Recommender Systems in Digital Marketing

Shanney Suhendra

Introduction to Machine Learning (CS 135)

[9]

The goal of a salesperson is to sell a product or service and one of the key characteristics of a good salesperson is active listening. Purpose of active listening is to inquire more information on the client's needs so that they'll be able to deliver the right solutions to the client, and the way a salesperson does this is by knowing the right questions to ask their client. A salesperson is exactly what a recommender system is, only digital and much more efficient. In technical terms, recommender systems are a subclass of information filtering. Instead of "active listening", recommender systems can simply access user data already stored in the database of the web. These datas answers questions that a salesperson would normally ask, such as previous purchases. By analyzing and identifying patterns in user activities, recommender systems are able to help users discover personalized products or services that suit their preference, which then increases the chance of making a purchase [1].

The main technique used by recommendation systems is collaborative filtering (CF) and there are two forms of CF, user-based CF and target-based CF. The same main target of these two forms is to

find the k-nearest neighbors to user a. Let's first take a look at user-based CF. User-based CF makes predictions on a user's preference and gives recommendations on the basis of historical information collected from many other users that have similar preferences. The logic behind this is that if person A and B have the same opinion on issue X, then it is more likely that person A has the same opinion on another issue Y as person B than with person C whose opinion on issue X differs from person A [2]. A real-world example that uses user-based CF is Netflix, which recommends users a list of movies features as "Movies You Might Like" in Netflix. A summary of how it works is that user A is paired with every other user and certain pairs are detected based on the behavior of both users giving high movie ratings on similar movies. User B (A's pair that was identified in 1.) rated highly on a movie Z which user A have not watched, so netflix recommends movie Z to user A.

$$sim(a,b) = \frac{\sum_p (r_{ap} - \overline{r_a})(r_{bp} - \overline{r_b})}{\sqrt{\sum (r_{ap} - \overline{r_a})^2}\sqrt{\sum (r_{bp} - \overline{r_b})^2}} \qquad r(a,p) = \overline{r_a} + \frac{\sum_b sim(a,b)(r_{bp} - \overline{r_b})}{\sum_b sim(a,b)}$$

fig 1: Pearson Correlation Formula [3]

In technical step:

Step 1, obtain the user-item matrix. This data stores all the movie rating histories every user has done. In a matrix (m×n) form, each row represents a single user and each column represents a movie. Each cell then stores the rating value given by a user on a specific movie.

Step 2, create the user-user similarity matrix. For every pair of users (a,b), similarity between the two is measured. This measure can be calculated using the Pearson Correlation formula $sim(a,b)$ on fig 1, where r_{up} represents rating of user u on movie p and $\overline{r_u}$ represents the average rating of user u. The Pearson Correlation formula is a measure of linear correlation between two sets of data which has a value between -1 and 1, with -1 being the two users have the complete opposite preference, 0 being no correlation and 1 being the two users having completely the same movie preference. We

now have a matrix of the form m×m, where both the rows and columns represent the users from 1 to m. Each cell then stores the corresponding similarity between the two users. Two things to point out is that the diagonal of the matrix will all have a value of 1, since it calculates the similarity between a user and itself, thus the cells contain the maximum value 1. Another thing to point out is that the matrix is symmetric from the diagonal, since for example, row 1 column 2 and row 2 column 1 are calculated for the same pair, user 1 and 2.

Step 3, for simplicity and this example, only the first row of the user-user similarity matrix is used to predict ratings of movies not watched or rated by user 1. Using the list of similar users b and their corresponding similarity value with user 1, s_{1b} , the prediction ratings can be calculated using the $r(a, p)$ formula on fig 1. These prediction ratings are then ranked accordingly from highest to lowest. However in the real world, there are thousands of unrated movies for a user and only a few can be showcased as recommendations. The idea of k-nearest neighbors is introduced here. If k=25, then the 25 highest prediction rating will be featured on "Movies You Might Like" [3, 4].

However, there are some limitations to using user-based CF. From the above example, we can conclude that this technique relies heavily on large amounts of data. This is an issue when it comes to new users or new items, otherwise known as "cold starts", as there is insufficient data on these new entries which will lead to the CF being not so accurate. Reason for this is, for new users, the absence of user's preference, where the users have not rated enough items for the algorithm to determine the kind of preference the user likes. Likewise, for new items, it has not received enough ratings. This will result in popularity bias, as the algorithm will give less weight on these new items and favor more popular items. Another limitation is the scalability. Complexity of the CF algorithm is very large as in normal cases, there would be millions of users O(M) and millions of items O(N), thus requiring many systems and very large memory machines to run the algorithms for millions of users at once.

$$sim(I_a, I_b) = \frac{\sum_{u \in users} r_{ua} \times r_{ub}}{\sqrt{\sum_{u \in users} r_{ua}^2} \sqrt{\sum_{u \in users} r_{ub}^2}} \qquad r(user\ u, Item_c) = \frac{\sum_{I_{not\ c} \in items} r_{ul} \times sim(C, I_{not\ c})}{\sum_{I_{not\ c} \in items} sim(C, I_{not\ c})}$$

Fig 2: cosine similarity formula

The second form of collaborative filtering, item-based CF, was invented by Amazon in 1998 and first published in 2001. It explores the relationship between pairs of items. Unlike the user-based CF, instead of matching the user to other users, item-based CF matches the user's rated items to other similar rated items and uses it to predict items unrated by the user. The technical steps of item-based CF are similar to that of user-based, but focusing on item-item similarity as opposed to user-user similarity. The first step is to form all possible pairs of items. For each pair, we compute their similarities by identifying all the users that had rated both of the items using the cosine similarity formula $sim(I_a, I_b)$ on fig 2, where r_{uv} represents the rating of item v by user u. Unrated ratings of items are then calculated. For simplicity, we will be focusing on rating unrated items by user 1. For example, we want to predict rating of item 7 for user 1, we can use the formula on fig 2, where $r(user\ u,\ Item_c) = r(1,\ 7)$. In a more general sense, all the unrated items of each user are calculated and for each user, items are recommended based on the rating value of the items that was calculated [6].

In short, the user-based CF recommends items based on other similar users while the item-based CF recommends items based on other similar items. We can further compare these two by discussing the benefit item-based CF has over user-based CF that results in item-based CF performing generally better than user-based CF. The item-based CF algorithm resolves the issues of user-based CF discussed previously. For example, it resolves cases where items have only few ratings, as the item-based CF model uses rating distribution per item, not per user. Since there are usually more users than items, this tells us that each item tends to have more ratings than each user having

rated.Thus it is more effective to calculate similarities between items than users as we have more data on items. Additionally, the item-based CF model has a more stable rating distribution. This is because an item's average rating will not be greatly affected by a new rating added to it, as the weight of the new rating will be small due to the many past ratings. This leads to the item-based CF being less complex than the user-based CF, requiring smaller memory machines as item-item similarity matrices have smaller dimensions than user-user similarity matrices [5, 6].

Algorithms for recommender systems have evolved significantly over the past few decades and they are still continuing. In the early 2000s, Netflix held an open competition called the Netflix Prize worth $1,000,000 as an incentive for engineers. The goal was to improve Netflix's own algorithm, *Cinematics*, by 10%. Similarly to the past two projects done in this CS135 course, teams have to generate a recommender systems algorithm that will predict unknown movie ratings by training the model with a training set of known ratings. Predictions are then submitted to a leaderboard and graded by calculating the root mean squared error (RMSE), with the goal to reduce this error as much as possible. Progress prizes were also given annually until in 2009, when the ultimate grand prize was awarded to the "BellKor's Pragmatic Chaos" team, achieving a 10.05% improvement over *Cinematics*. However, the algorithm was not adapted by Netflix. The algorithm design was centered on DVDs and with Netfliz transitioning to movie streaming at the time, Netflix stated that it "did not seem to justify the engineering effort needed to bring them into a production environment. [7]" Instead, Netflix adapted two algorithms from team "Korbell", who won the first progress prize in 2007 for an 8.43%. The two algorithms are Matrix Factorization (now generally known as Singular Value Decomposition, SVD) and Restricted Boltzmann Machines (RBM). Alone, the SVD generated a 0.8914 RMSE while the RBM generated a 0.8990 RSME. But when the two algorithms are implemented together, a 0.88 RMSE is achieved. With few alterations and upgrades, these two algorithms are still being used today as part of Netflix's recommendation engine [7, 8].

Recommender systems play a large role in many industries and generate a very large amount of income when applied efficiently. It helps firms stand out from other firms, allowing personalized "shopping" for customers thus making it convenient. It is a win-win situation for consumers and business owners.

References

[1] Shetty, Badreesh. "An in-Depth Guide to How Recommender Systems Work." *Built In*, 24 July 2019.

[2] Wikimedia Foundation. "Collaborative Filtering." *Wikipedia*, last edited 29 October 2021.

[3] Gupta, Rachit. "User-based Collaborative Filtering." *GeeksforGeeks*, last edited 16 July 2020.

[4] Katipoğlu, Mustafa. "Netflix Movie Recommendation Simulation: How User-Based Collaborative Filtering Works?" 19 November 2020.

[5] Wikimedia Foundation. "Item-item Collaborative Filtering." *Wikipedia*, last edited 9 December 2020.

[6] Gupta, Rachit. "Item-to-Item based Collaborative Filtering." *GeeksforGeeks*, last edited 16 July 2020.

[7] Xavier Amatriain and Justin Basilico "Netflix Recommendations: Beyond the 5 stars (Part 1)." *Netflix Tech Blog*, 6 April 2012.

[8] Wikimedia Foundation. "Netflix Prize." *Wikipedia*, last edited 10 October 2021.

[9] "Improving Customer Engagement with Recommender Systems." *Tricon Infotech*, 11 April 2019.