May 2023

A Recommender System for Movie Ratings with Matrix Factorization Algorithm

Amir Alipour Yengejeh
amir.alipouryengejeh@ucf.edu

A Recommender System for Movie Ratings with Matrix Factorization Algorithm

Amir Alipour Yengejeh
dept. Statistics and Data Science
University of Central Florida
Orlando, United States
amir.alipouryengejeh@ucf.edu

***Abstract*—Nowadays, a Recommender System is a technology that aims to predict preferences based on the user's selections. These systems are applied in numerous fields, such as movies, music, news, books, research articles, search queries, social tags, and various products. In this study, we use this potential tool to predict the ratings of users' preferences in *MovieLens* datasets. To do so, we applied the matrix factorization algorithm and calculate the RMSE as our evaluation metric. The results represent that RMSE estimated for the train and test set are 0.83 and 0.93 that are very close one another. This results indicates that our model performance is well.**

***Index Terms*—Recommender System, Matrix Factorization, Sparse matrix, Predicting Movie Rating, MovieLens Dataset.**

I. Introduction

Recommender systems refer to a program that try to suggest the most appropriate products or services to specific users, whether individuals or businesses, by predicting their interest in an item based on relevant data about the items, the users, and their interactions [1]. They reduces information overload by selecting the most relevant information and services from big data to provide customized experiences. The main feature of a recommender system is its ability to analyze a user's actions or the actions of other users to estimate their preferences and interests and offer personalized recommendations [2]. There are several types of the recommender systems, *content-based*, *collaborative filtering*, *hybrid* systems. The content-based systems tries to recommend the items the user already used or liked such as author, actor, genre etc, while the second system recommends the new items based on one's behavior or the other similar behaviors or interactions. However, the hybrid one is a mixture system of the two first approaches to develop the accurate and diverse recommendations.

In this study, we are interested in applying the model-based approach, or *matrix-factorization* of the *collaborative filtering* recommender system to predict movie ratings of the MovieLens datasets. In this system, the historical interactions or actions of the users are typically stores in a matrix known as the "User-Item Interactions Matrix."

II. Data set

In this section, we like to glance on our under study dataset MovieLens. The MovieLens datasets have extensive usage in academia, research, and business. Initially introduced in 1998, the MovieLens datasets contain data about individuals' stated preferences for movies. These preferences are presented as tuples of $<$ user, item, rating, timestamp $>$, where each tuple represents a person's expression of their preference (using a rating scale of 0-5 stars) for a specific movie at a particular time. These preferences were collected through the MovieLens website, which is a recommendation system that prompts its users to provide ratings for movies to obtain personalized movie suggestions [3]. The dataset are available as both training (*MovieLens 100K.base*) and test (*MovieLens 100K.test*) sets.

The dimension of the training set is 80000×4 in which 1650 movies was rated by 943 users, while the test set is 20000×4 where 1410 movies was rated by 459 users. The figure 1 represents the small part of the train set.

	user_id	movie_id	rating	timestamp
0	1	1	5	874965758
1	1	2	3	876893171
2	1	3	4	878542960
3	1	4	3	876893119
4	1	5	3	889751712
5	1	7	4	875071561
6	1	8	1	875072484
7	1	9	5	878543541
8	1	11	2	875072262
9	1	13	5	875071805

Fig. 1. The Train Dataset

III. Data Exploratory

This section focuses on the visualization of the characteristics or attributes of the "rating" variable used in the study. Figure 1 displays the distribution of this variable across the MovieLens datasets, including both the training and test sets. The findings indicate that the high percentage of the movies were rated 4 out of 5 in both sets.

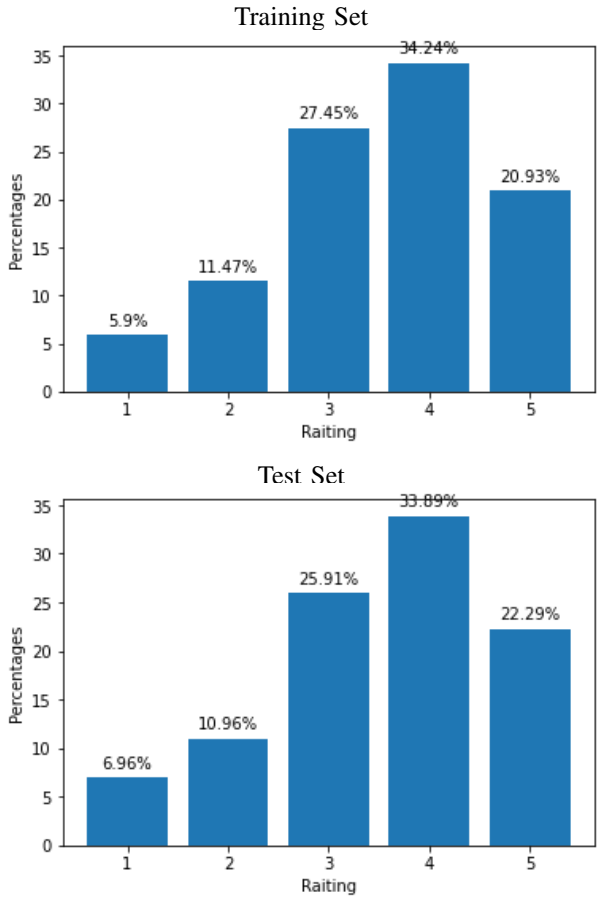

Fig. 2. The distribution of Ratings for MovieLens Datasets

IV. Methodology

In this section, we like to present a simple overview of the mathematical structure behind the matrix factorization for the Recommneder System. As a matter of fact, matrix factorization assumes the presence of a latent space with limited dimensions that can represent users and items. This assumption allows us to calculate the interaction between a user and an item by computing the dot product of their dense vectors in the same space [4]. Figure 2 illustrates the general structure of matrix factorization in the the Recommneder System where the interaction user-item matrix $m \times n$ reconstructed into the dot product of two small matrices, *user matrix* ($m \times l$) and *item matrix* ($l \times n$).

Mathematically speaking, the problem can be expressed in the below equation,

$$R_{m\times n} \approx P_{m\times k} \,.\, Q^T_{k\times n} = \hat{R} \quad (1).$$

Where R denote the *interaction matrix* that is usually a very big sparse matrix, but P and Q are rather small matrices that so-called *user* and *item embedding* matrices. Note that here *K* is a hyper parameter of the factorization to embed the original matrix R intro the small matrices. Thus, the main

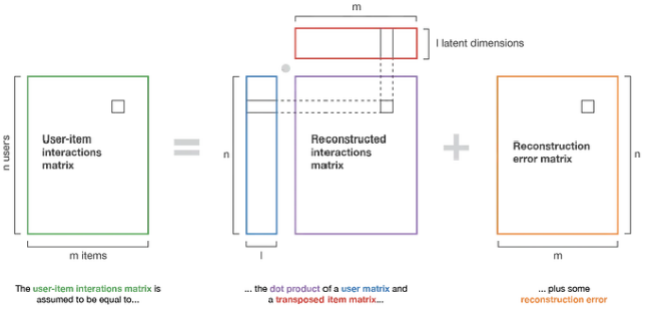

Fig. 3. The general structure of the matrix factorization: (source:https://towardsdatascience.com/introduction-to-recommender-systems-6c66cf15ada)

task is learning the embedded matrices to achieve a close approximation of the user-item matrix, $\hat{R}$ [5].

So, in the predicted matrix $\hat{R}$, any predicted entry $\hat{r}_{ij}$ can be expressed as a dot product of two embedded vectors p_i and q_j as follows [6],

$$\hat{r}_{ij} = p_i^T q_j = \sum_{k=1}^{K} p_{ik} q_{kj} \quad (2).$$

The goal is obtaining the P and Q such that minimizes the error of the actual and predicted ratings,

$$e_{ij}^2 = (r_{ij} - \hat{r}_{ij})^2 = \left(r_{ij} - \sum_{k=1}^{K} p_{ik} q_{kj}\right)^2 \quad (3).$$

To do so, we can apply the regularization algorithms to avoid the overfitting as follows,

$$\begin{aligned} e_{ij}^2 &= (r_{ij} - \hat{r}_{ij})^2 \\ &= \left(r_{ij} - \sum_{k=1}^{K} p_{ik} q_{kj}\right)^2 + \frac{\beta}{2} \sum_{k=1}^{K} \left(||P||^2 + ||Q||^2\right) \quad (4). \end{aligned}$$

where β is a new parameter to control the magnitude of the user and items vectors. By applying the gradient descant update rule, the new update as follows,

$$\begin{aligned} p'_{ik} &= p_{ik} + \alpha \frac{\partial}{\partial p_{ik}} e_{ij}^2 \\ q'_{kj} &= q_{kj} + \alpha \frac{\partial}{\partial q_{kj}} e_{ij}^2 \quad (5) \end{aligned}$$

Next, we can implement the the algorithm in Python [6] [7] [8] to optimize the error function. To do so, we trained our *matrix factorization model* to estimate the R matrix via the P and Q. To evaluate our results, we compare the actual rating values and estimated ones in the test set by calculating the Root Meas Square Error (RMSE) as follows,

$$RMSE = \sqrt{\sum_{u,i \in R} \frac{(r_{ui} - r_{ui}^t)^2}{Number\ of\ ratings}} \quad (4).$$

.

where r_{ui} and r_{ui}^t are the actual and predicted ratings of user u for item i respectively.

V. Results and Discussion

In practice, our user-interaction matrix, or user rating matrix (R) from the MovieLens dataset is very large-scale sparse matrix that need to be broken into small matrices to predict the unseen ratings. To do so, after some trail and errors in our experiment we set K=2. Note that, here K is the number of latent features can be use to embed the rating matrix into two user and item sets. We trained our defined model with the 5000 iterations to get the $\hat{R}$ and the RMSE values of both Training and Test sets. The obtained values summarized in the table 1,

TABLE I
THE EVALUATION TABLE

DataSet	RMSE
Train	0.8583
Test	0.9392

The results represent that both training and test RMSE are roughly close to each other (their difference is less than < 0.1.) It can indicate that our model performance is well, that is, neither overfitting and underfitting. So, we can generalize our model to predict the preference of the movies fans for any new data.

VI. Conclusion

In conclusion, this study implemented a recommender system using matrix factorization on the MovieLens dataset to predict movie ratings for users. The model was trained on a dataset consisting of user ratings for movies and obtained a low RMSE score of 0.939 on the test set, indicating its effectiveness in predicting user ratings for previously unrated movies. The matrix factorization technique used in this study was able to extract latent features from the movie ratings data and use them to make accurate predictions about user preferences.

The results of this study suggest that matrix factorization is a promising approach for building effective recommender systems on large-scale datasets such as the MovieLens dataset.

Future research could explore the use of different matrix factorization techniques, the incorporation of additional data sources such as user demographics and movie genres, and the evaluation of the model's performance using different metrics such as precision and recall. Overall, the findings of this study demonstrate the potential of matrix factorization in developing effective and efficient recommender systems on large-scale datasets such as the MovieLens dataset, which can improve user satisfaction and engagement in online movie recommendation systems.

References

[1] J. Bobadilla, F. Ortega, A. Hernando, A. Gutiérrez.(2013) "*Recommender systems survey*". Knowledge-Based Systems.

[2] P.Resnick, H. R. Varian "*Recommender systems*". (1997) Communications of the ACM.

[3] F Maxwell Harper, Joseph A Konstan.(2015) "". Acm transactions on interactive intelligent systems (tiis).

[4] B. Rocca (2019) *"Introduction to recommender systems"*, Toward Data Science.

[5] K. Chung (2019) *"Matrix Factorization for Recommender Systems"*, GitHub.

[6] A.A. Yeung (2010) *"Matrix Factorization: A Simple Tutorial and Implementation in Python"*", Quuxlabs.

[7] Hippocampus's Garden (2022) *"Meet Pandas: Converting DataFrame to CSR Matrix"*.

[8] Roland Fiagbe (2023) *"Movie Recommender System Using Matrix Factorization"*. stars.library.ucf.edu