CMSC 33250 Final Project: Collaborative Filtering

Jiayan Li
University of Chicago
jiayanli@uchicago.edu

Jiamin Yang
University of Chicago
jiaminy@uchicago.edu

1 Introduction

Collaborative filtering (CF) is a technique used in recommendation systems to predict a user's preferences or interests by collecting preferences or behaviors from many users or items (collaborating) and finding similarities or patterns among them..

Our project aims to explore two different techniques of CF in recommendation systems on highly sparse user-rating matrix. Su & Khoshgoftaar's survey paints a comprehensive picture of the techniques of collaborative filtering [1]. The dataset we are going to use is the MovieLens dataset[2].

2 Literature Review

CF and recommender systems have been extensively studied and evolved over the years, leading to various techniques and advancements[3].

CF methods, a cornerstone in recommender systems, encompass Memory-Based CF and Model-Based CF[4]. Memory-Based CF includes User-Based CF, identifying similar users based on their preferences to suggest items, and Item-Based CF, recommending items similar to those previously interacted with by the user. On the other hand, Model-Based CF utilizes statistical and machine learning models, such as matrix factorization techniques and latent factor models[5], to predict user preferences.

Significant contributions in CF revolve around Matrix Factorization models. Techniques like Singular Value Decomposition (SVD) and Alternating Least Squares (ALS) aim to decompose the user-item interaction matrix, minimizing the error between predicted and actual ratings[6]. Hybrid models amalgamate multiple recommendation strategies, blending Collaborative Filtering with content-based or knowledge-based approaches. Additionally, advancements like Neural Collaborative Filtering (NCF) and sequence models leverage neural networks to capture intricate patterns and sequential behavior for improved recommendations[7].

Challenges persist within CF and recommender systems[8]. Scalability issues arise due to large-scale datasets and the inherent sparsity of user-item interactions. Achieving personalized recommendations while ensuring diversity and serendipity remains a focal point, addressing the trade-off between accuracy and providing diverse suggestions. Furthermore, ethical considerations, particularly fairness and bias in recommendations, are vital areas requiring attention to ensure equitable recommendations across different user groups.

3 Methods and Results

3.1 Data

The raw data we used is MovieLens dataset - 1M[2]. The dataset contains rating of 6040 users on 3952 movies.

We first read the dataset as a dataframe including the columns `user_id`, `movie_id`, and `rating`. The dataframe was then pivoted into a user item matrix, with movie id as the index, user id as the columns, and rating as the values. The values are normalized per user, to account for different rating scales across users.

$$\text{normalized rating} = \frac{\text{original rating} - \text{mean(all ratings)}}{\text{max(rating)} - \text{min(rating)}}$$

3.2 Item-Based Filtering

Item-based collaborative filtering operates by examining the items that a target user has rated and recommending the k most similar items. In this method, each item is represented as a vector in an m-dimensional user-space. The similarity between two items is determined by calculating the cosine similarity between their respective vectors.

$$\text{Similarity}(i, j) = \frac{< v_i, v_j >}{\|v_i\| \cdot \|v_j\|}$$

Here, v_i and v_j are the vectors representing items i and j, and $< v_i, v_j >$ denotes the dot product of the two vectors.

To predict the rating of an item i for a user u, the algorithm utilizes the sum of ratings of similar items, weighted by the corresponding similarity terms.

$$\text{Prediction}(u, i) = \frac{\sum_{j \in N(u)} \text{Similarity}(i, j) \cdot \text{Rating}(u, j)}{\sum_{j \in N(u)} |\text{Similarity}(i, j)|}$$

Where:

- $N(u)$ is the set of items similar to item i that user u has rated.
- $\text{Rating}(u, j)$ is the rating user u has given to item j.

3.2.1 Implementation

From the sparse item-user matrix (dataframe with movie id as index, user id as columns, rating as values), we calculated a cosine similarity matrix of movies. The result is a symmetric matrix (dataframe) with movie id being both the index and columns.

We then implemented a function `get_predicted_rating()` that predicts the rating of a given user for a given movie based on the user-item matrix and cosine similarity matrix. For doing this, we calculated the cosine similarity of that given movie with all the movies this user has ever rated, ranked the similarity score in descending order, filtered out the first n most similar items. The predicted rating is derived by averaging the user's ratings of those similar items weighted by similarity score.

Let r_i represent the individual ratings and the s_i represent the corresponding similarity scores. The predicted rating is calculated as:

$$\text{predicted\_rating} = \frac{\sum_i r_i \times s_i}{\sum_i s_i}$$

We also implemented a function to recommend top n unwatched items to a given user. The functions loop over the movie ids that the given user has not watched and utilize the function `get_predicted_rating()` to obtain the predicted rating for that given title for the target user. By sorting the predicted rating, we get the top n recommendations for the user.

3.2.2 Evaluation

For evaluation of the item-based filtering method, we randomly chose rounded 5% of all the users (m=302) as the test user set. For each user, we randomly chose rounded 5% of the watched movies

of the user (if it rounded to 0, n=1), the ratings of those are taken as the test set. We applied the `get_predicted_rating()` function to get a predicted rating for the test set, getting the prediction. We then calculated the Mean Absolute Error (MAE) and Mean Squared Error (RMSE) between the predicted rating and true rating.

$$\text{MAE} = \frac{1}{n}\sum_{i=1}^{n}|\hat{y}_i - y_i|$$

$$\text{RMSE} = \sqrt{\frac{1}{n}\sum_{i=1}^{n}(\hat{y}_i - y_i)^2}$$

In these formula, $\hat{y_i}$ represents the predicted rating for the ith item. y_i represents the true rating for the ith item. n is the total number of items.

The results are:

- MAE = 3.57
- RMSE = 3.70

3.3 Matrix Factorization

The algorithm was first introduced by Simon Funk[9] during the Netflix Prize. Given a user-rating sparse matrix A, we try to estimate A with two lower-rank matrices U, V, i.e. $\tilde{A} = UV^T$. U can be seen as the features of the user, while V can be seen as the features of the movies. Those two matrices can be found by alternatively update U and V until convergence. The objective function to minimize U or V forms exactly the linear regression problem. We can also add b_u, b_v in the prediction of the ratings for the purpose of considering user and item biases.

Formally, suppose there are m users and n movies. We introduce the following notations:

- $A \in \mathbb{R}^{m\times n}$:the user-rating matrix, $\tilde{A} \in \mathbb{R}^{m\times n}$:the predicted user-rating matrix
- $r_{ij} \in A$: the rating for user $i, 1 \le i \le m$ and movie $j, 1 \le j \le n$, $\bar{r}_i$:the mean ratings of user i
- $U \in \mathbb{R}^{m\times p}, V \in \mathbb{R}^{n\times p}, b_u \in \mathbb{R}^m, b_v \in \mathbb{R}^n$: training parameters
- u^i: the i-th row of U, v^j: the j-th row of V, b_u^i: the i-th element of b_u, b_v^j: the j-th element of b_v.

Let Given i, j, we predict the rating $\tilde{r}$ as:

$$\tilde{r} = \bar{r} + b_u^i + b_v^j + u^i \cdot v^j$$

We aim to minimize the following objective function with hyperparameter λ for $L2$ regularization:

$$\sum_{\forall 1\le i\le m, 1\le j\le n, A_{ij}\neq 0} \frac{1}{2}(||A_{ij} - \tilde{A}_{ij}||_2^2 + \lambda((b_u^i)^2 + (b_v^j)^2 + ||u^i||_2^2 + ||v^j||_2^2))$$

Let $e_{ij} = A_{ij} - \tilde{A}_{ij}$. Apply stochastic gradient descent, we update U, V, b_u, b_i as follows with γ being the learning rate:

$$\begin{aligned}
u^i &= u^i + \gamma(e_{ij}v^j - \lambda u^i)\\
v^i &= v^i + \gamma(e_{ij}u^j - \lambda v^i)\\
b_u^i &= b_u^i + \gamma(e_{ij} - \lambda b_u^i)\\
b_v^j &= b_v^j + \gamma(e_{ij} - \lambda b_v^i)
\end{aligned}$$

3.3.1 Implementation and Evaluation

During preprocessing, we filled all missing values in the user-rating matrix with 0. U, V are initialized using a normal distribution with mean=0 and std=1/p, while all bias terms were initialized as 0. For training, we used $p = 2, \gamma = 0.0005, \lambda = 0.02$ and ran 2000 epochs.

We evaluated the performance also using MAE and RMSE on all existing ratings:

- MAE=0.66
- RMSE=0.84

4 Discussion

In this project, we implemented item-based filtering and matrix factorization for recommending movies on Movielens dataset. Comparing MAE and RMSE for both methods, matrix factorization showed significantly better performance. We suspected the difference was mostly contributed by the great sparsity of data that encouraged low-rank representation over similarity-based calculation. The dense representation of U, V, b_u, b_v with trainable parameters also provided more flexibility in representing data.

Interesting questions that can be proposed for moving forward on the project could include strengthening the recommendation with other predictors such as user search history, and user click rate on the current recommendations (learn from user feedback).

References

[1] Su, X., & Khoshgoftaar, T. M. (2009). A survey of collaborative filtering techniques. *Advances in artificial intelligence, 2009*.

[2] Harper, F. M., & Konstan, J. A. The movielens datasets: History and context. Acm transactions on interactive intelligent systems (tiis). 2015; 5 (4): 1–19. DOI=`http://dx.doi.org/10.1145/2827872`

[3] Koren, Y., Rendle, S., & Bell, R. (2021). Advances in collaborative filtering. *Recommender systems handbook*, 91-142.

[4] Herlocker, J. L., Konstan, J. A., & Riedl, J. (2000, December). Explaining collaborative filtering recommendations. In *Proceedings of the 2000 ACM conference on Computer supported cooperative work* (pp. 241-250).

[5] Seroussi, Y., Bohnert, F., & Zukerman, I. (2011, June). Personalised rating prediction for new users using latent factor models. In *Proceedings of the 22nd ACM conference on Hypertext and hypermedia* (pp. 47-56).

[6] Hastie, T., Mazumder, R., Lee, J. D., & Zadeh, R. (2015). Matrix completion and low-rank SVD via fast alternating least squares. *The Journal of Machine Learning Research, 16*(1), 3367-3402.

[7] Li, C., Hu, L., Shi, C., Song, G., & Lu, Y. (2021). Sequence-aware heterogeneous graph neural collaborative filtering. In *Proceedings of the 2021 SIAM International Conference on Data Mining (SDM)* (pp. 64-72). Society for Industrial and Applied Mathematics.

[8] Shi, Y., Larson, M., & Hanjalic, A. (2014). Collaborative filtering beyond the user-item matrix: A survey of the state of the art and future challenges. *ACM Computing Surveys (CSUR), 47*(1), 1-45.

[9] Funk, Simon `https://sifter.org/~simon/journal/20061211.html`