Tổng quan về một số thuật toán dùng trong hệ thống gợi ý

Trình bày:

Nguyễn Văn Sơn

sonnv47@viettel.com.vn

Ha Noi

21/09/2018

Mục lục

1. Giới thiệu chung

- Hai thực thể chính trong Recommendation Systems là *users* và *items*. *Users* là người dùng. *Items* là sản phẩm

- Mục đích chính của các Recommender Systems là dự đoán *mức độ quan tâm* của một *user* tới một *item* nào đó, qua đó có chiến lược *recommend* phù hợp.

1. Giới thiệu chung

- Các Recommendation Systems thường được chia thành 3 dạng chính:

  - *Content-based systems*: đánh giá đặc tính của *items* được *recommended*. Cách tiếp cận này yêu cầu việc sắp xếp các *items* vào từng nhóm hoặc đi tìm các đặc trưng của từng *item*.

  - *Collaborative filtering*: 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

  - *Hybrid Algorithm (giải thuật lai ghép):* sử dụng kết hợp cả hai dạng trên

2. Đặt vấn đề

- Các giải thuật hệ thống gợi ý nói chung đều quy về bài toán Matrix Completion, cụ thể là cần điền vào các vị trí còn thiếu trong Utility Matrix.

	u_0	u_1	u_2	u_3	u_4	u_5	u_6
i_0	5	5	2	0	1	?	?
i_1	3	?	?	0	?	?	?
i_2	?	4	1	?	?	1	2
i_3	2	2	3	4	4	?	4
i_4	2	0	4	?	?	?	5

- *Highly sparse !*
- Utility matrix thường có nhiều dạng:

  0->10 (rating),

  0 or 1 (interactive /non-interactive),

  -1 or 1 (like/dislike)

2. Đặt vấn đề

- Xét một hệ thống gồm:

  N users: $u_1, u_2, \dots, u_N$ và M items: $i_1, i_2, \dots, i_M$

- Ultility matrix Y có kích thước MxN, trong đó:
  - Giá trị phần tử y_{mn} chỉ mức độ quan tâm/tương tác của user n với item m.
  - Những vị trí chưa có giá trị cần được điền vào

- Xét thêm R là ma trận *rated or not* kích thước MxN thể hiện việc một *user* đã *rated* một *item* hay chưa, trong đó $r_{mn} = 1$ tức là user u_n đã rated item i_m, $r_{mn} = 0$ ngược lại.

3. Content-based recommendations
3.1. 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.

	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**

- mỗi items được biểu diễn bởi 2 features
- Việc xây dựng *feature vector* cho mỗi *item* thường bao gồm các kỹ thuật NLP, Generative model
- Ma trận feature-items: $X = \begin{bmatrix} x_1 \\ x_2 \\ \dots \\ x_M \end{bmatrix}$

3. Content-based recommendations
3.2. Xây dựng hàm mất mát

- Trong content-based, mỗi user được xét độc lập, cụ thể với u_n, ta có một bộ dữ liệu: $(x_m, y_{mn})_{m=1...M}$

Ví dụ ở hình bên, ta có các cặp input/label cho user A như sau:

$$(x_1, 5), (x_2, 5), (x_4, 1), (x_5, 1)$$

Xây dựng một mô hình cho mỗi u_n , tương ứng với việc học một vecto trong số w_n để fit bộ dữ liệu trên

→ Là một bài toán hồi quy hoặc phân loại

→ Mô hình tuyến tính: $y = xw + b$

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

3. Content-based recommendations
3.2. Xây dựng hàm mất mát

- Với mỗi user u_n tương ứng với bộ dữ liệu: $(x_m, y_{mn})_{m=1...M}$, xây dựng một mô hình (w_n, b_n) trong đó w_n là vecto trọng số có số chiều bằng số features của items và b_n là bias của mô hình.

  - Khi cần dự đoán rating, đưa bài toán gợi ý về một bài toán hồi quy → linear regression method (Ridge regression, Lasso, Elastic net,...)

  - Khi ultility matrix là 0 or 1 (interactive/non-interactive), hay -1 or 1 (like/dislike), đưa bài toán gợi ý về bài toán phân loại 2 lớp → Logistic regression, SVM, AdaBoost...

3. Content-based recommendations
3.2. Xây dựng hàm mất mát

- Tổng quát hàm mất mát cho user u_n có dạng:

$$L_n = \frac{1}{2} \sum_{m:r_{mn}=1} Loss(\hat{y}_{mn} || y_{mn}) + \frac{\lambda}{2} \|w_n\|_2^2$$

$$= \frac{1}{2} \sum_{m:r_{mn}=1} Loss(x_m w_n + b_n || y_{mn}) + \frac{\lambda}{2} \|w_n\|_2^2$$

  - Khi dự đoán giá trị rating → regression → Loss là mean-squares-error (MSE)
  - Khi dự đoán xác suất tương tác/ yêu thích → classification → Loss là cross-entropy, hinge-loss

3. Content-based recommendations
3.2. Xây dựng hàm mất mát

- Chẳng hạn khi dự đoán giá trị rating, hàm mất mát cho user u_n có dạng:

$$L_n = \frac{1}{2} \sum_{m:r_{mn}=1} (\hat{y}_{mn} - y_{mn})^2 + \frac{\lambda}{2} \|w_n\|_2^2$$

$$= \frac{1}{2} \sum_{m:r_{mn}=1} (x_m w_n + b_n - y_{mn})^2 + \frac{\lambda}{2} \|w_n\|_2^2$$

3. Content-based recommendations
3.2. Xây dựng hàm mất mát

- Trong thực hành, trung bình cộng của *lỗi* thường được dung → mean squares error (MSE)

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

trong đó $s_n = \sum_{m=1}^{M} r_{mn}$ là số lượng các items mà user n đã rated.

- Đặt $\hat{y}_n$ là *sub vector* của y_n chỉ chứa các vị trí có giá trị (đã rated) và $\hat{X}_n$ là *sub matrix* của ma trận feature X, được tạo bằng cách *trích* các hàng tương ứng với các *items* đã được *rated* bởi *user* u_n.

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

→ Đây là hàm mất mát của Ridge regression, sử dụng Stochastic GD hoặc Mini-batch GD để tối ưu hàm này.

3. Content-based recommendations
3.3. Complete utility matrix

- Sau khi học được các mô hình $(w_n, b_n)_{n=1...N}$ cho N user. Giả sử user u_n chưa rate item i_m , thì ta dự đoán rating là: $y_{mn}^{pred} = x_m w_n + b_n$.

- Recommend những item có rating dự đoán cao nhất cho user u_n

4. Neighborhood-based Collaborative filtering
4.1. Giới thiệu

- Các nhược điểm của Content-based recommendations:

  - Không tận dụng được thông tin từ các *users* khác
  - Không phải lúc nào chúng ta cũng có *bản mô tả* cho mỗi *item*

→ Neighborhood-based Collaborative Filtering (NBCF)

4. Neighborhood-based Collaborative filtering
4.1. Giới thiệu

- Ý tưởng của thuật toán:
  - *User-user Collaborative Filtering*: Gợi ý được thực hiện dựa trên các user tương tự. Cụ thể, với một user A cho trước, tìm tập các user **giống với A nhất**, i.e các users thăm các item mà A thăm, sau đó gợi ý cho A những item được thăm bởi tập user tương tự đó (có thể xếp hạng các item theo thứ tự phù hợp).

  - *Item-item Collaborative Filtering*: Gợi ý được thực hiện trên các item tương tự. Cụ thể, từ tập các item mà user A đã thăm trong quá khứ, gợi ý cho A các **item giống với tập các item đó nhất.**

  → 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*.

  → Tập trung vào *User-user Collaborative Filtering*

4. Neighborhood-based Collaborative filtering
4.2. Độ đo tương tự

- Vấn đề đặt ra: giả sử với user-user collaborative filtering
  - 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*?

4. Neighborhood-based Collaborative filtering
4.2. Độ đo tương tự

- Cosin similarity: coi mỗi user như một vector với mỗi chiều tương ứng với một item. Sự tương tự giữa 2 user được tính là Cosin giữa 2 vector tương ứng.

$$cosine - similarity(u_1, u_2) = \cos(u_1, u_2) = \frac{u_1^T u_2}{\|u_1\|_2 . \|u_2\|_2}$$

- Jaccard similarity: độ đo Jaccard giữa hai tập hợp A, B được định nghĩa như sau

$$jaccard-similarity(A,B) = \frac{|A \cap B|}{|A \cup B|}$$

Độ đo Jaccard áp dụng cho hai user cho trước chính là độ tương tự giữa 2 tập item tương ứng mà 2 user đó tương tác.

- Extend Jaccard similarity: với mỗi user u, gọi $P(u) - N(u)$ tương ứng là các items positive (interactive/like/rate)-negative(non-interactive/dislike/not-rate):

$$positive - jaccard - similarity(u, v) = S^+(u, v) = \frac{|P(u) \cap P(v)|}{|P(u) \cup P(v)|}$$

$$negative - jaccard - similarity(u, v) = S^-(u, v) = \frac{|N(u) \cap N(v)|}{|N(u) \cup N(v)|}$$

Khi đó, độ tương đồng của hai user u, v là:

$$S(u, v) = \frac{(|P(u)| + |P(v)|) * S^+(u, v) + (|N(u)| + |N(v)|) * S^-(u, v)}{(|P(u)| + |P(v)|) + (|N(u)| + |N(v)|)}$$

4. Neighborhood-based Collaborative filtering
4.2. Độ đo tương tự

- Độ đo Cosine phù hợp khi utility matrix là có dạng rating (dải giá trị rộng), tuy nhiên với các hệ thống gợi ý, dữ liệu thu thập được dưới dạng 0-1 (inactive/non-inactive, like/dislike), thì dùng độ do Jaccard phù hợp hơn vì tính hợp lý và giảm chi phí tính toán do độ đo Cosine sử dụng căn bậc 2, một thao tác khá tốn kém

- Với độ đo Cosin, để tính được độ tương tự giữa các users (items), cần tạm thời thay các vị trí chưa được điền trong utility matrix bằng một giá trị nào đó → thay bằng trung bình cộng của các *ratings* mà *user* tương ứng đã thực hiện → chuẩn hoá dữ liệu.

4. Neighborhood-based Collaborative filtering
4.3. Rating prediction

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

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

$N(u,i)$ là tập hợp k *users* trong *neighborhood* (tức có *similarity* cao nhất) của user u mà **đã rated** item i

- Trường hợp ultility matrix là 0-1 và độ đo similarity là Jaccard thì công thức trên đơn giản là gợi ý cho user u những những items được tương tác bởi các users similarity nhất với u.

4. Neighborhood-based Collaborative filtering
4.4. Ví dụ user-user CF

	u_0	u_1	u_2	u_3	u_4	u_5	u_6
i_0	5	5	2	0	1	?	?
i_1	4	?	?	0	?	2	?
i_2	?	4	1	?	?	1	1
i_3	2	2	3	4	4	?	4
i_4	2	0	4	?	?	?	5
	↓	↓	↓	↓	↓	↓	↓
$\bar{u}_j$	3.25	2.75	2.5	1.33	2.5	1.5	3.33

a) Original utility matrix $\mathbf{Y}$ and mean user ratings.

	u_0	u_1	u_2	u_3	u_4	u_5	u_6
i_0	1.75	2.25	-0.5	-1.33	-1.5	0	0
i_1	0.75	0	0	-1.33	0	0.5	0
i_2	0	1.25	-1.5	0	0	-0.5	-2.33
i_3	-1.25	-0.75	0.5	2.67	1.5	0	0.67
i_4	-1.25	-2.75	1.5	0	0	0	1.67

b) Normalized utility matrix $\bar{\mathbf{Y}}$.

	u_0	u_1	u_2	u_3	u_4	u_5	u_6
u_0	1	0.83	-0.58	-0.79	-0.82	0.2	-0.38
u_1	0.83	1	-0.87	-0.40	-0.55	-0.23	-0.71
u_2	-0.58	-0.87	1	0.27	0.32	0.47	0.96
u_3	-0.79	-0.40	0.27	1	0.87	-0.29	0.18
u_4	-0.82	-0.55	0.32	0.87	1	0	0.16
u_5	0.2	-0.23	0.47	-0.29	0	1	0.56
u_6	-0.38	-0.71	0.96	0.18	0.16	0.56	1

c) User similarity matrix $\mathbf{S}$.

	u_0	u_1	u_2	u_3	u_4	u_5	u_6
i_0	1.75	2.25	-0.5	-1.33	-1.5	0.18	-0.63
i_1	0.75	0.48	-0.17	-1.33	-1.33	0.5	0.05
i_2	0.91	1.25	-1.5	-1.84	-1.78	-0.5	-2.33
i_3	-1.25	-0.75	0.5	2.67	1.5	0.59	0.67
i_4	-1.25	-2.75	1.5	1.57	1.56	1.59	1.67

d) $\hat{\mathbf{Y}}$

Predict normalized rating of u_1 on i_1 with $k = 2$

Users who rated i_1 : $\{u_0, u_3, u_5\}$

Corresponding similarities: {0.83, -0.40, -0.23}

$\Rightarrow$ most similar users: $\mathcal{N}(u_1, i_1) = \{u_0, u_5\}$

with **normalized ratings {0.75, 0.5}**

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

e) Example

	u_0	u_1	u_2	u_3	u_4	u_5	u_6
i_0	5	5	2	0	1	1.68	2.70
i_1	4	3.23	2.33	0	1.67	2	3.38
i_2	4.15	4	1	-0.5	0.71	1	1
i_3	2	2	3	4	4	2.10	4
i_4	2	0	4	2.9	4.06	3.10	5

f) Full $\mathbf{Y}$

4. Neighborhood-based Collaborative filtering
4.4. Ví dụ item-item CF

	u_0	u_1	u_2	u_3	u_4	u_5	u_6		
i_0	5	5	2	0	1	?	?	→	2.6
i_1	4	?	?	0	?	2	?	→	2
i_2	?	4	1	?	?	1	1	→	1.75
i_3	2	2	3	4	4	?	4	→	3.17
i_4	2	0	4	?	?	?	5	→	2.75

a) Original utility matrix $\mathbf{Y}$ and mean item ratings.

	u_0	u_1	u_2	u_3	u_4	u_5	u_6
i_0	2.4	2.4	-.6	-2.6	-1.6	0	0
i_1	2	0	0	-2	0	0	0
i_2	0	2.25	-0.75	0	0	-0.75	-0.75
i_3	-1.17	-1.17	-0.17	0.83	0.83	0	0.83
i_4	-0.75	-2.75	1.25	0	0	0	2.25

b) Normalized utility matrix $\bar{\mathbf{Y}}$.

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

c) Item similarity matrix $\mathbf{S}$.

	u_0	u_1	u_2	u_3	u_4	u_5	u_6
i_0	2.4	2.4	-.6	-2.6	-1.6	-0.29	-1.52
i_1	2	2.4	-0.6	-2	-1.25	0	-2.25
i_2	2.4	2.25	-0.75	-2.6	-1.20	-0.75	-0.75
i_3	-1.17	-1.17	-0.17	0.83	0.83	0.34	0.83
i_4	-0.75	-2.75	1.25	1.03	1.16	0.65	2.25

d) Normalized utility matrix $\bar{\mathbf{Y}}$.

4. Neighborhood-based Collaborative filtering
4.5. Thảo luận

- Trên thực tế, số lượng *users* luôn lớn hơn số lượng *items* rất nhiều → *Similarity matrix* của user-user rất lớn nhưng của item-item thì nhỏ hơn rất nhiều → lưu trữ, tính toán

- Mỗi user thường rate rất ít items nhưng mỗi *item* có thể được *rated* bởi nhiều *users* → khi có rating mới, giá trị trung bình các cột của ultility matrix biến động nhiều hơn giá trị trung bình các hang → *Similarity matrix* item-item ít cần cập nhật hơn của user-user

4. Neighborhood-based Collaborative filtering
4.5. Thảo luận

- Thời gian inference của thuật toán Neighborhood-based Collaborative filtering là rất lâu, các điểm thắt nút cổ chai nằm ở việc *tính độ tương đồng* và *ranking* để tìm k users (items) tương đồng nhất.

  - Sử dụng độ đo thích hợp: *Cosin, Jaccard, Person Corelation*
  - Tính toán độ đo bằng các cách tối ưu
  - Thuật toán ranking

4. Neighborhood-based Collaborative filtering
4.5. Thảo luận

- **Ranking:** *Google News Personalization: Scalable Online Collaborative Filtering (Das, Abhinandan S., Mayur Datar, Ashutosh Garg, and Shyam Rajaram)* sử dụng độ đo Jaccard và ***min-hash*** function để ***hashing*** những users (items) similarity vào cùng một cụm với xác suất rất cao, độ phức tạp thuật toán O(M+N) → online recommendation.

- **Compute similarity**:
  - *Recommendations for streaming data (Subbian)* sử dụng ***min-hash*** function để ***xấp xỉ*** (với xác suất rất cao) độ đo Jaccard với chi phí (thời gian, bộ nhớ) thấp
    → online recommendation.
  - *TencentRec Real-time Stream Recommendation in Practice* (Yanxiang Huang) sử dụng ***co-rating*** để thực hiện ***scalable incremental update*** → online recommendation.

5. Matrix Factorization

- Ý tưởng:
  - Tồn tại các *latent features* (tính chất ẩn) mô tả sự liên quan giữa các *items* và *users:*
    - Mỗi *item* sẽ mang tính chất ẩn ở một mức độ nào đó tương ứng với các hệ số trong vector x 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 w của nó. Hệ số cao tương ứng với việc *user* thích các bộ phim có tính chất ẩn đó
  - Trong content-based recommendation, các items profile được xây dựng này độc lập với quá trình đi tìm hệ số phù hợp cho mỗi *user*. Việc xây dựng từng mô hình riêng lẻ cho mỗi *user* dẫn đến kết quả chưa thực sự tốt vì không khai thác được đặc điểm của những *users* gần giống nhau.
  - Neighborhood-based Collaborative filtering tận dụng được tính similarity giữa các users (items) nhưng lại có nhược điểm về chi phí tính toán và bộ nhớ.

5. Matrix Factorization

- Ý tưởng:
  - Không cần xây dựng từ trước các *item profile* x mà vector đặc trưng cho mỗi *item* này có thể được huấn luyện đồng thời với mô hình của mỗi *user* (w, b).

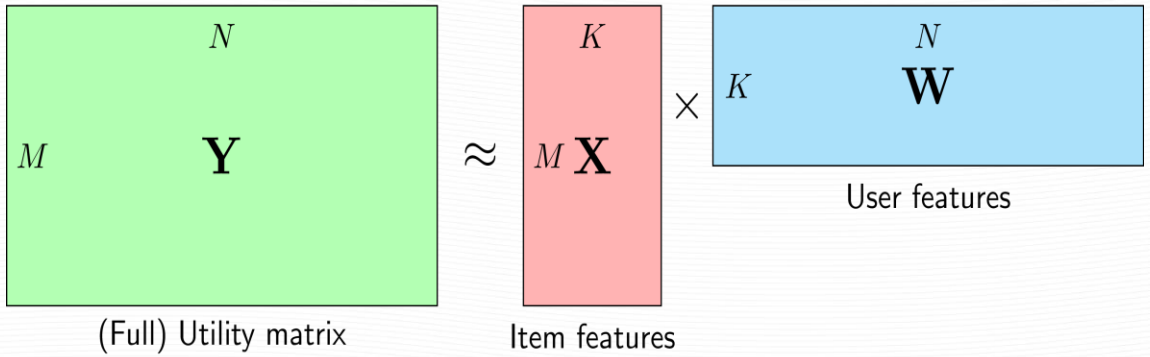
  - Điều này nghĩa là, biến số trong bài toán tối ưu là cả X và W, trong đó:
    - X là ma trận của toàn bộ *item profiles*, mỗi **hàng** tương ứng với 1 *item*
    - W là ma trận của toàn bộ *user models*, mỗi **cột** tương ứng với 1 *user*

5. Matrix Factorization

- Ý tưởng: Về bản chất là xấp xỉ *Utility Matrix* $Y \in R^{M \times N}$ bằng tích của hai ma trận $X \in R^{M \times K}$ và $W \in R^{K \times N}$.

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

$$\mathbf{Y} \approx \hat{\mathbf{Y}} = \mathbf{XW}$$



- Giá trị K ở đây còn được gọi là số latent factor hay số chiều của không gian ẩn.
- Vì $K \ll M, N$ nên còn gọi là Low-Rank Matrix Factorization

5. Matrix Factorization

- Hàm mất mát: tương tự như trong Content-based recommendations, hàm mất mát có dạng:

$$L(X,W) = \frac{1}{2s}\sum_{n=1}^{N}\sum_{m:r_{mn}=1} Loss(\hat{y}_{mn}||y_{mn}) + \frac{\lambda}{2}(\|X\|_F^2 + \|W\|_F^2)$$

$$= \frac{1}{2s}\sum_{n=1}^{N}\sum_{m:r_{mn}=1} Loss(x_m w_n + d_m + b_n||y_{mn}) + \frac{\lambda}{2}(\|X\|_F^2 + \|W\|_F^2)$$

với $\|*\|_F^2$ là Frobineous norm,), s là toàn bộ số ratings đã có, ở đây cần thêm bias d_m cho mỗi item

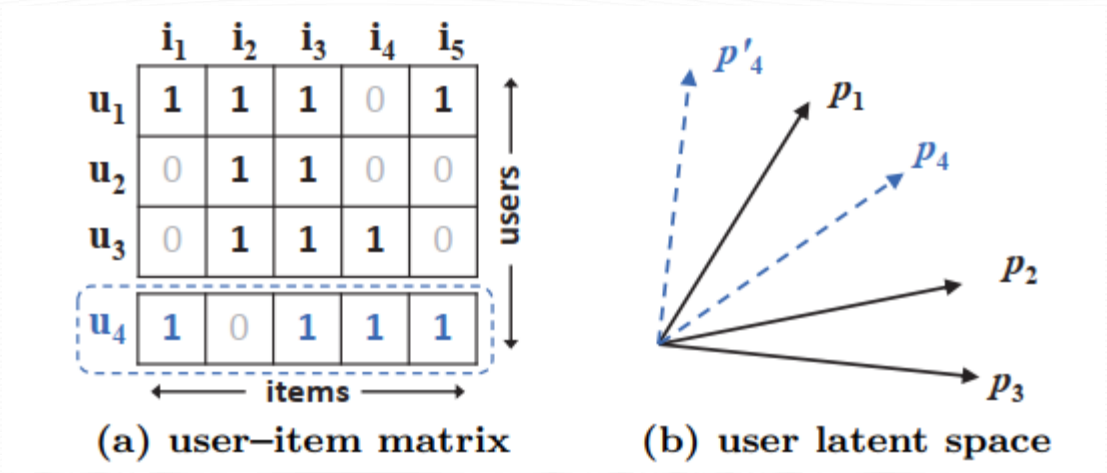
- Về phương pháp tối ưu, với hàm mục tiêu với hai biến, cố định từng biến và thực hiện tối ưu biến còn lại, thực hiện lặp quá trình này cho đến khi hội tụ.

5. Matrix Factorization

- Thảo luận:
  - Bản chất Matrix Factorization cũng là 1 phương pháp Collaborative filtering
  - Matrix Factorization có thời gian inference rất nhanh
  - Thời gian *training* lại khá lâu với các tập dữ liệu lớn. Hai ma trận X và W phải thường xuyên được cập nhật vì có thêm *users*, *items* cũng như các *ratings* mới . Điều này đồng nghĩa với việc ta phải thực hiện quá trình re-training vốn tốn khá nhiều thời gian.
  - Có rất nhiều phương pháp đã được đề xuất để cải thiện tốc độ training cho Matrix Factoiation
    - *Real-Time Top-N Recommendation in Social Streams*
    - *Fast Matrix Factorization for Online Recommendation with Implicit Feedback*
    - *Fast Maximum Margin Matrix Factorization for Collaborative Prediction*
    - *Fast incremental matrix factorization for recommendation with positive-only feedback.*

6. Neural Matrix Factorization

- Hạn chế của original-Matrix factorization:
  - Mục đích của matrix factorization là ánh xạ users và items vào cùng một không gian ẩn (latent space) và cố gắng capture được các tính chất ẩn liên kết giữa users-items
  - Tuy nhiên với hàm inner product không đủ phức tạp để có thể mô hình hóa interaction giữa user và item
  - Ví dụ:



→ Tham số K (*số latent factor*) ?

6. Neural Matrix Factorization

- Bài toán đặt ra:

Thay thế hàm inner product bằng một hàm phức tạp hơn, có khả năng mô hình hóa tốt hơn interaction giữa user và item.

6. Neural Matrix Factorization
6.1. Generalized Matrix Factorization

- Giả sử p_u, q_i ký hiệu là các latent vector của user u và item i (trong các phần trước, $p_u \sim w_n$ và $q_i \sim x_m$)
- Xây dựng một ánh xạ ϕ như sau:

$$\phi(p_u, q_i) = p_u \odot q_i$$

trong đó $\odot$ là element-wise product của các vectors.

- Prediction score có dạng:

$$\hat{y}_{ui} = a_{out}\left(h^T \phi(p_u, q_i)\right) = a_{out}(h^T(p_u \odot q_i))$$

trong đó a_{out} là một activation function và h là một vecto trọng số.

→ Generalized matrix factorization

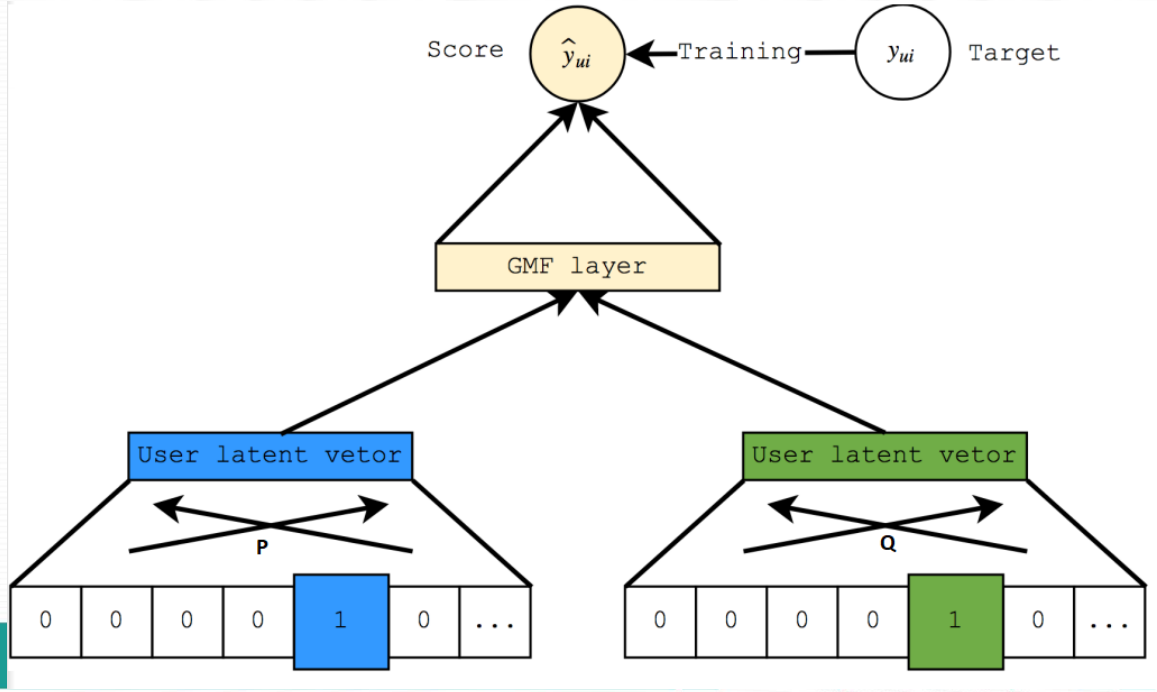
→ với a_{out} là hàm đồng nhất và h là vecto 1 thì ta có $\hat{y}_{ui} = p_u^T q_i$, tức là chúng ta có orginal-Matrix Factorzation

→ Ý tưởng: để capture tốt hơn sự liên kết ẩn giữa users-items, chọn a_{out} là một hàm phi tuyến, chẳng hạn sigmoid, và h là một vecto trọng số cần ***được học*** từ dữ liệu đầu vào (không fix cố định)

6. Neural Matrix Factorization
6.1. Generalized Matrix Factorization

- Mô hình hoá
  - v_u^U, v_i^I là các vecto one-hot coding tương ứng với user u và item I
  - $P \in R^{N \times K}, Q \in R^{M \times K}$ tương ứng là latent factor matrix của các users-items (trong các phần trước thì $P \sim W^T, Q \sim X$) → $p_u = P^T v_u^U$ và $q_i = Q^T v_i^I$

→ Khi đó có thể mô hình hoá GMF (Genralized Matrix Factorization) dưới dạng Neural net với một hidden layer như sau:



6. Neural Matrix Factorization
6.2. Neural Collaborative Filtering

- Ý tưởng: nếu thay 1 hidden layer (GMF layer) trong network trên bằng một deep network gồm nhiều hidden layer, sẽ được một framework, gọi là *Neural collaborative filtering*
- *Prediction score:* $\hat{y}_{ui} = f(\mathbf{P}^T\mathbf{v}_u^U, \mathbf{Q}^T\mathbf{v}_i^I|\mathbf{P}, \mathbf{Q}, \Theta_f)$

$$f(\mathbf{P}^T\mathbf{v}_u^U, \mathbf{Q}^T\mathbf{v}_i^I) = \phi_{out}(\phi_X(...\phi_2(\phi_1(\mathbf{P}^T\mathbf{v}_u^U, \mathbf{Q}^T\mathbf{v}_i^I))...))$$

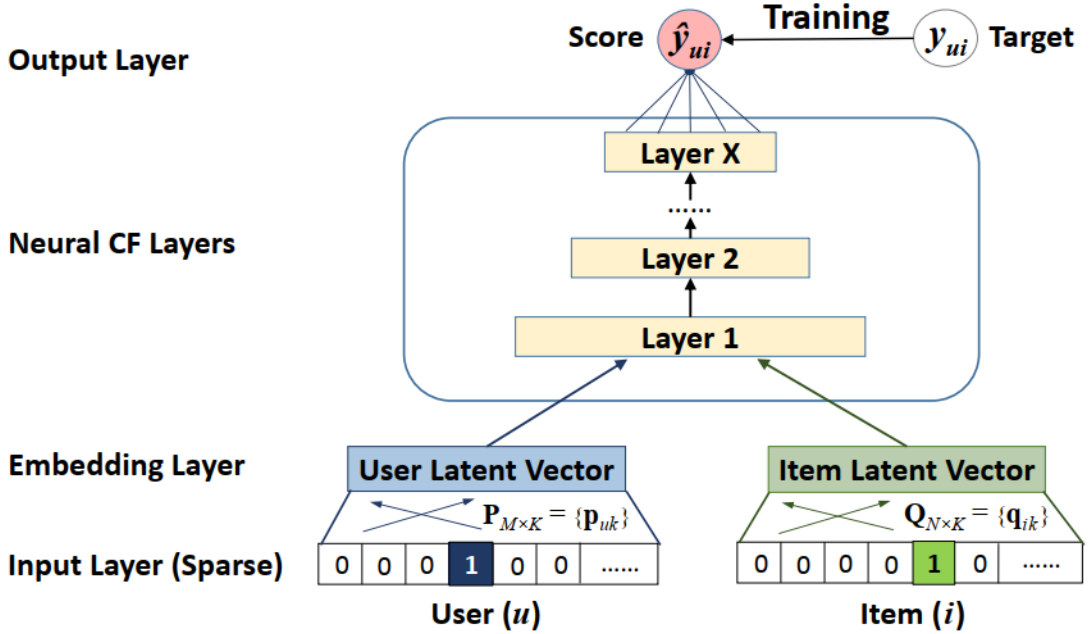


Figure : Neural collaborative filtering framework

6. Neural Matrix Factorization
6.2. Neural Collaborative Filtering

- Với deep network trong mô hình, chọn kiến trúc Multi-Layer Perceptron, với activation function ReLU (hoặc sigmoid, tanh…).

$$\mathbf{z}_1 = \phi_1(\mathbf{p}_u, \mathbf{q}_i) = \begin{bmatrix} \mathbf{p}_u \\ \mathbf{q}_i \end{bmatrix},$$

$$\phi_2(\mathbf{z}_1) = a_2(\mathbf{W}_2^T \mathbf{z}_1 + \mathbf{b}_2),$$

$$\ldots\ldots$$

$$\phi_L(\mathbf{z}_{L-1}) = a_L(\mathbf{W}_L^T \mathbf{z}_{L-1} + \mathbf{b}_L),$$

$$\hat{y}_{ui} = \sigma(\mathbf{h}^T \phi_L(\mathbf{z}_{L-1})),$$

trong đó W_k, b_k, a_k tương ứng là vecto trong số, bias và activation function của layer thứ k

6. Neural Matrix Factorization
6.3. Neural Matrix Factorization

- Ý tưởng: Kết hợp Generalized Matrix Factorization và Neural Collaborative Filtering → Neural Matrix Factorization

- GMF và NCF cùng chung nhau embeding layer → hạn chế do data có thể có "optimal embeding size" trong GMF và NCF chênh lệch nhiều.

→ tổ hợp 2 model tại hidden layer cuối cùng.

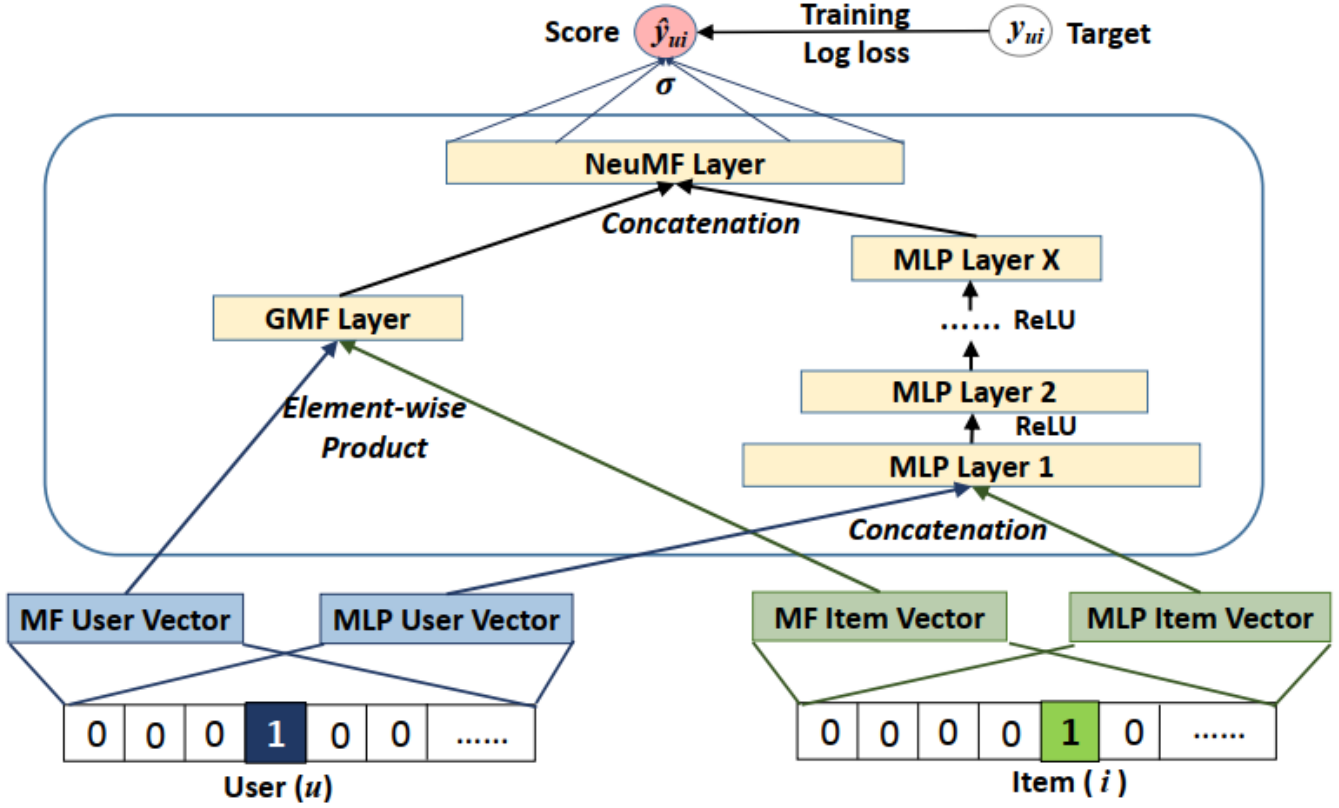


Figure : Neural matrix factorization model

6. Neural Matrix Factorization
6.3. Neural Matrix Factorization

$$\phi^{GMF} = \mathbf{p}_u^G \odot \mathbf{q}_i^G,$$

$$\phi^{MLP} = a_L(\mathbf{W}_L^T(a_{L-1}(...a_2(\mathbf{W}_2^T \begin{bmatrix} \mathbf{p}_u^M \\ \mathbf{q}_i^M \end{bmatrix} + \mathbf{b}_2)...)) + \mathbf{b}_L),$$

$$\hat{y}_{ui} = \sigma(\mathbf{h}^T \begin{bmatrix} \phi^{GMF} \\ \phi^{MLP} \end{bmatrix}),$$

- h là một tham số cần học
- Tuy nhiên loss function là non-convexity

→ pre-training → h^{GMF}, h^{NCF}

→ Khởi tạo: $h = \begin{bmatrix} \alpha h^{GMF} \\ (1-\alpha) h^{NCF} \end{bmatrix}$

Với α là hệ số trade-off

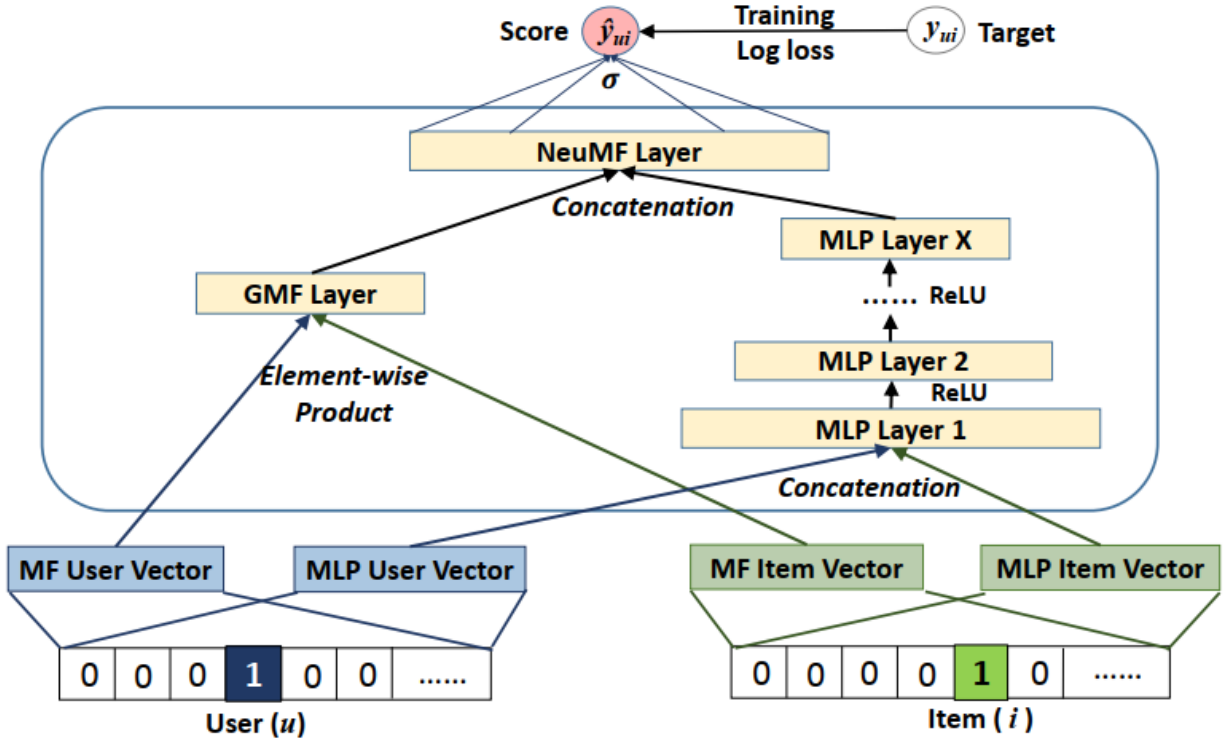


Figure : Neural matrix factorization model

Tài liệu tham khảo

- *Machine learning cơ bản* (Vũ Hữu Tiệp)
- *Neural Collaborative Filtering* (Xiangnan He, Lizi Liao, Hanwang Zhang, Liqiang Nie, Xia Hu, Tat-Seng Chua)
- *Google News Personalization: Scalable Online Collaborative Filtering* (Das, Abhinandan S., Mayur Datar, Ashutosh Garg, and Shyam Rajaram)
- *Recommendations for streaming data* (Subbian)
- *Real-Time Top-N Recommendation in Social Streams* (Ernesto Diaz-Aviles)
- *TencentRec Real-time Stream Recommendation in Practice* (Yanxiang Huang)

CÁM ƠN ĐÃ CHÚ Ý LẮNG NGHE !!!

Đề xuất hướng tiếp cận cho bài toán gợi ý gói khuyến mại thuê bao

- ...

Đề xuất hướng tiếp cận cho bài toán gợi ý gói khuyến mại thuê bao

Thêm feature vector là thuộc tính tiêu dùng của user