Bài 15 - collaborative và content-based filtering

04 Nov 2019 - phamdinhkhanh

Menu

1. Thuật toán recommendation

Tại sao thế giới lại cần recommendation

Recommendation hiểu một cách đơn giản là khuyến nghị một sản phẩm đến đúng người cần mua. Gỉa sử công ty của bạn hoạt động ở Việt Nam. Điều đó không có nghĩa rằng 95 triệu người Việt sẽ mua hàng của công ty bạn mà chỉ một phần trong số đó có nhu cầu. Trong số các khách hàng có nhu cầu thì không phải ai cũng sẽ mua tất cả các sản phẩm của công ty mà họ có khi chỉ mua một vài sản phẩm mà họ quan tâm.

Một công ty nếu muốn tối đa hóa lợi nhuận thì điều quan trọng nhất họ phải hiểu được khách hàng họ cần gì? Recommendation là một thuật toán kì diệu có thể giúp bạn thực hiện điều đó.

Hãy tưởng tượng tình hình kinh doanh sẽ ra sao nếu không có thuật toán này? Một loạt các hệ quả mà ta có thể hình dung ra:

- Công ty không thể tìm được đúng khách hàng tiềm năng khi người có nhu cầu đối với một sản phẩm lại không được chào bán. Người không có nhu cầu lại bị tiếp cận mời chào. Điều này gây lãng phí thời gian và dẫn tới mất thiện cảm của khác hàng về dịch vụ của công ty.
- Hiệu quả marketing gần như là không đáng kể nếu không tìm đúng tập khách hàng. Chi phí quảng cáo, chi phí cho nhân viên sale tăng lên nhưng doanh thu vẫn thế. Theo một nghiên cứu, một số doanh nghiệp sẵn sàng bỏ ra từ 30-40% lợi nhuận cho việc marketing. Đây là một chi phí không hề nhỏ nhưng để tồn tại họ không thể ngừng đốt tiền. Cuối cùng người hưởng lợi nhiều nhất lại là google, facebook.

Chính vì thể ngày nay thuật toán recommendation được phát triển và ứng dụng rộng rãi trong nhiều doanh nghiệp thuộc đa dạng các lĩnh vực khác nhau như thương mại điện tử, tài chính, ngân hàng, kinh doanh, bán lẻ, phim ảnh,….

Các phương pháp recommendation

Đối với những bạn đã quen thuộc với recommendation thì các phương pháp tôi sắp giới thiệu sẽ không còn mới lạ, đối với những bạn mới tiếp xúc đây có lẽ là những thông tin hữu ích. Lịch sử ra đời sau internet, recommendation chỉ mới xuất hiện và phát triển cách đây khoảng 10 năm. Mặc dù có tuổi đời còn khá trẻ nhưng recommendation là một lĩnh vực phát triển rất nhanh, có số lượng bài báo khoa học lớn và thu hút được nhiều nhà nghiên cứu. Theo trường phái machine learning cổ điển, recommendation sẽ bao gồm 2 nhánh chính:

Top

- **content-based**: Đưa ra các khuyến nghị mua bán cho người dùng dựa trên nội dung liên quan đến sản phẩm. Chẳng hạn một bài hát với các đặc điểm như: người biểu diễn - Xuân Mai, năm phát hành - 2002, thể loại nhạc thiếu nhi sẽ phù hợp với các bé học mẫu giáo. Một sản phẩm có đặc điểm: là xì gà, thương hiệu - Habanos, quốc gia sản xuất - Cuba sẽ phù hợp với những người giới tính nam, có thu nhập cao và sành hút thuốc lá.
- **collaborative filtering**: Hay còn gọi là lọc tương tác, sử dụng sự tương tác qua lại trong hành vi mua sắm giữa các khách hàng để tìm ra sở thích của một khách hàng đối với một sản phẩm. Hầu hết các hành vi hoặc sở thích của mọi người đều có những đặc điểm chung và có thể nhóm lại thành các nhóm tương đồng. Một phụ nữ A nếu đến siêu thị mua dầu ăn thường mua thêm nước tương và nước mắm. Hành vi này lặp lại đối với 100 lượt mua sắm là 90 lần thì khả năng cao một phụ nữ B nếu mua dầu ăn cũng sẽ mua thêm nước tương và nước mắm. Từ đó sẽ khuyến nghị sản phẩm cho khách hàng dựa trên hành vi của các khách hàng khác liên quan nhất.
- **kết hợp cả 2 phương pháp**: Ngoài ra chúng ta cũng có thể sử dụng kết hợp cả 2 phương pháp trên để tạo thành một thuật toán kết hợp. Ưu điểm của phương pháp này đó là vừa tận dụng được các thông tin từ phía sản phẩm và các thông tin về hành vi mua sắm của người dùng.

Về diễn giải thuật toán 2 phương pháp này tôi thấy các bài viết từ 23 đến 25 trên machine learning cơ bản (https://machinelearningcoban.com) là chi tiết và chuẩn mực nhất. Để hiểu cặn kẽ về thuật toán các bạn có thể tham khảo các bài viết này.

Ngoài ra, nếu bạn là một người yêu thích deep learning, sẽ có thêm rất nhiều những phương pháp khác được ứng dụng trong recommendation mang lại hiệu quả cao đó là:

- Thuật toán LSTM dự đoán sản phẩm có khả năng mua tiếp theo của khách hàng dựa vào lịch sử mua sắm.
- Sử dụng các thuật toán NLP (Natural language processing - Xử lý ngôn ngữ tự nhiên) để phân tích các thông tin như phần tên sản phẩm, mô tả sản phẩm, comment khách hàng về sản phẩm để tìm ra sản phẩm tương đồng.
- Sử dụng AutoEncoder để tìm kiếm sản phẩm tương đồng, khách hàng tương đồng. Về auto encoder có thể xem thêm tại Bài 3 - Mô hình Word2Vec (https://phamdinhkhanh.github.io/2019/04/29/ModelWord2Vec.html).
- Các hệ thống search engine dựa trên hình ảnh của sản phẩm.
- Sử dụng reignforcement learning để recommend sản phẩm dựa trên các dự báo về hành vi tiếp theo của khách hàng.
- Sử dụng LDA (https://phamdinhkhanh.github.io/2019/09/08/LDATopicModel.html) để clustering các nhóm sản phẩm có chung đặc điểm và có thể thay thế được cho nhau.
- Thuật toán association để tìm các sản phẩm hoặc nhóm khách hàng có mối liên hệ trong hành vi mua sắm thông qua một chỉ số là levarage.

Và hiện tại các thuật toán về recommendation vẫn đang tiếp tục phát triển mạnh mẽ tại các công ty và lab nghiên cứu trên toàn thế giới.

Recommendation giúp ích gì cho gian hàng của bạn?

Gỉa sử bạn có một gian hàng rất lớn, có thể chứa tới vài ngàn sản phẩm. Theo nguyên lý pareto thì 20% các sản phẩm quan trọng nhất sẽ mang lại 80% doanh thu, trong khi không gian trưng bày là có hạn nên bạn không thể show ra hết 100% các sản phẩm bạn có. Một cách thông thường nhất mà chúng ta thường nghĩ tới đó là:

Top

- Trưng bày những sản phẩm phổ biến hoặc bán chạy nhất tại vị trí dễ tiếp cận nhất.
- Gom các sản phẩm có công dụng tương tự nhau thành các nhóm sản phẩm và trưng bày một ít sản phẩm trong mỗi nhóm tới khách hàng.

Tuy nhiên việc này có nhược điểm là không phải 100% các khách hàng tìm đến đều có nhu cầu mua những sản phẩm phổ biến nhất. Do đó cần có một thuật toán hiểu được với từng nhóm khách hàng có đặc điểm cụ thể sẽ mua gì? cửa hàng cần bày trí các sản phẩm nào? tại đâu? đối với khách hàng nào? thì sẽ tối ưu được doanh thu, tận dụng được nguồn hàng sẵn có và gia tăng khả năng bán hàng.

Đây chính là một trong những yếu tố then chốt tạo nên sự khác biệt giữa các sàn thương mại điện tử vì số lượng hàng hóa của một sàn thương mại điện tử có thể lên tới vài triệu nhưng giao diện chỉ cho phép hiển thị một số lượng ít các sản phẩm. Vấn đề cá nhân hóa (personalization) để đưa ra những hiển thị sản phẩm hợp lý nhất tới từng cá nhân người dùng sẽ là bài toán mà mọi sàn thương mại điện tử luôn tìm cách cải thiện. Có nhiều cách để thực hiện việc cá nhân hóa người dùng và recommendation sẽ giúp bạn thực hiện việc đó.

Bên dưới chúng ta sẽ lần lượt đi qua các thuật toán này để hiểu rõ hơn về phương pháp, cơ chế hoạt động của nó.

2.1. Content-based filtering

Về phương pháp của content-based filtering đã được trình bày rất chi tiết tại Bài 23: Content-based Recommendation Systems (https://machinelearningcoban.com/2017/05/17/contentbasedrecommendersys/). Tôi sẽ giới thiệu khái quát nhất về thuật toán này.

Đối với mỗi một item chúng ta sẽ tìm cách khởi tạo một item profile bằng cách thu thập các trường thông tin liên quan đến item. Mỗi một item profile được đại diện bởi một véc tơ đặc trưng $\mathbf{x}$. Gọi S_i là tập hợp các sản phẩm mà khách hàng i đã rating và giá trị rating là véc tơ $\mathbf{y_i}$. Khi đó chúng ta cần tìm hệ số $\mathbf{w_i}$ là véc tơ cột chứa các hệ số hồi qui thể hiện mức độ yêu thích của khách hàng i đối với mỗi một chiều của sản phẩm.

Hàm loss function đối với những sản phẩm mà khách hàng i đã rating sẽ có dạng:

Double subscripts: use braces to clarify

Trong đó y_{ij} là một phần tử của véc tơ $\mathbf{y_i}$, b_i là hệ số tự do trong phương trình hồi qui tuyến tính và s_i là số lượng sản phẩm mà khách hàng i đã đánh giá.

Nếu ta trích xuất ra ma trận con $\mathbf{X_i}$ gồm những sản phẩm mà khách hàng i đã rating. Mỗi dòng của ma trận là một véc tơ các đặc trưng tương ứng với một sản phẩm. Khi đó hàm loss function có thể được viết gọn lại thành:

Double subscripts: use braces to clarify

Với $\mathbf{e_i}$ là véc tơ cột gồm s_i phần từ bằng 1.

Để rút gọn hơn nữa hàm loss function ta biểu diễn nó đưới dạng phương trình của ma trận mở rộng:

$$\mathcal{L}_i(\mathbf{X_i}; \mathbf{y_i}) = \frac{1}{2s_i}||\bar{\mathbf{X}}_\mathbf{i}\bar{\mathbf{w}}_\mathbf{i} - \mathbf{y_i}||_2^2$$

Ở đây ma trận $\bar{\mathbf{X}}_\mathbf{i}$ là ma trận mở rộng của $\mathbf{X_i}$ bằng cách thêm một véc tơ cột bằng 1 ở cuối, $\bar{\mathbf{w}}_\mathbf{i}$ cũng là véc tơ mở rộng của $\mathbf{w_i}$ khi thêm phần tử 1 ở cuối.

Top

Đây là một phương trình hồi qui tuyến tính quen thuộc nên việc giải nó khá dễ. Trong một số trường hợp để giảm overfiting thì ta sẽ thêm thành phần kiểm soát (regularization term) theo norm chuẩn bậc 2 của $\mathbf{w_i}$ với trọng số là λ (thường có giá trị rất nhỏ). Hàm loss function với thành phần kiểm soát sẽ như sau:

$$\mathcal{L}_i(\mathbf{X_i}; \mathbf{y_i}) = \frac{1}{2s_i}||\bar{\mathbf{X}}_\mathbf{i}\bar{\mathbf{w}}_\mathbf{i} - \mathbf{y_i}||_2^2 + \frac{\lambda}{2s_i}||\bar{\mathbf{w}}_\mathbf{i}||_2^2$$

Ưu điểm của phương pháp này là việc phân loại hoặc dự báo rating của các user sẽ độc lập nhau. Điểm rating của một khách hàng A lên sản phẩm P sẽ không bị phụ thuộc bởi những user khác mà chỉ phụ thuộc vào các đặc điểm liên quan đến sản phẩm P. Do đó chất lượng dự báo sẽ được tăng lên khi dữ liệu được thu thập về sản phẩm là những trường có quan trọng ảnh hưởng đến sở thích của khách hàng.

2.2. Collaborative filtering

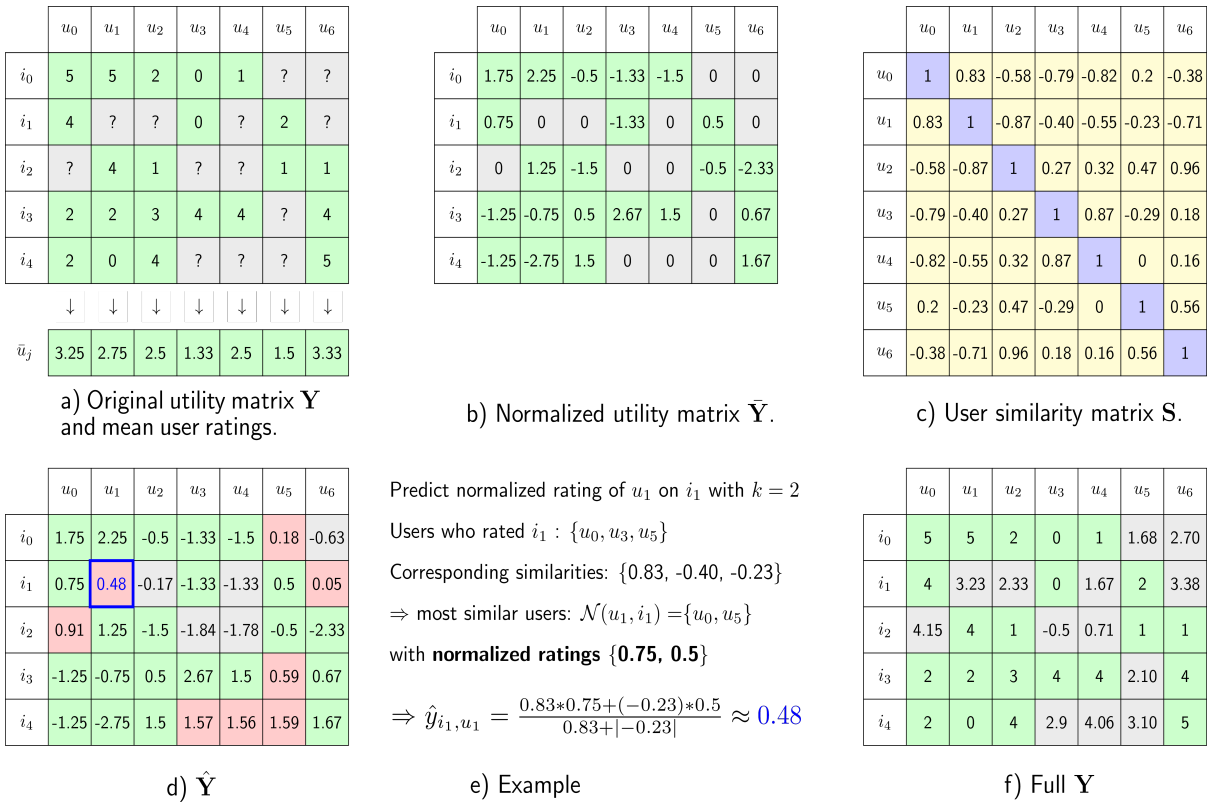
Collaborative filtering là thuật toán lọc tương tác tức là tìm ra sản phẩm mà khách hàng có khả năng ưa thích nhất dựa vào những sản phẩm mà những khách hàng khác có hành vi tương tự đã lựa chọn. Thuật toán sẽ không cần sử dụng thông tin sản phẩm là đầu vào cho dự báo rating. Đầu vào của thuật toán là một **ma trận tiện ích** (*ultility matrix*) chứa giá trị rating của các cặp (user, item). Mỗi cột là các rating mà một user đã rate và mỗi dòng là các rating của một item được rate. Có 2 phương pháp chính được sử dụng trong collaborative filtering bao gồm: Neighborhood-based collaborative Filtering và Matrix Fractorization.

2.2.1. Neighborhood-based collaborative Filtering

Ở phương pháp này ta sẽ cần xây dựng ma trận hệ số tương quan của véc tơ rating của các users để tìm ra nhóm users có cùng sở thích. Hệ số tương quan giữa các users càng lớn thì sở thích của họ càng giống nhau và trái lại thì họ càng có sở thích khác biệt. Thuật toán sẽ dự đoán giá trị rating tại một cặp (user, item) chưa được rate bằng cách tính tổng có trọng số các giá trị rating của k users tương quan nhất với user đó mà đã rate item trên. Trọng số thông thường sẽ bằng chính hệ số tương quan.

Để xây dựng một thuật toán Neighborhood-based collaborative Filtering chúng ta cần trải qua các bước cơ bản bên dưới. Để đơn giản hóa quá trình tính toán, tôi xin phép lấy hình vẽ minh họa từ blog machine learning cơ bản.

Top



	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 $\mathbf{Y}$

Hình 1: Sơ đồ các bước thực hiện thuật toán Neighborhood-based collabrative Filtering (source - Bài 24: Neighborhood-Based Collaborative Filtering (https://machinelearningcoban.com/2017/05/24/collaborativefiltering/)). Bao gồm các bước: a) Lấy trung bình các cột, b) Chuẩn hóa ma trận bằng cách trừ đi trung bình, c) Tính hệ số tương quan của ma trận chuẩn hóa, d) Dự đoán trên ma trận chuẩn hóa các vị trí chưa được rate, e) Diễn giải công thức dự báo rating, f) Chuyển đổi sang giá trị rating thực tế.

Thuật toán sẽ trải qua lần lượt các step sau đây:

1. Chuẩn hóa dữ liệu ở ma trận tiện ích $\mathbf{Y}$ bằng cách trừ đi ở mỗi cột (là các rating của cùng 1 user) trung bình giá trị rating của cột. Việc này là để loại bỏ sự khác biệt về mức độ cho điểm của các user. Vì ví dụ: Có một số user khó tính có thể cho điểm cao nhất là 3 nhưng user dễ tính thì điểm thấp nhất là 3. Khi đó nếu nhận định user khó tính không thích item (nếu ta coi 3 là điểm thấp) hoặc user dễ tính yêu thích item (nếu ta coi 3 là điểm cao) là không chuẩn xác. Chuẩn hóa giá trị rating nhằm mục đích đưa trung bình rating của các user sau khi chuẩn hóa về 0. Gía trị rating dương thể hiện user ưa thích item và trái lại âm sẽ là không thích, bằng 0 là trung lập.
2. Tính ma trận hệ số tương quan giữa các véc tơ cột. Ma trận tương quan thể hiện sự tương đồng trong hành vi mua sắm giữa các user. Từ ma trận tương quan ta có thể xác định ra các users có sở thích tương đồng nhất với một user xác định. Hệ số tương quan dương và càng gần 1 chứng tỏ 2 users có sở thích giống nhau. Hệ số tương quan âm là 2 users có hành vi trái ngược.

Top

3. Dự báo rating của một user u cho một item i bằng cách xác định trên ma trận hệ số tương quan một tập $\mathcal{S}(u,k|i)$ gồm k users có giá trị tương quan lớn nhất đối với user u mà đã rate item i. Gía trị dự báo rating của user u sẽ được tính bằng tổng có trọng số của các rating trong tập k users tương quan nêu trên theo công thức bên dưới:
$\hat{y}_{i,u} = \frac{\sum_{u_j \in \mathcal{S}(u,k|i)} \bar{y}_{i,u_j} \text{sim}(u,u_j)}{\sum_{u_j \in \mathcal{S}(u,k|i)} |\text{sim}(u,u_j)|}$

4. Chuyển giá trị dự báo ở ma trận chuẩn hóa sang giá trị dự báo rating bằng cách cộng các giá trị ở ma trận chuẩn hóa với giá trị trung bình của mỗi cột.

Hạn chế của phương pháp collaborative filtering:

- Thường phải lưu một ma trận hệ số tương quan với kích thước rất lớn. Việc này dẫn tới tốn tài nguyên lưu trữ và thời gian tính toán.
- Ở trên ta đã lựa chọn việc chuẩn hóa theo chiều user. Ngoài ra, ta cũng có thể lựa chọn chuẩn hóa theo chiều item mà không làm thay đổi phương pháp bằng cách chuyển vị ma trận tiện tích $\mathbf{Y}$. Việc lựa chọn chuẩn hóa theo chiều nào sẽ căn cứ trên kích thước theo chiều nào là lớn hơn. Thông thường số lượng user sẽ nhiều hơn item. Khi đó chuẩn hóa theo item sẽ có lợi thế hơn bởi: Kích thước ma trận hệ số tương quan giữa các user là nhỏ hơn nên tốn ít tài nguyên. Thêm nữa khi một user rating một item mới thì giá trị thay đổi về trung bình trên mỗi cột item là nhỏ hơn so với trường hợp chuẩn hóa theo user. Điều này dẫn tới ma trận hệ số tương quan ít thay đổi hơn và tần suất cập nhật cũng ít hơn.

Bên cạnh thuật toán Neighborhood-based collaborative Filtering, một thuật toán khác cũng thuộc lớp các bài toán collabrative filtering đó là matrix fractorization. Thuật toán này thường mang lại độ chính xác cao hơn và đồng thời yêu cầu ít tài nguyên lưu trữ hơn. Cụ thể về mặt mũi thuật toán như thế nào hãy xem phần giới thiệu bên dưới.

2.2.2. Matrix fractorization

Ngoài ra còn một phương pháp collaborative filtering khác dựa trên một phép phân rã ma trận (matrix fractorization). Tức là chúng ta sẽ phân tích ma trận tiện ích thành tích của các ma trận items và ma trận users.

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

Khi đó mỗi dòng của ma trận $\mathbf{X}$ đại diện cho một véc tơ nhân tố ẩn của một item, đó là những nhân tố bất kì, rất trừu tượng mà chúng ta không nên đặt tên cho chúng. Mỗi cột của ma trận $\mathbf{W}$ đại diện cho một véc tơ các hệ số thể hiện mức độ yêu thích của user đối với các nhân tố ẩn. Số lượng nhân tố ẩn thông thường là một số có giá trị rất nhỏ so với số lượng user và item nên dung lượng cần lưu trữ đối với 2 ma trận $\mathbf{X}$ và $\mathbf{W}$ là rất nhỏ so với lưu trữ toàn bộ ma trận $\mathbf{Y}$.

Sau khi tìm được các ma trận items $\mathbf{X}$ và ma trận users $\mathbf{W}$, giá trị ước lượng rating của một user j lên một item i sẽ chính bằng tích: $\hat{y_{ij}} = \mathbf{x_i}^T\mathbf{w_j}$ Như vậy giá trị dự báo được tính toán đơn giản hơn so với Neighborhood-based collaborative Filtering vì chỉ cần thực hiện phép nhân véc tơ mà không cần phải cộng với trung bình cột để chuyển về giá trị gốc.

Qúa trình dự báo hệ số cho mô hình hồi qui của mỗi user tương tự như phương pháp content-based filtering. Nhưng có sự kết hợp giữa tìm nghiệm tối ưu của ma trận items và ma trận users. Qúa trình này được thực hiện xen kẽ nhau nên không chỉ tận dụng được các thông tin là đầu vào

Top

của users mà còn tận dụng được sự giống nhau trong sở thích của các users. Chính vì thế phương pháp mới được xếp vào nhóm collaborative filtering.

Thuật toán

Thuật toán này yêu cầu ta phải thực hiện tối ưu đồng thời cả 2 ma trận users và ma trận items dựa trên hàm mất mát. Khi cần tối ưu ma trận users $\mathbf{W}$ ta sẽ cố định ma trận items $\mathbf{X}$ và dịch chuyển theo phương gradient descent đạo hàm của $\mathbf{W}$ và ngược lại. Thiết lập hàm loss function và quá trình tối ưu sẽ tương tự như thuật toán content-based Filtering thông thường ngoại trừ có thêm thành phần kiểm soát regularization của $\mathbf{X}$ và $\mathbf{W}$ và không chứa hệ số tự do. Hàm loss function có dạng như sau:

Double subscripts: use braces to clarify

Bên dưới là hàm mất mát của ma trận user $\mathbf{W}$ khi cố định ma trận item $\mathbf{X}$.

$$\mathcal{L}(\mathbf{W}) = \sum_{i=1}^{N} \frac{1}{2s_i} ||\mathbf{X_i w_i} - \mathbf{y_i}||_2^2 + \frac{\lambda}{2} ||\mathbf{W}||_F^2$$

Hàm mất mát của ma trận item $\mathbf{X}$ khi cố định ma trận user $\mathbf{W}$:
Double subscripts: use braces to clarify

Về bản chất bài toán của chúng ta có thể chia thành N bài toán nhỏ và mỗi bài toán tương ứng với đi tìm nghiệm tối ưu cho một user. Khi đó phương trình mất mát trên một user sẽ là:

$$\mathcal{L}(\mathbf{w_i}) = \frac{1}{2s_i} ||\mathbf{X_i w_i} - \mathbf{y_i}||_2^2 + \frac{\lambda}{2} ||\mathbf{w_i}||_F^2 \tag{1}$$

Đạo hàm đối với một user sẽ là:

$$\frac{\partial \mathcal{L}(\mathbf{w_i})}{\partial \mathbf{w_i}} = \frac{1}{s_i} \mathbf{X_i}^T (\mathbf{X_i w_i} - \mathbf{y_i}) + \lambda \mathbf{w_i} \tag{2}$$

Một quá trình cập nhật nghiệm theo gradient descent sẽ được thực hiện như sau:

$$\mathbf{w_i} = \mathbf{w_i} - \eta \left(\frac{1}{s_i} \mathbf{X_i}^T (\mathbf{X_i w_i} - \mathbf{y_i}) + \lambda \mathbf{w_i} \right)$$

Hoàn toàn tương tự ta cũng suy ra phương trình cập nhật nghiệm theo gradient descent trên $\mathbf{x_j}$:

$$\mathbf{x_j} = \mathbf{x_j} - \eta \left(\frac{1}{s} (\mathbf{x_j W_j} - \mathbf{y_j}) \mathbf{W_j}^T + \lambda \mathbf{x_j} \right)$$

Qúa trình huấn luyện theo gradient descent sẽ khá lâu vì tốc độ hội tụ phụ thuộc vào learning rate. Trong khi ta nhận thấy hàm loss function trên mỗi user hoặc item có thể tính toán được nghiệm tối ưu thông qua giải phương trình đạo hàm. Qúa trình hội tụ xen kẽ $\mathbf{X}$ và $\mathbf{W}$ khi tính theo phương trình nghiệm sẽ nhanh hơn rất nhiều so với gradient descent. Phương pháp này có tên là ALS (Alternating Least Square). Giải phương trình đạo hàm đối với user thứ i:

$$\begin{aligned} \frac{1}{s_i} \mathbf{X_i}^T (\mathbf{X_i w_i} - \mathbf{y_i}) + \lambda \mathbf{w_i} &= 0 \\ (\mathbf{X_i}^T \mathbf{X_i} + \lambda' \mathbf{I}) \mathbf{w_i} &= \mathbf{X_i}^T \mathbf{y_i} \\ \mathbf{w_i} &= (\mathbf{X_i}^T \mathbf{X_i} + \lambda' \mathbf{I})^{-1} \mathbf{X_i}^T \mathbf{y_i} \end{aligned}$$

Top

Với $\lambda' = s_i \lambda$.

Tương tự, nghiệm tối ưu đối với item thứ j:

$$\mathbf{x_j} = \mathbf{y_j}\mathbf{W_j}^T(\mathbf{W_j}\mathbf{W_j}^T + \lambda'\mathbf{I})^{-1}$$

Chúng ta sẽ thực hiện vòng lặp xen kẽ như sau:

Khởi tạo $\mathbf{X}, \mathbf{W}$

repeat

for i = 1,2,...,M do

$\mathbf{w_i} = (\mathbf{X_i}^T\mathbf{X_i} + \lambda'\mathbf{I})^{-1}\mathbf{X_i}^T\mathbf{y_i}$

end for

for j = 1,2,...,N do

$\mathbf{x_j} = \mathbf{y_j}\mathbf{W_j}^T(\mathbf{W_j}\mathbf{W_j}^T + \lambda'\mathbf{I})^{-1}$

end for

Huẩn luyện mô hình ALS trên python

Hiện tại trên pyspark đã hỗ trợ việc huấn luyện mô hình dưới dạng các file RDD (resilient distributed dataset) là một loại định dạng file hỗ trợ lưu trữ phân tán. Để hiểu thêm về các lệnh cơ bản trên pyspark các bạn có thể xem thêm tại Bài 5 - Model Pipeline - SparkSQL (https://phamdinhkhanh.github.io/2019/07/15/PySparkSQL.html). Bên dưới tôi sẽ giới thiệu qua cách thức xây dựng và huấn luyện một mô hình recommendation system theo phương pháp ALS (Alternating Least Square) trên pyspark.

Bước 1: Load dữ liệu và chia mẫu train/test.

Dữ liệu sẽ được load dưới dạng rdd. Đây là định dạng dữ liệu chịu lỗi tốt và có khả năng phân tán linh hoạt trên nhiều cụm xử lý của spark. Vì những tiện ích này chúng thường được sử dụng trong các tính toán và biến đổi dữ liệu.

Top

```
from pyspark.ml.evaluation import RegressionEvaluator
from pyspark.ml.recommendation import ALS
from pyspark.sql import Row

# Khởi tạo một sparkSession
spark = SparkSession. \
  .builder \
  .appName("ALS recommendation spark session") \
  .getOrCreate()

# Đọc dữ liệu từ dat file
lines = spark.read.text("data/ml-1m/ratings.dat")
# Sử dụng lazy load để map dữ liệu dưới dạng rdd
parts = lines.rdd.map(lambda row: row.value.split(":"))
ratingRDD = parts.map(lambda p: Row(userId=int(p[0]), movieId=int(p[1
                                    rating=float(p[2]), timestamp=int

# Khởi tạo sparkDataFrame từ pandas dataFrame
ratings = spark.createDataFrame(ratingsRDD)
# Chia mẫu train, test theo tỷ lệ 0.8/0.2
(trainig, testing) = ratings.randomSplit([0.8, 0.2])
```

Hiển thị 5 dòng đầu tiên của file ratings.dat.

```
lines.show(5)
```

```
+--------------------+
|               value|
+--------------------+
|1::1193::5::97830...|
|1::661::3::978302109|
|1::914::3::978301968|
|1::3408::4::97830...|
|1::2355::5::97882...|
+--------------------+
only showing top 5 rows
```

Bước 2: Xây dựng và đánh giá mô hình ALS trên pyspark.

Sử dụng module `pyspark.ml.recommendation` ta có thể dễ dàng xây dựng một mô hình recommendation theo phương pháp ALS. Các tham số khi khởi tạo mô hình đó là:

- maxIter: Số lượng vòng lặp tối đa
- regParam: Hệ số kiểm soát regularization
- userCol: tên cột chứa id của user
- itemCol: tên cột chứa id của item
- ratingCol: tên cột chứa giá trị của rating

Top

```
# Xây dựng mô hình recommendation sử dụng thuật toán ALS trên tập dữ l

from datetime import datetime
start_time = datetime.now()
als = ALS(maxIter=5, regParam=0.01, userCol="userId", itemCol="movieId
model = als.fit(training)
end_time = datetime.now()

print('Execute time {}'.format(end_tine - start_time))
```

Thời gian huấn luyện sẽ chỉ hết khoảng 5 phút trên máy tính của mình trong khi huấn luyện các mô hình content-based và collaborative-filtering mà mình đã thực hiện tại matrix fractorization movie length-1m (https://www.kaggle.com/phamdinhkhanh/matrix-factorization-movie-length-1m) trên cùng bộ dữ liệu hết gần 2 tiếng. Theo mình nhận thấy ALS training nhanh hơn rất nhiều so với các dạng thuật toán collaborative và content-based filtering khác. Cuối cùng là đánh giá mô hình thông qua chỉ số căn bậc 2 của trung bình sai số dự báo so với giá trị rating trên tập test, chỉ số RMSE.

```
predictions = model.transform(test)
evaluator = RegressionEvaluator(metricName="rmse", labelCol="rating",
                                predictionCol="prediction")
rmse = evaluator.evaluate(predictions)
print("Root-mean-square error = " + str(rmse))
```

```
Root-mean-square error = 0.8951146317977956
```

Giá trị sai số này thấp hơn so với giá trị sai số phổ biến của các thuật toán content-based và collaborative-filtering khác khi hầu hết RMSE đều khoảng 0.9x.

Bước 3: Khuyến nghị sản phẩm cho người dùng và khuyến nghị người dùng cho sản phẩm.

Thông thường các nền tảng recommendation sẽ tìm ra top 5 hoặc 10 sản phẩm để khuyến nghị cho người dùng vì đây là số lượng vừa đủ (không quá nhiều, cũng không quá ít) để người dùng lựa chọn. Trong pyspark, để tìm ra top 10 sản phẩm để khuyến nghị cho mỗi một user ta thực hiện lệnh `recommendForAllUsers(10)`, rất đơn giản phải không?

```
# Tạo ra top 10 bộ phim khuyến nghị cho mỗi một user
userRecs = model.recommendForAllUsers(10)
```

Trong thuật toán thì user và item là bình đẳng, chỉ cần hoán vị dòng cho cột tại ma trận tiện ích là ta thu được thuật toán recommend user cho item. Muốn tìm ra 10 bộ phim để khuyến nghị cho một user ta sử dụng hàm `recommendForAllItems(10)`.

```
# Tạo ra top 10 user khuyến nghị cho mỗi một bộ phim
movieRecs = model.recommendForAllItems(10)
```

Ngoài ra khi muốn khuyến nghị các bộ phim cho một nhóm các user cụ thể chúng ta chỉ cần xác định các userids và truyền vào hàm `recommendForUserSubset()`. Thực hiện như sau:

Top

```
# Chúng ta cũng có thể tạo ra top 10 bộ phim khuyến nghị cho một tợp h
users = ratings.select(als.getUserCol()).distinct().limit(3)
userSubsetRecs = model.recommendForUserSubset(users, 10)
# Hoặc chúng ta cũng có thể tạo ra top 10 users được khuyến nghị cho m
movies = ratings.select(als.getItemCol()).distinct().limit(3)
movieSubSetRecs = model.recommendForItemSubset(movies, 10)
```

Kết quả của top 10 users được khuyến nghị cho 3 bộ phim đầu tiên.

```
list(movieSubSetRecs.select('recommendations').toPandas()['recommendat
```

```
[[Row(userId=1341, rating=5.175587177276611),
  Row(userId=2339, rating=5.153608322143555),
  Row(userId=4102, rating=5.109521865844727),
  Row(userId=283, rating=5.0519866943359375),
  Row(userId=784, rating=5.00510311126709),
  Row(userId=2694, rating=4.987912178039551),
  Row(userId=1732, rating=4.95065975189209),
  Row(userId=210, rating=4.94167947769165),
  Row(userId=4755, rating=4.934195518493652),
  Row(userId=215, rating=4.904910087585449)],
 [Row(userId=527, rating=6.74972677230835),
  Row(userId=4565, rating=6.49608850479126),
  Row(userId=2138, rating=6.310940265655518),
  Row(userId=1083, rating=6.221969127655029),
  Row(userId=5240, rating=6.059254169464111),
  Row(userId=3979, rating=5.884949207305908),
  Row(userId=5052, rating=5.7887372970581055),
  Row(userId=1235, rating=5.761867046356201),
  Row(userId=1919, rating=5.707066059112549),
  Row(userId=1576, rating=5.6702494621276855)],
 [Row(userId=4996, rating=6.651019096374512),
  Row(userId=4871, rating=6.003638744354248),
  Row(userId=5022, rating=5.986178398132324),
  Row(userId=1363, rating=5.890908241271973),
  Row(userId=5062, rating=5.882445812225342),
  Row(userId=761, rating=5.85439395904541),
  Row(userId=1557, rating=5.842338562011719),
  Row(userId=768, rating=5.804270267486572),
  Row(userId=5440, rating=5.796679973602295),
  Row(userId=4069, rating=5.729244709014893)]]
```

Ta nhận thấy các user được lựa chọn để khuyến nghị cho một bộ phim đều có rating cao nhất. Thứ tự sắp xếp của rating cũng là từ cao xuống thấp.

Trên đây tôi đã giới thiệu sơ qua cho các bạn hầu hết các phương pháp recommendation cổ điển theo 2 nhóm chính collaborative filtering và content-based filtering. Các phương pháp này hiện nay ít được sử dụng vì không đáp ứng được yêu cầu tốc độ tính toán trên những hệ thống dữ liệu ngày càng lớn và độ chính xác cũng kém hơn so với các phương pháp sử dụng mạng neural network.

Top

Bài tiếp theo tôi sẽ dành để viết về các phương pháp recommendation sử dụng mạng neural network có độ chuẩn xác cao hơn.

3. Tài liệu tham khảo

1. collaborative filtering - wikipedia (https://en.wikipedia.org/wiki/Collaborative_filtering)
2. content-based filtering - google course (https://developers.google.com/machine-learning/recommendation/content-based/basics)
3. collaborative filtering - google course (https://developers.google.com/machine-learning/recommendation/collaborative/basics)
4. content-based filtering - machinelearningcoban (https://machinelearningcoban.com/2017/05/17/contentbasedrecommendersys/)
5. collaborative filtering - machinelearningcoban (https://machinelearningcoban.com/2017/05/24/collaborativefiltering/)
6. ALS - alternative least square - standford (https://stanford.edu/~rezab/classes/cme323/S15/notes/lec14.pdf)

Top