TRƯỜNG ĐẠI HỌC KHOA HỌC TỰ NHIÊN
KHOA TOÁN – TIN HỌC
CHUYÊN NGÀNH PHƯƠNG PHÁP TOÁN TRONG TIN HỌC

BIỆN HUỲNH HỮU THỊNH - 1611260

HỆ THỐNG ĐỀ XUẤT CẦU THỦ BÓNG ĐÁ

KHÓA LUẬN TỐT NGHIỆP CỬ NHÂN TOÁN TIN

GIÁO VIÊN HƯỚNG DẪN:
TS. Trần Anh Tuấn

KHÓA 2016-2020

NHẬN XÉT CỦA GIẢNG VIÊN HƯỚNG DẪN

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

NHẬN XÉT CỦA GIẢNG VIÊN PHẢN BIỆN

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

LỜI CẢM ƠN

Tôi xin chân thành cảm ơn Khoa Toán - Tin học trường đại học Khoa Học Tự Nhiên đã tạo điều kiện thuận lợi cho tôi học tập và thực hiện đề tài tốt nghiệp này.

Tôi xin bày tỏ lòng biết ơn sâu sắc tới thầy Trần Anh Tuấn đã tận tình hướng dẫn chỉ bảo tôi trong quá trình thực hiện đề tài.

Tôi xin chân thành cảm ơn quý thầy cô trong khoa đã tận tình giảng dạy, trang bị cho tôi những kiến thức quý báu trong năm vừa qua.

Tôi xin chân thành cảm ơn ông bà, cha mẹ đã luôn động viên ủng hộ vật chất lẫn tinh thần trong suốt thời gian qua.

Tôi xin cảm ơn sự quan tâm giúp đỡ và ủng hộ của các anh chị bạn bè trong quá trình thực hiện khóa đề tài. Mặc dù đã cố gắng hoàn thành khóa luận trong phạm vi và khả năng cho phép nhưng chắc chắn sẽ không tránh khỏi những thiếu sót.

Tôi rất mong nhận được sự thông cảm, góp ý và tận tình chỉ bảo của quý thầy cô và các bạn.

TP. Hồ Chí Minh, tháng 1 năm 2020

Sinh viên thực hiện

Biện Huỳnh Hữu Thịnh

LỜI NÓI ĐẦU

Cùng với sự phát triển của thương mại điện tử cũng như tương tác trực tuyến của con người trong giai đoạn hiện nay, hệ thống gợi ý đã được nghiên cứu và ứng dụng một cách mạnh mẽ và mang lại lợi ích cho cả người cung cấp dịch vụ và người sử dụng dịch vụ. Ý tưởng đằng sau một hệ thống gợi ý là sử dụng các dữ liệu thu thập được từ người dùng và dự đoán, gợi ý cho người dùng những sản phẩm, tính năng, dịch vụ mà người dùng có thể thích, từ đó nâng cao được chất lượng dịch vụ và thu lại lợi nhuận.

Ví dụ như:

- Youtube tự động chuyển các clip liên quan đến clip mình đang xem. Youtube cũng tự gợi ý những clip mà có thể mình sẽ thích.
- Khi mình mua một món hàng trên Amazon, hệ thống sẽ tự động gợi ý "Frequently bought together", hoặc nó biết mình có thể thích món hàng nào dựa trên lịch sử mua hàng của mình.
- Facebook hiển thị quảng cáo những sản phẩm có liên quan đến từ khoá mình vừa tìm kiếm.
- Facebook gợi ý kết bạn.
- Netflix tự động gợi ý phim cho người dùng.

Hệ thống đề xuất ngoài tạo sự tiện lợi, thoải mái cho người dùng còn có thể làm tăng tỷ lệ mua hàng của khách hàng cũng như thời gian giữ khách ở lại với website của mình.

MỤC LỤC

CÁC TỪ VIẾT TẮT

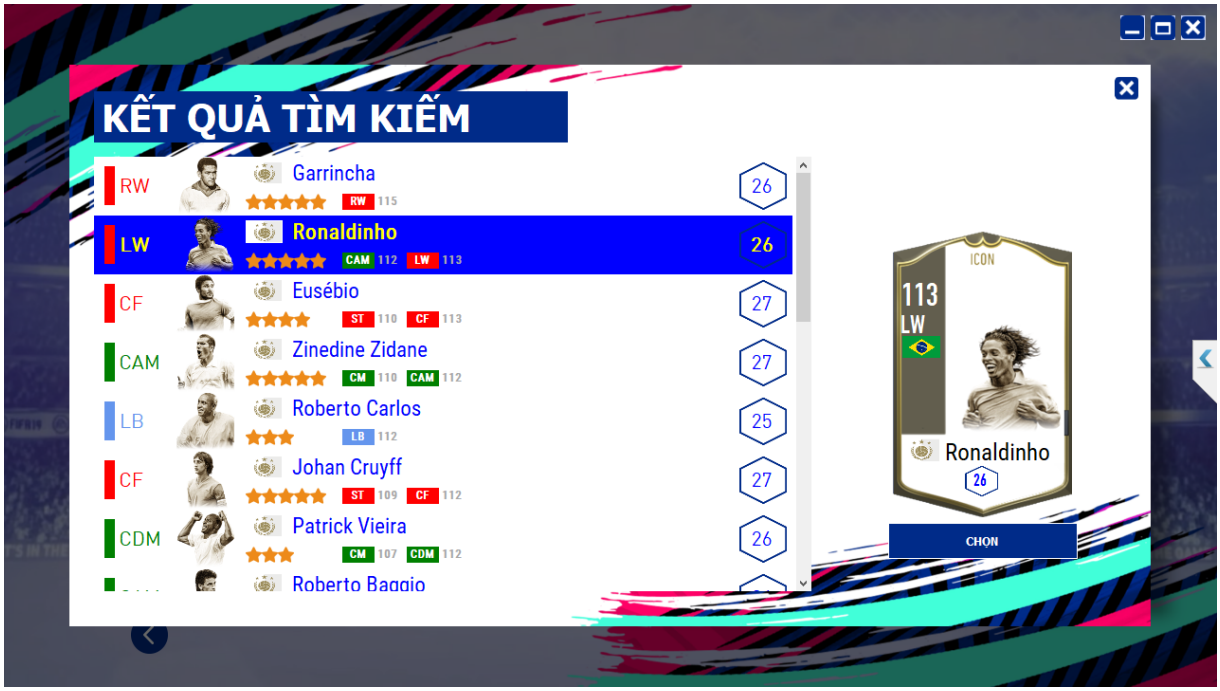
MF	Matrix Factorization	Phân rã ma trận
GMF	Generalizes Matrix Factorization	Phân rã ma trận dạng tổng quát
NeuMF	Neural Matrix Factorization	Phân rã ma trận với mạng neural
MLP	Multi-layer Perceptron	Perceptron nhiều lớp

CHƯƠNG 1: TỔNG QUAN

1.1. Giới thiệu:

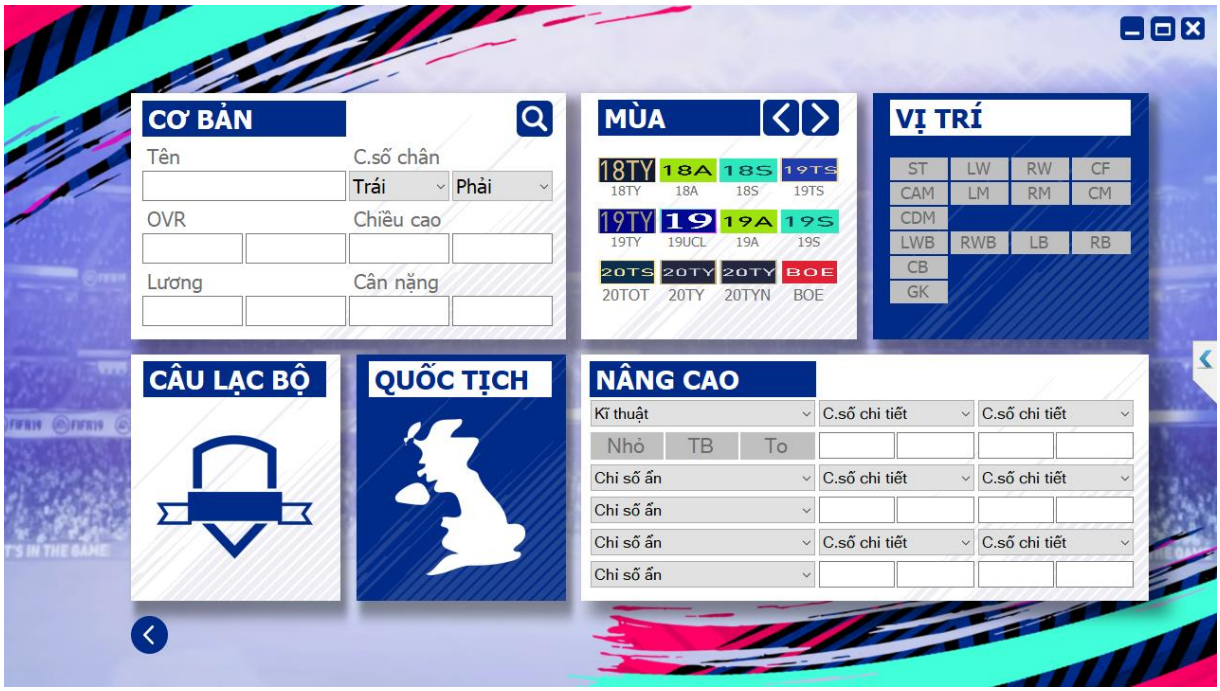
Bài toán là một chức năng trong phần mềm mà tôi đang thiết kế. Trong một tựa game liên quan tới thể thao nói chung và bóng đá nói riêng, mọi thao tác của người chơi đều liên quan trực tiếp tới cầu thủ. Câu hỏi thường gặp hai câu hỏi lớn nhất: Đó là bố trí và chiêu mộ nhân sự cho đội bóng của mình. Thông thường những người chơi sẽ tham khảo đội hình của nhau hoặc chọn theo sở thích của chính mình như chọn theo đội bóng yêu thích, thần tượng của họ.

Một cơ sở dữ liệu về cầu thủ chứa rất nhiều thông tin về nhiều cầu thủ khác nhau qua từng năm.



Hình 1.1.1: Một phần cơ sở dữ liệu cầu thủ của trò chơi FIFA Online 4.

Vì cơ sở dữ liệu về cầu thủ luôn cập nhật và khối lượng rất lớn, cách thông thường để ngời chơi có thể tìm thấy cầu thủ của mình trong dữ liệu rộng lớn là dùng bộ lọc thông tin (filter).



Hình 1.1.2: Một bộ lọc thông tin cơ bản để tìm kiếm cầu thủ.

Bộ lọc tuy đã giải quyết được vấn đề của người dùng, nhưng đôi khi gây khó khăn cho người chơi khi lựa chọn nhiều lần cho mỗi lần tìm kiếm mới.

Phần mềm sẽ dùng mô hình hệ thống đề xuất (recommendation system) và dữ liệu người dùng để đưa ra danh sách các cầu thủ phù hợp với từng người. Người dùng sẽ xem danh sách này để lựa chọn thêm nhanh vào đội hình của mình, thao tác thêm này sẽ được ghi chép lại trong cơ sở dữ liệu người dùng để làm tư liệu cho hệ thống học và đề xuất lại cho những người dùng khác.

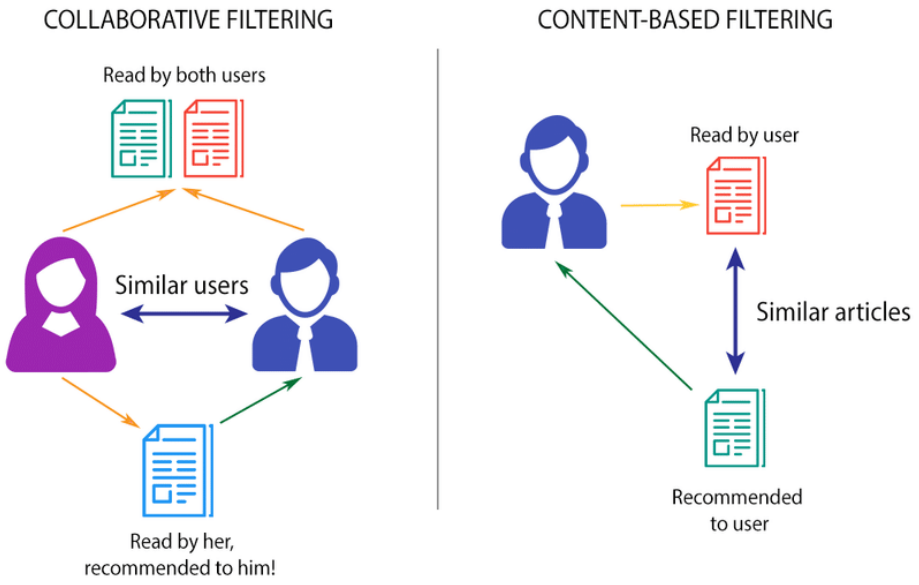
Nhờ vào tính năng này mà người dùng có thể dễ dàng thao tác, chọn được cầu thủ mà không cần phải qua quá nhiều bước lọc thông tin thông thường và cũng có thể tham khảo được các cầu thủ đang được nhiều người sử dụng.

Do thời gian chuẩn bị cho bài luận văn này có hạn nên tôi chỉ tập trung nghiên cứu về các một số hệ thống đề xuất.

1.2. Các hướng tiếp cận

Trong bài toán này đối tượng mà hệ thống gợi ý hướng đến được gọi là người dùng, còn sản phẩm mà hệ thống đưa ra gợi ý gọi là cầu thủ. Hệ thống thường dựa trên tương tác trong quá khứ giữa người dùng và cầu thủ để tạo nên dự đoán trong tương lai.

Các Hệ thống đề xuất thường được chia thành hai nhóm lớn [3].Mỗi hướng tiếp cận đều có điểm mạnh và hạn chế riêng của chúng:



Hình 1.2.1: Mô tả hướng tiếp cận
theo lọc cộng tác (phải) và dựa trên đặt tính (trái)

- Hướng tiếp cận dựa trên đặc tính (Content-based systems): Hệ thống dựa trên các thuộc tính đặc trưng của từng sản phẩm để đưa ra dự đoán về độ phù hợp giữa người dùng và sản phẩm.
- Hướng tiếp cận dựa theo lọc cộng tác (Collaborative filtering): Hệ thống dựa trên lịch sử tương tác của người dùng đối với sản phẩm để tìm ra độ tương đồng giữa các người dùng với nhau, qua đó đề xuất cho họ những sản phẩm thích hợp.

1.3. Đề xuất hướng giải quyết

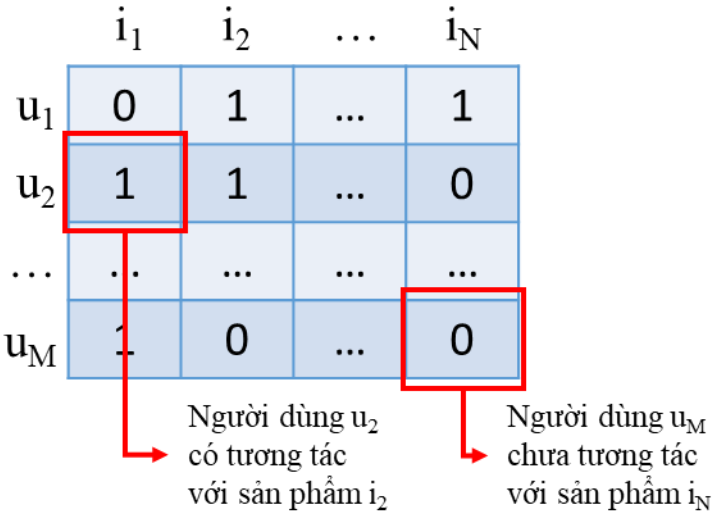
Trong thực tế, việc thu thập dữ liệu của cầu thủ, dữ liệu người dùng rất khó khăn và tốn nhiều thời gian, chi phí. Đối với hướng tiếp cận dựa trên đặc tính ta cần thu thập dữ liệu thông số của từng sản phẩm (cầu thủ) và thông tin sở thích của người dùng. Đối với hướng lọc cộng tác ta phải thu thập một lượng dữ liệu về thói quen, hành vi của người dùng khi tương tác với các sản phẩm có trong hệ thống.

Với nguồn dữ liệu tôi có và ý muốn cá nhân, tôi quyết định tiếp cận bài toán theo hướng dựa theo lọc cộng tác.

Khi xây dựng một hệ thống đề xuất bất kì, người ta thường chọn một hành vi của người dùng để làm mục tiêu tối ưu cho hệ thống và xây dựng một ma trận để biểu diễn sự tương tác của người dùng và sản phẩm.

Gọi $\{Y^1, Y^2, \dots, Y^R\}$ là ma trận tương tác của R hành vi của người dùng đối với sản phẩm. Mỗi ma trận tương tác có kích thước M x N - với M, N lần lượt là số lượng người dùng, số lượng sản phẩm trong hệ thống. Giả sử giá trị mỗi giá trị của ma trận có dạng nhị phân như sau:

$$y_{u,i}^r = \begin{cases} 1, & \text{nếu u tương tác với i với hành vi r} \\ 0, & \text{trường hợp khác} \end{cases} \quad (1)$$



Hình 1.3.1: Ma trận tương tác người dùng – sản phẩm

Cách thức giải quyết vấn đề được mô hình hóa như sau.

Đầu vào: Danh sách dữ liệu gồm các ma trận tương tác người dùng – sản phẩm của R hành vi mục tiêu.

Đầu ra: Mô hình máy học dự đoán sự tương tác giữa người dùng và sản phẩm. Sau khi tối ưu kết quả dự đoán của mô hình, ta có thể đánh giá và xếp hạng mức độ tương tác của từng người dùng đối với sản phẩm trong hệ thống. Từ đó ta có thể đưa ra danh sách sản phẩm khuyến nghị cho họ.

CHƯƠNG 2: CƠ SỞ LÝ THUYẾT

2.1. Không gian định chuẩn

Trong một không gian vector nhiều chiều, một chuẩn được sử dụng để đo độ lớn của một vector hoặc một ma trận.

Định nghĩa 2.1.1: Chuẩn trên $E \in \mathbb{R}^n$ (hoặc$\mathbb{R}^{m*n}$) là một hàm số thực kí hiệu là $||.||$, ánh xạ từ $E \to \mathbb{R}$ (hoặc$\mathbb{R}^{m*n} \to \mathbb{R}$), phải thỏa được cả 4 điều kiện dưới đây:

(i). $\|x\| \geq 0 \; \forall x \in E$

(ii). $\|x\| = 0 \iff x = 0$

(iii). $\|kx\| = |k|\|x\| \; \forall x \in E, k \in \mathbb{R}$

(iv). $\|x + y\| \leq \|x\| + \|y\| \; \forall x, y \in E$

Một số chuẩn được sử dụng phổ biến là chuẩn Euclidean và chuẩn Frobenius:

$$\|A\|_E = \sqrt{\sum_{i=1}^{m}\sum_{j=1}^{n}\left(a_{i,j}\right)^2} \quad (2)$$

$$\|A\|_F = \sqrt{\sum_{i=1}^{m}\sum_{j=1}^{n}\left|a_{i,j}\right|^2} \quad (3)$$

Trong đó $\|.\|_E, \|.\|_F$ lần lượt là chuẩn Euclidean và chuẩn Frobenius trên$\mathbb{R}^{m*n}$.

Để tính độ lớn của khoảng cách giữa 2 ma trận X, Y thuộc không gian định chuẩn $(\mathbb{R}^{m*n}, \|.\|_F)$ Ta có:

$$\|X - Y\|_F = \sqrt{\sum_{i=1}^{m}\sum_{j=1}^{n}\left|x_{i,j} - y_{i,j}\right|^2} \quad (4)$$

2.2. Phương pháp phân tích ma trận thành nhân tử

Phương pháp phân tích ma trận thành nhân tử (Matrix Factorization) là phương pháp chia một ma trận lớn $Y \in \mathbb{R}^{m*n}$ thành hai ma trận có kích thước nhỏ hơn là $P \in \mathbb{R}^{m*k}$ và $Q \in \mathbb{R}^{n*k}$, sao cho ta có thể xây dựng lại Y từ hai ma trận nhỏ hơn này càng chính xác càng tốt, nghĩa là:

$$Y \approx f(P, Q)$$

Với f(P, Q) là một phép tính đại số bất kì giữa P và Q.

Ý tưởng chính của việc áp dụng Matrix Factorization vào hệ thống đề xuất là phân tích ma trận tương tác người dùng – sản phẩm thành 2 ma trận nhỏ hơn là ma trận người dùng và ma trận sản phẩm có kích thước lần lượt là MxK và NxK – với M là số lượng người dùng, N là số lượng sản phẩm có trong hệ thống và K là số chiều ta muốn phân rã. Có thể xem P và Q là danh sách các vector đặc trưng (feature vector) hoặc thuộc tính ẩn (latent feature) với K đặc trưng của người dùng và sản phẩm.

Cụ thể hơn, ta phải xây dựng P và Q sao cho trung bình khoảng cách của $f(P, Q)$ với Y là nhỏ nhất có thể. Ta xây dựng hàm mất mát như sau:

$$\mathcal{L}(P, Q) = \frac{1}{2 * M * N} \|Y - f(P, Q)\| \quad (5)$$

Để tối ưu hàm mất mát ta sử dụng phương pháp Gradient Descent để tìm ra cực tiểu địa phương (local minimum) của hàm. Ta sẽ lặp lại quá trình đạo hàm, cập nhật giá trị trọng số cho ma trận P, Q cho tới khi giá trị của hàm mất mát hội tụ về 0.

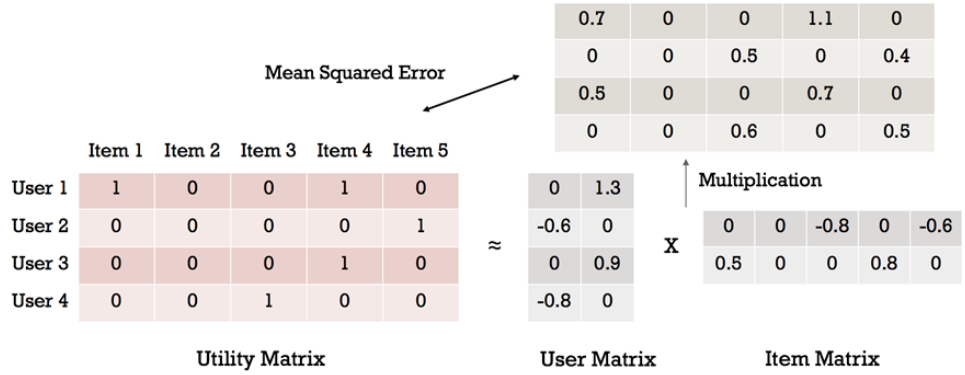
$$\begin{cases} P = P - \eta_P \nabla_P \mathcal{L}(P, Q) \\ Q = Q - \eta_Q \nabla_Q \mathcal{L}(P, Q) \end{cases} \quad (6)$$

$\boldsymbol{\eta_P}$, $\boldsymbol{\eta_Q}$ lần lượt là hệ số học máy của ma trận P và Q.

Thuật giải 1: Matrix Factorization

1. Khởi tạo P và Q
2. Lặp lại
3. Tính chênh lệch $\mathcal{L}(P,Q) = \frac{1}{2*M*N}\|Y - f(P,Q)\|$
4. Cập nhật $P \leftarrow P - \eta_P \nabla_P \mathcal{L}(P,Q)$
5. Cập nhật $Q \leftarrow Q - \eta_Q \nabla_Q \mathcal{L}(P,Q)$
6. Đến khi $\mathcal{L}(P,Q)$ hội tụ

Tùy vào phép tính đại số, các tính khoảng cách ta có một cách đạo hàm riêng tương ứng với từng phép toán. Với phương pháp phân rã ma trận điển hình thì phép toán $f(P,Q) = PQ^T$ thường được dùng là phép nhân ma trận thông thường và cách tính khoảng cách là bình phương của chuẩn Frobenius.



Hình 2.1.1: Một mô hình Matrix Factorization.

Ngoài ra để tránh hiện tượng overfitting ta thay đổi mô hình một chút để tránh overfitting trong khi vẫn giữ được tính tổng quát của nó, phương pháp này gọi là Regularization. Nên với hàm mất mát trên (5) ta có thể viết lại như bên dưới:

$$\mathcal{L}(P,Q) = \frac{1}{2*M*N}\|Y - PQ^T\|_F^2 + \underbrace{\frac{\lambda_P}{2}\|P\|_F^2 + \frac{\lambda_Q}{2}\|Q\|_F^2}_{Regularization} \quad (7)$$

Để việc tính đạo hàm của $\mathcal{L}(P,Q)$ theo biến P và Q thuận tiện hơn, ta có biến đổi (7) thành (8).

$$\mathcal{L}(P,Q) = \frac{1}{2*M*N} tr\left((Y - PQ^T) * (Y - PQ^T)^T\right)$$

$$+ \frac{\lambda_P}{2} tr(PP^T) + \frac{\lambda_Q}{2} tr(QQ^T) \quad (8)$$

Do $\|A\|_F = \sqrt{\sum_{i=1}^{m}\sum_{j=1}^{n}|a_{i,j}|^2} = \sqrt{tr(AA^T)^2}$ với $A \in \mathbb{R}^{m\times n}$

Trong đó $tr(A)$ là vết của ma trận vuông $A \in \mathbb{R}^{n\times n}$ được tính bằng:

$$tr(A) = a_{11} + a_{22} + a_{33} + \cdots + a_{nn} = \sum_{i=1}^{n} a_{ii} \quad (9)$$

Ta có các tính chất sau của vết của ma trận[5]:

(i). $tr(A + B) = tr(A) + tr(B)$ với $A, B \in \mathbb{R}^{n\times n}$

(ii). $tr(ABC) = tr(CAB) = tr(BCA)$

Với $A \in \mathbb{R}^{m\times n}, B \in \mathbb{R}^{n\times k}, C \in \mathbb{R}^{k\times m}$

(iii). $\nabla_X tr(AX) = A^T$ Với $A \in \mathbb{R}^{m\times n}, X \in \mathbb{R}^{n\times m}$

(iv). $\nabla_X tr(X^T A) = A$ Với $A \in \mathbb{R}^{m\times n}, X \in \mathbb{R}^{n\times m}$

(v). $\nabla_X tr(X^T AX) = (A + A^T)X$ Với $A \in \mathbb{R}^{m\times n}, X \in \mathbb{R}^{n\times m}$

(vi). $\nabla_X tr(XAX^T) = X(A + A^T)$ Với $A \in \mathbb{R}^{m\times n}, X \in \mathbb{R}^{n\times m}$

(vii). $\nabla_A tr(AA^T) = 2A$ Với $A \in \mathbb{R}^{n\times n}$

Ta dùng các tính chất trên để tính đạo hàm của $\mathcal{L}(P,Q)$ theo P và Q như sau:

$$\nabla_P \mathcal{L}(P,Q) = \frac{1}{2*M*N} \nabla_P tr\left((Y - PQ^T) * (Y - PQ^T)^T\right)$$

$$+ \frac{\lambda_P}{2} \nabla_P tr(PP^T) + \frac{\lambda_Q}{2} \nabla_Q tr(QQ^T) \quad (10)$$

$$= \frac{1}{2MN} \nabla_P tr\left((Y - PQ^T) * (Y^T - (PQ^T)^T)\right) + \lambda_P P + 0 \quad (11)$$

$$= \frac{1}{2MN} \nabla_P tr\big((Y - PQ^T) * (Y^T - QP^T)\big) + \lambda_P P \tag{12}$$

$$= \frac{1}{2MN} \nabla_P tr(YY^T - YQP^T - PQ^T Y^T + PQ^T QP^T) + \lambda_P P \tag{13}$$

$$\nabla_P \mathcal{L}(P,Q) = \frac{1}{2MN} [\nabla_P tr(YY^T) - \nabla_P tr(YQP^T) - \nabla_P tr(PQ^T Y^T) + \nabla_P tr(PQ^T QP^T)] + \lambda_P P \tag{14}$$

$$= \frac{1}{2MN} [0 - YQ - YQ + P(Q^T Q + (Q^T Q)^T)] + \lambda_P P \tag{15}$$

$$= \frac{1}{2MN} [0 - YQ - YQ + P(Q^T Q + (Q^T Q)^T)] + \lambda_P P \tag{16}$$

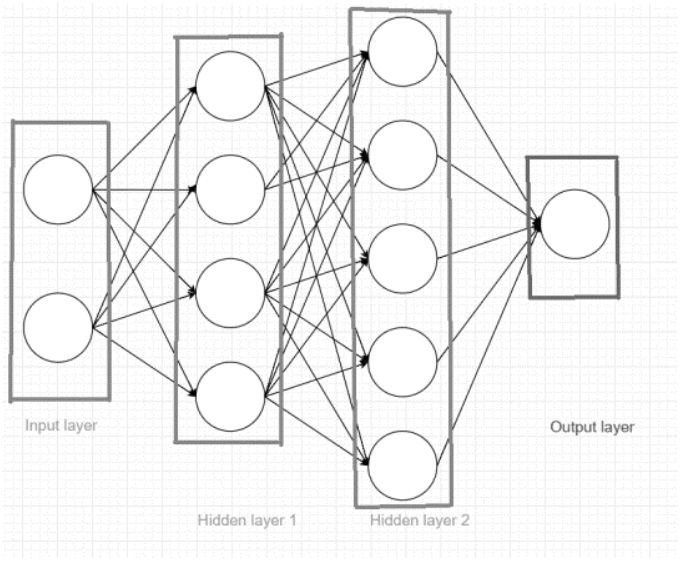
$$= \frac{1}{MN} [PQ^T Q - YQ] + \lambda_P P \tag{17}$$

Thực hiện tương tự với đạo hàm theo Q ta thu được kết quả (18)

$$\begin{cases} \nabla_P \mathcal{L}(P,Q) = \frac{1}{MN}(PQ^T Q - YQ) + \lambda_P P \\ \nabla_Q \mathcal{L}(P,Q) = \frac{1}{MN}(QP^T P - Y^T P) + \lambda_Q Q \end{cases} \tag{18}$$

2.3. Phương pháp Multilayer Perceptron

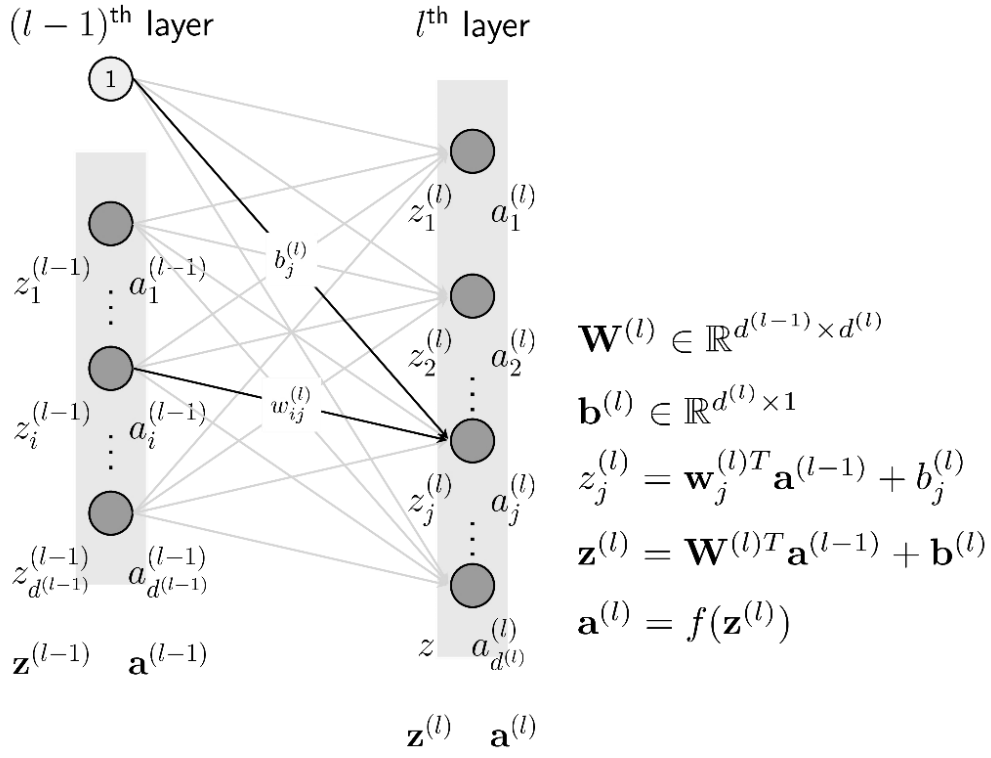
Mạng neural (Neural Network) là một mô hình có khả năng học được các khuôn mẫu phức tạp của dữ liệu qua các lớp (layer) neurons có nhiệm vụ biến đổi dữ liệu theo các công thức toán học. Các lớp neurons nằm giữa đầu vào (input layer) và đầu ra (output layer) được gọi là hidden layers. Một mạng neural có thể khám phá và học các mối quan hệ giữa các thuộc tính của dữ liệu và sử dụng các quan hệ này để đưa ra dự đoán.



Hình 2.2.1: Mô hình đơn giản của Mạng neural.

Mỗi mô hình mạng neural luôn có 1 input layer, 1 output layer, có thể có hoặc không các hidden layer. Tổng số layer trong mô hình được quy ước là số layer – 1 (Không tính input layer). Một mạng Multilayer Perceptron (MLP) phải có ít nhất 3 lớp bao gồm lớp đầu vào, lớp đầu ra và ít nhất 1 lớp ẩn ở giữa 2 lớp trên.

Ví dụ như ở hình 2.2.1 có 1 input layer, 2 hidden layers và 1 output layer. Số lượng layer của mô hình là 3 layers.



Hình 2.2.2: Các ký hiệu sử dụng trong mạng neural.

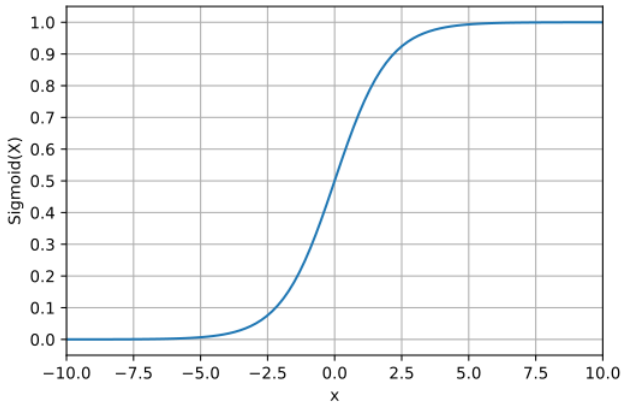
Một đơn vị neuron nhân tạo (còn được gọi là percepton) là một hàm biến đổi toán học nhận một hoặc nhiều đầu vào đã được nhân với các giá trị trọng số (weights), cộng các giá trị đó lại với nhau thành một giá trị duy nhất.

Sau đó giá trị này được đưa vào một hàm phi tuyến gọi là hàm kích hoạt (hay activation function) và kết quả của hàm này chính là đầu ra của neuron. Ta dùng hàm kích hoạt để làm cho mô hình học được thêm nhiều trường hợp hơn. Dưới đây là một số activation function phổ biến:

Sigmoid function (Logistic Function):

$$sigmoid(x) = \frac{1}{1+e^{-x}} \tag{19}$$

$$\frac{\partial sigmoid(x)}{\partial x} = sigmoid(x)\big(1 - sigmoid(x)\big) \tag{20}$$

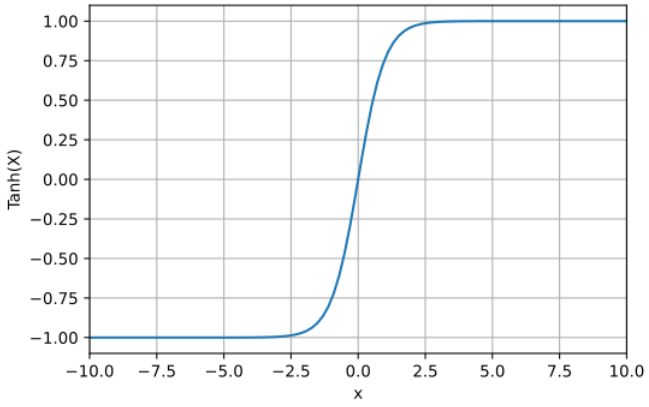


Hình 2.2.3: Biểu dồ hàm $sigmoid(x)$

Tanh:

$$tanh(x) = \frac{e^{x} - e^{-x}}{e^{x} + e^{-x}} \tag{21}$$

$$\frac{\partial tanh(x)}{\partial x} = 1 - tanh^{2}(x) \tag{22}$$

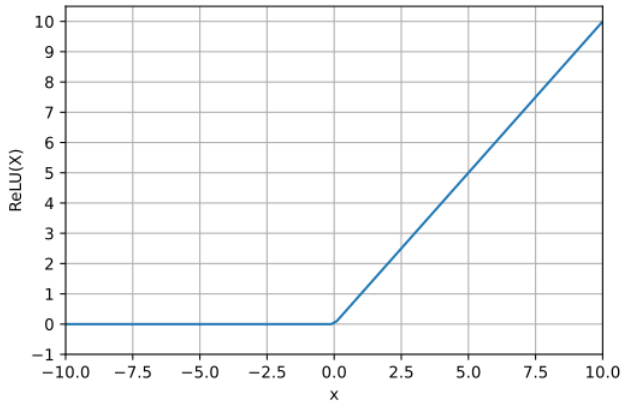


Hình 2.2.4: Biểu dồ hàm $tanh(x)$

Rectified Linear Unit (ReLU):

$$ReLU(x) = max(0, x) \tag{23}$$

$$\frac{\partial ReLU(x)}{\partial x} = \begin{cases} 0, x \leq 0 \\ 1, x > 0 \end{cases} \tag{24}$$



Hình 2.2.5: Biểu dồ hàm $ReLU(x)$

Mỗi percepton trong hidden layer và output layer:

- Liên kết với tất cả các percepton ở layer trước đó với các trọng số w riêng.
- Mỗi node có 1 hệ số bias b riêng.
- Diễn ra 2 bước: tính tổng linear và áp dụng activation function.

Trong mỗi mô hình mạng neural có 2 quá trình chính, thứ nhất là quá trình truyền thẳng (forward propagation), và thứ 2 là quá trình lan truyền ngược (Back propagation).

Quá trình truyền thẳng khá đơn giản, chỉ cần sử dụng các node đầu ra của lớp trước đó làm dữ liệu đầu vào cho lớp tiếp theo, tiếp tục cho tới khi tới lớp cuối cùng cũng là lớp output thì ta thu được kết quả của sự lan truyền thẳng.

Quá trình lan truyền ngược cơ bản cũng dựa trên các phương pháp tối ưu hàm mất mát. Nhưng do cấu trúc mạng neural có sự liên hệ giữa các lớp, lớp sau sử dụng thông tin của lớp trước đó, nên quá trình này trở nên phức tạp hơn [7]. Như đã trình bày ở trên quá trình truyền thẳng có thể xem như các chuỗi dài các phương trình lồng nhau. Như phương trình (6):

$$f(x) = A\Big(B\big(C(x)\big)\Big) \tag{25}$$

Với A, B, C là 3 hàm kích hoạt của 3 lớp khác nhau. Bằng việc áp dụng chain rule, chúng ta có thể tính được đạo hàm của $f(x)$ theo x như sau:

$$\frac{\partial f}{\partial x} = \frac{\partial A}{\partial x} = \frac{\partial A}{\partial B}\frac{\partial B}{\partial x} = \frac{\partial A}{\partial B}\frac{\partial B}{\partial C}\frac{\partial C}{\partial x} \tag{26}$$

Dựa theo cách làm trên ta có thể cập nhật trọng số $w_{i,j}^{(l)}$ ở lớp l bất kì, để hàm mất mát $J(W)$hội tụ:

$$\frac{\partial J}{\partial w_{i,j}^{(l)}} = \frac{\partial J}{\partial z_j^{(l)}}\frac{\partial z_j^{(l)}}{\partial w_{i,j}^{(l)}} \tag{27}$$

Nếu l là lớp output phương trình (21) có thể biến đổi như sau:

$$\frac{\partial J}{\partial z_j^{(l)}}\frac{\partial z_j^{(l)}}{\partial w_{i,j}^{(l)}} = e_j^{(l)} a_i^{(l-1)} \tag{28}$$

Với $e_j^{(l)}$là độ sai lệch giữa thực tế và dự đoán.

Ta làm tương tự quá trình cập nhật trọng số như hai phương pháp hồi quy trên cho tới khi hàm $J(W)$hội tụ.

2.4. Phương pháp Neural Matrix Factorization

Với công thức tổng quát của MF nêu trên ($Y \approx f(P, Q)$) ta có thể kết hợp giữa MF và MLP để tạo ra các mô hình lai bằng các thay đổi $f(P, Q)$ theo từng mục tiêu mà hệ thống đề xuất muốn nhắm đến.

Trường hợp $f(P, Q)$ là một phép nhân ma trận thông thường:

$$f(P, Q) = PQ^T \tag{29}$$

$$\phi^{dot} = \widehat{y_{m,n}} = p_m \odot q_n = \sum_{k=1}^{K} p_{m,k} * q_{n,k} \tag{30}$$

Với $m, n \in \mathbb{N}, 0 < m < M, 0 < n < N$; p_m, q_nlần lượt là latent feature vector của người dùng m và sản phẩm n. $\widehat{y_{m,n}}$ là kết quả dự báo của hệ thống cho khả năng

tương tác của người dùng m đối với sản phẩm n. k là số chiều của latent feature ta muốn phân rã.

Mô hình GMF – Generalized matrix factorization và trọng số máy học (learned weights) $w \in \mathbb{R}^k$:

$$\phi^{GMF} = \widehat{y_{m,n}} = \sigma(w(p_m \odot q_n)^T)$$

$$= \sigma\left(\sum_{k=1}^{K} w_k * p_{m,k} * q_{m,k}\right) \quad (30)$$

Mô hình MLP:

$$f_{W,b}(x) = \sigma(Wx + b) \quad (31)$$

$$\phi^{MLP} = \widehat{y_{m,n}} = f_{W_l,b_l}(\dots f_{W_1,b_1}([p_m, q_n]) \dots) \quad (32)$$

Mô hình NeuMF – là một mô hình lai giữa GMF và MLP:

$$\phi^{NeuMF} = \widehat{y_{m,n}} \quad (33)$$

$$= \sigma\left(w\left[\phi^{MLP}([p_{m,0\to j}, q_{n,0\to j}]), \phi^{GMF}([p_{m,j+1\to k}, q_{n,j+1\to k}])\right]^T\right)$$

Mô hình NeuMF sử dụng một phần của feature vector (từ 1 tới j đặc trưng) để sử dụng cho GMF và phần còn lại sử dụng cho MLP.

2.5. Các phương pháp đánh giá mô hình máy học

Đánh giá độ chính xác của mô hình tư vấn là một khâu không thể thiếu trong quy trình xây dựng hệ thống. Nó giúp cho người thiết kế mô hình lựa chọn mô hình, kiểm tra độ chính xác của mô hình trước khi đưa mô hình vào ứng dụng thực tế. Để đánh giá mô hình tư vấn lọc cộng tác, người xây dựng hệ thống có các cách sau.

2.5.1 Đánh giá dựa trên các xếp hạng.

Phương pháp này đánh giá độ chính xác của mô hình bằng cách so sánh giá trị xếp hạng dự đoán với giá trị thực hay chính xác hơn là tìm ra giá trị trung bình lỗi dựa vào ba đại lượng Root mean square error, Mean squared error và Mean absolute error. Mô hình được đánh giá là tốt khi các đại lượng này có giá trị thấp.

Root mean square error (RMSE): Độ lệch chuẩn giữa giá trị thực và giá trị xếp hạng.

$$\text{RMSE} = \sqrt{\frac{\sum_{i=0}^{|Y|}(y_i - \hat{y}_i)^2}{|Y|}} \text{ với } y_i \in Y, \hat{y}_i \in \hat{Y} \quad (34)$$

Mean squared error (MSE): Trung bình của bình phương độ lệch giữa giá trị thực và giá trị xếp hạng dự đoán.

$$\text{MSE} = \text{RMSE}^2 = \frac{\sum_{i=0}^{|Y|}(y_i - \hat{y}_i)^2}{|Y|} \text{ với } y_i \in Y, \hat{y}_i \in \hat{Y} \quad (35)$$

Mean absolute error (MAE): Trung bình trị tuyệt đối của độ lệch giữa giá trị thực và giá trị xếp hạng dự đoán.

$$\text{MSE} = \frac{\sum_{i=0}^{|Y|}|y_i - \hat{y}_i|}{|Y|} \text{ với } y_i \in Y, \hat{y}_i \in \hat{Y} \quad (36)$$

Với Y là tập tất cả các điểm đánh giá thực của người dùng cho cầu thủ;

$\hat{Y}$ là tập tất cả các điểm đánh giá dự đoán của người dùng cho cầu thủ.

2.5.2 Đánh giá dựa trên các đề xuất

Phương pháp này đánh giá độ chính xác của mô hình bằng cách so sánh các đề xuất của mô hình đưa ra với các lựa chọn thực của người dùng. Phương pháp này sử dụng ma trận hỗn độn 2x2 (Confusion matrix) để tính độ chính xác (Precision), độ bao phủ (Recall) và trung bình điều hòa giữa độ chính xác và độ bao phủ (F-measure). Mô hình được đánh giá là tốt khi ba chỉ số trên có giá trị cao.

Bảng 2.5.2: Ma trận hỗ độn

Lựa chọn của người dùng	Hệ thống đề xuất	
	Có	Không
Có	TP	FN
Không	FP	TN

Trong đó:

TP: Những cầu thủ được đề xuất đã có trong đội hình của người dùng.

FP: Những cầu thủ được đề xuất không có trong đội hình của người dùng.

FN: Những cầu thủ không đề xuất đã có trong đội hình của người dùng.

TN: Những cầu thủ không đề xuất không có trong đội hình của người dùng.

Độ chính xác (Precision):

$$\text{Precision} = \frac{TP}{TP+FP} \quad (37)$$

Độ bao phủ (Recall):

$$\text{Recall} = \frac{TP}{TP+FN} \quad (38)$$

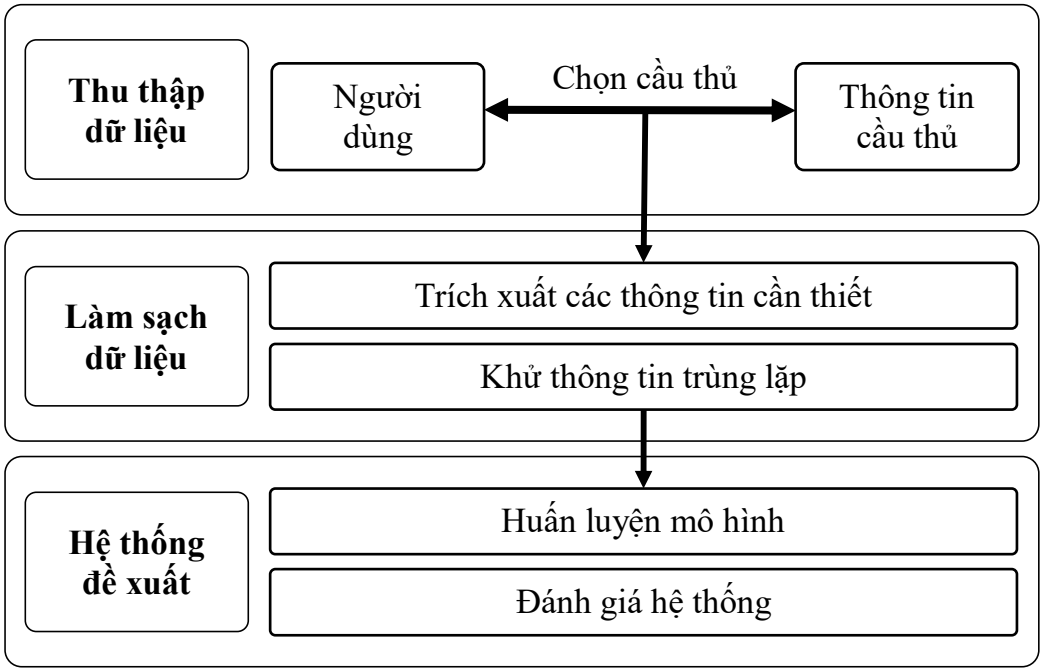
Trung bình điều hòa giữa độ chính xác và độ bao phủ (F-Measure):

$$\text{F-Measure} = \frac{2*\text{Precision}*\text{Recall}}{\text{Precision}+\text{Recall}} \quad (39)$$

CHƯƠNG 3: THỰC NGHIỆM

3.1. Mô tả phương pháp

Các bước thực nghiệm bắt đầu từ bước thu thập dữ liệu người dùng. Khi người dùng sử dụng phần mềm hệ thống sẽ ghi lại các thao tác người dùng, mã hóa thông tin người dùng thành các mã định danh. Sau một khoảng thời gian thu thập dữ liệu, ta sẽ trích xuất dữ liệu hiện có, tiến hành tái cấu trúc lại dữ liệu (nếu không phù hợp) sao cho phù hợp với đầu vào của phương pháp đề xuất.



Hình 3.1.1: Sơ đồ tổng quát quá trình thực hiện.

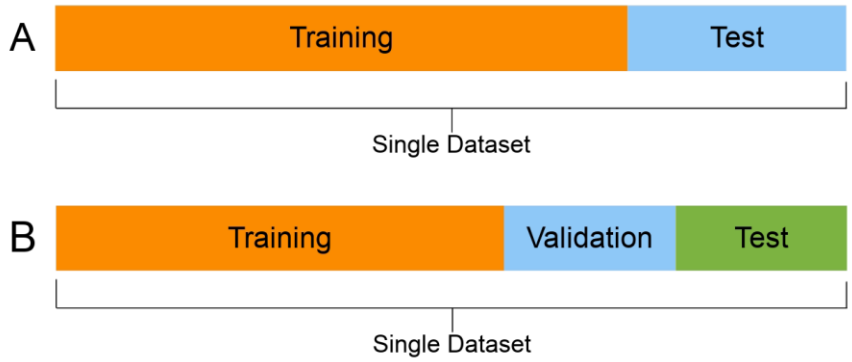
Ở bước đầu tiên, phần mềm sẽ ghi lại hành động thêm cầu thủ vào đội hình của người dùng bằng cách lưu mã định danh của họ và mã định danh cầu thủ tương ứng vào cơ sở dữ liệu. Sau đó ta sẽ trích xuất thông tin thành dạng file CSV và làm dữ liệu đầu vào cho mô mình của mình.

Hình 3.1.2: Mô tả quá trình thu thập dữ liệu.

Bước 2, sau khi có danh sách thông tin, ta tiến hành chọn lọc các trường dữ liệu cần thiết, cụ thể là trường dữ liệu mã định danh người dùng và cầu thủ. Nếu danh sách thông tin có các dòng trùng lặp ta phải tiến hành loại bỏ các dòng đó.

Hình 3.1.3: Quá trình lọc trùng thông tin

Bước 3, ta chia dữ liệu thành 3 tập dữ liệu con: Một dùng để huấn luyện (training dataset), một dùng để đánh giá quá trình huấn luyện (validation dataset) và phần còn lại dùng để đánh giá hiệu xuất của toàn bộ hệ thống khi đã hoàn thành huấn luyện (test dataset).



Hình 3.1.4: Mô tả cách phân chia dữ liệu.

Cuối cùng, ta sẽ sử dụng các tập dữ liệu đó trong các bước huấn luyện mô hình máy học tương ứng, ghi lại kết quả, tổng kết và đánh giá hiệu quả của từng mô hình.

3.2. Mô tả dữ liệu

Trong khuôn khổ bài luận văn này, tôi sẽ sử dụng 2 bộ dữ liệu sau để thực hiện các phương pháp trên:

Taobao Dataset. Bộ dữ liệu được công bố ở IJCAI15 challenge, được trích từ dữ liệu của Taobao, một trang web về thương mại điện tử ở Trung Quốc. Nó ghi lại 2 loại hành vi của người dùng đó là xem và mua trong khoảng thời gian từ tháng 5/2014 tới hết tháng 11/2014.

Players Dataset. Bộ dữ liệu mà phần mềm của tôi thu thập. Nó bao gồm các tương tác của người dùng khi sử dụng phần mềm, cụ thể là hành động thêm cầu thủ vào đội hình. Dữ liệu được ghi lại từ tháng 8/2018 tới giữa tháng 9/2020.

Bảng 3.2.1: Thống kê của bộ dữ liệu.

Bộ dữ liệu	Số người dùng	Số sản phẩm	Lượt xem	Lượt mua
Taobao	49,393	5,512	66,802	8,391
Players	242	902	2,101	-

3.3. Xây dựng mô hình

Sau bước làm sạch dữ liệu ta tiến hành tiền xử lý để cấu trúc lại dữ liệu sao cho khớp với yêu cầu đầu vào của mô hình. Bước này sẽ áp dụng cho tất cả các mô hình sử dụng trong luận văn.

Giả sử danh sách dữ liệu gồm C dòng, với danh sách mã định danh người dùng và sản phẩm ta thiết lập lại và gán cho nó một số thứ tự riêng biệt để tiết kiệm bộ nhớ và dễ dàng thao tác trên dữ liệu hơn. Do danh sách ghi lại sự tương tác (thêm) nên giá trị tương tác của danh sách này mặt định là bằng 1 tượng trưng cho xác suất để người dùng tương tác với sản phẩm là 100%, và các trường hợp không thêm sẽ có giá trị là 0.

Bảng 3.3.1: Mô tả cách số hóa dữ liệu.

(a) Dữ liệu trước khi số hóa **(b)Dữ liệu sau khi số hóa**

Mã người dùng	Mã cầu thủ		Số thứ tự người dùng	Số thứ tự cầu thủ	Giá trị
YPX56QBJ	zgqjravo	→	8	7	1
YPX56QBJ	nngznqno	→	8	55	1
M1PB6IO7	rzwogrld	→	45	39	1
…	…		…	…	…
YLAN7CIC	nngznqno	→	20	55	1

Từ danh sách dữ liệu như bảng 3.3.1(a) ta số hóa thành bảng 3.3.1(b). Ta gọi các cột của bảng dữ liệu 3.3.1(b) lần lượt như sau để thuận tiện theo dõi: U, I, Y.

Với mỗi số thứ tự của người dùng và sản phẩm, ta sử dụng phương pháp mã hóa one-hot vector. Phương pháp này có đầu vào là mã định danh i và đầu ra là một vector nhị phân v với duy nhất giá trị tại vị trí $v_i = 1$, tất cả giá trị còn lại bằng 0.

Bảng 3.3.2: Một số ví dụ về one-hot vector encoding.

ID	Tên cầu thủ	One-hot vector
1	Wayne Rooney	[0,1,0,0, … ,0]
3	Cristiano Ronaldo	[0,0,0,1,0, … ,0]
5	Lionel Messi	[0,0,0,0,0,1,0 … ,0]

Phương pháp này giúp ta có thể tính toán nhanh hơn và thay vì truy cập ngẫu nhiên từng dòng trong hai ma trận phân rã P, Q thì ta có thể tận dụng tính chất của phép nhân ma trận để thuận tiện trong quá trình tính toán.

Gọi $\rho_u = [\dots,0,1,0 \dots] \in [0,1]^M$ và $,\varrho_i \in [0,1]^N$ lần lượt là one-hot vector của người dùng u và sản phẩm i. Thì feature vector của người dùng u, $p_u \in \mathbb{R}^K$, sẽ được tính như sau:

$$p_u = \rho_u * P \quad (40)$$

Tương tự ta có $q_i \in \mathbb{R}^K$ là feature vector của sản phẩm.

$$q_i = \varrho_i * Q \quad (41)$$

Mở rộng ta có thể ghi:

$$q = \begin{bmatrix} q_0 \\ \dots \\ q_C \end{bmatrix}_{C\times K} = \begin{bmatrix} \varrho_0 \\ \dots \\ \varrho_C \end{bmatrix}_{C\times N} * \begin{bmatrix} q_{0,0} & \dots & q_{0,k} \\ \dots & \dots & \dots \\ q_{n,0} & \dots & q_{n,k} \end{bmatrix}_{N\times K} \quad (42)$$

$$[0 \quad 0 \quad 1] \times \begin{bmatrix} 0.7 & 0.5 & 0.0 \\ 0.65 & 0.9 & 0.1 \\ 0.4 & 0.6 & 0.001 \end{bmatrix} = [0.4 \quad 0.6 \quad 0.001]$$

Hình 3.3.1: One-hot vector có công dụng như một khóa tra cứu

Ta lần lượt xây dựng 3 mô hình máy học được đề cập ở Chương II – GMF, MLP và NeuMF – và so sánh độ chính xác và hiệu quả của từng mô hình. Ta cài đặt cấu hình của mô hình theo bảng bên dưới:

Bảng 3.3.1: Bảng cấu hình mô hình máy học.

	GMF	**NCF**	**NeuMF**
Layers	-	[20 Relu,10 Relu]	[20 Relu,10 Relu]
Activation function	Sigmoid	Sigmoid	Sigmoid
Latent features (K)	64	64	64
Learning rate	0.001	0.001	0.001

Để thực hiện phần thực nghiệm cho bài luận văn này tôi sử dụng ngôn ngữ lập trình Python phiên bản 3.6.8, Jupyter-Lab IDE, Tensorflow-GPU 2.2 và Keras 2.4. Các phần khởi tạo ma trận phân rã ta có thể dùng Embedding layer trong tensorflow để thay thế, và chúng trên lý thuyết là tương đương nhau.

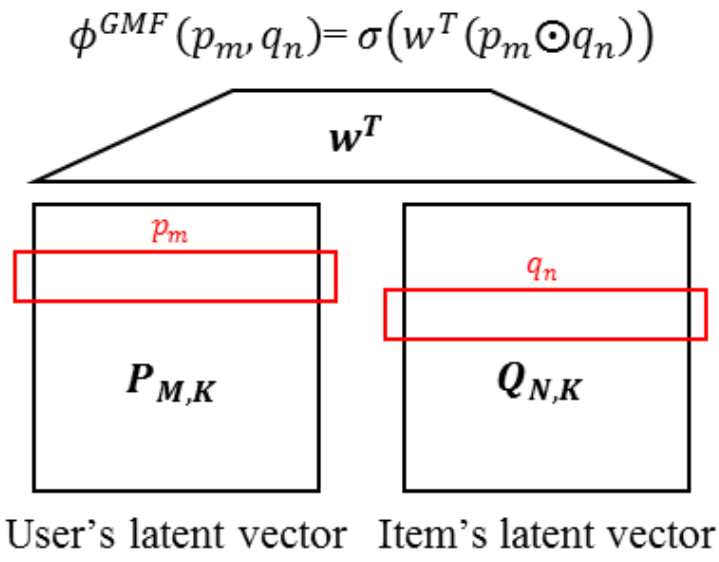
Generalized Matrix Factorization

Sau khi thu được kết quả mã hóa p, q từ bước one-hot encoding. Ta nối hai vector lại với nhau theo từng phần tử $(p \odot q)$. Kế tiếp ta nhân với ma trận trọng số $w \in \mathbb{R}^K$ và đưa vào activation function. Hàm kích hoạt được sử dụng cho phương pháp này là sigmoid.

$$\mathcal{L}(P,Q) = \left\| Y - sigmoid\left(\sum_{k=1}^{K} w_k * p_{m,k} * q_{m,k}\right)\right\|_F \quad (43)$$

Thuật giải 2: Generalized Matrix Factorization

1. Đọc và làm sạch dữ liệu U, I, V
2. Khởi tạo P và Q
3. Lặp lại
4. Duyệt từng dòng dữ liệu $l := 0 \rightarrow C$
5. Mã hóa u_l, i_l thành ρ_l và ϱ_l
6. Tính p và q
7. Tính $\mathcal{L}(P,Q) = \left\|Y - sigmoid\left(W^T(p \odot q)\right)\right\|_F$
8. Cập nhật P, Q, W
9. Đến khi $\mathcal{L}(P,Q)$ hội tụ



Hình 3.3.1: Mô hình GMF [8].

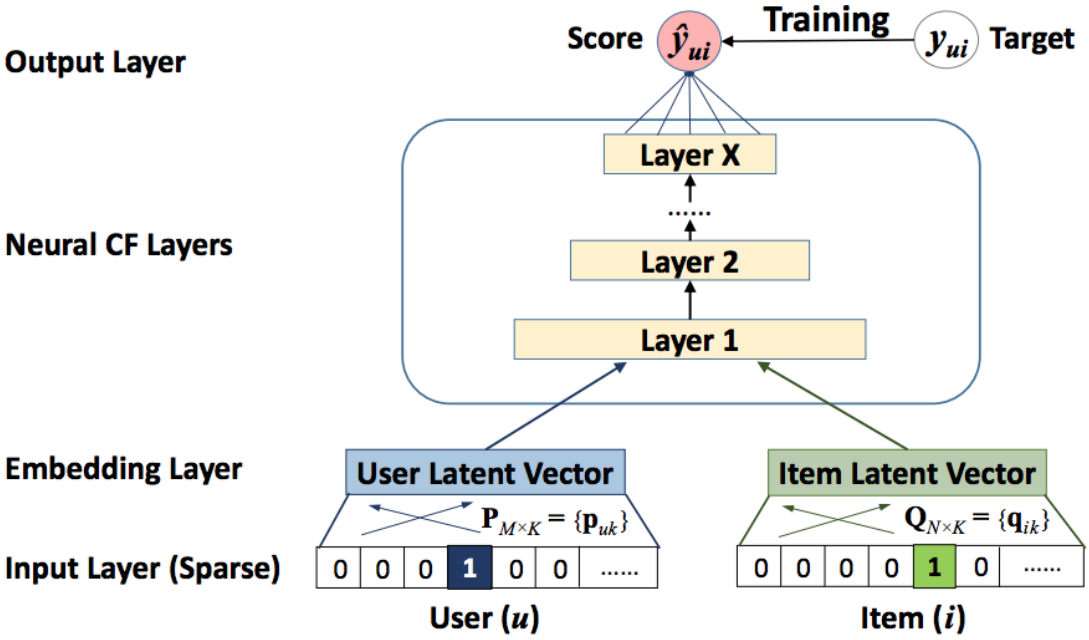
Multilayer Perceptron

Ở phương pháp này thay vì đưa các latent feature vào hàm kích hoạt ngay để ra kết quả dự đoán, ta nối 2 vector rồi đưa vào một feedforward neural network.

$$\mathcal{L}(P,Q) = \left\|Y - f_{W_l,b_l}\left(\dots f_{W_1,b_1}([p,q])\dots\right)\right\|_F \quad (44)$$

Thuật giải 3: Multilayer Perceptron

1. Đọc và làm sạch dữ liệu
2. Khởi tạo P và Q
3. Lặp lại
4. Duyệt từng dòng dữ liệu $l := 0 \rightarrow C$
5. Mã hóa u_l, i_l thành ρ_l và ϱ_l
6. Tính p và q
7. Tính $\mathcal{L}(P,Q) = \left\| Y - f_{W_l,b_l}(\dots f_{W_1,b_1}([p,q]) \dots) \right\|_F$
8. Cập nhật P, Q, W
9. Đến khi $\mathcal{L}(P,Q)$ hội tụ



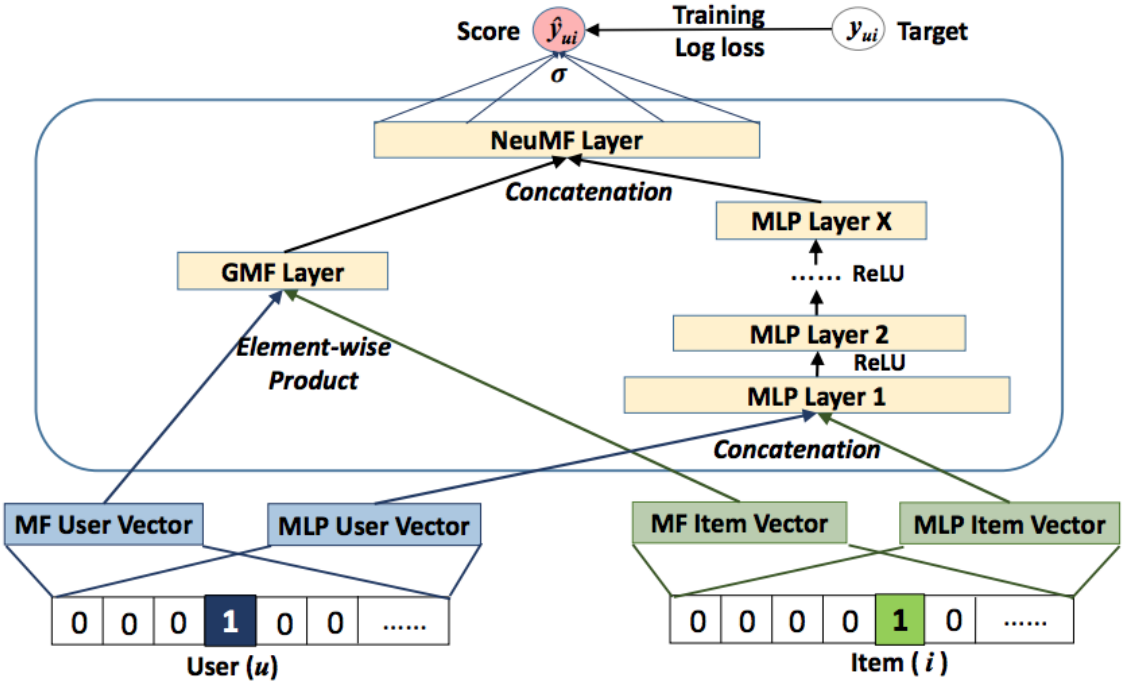
Hình 3.3.2: Mô hình MLP [8].

Neural Matrix Factorization

Tương tự với ý tưởng của phương pháp MLP trên, thì với phương pháp này ta chia 2 latent feature ra làm 2 nửa (có thể đều hoặc không). Và đưa 2 phần làm dữ liệu đầu vào cho hai phương pháp GMF và MLP. Cả 2 latent feature đều sẽ bị chia ra tại điểm j (tôi chọn j = K/3).

$$\mathcal{L}(P,Q) = \left\| Y - \sigma\left(w\left[\phi^{MLP}\left(\left[p_{m,0\to j},\ q_{n,0\to j}\right]\right), \phi^{GMF}\left(\left[p_{m,j+1\to k},\ q_{n,j+1\to k}\right]\right)\right]^T\right)\right\|_F \quad (44)$$

	Thuật giải 3: Neural Matrix Factorization
1.	Đọc và làm sạch dữ liệu
2.	Khởi tạo P và Q
3.	Chọn j để phân chia feature vector.
4.	Lặp lại
5.	Duyệt từng dòng dữ liệu $l := 0 \rightarrow C$
6.	Mã hóa u_l, i_l thành ρ_l và ϱ_l
7.	Tính p và q
8.	Tính $\Delta = Y - \sigma\left(w\left[\phi^{MLP}\left(\left[p_{m,0\to j},\ q_{n,0\to j}\right]\right), \phi^{GMF}\left(\left[p_{m,j+1\to k},\ q_{n,j+1\to k}\right]\right)\right]^T\right)$
9.	Tính $\mathcal{L}(P,Q) = \|\Delta\|_F$
10.	Cập nhật P, Q, W
11.	Đến khi $\mathcal{L}(P,Q)$ hội tụ

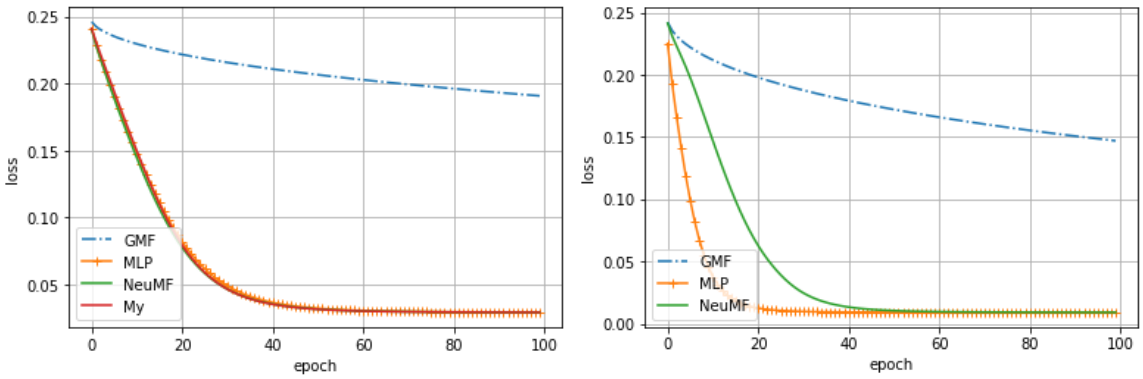


Hình 3.3.3: Mô hình NeuMF [8].

3.4. Kết quả và ứng dụng

Ta huấn luyện mô hình 100 epochs. Sau khi huấn luyện mô hình xong, ta thu được một số chỉ số đánh giá của mô hình bên dưới. Phương pháp được dùng để đánh giá độ chính xác của các mô hình là Mean Squared Error.

Biểu đồ 3.4.1: Quá trình hội tụ của hàm mất mát
khi sử dụng bộ dữ liệu Taobao (phải) và bộ dữ liệu của tôi (trái).



Nhận xét: Nhìn chung, phương pháp GMF có tốc độ hội tụ chậm và độ sai lệch cao hơn những phương pháp khác. Phương pháp MLP tuy có tốc độ hội tụ nhanh nhất nhưng càng về sau thì sai số của phương pháp này so với phương pháp NeuMF không quá lớn.

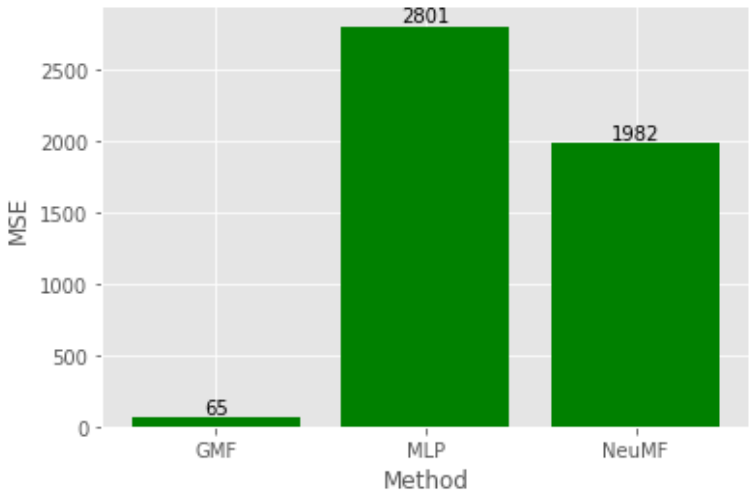
Bảng 3.4.1: Độ sai lệch của mô hình sau khi huấn luyện

	Dữ liệu Taobao	**Dữ liệu của tôi**
GMF	0.1904	0.1469
MLP	0.0281	0.0010
NeuMF	0.0280	0.0010

Đối với Taobao dataset với dữ liệu dồi dào nền ta có thể thấy sự chênh lệch về hiệu suất của 3 phương pháp thực nghiệm, còn với My dataset do dữ liệu còn hạn chế nên tuy hiệu suất vẫn có sự chênh lệch như Taobao dataset nhưng độ lớn của sự chênh lệch này không đáng kể.

Nếu ta huấn luyện với khoảng thời gian lâu hơn (tăng số epoch) thì sai số giữa phương pháp MLP và NeuMF sẽ gần tương đương nhau.

Biểu đồ 3.4.2: Số lượng tham số huấn luyện của 3 phương pháp



Khi quan sát số lượng tham số của từng mô hình (không tính số lượng tham số của 2 ma trận phân rã P và Q vì tất cả mô hình đều dùng 2 ma trận này) ta thấy sự tỉ lệ thuận giữa số lượng tham số, độ phức tạp của mô hình và độ chính xác của mô hình máy học, càng nhiều tham số thì mô hình có độ chính xác cao hơn.

Số lượng tham số của phương pháp NeuMF chỉ bằng 70% so với phương pháp MLP nhưng mang lại hiệu suất xấp xỉ nhau ($\text{loss}_{MLP} \approx 0.0010$ và $\text{loss}_{NeuMF} \approx 0.0010$- Kết quả có thể khác nhau đôi chút khi chạy lại phần đánh giá nhiều lần nhưng sai số không đáng kể).

Ta ứng dụng mô hình đã huấn luyện được vào dữ liệu cầu thủ. Ta thống kê lại dữ liệu đầu vào để có cái nhìn tổng quan về dữ liệu.

name	count
Ruud Gullit - Top Transfer	38
Patrick Vieira - Top Transfer	31
Ronaldo - Heroes Of the Team	28
Ronaldo - Top Transfer	26
Gianluigi Donnarumma - LIVE	24
Cristiano Ronaldo - Heroes Of the Team	22
Theo Hernández - LIVE	22
Cristiano Ronaldo - Top Transfer	20
Patrick Vieira - Nation Hero Debut	19
Thibaut Courtois - LIVE	18

Hình 3.4.1: Xếp hạng 10 cầu thủ được chọn nhiều nhất trong hệ thống

Ta sử dụng mô hình đã huấn luyện dự đoán sư tương tác của người dùng thứ i với tất cả cầu thủ còn lại, sắp xếp lại theo độ lớn. Ưu tiên đề xuất những cầu thủ có khả năng được người dùng i tương tác cao nhất. Trong hình 3.4.2 là trường hợp của người dùng thứ 15.

	name	season
0	Gianluigi Donnarumma	LIVE
1	Raphaël Varane	19 Team of the Year
2	Theo Hernández	LIVE
3	Ronaël PierreGabriel	Golden Rookies
4	Ruud Gullit	Top Transfer
5	Patrick Vieira	Nation Hero Debut
6	Thierry Henry	Top Transfer
7	Kylian Mbappe Lottin	19 Team of the Year
8	Cristiano Ronaldo	19 Team of the Year
9	Ronaldo	Top Transfer
10	Paolo Maldini	Nation Hero Debut

	name	season
293	Ronaldo	Heroes Of the Team
18	Emmanuel Petit	Nation Hero Debut
556	Raphaël Varane	Tournament Best
316	Gary Lineker	Heroes Of the Team
309	Harry Maguire	18 premium live class Spring
835	Kyle Walker	19 Uefa Champions League
84	Heung Min Son	Tournament Best
17	Ronaldo	Top Transfer
334	George Best	Tournament Champions
21	Ruud Gullit	Top Transfer

Hình 3.4.2: Dựa vào đội hình của người dùng thứ 15 (trái)
và kết quả đề xuất tương ứng (phải).

CHƯƠNG 4: KẾT LUẬN VÀ HƯỚNG PHÁT TRIỂN

Qua quá trình thực hiện bài luận văn này tôi đã có các kết luận sau. Phương pháp NeuMF có hiệu quả về mặt hiệu năng và khối lượng tiêu hao tài nguyên máy tính là cao nhất trong 3 phương pháp. MLP là phương pháp có thời gian hội tụ trong khoảng ngắn nhanh nhất. Và phương pháp GMF tuy không mang lại hiệu suất cao bằng hai phương pháp còn lại nhưng có cấu hình gọn nhẹ, thích hợp với các bộ dữ liệu nhỏ và điều kiện phần cứng không quá cao.

Tùy theo nguồn dữ liệu và điều kiện thiết bị có sẵn, ta có thể tùy chọn và áp dụng các mô hình, phương pháp phù hợp.

Hệ thống đề xuất trong bài luận văn này có thể nâng cấp lên thành hệ thống đề xuất đa mục tiêu trong tương lai. Hành vi của người dùng không chỉ còn là thêm vào đội hình mà còn có các hành vi khác như nhấp chuột, tìm kiếm, yêu thích… Những hành vi này có khi động lập cũng có khi phụ thuộc lẫn nhau. Như trước khi chọn cầu thủ người dùng thường tìm kiếm và xem các cầu thủ khác. Ta có thể phá triển hệ thống đề xuất theo hướng này.

Sau bài luận này tôi sẽ áp dụng thử hệ thống vào môi trường thực tiễn học hỏi, rút ra các kinh nghiệm để có thể cải tiến hệ thống ngày một tốt hơn.

CHƯƠNG 5: TÀI LIỆU THAM KHẢO

[1] Tat-Seng Chua, Lina Yao, Yang Song, and Depeng Jin, "Learning to Recommend with Multiple Cascading Behaviors" [Online]. Available: https://arxiv.org/pdf/1809.08161.pdf.

[2] Xiangnan He, Lizi Liao, Hanwang Zhang, Liqiang Nie, Xia Hu, Tat-Seng Chua, "Neural Collaborative Filtering", [Online]. Available: https://arxiv.org/pdf/1708.05031.pdf.

[3] Jure Leskovec, Anand Rajaraman and Jeffrey D. Ullman, "Mining of Massive Datasets" Recommendation Systems, pp. 324, 2012.

[4] Suryakant and Tripti Mahara, "A New Similarity Measure Based on Mean Measure of Divergence for Collaborative Filtering in Sparse Environment", [Online]. Available: https://www.sciencedirect.com/science/article/pii/S1877050916311644.

[5] Juan José Burred, "Detailed derivation of multiplicative update rules for NMF", [Online]. Available: https://www.jjburred.com/research/pdf/jjburred_nmf_updates.pdf.

[6] Sebastian Ruder, "An overview of gradient descent optimization algorithms", [Online]. Available: https://ruder.io/optimizing-gradient-descent.

[7] ML Glossary, "Backpropagation", [Online]. Available: https://ml-cheatsheet.readthedocs.io/en/latest/backpropagation.html#chain-rule-refresher.

[8] Abhishek Sharma, "Neural Collaborative Filtering", [Online]. Available: https://towardsdatascience.com/neural-collaborative-filtering-96cef1009401.