International Journal of Machine Learning and Cybernetics
https://doi.org/10.1007/s13042-021-01490-7

ORIGINAL ARTICLE

Improving recommendation quality through outlier removal

Yuan-Yuan Xu[1] · Shen-Ming Gu[2] · Fan Min[1]

Received: 27 May 2021 / Accepted: 6 December 2021

Abstract
Rating data collected by recommendation systems contain noise caused by human uncertainty and malicious attacks. Existing outlier removal approaches usually aim at detecting noise inserted into ground-truth ratings. However, in real applications, the ground-truth of the training data are unavailable, or even unimportant for the prediction task. In this paper, we propose an efficient and effective outlier removal algorithm to improve the quality of the training data. The noise is modeled by the mixture of Gaussian distribution, which can approximate any continuous distribution. First, we employ the expectation-maximization algorithm to calculate the low-rank matrices, whose product forms the recovered ratings. Second, we compare the original and recovered ratings to solicit suspected outliers. This process is repeated a number of times, and ratings that are suspected enough times will be treated as outliers. To validate the effectiveness of our algorithm, we compared the prediction quality of four popular recommendation algorithms. Results showed that several measures on the algorithms were improved with the new training data.

Keywords Matrix factorization · Outlier Removal · Recommender systems

1 Introduction

Human uncertainty [1, 2] and malicious attacks [3, 4] are common in real-world recommender systems (RSs) [5–7]. On the one hand, human uncertainty is usually associated with human fallibility and reflexivity [8, 9]. Fallibility makes it difficult for users to accurately express their true opinions [9]. Reflexivity prevents users from expressing their opinions steadily at different times [2]. On the other hand, malicious attacks [10, 11] take shilling and unorganized malicious forms. Shilling attacks choose the fixed target items, the fixed number of rated items, and the fixed attack functions [12]. Unorganized malicious attacks are not constrained to the same attack strategies [11, 13]. Naturally, these unreal ratings are likely to decrease the quality of the data, and give rise to misunderstanding of customer needs.

Consequently, there are two main types of strategies to locate contaminated data . The first strategy assumes that some users are uncertain and therefore they provide unreliable data [8]. O'Mahony et al. [5] employed the collaborative filtering (CF) algorithm to obtain the predictions of targets. User uncertainty is then defined as a normalized loss between the predictions and original ratings. Jasberg et al. [14] obtained the ground-truth rating through repeated ratings for each user-item pair. The variance of user multiple ratings is calculated to evaluate the user uncertainty. The second strategy assumes that the data is contaminated by malicious attacks [11]. The attack profiles are to fill the empty entries in the datasets on the assumption that the original data are authentic and clean. It leads to a reordering of the recommendation list and a change in the ranking of the target items [15]. Pang et al. [11] proposed an approach related to the augmented Lagrangian method to handle malicious attack detection. Williams et al. [16] employed a number of attributes to distinguish the attack profiles from genuine data.

Outlier removal is a technique for coping with data contamination. O'Mahony et al. [5] and Yap et al. [17] removed

✉ Fan Min
minfan@swpu.edu.cn

Yuan-Yuan Xu
yuanyuanxu.cn@gmail.com

Shen-Ming Gu
gushenming666@163.com

[1] School of Computer Science, Southwest Petroleum University, Chengdu 610500, People's Republic of China

[2] Key Laboratory of Oceanographic Big Data Mining & Application of Zhejiang Province, Zhejiang Ocean University, Zhoushan 316022, People's Republic of China

Published online: 19 January 2022

dirty ratings from the training data. Li et al. [18] and Kim et al. [19] cleansed all ratings of an unreliable user from the original dataset. These approaches require a pre-defined threshold of rating deviation. Pang et al. [11] detected the positions of attack fillers in the datasets and recovered the ground-truth rating matrix. Williams et al. [16] classified the data as either attacks or authentic ones through CF algorithms. Both Pang et al. [11] and Williams et al. [16] need to inject attack data into the original dataset to test the performance of algorithms.

In this paper, we propose an efficient and effective outlier removal algorithm to improve the quality of the training data. Our algorithm is different from the aforementioned ones in at least two aspects. One is that we do not need to artificially set a threshold for the determination of outliers. We use a probability-learning based technique for this purpose instead. The other is that we only remove outliers from the training data. Hence, the testing data is not influenced and the evaluation measure is rational.

We model the noise with the mixture of Gaussian (MoG) [20, 21], which can approximate any continuous distribution. MoG is widely applied in image denoising and RS clustering. For image denoising [20, 22], it acts as the approximator of unknown noise. For RS clustering [23, 24], different variances are used to obtain different clusters. In our approach, user uncertainty is expressed as the different preference meaning between the original and the recovered ratings. To the best of our knowledge, there is little work similar to ours.

First, we employ the expectation-maximization (EM) algorithm [25] to obtain the low-rank matrices and the parameters of the MoG model. The EM algorithm includes two steps: the expectation step (E step) and the maximization step (M step). In the E step, we learn the variances and the latent variables of noise components automatically. The variance of each noise component represents the degree of noise discreteness. The latent variable determines the component to which the rating noise belongs. In the M step, an optimization objective function based on the maximization likelihood estimation (MLE) is designed [26]. We update the low-rank matrices by alternative least squares to maximize the likelihood of the function. The product of the low-rank matrices is the recovered rating matrix.

Second, we compare the original and the recovered ratings to judge outliers. If an original rating and its corresponding recovered rating indicate different preferences, then the original one is suspected to be an outlier. Due to the uncertain nature of the EM algorithm, we run it a number of times. We learn a probability to determine whether to treat a rating as an outlier.

We choose the MOG model to cope with the following characteristics of the rating data. a) Data sparsity. Each user rates only a small proportion of items, hence most of the data in the rating matrix are unknown [27]. b) Data imbalance. Popular items are more likely to be purchased, and a small part of users are keen on rating, which leads to a long tail effect [28]. c) Data inconsistency. Tolerant and strict users have different rating habits. The former may give 5 points to good or excellent items, while the latter may give 5 points only to excellent ones. These reasons lead to unknown parameters and the distribution of noise in datasets. The MoG model is a universal approximator for any continuous distribution, the parameters of which can be estimated by the EM algorithm [22]. So we employ them to handle outlier detection.

We employ matrix factorization (MF) and k-nearest neighbor (kNN) algorithms as evaluation approaches. They take the new training data as input. Experiments were undertaken on four well-known datasets: MovieLens 100K,[1] Amazon,[2] Yelp,[3] and FilmTrust.[4] The source code and data are available at https://github.com/FansSmale/ORRS. Results showed that several measures on all tested approaches were improved with the new training data.

The rest of the paper is organized as follows. Section 2 reviews the related work, including popular MF algorithms, and outlier detection. Section 3 elaborates our approach to detect outliers. Section 4 shows extensive experiments to validate the effectiveness of our approach. Section 5 concludes the paper.

2 Related work

In this section, we revisit the recommender system, MF algorithms and existing outlier detection approaches for recommender systems. The rating system is the fundamental data model of recommender systems. MF algorithms support recommendations by filling in missing values in the rating system. They will be employed to compare the quality of the data before and after outlier removal. Table 1 lists notations used throughout the paper.

2.1 Recommender system

RSs are classified into memory-based and model-based ones based on whether historical data are required for recommendation. Memory-based RSs use similarity to find neighbors of users or items for recommendations. Based on users' similarity, an adapted kNN [29] only considered users who tagged the query resource. Based on items' similarity, an efficient CF recommendation [30] designed a multi-channel

[1] http://grouplens.org/datasets/movielens/

[2] http://snap.stanford.edu/data/web-Amazon-links.html

[3] https://www.yelp.com/dataset

[4] https://www.librec.net/datasets.html

Table 1 Notations

Notation	Meaning
$\mathbf{X}$	The original training rating matrix
m	The number of users
n	The number of items
$\mathbf{x}_i$	The ith row of $\mathbf{X}$
$\mathbf{x}^j$	The jth column of $\mathbf{X}$
x_{ij}	The rating for u_i to t_j in $\mathbf{X}$
$\mathbf{X}'$	The recovered matrix
x'_{ij}	The rating for u_i to t_j in $\mathbf{X}'$
K	The number of noise
$\mathbf{E}$	The noise matrix
$\mathbf{P}$	The first low-rank matrix
$\mathbf{Q}$	The second low-rank matrix
r	The rank of low-rank matrices
$\mathbf{\Gamma}$	The noise distribution matrix
γ_{ijk}	The posterior probability of x_{ij} belonging to k-th component
$\mathbf{X}''$	The new training rating matrix
ρ	The like threshold
λ	The threshold for identifying outliers

to calculate the item similarity. Model-based RSs mainly includes matrix factorization (MF)-based, singular value decomposition (SVD)-based and cluster-based ones [31]. MF employs the product of two low-rank matrices to approximate the rating matrix [32]. A novel recommendation [33] employed SVD to reduce the dimension to overcome the sparsity and scalability problems. A model-based recommendation [34] considered the dual error in the training set between users' and items' similarity. RSs can also be divided into two-way recommendation and three-way recommendation according to the recommendation behavior [27, 35]. The two-way behaviors are "recommend" and "not recommend". Three-way recommendation increases promotion behavior on the basis of two-way recommendation [36, 37].

Rating data are used as a core mechanism for expressing user interest in items. The rating system usually contains information about users, items, and ratings. Suppose the dataset $\mathbf{X}$ has m users and n items. Let $\mathbf{X} = (x_{ij})_{m\times n}$ be the original training rating matrix. In applications, $\mathbf{X}$ is often quite sparse.

Table 2 lists a training dataset with 100 users and 5 items. The ratings are marked as $\{1, 2, 3, 4, 5\}$ and 0 means that the user has not rated it.

Table 2 A training rating matrix $\mathbf{X}$

User \ item	t_0	t_1	t_2	t_3	t_4
u_0	0	2	0	0	4
u_1	1	0	4	0	0
u_2	0	2	0	0	5
u_3	2	0	0	1	0
u_4	0	3	0	0	5
...	...	...	...	...	...
u_{99}	2	4	0	0	0

2.2 Matrix factorization

MF is a classic mathematical problem [32]. It has achieved success in many machine learning fields, such as image retrieval [38] and face recognition [39]. In practical applications, rating matrices have thousands to billions of users and items [34]. Traditional item- or user-based CF techniques are quite inefficient. The MF takes advantages of the sparse characteristics of rating matrices to decrease the computational complexity [40, 41]. Consequently, it has been one of the mainstreams of CF since 2006 [42].

We will employ three popular MF-based recommendation algorithms to evaluate the quality of the rating matrix. The first one will be referred to as alternative-least-squares optimization based MF (ALS-MF) [43]. It employs the alternative-least-squares optimization to minimize the loss function

$$\arg\min_{\mathbf{H},\mathbf{S}} \sum_{(i,j)\in\Omega} \left\|\mathbf{X}-\hat{\mathbf{X}}\right\|_2^2 + \alpha\left(\sum_{i=0}^{m-1}\|\mathbf{h}_i\|_2 + \sum_{j=0}^{n-1}\left\|\mathbf{s}^j\right\|_2\right), \quad (1)$$

where $\mathbf{H}$ and $\mathbf{S}$ are two low-rank matrices and $\hat{\mathbf{X}} = \mathbf{H}\mathbf{S}^{\mathrm{T}}$. The first term is the difference between the observed ratings and their predicted ratings. The second term is L_2 regularization term to prevent overfitting, and α is the regularization coefficient.

The second one will be referred to as implicit feedback based MF (IFB-MF) [41]. It considers the implicit feedback from the missing ratings. It employs the popularity of an item to consider the potential meaning of an unknown rating. If the user does not rate an item with high popularity, it is likely that the user knows it but does not give it a rating. So it can be inferred that she does not like it. The objective function is designed as

$$\begin{aligned}\arg\min_{\mathbf{H},\mathbf{S}} &\sum_{(i,j)\in\Omega} \left\|\mathbf{X}-\hat{\mathbf{X}}\right\|_2^2 \\ &+ \alpha\left(\sum_{i=0}^{m-1}\|\mathbf{h}_i\|_2 + \sum_{j=0}^{n-1}\left\|\mathbf{s}^j\right\|_2\right) + \sum_{j\notin\tau_i}\psi_j\hat{x}_{ij}^2,\end{aligned} \quad (2)$$

where ψ_j is the normalized popularity of t_j, and τ_i is the set of items rated by u_i. The first and second terms are similar to Eq. (3). The third term considers the implicit feedback from the popularity of the items which are not rated by u_i. This is helpful to reduce the complexity of the algorithm.

The third one will be referred to as dynamic MF (D-MF) [40]. It also considers the impact of missing data and its objective function

$$\begin{aligned}\arg\min_{\mathbf{H},\mathbf{S}} &\sum_{(i,j)\in\Omega}\left\|\mathbf{X}-\hat{\mathbf{X}}\right\|_2^2 \\ &+\alpha\left(\sum_{i=0}^{m-1}\left\|\mathbf{h}_i\right\|_1+\sum_{j=0}^{n-1}\left\|\mathbf{s}^j\right\|_1\right)+\beta\sum_{(i,j)\notin\Omega}\hat{x}_{ij}^2,\end{aligned} \tag{3}$$

where the third term sets a small coefficient β to multiply the predicted values of the unknown ratings. Similarly, it can improve the running speed of the algorithm.

2.3 Outlier detection

There are two mainstream outlier detection approaches in the field of recommender systems. One is to employ the deviation between the original and the predicted ratings to identify outliers [5, 44, 45]. With neighborhood based approaches, the predicted rating is calculated according to the average rating of the user and the ratings of the neighbors. In this case, the deviation is defined as the normalized loss between the original and predicted ratings [5]. With MF-based approaches, the predicted rating is given by the multiplication of the low-rank matrices. In this case, the deviation is defined as the average loss [44, 45].

The other is to identify outliers according to rating consistency [18, 46, 47]. User inconsistency, also referred to as self-contradiction [18], means that a user gives different ratings to closely related items. Group-based recommendations assume that a user group has similar preferences for an item group. User-item group inconsistency means that a user gives different ratings from those of others [46]. Model based recommendations generated an axis-parallel hyperplane according to common neighbors between users. Neighborhood inconsistency means that the distance between a rating and the hyperplane exceeds a predefined threshold.

However, to the best of our knowledge, there is still no approach that considers the noise distribution in recommender systems.

3 The proposed approach

In this section, we first introduce the MoG model to represent the noise of the original dataset. Then we elaborate on how to learn the parameters of MoG using the EM algorithm.

3.1 Modeling MoG noise

Considering the existence of uncertainty, $\mathbf{X}$ is decomposed as

$$\mathbf{X}=\mathbf{X}'+\mathbf{E}, \tag{4}$$

where $\mathbf{E}$ is the noise with the MoG distribution. X' is factorized mathematically as the product of two low-rank matrices:

$$\mathbf{X}'=\mathbf{P}\mathbf{Q}^{\mathrm{T}}, \tag{5}$$

where $\mathbf{P}\in\mathbb{R}^{m\times r}$ and $\mathbf{Q}\in\mathbb{R}^{n\times r}$ are factor matrices, and r is often much smaller than m or n. Let $P=\left[\mathbf{P}_0,\mathbf{P}_2,\ldots,\mathbf{P}_{m-1}\right]$ and $\mathbf{Q}=\left[\mathbf{Q}_0,\mathbf{Q}_2,\ldots,\mathbf{Q}_{n-1}\right]$. The lengths of both $\mathbf{P}_i$ and $\mathbf{Q}_j$ are r. Every entry x_{ij} of $\mathbf{X}$ can be expressed as

$$x_{ij}=x'_{ij}+e_{ij}, \tag{6}$$

where $\mathbf{x'}_{ij}=\mathbf{p}_i(\mathbf{q}_j)^{\mathrm{T}}$.

Assume that the noise obeys MoG distribution, we consider a generative form of mixture noise

$$f^{point}\left(e_{ij}\right)=\sum_{k=0}^{K-1}w_k\mathcal{N}\left(e_{ij}|\mu_k,\sigma_k^2\right), \tag{7}$$

where $f^{point}(\cdot)$ is the probability of noise on each rating. K is the number of MoG components, w_k is the proportion of the k^{th} Gaussian noise component, and

$$\sum_{k=0}^{K-1}w_k=1. \tag{8}$$

Let $\mathcal{N}(e_{ij}|\mu_k,\sigma_k^2)$ be the kth Gaussian distribution with mean value μ_k and variance σ_k^2. Let $\boldsymbol{\Phi}=\left[\phi_0,\phi_1,\ldots,\phi_{K-1}\right]$, where $\phi_k=x'_{ij}+\mu_k$. Let $\Delta=\{\mathbf{P},\mathbf{Q},\mathbf{W},\boldsymbol{\Phi},\boldsymbol{\Sigma}\}$, $\mathbf{W}=\left[w_0,w_2,\ldots,w_{K-1}\right]$, $\boldsymbol{\Sigma}=\left[\sigma_0^2,\sigma_1^2,\cdots,\sigma_{K-1}^2\right]$, and Ω be the set of indexes for all valid ratings. The probability of $\mathbf{X}$ given the parameter Δ is

$$\begin{aligned}f^{data}(\mathbf{X}|\Delta)&=\prod_{(i,j)\in\Omega}f^{point}\left(x_{ij}|\Delta\right)\\&=\prod_{(i,j)\in\Omega}\left(\sum_{k=0}^{K-1}w_k\mathcal{N}(x_{ij}|\phi_k,\sigma_k^2)\right).\end{aligned} \tag{9}$$

Let $\mathcal{L}(\mathbf{X}|\Delta)=\log f(\mathbf{X}|\Delta)$. Our optimization objective is to maximize log-likelihood estimation [26] to compute the parameter Δ, namely,

$$\max_{\Delta}\mathcal{L}(\mathbf{X}|\Delta)=\sum_{(i,j)\in\Omega}\log\left(\sum_{k=0}^{K-1}w_k\mathcal{N}\left(x_{ij}|\phi_k,\sigma_k^2\right)\right). \tag{10}$$

3.2 Parameter learning

Here we discuss how to learn the parameters Δ according to Eq.(10). In particular, we employ the frame of the EM algorithm for this purpose.

Here we employ the EM algorithm [25] to find the value of Δ which maximizes the log-likelihood $\mathcal{L}(\mathbf{X}|\Delta)$ given an observed $\mathbf{X}$. The EM algorithm for estimating the parameters consists of two steps: the expectation step (E step) and the maximization step (M step), and iterates between the two steps until convergency.

In the E step, we compute the responsibility of each noise component. First we randomly initialize Δ, and the number of components K. Then, we compute the noise by Eqs. (4) and (5). The latent variables z_{ijk} is defined for each rating x_{ij} to indicate the assignment of noise e_{ij} to a specific component of the MoG, where $z_{ijk} \in \{0, 1\}$ and $\sum_{k=0}^{K-1} z_{ijk} = 1$. Here, we consider our problem as a Gaussian scale mixtures [48], where μ_k is supposed to be zero. Then we compute the posterior probability $E(z_{ijk})$ of the k^{th} mixture component to generate the noise e_{ij}. This can be represented by

$$E(z_{ijk}) = \gamma_{ijk} = \frac{w_k \mathcal{N}\left(e_{ij}|x'_{ij}, \sigma_k^2\right)}{\sum_{k=0}^{K-1} w_k \mathcal{N}\left(e_{ij}|x'_{ij}, \sigma_k^2\right)}. \tag{11}$$

In the M step, there are two groups of parameters to be updated by MLE. We employ Jensen inequality [49, 50] to obtain the lower bounder of Eq. (10), represented as

$$\sum_{(i,j)\in\Omega} \log\left(\sum_{k=0}^{K-1} w_k \mathcal{N}\left(x_{ij}|x'_{ij}, \sigma_k^2\right)\right) \geq \sum_{(i,j)\in\Omega} \sum_{k=0}^{K-1} \gamma_{ijk} \log \frac{w_k \mathcal{N}\left(x_{ij}|x'_{ij}, \sigma_k^2\right)}{\gamma_{ijk}}. \tag{12}$$

Update the first group: W and Σ. Except the constant terms, the right hand of Eq. (12) is rewritten as

$$\sum_{(i,j)\in\Omega} \sum_{k=0}^{K-1} \gamma_{ijk} \Big(\log w_k - \log \sigma_k - \log \gamma_{ijk} - \frac{1}{2\sigma_k^2}\left(x_{ij} - x'_{ij}\right)\left(x_{ij} - x'_{ij}\right)^{\mathrm{T}}\Big). \tag{13}$$

We maximize Eq. (13) to update the parameters as follows:

$$w_k = \frac{1}{N} \sum_{(i,j)\in\Omega} \gamma_{ijk}, \tag{14}$$

$$\sigma_k^2 = \frac{\sum_{(i,j)\in\Omega} \gamma_{ijk}\left(x_{ij} - x'_{ij}\right)\left(x_{ij} - x'_{ij}\right)^{\mathrm{T}}}{\sum_{(i,j)\in\Omega} \gamma_{ijk}}. \tag{15}$$

As a result, we can update the posterior probability γ_{ijk} in E step.

Update the second group: P and Q. Except the constant terms, the right hand of Eq. (12) is rewritten as

$$\sum_{(i,j)\in\Omega} \sum_{k=0}^{K-1} \gamma_{ijk}\left(-\frac{1}{2\sigma_k^2}\left(x_{ij} - x'_{ij}\right)\left(x_{ij} - x'_{ij}\right)^{\mathrm{T}}\right). \tag{16}$$

We maximize Eq. (16) to update two low-rank matrices

$$\mathbf{p}^j = \left(\left(\mathbf{q}^j\right)^{\mathrm{T}} \mathbf{q}^j\right)^{-1} \cdot \left(\mathbf{C} \circ \mathbf{X}\mathbf{q}^j\right), \tag{17}$$

$$\mathbf{q}^j = \left(\left(\mathbf{p}^j\right)^{\mathrm{T}} \mathbf{p}^j\right)^{-1} \cdot \left(\mathbf{C}^{\mathrm{T}} \circ \mathbf{X}^{\mathrm{T}}\mathbf{p}^j\right), \tag{18}$$

where $\circ$ denotes Hadamard product, $\mathbf{C} = (c_{ij})_{m\times n}$ represents the weight of each valid entry in $\mathbf{X}$, and

$$c_{ij} = \begin{cases} \sqrt{\sum_{k=0}^{K-1} \frac{\gamma_{ijk}}{2\sigma^2}}, & (i,j) \in \Omega; \\ 0, & \text{otherwise.} \end{cases} \tag{19}$$

After a number of iterations between E and M steps, we obtain the factor matrices $\mathbf{P}$ and $\mathbf{Q}$. Next, we will return to the E step to continue the iteration. We terminate the EM algorithm when the average posterior probability between consecutive iterations is below a pre-determined threshold.

3.3 Algorithm description

Algorithm 1 lists the main process of outlier detection. The input includes the training rating matrix $\mathbf{X}$, the iteration times T, the like threshold ρ, the outlier identification thresholds λ. The output is the set of outliers L.

Line 1 initializes the count matrix CT. It indicates the number of times that each rating is viewed as an outlier.

Lines 2 through 12 run MoG T times to update the count matrix.

Line 3 calculates $\mathbf{P}$ and $\mathbf{Q}$ using the EM algorithm with MoG.

Lines 4 through 6 obtains $\mathbf{X}'$ by the product of $\mathbf{P}$ and $\mathbf{Q}$.

Table 3 The time complexity of Algorithm 1

Lines	Complexity
Line 1	$O(\zeta)$
Line 2	$O(T)$
Lines 3	$O(\xi mnr)$
Lines 4–6	$O(mnr)$
Lines 7–11	$O(\zeta)$
Lines 14–18	$O(\zeta)$
Total	$O(\zeta) + O(T)(O(\xi mnr) + O(mnr) + O(\zeta)) + O(\zeta) = O(T\xi mnr)$

Table 4 The noise distribution matrix Γ in one run of EM

Rating \ k	0	1	2
$x_{0,1}$	0.0407	0.1000	0.8593
$x_{0,4}$	0.1499	0.1131	0.7370
$x_{1,0}$	0.2733	0.4538	0.2728
$x_{1,2}$	0.2711	0.2974	0.4315
$x_{2,1}$	0.2014	0.1116	0.6870
...	...	...	...
$x_{99,0}$	0.6714	0.2539	0.0747
$x_{99,1}$	0.4958	0.2361	0.2681

Table 5 The recovered matrix X'

User \ Item	t_0	t_1	t_2	t_3	t_4
u_0	*1.23*	2.72	*2.89*	*3.57*	4.53
u_1	2.11	*1.14*	2.65	*4.51*	*1.34*
u_2	*3.21*	2.13	*2.78*	*1.15*	2.42
u_3	2.70	*0.78*	*1.56*	3.92	*4.78*
u_4	*3.78*	4.56	*2.11*	*4.13*	5.00
...	...	...	...	...	...
u_{99}	1.33	4.21	*2.35*	*1.05*	*3.61*

The Italics indicate that users has not rated the items, and it will be processed as 0 according to Table 2

Lines 7 through 11 calculate the number of times that each rating is suspected as an outlier. If one of x_{ij} and x'_{ij} is less than ρ and the other is greater than ρ, x_{ij} will be regarded as an outlier.

Lines 14 through 18 determine the set of outliers according to previously collected counters.

Line 19 returns L.

Note that we use λT instead of λ. In this way, we need only to set $\lambda \in (0, 1]$.

Algorithm 1 The outlier detection algorithm

Input: The training rating matrix $\mathbf{X}$, the iteration times T, the number of noise K, the outlier identification thresholds λ and the like threshold ρ;
Output: The set of outliers L;

1: $CT = (ct_{ij})_{m\times n} = (\mathbf{0})_{m\times n}$; //The count matrix
2: **for** $(iteration = 1$ to $T)$ **do**
3: calculate $\mathbf{P}$ and $\mathbf{Q}$ using the EM algorithm presented in Section 3.2;
4: **for** $((i, j) \in \Omega)$ **do**
5: calculate $x'_{ij} = \mathbf{P}_i\mathbf{Q}_j^T$;
6: **end for**
7: **for** $((i, j) \in \Omega)$ **do**
8: **if** $(x'_{ij} < \rho$ **and** $x_{ij} > \rho)$ || $(x'_{ij} > \rho$ **and** $x_{ij} < \rho)$ **then**
9: ct_{ij} ++; //x_{ij} is a suspiciously noisy rating
10: **end if**
11: **end for**
12: **end for**
13: $L = \emptyset$; //Store outlier indices
14: **for** $((i, j) \in \Omega)$ **do**
15: **if** $(ct_{ij} \geq \lambda T)$ **then**
16: $L = L \cup \{(i, j)\}$; //We are quite sure that x_{ij} is an outlier.
17: **end if**
18: **end for**
19: return L;

Table 6 The count matrix CT

User \ Item	t_0	t_1	t_2	t_3	t_4
u_0	0	1	0	0	3
u_1	1	0	0	0	0
u_2	0	2	0	0	1
u_3	0	0	0	1	0
u_4	0	1	0	0	0
...	...	...	...	...	...
u_{99}	0	0	0	0	0

Table 7 The new training data $\mathbf{X}''$. The bold zero indicates the removed rating

User \ Item	t_0	t_1	t_2	t_3	t_4
u_0	0	2	0	0	**0**
u_1	1	0	2	0	0
u_2	0	**0**	0	0	5
u_3	2	0	0	1	0
u_4	0	3	0	0	5
...	...	...	...	...	...
u_{99}	2	4	0	0	0

3.4 Time complexity analysis

Our algorithm include running the EM algorithm T times, recording the number of times that each rating is suspected of being an outlier, and finally calculating the set of outliers. We will elaborate the time complexity of our algorithm as follows.

Proposition 1 *The time complexity of our algorithm is $O(T\xi mnr)$. ξ is the iteration times of the EM algorithm.*

Proof Suppose ζ is the number of valid ratings in $\mathbf{X}$, which is much smaller than $m \times n$. Line 1 in Algorithm 1 initializes the count matrix CT with $O(\zeta)$ of time. Lines 2 through 12 run T times to update the count matrix. Line 3 calculates $\mathbf{P}$, $\mathbf{Q}$, and other parameters using the EM algorithm with MoG, and takes $O(\xi mnr)$ of time. Line 4 through 6 obtain $\mathbf{X}'$ by the product of $\mathbf{P}$ and $\mathbf{Q}$ with $O(mnr)$ of time. Lines 7 through 11 calculate the number of times that each rating is suspected as an outlier with $O(\zeta)$ of time. Lines 14 through 18 determine the set of outliers according to previously collected counters with $O(\zeta)$ of time. Hence, the total time complexity is $O(\zeta) + O(T)(O(\xi mnr) + O(mnr) + O(\zeta)) + O(\zeta) = O(T\xi mnr)$ This completes the proof. Table 3 describes the complexity of Algorithm 1 briefly. □

3.5 A running example

Based on Table 2, we depict our proposed algorithm through a running example. There are three steps as follows.

Table 8 Characteristics of the datasets used

Datasets	Users	Items	Ratings
MovieLens 100 K	943	1682	100, 000
Amazon	1094	1673	34, 120
Yelp	11, 916	3815	133, 958
FilmTrust	1508	2071	35, 497

First, the number of components K is set as 3. We employ EM algorithm to compute the variances $\mathbf{\Sigma} = [\sigma_0^2, \sigma_1^2, \sigma_2^2]$ and the noise distribution matrix $\mathbf{\Gamma}$ of MoG. One run of EM produces the variances $\mathbf{\Sigma} = [0.2955, 0.7334, 1.1300]$. Table 4 lists a part of the noise matrix $\mathbf{\Gamma} \in \mathbb{R}^{100\times 3}$. To compute $\mathbf{X}'$, we use Eq. (5), as shown in Table 5.

According to line 8 of Algorithm 1, x_{24}, x_{33}, x_{41} are viewed as outliers in this run.

Second, Table 6 lists the count matrix of Algorithm 1 by setting $T = 5$ and $\lambda = 0.4$. Here we observe that x_{04} and x_{21} are suspected of being outliers no less than 2 times. Consequently, both of them are identified as outliers.

Finally, we construct the new training data by removing the outliers x_{04} and x_{21} from Table 2. The bold zero indicates the removed rating as shown in Table 7.

4 Experiments

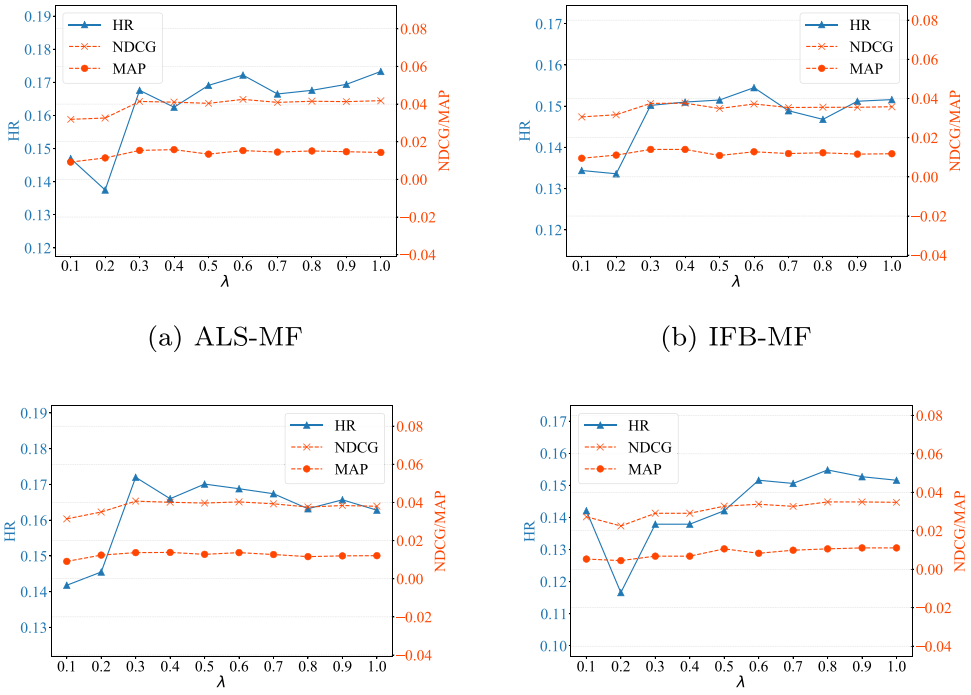
In this section, we present the experiment results to answer the following questions.

(1) Are there some hyper-parameter settings suitable for different datasets?
(2) Is the runtime consistent with the time complexity analysis?
(3) What is the distribution of outliers across rating levels?
(4) Is the proposed approach effective in enhancing the data quality?

Experiments are undertaken on MovieLens 100 K, Amazon, Yelp and FilmTrust datasets, which are widely used to test recommender systems [51–53]. Table 8 summarizes the characteristics of datasets in our experiments. Taking MovieLens 100 K as an example, there are 943 users, 1,682 items, and 100,000 valid ratings. We divide it into 80% training set and 20% testing set.

4.1 Parameter settings

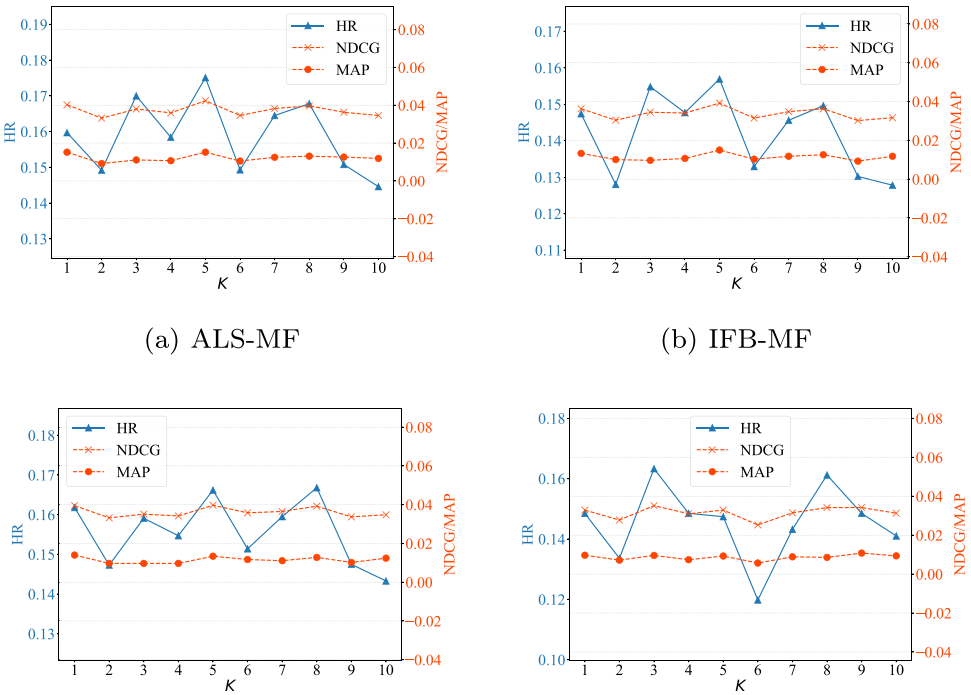
Our algorithm has two parameters: the outliers identification threshold λ and the number of noises K. We will elaborate them as follows.



(a) ALS-MF (b) IFB-MF

(c) D-MF (d) kNN

Fig. 1 The impact of λ to three measures (i.e., HR, NDCG and MAP) on Movielens 100 K. The blue ordinate on the left is the value of HR, and the red ordinate on the right is the value of NDCG and MAP



(a) ALS-MF (b) IFB-MF

(c) D-MF (d) kNN

Fig. 2 The impact of K to three measures (i.e., HR, NDCG and MAP) on Movielens 100 K. The blue ordinate on the left is the value of HR, and the red ordinate on the right is the value of NDCG and MAP

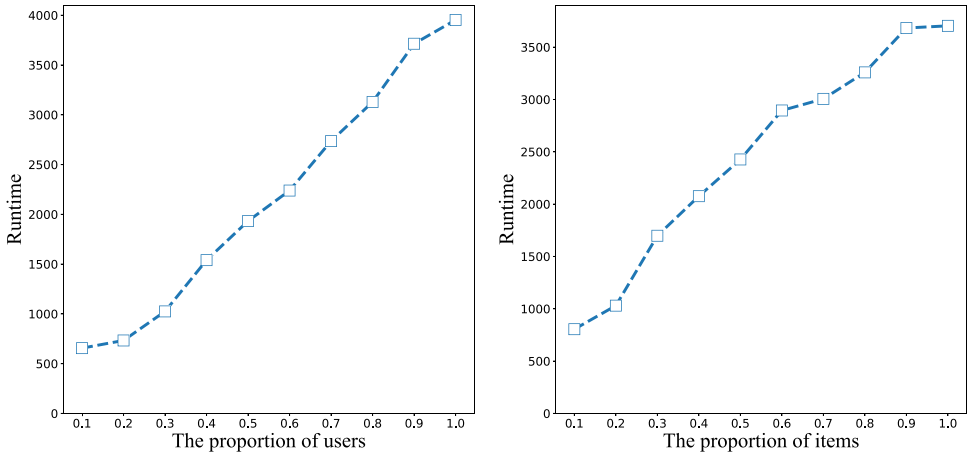


Fig. 3 Runtime on Movielens 100 K

(a) Different proportions of users (b) Different proportions of items

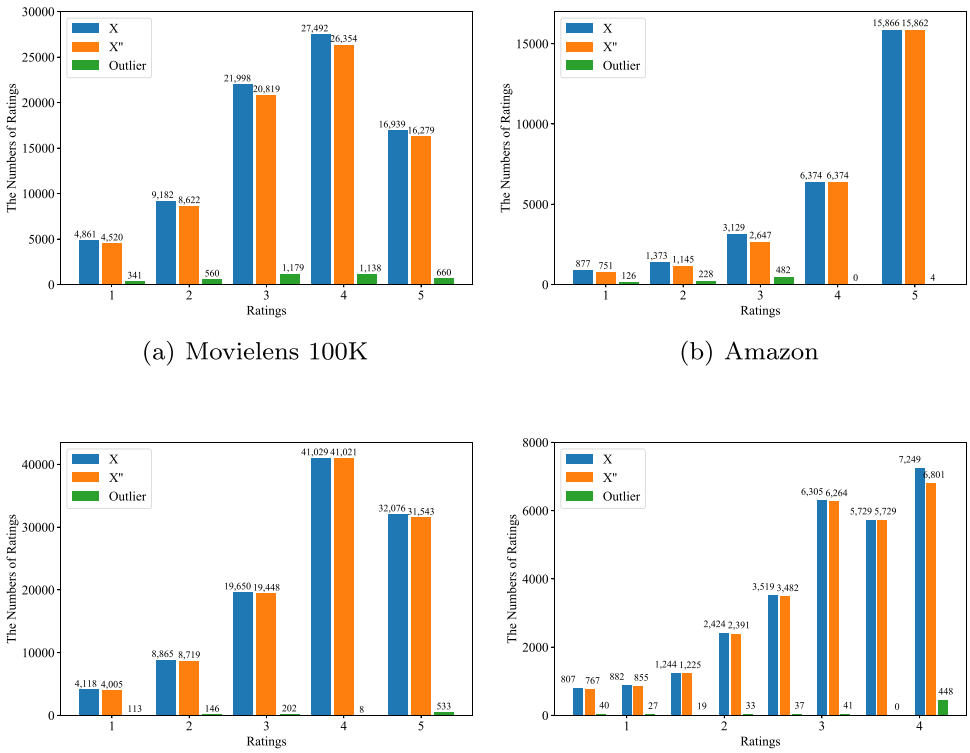


Fig. 4 Comparison of distributions before and after outlier removal

(a) Movielens 100K (b) Amazon

(c) Yelp (d) FilmTrust

4.1.1 The setting of λ

Figure 1 shows the change of three measures for different λ values settings. Figure 1a and b indicate that the peaks are obtained when λ is 0.6 in most cases. According to Fig. 1c and d, the peaks can be achieved at $\lambda = 0.3$ and $\lambda = 0.8$, respectively. The values of three measures at $\lambda = 0.6$ is not much different from it. So, $\lambda = 0.6$ is the best for Movielens 100 K. In the same way, 0.9, 0.9 and 0.7 are generally best for Amazon, Yelp, and FilmTrust, respectively. In summary, there is no universal best setting for all datasets on different measures. We may obtain the setting with the help of the validation set.

4.1.2 The setting of K

Figure 2 shows that three measures vary with K. We fix λ to study the impact of K to the performance. We find that when $K \in \{2, 4, 6, 9\}$, the value of HR is relatively low. NDCG and MAP are much more stable than HR. The reasons are

Table 9 Data quality comparison by our approach. "Orig" stands for the original training set, "New" stands for the new training set. Average values are shown alongside their standard deviation over 20 runs

	ALS-MF		IFB-MF		D-MF		kNN	
	Orig	New	Orig	New	Orig	New	Orig	New
(a) HR								
Movielens 100 K	0.1572 ± 0.0100	0.1720 ± 0.0060	0.1328 ± 0.0064	0.1510 ± 0.0050	0.1663 ± 0.0093	0.1669 ± 0.0121	0.1516	0.1517
Amazon	0.0772 ± 0.0014	0.0783 ± 0.0003	0.0715 ± 0.0033	0.0731 ± 0.0042	0.0747 ± 0.0050	0.0725 ± 0.0096	0.0750	0.0750
Yelp	0.039 ± 30.0015	0.0396 ± 0.0007	0.0654 ± 0.0011	0.0657 ± 0.0023	0.0687 ± 0.0188	0.0757 ± 0.0065	0.0213	0.0212
FilmTrust	0.2959 ± 0.0038	0.3103 ± 0.0044	0.1590 ± 0.0090	0.1741 ± 0.0010	0.3869 ± 0.0066	0.3891 ± 0.0044	0.3395	0.3634
	ALS-MF		IFB-MF		D-MF		kNN	
	Orig	New	Orig	New	Orig	New	Orig	New
(b) NDCG								
Movielens 100 K	0.0336 ± 0.0017	0.0420 ± 0.0014	0.0368 ± 0.0035	0.0400 ± 0.0017	0.0378 ± 0.0014	0.0401 ± 0.0031	0.0348	0.0327
Amazon	0.0156 ± 0.0003	0.0164 ± 0.0007	0.0146 ± 0.0008	0.0147 ± 0.0004	0.0153 ± 0.0013	0.0150 ± 0.0021	0.0153	0.0150
Yelp	0.0160 ± 0.0004	0.0161 ± 0.0006	0.0075 ± 0.0003	0.0077 ± 0.0002	0.0160 ± 0.0054	0.0180 ± 0.0016	0.0042	0.0043
FilmTrust	0.0656 ± 0.0004	0.0705 ± 0.0005	0.0964 ± 0.0006	0.1026 ± 0.0005	0.1160 ± 0.0030	0.1242 ± 0.0030	0.0909	0.1003
	ALS-MF		IFB-MF		D-MF		kNN	
	Orig	New	Orig	New	Orig	New	Orig	New
(c) MAP								
Movielens 100 K	0.0096 ± 0.0027	0.0133 ± 0.0015	0.0125 ± 0.0012	0.0119 ± 0.0010	0.0112 ± 0.0012	0.0121 ± 0.0019	0.0111	0.0090
Amazon	0.0033 ± 0.0003	0.0041 ± 0.0010	0.0032 ± 0.0006	0.0034 ± 0.0003	0.0038 ± 0.0009	0.0038 ± 0.0010	0.0036	0.0040
Yelp	0.0014 ± 0.0001	0.0015 ± 0.0001	0.0057 ± 0.0003	0.0057 ± 0.0006	0.0060 ± 0.0026	0.0060 ± 0.0016	0.0008	0.0090
FilmTrust	0.0488 ± 0.0008	0.0527 ± 0.0006	0.0404 ± 0.0005	0.0428 ± 0.0006	0.0555 ± 0.0021	0.0604 ± 0.0030	0.0351	0.0405
	ALS-MF		IFB-MF		D-MF		kNN	
	Orig	New	Orig	New	Orig	New	Orig	New
(d) RMSE								
Movielens 100 K	1.0819 ± 0.0043	1.0893 ± 0.0046	1.0580 ± 0.0027	1.0684 ± 0.0039	1.0611 ± 0.0053	1.0703 ± 0.0045	1.0570	1.0550
Amazon	0.7502 ± 0.0002	0.7862 ± 0.0004	0.7617 ± 0.0073	0.7966 ± 0.0354	0.6967 ± 0.0124	0.7417 ± 0.0045	0.9349	0.9440
Yelp	1.0541 ± 0.0031	1.0546 ± 0.0037	1.0325 ± 0.0036	1.0325 ± 0.0038	1.0560 ± 0.0041	1.0550 ± 0.0036	1.0428	1.0427
FilmTrust	0.9296 ± 0.0006	0.9313 ± 0.0006	0.9457 ± 0.0007	0.9463 ± 0.0007	0.9004 ± 0.0055	0.8996 ± 0.0011	1.0120	1.0081

as follows: (1) HR measures whether the ground-truth item presents on the recommended list. (2) The rating matrix is sparse and MF has a certain degree of randomness. Therefore, a small difference in the predicted value of the same rating will cause a dramatic change in the ranking list.

4.2 The efficiency of our algorithm

Figure 3a and b demonstrate the runtime for different proportions of users and items on Movielens 100 K. Experiments are implemented in Java and run on the same machine (Intel i5-4460 CPU and 4GB RAM). The runtime increased linearly wrt. the number of users or items. It validated the overall complexity of $O(T\xi mnr)$. The different numbers of users or items will lead to different ξ, so we set $T = 1$ and $\xi = 1$ here.

Figure 3a shows that when the proportion of users changes from 0.1 to 1.0, the runtime increases from 656 to 3953 milliseconds. For example, when the proportion of users was 1.0, the total runtime was 3953 milliseconds. It did not include the overhead operations such as data loading and space allocation. We also noticed that when the user proportion increased by 0.1, the runtime increased by 366 miliseconds on average. Figure 3b shows that when the proportion of items changes from 0.1 to 1.0, the runtime increases from 805 to 3704 milliseconds. When the proportion of items increased by 0.1, the runtime increased by 322 miliseconds on average.

Table 10 The data quality improvement of our approach with impact on popular recommendation algorithms. ↑ indicates that the measure value increases and ↓ indicates that it decreases

	ALS-MF	IFB-MF	D-MF	kNN
(a) HR (win = 13, lose = 2, tie = 1)				
Movielens 100 K	↑9.41%	↑8.43%	↑0.36%	↑0.07%
Amazon	↑1.42%	↑2.24%	↓2.95%	0.00%
Yelp	↑0.76%	↑0.46%	↑10.19%	↓0.47%
FilmTrust	↑4.87%	↑10.50%	↑6.20%	↑ 7.04%
	ALS-MF	IFB-MF	D-MF	kNN
(b) NDCG (win = 14, lose = 2, tie = 0)				
Movielens 100 K	↑ 22.02%	↑14.29%	↑6.08%	↑6.03%
Amazon	↑ 5.13%	↑0.68%	↓1.96%	↓1.96%
Yelp	↑ 0.62%	↑2.67%	↑12.50%	↑2.38%
FilmTrust	↑ 7.47%	↑6.43%	↑7.07%	↑10.34%
	ALS-MF	IFB-MF	D-MF	kNN
(c) MAP (win = 13, lose = 2, tie = 1)				
Movielens 100 K	↑38.54%	↓4.80%	↑8.04%	↓18.92%
Amazon	↑24.24%	↑6.25%	↑2.94%	↑11.11%
Yelp	↑7.14%	0.00%	↑11.67%	↑12.50%
FilmTrust	↑7.99%	↑5.94%	↑15.38%	↑15.38%
	ALS-MF	IFB-MF	D-MF	kNN
(d) RMSE (win = 5, lose = 10, tie = 1)				
Movielens 100 K	↑0.68%	↑0.98%	↑0.87%	↓0.19%
Amazon	↑4.80%	↑4.58%	↑6.46%	↑0.97%
Yelp	↑0.05%	0.00%	↓0.09%	↓0.01%
FilmTrust	↑0.18%	↑0.06%	↓0.09%	↓0.38%

4.3 Rating distribution after outlier removal

Figure 4 illustrates the rating distributions before and after outlier removal. On Movielens 100 K, before removal, the proportions of the five ratings are 6.04%, 11.41%, 27.34%, 34.16% and 21.05%, respectively. After removal, the proportions are 5.90%, 11.26%, 27.19%, 34.41% and 21.25%, respectively. The numbers of the rating level 3 and 4 account for the two highest proportions. The original and new training sets basically obey the normal distribution. The means of $\mathbf{X}$ and $\mathbf{X}''$ are 3.5269 and 3.6065, respectively, and the variances of them are 1.2641 and 1.0880. Hence, the new training set still maintains the characteristics of the original training data.

4.4 Performance comparison

We use three popular matrix factorization (MF) recommendation algorithms and the kNN algorithm to examine the effects of outlier removal. MF-based algorithms include ALS-MF [43], D-MF [40], and IFB-MF [41]. In fact, MF characterizes users and items by vectors of latent factors. It has proven to be successful in dealing with large-scale and sparse data. ALS-MF integrates the implicit feedback into MF, and updates the latent sub-spaces by alternative least square. D-MF considers the impact of missing data to improve the efficiency of MF. IFB-MF considers not only missing data but also item popularity for further speedup. kNN [54, 55] employs the cosine distance to evaluate the item similarity. The similarity is used to find the neighbors of the items, ratings of which are used to make predictions.

Evaluation measures include HR, NDCG, mean average precision (MAP), and root mean square error (RMSE). HR evaluates the recall rate in the Top-k recommendation. NDCG reflects the quality of the ranking recommendation list by calculating its gap with the user's actual list. MAP indicates the position of the favorite item in the ranking list. RMSE represents the optimization degree of MF over all ratings. Although popular in many tasks such as regression, it is indirect in recommendation systems that require a ranking list.

Tables 9 and 10 compare the data quality before and after outlier removal. Table 9 shows the comparison of data quality. "Orig" represents the measures obtained by using the

Table 11 Comparison with other approaches

	ALS-MF			IFB-MF			D-MF			kNN		
	DNRS	ODFTD	Ours	DNRS	ODFTD	Ours	DNRS	ODFTD	Ours	DNRS	ODFTD	Ours
(a) HR												
Movielens100 K	0.1572	0.0914	**0.1720**	0.145	0.0820	**0.1510**	0.1587	0.0875	**0.1669**	**0.1601**	0.0551	0.1517
Amazon	0.0757	0.0716	**0.0783**	**0.0768**	0.0724	0.0731	**0.0793**	0.0724	0.0725	0.0704	0.0695	**0.0750**
Yelp	**0.0436**	0.0305	0.0396	0.0393	0.0281	**0.0657**	0.0458	0.0315	**0.0757**	**0.0232**	0.0224	0.0212
FilmTrust	0.2866	0.2664	**0.3103**	0.1441	0.0101	**0.1741**	0.3648	0.0074	**0.3891**	**0.3740**	0.3156	0.3634
Average Rank	1.75	3.00	**1.25**	1.75	3.00	**1.25**	1.75	3.00	**1.25**	**1.25**	2.75	2.00
	ALS-MF			IFB-MF			D-MF			kNN		
	DNRS	ODFTD	Ours	DNRS	ODFTD	Ours	DNRS	ODFTD	Ours	DNRS	ODFTD	Ours
(b) NDCG												
Movielens100 K	0.0368	0.0201	**0.0420**	0.0342	0.0181	**0.0400**	0.0361	0.0187	**0.0401**	**0.0366**	0.0104	0.0327
Amazon	0.0160	0.0152	**0.0164**	**0.0167**	0.0156	0.0147	**0.0163**	0.0158	0.0150	0.0141	0.0140	**0.0150**
Yelp	0.0093	0.0062	**0.0161**	**0.0084**	0.0055	0.0077	0.0096	0.0062	**0.0180**	**0.0049**	0.0047	0.0043
FilmTrust	0.0674	**0.0991**	0.0705	0.0654	0.0019	**0.1026**	0.1208	0.0015	**0.1242**	0.0971	**0.1127**	0.1003
Average Rank	2.25	2.50	**1.25**	2.00	2.25	**1.75**	2.25	2.25	**1.50**	**1.75**	2.25	2.00
	ALS-MF			IFB-MF			D-MF			kNN		
	DNRS	ODFTD	Ours	DNRS	ODFTD	Ours	DNRS	ODFTD	Ours	DNRS	ODFTD	Ours
(c) MAP												
Movielens100 K	0.0120	0.0056	**0.0133**	0.0112	0.0049	**0.0119**	0.0108	0.0046	**0.0121**	**0.0115**	0.0018	0.0090
Amazon	0.0036	0.0037	**0.0041**	**0.0041**	0.0039	0.0034	0.0034	**0.0042**	0.0035	0.0031	0.0029	**0.0040**
Yelp	**0.0022**	0.0014	0.0015	0.0023	0.0011	**0.0057**	0.0024	0.0012	**0.0060**	**0.0012**	0.0011	0.0090
FilmTrust	0.0526	0.0557	**0.0527**	**0.0443**	0.0004	0.0428	**0.0614**	0.0004	0.0604	0.0341	**0.061**	0.0405
Average Rank	2.00	2.75	**1.25**	**1.50**	2.75	1.75	2.00	2.50	**1.50**	**1.75**	2.25	2.00
	ALS-MF			IFB-MF			D-MF			kNN		
	DNRS	ODFTD	Ours	DNRS	ODFTD	Ours	DNRS	ODFTD	Ours	DNRS	ODFTD	Ours
(d) RMSE												
Movielens100 K	1.0911	1.1365	**1.0893**	1.0742	1.1300	**1.0684**	1.0703	1.1298	**1.0701**	1.0936	1.1041	**1.0550**
Amazon	0.8735	0.8636	**0.7862**	0.8823	0.8763	**0.7966**	0.8886	0.8697	**0.7417**	1.0224	0.9966	**0.9440**
Yelp	1.0546	**1.0505**	1.0544	1.0516	1.0492	**1.0325**	1.0551	**1.0504**	1.0550	1.0597	1.0521	**1.0427**
FilmTrust	**0.9269**	0.9456	0.9313	**0.9369**	0.9455	0.9463	**0.8905**	0.9450	0.8996	1.0469	**0.9641**	1.0081
Average Rank	2.25	2.25	**1.50**	2.25	2.25	**1.50**	2.25	2.25	**1.50**	2.75	2.00	**1.25**

Bold text indicates the best results. Average values of measures are shown for briefly

original training set for prediction. "New" means the measures obtained by using the new training set generated by our approach. Table 10 shows the performance changes of popular recommendation algorithms with the help of our outlier removal method. Table 10a shows that three MF-based approaches benefit from outlier removal while evaluated by HR. The HRs of the first twelve experiments all increase, and the two bigger improvements occur on Yelp and FilmTrust. However, kNN had not always benefited. Table 10b shows that the NDCGs of the fifteen experiments go up, except for kNN on Amazon. The maximum increase is 12.50%, when using the D-MF algorithm on Yelp. Table 10c shows that MAP is improved in twelve out of sixteen experiments. MAP remains unchanged when using IFB-MF on Yelp. In summary, the data quality can be improved by our proposed approach.

It should be noted that, according to Table 10d, RMSE increased in most cases. In other words, the data quality is decreased from the viewpoint of RMSE. As we know, the removal of non-outliers is inevitable. With closer observation, we found that some predictions for ratings of around 1 or 5 have larger biases after outlier removal. However, the respective recommendation list is not influenced by such bias. This may be the reason why RMSE becomes worse while the other three measures become better.

Table 11 compares the average rank of our approach with DNRS [5] and ODFTX [56]. Boldfaced average ranks are the top average ranks, and a smaller value represents a better performance. Table 11a shows that our approach achieves the rank of 1, 1, 2, and 1 in improving ALS-MF, the rank of 1, 2, 1 and 1 in improving IFB-MF and the rank of 1, 2, 1 and 1 in improving D-MF. So our approach obtains the best average rank of 1.25 in improving three MF-based algorithms, respectively, and DNRS achieves the best average rank of 1.25 in improving kNN. Table 11b shows that our approach achieves the best average rank of 1.25, 1.75 and 1.50 in improving three MF-based algorithms, respectively, and DNRS achieves the best average rank of 1.75 in improving kNN. Table 11c shows that our approach achieves the best average rank of 1.25 and 1.50 in improving ALS-MF and D-MF, respectively, and DNRS achieves the best average rank of 1.50 and 1.75 in improving kNN. Table 11d shows that our approach achieves the best average rank in four algorithms. In summary, our approach is more advantageous than the other two in improving the three MF algorithms, and DNRS outweighs our approach in improving kNN algorithm.

4.5 Discussion

Now we will answer the questions at the beginning of this section.

(1) The parameter settings of different datasets are different. The threshold for judging outliers λ is too large, when a few outliers are removed, the performance cannot be improved. The parameter λ is too small, the number of removed ratings will be increased, which will leads to the removal of non-outliers and decrease the performance.
(2) Our algorithm is efficient. It is linear wrt. the number of users or items, hence scalable to large datasets.
(3) The rating distribution of $\mathbf{X}$ is similar to that of $\mathbf{X}''$. So, outlier removal has little effect on the distribution of $\mathbf{X}$, and retains the characteristics of $\mathbf{X}$.
(4) Our algorithm is effective. The prediction accuracy is improved with the denoised training set.

5 Conclusion and future work

We have proposed an efficient and effective outlier removal algorithm to enhance the quality of data. The universal MoG is introduced to model the noise of training data. The evaluation measure for outliers is the number of times they have been identified. Results show that popular MF algorithms benefit from the outlier removal process.

In the future, we will design new algorithms based on the Poisson-Gaussian mixture [57], the mixture of exponential power [58], and the Laplacian scale mixture [59] distributions. These distributions may be better suited to modeling noise or user uncertainty. User comments or context information will also be considered. This information will make the problem more complicated, more challenging, and more interesting.

Acknowledgements This work is supported in part by the National Natural Scientific Foundation of China (61976194, 41631179), the Open project of Key Laboratory of Oceanographic Big Data Mining and Application of Zhejiang Province (OBMA202005), the Zhejiang Provincial Natural Science Foundation of China (LY18F030017), the Natural Science Foundation of Sichuan Province (2019YJ0314).

References

1. Soros G (2013) Fallibility, reflexivity, and the human uncertainty principle. J Econ Methodol 20(4):309–329
2. Beinhocker ED (2013) Reflexivity, complexity, and the nature of social science. J Econ Methodol 20(4):330–342
3. Aggarwal CC et al (2016) Recommender systems. Springer, New York
4. Gunes I, Kaleli C, Bilge A, Polat H (2014) Shilling attacks against recommender systems: a comprehensive survey. Artif Intell Rev 42(4):767–799
5. O'Mahony MP, Hurley NJ, Silvestre G (2006) Detecting noise in recommender system databases. In: Proceedings of the 11th international conference on intelligent user interfaces, ACM pp 109–115
6. Wu C, Zhang Q, Zhao F, Cheng Y, Wang G (2021) Three-way recommendation model based on shadowed set with uncertainty invariance. Int J Approx Reason 135:53–70
7. Wang YP, Yu H, Wang GY, Xie YF (2020) Cross-domain recommendation based on sentiment analysis and latent feature mapping. Entropy 22(4):473
8. Zhang HR, Min F, Wu YX, Fu ZL, Gao L (2018) Magic barrier estimation models for recommended systems under normal distribution. Appl Intell 48(12):4678–4693
9. Sah RK (1991) Fallibility in human organizations and political systems. J Econ Perspect 5(2):67–88
10. Xu YS, Zhang FZ (2019) Detecting shilling attacks in social recommender systems based on time series analysis and trust features. Knowl-Based Syst 178:25–47
11. Pang M, Gao W, Tao M, Zhou ZH (2018) Unorganized malicious attacks detection. In: NIPS. pp 6976–6985
12. Lam SK, Riedl J (2004) Shilling recommender systems for fun and profit. In: WWW. pp 393–402
13. Luca M.: Reviews, reputation, and revenue: the case of yelp. com. Harvard Business School Working Papers 12-016, Harvard Business School (2016)
14. Jasberg K, Sizov S (2017) The magic barrier revisited: accessing natural limitations of recommender assessment. In: RecSys. pp 55–64
15. Ling G, King I, Lyu MR (2013) A unified framework for reputation estimation in online rating systems. IJCA I:2670–2676
16. Williams CA, Mobasher B, Burke R (2007) Defending recommender systems: detection of profile injection attacks. Serv Orient Comput Appl 1(3):157–170

17. Yap GE, Tan AH, Pang HH (2007) Discovering and exploiting causal dependencies for robust mobile context-aware recommenders. IEEE Trans Knowl Data Eng 19(7):977–992
18. Li B, Chen L, Zhu XQ, Zhang CQ (2013) Noisy but non-malicious user detection in social recommender systems. World Wide Web 16(5–6):677–699
19. Kim E, Pyo S, Park E, Kim M (2011) An automatic recommendation scheme of TV program contents for (IP)TV personalization. IEEE Trans Broadcast 57(3):674–684
20. Chen XA, Han Z, Wang Y, Zhao Q, Meng DY, Tang YD (2016) Robust tensor factorization with unknown noise. In: CVPR pp 5213–5221
21. McLachlan GJ, Basford KE (1988) Mixture models: inference and applications to clustering. Marcel Dekker
22. Meng DY, De La Torre F (2013) Robust matrix factorization with unknown noise. In: ICCV. pp 1337–1344
23. Hofmann T (2003) Collaborative filtering via Gaussian probabilistic latent semantic analysis. In: SIGIR pp 259–266
24. Si L, Jin R (2003) Flexible mixture model for collaborative filtering. In: ICML pp 704–711
25. Moon TK (1996) The expectation-maximization algorithm. IEEE Signal Process Mag 13(6):47–60
26. Dempster AP, Laird NM, Rubin DB (1977) Maximum likelihood from incomplete data via the EM algorithm. J R Stat Soc Ser B 85:1–38
27. Liu D (2021) The effectiveness of three-way classification with interpretable perspective. Inform Sci 567:237–255
28. Xu YY, Zhang HR, Min F (2017) A three-way recommender system for popularity-based costs. In: Proceedings of international joint conference on rough set. pp 278–289
29. Gemmell J, Schimoler T, Ramezani M, Mobasher B (2009) Adapting k-nearest neighbor for tag recommendation in folksonomies. In: ITWP
30. Zhang HR, Min F, Zhang ZH, Wang S (2018) Efficient collaborative filtering recommendations with multi-channel feature vectors. Int J Mach Learn Cybernet 10:1–8
31. Tsai CF, Hung C (2012) Cluster ensembles in collaborative filtering recommendation. Appl Soft Comput 12:1417–1425
32. Liu D, Ye XQ (2020) A matrix factorization based dynamic granularity recommendation with three-way decisions. Knowl Based Syst 191:105243
33. Nilashi M, Ibrahim O, Bagherifard K (2018) A recommender system based on collaborative filtering using ontology and dimensionality reduction techniques. Expert Syst Appl 92:507–520
34. Panagiotakis C, Papadakis H, Papagrigoriou A, Fragopoulou P (2021) Improving recommender systems via a dual training error based correction approach. Expert Syst Appl 183:115386
35. Zhang HR, Min F (2016) Three-way recommender systems based on random forests. Knowl-Based Syst 91:275–286
36. Zhang HR, Min F, Shi B (2017) Regression-based three-way recommendation. Inform Sci 378:444–461
37. Ye XQ, Liu D (2021) An interpretable sequential three-way recommendation based on collaborative topic regression. Expert Syst Appl 168:114454
38. Revaud J, Almazán J, Rezende RS, Souza CRD (2019) Learning with average precision: training image retrieval with a listwise loss. In: ICCV. pp 5107–5116
39. Chen WS, Zhao Y, Pan B, Chen B (2019) Supervised kernel nonnegative matrix factorization for face recognition. Neurocomputing 205:165–181
40. Devooght R, Kourtellis N, Mantrach A (2015) Dynamic matrix factorization with priors on unknown values. In: SIGKDD. pp 189–198
41. He X, Zhang H, Kan MY, Chua TS (2016) Fast matrix factorization for online recommendation with implicit feedback. In: SIGIR. pp 549–558
42. Funk S (2006) Netflix update: try this at home
43. Hu Y, Koren Y, Volinsky C (2008) Collaborative filtering for implicit feedback datasets. In: ICDM. pp 263–272
44. Davoudi A, Chatterjee M (2017) Detection of profile injection attacks in social recommender systems using outlier analysis. In: ICBD. pp 2714–2719
45. Panagiotakis C, Papadakis H, Fragopoulou P (2020) Unsupervised and supervised methods for the detection of hurriedly created profiles in recommender systems. Int J Mach Learn Cybernet 11(9):2165–2179
46. Toledo RY, Mota YC, Martínez L (2015) Correcting noisy ratings in collaborative recommender systems. Knowl-Based Syst 76:96–108
47. Chakraborty PS (2020) Attack detection in recommender systems using subspace outlier detection algorithm. In: Proceedings of the 2nd international conference on communication, devices and computing. pp 679—685
48. Scheunders P, De Backer S (2007) Wavelet denoising of multicomponent images using Gaussian scale mixture models and a noise-free image as priors. IEEE Trans Image Process 16(7):1865–1872
49. Hansen F, Pedersen GK (1982) Jensen's inequality for operators and löwner's theorem. Math Ann 258(3):229–241
50. Peajcariaac JE, Tong YL (1992) Convex functions, partial orderings, and statistical applications. Academic Press, San Diego
51. Yu H, Zhou B, Deng MY, Hu F (2018) Tag recommendation method in folksonomy based on user tagging status. J Intell Inform Syst 14:1–22
52. Ma TH, Zhou JJ, Tang ML, Tian Y, Al-Dhelaan A, Al-Rodhaan M, Lee S (2015) Social network and tag sources based augmenting collaborative recommender system. IEICE Trans Inform Syst 98(4):902–910
53. Harper FM, Konstan JA (2016) The movielens datasets: history and context. Acm Trans Interact Intell Syst 5(4):1–19
54. Sarwar B. Karypis G, Konstan J, Riedl J (2001) Item-based collaborative filtering recommendation algorithms. In: WWW. pp 285–295
55. Adeniyi D, Wei ZQ, Yang YQ (2016) Automated web usage data mining and recommendation system using K-nearest neighbor (KNN) classification method. Appl Comput Inform 12(1):90–108
56. Kannan R, Woo H, Aggarwal CC, Park H (2017) Outlier detection for text data. In: Proceedings of the 2017 SIAM international conference on data mining. pp 489–497
57. Marnissi Y, Zheng Y, Chouzenoux E, Pesquet JC (2017) A variational Bayesian approach for image restoration—application to image deblurring with poisson-gaussian noise. IEEE Trans Comput Imaging 3(4):722–737
58. Cao XY, Chen Y, Zhao Q, Meng DY, Wang Y, Wang D, Xu ZB (2015) Low-rank matrix factorization under general mixture noise distributions. In: ICCV. pp 1493–1501
59. Yang ZZ, Fan L, Yang YP, Yang Z, Gui G (2020) Generalized nuclear norm and Laplacian scale mixture based low-rank and sparse decomposition for video foreground-background separation. Signal Process 172:107527