Detect User’s Rating Characteristics by Separate Scores for Matrix Factorization Technique

Jia Zhao and Gang Sun *

School of Computer and Information Engineering, Fuyang Normal University, Fuyang 236037, China; zhaojia11b@mails.ucas.ac.cn
* Correspondence: sungang@fync.edu.cn; Tel.: +86-153-8582-3532

Received: 2 October 2018; Accepted: 7 November 2018; Published: 9 November 2018

Abstract: A recommender system can effectively solve the problem of information overload in the era of big data. Recent research on recommender systems, specifically Collaborative Filtering, has focused on Matrix Factorization methods, which have been shown to have excellent performance. However, these methods do not pay attention to the influence of a user’s rating characteristics, which are especially important for the accuracy of prediction or recommendation. Therefore, in order to get better performance, we propose a novel method based on matrix factorization. We consider that the user’s rating score is composed of two parts: the real score, which is decided by the user’s preferences; and the bias score, which is decided by the user’s rating characteristics. We then analyze the user’s historical behavior to find his rating characteristics by using the matrix factorization technique and use them to adjust the final prediction results. Finally, by comparing with the latest algorithms on the open datasets, we verified that the proposed method can significantly improve the accuracy of recommender systems and achieve the best performance in terms of prediction accuracy criterion over other state-of-the-art methods.

Keywords: bias score; collaborative filtering; matrix factorization; rating characteristics; recommender system

1. Introduction

With the rapid development of the Internet, network information has increased (and continues to grow) exponentially, leading to rising difficulties in getting useful information. The recommender system is able to put forward to users information they are interested in; to some extent, it alleviates the problems mentioned above. Therefore, the role of the recommendation system is becoming more and more prominent and has been widely used in shopping [1], tourism [2], movies [3], and other internet platforms.

Recommender systems can accurately suggest valuable information to users because of the Collaborative Filtering (CF) [4–6] algorithm, which does not need to provide extra information (such as content of the item), and can make accurate recommendations on the basis of the user’s historical behavior only, such as clicking, browsing, and rating. This is an efficient recommendation algorithm and one of the most widely used. Other studies [7,8] deal with the content and collaborative-based hybrid mechanisms in multimedia information retrieval.

Matrix factorization [9] technique is one of the main technologies for collaborative filtering algorithm, in that the users’ rating matrix is decomposed into two low dimensional matrices, which represent the users’ interests and the items’ features respectively, and then they are used to make predictions or recommendations.

However, sometimes, the user’s rating score does not really indicate how much the user likes the item. In fact, all users have their own rating characteristics; some of them are more stringent and
make a lower score commonly, while others tend to give higher scores. Through the above analysis, we believe that the user’s final rating score is a “virtual score”, which has been adjusted according to his rating characteristics. The rating score is composed of two parts: one part is related to the user’s preference, we call it the “real score”, and another part is the “bias score” decided by the user’s rating characteristics. Obviously, it is not accurate enough to analyze the user’s preferences from historical rating data, which contains the bias score. Koren et al. [10] proposed that there is a deviation from user’s personal factors in the rating score, but they think that the value of the deviation is simply determined by the average score of both user and item. Averaging the rating characteristic can simplify the model, but the experiments in this paper show that the prediction accuracy will be greatly reduced.

Based on the above reasons, we propose a novel CF method based on matrix factorization and show the basic framework of this method in Figure 1. We analyze the user’s historical rating data and separate it to the real score and the bias score, and then learn the user’s preferences on the real score as well as from the bias score to learn the user’s rating characteristics, which are quantified by the vectors called “bias feature”. Finally, these features can be used to predict the user’s rating score on other items, and the items with higher prediction score are recommended to them. By comparing with the latest algorithms on the open data sets, we verified that the proposed method can significantly improve the accuracy of recommender systems.

![Figure 1. The framework of the proposed method, in which the user-item rating matrix is divided into a real rating matrix and a bias rating matrix. We then learn the relevant features from these two matrices and use them to predict the user’s rating score.](image)

The remainder of this paper is organized as follows: In Section 2, we introduce the CF methods and the matrix factorization techniques. Section 3 gives a detailed description of the proposed method and provides a specific solution. In Section 4, the experimental evaluations and discussions are presented. We have a brief conclusion in Section 5.

2. Related Work

The CF algorithm is one of the most successful algorithms in a recommendation system, which can be divided into two classes [11]: memory-based methods and model-based methods.
Memory-based methods are mainly based on the similarity between users [12] and items [13] to make a recommendation. Taking the user-based CF method as an example: in order to make recommendations for a given user, we first calculates the similarity between the active user and all the other users, and then choose the users who are more similar to that active user as his neighbors based on the Top-N method. We finally make recommendations to the active user according to the behaviors of these neighbors. Although this method is very efficient and easy to implement, there are a lot of problems that are not easy to solve in practical applications, such as scalability and sparsity.

In order to solve the existing problems of the memory-based CF algorithm, many models based on CF algorithms have been proposed: regression-based model [14], latent semantic model [15], clustering model [16], Bayesian model [17], and matrix factorization model [18,19]. In these models, the matrix factorization model has become an outstanding one to learn potential features, such as the famous Singular Value Decomposition (SVD) [20] algorithm. Because of the high complexity of the SVD algorithm, the researchers began to look for an approximate decomposition, such that the user-item rating matrix is decomposed into two low dimensional matrices, which are expressed as the users’ preferences and the items’ features, respectively. The representative works of this method include Non-negative Matrix Factorization (NMF) [21], Nonparametric Probabilistic Principal Component Analysis (NPCA) [22], and Maximum Margin Matrix Factorization (MMMF) [23]; they all show good performance in dealing with largescale data. In order to further improve the accuracy of the matrix factorization model, many models add additional information to enhance the ability to learn the features of users and items, Zhang et al. [24] added the users’ social information by obtaining the similarity between them to the model, and Liang et al. [25] added the co-occurrence relation between items to improve the performance of the matrix factorization model. In addition, Wibowo et al. [26] improved the density of the matrix by generating pseudo transactions to make the matrix decomposition results more stable and accurate.

However, these improvements mentioned above do not essentially solve the problem that the user’s rating data has been modified by his rating characteristics, which makes it difficult to get an accurate result of the user’s preferences. Commonly, the unexpectedness included in rating data is taken as an error, and some other researchers, such as Mori [18] and Ortega [27], think that the user’s bias is the difference between the average score of the user and the average score of the whole dataset. However, the unexpectedness can help to increase accuracy [28] and can include large user and item biases, i.e., systematic tendencies for some users to give higher ratings than others, and for some items to receive higher ratings than others [10]; Wang [29] verifies this idea. Koren [30] also considers dynamic CF, which thinks that biases change with time. Obviously, in practice, the user’s bias may be more complex; Jonathan [31] discusses the difference between bias and magic barrier. Some other types of bias are also discussed, such as popular bias [32,33] and position bias [34].

In this paper, we use a vector to represent the user’s bias feature, which is a numerical representation of the user’s rating characteristics. We use the matrix factorization technique to get the bias features from the user’s historical rating behavior, and then apply these features to correct the final prediction result, so as to improve the accuracy of recommendation.

3. Methods and Algorithms

In this section, we will introduce the details of the proposed method and the implementation process.

3.1. Constructing the Model

3.1.1. Real Score and Bias Score

Let the user-item rating matrix of $n$ users to $m$ items be $R \in \mathbb{R}^{n \times m}$, the entry $R_{ij}$ represents the $i$-th user’s rating score on the $j$-th item, if the rating record does not exist, then $R_{ij} = 0$. According to the previous discussion, when $R_{ij} \neq 0$, $R_{ij}$ should contains two parts, one part is real $R_{ij}$ decided by the
i-th user’s rating characteristics, and another part is $biasR_{ij}$ decided by the i-th user’s preferences. So we have:

$$R_{ij} \approx biasR_{ij} + realR_{ij}$$  \hspace{1cm} (1)

and on the whole rating matrix:

$$R \approx biasR + realR$$  \hspace{1cm} (2)

Although there is a certain bias of the user’s rating, it is clear that the bias score should not be too large; therefore, we have some limitations on $biasR$, which we will discuss later.

### 3.1.2. User Preferences and Item Features

Here, we discuss the relationship between the real score and the user’s preferences.

According to [10], item $j$ is associated with a vector $V_j \in \mathbb{R}^{1 \times a}$ that represents the features of the $j$-th item, where the dimension $a$ indicates that the item has $a$ features, similarly, user $i$ is also associated with a vector $U_i \in \mathbb{R}^{1 \times a}$ that represents the $i$-th user’s preferences, and the $k$-th element $U_{ik}$ measures the extent of interest the $i$-th user has in the $k$-th feature of the items. Finally, we use the product of $U_i$ and $V_j$ to represent the real score $realR_{ij}$:

$$U_iV_j^T = realR_{ij}.$$  \hspace{1cm} (3)

Assuming that all users’ interests are represented by a matrix $U \in \mathbb{R}^{n \times a}$, and the features of all items are expressed by the matrix $V \in \mathbb{R}^{m \times a}$, we get:

$$UV^T = realR.$$  \hspace{1cm} (4)

### 3.1.3. User Bias Features and Item Bias Weights

Here, we discuss the impact of the user’s bias features on the $biasR$. Like the user’s preferences, we believe that the user’s bias features are also the user’s own inherent attributes, so we define a vector $P_i \in \mathbb{R}^{1 \times b}$ as the $i$-th user’s bias features, where the parameter $b$ represents the dimension of the vector. Although the user’s bias features are fixed, the bias score for different items is not the same; therefore, the $biasR$ is also affected by items. Thus, we define another vector $Q_j \in \mathbb{R}^{1 \times b}$ as the bias weights for the $j$-th item, and then we have:

$$P_iQ_j^T = biasR_{ij}.$$  \hspace{1cm} (5)

If the bias features of all users are represented by a matrix $P \in \mathbb{R}^{n \times b}$, and the bias weights of all items are represented by a matrix $Q \in \mathbb{R}^{m \times b}$, then on the whole data:

$$PQ^T = biasR.$$  \hspace{1cm} (6)

### 3.1.4. The Unified Model

We combine the above description to form the final model.

Include Equations (3) and (5) in Equation (1) and we get:

$$R_{ij} \approx P_iQ_j^T + U_iV_j^T,$$  \hspace{1cm} (7)

Therefore, in order to get the matrix $P$, $Q$, $U$ and $V$ to satisfy Equation (7), we obtain the following loss function:

$$L = \frac{1}{2} \sum_{i,j} \delta_{ij} ||U_iV_j^T + P_iQ_j^T - R_{ij}||_F^2,$$  \hspace{1cm} (8)

where $|| \cdot ||_F^2$ denotes the Frobenius norm, and the variable $\delta_{ij}$ is used to control the elements that should be added to the loss, when $R_{ij} \neq 0$, then $\delta_{ij} = 1$, means that the element should be included in the loss, otherwise $\delta_{ij} = 0.$
Obviously, the smaller the $L$ is, the matrix $P$, $Q$, $U$ and $V$ obtained are more accurate. We add some regularization terms to prevent overfitting and avoid the circumstance that one of the matrices increases infinitely and the other matrix tends to 0, so we get the loss function as follows:

$$L = \frac{1}{2} \sum_{i,j} \delta_{ij} ||U_i V_j^T + p_i q_j^T - r_{ij}||^2 + \alpha \left( ||U||^2 + ||V||^2 \right) + \beta \left( ||P||^2 + ||Q||^2 \right)$$

(9)

where $\alpha$ and $\beta$ are regular parameters and are both larger than 0. As mentioned above, the bias score should not be too large; therefore, the proportion of the bias score is controlled by adjusting the parameters $\alpha$ and $\beta$. We set $\alpha = k\beta$, where $k \geq 1$ and replace $\beta$ with the parameter $\lambda$, and the final model of this paper is as follows:

$$L = \frac{1}{2} \sum_{i,j} \delta_{ij} ||U_i V_j^T + p_i q_j^T - r_{ij}||^2 + \frac{k\lambda}{2} \left( ||U||^2 + ||V||^2 \right) + \frac{\lambda}{2} \left( ||P||^2 + ||Q||^2 \right),$$

(10)

By solving the optimization problem in Equation (10), we will be able to get the required four matrices $P$, $Q$, $U$ and $V$.

3.2. The Solution of Our Method

Here, we will give a specific solution of the proposed method. We update the variables alternately, according to the following two steps in order to get the minimum value of $L$:

1. keep $U$ and $P$, update $V$ and $Q$;
2. keep $V$ and $Q$, update $U$ and $P$;

Repeat step (1) and (2) until $L$ converges. What follows is the method of updating parameters.

3.2.1. The Updating Rules for $V$ and $Q$

We obtain the partial derivative of $V_j$ from Equation (10):

$$\frac{\partial L}{\partial V_j} = V_j (U^T C^n U + k\lambda I) + (Q_j P^T - R_j^T) C^n U,$$

(11)

where $C^n \in \mathbb{R}^{n \times n}$ is a diagonal matrix, and $C^n_{ii} = 0$ if $R_{ij} = 0$, otherwise $C^n_{ii} = 1$.

Similarly, we get the partial derivative of $Q_j$ from Equation (10):

$$\frac{\partial L}{\partial Q_j} = Q_j (P^T C^n P + \lambda I) + (V_j U^T - R_j^T) C^n P,$$

(12)

We set the derivatives (11) and (12) to zero and obtain the analytical solutions as:

$$V_j = (R_j^T - Q_j P^T) C^n U (U^T C^n U + k\lambda I)^{-1},$$

(13)

$$Q_j = (R_j^T - V_j U^T) C^n P (P^T C^n P + \lambda I)^{-1},$$

(14)

It can be seen from Equations (13) and (14) that updating $V$ needs to use the value of $Q$ and updating $Q$ needs to use the value of $V$; therefore, you can use a temporary variable to remember the value of the last updated $V$ and $Q$ for the current updating round. However, this paper uses another strategy to solve this problem: we can update bias$R$ and real$R$ before updating $V$ and $Q$ according to Equations (4) and (6), and then we get the final updating rules:

$$V_j = (R_j^T - \text{bias}$R$) C^n U (U^T C^n U + k\lambda I)^{-1},$$

(15)
\[ Q_j = (R_j^T - \text{real} R_j^T)C^n P (P^T C^n P + \lambda I)^{-1}, \quad (16) \]

### 3.2.2. The Updating Rules for \( U \) and \( P \)

Same as the method to get updating rules for \( V \) and \( Q \), we obtain partial derivatives of \( U \) and \( P \) respectively from Equation (10):

\[
\frac{\partial L}{\partial U_i} = U_i (V^T C_m V + k\lambda I) + (P_i Q^T - R_i) C_m V, \quad (17)
\]
\[
\frac{\partial L}{\partial P_i} = P_i (Q^T C_m Q + \lambda I) + (U_i V^T - R_i) C_m Q, \quad (18)
\]

We set the derivatives (17) and (18) to zero and obtain the analytical solutions as:

\[
U_i = (R_i - P_i Q^T) C_m V (V^T C_m V + k\lambda I)^{-1}, \quad (19)
\]
\[
P_i = (R_i - U_i V^T) C_m Q (Q^T C_m Q + \lambda I)^{-1}, \quad (20)
\]

We also update \( \text{bias} R \) and \( \text{real} R \) before updating \( U \) and \( P \) to avoid the interdependence between parameters, so we get the final update rules for \( U \) and \( P \):

\[
U_i = (R_i - \text{bias} R_i) C_m V (V^T C_m V + k\lambda I)^{-1}, \quad (21)
\]
\[
P_i = (R_i - \text{real} R_i) C_m Q (Q^T C_m Q + \lambda I)^{-1}, \quad (22)
\]

### 3.2.3. Algorithm Overview

Based on the above derivation, we give a summary of the algorithm in Algorithm 1.

---

**Algorithm 1. Algorithm of Proposed Model**

**Input:**
- User-item rating matrix \( R \)
- Parameters \( n, m, a, b, k, \lambda \)

**Output:**
- Matrix \( P, Q, U \) and \( V \)

1. Randomly initialize \( P, Q, U \) and \( V \);
2. repeat
3. Compute \( \text{bias} R \) and \( \text{real} R \) based on Equations (4) and (6);
4. for \( i = 1 \) to \( n \) do
5. \( U_i \) according to Equation (15);
6. \( V_j \) according to Equation (16);
7. end for
8. Go to step 3;
9. for \( j = 1 \) to \( m \) do
10. \( U_i \) according to Equation (21);
11. \( P_i \) according to Equation (22);
12. end for
13. until convergence

When the matrix \( U, V, P, \) and \( Q \) are obtained through the above algorithm, the unknown entries in rating matrix \( R \) can be predicted and we can recommend to the users items with higher prediction score. In this paper, the prediction score \( \hat{R}_{ij} \) of the \( i \)-th user on the \( j \)-th item is calculated with the following expression:

\[
\hat{R}_{ij} = U_i V_j^T + P_i Q_j^T, \quad (23)
\]
4. Experimental Evaluation

4.1. Data Description

In order to test the performance of our method, we select three different open datasets for the experiments: MovieLens-100K, MovieLens-1M, and Epinions.

The MovieLens-100K data set is from the MovieLens website, which contains nearly 100,000 rating records of 943 users on 1682 films and all the rating scores are a positive integer and not greater than 5. The source of the MovieLens-1M data set is the same as that of the MovieLens-100K data set, but this one is released later and contains 1,000,209 rating records of 6040 users for 3952 movies, where each user rated at least 20 movies. The Epinions dataset consists of the user’s ratings of merchandise from the Epinions website; however, it contains several users with fewer ratings and many items that lack valuable evaluation information. Therefore, before the experiments, we first preprocess the dataset such that users who have scored less than 10 times and items that have been scored less than 10 times are removed. Finally, 354,857 rating records of 15,687 users on 11,657 items are acquired. The statistics of the three datasets are shown in Table 1.

|                      | MovieLens-100K | MovieLens-1M | Epinions |
|----------------------|----------------|--------------|----------|
| # of users           | 943            | 6040         | 15,687   |
| # of items           | 1682           | 3952         | 11,657   |
| # of ratings         | 100,000        | 1,000,209    | 354,857  |
| # of ratings per user| 106.4          | 165.60       | 22.62    |
| # of ratings per item| 59.45          | 253.09       | 30.44    |
| Rating Sparsity      | 93.70%         | 95.81%       | 99.81%   |

As can be seen from Table 1, the MovieLens-100K dataset is the smallest, the average of users’ rating times is the most in the MovieLens-1M; and the Epinions dataset is the most sparse.

4.2. Evaluation Measures

The proposed method is first to predict the user’s rating score, and then to make recommendations based on the prediction. Therefore, we adopt the Mean Absolute Error (MAE) [31] method, which is commonly used in the field of recommendation system to evaluate the performance of the algorithms, and the expression is as follows:

$$MAE = \frac{1}{N} \sum_{i,j} |R_{ij} - \hat{R}_{ij}|,$$

where \( N \) is the number of test data, \( R_{ij} \) represents the rating score on the \( j \)-th item by the \( i \)-th user in the test data, and \( \hat{R}_{ij} \) represents the predicted score obtained by using the recommendation algorithm. Obviously, the lower the MAE value, the better the predicted results coincident with the user’s real situation, which means better performance of the algorithm.

4.3. Compared Methods

In order to verify the effectiveness and accuracy of our method, we choose a variety of recommendation algorithms to compare with ours, which are listed as follows:

Item-Based (IB) [13]: The basic idea of this method is to calculate the similarity between items and then recommend items similar to what is preferred by the active user. In this paper, we use the vector cosine similarity model to compute the item-item similarities.

SVD [20]: This algorithm is based on the Singular Value Decomposition method, wherein the rating matrix is decomposed into two low dimensional matrices, which will be further used to make the prediction.
PMF [35]: A widely used matrix factorization model, based on the Gaussian model for the recommendation. In this paper, we set the regularization parameters by the grid \{0.01, 0.05, 0.1, 0.5, 1, 5, 10\}.

MCoC [15]: The main idea of this method is clustering the users and items into several subgroups, and then making recommendations within each subgroup by using the basis CF method.

DsRec [36]: A hybrid model by combining the basis clustering model and the matrix factorization model to improve prediction accuracy.

PMMMF [37]: This method was proposed by Kumar, who improved the traditional MMMF method by using Proximal Support Vector Machine (PSVM).

Hern [38]: This method was proposed by Hernando, so we use the first author’s name to name this algorithm; they use the matrix factorization method based on Bayesian probability model to decompose the user-item rating matrix into two nonnegative matrices, where the values of all elements are between 0 and 1.

SCC [39]: This method is the same as MCoC, which is a recommendation algorithm based on clustering; the difference is that it clusters items by using a self-constructing clustering method.

TyCo [40]: This method borrows ideas of object typicality from cognitive psychology and finds “neighbors” of users based on user typicality degrees in user groups.

### 4.4. Experimental Results

In our experiments, we divide the three public data sets into two parts, respectively, randomly select 80% of the data as the training data, and the remaining 20% as the test data. Each dataset is randomly divided five times and the average value of the MAE obtained in all the five tests is regarded as the final result. The predicted score calculated by Equation (25) may not be an integer, while the user’s rating scores are all integers in the test data; therefore, the prediction results are rounded and then the integer obtained is used as the final prediction score.

The experimental results on MovieLens-100K are shown in Table 2, where we set the parameters \( k = 9, \lambda = 0.8, a = 3 \) and \( b = 1 \). Table 3 summarizes the performance of different methods on MovieLens-1M, where the configuration of parameters is the same as in MovieLens-100K, except that we set \( a \) to 10; we will discuss the impact of the parameters on the experimental results later. Table 4 shows the experimental results on Epinions, where we set the parameters \( k = 19, \lambda = 0.6, a = 1 \) and \( b = 1 \).

#### Table 2. Comparison of the MAE Values on MovieLens-100K.

|    | IB | SVD | PMF | MCoC | DsRec | PMMMF | Hern | SCC | TyCo | Our |
|----|----|-----|-----|------|-------|-------|------|-----|------|-----|
| 1  | 0.7325 | 0.7412 | 0.7248 | 0.7024 | 0.7105 | 0.7221 | 0.7068 | 0.7589 | 0.7356 | 0.6820 |
| 2  | 0.7290 | 0.7335 | 0.7213 | 0.7210 | 0.7098 | 0.7104 | 0.7087 | 0.7546 | 0.7247 | 0.6713 |
| 3  | 0.7329 | 0.7401 | 0.7205 | 0.7138 | 0.7122 | 0.7156 | 0.7080 | 0.7574 | 0.7289 | 0.6753 |
| 4  | 0.7432 | 0.7388 | 0.7280 | 0.7189 | 0.7201 | 0.7166 | 0.7133 | 0.7621 | 0.7381 | 0.6894 |
| 5  | 0.7405 | 0.7342 | 0.7235 | 0.7242 | 0.7164 | 0.7190 | 0.7064 | 0.7608 | 0.7298 | 0.6818 |
| avg| 0.7356 | 0.7376 | 0.7236 | 0.7197 | 0.7138 | 0.7167 | 0.7086 | 0.7588 | 0.7298 | 0.6800 |

#### Table 3. Comparison of the MAE Values on MovieLens-1M.

|    | IB | SVD | PMF | MCoC | DsRec | PMMMF | Hern | SCC | TyCo | Our |
|----|----|-----|-----|------|-------|-------|------|-----|------|-----|
| 1  | 0.7012 | 0.6956 | 0.6923 | 0.6824 | 0.6795 | 0.6745 | 0.6689 | 0.6913 | 0.6643 | 0.6247 |
| 2  | 0.6946 | 0.6915 | 0.6886 | 0.6836 | 0.6823 | 0.6757 | 0.6675 | 0.6874 | 0.6613 | 0.6231 |
| 3  | 0.7088 | 0.7013 | 0.6894 | 0.6876 | 0.6804 | 0.6742 | 0.6681 | 0.6902 | 0.6620 | 0.6221 |
| 4  | 0.6985 | 0.6950 | 0.6902 | 0.6824 | 0.6794 | 0.6750 | 0.6692 | 0.6895 | 0.6637 | 0.6222 |
| 5  | 0.7023 | 0.6968 | 0.6911 | 0.6853 | 0.6811 | 0.6762 | 0.6684 | 0.6908 | 0.6640 | 0.6264 |
| avg| 0.7011 | 0.6960 | 0.6903 | 0.6843 | 0.6805 | 0.6751 | 0.6684 | 0.6898 | 0.6631 | 0.6237 |
By observing the experimental results on three data sets from Tables 2–4, we can get the following conclusions:

- The experimental results show that the proposed method has the lowest MAE value in all rounds of tests, which means that our method has a higher prediction accuracy. Therefore, it is verified that considering the user’s rating characteristics can indeed improve the performance of the recommendation system.

- Although our algorithm has achieved the best results in all three datasets, the degree of improvement is different. By comparing with the best results of other algorithms, we reduced the MAE value by about 0.03 in MovieLens-100K, by nearly 0.04 in MovieLens-1M and by 0.02 in Epinions. By observing the statistics of the three datasets from Table 1, we believe that the reason for this difference may be due to the number of ratings per user. Because on the MovieLens-1M dataset, the average number of users’ rating times is the largest, reached about 165 times, which is very useful to get accurate user’s rating characteristics, while in Epinions, the average number of users rating times is only about 23, which brings a certain degree of difficulty to accurately get the user’s rating characteristics.

- We set different values for the parameter $a$ on the three datasets, which means that the more user rating times, the more complex the user’s rating characteristics we can get; thus, we need to use a higher dimensional vector to express bias features.

On all the three datasets, we set $b = 1$, which means that most of the users’ realR is only decided by their main preference, and the influence of the secondary preference on realR is very small. From another perspective, we can consider that these subtle effects are also caused by the users’ rating characteristics. Therefore, if we set the value of $b$ to be greater than 1, there will be an overfitting situation. We will discuss the impact of parameters on the performance of the proposed method in the paragraphs below.

4.5. Experimental Results

In this part, we mainly discuss the impact of parameters in the proposed model.

4.5.1. Impact of $a$

We selected 20 different values for $a$ on the three datasets for the experiment. The values are all between 1 and 20. We set $k = 9, \lambda = 0.8$ and $b = 1$ on MovieLens-100K and MovieLens-1M, and set $k = 19, \lambda = 0.6$ and $b = 1$ on Epinions. The experimental results are shown in Figure 2a.

It can be seen from Figure 2a that on Epinions, the MAE value increases with the increase of $a$, while on the MovieLens-100K and MovieLens-1M data sets, it decreases first and then increases gradually. After analysis, we believe that because the number of ratings per user on the Epinions data set is very small, the bias feature vector with the dimension of 1 is enough to be a good characterization of the user’s rating characteristics. If we continue to increase the dimension of $a$, it will be overfitting, so the greater the value of $a$ on Epinions, the worse the performance. However, on MovieLens-100K and MovieLens-1M, the number of ratings per user is much bigger than on the previous one; it is difficult to completely characterize the user’s rating characteristics by a lower dimensional vector.
Therefore, with the increase of $a$, prediction performance will be improved. The best performance is achieved when dimension $a$ is appropriate. When the parameter $a$ continues to increase, it will be over fitting, which appeared on Epinions, thus the performance will gradually decrease.

![Figure 2](image)

**Figure 2.** The impact of parameters $a$, $b$, $k$ and $\lambda$. In the experiment on each parameter, we fix three parameters and change the remaining one. The framework of the proposed method divides the user-item rating matrix into a real rating matrix and a bias rating matrix, and then we learn the relevant features from these two matrices and use them to predict the user’s rating score.

### 4.5.2. Impact of $b$

As with the test for the parameter $a$, we did the same for parameter $b$. We set $k = 9$, $\lambda = 0.8$ and $a = 1$ on MovieLens-100K and MovieLens-1M, and set $k = 19$, $\lambda = 0.6$ and $a = 1$ on Epinions. The experimental results are shown in Figure 2b.

As can be seen from Figure 2b, with the increase of $b$, the value of MAE on all the three datasets have maintained a rising trend, although the MAE value on MovieLens-1M was slightly reduced at first. The value was too small to be ignored; therefore, we believe that increasing the parameter $b$ will make the performance of the proposed model worse. The reasons for this situation may be that the user’s secondary preferences have little effect on the real score. When the user’s bias feature is added to the model, the user’s secondary preferences are interpreted as part of it; therefore, only the user’s main preference needs to be considered in the model.

### 4.5.3. Impact of $\lambda$ and $k$

The parameter $\lambda$ is used to control the fitting degree, while the parameter $k$ is used to control the proportion of the bias score. The impact of these two parameters on the performance of our model is shown in Figure 2c,d, respectively.

It can be seen from Figure 2c that with the increase of $\lambda$, the MAE value has the same trend on all the three datasets and has reached the minimum value at about $\lambda = 0.6$. A smaller $\lambda$ will lead to over fitting, while a larger $\lambda$ will result in poor fitting results; both of the two cases will lead to poor performance of the model. As can be seen from Figure 2d, when the value of $k$ is small, with the increase of $k$, the value of MAE decreases at a faster rate. This is due to the fact that the smaller value of $k$ means that the bias score in the model is relatively large, which is not consistent with the actual situation. Therefore, an appropriate value of $k$ can make the model more reasonable, so as to get a better performance.
5. Conclusions

In this paper, we propose a novel collaborative filtering method based on the matrix factorization technique. The method considers that the user’s final rating score has been affected by the user’s rating characteristics, which makes the ratings not fully representative of the user’s real preferences. Because the matrix factorization method is suitable for the mining of features, we use it to explore the users’ rating characteristics through their historical rating behaviors. It helps estimate the degree of deviation when they make a new rating and the prediction results are more in line with the actual situation. Finally, the experimental results on three real datasets show that the proposed method can significantly improve prediction accuracy. We think that the content that the user recommends causes a user bias; thus, we will consider the recommendation context’s influence on the user bias in future work. Moreover, we will also discuss the dynamic rating characteristic that ensures that at different times, the user’s rating characteristics will be different, which is more in line with actual user behavior.

Author Contributions: All the authors discussed the algorithm required to complete the manuscript. J.Z. and G.S. conceived the paper and performed the experiments. J.Z. discussed the impact of the parameters and revised the paper.

Funding: The research was funded by the National Natural Science Foundation of China (Grant No. 61672006) in part, Natural Science Foundation Anhui Provincial Education Department (Grant No. KJ2017A332, KJ2018A0328) in part and Natural Science Foundation of Anhui Province (Grant No. 1808085QF209) in part. And The APC was funded by Natural Science foundation Anhui Provincial Education Department (Grant No. KJ2017A332).

Conflicts of Interest: The authors declare no conflict of interest.

References

1. Felfernig, A.; Isak, K.; Szabo, K.; Zachar, P. The VITA financial services sales support environment. In Proceedings of the National Conference on Innovative Applications of Artificial Intelligence, Vancouver, BC, Canada, 22–26 July 2007; AAAI Press: Menlo Park, CA, USA, 2007; pp. 1692–1699.
2. Petrevska, B.; Koceski, S. Tourism recommendation system: Empirical investigation. Revista De Turism Studii Si Cercetari in Turism 2012, 11, 11–18.
3. Gomez-Uribe, C.A.; Hunt, N. The Netflix Recommender System. ACM Trans. Manag. Inf. Syst. 2015, 6, 1–19. [CrossRef]
4. Schaefer, J.B.; Dan, F.; Herlocker, J.; Sen, S. Collaborative Filtering Recommender Systems. The Adaptive Web; Springer: Berlin/Heidelberg, Germany, 2007; pp. 291–324.
5. Su, X.; Khoshgoftaar, T.M. A survey of collaborative filtering techniques. Adv. Artif. Intell. 2009, 2009, 421425. [CrossRef]
6. Huang, Z.; Zeng, D.; Chen, H. A comparison of collaborative-filtering recommendation algorithms for e-commerce. IEEE Intell. Syst. 2007, 22, 68–78. [CrossRef]
7. Stai, E.; Kafetzoglou, S.; Tsiropoulou, E.E.; Papavassiliou, S. A holistic approach for personalization, relevance feedback & recommendation in enriched multimedia content. Multimedia Tools Appl. 2016, 77, 1–44.
8. Pouli, V.; Kafetzoglou, S.; Tsiropoulou, E.E.; Dimitriou, A.; Papavassiliou, S. Personalized multimedia content retrieval through relevance feedback techniques for enhanced user experience. In Proceedings of the 13th International Conference on Telecommunications (ConTEL), Graz, Austria, 13–15 July 2015; pp. 1–8.
9. Takacs, G.; Pilaszy, I.; Nemeth, B.; Tikk, D. Investigation of various matrix factorization methods for large recommender systems. In Proceedings of the 2nd KDD Workshop on Large-Scale Recommender Systems and the Netflix Prize Competition, Las Vegas, NV, USA, 24–27 August 2008; ACM: New York, NY, USA, 2008; p. 6.
10. Koren, Y.; Bell, R.; Volinsky, C. Matrix factorization techniques for recommender systems. Computer 2009, 42, 8. [CrossRef]
11. Breese, J.S.; Heckerman, D.; Kadie, C. Empirical analysis of predictive algorithms for collaborative filtering. In Proceedings of the Fourteenth conference on Uncertainty in artificial intelligence, Madison, WI, USA, 24–26 July 1998; Morgan Kaufmann Publishers Inc.: San Francisco, CA, USA, 1998; pp. 43–52.
12. Herlocker, J.L.; Konstan, J.A.; Borchers, A.; Riedl, J. An algorithmic framework for performing collaborative filtering. In Proceedings of the 22nd annual international ACM SIGIR conference on Research and
development in information retrieval, Berkeley, CA, USA, 15–19 August 1999; ACM: New York, NY, USA, 1999; pp. 230–237.

13. Sarwar, B.; Karypis, G.; Konstan, J.; Riedl, J. Item-based collaborative filtering recommendation algorithms. In Proceedings of the 10th international conference on World Wide Web, Hong Kong, China, 1–5 May 2001; ACM: New York, NY, USA, 2001; pp. 285–295.

14. Vucetic, S.; Obradovic, Z. Collaborative filtering using a regression-based approach. Knowl. Inf. Syst. 2005, 7, 1–22. [CrossRef]

15. Xu, B.; Bu, J.; Chen, C.; Cai, D. An exploration of improving collaborative recommender systems via user-item subgroups. In Proceedings of the 21st international conference on World Wide Web, Lyon, France, 16–20 April 2012; ACM: New York, NY, USA, 2012; pp. 21–30.

16. O’Connor, M.; Herlocker, J. Clustering items for collaborative filtering. In Proceedings of the ACM SIGIR workshop on recommender systems, Berkeley, CA, USA, 19 August 1999; p. 128.

17. Miyahara, K.; Pazzani, M. Collaborative filtering with the simple Bayesian classifier. In PRICAI 2000 Topics in Artificial Intelligence, Proceedings of the Pacific Rim International Conference on Artificial Intelligence, Melbourne, VIC, Australia, 28 August–1 September 2000; Lecture Notes in Computer Science; Springer: Berlin/Heidelberg, Germany, 2000; Volume 1886.

18. Mori, K.; Nguyen, T.; Harada, T.; Thawonmas, R. An Improvement of Matrix Factorization with Bound Constraints for Recommender Systems, Advanced Applied Informatics (IIAI-AAI). In Proceedings of the 5th IIAI International Congress on IEEE, Kumamoto, Japan, 10–14 July 2016; pp. 103–106.

19. Kunaver, M.; Fajfar, I. Grammatical Evolution in a Matrix Factorization Recommender System. In Proceedings of the International Conference on Artificial Intelligence and Soft Computing, Zakopane, Poland, 12–16 June 2016; pp. 392–400.

20. Sarwar, B.; Karypis, G.; Konstan, J.; Riedl, J. Application of Dimensionality Reduction in Recommender System-A Case Study; DTIC Document. Technology Report; Minnesota Univ Minneapolis Dept of Computer Science: Minneapolis, MN, USA, 2000.

21. Lee, D.D.; Seung, H.S. Algorithms for non-negative matrix factorization. Adv. Neural Inf. Process. Syst. 2001, 556–562.

22. Yu, K.; Zhu, S.; Lafferty, J.; Gong, Y. Fast nonparametric matrix factorization for large-scale collaborative filtering. In Proceedings of the 32nd international ACM SIGIR conference on Research and development in information retrieval, Boston, MA, USA, 19–23 July 2009; ACM: New York, NY, USA, 2009; pp. 211–218.

23. Srebro, N.; Rennie, J.D.; Jaakkola, T.S. Maximum-margin matrix factorization. In Proceedings of the Conference on Neural Information Processing Systems, Montreal, QC, Canada, 8–13 December 2004; Volume 17, pp. 1329–1336.

24. Zhang, G.; He, M.; Wu, H.; Cai, G.; Ge, J. Non-negative multiple matrix factorization with social similarity for recommender systems. In Proceedings of the 3rd IEEE/ACM International Conference on Big Data Computing, Applications and Technologies, Shanghai, China, 6–9 December 2016; ACM: New York, NY, USA, 2016; pp. 280–286.

25. Liang, D.; Altosaar, J.; Charlin, L.; Blei, D.M. Factorization meets the item embedding: Regularizing matrix factorization with item co-occurrence. In Proceedings of the 10th ACM Conference on Recommender Systems, Boston, MA, USA, 15–19 September 2016; ACM: New York, NY, USA, 2016; pp. 59–66.

26. Wibowo, A.T. Generating pseudotransactions for improving sparse matrix factorization. In Proceedings of the 10th ACM Conference on Recommender Systems, Boston, MA, USA, 15–19 September 2016; ACM: New York, NY, USA, 2016; pp. 439–442.

27. Ortega, F.; Hernando, A.; Bobadilla, J.; Kang, J.H. Recommending items to group of users using Matrix Factorization based Collaborative Filtering. Inf. Sci. 2016, 345, 313–324. [CrossRef]

28. Adamopoulos, P.; Tuzhilin, A. On Unexpectedness in Recommender Systems: Or How to Better Expect the Unexpected. ACM Trans. Intell. Syst. Technol. 2014, 5, 1–32. [CrossRef]

29. Wang, J.; Liu, R.; Liu, Y. Non-negative Matrix Factorization Algorithm with Bias in Recommender System. J. Chin. Comput. Syst. 2018, 39, 69–73.

30. Koren, Y. Collaborative filtering with temporal dynamics. Commun. ACM 2010, 53, 89–97. [CrossRef]

31. Herlocker, J.L.; Konstan, J.A.; Terveen, L.G.; Riedl, J.T. Evaluating collaborative filtering recommender systems. ACM Trans. Inf. Syst. 2004, 22, 5–53. [CrossRef]
32. Bedi, P.; Gautam, A.; Sharma, C. Using novelty score of unseen items to handle popularity bias in recommender systems. In Proceedings of the International Conference on Contemporary Computing and Informatics, Mysore, India, 27–29 November 2014; pp. 934–939.
33. Abdollahpouri, H.; Burke, R.; Mobasher, B. Controlling Popularity Bias in Learning-to-Rank Recommendation. In Proceedings of the Eleventh ACM Conference on Recommender Systems, Como, Italy, 27–31 August 2017; ACM: New York, NY, USA.
34. Collins, A.; Tkaczyk, D.; Aizawa, A.; Beel, J. Position Bias in Recommender Systems for Digital Libraries. In Transforming Digital Worlds, Proceedings of the Transforming Digital Worlds Conference, Sheffield, UK, 25–28 March 2018; Chowdhury, G., McLeod, J., Gillet, V., Willett, P., Eds.; Lecture Notes in Computer Science; Springer: Cham, Switzerland, 2018.
35. Salakhutdinov, R.; Mnih, A. Probabilistic Matrix Factorization. NIPS 2007, 1, 1–2.
36. Liu, J.; Jiang, Y.; Li, Z.; Lu, H. Domain-sensitive recommendation with user-item subgroup analysis. IEEE Trans. Knowl. Data Eng. 2016, 28, 939–950. [CrossRef]
37. Kumar, V.; Pujari, A.K.; Sahu, S.K.; Kagita, V.R.; Padmanabhan, V. Proximal maximum margin matrix factorization for collaborative filtering. Pattern Recognit. Lett. 2017, 86, 62–67. [CrossRef]
38. Hernando, A.; Bobadilla, J.; Ortega, F. A non-negative matrix factorization for collaborative filtering recommender systems based on a Bayesian probabilistic model. Knowl. Based Syst. 2016, 97, 188–202. [CrossRef]
39. Liao, C.L.; Lee, S.J. A clustering based approach to improving the efficiency of collaborative filtering recommendation. Electron. Commer. Res. Appl. 2016, 18, 1–9. [CrossRef]
40. Cai, Y.; Leung, H.; Li, Q.; Min, H.; Tang, J.; Li, J. Typicality-based collaborative filtering recommendation. IEEE Trans. Knowl. Data Eng. 2014, 26, 766–779. [CrossRef]

© 2018 by the authors. Licensee MDPI, Basel, Switzerland. This article is an open access article distributed under the terms and conditions of the Creative Commons Attribution (CC BY) license (http://creativecommons.org/licenses/by/4.0/).