Recommendation Systems

I. Giới thiệu: [1]

Có một lớp rộng rãi các ứng dụng Web liên quan đến việc dự đoán phản ứng của người dùng đối với các tùy chọn. Cơ sở như vậy được gọi là hệ thống khuyến nghị. Chúng ta sẽ bắt đầu chương này với một cuộc khảo sát về các ví dụ quan trọng nhất của các hệ thống này. Tuy nhiên, để đưa vấn đề vào trọng tâm, hai ví dụ điển hình về hệ thống khuyến nghị là:

1. Cung cấp các tin bài cho độc giả báo trực tuyến, dựa trên dự đoán về sở thích của độc giả.

2. Cung cấp cho khách hàng của một nhà bán lẻ trực tuyến đề xuất về những gì họ có thể muốn mua, dựa trên lịch sử mua hàng và / hoặc tìm kiếm sản phẩm trong quá khứ của họ.

Hệ thống khuyến nghị sử dụng một số công nghệ khác nhau. Chúng ta có thể phân loại các hệ thống này thành hai nhóm lớn.

• Content-based systems: đánh giá đặc tính của *items* được *recommended*. Ví dụ: một *user* xem rất nhiều các bộ phim về cảnh sát hình sự, vậy thì gợi ý một bộ phim trong cơ sở dữ liệu có chung đặc tính *hình sự* tới *user* này, ví dụ phim *Người phán xử*. Cách tiếp cận này yêu cầu việc sắp xếp các *items* vào từng nhóm hoặc đi tìm các đặc trưng của từng *item*. Tuy nhiên, có những *items* không có nhóm cụ thể và việc xác định nhóm hoặc đặc trưng của từng *item* đôi khi là bất khả thi.

• Collaborative filtering systems: hệ thống gợi ý *items* dựa trên sự tương quan (similarity) giữa các *users* và/hoặc *items*. Có thể hiểu rằng ở nhóm này một *item* được *recommended* tới một *user* dựa trên những *users* có *hành vi* tương tự. Ví dụ: *users A, B, C* đều thích các bài hát của Noo Phước Thịnh. Ngoài ra, hệ thống biết rằng *users B, C* cũng thích các bài hát của Bích Phương nhưng chưa có thông tin về việc liệu *user A* có thích Bích Phương hay không. Dựa trên thông tin của những *users* tương tự là *B và C*, hệ thống có thể dự đoán rằng *A* cũng thích Bích Phương và gợi ý các bài hát của ca sĩ này tới *A*.

2. A Model for Recommendation Systems

2.1. The Utility Matrix: [2]

Trong một ứng dụng hệ thống khuyến nghị có hai lớp thực thể, mà chúng ta sẽ gọi là *user* và *items*. User có các tùy chọn cho các items nhất định và các tùy chọn này phải được đưa ra khỏi dữ liệu. Bản thân dữ liệu được biểu diễn dưới dạng *utility matrix*, cung cấp cho mỗi cặp user-items, một giá trị đại diện cho những gì đã biết về mức độ ưa thích của người dùng đó đối với mặt hàng đó. Giá trị đến từ một

tập hợp có thứ tự, ví dụ: số nguyên 1–5 đại diện cho số sao mà người dùng đưa ra làm xếp hạng cho mặt hàng đó. Chúng tôi giả định rằng ma trận là thưa thớt, có nghĩa là hầu hết các mục nhập là *"Unknown"*. Xếp hạng không xác định ngụ ý rằng chúng ta không có thông tin rõ ràng về sở thích của người dùng đối với mặt hàng.

2.1.1 Ví dụ về Utility Matrix:

Như đã đề cập, có hai thực thể chính trong các Recommendation Systems là *users* và *items*. Mỗi *user* sẽ có *mức độ quan tâm* (*degree of preference*) tới từng *item* khác nhau. Mức độ quan tâm này, *nếu đã biết trước*, 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*, ta tạm gọi giá trị này là *rating*. Tập hợp tất cả các *ratings*, bao gồm cả những giá trị chưa biết cần được dự đoán, tạo nên một ma trận gọi là *utility matrix*. Xét ví dụ sau:

	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	?	?	?

Figure 1. Ví dụ về utility matrix với hệ thống Gợi ý bài hát. Các bài hát được người dùng đánh giá theo mức độ từ 0 đến 5 sao. Các dấu '?' nền màu xám ứng với việc dữ liệu chưa tồn tại trong cơ sở dữ liệu. Recommendation Systems cần phải tự điền *các giá trị này.*

Trong ví dụ này, có 6 *users A, B, C, D, E, F* và 5 bài hát. Các ô màu xanh thể hiện việc một *user* đã đánh giá một bài hát với *ratings* từ 0 (không thích) đến 5 (rất thích). Các ô có dấu '?' màu xám tương ứng với các ô chưa có dữ liệu. Công việc của một Recommendation Systems là dự đoán giá trị tại các ô màu xám này, từ đó đưa ra gợi ý cho người dùng. Recommendation Systems, vì vậy, đôi khi cũng được coi là bài toán *Matrix Completion* (*Hoàn thiện ma trận*).

Trong ví dụ đơn giản này, dễ thấy có 2 thể loại nhạc khác nhau: 3 bài đầu là nhạc *Bolero* và 2 bài sau là nhạc *Thiếu nhi*. Từ dữ liệu này, ta cũng có thể đoán được rằng *A, B* thích thể loại *Bolero*; *C, D,* E, F thích thể loại *Thiếu nhi*. Từ đó, một hệ thống tốt nên gợi ý *Cỏ úa* cho *B*; *Vùng lá me bay* cho *A,*; *Em yêu trường em* cho *D, E, F*. Giả sử chỉ có hai thể loại nhạc này, khi có một bài hát mới, ta chỉ cần phân lớp nó vào thể loại nào, từ đó đưa ra gợi ý với từng người dùng.

Thông thường, có rất nhiều *users* và *items* trong hệ thống, và mỗi *user* thường chỉ *rate* một số lượng rất nhỏ các *item*, thậm chí có những *user* không rate *item* nào (với những *users* này thì cách tốt nhất là gợi ý các *items* phổ biến nhất). Vì vậy, lượng ô màu xám của utility matrix trong các bài toán đó thường là rất lớn, và lượng các ô đã được điền là một số rất nhỏ.

Rõ ràng rằng càng nhiều ô được điền thì độ chính xác của hệ thống sẽ càng được cải thiện. Vì vậy, các hệ thống luôn luôn *hỏi* người dùng về sự quan tâm của họ tới sản phẩm, và muốn người dùng đánh giá càng nhiều sản phẩm càng tốt. Việc đánh giá các sản phẩm, vì thế, không những giúp các người dùng khác biết được chất lượng sản phẩm mà còn giúp hệ thống *biết* được sở thích của người dùng, qua đó có chính sách quảng cáo hợp lý.

2.1.2 Xây dựng Utility Matrix

Không có Utility matrix, gần như không thể gợi ý được sản phẩm tới người dùng, ngoài cách luôn luôn gợi ý các sản phẩm phổ biến nhất. Vì vậy, trong các Recommender Systems, việc xây dựng Utility Matrix là tối quan trọng. Tuy nhiên, việc xây dựng ma trận này thường có gặp nhiều khó khăn. Có hai hướng tiếp cận phổ biến để xác định giá trị *rating* cho mỗi cặp *user-item* trong Utility Matrix:

1. *Nhờ* người dùng *rate* sản phẩm. Amazon luôn *nhờ* người dùng *rate* các sản phẩm của họ bằng cách gửi các email nhắc nhở nhiều lần. Rất nhiều hệ thống khác cũng làm việc tương tự. Tuy nhiên, cách tiếp cận này có một vài hạn chế, vì thường thì người dùng ít khi *rate* sản phẩm. Và nếu có, đó có thể là những đánh giá thiên lệch bởi những người sẵn sàng *rate*.

2. Hướng tiếp cận thứ hai là dựa trên hành vi của *users*. Rõ ràng, nếu một người dùng mua một sản phẩm trên Amazon, xem một clip trên Youtube (có thể là nhiều lần), hay đọc một bài báo, thì có thể khẳng định rằng người dùng đó *thích* sản phẩm đó. Facebook cũng dựa trên việc bạn *like* những nội dung nào để hiển thị *newsfeed* của bạn những nội dung liên quan. Bạn càng đam mê facebook, facebook càng được hưởng lợi, thế nên nó luôn mang tới bạn những thông tin mà khả năng cao là bạn *muốn* đọc. (*Đừng đánh giá xã hội qua facebook*). Thường thì với cách này, ta chỉ xây dựng được một ma trận với các thành phần là 1 và 0, với 1 thể hiện người

dùng thích sản phẩm, 0 thể hiện chưa có thông tin. Trong trường hợp này, 0 không có nghĩa là thấp hơn 1, nó chỉ có nghĩa là người dùng chưa cung cấp thông tin. Chúng ta cũng có thể xây dựng ma trận với các giá trị cao hơn 1 thông qua thời gian hoặc số lượt mà người dùng xem một sản phẩm nào đó. Đôi khi, nút *dislike* cũng mang lại những lợi ích nhất định cho hệ thống, lúc này có thể gán giá trị tương ứng bằng -1 chẳng hạn.

2.2. Hiện tượng Long tail: [2],[3]

Long-tail bao gồm một số ít các mặt hàng phổ biến, các bản hit nổi tiếng, và phần còn lại nằm ở phần đuôi hạng nặng, những mặt hàng không bán chạy. Long-tail cung cấp khả năng khám phá và khám phá — sử dụng các công cụ tự động; chẳng hạn như giới thiệu hoặc bộ lọc được cá nhân hóa — một lượng lớn dữ liệu. [3]

Chúng ta cùng đi vào việc so sánh điểm khác nhau căn bản giữa các *cửa hàng truyền thống* và *cửa hàng trực tuyến*, xét trên khía cạnh lựa chọn sản phẩm để quảng bá. [2]

Các *cửa hàng truyền thống* thường có hai khu vực, một là khu trưng bày, hai là kho. Nguyên tắc dễ thấy để đạt doanh thu cao là trưng ra các sản phẩm phổ biến nhất ở những nơi dễ nhìn thấy và những sản phẩm ít phổ biến hơn được cất trong kho. Cách làm này có một hạn chế rõ rệt: những sản phẩm được trưng ra mang tính phổ biến chứ chưa chắc đã phù hợp với một khách hàng cụ thể. Một cửa hàng có thể có món hàng một khách hàng tìm kiếm nhưng có thể không bán được vì khách hàng không nhìn thấy sản phẩm đó trên giá; việc này dẫn đến việc khách hàng không tiếp cận được sản phẩm ngay cả khi chúng đã được trưng ra. Ngoài ra, vì không gian có hạn, cửa hàng không thể trưng ra tất cả các sản phẩm mà mỗi loại chỉ đưa ra một số lượng nhỏ. Ở đây, phần lớn doanh thu (80%) đến từ phần nhỏ số sản phẩm phổ biến nhất (20%). Nếu sắp xếp các sản phẩm của cửa hàng theo doanh số từ cao đến thấp, ta sẽ nhận thấy có thể phần nhỏ các sản phẩm tạo ra phần lớn doanh số; và một danh sách dài phía sau chỉ tạo ra một lượng nhỏ đóng góp. Hiện tượng này còn được gọi là *long-tail phenomenon*, tức phần *đuôi dài* của những sản phẩm ít phổ biến.

Với các *cửa hàng trực tuyến*, nhược điểm trên hoàn toàn có thể tránh được. Vì *gian trưng bày* của các *cửa hàng online* gần như là vô tận, mọi sản phẩm đều có thể được trưng ra. Hơn nữa, việc sắp xếp online là linh hoạt, tiện lợi với chi phí chuyển đổi gần như bằng 0 khiến việc mang đúng sản phẩm tới khách hàng trở nên thuận tiện hơn. Doanh thu, vì thế có thể được tăng lên.

2.3. Các ứng dụng của Recommendation System: [1]

Chúng tôi đã đề cập đến một số ứng dụng quan trọng của hệ thống khuyến nghị, nhưng ở đây chúng tôi sẽ tổng hợp danh sách ở một nơi duy nhất.

1. *Product Recommendations*: Có lẽ việc sử dụng hệ thống khuyến nghị quan trọng nhất là tại các nhà bán lẻ trực tuyến. Chúng tôi đã ghi nhận cách Amazon hoặc các nhà cung cấp trực tuyến tương tự cố gắng cung cấp cho mỗi người dùng cũ một số gợi ý về sản phẩm mà họ có thể muốn mua. Những đề xuất này không phải là ngẫu nhiên, mà dựa trên quyết định mua hàng của những khách hàng tương tự hoặc dựa trên các kỹ thuật khác mà chúng ta sẽ thảo luận trong chương này.

2. *Movie Recommendations*: Netflix cung cấp cho khách hàng các đề xuất về phim mà họ có thể thích. Các khuyến nghị này dựa trên xếp hạng do người dùng cung cấp. Tầm quan trọng của việc dự đoán xếp hạng chính xác cao đến mức Netflix đã đưa ra giải thưởng trị giá một triệu đô la cho thuật toán đầu tiên có thể đánh bại hệ thống khuyến nghị của chính mình 10% .1 Giải thưởng cuối cùng đã giành được vào năm 2009, bởi một nhóm các nhà nghiên cứu có tên " Bellkor's Pragmatic Chaos, "sau hơn ba năm cạnh tranh.

3. *News Articles:* Các dịch vụ tin tức đã cố gắng xác định các bài báo mà độc giả quan tâm, dựa trên các bài báo mà họ đã đọc trong quá khứ. Sự giống nhau có thể dựa trên sự giống nhau của các từ quan trọng trong tài liệu, hoặc trên các bài báo được đọc bởi những người có cùng sở thích đọc. Các nguyên tắc tương tự cũng áp dụng cho việc đề xuất các blog trong số hàng triệu blog có sẵn, video trên YouTube hoặc các trang web khác nơi nội dung được cung cấp thường xuyên.

2.4. Content-Based Recommendations: [2]

2.4.1. Item profiles:

Trong các hệ thống content-based, tức dựa trên *nội dung* của mỗi *item*, chúng ta cần xây dựng một bộ hộ sơ (profile) cho mỗi item. *Profile* này được biểu diễn dưới dạng toán học là một feature vector. Trong những trường hợp đơn giản, *feature vector* được trực tiếp trích xuất từ *item*. Ví dụ, xem xét các *features* của một bài hát mà có thể được sử dụng trong các Recommendation Systems:

1. *Ca sĩ*. Cùng là bài *Thành phố buồn* nhưng có người thích bản của Đan Nguyên, có người lại thích bản của Đàm Vĩnh Hưng.
2. *Nhạc sĩ sáng tác*. Cùng là nhạc trẻ nhưng có người thích Phan Mạnh Quỳnh, người khác lại thích MTP.
3. *Năm sáng tác*. Một số người thích nhạc xưa cũ hơn nhạc hiện đại.
4. *Thể loại*. Điều này thì chắc rồi, Quan họ và Bolero sẽ có thể thu hút những nhóm người khác nhau.

Có rất nhiều yếu tố khác của một bài hát có thể được sử dụng. Ngoại trừ *Thể loại* khó định nghĩa, các yếu tố khác đều được xác định rõ ràng.

Trong ví dụ ở Hình 1 phía trên, chúng ta đơn giản hoá bài toán bằng việc xây dựng một feature vector hai chiều cho mỗi bài hát: chiều thứ nhất là mức độ *Bolero*, chiều thứ hai là mức độ *Thiếu nhi* của bài đó. Đặt các feature vector cho mỗi bài hát là x1,x2,x3,x4,x5x1,x2,x3,x4,x5. Giả sử các feature vector (ở dạng hàng) cho mỗi bài hát được cho trong Hình 2 dưới đây:

	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	$\leftarrow$ **need to optimize**

Figure 2. Giả sử feature vector cho mỗi item *được cho trong cột cuối cùng. Với mỗi* user*, chúng ta cần tìm một mô hình θ_i tương ứng sao cho mô hình thu được là* tốt nhất.

Bài toán đi tìm mô hình θ_i cho mỗi *user* có thể được coi là một bài toán Regression trong trường hợp *ratings* là một dải giá trị, hoặc bài toán *Classification* trong trường hợp *ratings* là một vài trường hợp cụ thể, như *like/dislike* chẳng hạn. Dữ liệu training để xây dựng mỗi mô hình θ_i là các cặp (*item profile, ratings*) tương ứng với các *items* mà *user* đó đã *rated*. Việc *điền* các giá trị còn thiếu trong ma trận Utility chính là việc dự đoán đầu ra cho các *unrated items* khi áp dụng mô hình θ_i lên chúng.

Việc lựa chọn mô hình Regression/Classification nào tuỳ thuộc vào ứng dụng. Tôi sẽ lấy ví dụ về một mô hình đơn giản nhất: mô hình tuyến tính, mà cụ thể là Linear Regression với regularization, tức Ridge Regression.

2.4.2. Xây dựng hàm mất mát:

Giả sử rằng số *users* là *N*, số *items* là *M*, *utility maxtrix* được mô tả bởi ma trận *Y*. Thành phần ở hàng thứ *m*, cột thứ *n* của *Y* là *mức độ quan tâm* (ở đây là số sao đã *rate*) của *user* thứ *n* lên sản phẩm thứ *m* mà hệ thống đã thu thập được. Ma trận *Y* bị khuyết rất nhiều thành phần tương ứng với các giá trị mà hệ thống cần dự đoán. Thêm nữa, gọi *R* là ma trận *rated or not* thể hiện việc

một *user* đã *rated* một *item* hay chưa. Cụ thể, rijrij bằng 1 nếu *item* thứ *i* đã được *rated* bởi *user* thứ *j*, bằng 0 trong trường hợp ngược lại.

Mô hình tuyến tính:

Giả sử rằng ta có thể tìm được một mô hình cho mỗi *user*, minh hoạ bởi vector cột hệ số w_i và bias b_n sao cho *mức độ quan tâm* của một *user* tới một *item* có thể tính được bằng một hàm tuyến tính:

$$y_{mn} = \mathbf{x}_m \mathbf{w}_n + b_n \quad (1)$$

(*Chú ý rằng* x_m *là một vector hàng,* w_n *là một vector cột.*)

Xét một *user* thứ *n* bất kỳ, nếu ta coi training set là tập hợp các thành phần đã được *điền* của ynyn, ta có thể xây dựng hàm mất mát tương tự như Ridge Regression như sau:

$$\mathcal{L}_n = \frac{1}{2} \sum_{m\,:\,r_{mn}=1} (\mathbf{x}_m \mathbf{w}_n + b_n - y_{mn})^2 + \frac{\lambda}{2} ||\mathbf{w}_n||_2^2$$

Trong đó, thành phần thứ hai là regularization term và λ là một tham số dương. Chú ý rằng regularization thường không được áp dụng lên bias *bn*. Trong thực hành, trung bình cộng của *lỗi* thường được dùng, và *mất mát Ln* được viết lại thành:

$$\mathcal{L}_n = \frac{1}{2s_n} \sum_{m\,:\,r_{mn}=1} (\mathbf{x}_m \mathbf{w}_n + b_n - y_{mn})^2 + \frac{\lambda}{2s_n} ||\mathbf{w}_n||_2^2$$

Trong đó s_n là số lượng các *items* mà *user* thứ *n* đã *rated*. Nói cách khác:

$$s_n = \sum_{m=1}^{M} r_{mn},$$

là tổng các phần tử trên cột thứ nn của ma trận *rated or not* RR.

Các bạn đã thấy *loss function* này *quen* chưa?

Vì biểu thức *loss function* chỉ phụ thuộc vào các *items* đã được rated, ta có thể rút gọn nó bằng cách đặt Y_n là *sub vector* của *y* được xây dựng bằng cách *trích* các thành phần *khác dấu ?* ở cột thứ *n*,

tức đã được *rated* bởi *user* thứ nn trong Utility Matrix *Y*. Đồng thời, đặt X_n là *sub matrix* của ma trận feature XX, được tạo bằng cách *trích* các hàng tương ứng với các *items* đã được *rated* bởi *user* thứ nn. (*Xem ví dụ phía dưới để hiểu rõ hơn*). Khi đó, biểu thức hàm mất mát của mô hình cho *user* thứ nn được viết gọn thành:

$$\mathcal{L}_n = \frac{1}{2s_n}||\hat{\mathbf{X}}_n\mathbf{w}_n + b_n\mathbf{e}_n - \hat{\mathbf{y}}_n||_2^2 + \frac{\lambda}{2s_n}||\mathbf{w}_n||_2^2$$

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

Tài liệu tham khảo:

[1] Recommendation Systems - Stanford InfoLab

http://infolab.stanford.edu/~ullman/mmds/ch9.pdf

[2] Bài 23: Content-based Recommendation Systems

https://machinelearningcoban.com/2017/05/17/contentbasedrecommendersys/

[2] The Long Tail in Recommender Systems:

https://link.springer.com/chapter/10.1007/978-3-642-13287
2_4#:~:text=The%20Long%20Tail%20is%20composed,filters%E2%80%94vast%20amounts%
0of%20data.