – với – người” (people-to-people correlation).

## Content-based recommendation systems

* Hệ thống gợi ý dựa theo nội dung (Content-based recommendation systems): là sự kế thừa và mở rộng của lĩnh vực nghiên cứu lọc thông tin.
* Trong hệ thống thì các đối tượng được biểu diễn bởi các đặc điểm liên quan tới chúng.
* Ví dụ, hệ thống gợi ý văn bản như hệ thống lọc tin NewsWeeder sử dụng những từ của các văn bản như các đặc điểm.
* Một số hệ thống gợi ý dựa trên nội dung học một hồ sơ cá nhân về sở thích của người sử dụng dựa trên các đặc điểm xuất hiện trong chính các đối tượng người sử dụng đã đánh giá (rated). Schafer, Konstan & Riedl gọi gợi ý theo nội dung là “tương quan đối tượng với đối tượng” (item-to-item correlation). Hồ sơ người sử dụng của một hệ thống gợi ý theo nội dung phụ thuộc vào phương pháp học máy được dùng.
* Cây quyết định (Decision trees), mạng noron (neural nets) và biểu diễn dựa theo vector (vector-based representations) đều có thể được sử dụng để học hồ sơ người dùng. Cũng giống như trong lọc cộng tác, hồ sơ người dùng trong gợi ý dựa theo nội dung là những dữ liệu lâu dài và được cập nhật theo thời gian.

## Knowledge-based recommenders systems

* Hệ thống gợi ý dựa trên cơ sở tri thức (Knowledge-based recommenders systems): gợi ý các đối tượng dựa trên các suy luận về nhu cầu và sở thích của người dùng. Theo một nghĩa nào đó, tất cả các kỹ thuật gợi ý có thể mô tả như là làm một số suy luận. Phương pháp tiếp cận dựa trên cơ sở tri thức được phân biệt ở chỗ: chúng có kiến thức làm thế nào một đối tượng cụ thể đáp ứng nhu cầu một người dùng cụ thể, và do đó có thể lập luận về mối quan hệ giữa nhu cầu và các gợi ý cụ thể.
* Sử dụng miền tri thức rõ ràng, có liên quan tới mối quan hệ giữa yêu cầu của người dùng và sản phẩm cụ thể. Ban đầu người ta đưa ra 3 dạng tri thức: tri thức về danh mục (tri thức về sản phẩm được gợi ý), tri thức người sử dụng (tri thức về các yêu cầu của người sử dụng), tri thức về các chức năng (tri thức để ánh xạ các yêu cầu của người sử dụng tới các sản phẩm thoả mãn các yêu cầu đó).
* Phương pháp này không dựa trên tiểu sử người sử dụng nên không gặp phải khó khăn về sản phẩm mới và người dùng mới. Gợi ý trên cơ sở tri thức có khả năng suy diễn, khả năng suy diễn phụ thuộc vào độ phù hợp của yêu cầu người sử dụng với các thuộc tính của sản phẩm.
* Mọi hệ thống dựa trên cơ sở tri thức đều là mối quan hệ thu nhận tri thức. Thực tế, chất lượng của các phương án gợi ý tùy thuộc vào độ chính xác của cơ sở tri thức. Đây cũng là hạn chế lớn nhất của phương pháp này.

# Comparing

|  |  |  |
| --- | --- | --- |
|  | Adv | DisAdv |
| Collaborative | * Khả năng đa hạng mục * Không cần tri thức miền * Chất lượng tăng theo thời gian * Đủ thông tin phản hồi không tường minh | * Vấn đề người dùng mới * Vấn đề sản phẩm/đối tượng mới * Vấn đề “Gray sheep” * Chất lượng phụ thuộc vào độ lớn dữ liệu lịch sử thao tác của người sử dụng * Vần đề về tính bền vững và mềm dẻo |
| Content-based | * Không cần tri thức miền * Chất lượng tăng theo thời gian * Đủ thông tin phản hồi không tường minh | * Vấn đề người dùng mới * Chất lượng phụ thuộc vào độ lớn dữ liệu lịch sử thao tác của người sử dụng * Vần đề về tính bền vững và mềm dẻo |
| Knowledge-based | * Có thể ánh xạ giữa nhu cầu người dùng và sản phẩm/đối tượng | * Cần phải thu thập tri thức. |

# Conclusion

Các hệ thống kiến nghị ngày nay đã mở ra nhiều cơ hội mới để thu thập thông tin cá nhân trên Internet. Nó cũng giúp giảm bớt vấn đề về quá tải thông tin – một hiện tượng rất phổ biến với các hệ thống truy xuất thông tin. Trên đây đã giới thiệu một cái nhìn sơ lược về nội dung cũng như các kỹ thuật về hệ thống kiến nghị.

# Neighborhood-Based Collaborative Filtering

Trong phần này, chúng ta sẽ tìm hiểu kĩ hơn về hệ thống gợi ý dựa theo lọc cộng tác dưới phương pháp NBCF. Ý tưởng cơ bản của NBCF là xác định *mức độ quan tâm* của một *user* tới một *item* dựa trên các *users*khác *gần giống* với *user* này. Việc *gần giống nhau* giữa các *users* có thể được xác định thông qua *mức độ quan tâm* của các *users* này tới các *items* khác mà hệ thống đã biết. Ví dụ, *A, B* đều thích phim *Cảnh sát hình sự*, tức đều *rate* bộ phim này 5 sao. Ta đã biết *A* cũng thích *Người phán xử*, vậy nhiều khả năng *B* cũng thích bộ phim này.

Như vậy hai câu hỏi quan trọng nhất trong một hệ thống Neighborhood-based Collaborative Filtering là:

* Làm thế nào xác định được *sự giống nhau* giữa hai *users*?
* Khi đã xác định được các *users* *gần giống nhau* (*similar users*) rồi, làm thế nào dự đoán được *mức độ quan tâm* của một *user* lên một *item*?

Việc xác định mức độ quan tâm của mỗi *user* tới một *item* dựa trên mức độ quan tâm của *similar users*tới *item* đó còn được gọi là *User-user collaborative filtering*. Có một hướng tiếp cận khác được cho là làm việc hiệu quả hơn là *Item-item collaborative filtering*. Trong hướng tiếp cận này, thay vì xác định *user similarities*, hệ thống sẽ xác định *item similarities*. Từ đó, hệ thống gợi ý những *items* *gần giống với*những *items* mà user có mức độ quan tâm cao.

## **User-user Collaborative Filtering**

Công việc quan trọng nhất phải làm trước tiên trong User-user Collaborative Filtering là phải xác định được sự giống nhau (similarity) giữa hai users. Dữ liệu duy nhất chúng ta có là Utility matrix YY, vậy nên sự giống nhau này phải được xác định dựa trên các cột tương ứng với hai users trong ma trận này. Xét ví dụ trong Hình 1.

|  |  |
| --- | --- |
| https://machinelearningcoban.com/assets/24_collaborativefiltering/utility.png | Hình 1: Ví dụ về utility matrix dựa trên số sao một user rate cho một item. Một cách trực quan, hành vi của u0 giống với u1 hơn là u2, u3, u4, u5, u5, u6. Từ đó có thể dự đoán rằng u0 sẽ quan tâm tới i2 vì u1 cũng quan tâm tới item này. |

Giả sử có các users từ u0 đến u6 và các items từ i0 đến i4 trong đó các số trong mỗi ô vuông thể hiện số sao mà mỗi user đã rated cho item với giá trị cao hơn thể hiện mức độ quan tâm cao hơn. Các dấu hỏi chấm là các giá trị mà hệ thống cần phải đi tìm. Đặt mức độ giống nhau của hai users ui, uj là sim(ui,uj)

Quan sát đầu tiên chúng ta có thể nhận thấy là các u0, u1 thích i0, i1, i2 và không thích i3, i4 cho lắm. Điều ngược lại xảy ra ở các users còn lại. Vì vậy, một similiarity function tốt cần đảm bảo: sim(u0,u1) > sim(u0,ui), ∀i>1.

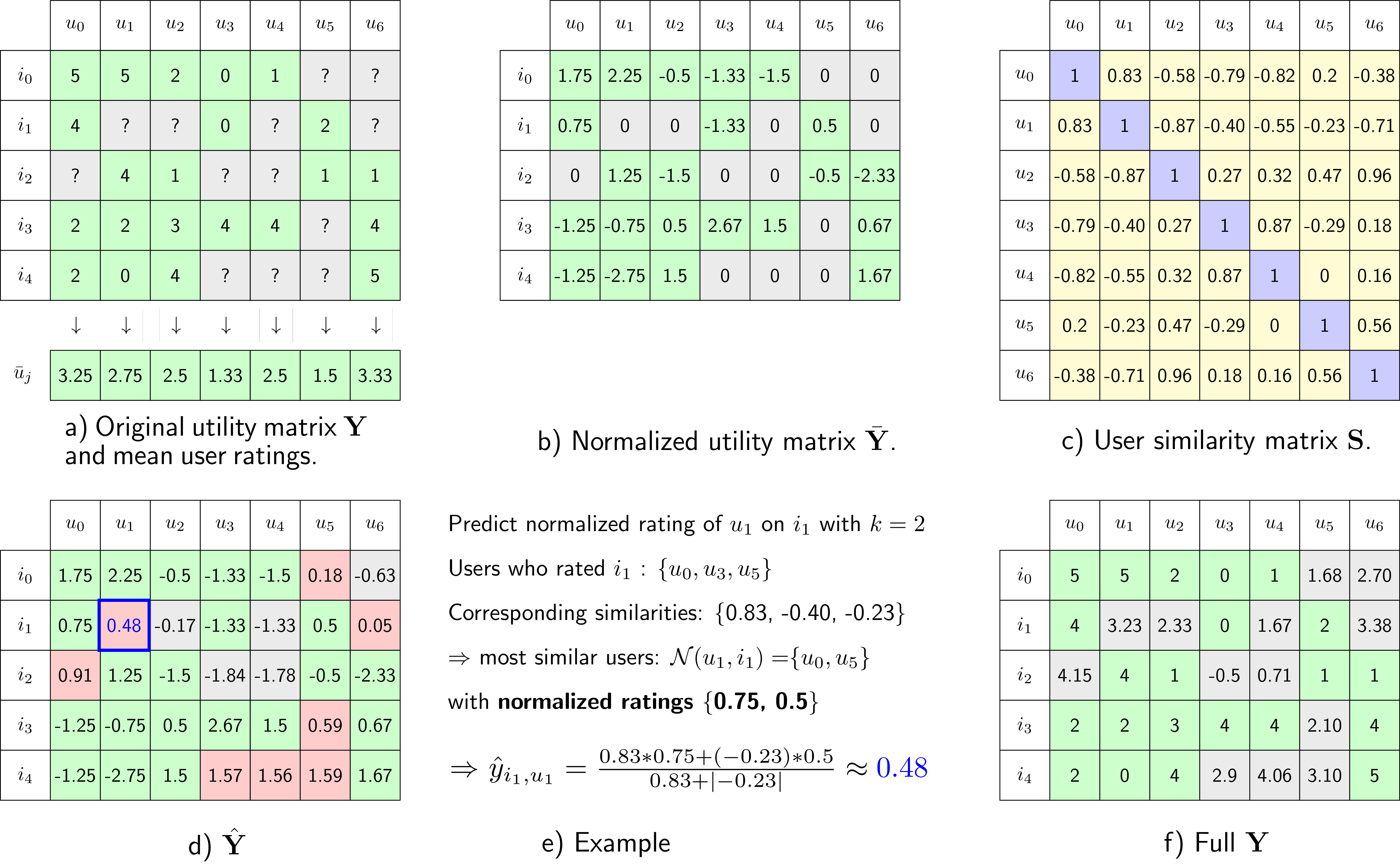
Từ đó, để xác định mức độ quan tâm của u0 lên i2, chúng ta nên dựa trên hành vi của u1 lên sản phẩm này. Rất may rằng u1 đã thích i2 nên hệ thống cần recommend i2 cho u0.

Câu hỏi đặt ra là: hàm số similarity nào là tốt? Để đo similarity giữa hai users, cách thường làm là xây dựng feature vector cho mỗi user rồi áp dụng một hàm có khả năng đo similarity giữa hai vectors đó. Chú ý rằng việc xây dựng feature vector này khác với việc xây dựng [item profiles](https://machinelearningcoban.com/2017/05/24/collaborativefiltering/2017/05/17/contentbasedrecommendersys/#-item-profiles) như trong Content-based Recommendation Systems. Các vectors này được xây dựng trực tiếp dựa trên Utility matrix chứ không dùng dữ liệu ngoài như item profiles. Với mỗi user, thông tin duy nhất chúng ta biết là các ratings mà userđó đã thực hiện, tức cột tương ứng với user đó trong Utility matrix. Tuy nhiên, khó khăn là các cột này thường có rất nhiều mising ratings vì mỗi user thường chỉ rated một số lượng rất nhỏ các items. Cách khắc phục là bằng cách nào đó, ta giúp hệ thống điền các giá trị này sao cho việc điền không làm ảnh hưởng nhiều tới sự giống nhau giữa hai vector. Việc điền này chỉ phục vụ cho việc tính similarity chứ không phải là suy luận ra giá trị cuối cùng.

Vậy mỗi dấu ‘?’ nên được thay bởi giá trị nào để hạn chế việc sai lệch quá nhiều? Một lựa chọn bạn có thể nghĩ tới là thay các dấu ‘?’ bằng giá trị ‘0’. Điều này không thực sự tốt vì giá trị ‘0’ tương ứng với mức độ quan tâm thấp nhất. Một giá trị an toàn hơn là 2.5 vì nó là trung bình cộng của 0, mức thấp nhất, và 5, mức cao nhất. Tuy nhiên, giá trị này có hạn chế đối với những users dễ tính hoặc khó tính. Với các usersdễ tính, thích tương ứng với 5 sao, không thích có thể ít sao hơn 1 chút, 3 sao chẳng hạn. Việc chọn giá trị 2.5 sẽ khiến cho các items còn lại là quá negative đối với user đó. Điều ngược lại xảy ra với những user khó tính hơn khi chỉ cho 3 sao cho các items họ thích và ít sao hơn cho những items họ không thích.

Một giá trị khả dĩ hơn cho việc này là trung bình cộng của các ratings mà user tương ứng đã thực hiện. Việc này sẽ tránh được việc users quá khó tính hoặc dễ tính, tức lúc nào cũng có những items mà một user thích hơn so với những items khác.

Hãy cùng xem ví dụ trong Hình 2a) và 2b).



Hình 2: Ví dụ mô tả User-user Collaborative Filtering. a) Utility Matrix ban đầu. b) Utility Matrix đã được chuẩn hoá. c) User similarity matrix. d) Dự đoán các (normalized) ratings còn thiếu. e) Ví dụ về cách dự đoán normalized rating của u1u1cho i1i1. f) Dự đoán các (denormalized) ratings còn thiếu.

Hàng cuối cùng trong Hình 2a) là giá trị trung bình của ratings cho mỗi user. Giá trị cao tương ứng với các user dễ tính và ngược lại. Khi đó, nếu tiếp tục trừ từ mỗi rating đi giá trị này và thay các giá trị chưa biết bằng 0, ta sẽ được normalized utility matrix như trong Hình 2b). Bạn có thể thắc mắc tại sao bước chuẩn hoá này lại quan trọng, câu trả lời ở ngay đây:

* Việc trừ đi trung bình cộng của mỗi cột khiến trong trong mỗi cột có những giá trị dương và âm. Những giá trị dương tương ứng với việc user thích item, những giá trị âm tương ứng với việc user không thích item. Những giá trị bằng 0 tương ứng với việc chưa xác định được liệu user có thích item hay không.
* Về mặt kỹ thuật, số chiều của utility matrix là rất lớn với hàng triệu users và items, nếu lưu toàn bộ các giá trị này trong một ma trận thì khả năng cao là sẽ không đủ bộ nhớ. Quan sát thấy rằng vì số lượng ratings biết trước thường là một số rất nhỏ so với kích thước của utility matrix, sẽ tốt hơn nếu chúng ta lưu ma trận này dưới dạng sparse matrix, tức chỉ lưu các giá trị khác không và vị trí của chúng. Vì vậy, tốt hơn hết, các dấu ‘?’ nên được thay bằng giá trị ‘0’, tức chưa xác định liệu user có thích item hay không. Việc này không những tối ưu bộ nhớ mà việc tính toán similarity matrix sau này cũng hiệu quả hơn.

Sau khi đã chuẩn hoá dữ liệu như trên, một vài similiraty function thường được sử dụng là:

**Cosine Similarity:**

Đây là hàm được sử dụng nhiều nhất, và cũng quen thuộc với các bạn nhất. Nếu các bạn không nhớ công thức tính coscos của góc giữa hai vector u1, u2 trong chương trình phổ thông, thì dưới đây là công thức:

Trong đó u1, u2 là vectors tương ứng với users 1, 2 **đã được chuẩn hoá** như ở trên.

Có một tin vui là python có hàm hỗ trợ tính toán hàm số này một cách hiệu quả.

Độ similarity của hai vector là 1 số trong đoạn [-1, 1]. Giá trị bằng 1 thể hiện hai vector hoàn toàn similarnhau. Hàm số cos của một góc bằng 1 nghĩa là góc giữa hai vector bằng 0, tức một vector bằng tích của một số dương với vector còn lại. Giá trị cos bằng -1 thể hiện hai vector này hoàn toàn trái ngược nhau. Điều này cũng hợp lý, tức khi hành vi của hai users là hoàn toàn ngược nhau thi similarity giữa hai vector đó là thấp nhất.

Ví dụ về cosine\_similarity của các users trong Hình 2b) được cho trong Hình 2c). Similarity matrix SS là một ma trận đối xứng vì cos là một hàm chẵn, và nếu user A giống user B thì điều ngược lại cũng đúng. Các ô màu xanh trên đường chéo đều bằng 1 vì đó là cos của góc giữa 1 vector và chính nó, tức cos(0)=1. Khi tính toán ở các bước sau, chúng ta không cần quan tâm tới các giá trị 1 này. Tiếp tục quan sát các vector hàng tương ứng với u0,u1,u2u0,u1,u2, chúng ta sẽ thấy một vài điều thú vị:

* u0 gần với u1 và u5 (độ giống nhau là dương) hơn các users còn lại. Việc similarity cao giữa u0 và u1 là dễ hiểu vì cả hai đều có xu hướng quan tâm tới i0, i1, i2 hơn các items còn lại. Việc u0 gần với u5 thoạt đầu có vẻ vô lý vì u5 đánh giá thấp các items mà u0 đánh giá cao (Hình 2a)); tuy nhiên khi nhìn vào ma trận utility đã chuẩn hoá ở Hình 2b), ta thấy rằng điều này là hợp lý. Vì item duy nhất mà cả hai users này đã cung cấp thông tin là i1 với các giá trị tương ứng đều là tích cực.
* u1 gần với u0 và xa các users còn lại.
* u2 gần với u3, u4, u5, u6 và xa các users còn lại.

Từ similarity matrix này, chúng ta có thể phân nhóm các users ra làm hai nhóm (u0, u1) và (u2, u3, u4, u5, u6). Vì ma trận S này nhỏ nên chúng ta có thể dễ dàng quan sát thấy điều này; khi số users lớn hơn, việc xác định bằng mắt thường là không khả thi.

Có một chú ý quan trọng ở đây là khi số lượng users lớn, ma trận SS cũng rất lớn và nhiều khả năng là không có đủ bộ nhớ để lưu trữ, ngay cả khi chỉ lưu hơn một nửa số các phần tử của ma trận đối xứngnày. Với các trường hợp đó, mới mỗi user, chúng ta chỉ cần tính và lưu kết quả của một hàng của similarity matrix, tương ứng với việc độ giống nhau giữa user đó và các users còn lại.

## **Item-item Collaborative Filtering**

Một số hạn chế của User-user CF:

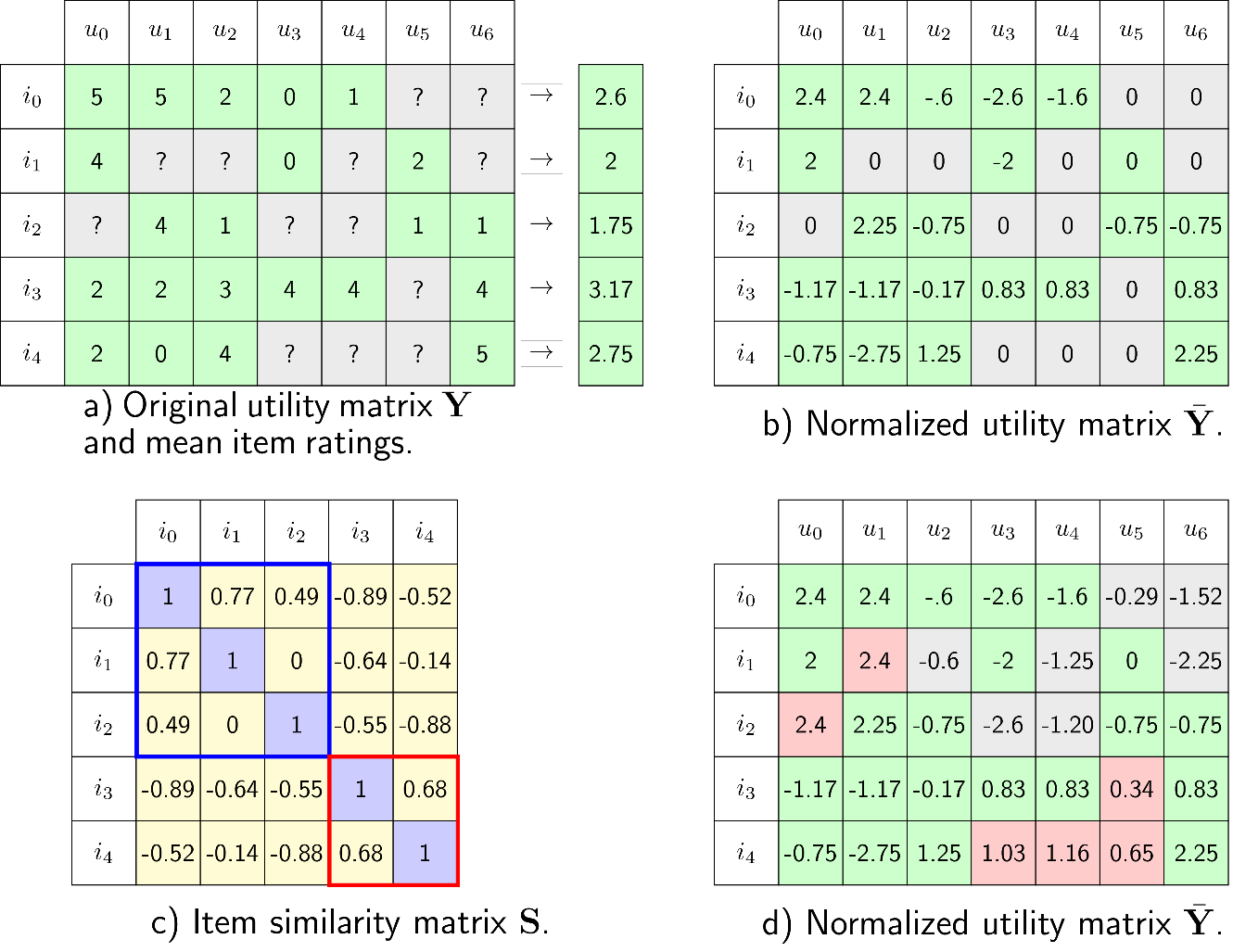
* Trên thực tế, số lượng *users* luôn lớn hơn số lượng *items* rất nhiều. Kéo theo đó là *Similarity matrix* là rất lớn với số phần tử phải lưu giữ là hơn 1 nửa của bình phương số lượng *users* (chú ý rằng ma trận này là đối xứng). Việc này, như đã đề cập ở trên, khiến cho việc lưu trữ ma trận này trong nhiều trường hợp là không khả thi.
* Ma trận Utility Y thường là rất *sparse*. Với số lượng *users* rất lớn so với số lượng *items*, rất nhiều cột của ma trận này sẽ rất *sparse*, tức chỉ có một vài phần tử khác 0. Lý do là *users* thường *lười*rating. Cũng chính vì việc này, một khi *user* đó thay đổi *rating* hoặc rate thêm *items*, trung bình cộng các *ratings* cũng như vector chuẩn hoá tương ứng với *user* này thay đổi nhiều. Kéo theo đó, việc tính toán ma trận Similarity, vốn tốn nhiều bộ nhớ và thời gian, cũng cần được thực hiện lại.

Ngược lại, nếu chúng ta tính toán *similarity* giữa các *items* rồi *recommend* những *items* gần *giống* với *item* yêu thích của một *user* thì sẽ có những lợi ích sau:

* Vì số lượng *items* thường nhỏ hơn số lượng *users*, Similarity matrix trong trường hợp này cũng nhỏ hơn nhiều, thuận lợi cho việc lưu trữ và tính toán ở các bước sau.
* Vì số lượng phần tử đã biết trong Utility matrix là như nhau nhưng số hàng (*items*) ít hơn số cột (*users*), nên trung bình, mỗi hàng của ma trận này sẽ có nhiều phần tử đã biết hơn số phần tử đã biết trong mỗi cột. Việc này cũng dễ hiểu vì mỗi *item* có thể được *rated* bởi nhiều *users*. Kéo theo đó, giá trị trung bình của mỗi hàng ít bị thay đổi hơn khi có thêm một vài *ratings*. Như vậy, việc cập nhật ma trận Similarity Matrix có thể được thực hiện ít thường xuyên hơn.

Cách tiếp cận thứ hai này được gọi là *Item-item Collaborative Filtering*. Hướng tiếp cận này được sử dụng nhiều trong thực tế hơn.

Quy trình dự đoán *missing ratings* cũng tương tự như trong User-user CF. Hình 3 mô tả quy trình này với ví dụ nêu ở phần trên.



Hình 3: Ví dụ mô tả Item-Item Collaborative Filtering. a) Utility Matrix ban đầu. b) Utility Matrix đã được chuẩn hoá. c) User similarity matrix. d) Dự đoán các (normalized) *ratings* còn thiếu.

Có một điểm thú vị trong Similarity matrix ở Hình 3c) là có các phần tử trong hai hình vuông xanh và đỏ đều là các số không âm, các phần tử bên ngoài là các số âm. Việc này thể hiện rằng các *items* có thể được chia thành 2 nhóm rõ rệt với những *items* có *similarity* không âm vào 1 nhóm. Như vậy, một cách *vô tình*, chúng ta đã thực hiện việc *item clustering*. Việc này sẽ giúp ích rất nhiều trong việc dự đoán ở phần sau.

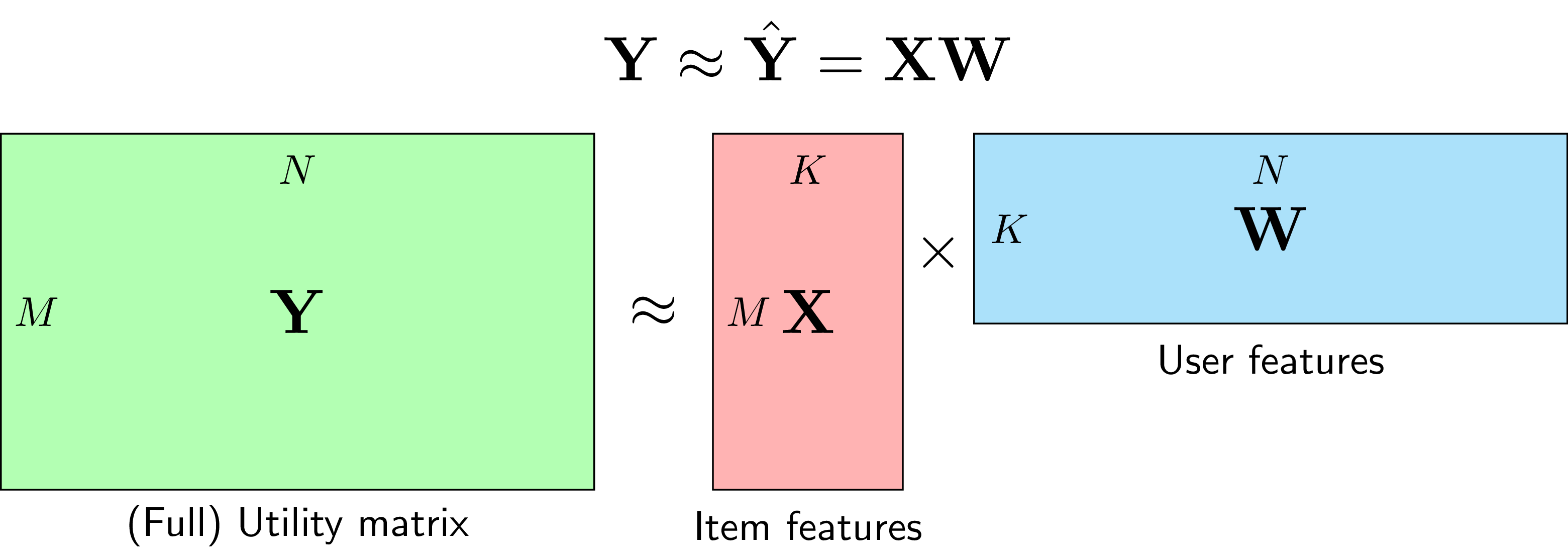
Kết quả về việc chọn *items* nào để *recommend* cho mỗi *user* được thể hiện bởi các ô màu đỏ trong Hình 3d). Kết quả này có khác một chút so với kết quả tìm được bởi User-user CF ở 2 cột cuối cùng tương ứng với u5,u6u5,u6. Dường như kết quả này *hợp lý* hơn vì từ Utility Matrix, có hai nhóm *users* thích hai nhóm *items* khác nhau.

**Về mặt tính toán, Item-item CF có thể nhận được từ User-user CF bằng cách chuyển vị (transpose) ma trận utility, và coi như *items* đang *rate* *users*. Sau khi tính ra kết quả cuối cùng, ta lại chuyển vị một lần nữa để thu được kết quả.**

# Matrix Factorization Collaborative Filtering

## Introduction

Chúng ta đã làm quen với một hướng tiếp cận trong Collaborative Filtering dựa trên hành vi của các users hoặc items lân cận có tên là Neighborhood-based Collaborative Filtering. Trong phần này, chúng ta sẽ làm quen với một hướng tiếp cận khác cho Collaborative Filtering dựa trên Matrix Factorization (hoặc Matrix Decomposition), tức Phân tích ma trận thành nhân tử.



Hình 1: Matrix Factorization. Utility matrix Y được phân tích thành tích của hai ma trận low-rank X và W

Có một vài điểm lưu ý ở đây:

* Ý tưởng chính đằng sau Matrix Factorization cho Recommendation Systems là tồn tại các latent features (tính chất ẩn) mô tả sự liên quan giữa các items và users. Ví dụ với hệ thống gợi ý các bộ phim, tính chất ẩn có thể là 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. 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.
* Vậy tại sao Matrix Factorization lại được xếp vào Collaborative Filtering? Câu trả lời đến từ việc đi tối ưu hàm mất mát mà chúng ta sẽ thảo luận ở Mục 2. Về cơ bản, để tìm nghiệm của bài toán tối ưu, ta phải lần lượt đi tìm X và W khi thành phần còn lại được cố định. Như vậy, mỗi hàng của XXsẽ phụ thuộc vào toàn bộ các cột của W. Ngược lại, mỗi cột của W lại phục thuộc vào toàn bộ các hàng của X. Như vậy, có những mỗi quan hệ ràng buộc chằng chịt giữa các thành phần của hai ma trận trên. Tức chúng ta cần sử dụng thông tin của tất cả để suy ra tất cả. Vậy nên phương pháp này cũng được xếp vào Collaborative Filtering.
* Trong các bài toán thực tế, số lượng items M và số lượng users N thường rất lớn. Việc tìm ra các mô hình đơn giản giúp dự đoán ratings cần được thực hiện một cách nhanh nhất có thể. Neighborhood-based Collaborative Filtering không yêu cầu việc learning quá nhiều, nhưng trong quá trình dự đoán (inference), ta cần đi tìm độ similarity của user đang xét với toàn bộ các userscòn lại rồi suy ra kết quả. Ngược lại, với Matrix Factorization, việc learning có thể hơi phức tạp một chút vì phải lặp đi lặp lại việc tối ưu một ma trận khi cố định ma trận còn lại, nhưng việc inferenceđơn giản hơn vì ta chỉ cần lấy tích của hai vector xw, mỗi vector có độ dài K là một số nhỏ hơn nhiều so với M, N. Vậy nên quá trình inference không yêu cầu khả năng tính toán cao. Việc này khiến nó phù hợp với các mô hình có tập dữ liệu lớn.
* Thêm nữa, việc lưu trữ hai ma trận X và W yêu cầu lượng bộ nhớ nhỏ khi so với việc lưu toàn bộ Similarity matrix trong Neighborhood-based Collaborative Filtering. Cụ thể, ta cần bộ nhớ để chứa K(M+N) phần tử thay vì lưu M2 hoặc N2 của Similarity matrix.

## Lost function

Việc xây dựng hàm mất mát cũng được dựa trên các thành phần đã được điền của Utility Matrix Y, có khác một chút là không có thành phần bias và biến tối ưu là cả X và W. Việc xây dựng hàm mất mát cho Matrix Factorization là tương đối dễ hiểu:

trong đó rmn=1 nếu item thứ mm đã được đánh giá bởi user thứ n, là Frobineous norm, tức căn bậc hai của tổng bình phương tất cả các phần tử của ma trận (giống với norm 2 trong vector), 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 trên là l2 regularization, giúp tránh overfitting.

Khi cố định XX, việc tối ưu WW chính là bài toán tối ưu trong Content-based Recommendation Systems. Bài toán này sẽ được tối ưu bằng Gradient Descent.

# References

* + <https://viblo.asia/p/gioi-thieu-ve-he-thong-goi-y-recommender-systems-hoac-recommendation-systems-maGK78yOZj2>
  + <http://www.sciencedirect.com/science/article/pii/S1110866515000341>
  + <http://infolab.stanford.edu/~ullman/mmds/ch9.pdf>
  + <https://www.coursera.org/learn/machine-learning/supplement/gXdW5/lecture-slides>
  + <https://www.coursera.org/learn/machine-learning/supplement/gXdW5/lecture-slides>

# Next steps

Tiến hành xây dựng và training thử một chatbot - ứng dụng của Machine Learning.