Project 3: Recommendation Algorithms

Ronald Han, rsh98

1.INTRODUCTION

Recommender systems have become a core component of almost every online platform and digital service. These include Netflix, Amazon, Spotify, YouTube and many more. These systems are meant to help users navigate catalogs of products/music/videos without having to look through each and every single one. Because user attention is limited and the amount of content is astronomically more than what a single person can look through, effective recommendation algorithms are essential for a successful user experience.

A central challenge in building a recommender system is trying to figure out which system would be the best one to use. Different services have different catalogs and different user bases, causing them to have different datasets on what their users would want. So which one works the best? How can different datasets impact our results? Which algorithm should we be using? Can we improve on these techniques?

To help build our understanding of a few popular recommendation algorithms, we will be testing and comparing them using different data conditions, and try to figure out which one would be better overall.

2. Algorithms

For our experiment, we will be making use of 2 popular recommendation algorithms: Matrix Factorization, and Collaborative Filtering. By doing this, we are comparing 2 different approaches to recommendation algorithms; Memory-based and Model based.

2.1. Matrix Factorization

Matrix Factorization (MF) is a model-based recommendation approach that learns latent representations of users and items. Unlike memory-based methods, MF is supposed to generalize well even in sparse settings by capturing underlying patterns and hidden factors that influence user preferences.

The MF model approximates a feedback matrix A using the equation:

$$A \approx U V^{\top}, A \in \mathbb{R}^{m \times n}$$

Where m is the number of users, n is the number of items, and U and V represent user and item embedding matrices.

Matrix factorization predicts the interaction between user i and item j using the dot product of their embeddings:

$$A_{ij} = \langle U_i, V_j \rangle$$

However, to learn U and V in the first place, we minimize how far the predicted interactions are from the known ones. A basic objective function often used for matrix factorization is:

$$\sum_{(i,j) \in \text{obs}} \left(A_{ij} - \langle U_i, V_j \rangle \right)^2$$

In this objective function, only known values are used in the model. This is essential as most entries in A will be blank; after all, most people will only be interacting with a tiny fraction of the items that would normally be available.

Using all available rating values from all users, the algorithm then checks to see which items have the largest predicted feedback value (A_{ij}) and recommends the items in order of greatest to least.

2.2. Item-Based Collaborative Filtering

Item-Based Collaborative Filtering (IBCF) is memory based recommendation approach. This method looks at the similarity between any available items: recommending things that are similar to what a user has already liked or interacted with in the past.

The core idea of this system is that users rated 2 separate items, item A and item B, then items A and B are likely similar.
For a given user, the algorithm examines the items they've already interacted with, and goes to find other items that are similar, and uses that to recommend new items.

To measure similarity between pairs of items, Cosine Similarity is used. Given two item vectors (constructed from user ratings), the similarity index between two items can be determined using the equation:

$$\text{sim}(i, j) = \frac{\sum_{u \in U_{i,j}} r_{u,i} \, r_{u,j}}{\sqrt{\sum_{u \in U_{i,j}} r_{u,i}^2} \sqrt{\sum_{u \in U_{i,j}} r_{u,j}^2}}$$

Where $U_{i,j}$ is the set of users that have rated both items i and j, and $r_{u,i/j}$ represents the ratings that these users gave these items.

Once you have the item similarities, you can predict how a given user u would rate an item i (that they haven't rated) by looking at the items they have rated and weighing them by similarity.
This is done using the formula:

$$\hat{r}(u, i) = \frac{\sum_{j \in S(i)} r(u, j) \cdot s_{ij}}{\sum_{j \in S(i)} s_{ij}}$$

In which $S(i)$ is the set of items similar to i, and s_{ij} is the similarity between items i and j.

The user is then recommended items in the order of what the algorithm believes would receive the greatest rating from the user.

3. Experiment and Result Evaluation

3.1. Datasets

For this experiment, the datasets that were used were the MovieLens 100K dataset and the MovieLens 1M dataset.

The MovieLens datasets are benchmark datasets for building and evaluating recommender systems; and they are curated and maintained by GroupLens Research (University of Minnesota). Because they contain explicit user–item rating interactions and basic movie metadata, they are widely used for evaluating collaborative filtering, matrix factorization, and many other recommendation algorithms.

As indicated by their names, MovieLens are datasets that contain reviews for movies by a number of people. Both MovieLens 100K and MovieLens 1M contain User IDs, Movie IDs, Explicit integer ratings (1–5), Movie length, Movie titles, and Movie genres.

The biggest differences between the two datasets is that 1M is a significantly bigger sample size, 100K is significantly more sparse in its data, and 1M is also denser than 100K is.

3.2. Evaluation Metrics

When measuring our results, I had the models return 5 different metrics: the RMSE, MAE, Precision, Recall, and NDCG.

RMSE stands for Root Mean Squared Error, and MAE stands for Mean Absolute Error. These two metrics are measuring the overall rating prediction error and the error of overall prediction accuracy across the entire dataset. For both of these metrics, lower scores are better.

NDCG stands for Normalized Discounted Cumulative Gain. This measures the ranking quality of each recommendation. Precision measures the accuracy of the recommended top 10 for each user, and the recall measures how well the system covers all relevant items for each user. For these metrics, higher scores are better.

While the RMSE and MAE focus more on the results in consideration of the overall dataset, while the other 3 metrics focus more on the results for each individual user in the dataset.

3.3. Results

Below are 2 tables which show the result metrics for both the Matrix Factorization algorithm and the Collaborative Filtering algorithm.

The first table shows the results for the algorithms when using the MovieLens 100K dataset.

The second table shows the results for the algorithms when using the MovieLens 1M dataset.

Table 1. Using MovieLens 100k

Algorithm	RMSE	MAE	Prec	Recall	NDGC
MF	0.942	0.741	0.087	0.036	0.096
IBCF	0.976	0.763	0.043	0.010	0.044

Table 2. Using MovieLens 1m

Algorithm	RMSE	MAE	Prec	Recall	NDGC
MF	0.872	0.687	0.078	0.027	0.083
IBCF	0.943	0.736	0.032	0.004	0.029

4. Performance Comparison

In table 1, we can see that MF achieved a RMSE of 0.942 while IBCF achieved a values of 0.976. Meanwhile, for the MAE values, MF got a score of 0.741, while IBCF got a score of 0.763.

While the scores are similar, MF is slightly lower, meaning that it is slightly more accurate than IBCF (although it is only by approximately 3%).

For precision, recall and NDGC, MF got the scores 0.087, 0.036 and 0.096. Meanwhile, IBCF got 0.043, 0.01 and 0.044. based off of these results, we can see that MF outperformed IBCF by over 100% in all 3 metrics, making it significantly better.

In table 2, we can see that MF achieved a RMSE of 0.872 while IBCF achieved a values of 0.943. Meanwhile, for the MAE values, MF got a score of 0.687, while IBCF got a score of 0.736.

Once again, MF has the lower scores; meaning that it is still more accurate than IBCF on the 1M dataset (this time by approximately 7.5%).

For precision, recall and NDGC, MF got the scores 0.078, 0.027 and 0.083. Meanwhile, IBCF got 0.032, 0.04 and 0.029. based off of these results, we can see that MF, once again outperformed IBCF by over 100% in all 3 metrics, making it significantly better for 1M as well.

Overall, we can see that despite testing the two models on data with differing levels of sparsity, density, and differing amounts of data, Matrix Factorization was able to outperform Item-Based Collaborative Filtering across every single metric. This suggests that MF is universally stronger than IBCF across all metrics and datasets.

Additionally, the precision recall and NDGC scores for IBCF all experienced a decline when moving from the 100K dataset to the 1M dataset; telling us that IBCF degrades sharply on large, sparse datasets like MovieLens-1M.
From this drop in scores, we can infer that Dataset sparsity and item count are major factors driving IBCF's weakness, meaning that it may be possible for IBCF to become more optimal to use in situations with low sparsity, such as a system with few products, or a system where there are a high percentage of completed reviews.

Contrarily, MF benefits significantly from more data, as we can see in our results that its RMSE and MAE dropped by about 7%.

5. Can We Improve On This?

5.1. Results

While our results for IBCF were somewhat disappointing, there is something that I feel it does better than MF. As discussed before, IBCF uses individual item similarities to influence its recommendations, while MF learns global latent representations, which can sometimes cause recommendations to be influenced more by overall item popularity than by specific item similarity. While this method clearly worked out in the scenario, what would

happen if we were to combine this one aspect of IBCF with the rest of MF? Would our performance improve? Or would our performance deteriorate in accordance with our already collected data?

Below are the results from our MF algorithm combined with IBCF item similarities.

Table 3. Using MovieLens 100k

Combined Algorithm	RMSE	MAE	Prec	Recall	NDGC
100K	0.942	0.740	0.089	0.037	0.099
1M	0.872	0.686	0.080	0.028	0.085

5.2. Analysis

Looking at the table above, we can see that our RMSE and MAE scores for 100K and 1M were approximately the same as the results for the standard MF algorithm; with each result being either less than 1% apart or exactly the same.

However, for the precision, recall, and NDGC values, we can see that there was a consistent increase from the standard MF scores; with an overall increase of 3% in each of these metrics.

While these differences may not appear very significant, these metrics directly reflect the quality of the recommended Top-K items, indicating that there was indeed an increase in recommendation quality after making our item mapping change.

6. Conclusion

In this project, we compared two widely used recommendation strategies: Item-Based Collaborative Filtering (IBCF) and Matrix Factorization (MF). Our results indicate that MF consistently outperformed IBCF across both rating-prediction metrics (RMSE, MAE) and ranking-based metrics (Precision, Recall, and NDCG); which demonstrates that MF provides a more accurate and effective approach for generating personalized recommendations.

Although our modified MF variant showed only modest improvements, the gains were measurable, suggesting that even strong baseline models such as MF can benefit from additional optimization or hybridization. More broadly, the findings suggest that while MF is among the most effective collaborative filtering methods, other algorithms capture complementary aspects of user–item interactions. Integrating these strengths—through hybrid models or auxiliary features—remains a promising direction for future improvement.