Knowl Inf Syst
https://doi.org/10.1007/s10115-018-1198-6

REGULAR PAPER

Preference modeling by exploiting latent components of ratings

Junhua Chen[1] · **Wei Zeng**[1] · **Junming Shao**[2] · **Ge Fan**[1]

Received: 27 June 2017 / Revised: 12 March 2018 / Accepted: 18 April 2018

Abstract Understanding user preference is essential to the optimization of recommender systems. As a feedback of user's taste, the rating score can directly reflect the preference of a given user to a given product. Uncovering the latent components of user ratings is thus of significant importance for learning user interests. In this paper, a new recommendation approach was proposed by investigating the latent components of user ratings. The basic idea is to decompose an existing rating into several components via a cost-sensitive learning strategy. Specifically, each rating is assigned to several latent factor models and each model is updated according to its predictive errors. Afterward, these accumulated predictive errors of models are utilized to decompose a rating into several components, each of which is treated as an independent part to further retrain the latent factor models. Finally, all latent factor models are combined linearly to estimate predictive ratings for users. In contrast to existing methods, our method provides an intuitive preference modeling strategy via multiple component analysis at an individual perspective. Meanwhile, it is verified by the experimental results on several benchmark datasets that the proposed method is superior to the state-of-the-art methods in terms of recommendation accuracy.

Keywords Collaborative filtering · Matrix factorization · Multi-criteria recommender systems

1 Introduction

With the rapid growth of the Internet and the overwhelming amount of contents and choices that people are confronted with, recommender systems have been developed to facilitate the

✉ Wei Zeng
zwei504@uestc.edu.cn

1 Trusted Computing and Automated Reasoning Lab, University of Electronic Science and Technology of China, Chengdu 611731, People's Republic of China

2 Web Sciences Center, Big Data Research Center, University of Electronic Science and Technology of China, Chengdu 611731, People's Republic of China

Published online: 26 April 2018

decision-making process. During the past decades, more and more researchers have started to study the multi-criteria recommender system, which allows individual users to rate multiple attributes of an item [1,2]. For instance, a two-criteria movie recommender system allows users to express their preferences on two attributes of a movie (e.g., story novelty and visual effect). A user may be fond of the visual effects but dislike the story of a movie. In such a case, the movie is rated by a user based on two ratings, namely the story and visual effect attributes.

In a multi-criteria recommender system, an individual user can make a choice based on more than one utility-related aspect. Actually, a user usually rates a movie after a comprehensive consideration. For example, she may firstly consider the movie's director, actors, story and visual effects, and then make the choice. Therefore, the accuracy of item recommendation can be enhanced by the additional information provided by multi-criteria ratings because it can represent more complex preferences of each user. Recent works also demonstrated that the multi-criteria technique is superior to the traditional methods that utilize single-criterion ratings [3].

Multi-criteria rating systems require comparably higher users' involvement because each user need to complete rating on all criteria, which increases the likelihood of obtaining missing or incomplete data. Additionally, consistent family of exhaustive and non-redundant criteria raises the costs of establishing a recommender system. Therefore, a number of known Web sites still prefer the single-criterion rating system [4]. Most collected datasets accordingly only contain single-criterion ratings. Some previous studies indicate that users' evaluation of items (products and service, etc.) generally relies on several criteria, since there are certain decision problems that have to be addressed simultaneously [5,6]. There arises a question whether the users' preferences on diverse criteria can be uncovered by exploiting the information of single-criterion ratings. If the problem is solved, there will be lower cost of establishing a recommender system along with more accurate item recommendation.

In this paper, it is assumed that individual users' evaluations on items are multi-criteria, and their ratings consist of multiple latent components in a single-criterion recommender system. In order to determine these latent components, we make use of several latent models based on the matrix factorization method which maps both users and items to a joint latent factor space. The gradient descent approach is adopted to learn the parameters of the model in an iterative way. At each iteration, the predictive score between a user and an item is calculated independently by each latent model, and the deviation between the predictive score and the real rating is recorded. Each model is then assigned a weight according to its predictive deviation, where a smaller deviation reflects a larger weight. Afterward, each rating is decomposed into several latent components with respect to the weight of each latent model. A latent component reflects a user's preference on one criterion, and all ratings' latent components are subsequently exploited to retrain those learned latent models to further improve their predictive accuracy. Finally, we linearly combine the predictive rating calculated by those latent models.

In a traditional multi-criteria recommender system, all the ratings on attributes of an item are explicitly given by individual users [3], whereas in our method, all latent components of a rating are automatically learned by several latent factor models. Therefore, our method not only overcomes the drawback of single-criterion ratings but also requires less information than the multi-criteria technique. The contributions of our work can be summarized as follows:

1.1 Intuitive preference model

Instead of utilizing multiple ratings explicitly given by users, we decompose a rating into multiple latent components. Those latent components can better reflect the user's preference than a single-criterion rating. Moreover, less information is required by our method when compared to the multi-criteria recommender system. Therefore, our method enjoys better application possibilities.

1.2 Predictive performance

An individual user's evaluation on items normally relies on several criteria, which has been verified in the multi-criteria recommender system [3,4]. Motivated by this idea, we uncover latent ratings of users on diverse criteria in a single-criterion recommender system. Within our method, more information can be uncovered from individual ratings, which enables our method to alleviate the sparsity problem of recommender systems to some extent. Meanwhile, the experimental results on five real datasets show that our method outperforms the state-of-the-art approaches, which indicate that our method learns the structure of the data better than existing approaches.

The rest of the paper is organized as follows. Section 2 gives a brief overview of the related research close to our work. Section 3 covers preliminaries and our proposed model. The Experiments are shown in Sect. 4, and finally, this paper concludes with Sect. 5.

2 Related works

Based on the assumption that an individual user's evaluation on items generally relies on several criteria, we decomposed each single-criterion rating into several latent components. Each component can be considered as an individual rating on one criterion. Therefore, our work is related with both the single-criterion and multi-criteria approaches. In this section, the related works are reviewed.

2.1 Single-criterion recommender systems

Up to now, plenty of methods have been proposed to solve the personalization problem. Among them, the collaborative filtering (CF) methods are considered to be the most popular one, which have been widely investigated and applied in online systems [7]. The CF methods can be categorized into two classes: neighborhood [8,9] and model-based methods [10–13]. The neighborhood CF approaches can be further divided into two groups: the user-based CF and the item-based CF [14–16]. The assumption of the user-based CF is that similar users have similar tastes or preferences, whereas the item-based CF assumes that a user tends to collect similar items. The key of neighborhood CF approaches is to compute the similarity between neighbors or to provide a method to find the candidate neighbors. In order to find a user's neighbors efficiently, certain cluster-based methods have been proposed [17,18]. Cluster-based methods first cluster the users and then calculate the user's neighbors within the cluster. For instance, Alqadah et al. [18] proposed a biclustering neighborhood CF approach for the top-n recommendation which combines local similarity of biclusters with global similarity. Liu et al. [17] exploited the global k-means method to divide users into disjoint groups by making use of users' multi-criteria ratings.

Due to the simplicity and efficiency of neighborhood CF methods, they have a very wide application. For example, GroupLens and Bellcore video exploited individual ratings to predict their interests in articles and movies [14,15]. In most recommender systems, the number of users is usually greater than the number of items. In addition, the amount of ratings for each item is often greater than the amount of ratings for each user. Thus, item-based CF methods are preferable by online retailers, such as Amazon.com and Last.fm [8,16].

Unlike neighborhood CF methods, model-based CF methods make use of users' ratings to learn a predictive model. Breese et al. [19] proposed a Bayesian clustering model which introduces certain groups to reflect the common preferences and tastes of users. The latent semantic analysis models users' ratings as a mixture of individual communities [10]. Latent Dirichlet Allocation (LDA) is a flexible generative probabilistic model for the collection of discrete data, which can be used for document classification and collaborative filtering [11]. The Restricted Boltzmann Machines, a kind of two-layer undirected graphical models, was presented to recommend movies for users [12].

It is worth mentioning that the matrix factorization (MF) approach is becoming increasingly popular in recent years due to its good scalability and predictive accuracy [20–25]. The rating matrix is composed of two parts in this approach, i.e., the user latent factor matrix and the item latent feature matrix. The predictive score is obtained by the computation based on the inner product of the user latent feature vector and the item latent feature vector [20]. Generally, the low-rank approximation method and the regularization method are used to get the latent feature matrices and prevent overfitting, respectively [21–24].

Recently, a number of enhanced matrix factorization methods are proposed [21,22,26]. For instance, Koren [26] proposed a combined approach which improves prediction accuracy by capitalizing the advantages of both the neighborhood-based method and the matrix factorization method. The probabilistic matrix factorization is a probabilistic interpretation of the traditional matrix factorization methods [21], which yields better scalability and robustness. Furthermore, a Bayesian treatment is applied in the probabilistic matrix factorization method which makes use of Markov Chain Monte Carlo (MCMC) method for approximation inference. The new method is applicable to a large dataset and achieves significantly higher predictive accuracy than the base probabilistic matrix factorization method [22].

As mentioned above, the user and item latent factor are normally obtained by the low-rank approximation method which constructs a matrix that approximates the rating matrix at its observed entries. Recently, Lee et al. [23] proposed a local low-rank approximation which firstly divides the rating matrix into several submatrices consisting of certain row–column combinations and then each submatrix is constructed by a low-rank approximation. Furthermore, Lee et al. [27] replaced the squared loss reconstruction by a ranked loss minimization in his method which outperforms state-of-the-art methods in terms of item ranking.

The matrix factorization is quite a flexible method, which allows incorporation of additional information, such as time factor [28–31], geographical information [32–35] and social information [36–39]. For instance, Koren [28] investigated the temporal dynamics of customer preferences and modeled the temporal dynamics along the whole time period. Authors applied the methodology with two recommender techniques: the factorization model and the neighborhood model. In both models, the temporal dynamics can be useful in improving the quality of rating predictions. McAuley et al. [31] developed a latent factor model which explicitly accounts for each user's level of experience. The time-aware model can not only achieve better recommendations but also allow to study the role of user experience and expertise. Lian et al. [35] incorporated the geographical information and the user activity data by the weighted matrix factorization which can alleviate the sparsity problem. Shen et al. [39]

integrated the user-item ratings with the social information by a probabilistic model, and the expectation maximization algorithm was used to infer parameters of the model.

2.2 Multi-criteria recommender systems

Compared to single-criterion recommender systems, the multi-criteria recommender system contains more information, including ratings of item attributes. The complexity of algorithms is increased by the additional information; however, in most cases the quality of the recommendation can also be improved by incorporating the auxiliary data [3,17,40,41]. To the best of our knowledge, examples of multi-criteria recommender systems include Zagat's Guide, Buy.com and Yahoo! Movies [42]. To exploit the information of multi-criteria ratings, a commonly used method is to extend the computation of similarity, from single-criterion ratings to multi-criteria ratings [40,41,43]. For example, within a user-based collaborative filtering approach, the similarity of two users is computed by making use of their single-criterion ratings while in a multi-criteria recommender system, the computation of similarity is performed on each criterion and finally averages the result over all criteria. Lee et al. [44] extended the concept of single-criterion rating to multi-criteria ones and utilized skyline query algorithm to find candidate items. In paper [45], an adaptive neuro-fuzzy inference method is used to discover the relationship between each criterion and the overall ratings. A fusion of fuzzy cosine and Jaccard similarities is further adopted to calculate the total similarities between users/movies.

Recently, regression methods are exploited in the multi-criteria recommender system with the assumption that a user's global rating is a linear/nonlinear combination of her ratings on each criterion [2,40,46]. For instance, Jannach et al. [40] made use of support vector regression to determine the relative importance of the multi-criteria ratings and combined user-based and item-based regression models in a weighted way. Zheng [46] utilized the biased matrix factorization to estimate a user's rating on each criterion and chose the support vector regression to learn the relevance of the multi-criteria ratings. There usually exists user clusters in many recommender systems. Within the same cluster, individual users behave similarly. Paper [2] made use of multi-criteria ratings to detect user clusters and then to learn the importance weight of multi-criteria ratings through the regression model. In paper [47], authors applied Self-Organizing Map clustering algorithm (SOM) to detect user clusters and Adaptive Neuro-Fuzzy Inference Systems (ANFIS) to discover the relative importance of the individual criterion ratings.

Another way to make use of multi-criteria ratings is to construct a predictive model by learning from the observed data, including probabilistic modeling, multi-linear SVD model and matrix factorization [24]. For instance, Saboo et al. [48] extended the flexible mixture model to multi-criteria rating systems, where the user behavior and item characteristics were characterized separately by two latent variables. Li et al. [49] improved the traditional collaborative filtering method by expanding the criterion to a tensor and utilizing the multi-linear SVD model. Furthermore, the multi-linear SVD model is combined with the adaptive neuro-fuzzy inference method to solve the scalability and sparsity problem in a multi-criteria system [1,50]. McAuley et al. [51] learned attitudes and attributes from explicit multi-aspect reviews with a joint probabilistic model to yield better recommendations. Based on the matrix factorization technique, Pozo et al. incorporated individual users' explicit ratings with their implicit interest in the attributes of items [52].

2.3 Limitations of related works

The user decision-making process is considerably complex [4], since they may rely on several criteria to evaluate an item. For instance, in a movie recommender system, a user may take into account a movie's story, sound and graphic effects before she decided to see the movie. Although many existing single-criterion-based recommendation methods achieved a remarkable success, they failed to uncover the individual user's preference on different criteria [3]. In a multi-criteria recommender system, users are allowed to rate attributes of items explicitly; however, it needs the high-level involvement of users. Too few criteria lead to a poor understanding of users' preferences, while too many criteria lead to impatientness of users to rate each criterion. Both of scenario greatly raise the likelihood of obtaining incomplete data; thus, the incompleteness of data significantly affects the accuracy of the recommender system.

Therefore, given the limitation of related works, we uncover users' preferences on different criteria by exploring single-criterion ratings. Our method does not require additional information and it outperforms the state-of-the-art approaches, which indicates our method learns the structure of data better than existing ones.

3 Latent components of ratings

In this section, we mainly discuss our method, which investigates the latent components of user ratings, namely LCR model. A rating system can be represented by a weighted adjacency matrix $R_{n\times m} = \{r_{ui}\}$, Where r_{ui} is the rating which user u gives to item i. If item i is not rated by user u, then $r_{ui} = 0$. n and m are the number of users and the number of items in the system, respectively. As we know, the goal of a recommender system is to predict scores between each user and her uncollected items and then to recommend the top-L items with the highest scores. Thus, research on recommender systems is always challenged by finding the most accurate recommendation algorithms.

3.1 The problem statement

In the multi-criteria recommender system, an individual user's diverse and complex preferences can be demonstrated by giving ratings to attributes of an item. However, in most existing recommender systems, a particular user is constrained to give a single-criterion value to an item. It is shown by the recent works that this mechanism has some potential limitations, because a user may make a choice based on more than one utility-related aspect [4]. By borrowing ideas from multi-criteria recommender system, the goal of our work is to decompose the rating matrix R into c latent matrices: $R = \sum_{\alpha=1}^{c} R_\alpha$. We assumed that those latent matrices are independent to each other, and the latent matrix R_α has the same size with the original rating matrix R. There are plenty of composition forms. For instance, the rating matrix R can be composed randomly of several submatrices. However, this method is useless in improving the accuracy of recommendation algorithms. More details can be found in the *Experiments* section.

In this paper, it is assumed that preferences of users are multi-dimensional, and each dimension can be represented by a latent factor model. It is like, in a multi-criteria recommender system, different models are used to denote a user's diverse preferences on several attributes of an item. Thus, with our method, the complex preferences of users can be uncovered with

Table 1 Notations used in this paper

Notation	Description
n	The number of users
m	The number of items
c	The number of latent component of ratings
R	The rating matrix
r_{ui}	The rating that user u gives to item i
$\widehat{r}_{ui}$	The predictive rating between user u and item i
$x_u,\ y_i$	The latent factor vector of user u and item i, respectively
μ	The global average rating
b_i, b_u	The deviation of user u and item i, respectively
λ	The coefficient of the regularization
γ	The learning rate
$\widehat{r}_{ui}^{(\alpha)}$	The predictive score between user u and item i by latent model Θ_α
$x_u^{(\alpha)}, y_i^{(\alpha)}$	The latent factor vector of user u and item i with respective to model Θ_α
$w_{ui}^{(\alpha)}$	The weight allocated for model Θ_α given the training rating r_{ui}

no necessity for additional information. Moreover, to simplify the computation, those latent factor models are supposed to be independent of each other and are trained simultaneously.

The cost-sensitive approach was adopted to learn latent factor models. More specially, c latent factor models, $\{\Theta_1, \Theta_2, \ldots, \Theta_c\}$, were randomly initialized, and then the predictive score $\widehat{r}_{ui}^{(\alpha)}$ between user u and item i was computed by model Θ_α. By referring to the deviation between $\widehat{r}_{ui}^{(\alpha)}$ and the real rating r_{ui}, the weight of model Θ_α with respect to rating r_{ui} was recorded. Finally, by exploiting the accumulated weight, the rating matrix R was decomposed into c latent matrices. All of the major notations used in this paper are given in Table 1.

3.2 Latent factor models

In this paper, latent factor models were adopted to decompose ratings, based on the matrix factorization. Firstly, users and items were mapped by the matrix factorization approach to a joint latent factor space of dimensionality k. Then each user was associated with a vector x_u, and each item was associated with a vector y_i. And the predictive score $\widehat{r}_{ui}$ between user u and item i was obtained by the inner product of x_u and y_i:

$$\widehat{r}_{ui} = x_u^T y_i, \tag{1}$$

which is the basic form of the matrix factorization. One advantage of the matrix factorization approach is its adaptability to various data resources and other application-specific requirements. For example, biases of users and items are added in Eq. 1, to indicate the observed deviations of users and items, respectively [53]. Therefore, the equation of matrix factorization with biases can be defined as:

$$\widehat{r}_{ui} = \mu + b_i + b_u + x_u^T y_i, \tag{2}$$

where μ is the global average rating, and b_u and b_i are the deviation of user u and item i, respectively, from the average.

At last, the biased matrix factorization approach was chosen to decompose each rating due to the following concerns. Firstly, the biased matrix factorization approach is superior to other recommendation algorithms such as collaborative filtering, basic matrix factorization, nonnegative matrix factorization and probabilistic matrix factorization in terms of recommendation accuracy [53–55]. Secondly, the complexity of the biased matrix factorization is comparable to the basic matrix factorization. In this paper, our algorithm was implemented based on the *LibRec* package, which is a GPL-licensed Java library for recommender systems.[1] In general, it runs much faster than other packages while achieving competitive performance. Thirdly, multiple preferences of users can be further explored by taking into account the biases of users and items [20]. As mentioned above, users may rely on more than one aspect to make a choice, which may cause interest biases of individual users.

Given the observed ratings, the biased matrix factorization approach is learned by minimizing the objective function:

$$\min_{x^*,y^*,b^*} \sum_{u,i} (r_{ui} - \widehat{r}_{ui})^2 + \lambda(\|x_u\|^2 + \|y_i\|^2 + b_u^2 + b_i^2), \tag{3}$$

where λ is the regularization parameter which is used to prevent overfitting and is set to 0.005. The gradient descent method is used to obtain x_u, y_i, b_u, b_i whose initial values are generated randomly from the Gaussian distribution with mean 0 and variance 0.1. x^*, y^*, b^* stand for the general form of x_u, y_i, b_u and b_i, respectively. In order to get the optimal parameters in Eq. 3, the stochastic gradient descent approach is applied to update parameters in the opposite direction of the gradient of the cost function:

$$\begin{aligned} b_u &\leftarrow b_u + \gamma \cdot ((r_{ui} - \widehat{r}_{ui}) - \lambda \cdot b_u) \\ b_i &\leftarrow b_i + \gamma \cdot ((r_{ui} - \widehat{r}_{ui}) - \lambda \cdot b_i) \\ x_u &\leftarrow x_u + \gamma \cdot ((r_{ui} - \widehat{r}_{ui}) \cdot y_i - \lambda \cdot x_u) \\ y_i &\leftarrow y_i + \gamma \cdot ((r_{ui} - \widehat{r}_{ui}) \cdot x_u - \lambda \cdot y_i) \end{aligned} \tag{4}$$

where γ is a step size which is usually set to a small value (e.g., 0.005).

3.3 Rating decompositions

Although with biased matrix factorization approach both users and items can be well modeled, its assumption that a rating is a single component would limit potential methods for better modeling the overall data. Therefore, we supposed that each rating is comprised of several latent components, and each component can be learned by an independent model. As mentioned above, we firstly initialized c latent factor models: $\{\Theta_1, \Theta_2, \ldots, \Theta_c\}$. Each model Θ_α was based on an independent biased matrix factorization. The predictive score $\widehat{r}_{ui}^{(\alpha)}$ between user u and item i by model Θ_α can be given by:

$$\widehat{r}_{ui}^{(\alpha)} = \mu + b_i^{(\alpha)} + b_u^{(\alpha)} + x_u^{(\alpha)T} y_i^{(\alpha)}, \tag{5}$$

where $x_u^{(\alpha)}$ and $y_i^{(\alpha)}$ are latent vector of user and item with respect to model Θ_α, respectively. Secondly, the cost-sensitive approach was applied to assign rating r_{ui} to the latent models for training. More specifically, the following cost function was minimized as:

[1] http://www.librec.net/index.html.

$$\min_{x^*,y^*,b^*} \sum_{u,i} \sum_{\alpha=1}^{c} (w_{ui}^{(\alpha)} r_{ui} - \widehat{r}_{ui}^{(\alpha)})^2 + \lambda \left(\sum_{\alpha=1}^{c} \| x_u^{(\alpha)} \|^2 + \sum_{\alpha=1}^{c} \| y_i^{(\alpha)} \|^2 + (b_u^{(\alpha)})^2 + (b_i^{(\alpha)})^2 \right), \tag{6}$$

where the meta-parameter $w_{ui}^{(\alpha)}$ is the weight allocated for model Θ_α when training the rating r_{ui}. The parameter λ is set to 0.005 in all models and initial values of $x_u^{(\alpha)}$, $y_i^{(\alpha)}$, $b_u^{(\alpha)}$, $b_i^{(\alpha)}$ are generated randomly from the Gaussian distribution with mean 0 and variance 0.1. Meanwhile, $w_{ui}^{(\alpha)}$ was computed as follows:

$$w_{ui}^{(\alpha)} = \frac{e^{-|r_{ui} - \widehat{r}_{ui}^{(\alpha)}|}}{\sum_{\alpha=1}^{c} e^{-|r_{ui} - \widehat{r}_{ui}^{(\alpha)}|}}. \tag{7}$$

From Eq. 7, it can be seen that the weight of model Θ_α is inversely proportional to the absolute error between the predictive rating $\widehat{r}_{ui}^{(\alpha)}$ and the real rating r_{ui}. The assumption behind our method is that the weight of model Θ_α can be reflected by its predictive performance. Similarly, the stochastic gradient descent method was utilized to acquire parameters in Eq. 6. In order to reduce the complexity of our method, we compute the $w_{ui}^{(\alpha)}$ by $\widehat{r}_{ui}^{(\alpha)}$ obtained in the last iterative step. The weight $w_{ui}^{(\alpha)}$ at each iteration was preserved, in order to decompose ratings further. Thus, for a given training rating r_{ui}, we updated the parameters in model Θ_α as follows:

$$\begin{aligned}
b_u^{(\alpha)} &\leftarrow b_u^{(\alpha)} + \gamma \cdot \left(\left(w_{ui}^{(\alpha)} r_{ui} - \widehat{r}_{ui}^{(\alpha)} \right) - \lambda \cdot b_u^{(\alpha)} \right) \\
b_i^{(\alpha)} &\leftarrow b_i^{(\alpha)} + \gamma \cdot \left(\left(w_{ui}^{(\alpha)} r_{ui} - \widehat{r}_{ui}^{(\alpha)} \right) - \lambda \cdot b_i^{(\alpha)} \right) \\
x_u^{(\alpha)} &\leftarrow x_u^{(\alpha)} + \gamma \cdot \left(\left(w_{ui}^{(\alpha)} r_{ui} - \widehat{r}_{ui}^{(\alpha)} \right) \cdot y_i^{(\alpha)} - \lambda \cdot x_u^{(\alpha)} \right) \\
y_i^{(\alpha)} &\leftarrow y_i^{(\alpha)} + \gamma \cdot \left(\left(w_{ui}^{(\alpha)} r_{ui} - \widehat{r}_{ui}^{(\alpha)} \right) \cdot x_u^{(\alpha)} - \lambda \cdot y_i^{(\alpha)} \right)
\end{aligned} \tag{8}$$

Given the training rating r_{ui}, we re-computed the weight $w_{ui}^{(\alpha)}$ of model Θ_α at the end of each iteration presented above. After the training process, the weight $w_{ui}^{(\alpha)}$ of model Θ_α was then accumulated as:

$$w_{ui}^{*(\alpha)} = \sum_{q=1}^{p} w_{ui}^{(\alpha)}(q), \tag{9}$$

where p is the frequency that training rating r_{ui} is used in the iterative process. Finally, the weight is normalized as:

$$\overline{w}_{ui}^{(\alpha)} = \frac{w_{ui}^{*(\alpha)}}{\sum_{\alpha=1}^{c} w_{ui}^{*(\alpha)}}. \tag{10}$$

Based on the normalization weight $\overline{w}_{ui}^{(\alpha)}$, we can decompose rating r_{ui} into c latent components. The score of the αth component can be obtained by $\overline{r}_{ui}^{(\alpha)} = \overline{w}_{ui}^{(\alpha)} r_{ui}$. The pseudo-code of the rating decomposition is presented in Algorithm 1.

Input: the rating matrix $R \in \mathbb{R}^{n \times m}$
Initialize: $b_u^{(\alpha)}, b_i^{(\alpha)}, x_u^{(\alpha)}, y_i^{(\alpha)}, \alpha = 1, ..., c$

for iteration $q = 1$ to p **do**

  for rating r_{ui} in R **do**

    for model $\alpha = 1$ to c **do**

$$b_u^{(\alpha)} \leftarrow b_u^{(\alpha)} + \gamma \cdot ((w_{ui}^{(\alpha)} r_{ui} - \widehat{r}_{ui}^{(\alpha)}) - \lambda \cdot b_u^{(\alpha)})$$

$$b_i^{(\alpha)} \leftarrow b_i^{(\alpha)} + \gamma \cdot ((w_{ui}^{(\alpha)} r_{ui} - \widehat{r}_{ui}^{(\alpha)}) - \lambda \cdot b_i^{(\alpha)})$$

$$x_u^{(\alpha)} \leftarrow x_u^{(\alpha)} + \gamma \cdot ((w_{ui}^{(\alpha)} r_{ui} - \widehat{r}_{ui}^{(\alpha)}) \cdot y_i^{(\alpha)} - \lambda \cdot x_u^{(\alpha)})$$

$$y_i^{(\alpha)} \leftarrow y_i^{(\alpha)} + \gamma \cdot ((w_{ui}^{(\alpha)} r_{ui} - \widehat{r}_{ui}^{(\alpha)}) \cdot x_u^{(\alpha)} - \lambda \cdot y_i^{(\alpha)})$$

$$w_{ui}^{(\alpha)}(q) \leftarrow \frac{e^{-|r_{ui} - \widehat{r}_{ui}^{(\alpha)}|}}{\sum_{\alpha=1}^{c} e^{-|r_{ui} - \widehat{r}_{ui}^{(\alpha)}|}}.$$

    end for

  end for

end for

for rating r_{ui} in R **do**

$$w_{ui}^{*(\alpha)} = \sum_{q=1}^{p} w_{ui}^{(\alpha)}(q)$$

$$\overline{w}_{ui}^{(\alpha)} = \frac{w_{ui}^{*(\alpha)}}{\sum_{\alpha=1}^{c} w_{ui}^{*(\alpha)}}$$

  for model $\alpha = 1$ to c **do**

$$\overline{r}_{ui}^{(\alpha)} = \overline{w}_{ui}^{(\alpha)} r_{ui}$$

  end for

end for

Output: latent component matrix $R^{(\alpha)} = \{\overline{r}_{ui}^{(\alpha)}\}_{n \times m}, \alpha = 1, ..., c$

Algorithm 1: The decomposition of user ratings.

Input: latent component matrix $R^{(\alpha)}, \alpha = 1, ..., c$

for model $\alpha = 1$ to c **do**

  for latent rating $\overline{r}_{ui}^{(\alpha)}$ in $R^{(\alpha)}$ **do**

$$b_u^{(\alpha)} \leftarrow b_u^{(\alpha)} + \gamma \cdot ((\overline{r}_{ui}^{(\alpha)} - \widehat{r}_{ui}^{(\alpha)}) - \lambda \cdot b_u^{(\alpha)})$$

$$b_i^{(\alpha)} \leftarrow b_i^{(\alpha)} + \gamma \cdot ((\overline{r}_{ui}^{(\alpha)} - \widehat{r}_{ui}^{(\alpha)}) - \lambda \cdot b_i^{(\alpha)})$$

$$x_u^{(\alpha)} \leftarrow x_u^{(\alpha)} + \gamma \cdot ((\overline{r}_{ui}^{(\alpha)} - \widehat{r}_{ui}^{(\alpha)}) \cdot y_i^{(\alpha)} - \lambda \cdot x_u^{(\alpha)})$$

$$y_i^{(\alpha)} \leftarrow y_i^{(\alpha)} + \gamma \cdot ((\overline{r}_{ui}^{(\alpha)} - \widehat{r}_{ui}^{(\alpha)}) \cdot x_u^{(\alpha)} - \lambda \cdot y_i^{(\alpha)})$$

  end for

end for

Output: $b_u^{(\alpha)}, b_i^{(\alpha)}, x_u^{(\alpha)}, y_i^{(\alpha)}, \alpha = 1, ..., c$

Algorithm 2: Model retraining.

3.4 Models retraining

After the above processes, c learned models, as well as c latent components of each rating can be obtained. For those learned models, if we combine them linearly to generate predictive ratings, most of the predictive scores would exceed 5 point (assuming the rating score ranges from 1 to 5). Therefore, it is necessary to retrain those c latent models. For all observed ratings, we pick out their αth latent component to retrain the learned model Θ_α mentioned above. The objective function for the according model Θ_α is similar to Eq. 3:

$$\min_{x^*,y^*,b^*} \sum_{u,i} (\overline{r}_{ui}^{(\alpha)} - \widehat{r}_{ui}^{(\alpha)})^2 + \lambda(\|x_u^{(\alpha)}\|^2 + \|y_i^{(\alpha)}\|^2 + (b_u^{(\alpha)})^2 + (b_i^{(\alpha)})^2), \tag{11}$$

where r_{ui} in Eq. 3 is replaced by latent component $\overline{r}_{ui}^{(\alpha)}$. We repeated this process for each learned model. It is worth mentioning that the learned model Θ_α cannot be reinitialized since they have preserved the information of original ratings. The model retraining process is given in Algorithm 2.

3.5 Rating predictions

Then, c new latent factor models were obtained. By combining those models linearly, we can get the final predictive score between user u and item i:

$$\widehat{r}_{ui} = \sum_{\alpha=1}^{c} \widehat{r}_{ui}^{(\alpha)}, \tag{12}$$

where $\widehat{r}_{ui}^{(\alpha)}$ is computed by Eq. 5.

In order to demonstrate the process of our method, we randomly select five real ratings in the *MovieLens* dataset and present their decompositions in Fig. 1. Firstly, the gradient descent method is utilized to compute the weights of latent models with respect to those five ratings, namely $W^{(1)}, W^{(2)}, \ldots, W^{(5)}$ in Fig. 1. Secondly, the weights of latent models are exploited to decompose each rating into five latent components. For instance, the rating $r_1 = 4$ is decomposed into 0.92, 0.88, 0.76, 0.68 and 0.76. Thirdly, each group of the latent component is chosen to retrain the previous learned models. For example, the first group of latent component $\overline{R}^{(1)}$ is selected to retrain the first latent model Θ_1. Finally, each new latent model can generate a predictive rating, and we linearly combine them as the final rating for a given user.

To sum up, our method can be divided into two steps: rating decomposition and model retraining. Our method is more complicated than the traditional matrix factorization method but its running time grows linearly with the traditional MF. The details can be found in the *Experiments* section.

4 Experiments

4.1 Experimental Setup

In order to evaluate the accuracy of our method, five benchmark datasets are selected, namely MovieLens, Douban, Movietweetings, Epinions and Goodreads. MovieLens is a

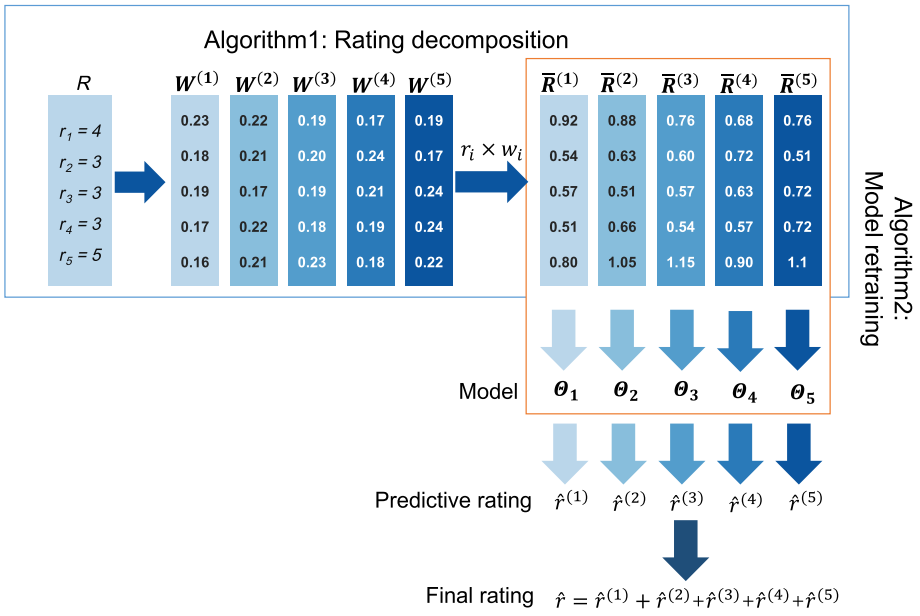

Fig. 1 A visualization of LCR method

movie recommendation Web site, which uses individual users' rating to generate personalized recommendations [56]. Douban, launched on March 6, 2005, is a Chinese Web 2.0 Web site providing user reviews and recommendation services of movies, books, and music [57]. The raw data contain user activities before August 2010, and we filtered out those users who have rated fewer than 20 movies. Movietweetings is a dataset consisting of ratings on movies which were contained in well-structured tweets on Twitter [58]. The raw dataset consists of 45,604 users and 26,165 items. Epinions is a Web site where people can review products, and the raw data consist of 876,252 users and 120,492 items. Goodreads is a book sharing and recommendation Web site, and the collected data contain user activities before August 2010. As it is quite difficult to provide accurate recommendations for inactive users, we filtered out users who have rated less than 20 items. The detailed information of datasets is presented in Table 2, where the user degree k_u is defined as the number of items that the user has rated and $\langle k_u \rangle$ denotes the average user degree over all users. The more items a user has rated, the more accurate a predictive model can be obtained. Both the sparsity and the user degree affect the accuracy of a recommendation method [59,60].

Each dataset is randomly divided into two parts: the training set (E^T) and the test set (E^P). The training set and test set consist of 70 and 30% of the original data, respectively. We make use of the cross-validation to obtain the accuracy of each recommendation method, which is computed based on 5 independent instances of training and test set [31]. The commonly used root-mean-square error (RMSE) was adopted to evaluate the accuracy of methods, which could be expressed as:

$$RMSE = \sqrt{\frac{1}{|E^P|} \sum_{r_{ui} \in E^P} \left(\hat{r}_{ui} - r_{ui}\right)^2}. \tag{13}$$

Table 2 The statistics of five benchmark datasets, MovieLens, Douban, Movietweetings, Epinions and Goodreads

Dataset	#Users, n	#Items, m	#Ratings, l	Sparsity, s (%)	$\langle k_u \rangle$
MovieLens	6040	3706	1,000,209	4.47	165.6
Douban	6472	7755	2,147,843	4.28	331.9
Movietweetings	2331	1669	196,359	5.04	84.2
Epinions	66,512	12,631	5,909,085	0.70	88.8
Goodreads	96,131	39,704	12,577,677	0.33	130.8

The table shows the number of user (n), number of items (m) and number of ratings (l) for each dataset accordingly. The $Sparsity$ is computed by $\frac{l}{n \times m}$

4.2 Comparison models

Eight methods in total were selected to compare with our method. *SlopOne* is an item-based collaborative filtering approach which was chosen as the benchmark method [61]. Five matrix factorization methods were selected, which were *biased matrix factorization* (BMF for short), *SVD++*, *nonnegative matrix factorization* (NMF for short), *probabilistic matrix factorization* (PMF for short) and *Bayesian probabilistic matrix factorization* (BPMF for short). Two cluster-based methods, which were *Latent Dirichlet Bayesian co-clustering method* (LDCC for short) and *Bayesian user community model* (BUCM for short). The cluster-based method partitions individual ratings in a macro-level, whereas our method decomposes ratings in a micro-level. Therefore, we adopted those two cluster-based methods to compare with our method. Details of these methods are given as follows:

- **SlopOne** SlopOne is the simplest form of non-trivial item-based collaborative filtering algorithm based on ratings [61], which is chosen as the benchmark method.
- **Biased Matrix Factorization (BMF)** A latent factor model which directly models only the observed ratings and prevents overfitting with a regularized term [26]. BMF is adopted as the base model of LCR. Since our method has more parameters than BMF, we train multiple BMF models simultaneously and average their predictive scores (MBMF for short). This method is chosen since it shares the same number of parameters with our method.
- **SVD++** A latent factor model which makes use of implicit feedback information of users [26]. The implicit feedback refers to users' history information which indicates their preference.
- **Nonnegative matrix factorization (NMF)** A matrix factorization method which constraints its factorization results to be nonnegative. The learned nonnegative vectors are the sparse representation of the users and items [62].
- **Probabilistic Matrix Factorization (PMF)** and **Bayesian Probabilistic Matrix Factorization (BPMF)** As a probabilistic view of matrix factorization, the PMF is able to scale linearly with the observed ratings and has good performance on very sparse and imbalance data [21]. The BPMF can prevent overfitting through integrating over all model parameters and hyperparameters [22].
- **Latent Dirichlet Bayesian Co-Clustering (LDCC)** This method is an extension of Bayesian co-clustering model, and uses collapsed Gibbs sampling and collapsed variational inference for parameter estimation [63].

– **Bayesian User Community Model (BUCM)** Relied on both item selection and rating emission, the BUCM generates communities for users who experience the same items and tend to adopt the same rating pattern. Each user is modeled as a random mixture of topics, where each topic is characterized by a distribution modeling the popularity of items within the respective user community and by a distribution over preference values for those items [64].

4.3 Results and analysis

4.3.1 The weights of models

In our method, the weight w_{ui}^{α} obtained by Eq. 7 is used to decompose a rating. Therefore, the stability of our method would be significantly affected by the convergency of w_{ui}^{α}. Thus, we randomly selected a rating and plotted the convergency of w_{ui}^{α} as shown in the left of Fig. 2. In total, five latent factor models were exploited to decompose ratings. From Fig. 2, it can be seen that the weights of models stay stable after the third iteration. In addition, it can be indicated by the varied weights of latent models that users may have complex preferences. Moreover, the average weight of each latent model over all ratings was computed, and the results can be found in the right of Fig. 2. It can be seen that there is almost no difference among the average weights of latent models, which means in a micro-level (a single rating), the latent models are different from each other; however, they are unbiased in a macro-level (all ratings). It is of great importance for our method to be unbiased. If our method is biased, then one of the latent models (denoted by Θ_{α}) will have the maximum weight, which is significantly larger than the weight of remaining models. As times of iteration grow infinitely, the weight of model Θ_{α} will converge to 1 and weights of the remaining models will converge to 0. If such a result is obtained, then it indicates that individual ratings cannot be decomposed into latent components.

For a given rating, each model is assigned a weight according to its predictive error. We studied the weight distribution of a model over all individual ratings. The result is presented in Fig. 3, where the weight distribution of each model is presented independently in subfigures. It can be seen that the weight distribution is close to Gaussian distribution with mean 0.2.

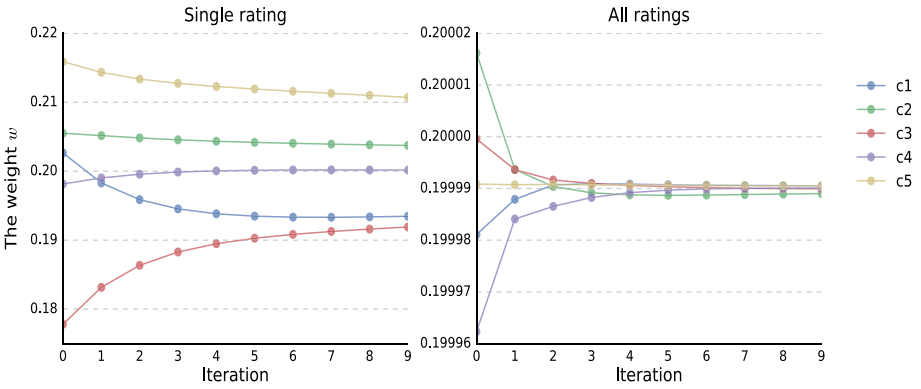

Fig. 2 The convergence of w, i.e., the weights of rating components. The left plot shows the convergence of w in a single rating entity and the right plot shows that with respect to all ratings, that is, the average of all entities. The weights of different components are denoted by different colors

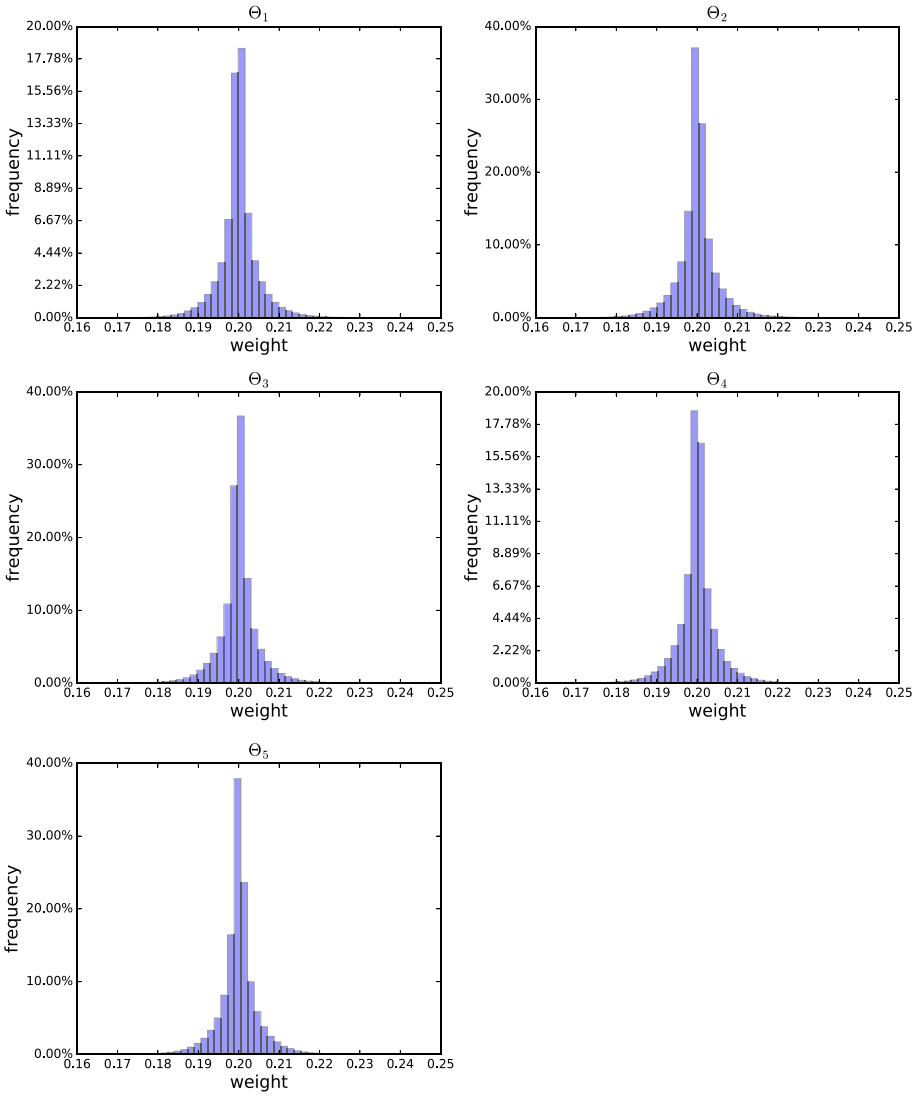

Fig. 3 The weight distribution of latent models

This is because in the decomposition of user ratings, we utilized the same predictive model and the same weight calculation formula. As a result, weights of different models are close to each other. The rating prediction results in Table 3 show that our method is superior to existing approaches, which implies the rationality of the weight distribution. Users might have similar ratings on different criteria.

4.3.2 The performance of LCR

Meanwhile, we compared our method with state-of-the-art approaches. The results are presented in Table 3, where it can be seen that our method enjoys the best predictive accuracy

Table 3 The accuracy of recommendation methods with respect to RMSE with the standard error shown within the brackets

	MovieLens	Douban	Movietweetings	Epinions	Goodreads
SlopOne	0.9024 (.0013)	0.7266 (.0005)	1.3266 (.0043)	0.3627 (.0008)	0.8602 (.0003)
BMF	0.8786 (.0014)	0.7303 (.0003)	1.3301 (.0073)	0.3691 (.0006)	0.8580 (.0002)
MBMF	0.8774 (.0019)	0.7303 (.0003)	1.3002 (.0065)	0.3690 (.0006)	0.8567 (.0005)
NMF	0.9322 (.0026)	0.7458 (.0065)	1.4067 (.0069)	0.4071 (.0005)	0.8959 (.0004)
PMF	0.8875 (.0018)	0.7371 (.0012)	1.4312 (.0032)	0.3908 (.0029)	0.8737 (.0016)
BPMF	0.8786 (.0015)	0.7227 (.0002)	1.4064 (.0032)	0.3752 (.0009)	0.8590 (.0002)
SVD++	0.8729 (.0021)	0.7266 (.0005)	1.3211 (.0059)	0.3672 (.0007)	0.8535 (.0006)
LDCC	0.9387 (.0066)	0.7472 (.0009)	1.4413 (.0040)	0.3820 (.0019)	0.8883 (.0003)
BUCM	0.9605 (.0057)	0.7775 (.0008)	1.4711 (.0295)	0.3910 (.0116)	0.9404 (.0010)
LCR	**0.8653 (.0008)**	**0.7191 (.0001)**	**1.2937 (.0005)**	**0.3619 (.0001)**	**0.8422 (.0005)**

Bold values indicate the best results

in all datasets. Its worth mentioning that the benchmark method SlopOne outperforms some matrix factorization-based methods in Douban and Epinions datasets. It may be caused by the sparsity of datasets [65]. Those matrix factorization-based methods require adequate data for training to generate accurate recommendations. Nevertheless, our method is better than SlopOne in both relatively dense and sparse datasets, which indicates that our method can alleviate the sparsity problem of recommender systems to some extent. Motivated by the results in Table 3, we explored the relationship between the accuracy of our method and the sparsity of datasets further. We change the sparsity of training set from 50 to 90% of the original data. Since different divisions of the dataset may have an influence on the sparsity of the training set, we run recommendation approaches based on one instance of training set and test set to eliminate the impact of different data partitions on the results. The results are presented in Fig. 4, in which the x-axis denotes the ratio of the training set to the whole dataset (training size) and the y-axis is the accuracy of algorithms running on the corresponding training set. For MovieLens, Douban and Goodreads datasets, the improvement in our method is more significant when taking into account fewer ratings. However, for Movietweetings and Epinions datasets, we have obtained a contrary result that the improvement in our method is less significant with a smaller size of the training set.

In order to find the possible reason, we studied the user degree which is defined as the number of items that the user has collected, as presented in Table 2. It can be seen that Movietweetings and Epinions have smaller user degrees compared to the remaining datasets. When the size of the training set becomes smaller, many users lack sufficient data to train an accurate prediction model, which leads to a decrease improvement in our method. Thus, our method could be helpful to alleviate the sparsity problem on condition that users' data are adequate to train a prediction model.

Our method has c latent factor models, and thus, our method has c times additional parameters than the base biased matrix factorization model. For example, our method exploits c vectors with $k - dimension$ to represent a user's preference, while BMF just applies one vector. Therefore, the possibility of improvement in our method may be caused by introducing the additional parameters. For this claim to be convincing, our method was compared with matrix factorization for overall ratings but with c times the number of latent factors (MBMF in Table 3). In this way, the two approaches would share the same number of parameters, and

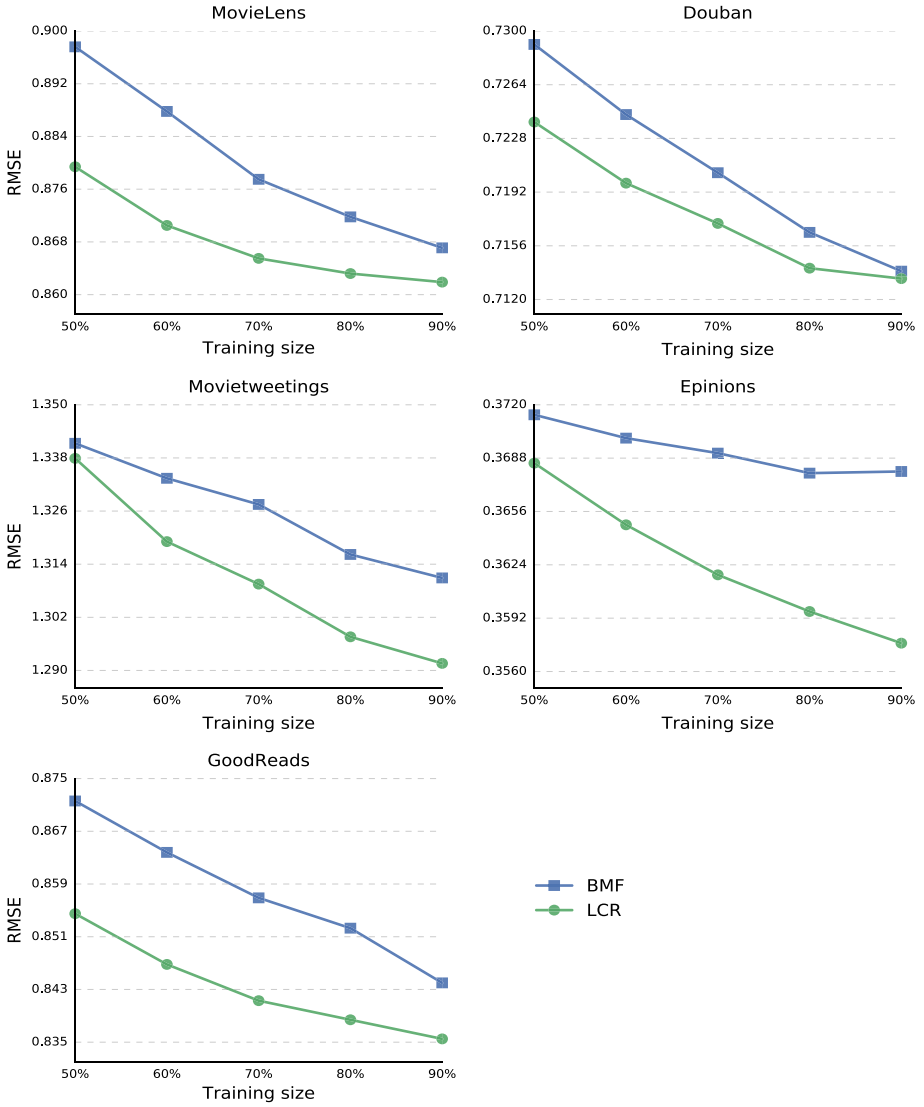

Fig. 4 The relationship between LCR's accuracy and the sparsity of dataset is shown in the plots. Each plot represents the prediction result in one dataset on RMSE. We use blue-square line and green-circle line to denote BMF and LCR, respectively

we can see if one model learns the structure of the data better than the other. From Table 3, it can be seen that our method outperforms MBMF, which demonstrates the rationality of our method.

From Table 3, it can be seen that the performance of our method is close to the baseline approaches in some datasets. In order to analyze the differences between our method and the baseline approaches, we employ the sign test which requires few assumptions about the distributional form of the data [4]. In the sign test, we count the number of users for whom our proposed algorithm outperforms the baseline algorithm (n_A) and the number of users for

Table 4 The significant test of the performance of algorithms

	MovieLens	Douban	Movietweetings	Epinions	Goodreads
$n_A/(n_A+n_B)*100\%$					
SlopOne	71%	64%	59%	54%	62%
BMF	60%	70%	67%	69%	66%
MBMF	55%	57%	60%	68%	61%
NMF	81%	81%	77%	85%	81%
PMF	73%	74%	78%	81%	73%
BPMF	69%	61%	71%	70%	67%
SVD++	63%	66%	54%	64%	65%
LDCC	78%	77%	77%	41%	77%
BUCM	78%	82%	77%	55%	79%
Average	70%	70%	69%	65%	70%
z^* score					
SlopOne	33.22	23.24	8.29	21.50	76.90
BMF	16.21	32.79	16.41	99.52	99.39
MBMF	7.23	10.52	9.65	94.23	70.43
NMF	48.61	50.52	26.56	182.80	192.50
PMF	35.20	38.61	27.35	157.66	143.35
BPMF	28.77	17.80	20.34	104.95	105.93
SVD++	20.18	25.93	4.14	72.03	91.36
LDCC	43.21	43.43	25.73	−47.82	166.62
BUCM	43.57	51.56	26.10	27.57	182.21

whom the baseline algorithm outperforms our algorithm (n_B). The significance level is the probability that our method is not truly better than the baseline method, and is estimated as the probability of at least n_A out of $n = n_A + n_B$ 0.5-probability Binomial trials succeeding, and is given by $p = (0.5)^n \sum_{i=n_A}^{n} \frac{n!}{i!(n-i)!}$. When n is large, we can take advantage of normal distribution to approximate the binomial. We then compute z^* score by $z^* = \frac{n_A - 0.5n}{\sqrt{n/4}}$. If $|z^*| < 1.96$, our method and the baseline method are not significantly different with at least 95% confidence. If $z^* > 0$, there are at least half users for whom our method outperforms the baseline method.

The result of $n_A/(n_A+n_B)*100\%$ and z^* scores are presented in Table 4. From the table, one can see that for most users (about 60–70% on average) our proposed method outperforms those baseline methods. There is one exception that our method is not as good as the LDCC algorithm for the Epinions dataset. There are only about 41% users for whom our method outperforms the LDCC method. From Table 3, the overall RMSE of our method is smaller than the overall RMSE of LDCC method. The result is possible because we count n_A and n_B without considering the differences in RMSE scores for different users. As mentioned before, Epinions dataset has a number of inactive users which may lack sufficient data to train an accurate model. Moreover, Park et al. pointed out that the cluster-based method may enjoy a better performance in the sparse dataset [66]. From the result of z^* scores, it can be seen that the differences between our method and those baseline approaches are significant ($|z^*| > 1.96$).

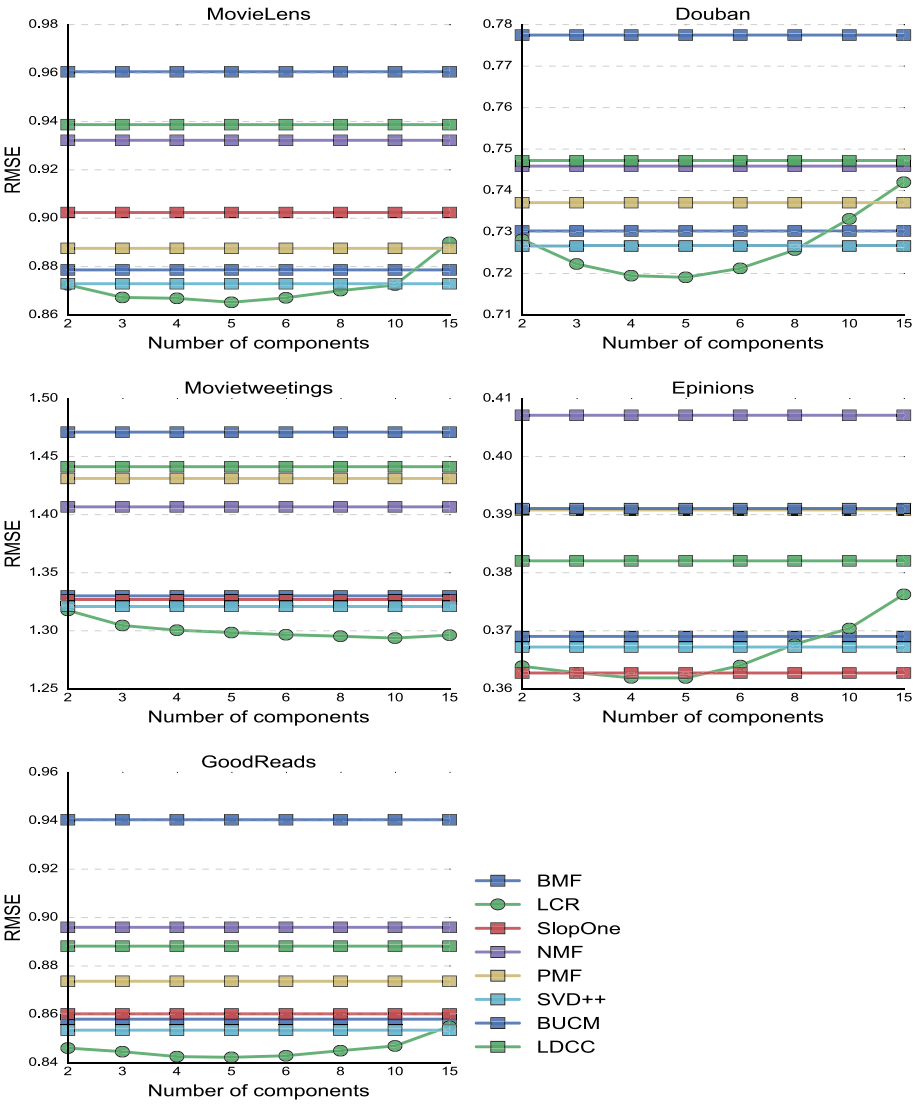

Fig. 5 The relationship between LCR's accuracy and the number of latent components is presented in the plots. Each plot represents the prediction results in one dataset on RMSE. Different approaches are denoted by different colors

4.3.3 The number of the latent components

In our method, the number of latent component is in fact a parameter, which can affect the performance of the method if an improper value is given. Thus, we investigated the relationship between the accuracy of our method and the number of latent components of ratings, as shown in Fig. 5. From the figure, it can be seen that our method achieves the best predictive accuracy when the number of the latent component equals to 5 except the Movietweetings dataset, in which the optimal number of the latent component is 10. It is interesting to obtain the result since ratings of Movietweetings dataset range from 1 to 10,

and ratings of the remaining datasets range from 1 to 5, which mean the optimal number of the latent component uncovered by our method equals to the maximal value of ratings in the system. Therefore, it is quite easy to determine the optimal number of these latent components.

4.3.4 Comparison with multi-criteria-based methods

BeerAdvocate and TripAdvisor datasets are selected to compare our method with five multi-criteria-based methods. In both datasets, users give not only an overall rating on an item, but also multiple ratings on different attributes (i.e., multi-criteria ratings) of the items. The BeerAdvocate dataset allows an individual user to rate four attributes (aroma, appearance, palate and taste) of beer, which consists of 1000 users, 3595 items and 586,826 ratings [31]. The TripAdvisor dataset was crawled by Wang et al. [67] which consists of 1725 users, 3343 items and 29,962 ratings. We filtered out those users who have rated fewer than 10 items. Our purpose is to predict overall ratings of users, and we also make use of RMSE to measure the accuracy of algorithms. The test set contains 30% of ratings, and the remaining ratings are used to train models. Our method only exploits overall ratings to learn parameters in models, and those multi-criteria-based methods take advantage of both overall ratings and multi-criteria ratings. For those methods, parameters are initialized from the Gaussian distribution with mean 0 and variance 0.1. Details of these methods are given as follows:

- **Multi-linear singular value decomposition (MSVD) [49]** The MSVD can take full advantages of the multi-dimensional representation capability of a higher-order tensor. The approximation of MSVD is obtained by discarding the smallest n-mode singular values for a given value (e.g., 0).
- **Support vector regression (SVR) [46]** This method firstly predicts the multiple criteria rating by context-aware matrix factorization and then uses the aggregation-based approach to build a linear hybrid of use-specific and item-specific aggregation models.
- **Clustering and Regression collaborative filtering (CRCF) [68]** This method takes advantage of both clustering and regression techniques. More specifically, it employs the principal component analysis (PCA) for dimensionality reduction and makes use of Classification and Regression Tree (CART) and Expectation Maximization (EM) for accuracy improvement in multi-criteria recommender systems.
- **Multi-Criteria Aggregation (MA) [69]** This method aggregates performance measures over all criteria based on inferences about preferences from the decision-maker's input and exploits the Choquet integral of a fuzzy measure to determine a total ordering of the subset of criteria.
- **Multiple Regressions (MR) [70]** This method makes use of multiple regressions to analyze the relationship between users' overall assessment and multi-criteria rating dimensions. We utilize the multiple regressions method to predict the overall rating of users by integrating the multi-criteria ratings.

  The comparison results are presented in Table 5. For the BeerAdvocate dataset, our method significantly outperforms those multi-criteria-based approaches. For the TripAdvisor dataset, some regression-based methods (SVR, MA and MR) are better than our method. This is because plenty of users in TripAdvisor dataset have few ratings which make the data insufficient to train accurate models. In addition, only the overall ratings are taken into consideration in our method. Therefore, one possible way to improve our method is to combine those multi-criteria ratings. For instance, we can make use of those multi-criteria ratings to train several latent models and integrate their predictive scores.

Table 5 The performance of algorithms on the multi-criteria datasets

	BeerAdvocate	TripAdvisor
MSVD	1.231	2.467
SVR	1.340	1.030
CRCF	1.213	1.533
MA	0.600	1.030
MR	0.648	1.090
LCR	0.571	1.297

The dimensions (k_1, k_2, k_3) of the matrix S in MSVD are 100, 100 and 5, respectively. The cluster number of CRCF method is set to 5

Table 6 The running time of LCR, in seconds. We compare the running time of LCR to its base model BMF

	BMF	LCR^2	LCR^3	LCR^4	LCR^5	LCR^5(parallel)
MovieLens	24.0	80.6	111.4	143.3	184.8	64.6
Douban	48.9	173.5	234.2	327.4	400.9	116.9
Movietweetings	6.9	24.5	31.6	40.5	50.3	17.2
Epinions	167.6	576.8	813.7	1131.1	1327.6	472.4
Goodreads	574.2	1717.6	2387.8	3003.5	3482.2	1592.5

The number of components for LCR is denoted by the superscript, for example, LCR^3 denoted LCR with three components

4.3.5 The complexity of LCR

As mentioned above, the running time of our method grows linearly with the traditional matrix factorization. It can be found in Table 6 the comparison between the running time of LCR and its based model BMF. As it is mentioned in Sect. 3, the LCR model requires rating decomposition and retraining to accomplish its life cycle. In rating decomposition, LCR requires a minimization process by Eq. 6 to learn the weight of each component, which consumes one BMF time. In retraining, LCR needs to retrain all desired latent component models with Eq. 11. For each latent component model, LCR needs one BMF time. It turns out that the running time of LCR is approximately $1 + C$ times to the standard matrix factorization model, where C is the number of desired latent components. The $1 + C$ times ratio can be reflected by our running time experiments. In Table 6, taking LCR^5(with 5 latent components), for example, the running time of LCR is approximately six to seven times to the running time of BMF in all datasets. In practice, the running time ratio might exceed a bit due to the normalization for the weight of components during the operation of algorithms in Eqs. 7, 9 and 10. Thus, it is verified that the running time of our model grows linearly with the traditional matrix factorization. As a matter of fact, the retraining process in our method can be executed in parallel since those latent models are independent to each other. We test LCR^5 (with 5 latent components) and the running time is given in the last column of Table 6. It can be seen that the running time of our method is significantly decreased.

5 Conclusion

In this paper, it is assumed that a user's evaluation on items relies on several criteria, and her explicit ratings can be decomposed into different latent components. To determine the

latent components, a new recommendation approach, called LCR, was proposed, which made use of cost-sensitive learning strategy to train several latent models simultaneously and recorded the weight of each model according to their predictive error. By exploiting the weight of models, an individual rating is finally decomposed into several latent components. Those latent components are then used again to retrain the latent models. Finally, we linearly combine the latent model to generate the final rating of a given user.

When we study the weight of models, it is found that the weight of the model is relatively close. For example, when a rating is decomposed into five latent components, the weight of each model is close to 0.2. We compared the latent rating uncovered by our method with the real multi-criteria rating data (Beeradvocate.com) [31], and it is found that there is a significant difference between them. One possible reason is that before the training, all latent models are treated equally in our method. As a result, their weights are close to each other. Therefore, one possible way to improve our method is treating those latent models differently. For instance, we can utilize different ways to compute weights of different models. Moreover, to simplify the computation, those latent factor models are supposed to be independent to each other and are trained simultaneously. In the future work, the situation that latent models are dependent to each other will be considered. Then, potential approaches such as collective matrix factorization [71] and transfer learning [72] can be taken into account.

In our method, the number of latent component is in fact a parameter, which can affect the performance of the method if an improper value is given. It is found that the optimal number of the latent component uncovered by our method equals to the maximal value of ratings in the system. Thus, it is quite easy to determine the optimal number of latent models, which indicates that our method is not sensitive to the number of latent components. There may be varied reasons for this result. One possible reason may be that our method averages predictive scores over several latent models, which may eliminate the fluctuations caused by one single model to some extent. When the number of models equals to the maximal value of the rating, the elimination effect reaches a maximum. Another possible reason may be that individual users have several criteria in mind and the number of criteria is related to the range of the rating. For instance, if the maximal value of a rating is set to 5, users may choose five criteria in mind when they evaluate an item. If this assumption holds true, our method may be helpful to determine the range of a rating. We will investigate this issue in our future work.

In the paper, we compared our method with some multi-criteria-based approaches. For the relatively sparse dataset, some regression-based methods are better than our method since they take into account more information than our method. This result provides a potential way to improve our method. As a matter of fact, our method can be well extended to the multi-criteria recommender system. The explicit rating of users on varied criteria can be used to train a better predictive model. We will try this method in our future work.

Acknowledgements This work was supported by the National Natural Science Foundation of China (61502078) and Scientific Research start-up Foundation (ZYGX2015KYQD073).

References

1. Nilashi M, Ob Ibrahim, Ithnin N (2014) Multi-criteria collaborative filtering with high accuracy using higher order singular value decomposition and Neuro-Fuzzy system. Knowl Based Syst 60:82–101
2. Nilashi M, Jannach D, Ob Ibrahim, Ithnin N (2015) Clustering- and regression-based multi-criteria collaborative filtering with incremental updates. Inf Sci 293:235–250

3. Adomavicius G, Kwon Y (2007) New recommendation techniques for multicriteria rating systems. IEEE Intell Syst 22(3):48–55
4. Ricci F, Rokach L, Shapira B (2011) Recommender systems handbook
5. Plantie M, Montmain J, Dray G (2005) Movies recommenders systems: automation of the information and evaluation phases in a multi-criteria decision-making process. In: International conference on database and expert systems applications, Copenhagen, Denmark, pp 633–644
6. Matsatsinis NF, Samaras AP (2001) MCDA and preference disaggregation in group decision support systems. Eur J Oper Res 130(2):414–429
7. Adomavicius G, Tuzhilin A (2005) Toward the next generation of recommender systems: a survey of the state-of-the-art and possible extensions. IEEE Trans Knowl Data Eng 17(6):866–883
8. Deshpande M, Karypis G (2004) Item-based top-N recommendation algorithms. ACM Trans Inf Syst 22(1):143–177
9. Hill WC, Stead L, Rosenstein M, Furnas GW (1995) Recommending and evaluating choices in a virtual community of use. In: Proceedings of the SIGCHI conference on human factors in computing systems, Denver, Colorado, USA, pp 194–201
10. Hofmann T (2003) Collaborative filtering via gaussian probabilistic latent semantic analysis. In: Proceedings of the 26th annual international ACM SIGIR conference on research and development in information retrieval, Toronto, Canada, pp 259–266
11. Blei MD, Ng AY, Jordan MI (2003) Latent Dirichlet allocation. J Mach Learn Res 3(1):993–1022
12. Salakhutdinov R, Mnih A, Hinton GE (2007) Restricted Boltzmann machines for collaborative filtering. In: Proceedings of the 24th international conference on machine learning, Corvallis, Oregon, USA, pp 791–798
13. Tang J, Wu S, Sun J, Su H (2012) Cross-domain collaboration recommendation. In: Proceedings of the 18th ACM SIGKDD international conference on knowledge discovery and data mining, Beijing, China, pp 1285–1293
14. Resnick P, Iacovou N, Suchak M, Bergstrom P, Riedl J (1994) GroupLens: an open architecture for collaborative filtering of netnews. In: Proceedings of the 1994 ACM conference on computer supported cooperative work, Chapel Hill, NC, USA, pp 175–186
15. Konstan JA, Miller BN, Maltz D, Herlocker JL, Gordon LR, Riedl J (1997) GroupLens: applying collaborative filtering to usenet news. Commun ACM 40(3):77–87
16. Linden G, Smith B, York J (2003) Amazon.com recommendations: item-to-item collaborative filtering. IEEE Internet Comput 7(1):76–80
17. Liu L, Meh N, jiev , Xu D (2011) Multi-criteria service recommendation based on user criteria preferences. In: Proceedings of the 5th ACM conference on recommender systems, Chicago, IL, USA, pp 77–84
18. Alqadah F, Reddy CK, Hu J, Alqadah HF (2015) Biclustering neighborhood-based collaborative filtering method for top-n recommender systems. Knowl Inf Syst 44(2):475–491
19. Breese JS, Heckerman D, Kadie CM (1998) Empirical analysis of predictive algorithms for collaborative filtering. In: Proceedings of the 14th conference on uncertainty in artificial intelligence, Madison, Wisconsin, USA, pp 43–52
20. Koren Y, Bell RM, Volinsky C (2009) Matrix factorization techniques for recommender systems. IEEE Comput 42(8):30–37
21. Ruslan S, Andriy M (2007) Probabilistic matrix factorization. In: Proceedings of the 20th international conference on neural information processing systems, Vancouver, BC, Canada, pp 1257–1264
22. Salakhutdinov R, Mnih A (2008) Bayesian probabilistic matrix factorization using Markov chain Monte Carlo. In: Proceedings of the 25th international conference on machine learning, Helsinki, Finland, pp 880–887
23. Lee J, Kim S, Lebanon G, Singer Y (2013) Local low-rank matrix approximation. In: Proceedings of the 30th international conference on machine learning, Atlanta, GA, USA, pp 82–90
24. Fu Y, Liu B, Ge Y, Yao Z, Xiong H (2014) User preference learning with multiple information fusion for restaurant recommendation. In: Proceedings of the 2014 SIAM international conference on data mining, Philadelphia, Pennsylvania, USA, pp 470–478
25. Kannan R, Ishteva M, Park H (2014) Bounded matrix factorization for recommender system. Knowl Inf Syst 39(3):491–511
26. Koren Y (2008) Factorization meets the neighborhood: a multifaceted collaborative filtering model. In: Proceedings of the 14th ACM SIGKDD international conference on knowledge discovery and data mining, Las Vegas, Nevada, USA, pp 426–434
27. Joonseok L, Samy B, Seungyeon K, Guy L, Yoram S (2014) Local collaborative ranking. In: Proceedings of the 23rd international conference on world wide web, Seoul, Republic of Korea, pp 85–96
28. Koren Y (2010) Collaborative filtering with temporal dynamics. Commun ACM 53(4):89–97

29. Chua FCT, Oentaryo RJ, Lim E (2013) Modeling temporal adoptions using dynamic matrix factorization. In: IEEE 13th international conference on data mining, Dallas, TX, USA, pp 91–100
30. Zhang C, Wang K, Yu H, Sun J, Lim E (2014) Latent factor transition for dynamic collaborative filtering. In: Proceedings of the 2014 SIAM international conference on data mining, Philadelphia, Pennsylvania, USA, pp 452–460
31. McAuley JJ, Leskovec J (2013) From amateurs to connoisseurs: modeling the evolution of user expertise through online reviews. In: Proceedings of the 22nd international conference on world wide web, Rio de Janeiro, Brazil, pp 897–908
32. Li X, Cong G, Li X, Pham TN, Krishnaswamy S (2015) Rank-GeoFM: a ranking based geographical factorization method for point of interest recommendation. In: Proceedings of the 38th international ACM SIGIR conference on research and development in information retrieval, Santiago, Chile, pp 433–442
33. Zhang J, Chow C (2015) GeoSoCa: exploiting geographical, social and categorical correlations for point-of-interest recommendations. In: Proceedings of the 38th international ACM SIGIR conference on research and development in information retrieval, Santiago, Chile, pp 443–452
34. Gao H, Tang J, Hu X, Liu H (2015) Content-aware point of interest recommendation on location-based social networks. In: Proceedings of the 29th AAAI conference on artificial intelligence, Austin, Texas, pp 1721–1727
35. Lian D, Zhao C, Xie X, Sun G, Chen E, Rui Y (2014) GeoMF: joint geographical modeling and matrix factorization for point-of-interest recommendation. In: Proceedings of the 20th ACM SIGKDD international conference on knowledge discovery and data mining, New York, NY, USA, pp 831–840
36. Qian X, Feng H, Zhao G, Mei T (2014) Personalized recommendation combining user interest and social circle. IEEE Trans Knowl Data Eng 26(7):1763–1777
37. Chaney AJ, Blei DM, Eliassi-Rad T (2015) A probabilistic model for using social networks in personalized item recommendation. In: Proceedings of the 9th ACM conference on recommender systems, Vienna, Austria, pp 43–50
38. Zhao Z, Zhang L, He X, Ng W (2015) Expert finding for question answering via graph regularized matrix completion. IEEE Trans Knowl Data Eng 27(4):993–1004
39. Shen Y, Jin R (2012) Learning personal + social latent factor model for social recommendation. In: Proceedings of the 18th ACM SIGKDD international conference on knowledge discovery and data mining, Beijing, China, pp 1303–1311
40. Jannach D, Karakaya Z, Gedikli F (2012) Accuracy improvements for multi-criteria recommender systems. In: Proceedings of the 13th ACM conference on electronic commerce, Valencia, Spain, pp 674–689
41. Leung CW, Chan SC, Chung F (2006) A collaborative filtering framework based on fuzzy association rules and multiple-level similarity. Knowl Inf Syst 10(3):357–381
42. Mikeli A, Sotiros D, Apostolou D, Despotis DK (2013) A multi-criteria recommender system incorporating intensity of preferences. In: 4th international conference on information, intelligence, systems and applications, Piraeus, Greece, pp 1–6
43. Manouselis N, Costopoulou C (2007) Experimental analysis of design choices in multiattribute utility collaborative filtering. Int J Pattern Recognit Artif Intell 21(2):311–331
44. Lee H, Teng W (2007) Incorporating multi-criteria ratings in recommendation systems. In: IEEE international conference on information reuse and integration, Las Vegas, Nevada, pp 273–278
45. Naime RK, Sasan AH (2017) A hybrid multi-criteria recommender system using ontology and neuro-fuzzy techniques. Electron Commer Res Appl 21(C):50–64
46. Zheng Y (2017) Criteria chains: a novel multi-criteria recommendation approach. In: 22nd international conference on intelligent user interfaces, Limassol, Cyprus, pp 29–33
47. Mehrbakhsh N, Bin IO, Norafida I (2014) Hybrid recommendation approaches for multi-criteria collaborative filtering. Expert Syst Appl 41(8):3879–3900
48. Sahoo N, Krishnan R, Duncan G, Callan JP (2006) Collaborative filtering with multi-component rating for recommender systems. In: Proceedings of the 16th workshop on information technologies and systems, Dublin, Republic of Ireland
49. Li Q, Wang C, Geng G (2008) Improving personalized services in mobile commerce by a novel multicriteria rating approach. In: Proceedings of the 17th international conference on world wide web, Beijing, China, pp 1235–1236
50. Nilashi M, Ibrahim OB, Ithnin N, Zakaria R (2015) Hybrid recommendation approaches for multi-criteria collaborative filtering. Soft Comput 19(11):3173–3207
51. McAuley JJ, Leskovec J, Jurafsky D (2012) Learning attitudes and attributes from multi-aspect reviews. In: the 12th IEEE international conference on data mining, Brussels, Belgium, pp 1020–1025
52. Pozo M, Chiky R, Metais E (2016) Enhancing collaborative filtering using implicit relations in data. Lectures Notes Comput Sci 9655:125–146

53. Arkadiusz P (2007) Improving regularized singular value decomposition for collaborative filtering. In: Proceedings of KDD cup and workshop, San Jose, CA, USA, pp 39–42
54. Guo G, Zhang J, Sun Z, Yorke-Smith N (2015) LibRec: A Java Library for Recommender Systems. In: Posters, demos, late-breaking results and workshop proceedings of the 23rd conference on user modelling, adaptation and personalization, Dublin, Ireland
55. Guo G, Zhang J, Yorke-Smith N (2015) TrustSVD: collaborative filtering with both the explicit and implicit influence of user trust and of item ratings. In: Proceedings of the 29th AAAI conference on artificial intelligence, Austin, Texas, USA, pp 123–129
56. Karumur RP, Nguyen TT, Konstan JA (2016) Exploring the value of personality in predicting rating behaviors: a study of category preferences on MovieLens. In: Proceedings of the 10th ACM conference on recommender systems, Boston, MA, USA, pp 139–142
57. Huang J, Cheng X, Shen H, Zhou T, Jin X (2012) Exploring social influence via posterior effect of word-of-mouth recommendations. In: Proceedings of the 5th international conference on web search and web data mining, Seattle, WA, USA, pp 573–582
58. Simon D, Toon DP, Luc M (2013) Movietweetings: a movie rating dataset collected from twitter. In: workshop on Crowdsourcing and human computation for recommender systems, Vienna, Austria, pp 43
59. Zeng W, Zeng A, Liu H, Shang MS, Zhou T (2014) Uncovering the information core in recommender systems. Sci Rep 4:6140
60. Shang MS, Lu L, Zhang YC, Zhou T (2010) Empirical analysis of web-based user-object bipartite networks. Europhys Lett 90(4):48006
61. Lemire D, Maclachlan A (2005) Slope one predictors for online rating-based collaborative filtering. In: Proceedings of the 2005 SIAM international conference on data mining, Newport Beach, CA, USA, pp 471–475
62. Lee DD, Seung HS (2000) Algorithms for non-negative matrix factorization. In: Proceedings of the 13th international conference on neural information processing systems, Denver, CO, USA, pp 556–562
63. Wang P, Domeniconi C, Laskey KB (2009) Latent Dirichlet Bayesian co-clustering. In: Proceedings of the European conference on machine learning and knowledge discovery in databases: Part II, Bled, Slovenia, pp 522–537
64. Barbieri N, Costa G, Manco G, Ortale R (2011) Modeling item selection and relevance for accurate recommendations: a bayesian approach. In: Proceedings of the 5th ACM conference on recommender systems, Chicago, IL, USA, pp 21–28
65. Fidel C, Victor C, Fernandez D, Formoso V (2011) Comparison of collaborative filtering algorithms: limitations of current techniques and proposals for scalable, high-performance recommender systems. ACM Trans Web 5(1):2:1–2:33
66. Park YJ, Tuzhilin A (2008) The long tail of recommender systems and how to leverage it. In: Proceedings of the 2008 ACM conference on recommender systems, Lausanne, Switzerland, pp 11–18
67. Wang H, Lu Y, Zhai C (2011) Latent aspect rating analysis without aspect keyword supervision. In: Proceedings of the 17th ACM SIGKDD international conference on knowledge discovery and data mining, San Diego, CA, USA, pp 618–626
68. Nilashi M, Esfahani MD, Roudbaraki MZ, Ramayah T, Ibrahim O (2016) A multi-criteria collaborative filtering recommender system using clustering and regression techniques. J Soft Comput Decis Support Syst 3(5):24–30
69. Fomba S, Zarate P, Kilgour M, Camilleri G, Konate J, Tangara F (2016) A recommender system based on multi-criteria aggregation. In: 2nd international conference on decision support systems technology—EURO working group on decision support systems, Plymouth, UK, pp 1–7
70. Fuchs M, Zanker M (2012) Multi-criteria ratings for recommender systems: an empirical analysis in the tourism domain. In: 13th international conference on electronic commerce and web technologies, Vienna, Austria, pp 100–111
71. Singh AP, Gordon GJ (2008) Relational learning via collective matrix factorization. In: Proceedings of the 14th ACM SIGKDD international conference on knowledge discovery and data mining, Las Vegas, Nevada, USA, pp 650–658
72. Jialin PS, Qiang Y (2010) A survey on transfer learning. IEEE Trans Knowl Data Eng 22(10):1345–1359

Junhua Chen received his B.Sc. degree at school of Computer Science and Engineering from University of Electronic Science and Technology of China (UESTC) in 2015 and he studied in a postgraduate program under the supervision of professor Wei Zeng in UESTC. Currently, Chen serves one-year internship at *Tencent* and focuses on the research of session-based recommendation. His research interests are recommender systems, point process and deep probabilistic models.

Wei Zeng received his Ph.D. degree at the University of Electronic Science and Technology of China, Chengdu, China, in 2015. Currently, he is an associate professor of Department of Computer Science at University of Electronic Science and Technology of China. He visited Department of Computer Science at Hong Kong Baptist University and Physics Department at University of Fribourg in 2011 and 2012, respectively. His main research interests include the data mining, network science and recommender systems.

Junming Shao received his Ph.D. degree with highest honor (Summa Cum Laude) at the University of Munich, Germany, in 2011. He became the Alexander von Humboldt Fellow in 2012. Currently, he is professor of Computer Science at the University of Electronic Science and Technology of China. His research interests include data mining and neuroimaging. He not only published papers on top-level data mining conferences like KDD, ICDM, SDM (two of those papers have won the Best Paper Award), but also published data mining-related interdisciplinary work in leading journals including Brain, Neurobiology of Aging, and Water Research.

Ge Fan is a M.Sc. student at school of Computer Science and Engineering, University of Electronic Science and Technology of China. In 2016, he obtained his B.Sc. degree in Information and Computing Science and B.B.M. degree in Financial Management from Sichuan Agricultural University, China. His research interests include Machine learning, Data Mining and Recommender Systems. He received the First-Class Award from China Postgraduate Mathematic Contest in Modeling. He also received the *First-Class Academic Scholarship*, *Excellent Program Award* and *special Contest Award* from UESTC and SAU.