Báo cáo bài tập lớn

Nhập Môn học máy và khai phá dữ liệu

Hệ gợi ý Phim

Giảng viên hướng dẫn : TS. Nguyễn Nhật Quang

Danh sách thành viên :

Vũ Tuấn Trường - 20183649

Nguyễn Minh Tuấn - 20183657

Nguyễn Xuân Tùng - 20183662

Vũ Hoàng Trung - 20183645

I. Giới thiệu và mô tả về bài toán thực tế được giải quyết

Chắc hẳn, các bạn đã từng thấy những điều này:

- Bạn mới sử dụng facebook và loay hoay chưa biết bạn bè mình sử dụng tài khoản là gì để kết bạn. Và sau vài hôm, facebook tự động gợi ý cho bạn những người bạn quen biết.
- Bạn mua sắm trên shopee với mục đích ban đầu là một chiếc quần jean nhưng sau đó bạn được gợi ý thêm một loạt giày dép, balo, túi xách,… và sau một hồi thì bạn cuốn theo chúng luôn.
- Bạn xem phim trên Youtube hay Netflix và sau khi xem xong một bộ phim thì Youtube hay Netflix tự gợi ý sang các bộ phim khác có liên quan và bạn cảm thấy hứng thú với chúng.

Những điều trên đều có một điểm chung đó là hệ thống có khả năng tự tìm kiếm và gợi ý những thứ mà có thể chúng ta thích nhưng chưa nghĩ đến, hoặc chưa biết đến nó trong thời điểm hiện tại. Đó chính là công việc của Hệ gợi ý, bằng cách quảng cáo hướng đúng đối tượng như thế này, hiệu quả của việc marketing cũng sẽ tăng lên.

Nghiên cứu và ứng dụng hệ gợi ý giúp nâng cao tính thông minh của ứng dụng, cụ thể ở bài toán này là trong hàng ngàn bộ phim, với một lượng người dùng nhất định thì cần phải đưa ra những gợi ý về các bộ phim phù hợp với từng người dùng.

Yêu cầu đặt ra xây dựng một hệ gợi ý phim (movie recommender system) giải quyết bài toán bằng cách khai phá dữ liệu, xử lý chúng và đưa ra thông tin phim cho người dùng một cách hiệu quả nhất trong hàng ngàn, hàng vạn bộ phim.

II. Các phương pháp sử dụng và tập dữ liệu sử dụng

1. Bộ dữ liệu

MovieLens 20M Dataset, kích thước 190 MB

Bộ dữ liệu này có 20 triệu đánh giá phim (từ 0 đến 5 sao, làm tròn đến 0,5) từ 138.000 người dùng đánh giá 37.000 phim, và còn có các thẻ tag cho mỗi bộ phim (tổng cộng có 465.000 tag).

Sử dụng tập con của bộ dữ liệu trên gồm 1000 người dùng, 8621 phim với 150.000 đánh giá.

Link: MovieLens 20M (Bộ dữ liệu gốc)

Movielens20m-subset (Tập con của bộ dữ liệu được nhóm sử dụng)

2. Phương pháp 1: Lọc công tác (Collaborative filtering)

a. Ý tưởng

Ý tưởng cơ bản của lọc cộng tác là dựa trên thông tin xem phim của người dùng, trong quá khứ nếu nếu hai người dùng cùng thích xem những bộ phim giống nhau (tương tự nhau) thì trong tương lai, hai người họ khả năng cao sẽ cùng thích xem nhưng bộ phim giống nhau.

Ví dụ: người dùng 1 đã xem và thích phim A, B, G, K, P. Người dùng 2 thích xem phim A, B, G, K thì khả năng cao người dùng 2 cũng thích xem P.

b. Cách thực hiện

Xây dựng ma trận user – item từ tập dữ liệu

Ví dụ: (các đánh giá chỉ từ 0 đến 5 sao và làm tròn đến 0,5)

User \ Item	I1	I2	I3	I4	I5	I6	I7	

U1	2	?	5	3.5	?	4.5	1	3.2
U2	1.5	?	5	?	4	5	?	3.875
U3	4	3	1	?	1	?	?	2.25
U4	?	3	?	5	?	?	4	4

Các bộ phim mà người dùng chưa đánh giá (rate), chúng tôi để là dấu "?"

Với phương pháp lọc cộng tác, thì ta cần tính độ tương quan giữa mỗi người dùng với các người dùng còn lại, thông qua các ma trận user-item. Tuy nhiên do có rất nhiều rating còn trống trong ma trận nên ta cần phải điền một giá trị ô đó sao cho không làm ảnh hưởng lớn để độ tương quan giữa các người dùng.

Với vấn đề này, chúng tôi chọn cách điền giá trị trung bình cộng các đánh giá của mỗi người dùng (giá trị đỏ).

User \ Item	I1	I2	I3	I4	I5	I6	I7
U1	2	3.2	5	3.5	3.2	4.5	1
U2	1.5	3.875	5	3.875	4	5	3.875
U3	4	3	1	2.25	1	2.25	2.25
U4	4	3	4	5	4	4	4

Sau đó, trước khi tính độ tương quan giữa các người dùng, chúng tôi làm thêm một bước nữa là trừ các rating của mỗi người dùng cho giá trị trung bình. Việc trừ đi trung bình cộng 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 ma trận là rất lớn với hàng triệu users và items, nếu lưu toàn bộ các giá trị này trong một ma trận thì khả năng xảy ra hiện tượng không đủ bộ nhớ là rất cao. Thông thường mỗi người dùng sẽ chỉ đánh giá một lượng nhỏ các phim nên phần lớn trong ma trận sẽ là các giá trị Null, 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 ma trận tương quan sau này cũng hiệu quả hơn.

User \ Item	I1	I2	I3	I4	I5	I6	I7
U1	-1.2	0	1.8	0.3	0	1.3	-2.2
U2	-2.375	0	1.125	0	0.125	1.125	0
U3	1.75	0.75	-1.25	0	-1.25	0	0
U4	0	-1	0	1	0	0	0

Vậy là đã chuẩn hóa xong dữ liệu, ta cần chọn hàm tương quan để tính độ tương quan giữa các user. Với vấn đề này, chúng tôi dùng hàm cosine_similarity:

$$\text{cosine\_similarity}(\mathbf{u}_1, \mathbf{u}_2) = \cos(\mathbf{u}_1, \mathbf{u}_2) = \frac{\mathbf{u}_1^T \mathbf{u}_2}{||\mathbf{u}_1||_2 . ||\mathbf{u}_2||_2}$$

u_1 và u_2 là vector

Sau khi tính độ tương quan giữa các user thông qua các rating, ta thu được ma trận tương quan như sau:

User \ Item	U1	U2	U3	U4
U1	1	0.72	-0.54	0.07
U2	0.72	1	-0.77	0
U3	-0.54	-0.77	1	-0.2
U4	0.07	0	-0.2	1

Ta thấy kích thước ma trận tương quan chỉ phụ thuộc vào số lượng người dùng, nên khi số lượng người dùng quá lớn thì kích thước ma trận cũng rất lớn, dẫn đến hiện tượng không đủ bộ nhớ. Ngay cả khi ma trận tương quan là ma trận đối xứng và đường chéo là 1 thì việc trữ dữ liệu cũng khó khăn. Với vấn đề này, việc quan tâm tới một user là cần thiết và khi đó, ta chỉ cần độ tương quan của user đó với các user còn lại (chỉ 1 hàng dữ liệu).

Sau khi có được ma trận thể hiện độ tương quan giữa các người dùng, công việc tiếp theo của mô hình là xác định mức độ quan tâm của một user với một item (thông qua các user gần nhất). Tất nhiên là ta chỉ quan tâm tới các user đã rated item chúng ta đang xét. Và với vấn đề này, chúng tôi sử dụng công thức sau:

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

Dưới mẫu số là trị tuyệt đối để xử lý các số âm

- u và i lần lượt là user và item đang xét.
- **N**(u,i) là tập hợp k users trong neighborhood (tức có similarity cao nhất) của u mà đã rated i.
- $Y_{i,uj}$ là rating mà người dùng U_j đã rate cho item i

Tất nhiên là số hàng xóm (users trong neighborhood) là do ta chọn, ví dụ

với k = 2 hàng xóm, ta dự đoán U_4 cho I_3:

User \ Item	I1	I2	I3	I4	I5	I6	I7
U1	-1.2	0	1.8	0.3	0	1.3	-2.2
U2	-2.375	0	1.125	0	0.125	1.125	0
U3	1.75	0.75	-1.25	0	-1.25	0	0
U4	0	-1	0	1	0	0	0

Theo ma trận tương quan thì 2 hàng xóm gần nhất của U_4 là U_1 và U_2 với độ tương quan lần lượt là 0,07 và 0 (tất nhiên U_1 và U_2 đã rated cho I_3)

$$\Rightarrow Y_{i3,u4} = \frac{0.07*1.8+0*1.125}{|0.07+0|} = 1.8$$

Ta có kết quả là 1.8 rồi ta chỉ cần cộng ngược lại với giá trị trung bình đánh giá của user U_4 đã trừ đi lúc trước là chúng ta đã có kết quả dự đoán.

Nhận xét:

- Với mô hình user user này thì có hạn chế là do số lượng user thường là rất rất lớn và thường xuyên tăng lên, dẫn đến việc lưu trữ và tính toán ma trận tương quan càng khó khăn.
- Ngoài ra, Ma trận rating giữa user và item thường là rất thưa. Với số lượng users rất lớn so với số lượng items thì mỗi hàng sẽ chỉ có một vài giá trị khác Null. 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 hóa 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.

Ta có một cách tiếp cận khác là item-item, ta tính toán độ tương quan giữa các items rồi gợi ý 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 ma trận là như nhau nhưng số item ít hơn user , nên trung bình, mỗi cột của ma trận này sẽ có nhiều phần tử đã biết hơn. 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.

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.

3. Phương pháp 2: Lọc theo nội dung (content-based)

a. Ý tưởng

Ý tưởng dựa trên việc tìm mối liên quan giữa các phim dựa vào nội dung của phim, sau đó gợi ý những bộ phim có nội dung tương tự với những bộ phim mà người dùng đã thích. Ví dụ: người xem thích xem phim “Cảnh sát hình sự” thì chúng ta sẽ gợi ý cho người dùng phim “Người phán xử”.

b. Thực hiện

Vì hệ thống lọc theo nội dung (content-based) phụ thuộc chủ yếu vào nội dung của phim(item) nên việc quan trọng nhất của phương pháp này là biểu diễn từng bộ phim thành những vector dựa vào nội dung của nó.

Dựa vào tập dữ liệu với những nội dung phim đã được cho trước(thể hiện dưới các tag) chúng tôi thực hiện 3 cách chuyển những nội dung này thành vector, dó là one-hot, word-vector và kết hợp one-hot, word-vector.

- *One-hot*

Đây là phương pháp đơn giản nhất để chuyển nội dung phim thành vector, chúng ta xây dựng một vector cho phim chỉ chứa toàn phần tử 0 và 1. Phần tử 1 ứng với những nội dung mà phim được gắn tag, còn phần tử 0 là những nội dung phim không được gắn tag.

Ví dụ:

Phim/Tag	Khoa học-viễn tưởng	Hành động	Lịch sử	Hoạt hình
Avenger	1	1	0	0
Doraemon	0	0	0	1
Điện Biên Phủ	0	1	1	0

Ở đây chúng ta sẽ có 3 vector tương ứng với 3 phim:

Avenger: (1,1,0,0)

Doraemon:(0,0,0,1)

Điện Biên Phủ:(0,1,1,0)

- *Word-vector:*

Chúng tôi nhận thấy rằng việc chỉ sử dụng one-hot vector để biểu diễn các phim là chưa thể hiện được đầy đủ tính chất của một bộ phim, ví dụ 2 bộ phim cùng thể loại là "Hành động" như một bộ phim về siêu anh hùng và một bộ phim về chiến tranh chắc chắn phải có cách biểu diễn khác nhau. Do đó chúng tôi áp dụng cách biểu diễn word-vector cho mỗi bộ phim.

Chúng tôi sử dụng các tag được gán cho mỗi bộ phim được cung cấp sẵn trong bộ dữ liệu MovieLens20M với độ liên quan đến từng phim. Ví dụ một bộ phim siêu anh hùng sẽ có độ liên quan đến tag "superhero" là 0.99 trong khi đó một bộ phim về chiến tranh sẽ có độ liên quan đến tag "war" là 0.98. Đối với mỗi bộ phim, chúng tôi sẽ lựa chọn ra 10 tag có độ liên quan đến phim cao nhất. Sau đó sử dụng một mô hình word2vec (ở đây chúng tôi sử dụng pretrained model glove-100 có sẵn trong thư viện gensim, biểu diễn mỗi từ thành 1 vector 100 chiều) để biến đổi các tag này thành các vector và nhân với độ liên quan. Sau đó các vector này sẽ được kết hợp lại thành 1 vector như sau: giá trị lớn nhất theo từng chiều, giá trị nhỏ nhất theo từng chiều, giá trị trung bình theo từng chiều. Cuối cùng chúng ta sẽ được 1 biểu diễn vector 300 chiều cho mỗi bộ phim.

- *Kết hợp Word-vector và One-hot*

Lúc này chúng tôi sử dụng vector biểu diễn phim là kết hợp của 2 phương pháp trên, với 20 phần tử đầu là biểu diễn one-hot và 300 phần tử sau là biểu diễn word-vector

Sau khi đã có biểu diễn dưới dạng vector của từng bộ phim, chúng ta sẽ tìm model phù hợp cho từng user. Ví dụ:

	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**

Ở đây các phim được biểu diễn dưới dạng vector 2 chiều, A,B,C,D,E,F là các người dùng và giá trị ở mỗi hàng là rating của người dùng cho phim, có thể thấy còn rất nhiều ô chưa có giá trị, đó là do người dùng chưa đánh giá phim này. Giờ công việc của chúng ta là tìm model θi thích hợp cho người dùng. Ở đây chúng tôi sử dụng nhiều model khác nhau để đánh giá và kết quả sẽ được thể hiện ở phần dưới

Xây dựng hàm mất mát

Ở đây chúng tôi sử dụng hàm mất mát RMSE để đánh giá mô hình.

$$\text{MSE} = \frac{1}{n}\sum_{i=1}^{n}(Y_i - \hat{Y}_i)^2$$

với: n là số phim người dùng đã đánh giá

Y là đánh giá của người dùng được mô hình dự đoán ra

$\hat{Y}$ là đánh giá thật của người dùng

Nhận xét:

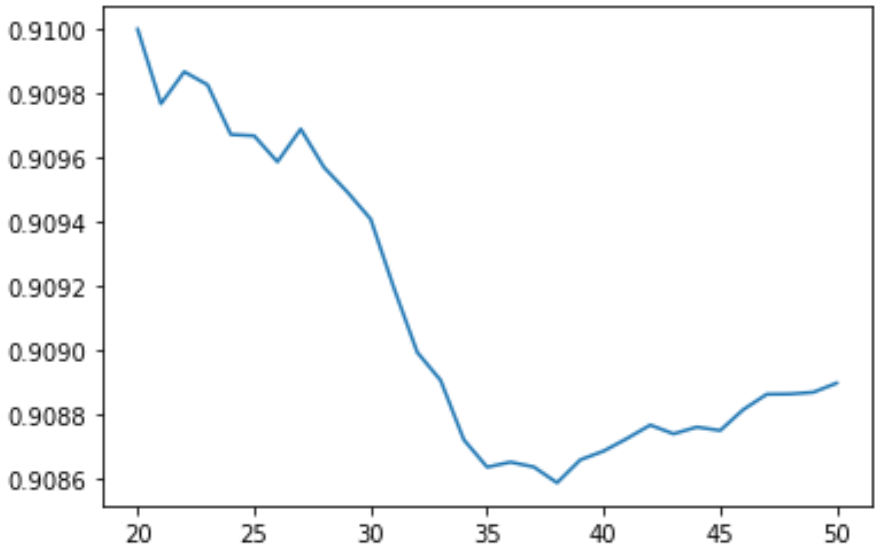
- Content-based Recommendation Systems là phương pháp đơn giản nhất trong các hệ thống Recommendation Systems. Đặc điểm của phương pháp này là việc xây dựng mô hình cho mỗi *user* không phụ thuộc vào các *users* khác.
- Việc xây dựng mô hình cho mỗi *user* có thể được coi như bài toán Regression hoặc Classification với training data là cặp dữ liệu (*item profile*, *rating*) mà *user* đó đã *rated*. *item profile* không phụ thuộc vào *user*, nó thường phụ thuộc vào các đặc điểm mô tả của *item* hoặc cũng có thể được xác định bằng cách yêu cầu người dùng gắn *tag*.

III. Kết quả thí nghiệm

1. Lọc cộng tác (Collaborative filtering)

a) Mô hình user-user

Sau khi train mô hình với tập dữ liệu Train-data, chúng tôi đã thử tính RMSE đối với tập Test-data, do số hàng xóm k (neighborhood) là do chúng ta tự chọn nên chúng tôi cho chạy k tăng dần từ 1 -> 50 và thu được bảng số liệu như sau:



Hình 1: Bảng số liệu user-user với k hàng xóm

Do số hàng xóm từ 1 -> 19 lớn hơn nên chúng tôi không đưa vào và chúng tôi đã thử với số hàng xóm lớn hơn 50 nhưng cũng không cho kết quả khả quan hơn.

Theo bảng số liệu thì với tập dữ liệu train-data và test-data thì mô hình thu được đạt giá trị tốt nhất với số hàng xóm là khoảng 38 với giá trị RMSE là xấp xỉ 0.9086 , tức là trung bình mô hình đưa ra dự đoán sai gần 1 sao so với rating user dự đoán.

Để tìm hiểu sâu hơn tại sao lại có sai số có vẻ lớn như vậy, chúng tôi tính và tìm ra 1% các dự đoán sai nhiều nhất,và kết quả thu được là sai số trải dài từ

khoảng 7,0 -> 30 (sai số này chưa lấy căn) tuy nhiên trong số 1% này thì phần lớn các sai số chỉ không quá 10.

Nhưng với các sai số lớn hơn 7 thì khi lấy căn bậc 2 thì trung bình các dự đoán của chúng ta đối với giá trị rating mà user rate cũng đã khá lớn (gần 3 sao), để tìm hiểu tại sao lại có sai số lớn như vậy, từ 1% các sai số lớn nhất đó, chúng tôi truy vết ngược lại ra các user và item, và thu được kết quả là:

User_ID	Số lần xuất hiện	Item_ID	Số lần xuất hiện
741	14	2346	3
11	6	2405	3
775	5	269	3
31	5	1287	2
471	5	1014	2
856	4	633	2
422	4	67	2
394	4	3374	2
892	4	1758	2

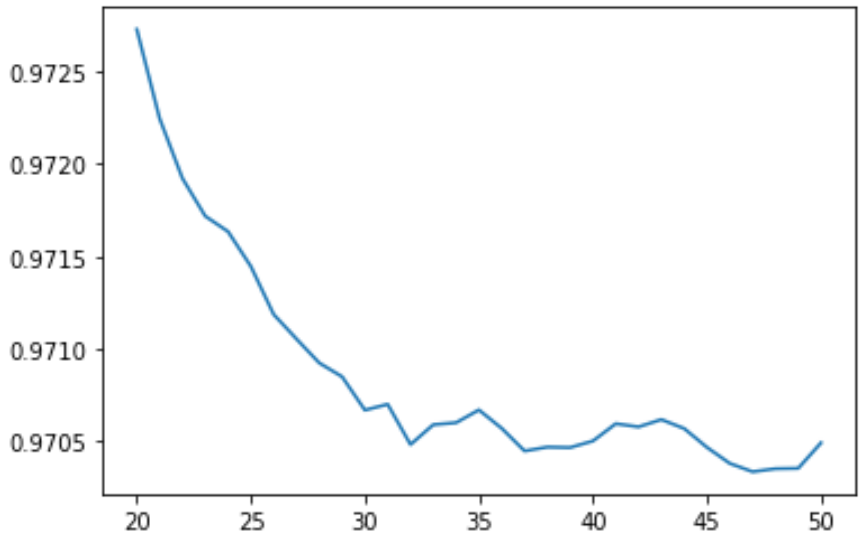
Thông tin về User có ID là 741: đã đánh giá 1603 bộ phim trong tập train-data và có 376 rating trong tập test-data -> nhận xét là user này có nhiều rating trong tập test (sai số 14/376 là chưa đến 4%)

Thông tin về User có ID là 11: đã đánh giá 388 bộ phim trong tập train-data và có 107 rating trong tập test-data -> nhận xét là user này đã rated ít phim nhưng trong tập test-data lại có nhiều rating.

Số lần xuất hiện của các Item dàn trải đề.

b) Mô hình item-item

Tương tự như mô hình user-user, với mô hình item-item chúng tôi cũng thử với số lượng hàng xóm tăng dần từ 1 -> 50 và thu được kết quả:



Hình 2: Bảng số liệu item-item với k hàng xóm

Theo bảng số liệu thì với tập dữ liệu train-data và test-data thì mô hình thu được đạt giá trị tốt nhất với số hàng xóm là khoảng 47 với giá trị RMSE là xấp xỉ 0.9705 , tức là trung bình mô hình này cũng đưa ra dự đoán sai gần 1 sao so với rating user dự đoán.

Chúng tôi cũng tìm ngược lại 1% sai số lớn nhất và thu được kết quả là 1% các sai số này nằm trong khoảng từ 10 -> 25 và các sai số dàn trải khá đều. Chúng tôi cũng truy vết ngược lại các user và item và tất nhiên cũng thu được bảng số liệu y như mô hình user-user.

2. Loc theo nôi dung (Content-based)

Sau khi sử dụng 4 model là Linear Regression, Ride regression, SVR và Kernel Ridge, chúng tôi lưu được những kết quả tốt nhất ứng với từng model và với từng phương pháp biểu diễn vector như sau.

Model	onehot	wordvector	onehot + wordvector
Linear regression	1.103	3.506	rất lớn
Ridge regression	0.940	0.929	0.926
SVR	0.956	0.910	0.901
Kernel ridge	2.257	0.884	0.879

Chúng tôi nhận thấy với cách biểu diễn vector càng phức tạp thì những mô hình đơn giản thường rất tệ và có độ sai cao, tuy nhiên với những mô hình phức tạp hơn thì những cách biểu diễn vector phức tạp lại đem lại kết quả rất tốt.

3. Kết hợp cả 2 mô hình

Chúng tôi sau khi xem xét các đánh giá bị dự đoán sai nhiều nhất trên hai mô hình và đưa ra giả thuyết rằng: các mô hình collaborative filtering (user-user) dễ bị sai trên những người dùng đánh giá nhiều phim hoặc có cách đánh giá khác với những người dùng khác và mô hình content-based recommendation có dự đoán tốt hơn trên những người dùng này.

Lấy trung bình kết quả dự đoán của 2 mô hình trên, ta được kết quả tốt hơn (RMSE trên tập test: 0.855, lấy trung bình của mô hình collaborative filtering

tốt nhất (RMSE: 0.908) và mô hình content-based tốt nhất (RMSE: 0.879)). Do đó chúng tôi chắc chắn việc kết hợp 2 mô hình trên là hướng đi đúng đắn. Chúng tôi tin rằng sẽ có cách kết hợp 2 mô hình hiệu quả hơn, và sẽ để việc đó cho phát triển trong tương lai.

IV. Các chức năng chính của hệ thống

- Huấn luyện 2 mô hình tương ứng với 2 phương pháp collaborative filtering và content-based recommendation.
- Sử dụng các mô hình đã huấn luyện để đưa ra gợi ý cho người dùng bất kỳ trong hệ thống.

V. Các vấn đề/khó khăn gặp phải trong quá trình thực hiện công việc của đồ án, và cách thức giải quyết (khắc phục)

1. Hiện tượng Cold-Start

Hiện tượng Cold-Start là hiện tượng user mới tương tác với hệ thống, user này trước đây user này chưa từng rate cho bất kỳ bộ phim nào.

Cụ thể hơn một chút tức là, user mới này không có ID trong tập train_data mà chỉ có trong tập test_data.

Khi gặp vấn đề này, chúng tôi đưa ra một giải pháp là gợi ý cho user mới này một vài bộ phim được đã được rate bởi nhiều user khác với số rating trung bình ở mức khá (ví dụ: là các bộ phim được rate trên 30 lần và rating trung bình là 4 sao)

Sâu hơn là chúng ta có một bộ phim mới và cần gợi ý phim đó cho người dùng. Với vấn đề này thì chúng tôi đưa ra giải pháp là sử dụng mô hình lọc theo nội dung (content-based), nếu chưa có các thuộc tính nội dung của phim thì chúng tôi sẽ gợi ý phim mới này cho các user đã rate một lượng lớn phim và rating trung bình ở mức khá.

Chú ý: Chúng tôi mới chỉ đưa ra giải pháp nhưng chưa code

VI. Các tranh luận/khám phá/kết luận, và các đề cử cho việc tiếp tục phát triển và cải tiến trong tương lai

1. Các tranh luận/khám phá/kết luận

- ❏ Đối với mô hình trên, chúng tôi xây dựng chúng chưa có khả năng mở rộng, tức là khi có thêm người dùng mới, item mới (ngoài tập train-data) thì hệ thống phải huấn luyện lại mô hình từ đầu.
- ❏ Khám phá ra hiện tượng Cold Start thông qua một vài bộ dữ liệu khi test
- ❏ Bộ dữ liệu chúng tôi sử dụng có thể chưa đầy đủ và đa dạng, nên kết quả chúng tôi thí nghiệm ra chưa sát sao với thực tế

2. Đề cử phương án phát triển trong tương lai

- Hoàn thiện và xử lý hiện tượng Cold Start đã đề cập ở trên
- Kết hợp hai mô hình lọc cộng tác và lọc theo nội dung theo hướng khác để có được kết quả tốt hơn.

References:

1. Chúng tôi có theo dõi và bám theo Blog Machine Learning cơ bản Machine Learning cơ bản (machinelearningcoban.com)

Các package sử dụng:

- Numpy
- Matplotlib
- Pandas
- Scikit-learn
- Gensim
- Scipy