User Rating Predictions using Matrix Factorization for MovieLens

Manos Chatzakis
emmanouil.chatzakis@epfl.ch

Hind El Bouchrifi
hind.elbouchrifi@epfl.ch

Lluka Stojollari
lluka.stojollari@epfl.ch

Kaggle notebook link

ABSTRACT

This report presents a movie recommender system for the MovieLens dataset, developed for a competition hosted on the Kaggle platform. Our recommender is based on an optimized variation of Matrix Factorization which allows our system to perform accurate recommendations by predicting user ratings for unseen movies. The automated evaluation of the recommender system, which is performed on Kaggle by calculating the RMSE of the predictions, demonstrates the superior performance of our work, which scores a top-ranking value of 0.825.

1 INTRODUCTION

Context. Recommender systems have a pivotal role in user interaction with multimedia platforms ([1][7][3][6]), while the need for efficient and more accurate approaches is increasing. This report presents a movie recommender system for MovieLens [2] dataset, which can accurately predict user ratings for unseen movies. Our system is evaluated through a Kaggle[1] competition using the RMSE[4] loss function.

Approach. Our recommender generates user predictions using an optimized version of Matrix Factorization. Our final, optimized model can achieve an RMSE of 0.825, which is one of the top in the competition, and demonstrates the efficiency of Matrix Factorization recommendation methods.

2 DATASET

Description. We use a version of the MovieLens dataset [2] which contains 9742 movies with their corresponding genres, as well as the ratings of 610 users for the movies of the dataset. In addition, we have user-generated tags for some movies. Tags can be interpreted as keywords for each movie.

Additional Data. Since some of our recommender models require descriptive data for each movie, we scrapped additional information from the full version of MovieLens, which contained an overview text describing the plot in a few sentences, as well as information about the cast and the crew that worked for the movie.

3 PRELIMINARY APPROACHES

In this section, we present our initial approaches to generating user predictions.

3.1 Collaborative Filtering

Collaborative filtering is a recommendation system technique that leverages user behavior and preferences to make personalized suggestions. It operates on the premise that users with similar past preferences will have similar future preferences. There are two main types: user-based and item-based. User-based compares users' behavior, while item-based focuses on item similarities.

User-Based Collaborative Filtering: $\hat{r}_{u,i} = \frac{\sum_{v \in N_u} (r_{v,i})}{|N_u|}$ where: - $\hat{r}_{u,i}$ is the predicted rating for user u on item i. - N_u represents the set of users similar to user u. - $r_{v,i}$ is the rating of user v on item i.

[1]https://www.kaggle.com/competitions/dis-project-2-recommender-systems

Item-Based Collaborative Filtering: $\hat{r}_{u,i} = \frac{\sum_{j \in N_i} (w_{i,j} \cdot r_{u,j})}{\sum_{j \in N_i} |w_{i,j}|}$ where: - $\hat{r}_{u,i}$ is the predicted rating for user u on item i. - N_i represents the set of items similar to item i. - $w_{i,j}$ is the similarity weight between item i and item j. - $r_{u,j}$ is the rating of user u on item j.

We used the cosine similarity as a metric to measure similarities between items and users. For fast computation, we expressed equations in terms of NumPy functions instead of for loops.

3.1.1 Top-K collaborative filtering. To improve our prediction, we considered only the top k users who are most similar to the input user, and similarly, the top k items. We also tuned the parameter k to find the optimal value for minimizing our testing MSE.

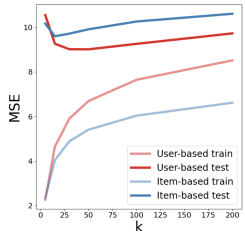

Figure 1: MSE evolution with parameter k change

A k of 40 and 15 produces a good minimum in the test error for user and item based collaborative filtering, respectively.

3.1.2 Result. Uer based Collaborative Filtering achieved a score of 0.975 on Kaggle. While this beats the baseline, it is still not the optimal method which may be due to data sparsity: 1.47% for our dataset and the "cold start" problem. Another reason is the struggle to introduce users to new items that don't have many ratings yet.

3.2 Content-Based Filtering

3.2.1 Description. Content-based recommendations [5] aim to use each movie's content and the available user ratings to predict a user's rating for an unseen movie. As the content, we processed the overview data of each movie using a standard NLP pipeline (lowercasing, tokenization, stopword removal, lemmatization). We calculated the *TF-iDF* matrix to produce a vector per movie. Then, we predict the rating of a user x for a movie i using the formula $r_x(i) = \frac{\sum_{j \in R_x} sim(i,j) r_x(j)}{\sum_{j \in R_x} |sim(i,j)|}$ where $sim(i, j)$ denotes the cosine similarity of the movies i, j based on the TF-iDF matrix, R_x is the set of movies that user x rated, and $r_x(y)$ denotes the rating of user x for movie y.

3.2.2 Optimizations. Using the overview description of a movie as content is only the initial step to incorporate the content of each movie item into the model. Thus, we extended the overview by adding to it the genres of each movie, the tags, as well as the cast participated, intending to enhance the representation power of the vector generated for each movie based on the content. We also tried including crew information, but this led to worse results.

3.2.3 Result. The initial version of the recommender using only the movie overviews scores approximately 1.0 in the Kaggle competition. The optimized version which included the genres, tags, and cast scored approximately 0.9 in the Kaggle competition, which is the best score we achieved using Content-based filtering.

4 MATRIX FACTORIZATION

Matrix factorization is a technique used in recommendation systems, by breaking down a user-item interaction matrix into two or more lower-dimensional matrices to capture latent features or characteristics of users and items. These latent features or embeddings are hidden patterns or traits that influence user preferences.

4.1 Matrix Factorization Exclusively Based on Latent Features

In the calculation of predicted ratings, denoted as $Y_{ij} := y[i, j]$, movie and user features are represented as vectors $\tilde{\mathbf{a}} \in \mathbb{R}^m$ and $\tilde{\mathbf{b}} \in \mathbb{R}^n$. The expression $Y \approx \tilde{\mathbf{a}}\tilde{\mathbf{b}}^T$ encapsulates the core of matrix factorization. To tackle the optimization challenge, the fitting process involves minimizing $\|Y - \tilde{\mathbf{a}}\tilde{\mathbf{b}}^T\|_F$ while adhering to the constraint $\|\tilde{\mathbf{a}}\|_2 = 1$.

In the pure implementation of this formula for our problem, we employed Stochastic Gradient Descent (SGD) to minimize the loss, utilizing Mean Squared Error (MSE) as the loss function. We implemented early stopping after a maximum of 200 iterations, halting the process if, after 10 iterations, there was no significant drop in loss or improvement in validation RMSE. This implementation resulted in a testing set RMSE loss of 0.87 on Kaggle.

4.2 Matrix Factorization with Latent Features, User and Movie Biases, and Regularization

In this variant of matrix factorization, biases for both users and movies are incorporated into the corresponding rating predictions, and a regularization term (λ) is introduced for all learned parameters. Building on the formula $Y_{ij} := y[i, j]$, where movie and user features are represented by vectors $\tilde{\mathbf{a}} \in \mathbb{R}^m$ and $\tilde{\mathbf{b}} \in \mathbb{R}^n$, along with biases b_i and c_j for users and movies, respectively, the expression $Y \approx \tilde{\mathbf{a}}\tilde{\mathbf{b}}^T + b_i + c_j$ encapsulates the core of matrix factorization. To address the optimization challenge, the fitting process involves minimizing $\|Y - (\tilde{\mathbf{a}}\tilde{\mathbf{b}}^T + b_i + c_j)\|_F + \lambda(\|\tilde{\mathbf{a}}\|_F^2 + \|\tilde{\mathbf{b}}\|_F^2 + b_i^2 + c_j^2)$. The inclusion of biases and the application of the same approach for learning through SGD, MSE, and early stopping enhance the model's predictive capabilities. Implementing the aforementioned approach, we achieved a testing RMSE of 0.847 on Kaggle by employing a substantial number of factors (200) to represent user and movie embeddings.

4.3 Neural Network-Enhanced Matrix Factorization with Latent Features from Genres

Building upon the previous implementation results, we extended the matrix factorization approach by incorporating a neural network architecture. In this architecture, the principal components are the learned embeddings of users and movies. The dot product of these embeddings, along with the corresponding biases, was optimized using the ADAW optimizer, resulting in slightly improved convergence compared to the pure matrix factorization implementation. To enhance the model further, genre embeddings were incorporated through a linear layer, including the corresponding bias. This involved encoding and mapping genre information to the same number of factors and adding it to the corresponding item embeddings.

4.3.1 Model Definition. We implemented a custom neural network model for matrix factorization with the following steps:

(1) Generate user, movie, and genre embeddings, each of length 15.
(2) Process genre information through a linear layer and add it to the movie embedding, integrating genre-specific details into the movie representation.
(3) Calculate the element-wise product of user and modified item embeddings and sum up the values (dot product).
(4) Add user and movie biases, along with the global learned offset, to the dot product.
(5) Pass the final score through the sigmoid function to scale the rating predictions between 0 and 5.5 which has proven to be the most effective approach.

For the training phase, we used 15 factors, which proved sufficient to represent latent features for both users and movies, incorporating genre embeddings. A learning rate of 0.001 exhibited optimal convergence, along with a regularization decay of 0.06. This enrichment yielded better results, achieving a new best local minimum with a testing set Kaggle RMSE of 0.825.

We also experimented with incorporating actor information, but the results remained largely unchanged. The challenge may stem from the vast number of actors, making it challenging to obtain a meaningful representation for enriching the corresponding embeddings.

Table 1: RMSE Evaluation

Method	CF	CB	Imp. CB	MF	MF Bias	MF Genres
RMSE	0.975	0.928	0.918	0.87	0.847	0.825

5 CONCLUSION

In conclusion, our extensive exploration encompassed a variety of recommender system approaches , revealing the neural network enhanced matrix factorization, by incorporating latent features from genres as a robust technique for uncovering latent features within user-item interaction matrices. This enabled a deeper understanding of user-item preferences and item characteristics. Looking ahead , the potential of these models for enrichment lies in incorporating additional contextual features to further enhance the precision of recommendations. Also exploring hybrid models that integrate matrix factorization with other advanced techniques represents a promising avenue for achieving even greater adaptability and accuracy.

REMARKS

We provide all our scripts, evaluation code, models, and datasets in our public GitHub repository.

REFERENCES

[1] Debashis Das, Laxman Sahoo, and Sujoy Datta. 2017. A survey on recommendation system. *International Journal of Computer Applications* 160, 7 (2017).

[2] F Maxwell Harper and Joseph A Konstan. 2015. The movielens datasets: History and context. *Acm transactions on interactive intelligent systems (tiis)* 5, 4 (2015), 1–19.

[3] Ayush Singhal, Pradeep Sinha, and Rakesh Pant. 2017. Use of deep learning in modern recommendation system: A summary of recent works. *arXiv preprint arXiv:1712.07525* (2017).

[4] Harald Steck. 2013. Evaluation of recommendations: rating-prediction and ranking. In *Proceedings of the 7th ACM conference on Recommender systems*. 213–220.

[5] Robin Van Meteren and Maarten Van Someren. 2000. Using content-based filtering for recommendation. In *Proceedings of the machine learning in the new information age: MLnet/ECML2000 workshop*, Vol. 30. Barcelona, 47–56.

[6] Jiang Zhang, Yufeng Wang, Zhiyuan Yuan, and Qun Jin. 2019. Personalized real-time movie recommendation system: Practical prototype and evaluation. *Tsinghua Science and Technology* 25, 2 (2019), 180–191.

[7] Renjie Zhou, Samamon Khemmarat, and Lixin Gao. 2010. The impact of YouTube recommendation system on video views. In *Proceedings of the 10th ACM SIGCOMM conference on Internet measurement*. 404–410.