Front. Comput. Sci., 2014, 8(2): 289–297
DOI 10.1007/s11704-013-3012-7

RESEARCH ARTICLE

Recommendation algorithm based on item quality and user rating preferences

Yuan GUAN, Shimin CAI, Mingsheng SHANG (✉)

Web Sciences Center, School of Computer Science and Engineering,
University of Electronic Science and Technology of China, Chengdu 611731, China

Abstract Recommender systems are one of the most important technologies in e-commerce to help users filter out the overload of information. However, current mainstream recommendation algorithms, such as the collaborative filtering CF family, have problems such as scalability and sparseness. These problems hinder further developments of recommender systems. We propose a new recommendation algorithm based on item quality and user rating preferences, which can significantly decrease the computing complexity. Besides, it is interpretable and works better when the data is sparse. Through extensive experiments on three benchmark data sets, we show that our algorithm achieves higher accuracy in rating prediction compared with the traditional approaches. Furthermore, the results also demonstrate that the problem of rating prediction depends strongly on item quality and user rating preferences, thus opens new paths for further study.

Keywords recommendation algorithm, item quality, user rating preferences, RMSE

1 Introduction

The rapid development of the Internet has taken us into a high-speed developing information age where it is impossible to browse all the information even once. The excess of information has seriously reduced the utilization ratio of information. To address this problem, recommender systems provide automatical intelligent recommendations for users based on their historical behavior on the Internet, e.g., their previous transactions or product ratings. There has been a great deal of research over the last 20 years in exploring recommender systems and a wide variety of methods have been proposed [1–3]. In addition to academic interest, recommender systems have been seeing significant interest from industry and been widely applied to e-commerce, specifically by Netflix, Amazon, and Taobao.

Broadly speaking, existing recommender systems follow two different strategies [1]: the content analysis approach and the collaborative filtering (CF). CF approaches [4, 5] are easy to understand and have been the most widely used methods so far. However, CF methods suffer from several issues of which scalability and sparseness are the most serious. Recommender systems need a large amount of computation power to work on large data sets, and in real applications, users (even the most active ones) have rated a very small subset of the overall number of items. Although many improved algorithms have been put forward, none of them have completely solved these problems. For example, matrix factorization [5–7] based on various mathematical theories as well as physical diffusion process [8–10] based on bipartite networks, aim to achieve scalability. And algorithms using additional sources of information such as associative retrieval [11], social relationships [12], social tags [13], attempt to deal with the problem of sparseness.

In this work, we concentrate on scalable solution of recommender systems and present a simpler and more efficient algorithm. The idea is inspired by a tendency-based (TB) al-

Received January 25, 2013; accepted September 18, 2013

E-mail: msshang@uestc.edu.cn

gorithm proposed recently in [14], which has very low computation requirement and can be used in large recommender systems. But our mechanism is rather different. In our proposed algorithm, we think the rating given by a user to an item should depend on the item quality and the user rating preference, where preference means relative rating bias (higher or lower rating). Different users may rate the same item differently: some users are more inclined to give high ratings, whereas other users give lower ratings. On the other hand, the rating given by the user is not completely subjective to this preference and an item with good quality is unlikely to get a low rating, and it is unusual if a bad item is given a high rating. Thus, we suppose that an active user's rating of an item depends on the item quality, and is influenced by the user rating preference.

Through experimentation we show that the two factors, item quality and user rating preferences, are crucial. By extensive experiments in Section 4 on three benchmark datasets, we show that the proposed algorithm can achieve a higher recommendation accuracy compared with the TB algorithm. In addition, our proposed algorithm does not need complex computation and is scalable to large systems. Moreover, it is interpretable and works when the data is sparse.

Section 2 introduces the related work. Section 3 presents the proposed algorithm. Then Section 4 gives empirical experiments. Finally, the summary and discussion are conducted in Section 5.

2 Related work

Most recommendation algorithms can be boiled down to the problem of score prediction, like the well-known Netflix competition which aims at the minimum prediction error. In real systems, users are asked to rate an item they have experienced with a score with five level ratings from 1 (worst) to 5 (best) to express their tastes. For a recommender system with M users and N items, it can be represented by a score matrix $\boldsymbol{V}_{MN}$. The observed element of $v_{\alpha,i} \in \boldsymbol{V}$ indicates the rating score of user α for item i. An unobserved element $v_{\alpha,i} = \varnothing$ implies user α has not rated item i. The task of recommender systems is to turn observed ratings for users into predictions for users' unobserved ratings. Table 1 lists all the notations of symbols which will be referred to in herein. We use Greek letters to represent user indexes, and use Latin letters to represent item indexes.

In the following section, we will describe several algorithms most related to our work, which will be compared in our experiments.

Table 1 The notations of symbols

Symbols	Description
$v_{\alpha,i}$	the rating score of user α for item i
$P_{\alpha,j}$	the predicted rating score of user α for item i
U	the set of all users in the system
I	the set of all items in the system
U_i	the set of users who have rated item i
I_α	the set of items user α has rated
U_α	the set of users who are similar to user α
I_i	the set of items which are similar to item i
$\overline{v_\alpha}$	the average rating score of user α
$\overline{v_i}$	the average score for item i
τ_α	the rating tendency of user α
τ_i	the score tendency of item i

2.1 Similarity-based CF

CF algorithms are quite understandable, and they obtain reasonably accurate results. Considering the similarity is based on users or items, CF can be divided into user-based (UCF) and item-based (ICF) types.

UCF [15] predicts the unknown ratings based on the ratings of similar neighbors for that target item. Pearson correlation coefficient concerns the rating inconsistency of different users, and we apply it to measure the similarity between two users. The Pearson similarity between user α and β is calculated as:

$$s(\alpha,\beta) = \frac{\sum\limits_{i\in I_\alpha \bigcap I_\beta} (v_{\alpha,i} - \overline{v_{\alpha.}})(v_{\beta,i} - \overline{v_{\beta.}})}{\sqrt{\sum\limits_{i\in I_\alpha \bigcap I_\beta} (v_{\alpha,i} - \overline{v_{\alpha.}})^2 \sum\limits_{i\in I_\alpha \bigcap I_\beta} (v_{\beta,i} - \overline{v_{\beta.}})^2}}. \quad (1)$$

After getting the similarity between the active user and other users, the predicted rating of user α for item i can be calculated as

$$p_{\alpha,i} = \overline{v_\alpha} + \frac{\sum\limits_{\beta\in U_\alpha} s(\alpha,\beta) \times (v_{\beta,i} - \overline{v_{\beta.}})}{\sum\limits_{\beta\in U_\alpha} |s(\alpha,\beta)|}. \quad (2)$$

Similarly, ICF [16] first calculates the similarity between items. The similarity between two items is defined by

$$s(i,j) = \frac{\sum\limits_{\alpha\in U_i \bigcap U_j} (v_{\alpha,i} - \overline{v_{\alpha.}})(v_{\alpha,j} - \overline{v_{\alpha.}})}{\sqrt{\sum\limits_{\alpha\in U_i \bigcap U_j} (v_{\alpha,i} - \overline{v_{\alpha.}})^2 \sum\limits_{\alpha\in U_i \bigcap U_j} (v_{\alpha,j} - \overline{v_{\alpha.}})^2}}. \quad (3)$$

Based on the similarity between items, ICF then makes

predictions with

$$p_{\alpha,i} = \frac{\sum_{j \in I_i} (s(i,j) \times v_{\alpha,j})}{\sum_{j \in I_i} |s(i,j)|}. \quad (4)$$

2.2 TB method

In 2011, Cacheda et al. [14] proposed a TB recommendation algorithm, which makes a big step forward towards making scalable recommender systems. There are four features involved in TB, namely the average rating a user gives cross all items, the average score an item gets, the score tendency of the item and the rating tendency of the user. Tendency is the relative rating bias with respect to the mean rating. The algorithm divides the recommendation situations into four cases: (1) when the tendency of user and the tendency of item are both positive; (2) when the tendency of user and the tendency of item are both negative; (3) when the tendency of user and the tendency of item are opposite in sign and the average rating scores of them corroborate their tendencies; (4) when the tendency of user and the tendency of item are opposite in sign and the average rating scores of them do not corroborate their tendencies. The prediction rule corresponding to each case is formulated by:

$$\begin{cases} p_{\alpha,i} = \max(\overline{v_{\alpha.}} + \tau_i, \overline{v_i} + \tau_\alpha), \\ p_{\alpha,i} = \min(\overline{v_{\alpha.}} + \tau_i, \overline{v_i} + \tau_\alpha), \\ p_{\alpha,i} = \min(\max(\overline{v_{\alpha.}}, (\overline{v_i}+\tau_\alpha)\mu + (\overline{v_{\alpha.}}+\tau_i)(1-\mu)), \overline{v_{.i}}), \\ p_{\alpha,i} = \overline{v_{.i}}\mu + \overline{v_{\alpha.}}(1-\mu). \end{cases} \quad (5)$$

3 The proposed algorithm

3.1 Item quality and user rating preferences

Users may rate items with different ratings: some users are more inclined to give high ratings, leaving low ratings for really bad items; other users, however, save their highest ratings for the best items and tend to give low ratings. We use the rating preferences to reflect this difference between users. The quality of the item is also an important reference. For example, an item with good quality is unlikely to get a low rating, and it is unusual if a bad item gets high ratings. In this sense, the active user's rating of an item depends on not only the item quality, but is also influenced by the user rating preference.

In real applications, recommender systems usually provide users with public evaluation results of the items. A simple way to measure the public evaluation is the average score the item gets. Another more sophisticated way is to show different ratios of users who rate the item with the rating value of r ($r = 1, 2, 3, 4, 5$). The former method is more common. An item's average score is often used to reflect the its comparative quality. Items with high average score are related to items with good quality, and vice versa. For simplicity, we define the quality of the item to be the average score it gets by:

$$q_i = \frac{\sum_{\alpha \in U_i} v_{\alpha,i}}{|U_i|}. \quad (6)$$

A simple measurement of user rating preference is the average rating score a user gives. The average rating score can reflect the general rating the user gives to items. It is also a main component for rating predictions in UCF. But there remain some problems. A user's ultimate ratings are the comprehensive results of multiple factors, so the average of these ratings cannot exactly describe the true nature of the user. A user who is inclined to give high ratings usually rates items with a score higher than the item's public evaluation rating. And the public evaluation rating of an item is an indicator of item quality. So we can get the preference of a user by subtracting the quality of items from the original ratings. A user's preference can be defined accordingly by:

$$\tau_\alpha = \frac{\sum_{i \in I_\alpha} (v_{\alpha,i} - q_i)}{|I_\alpha|}. \quad (7)$$

It is worth mentioning that since we define q_i to be $\overline{v_{.i}}$ in Eq. (6), the rating preferences of the users are equal to their rating tendencies. From the definition above, we know that user rating preference reflects the mean deviation of user's rating from the item quality score. The value of the rating preference can either be positive or negative. Positive preference indicates the user tends to rate items above their quality score, while negative preference reflects the opposite.

3.2 The proposed algorithm

Our algorithm is based on item quality and user rating preferences (QP). In this algorithm, the predicted rating can be calculated by simply combining the target item's quality score and the active user's rating preference. Once we have predicted all the missing ratings for the active user, the item list can be recommended to the active user by sorting their predicted rating scores.

$$p_{\beta,j} = q_j + \tau_\beta. \quad (8)$$

For each item $j \in I - I_\beta$, we need to go through that item's

Algorithm QP

Input: the whole rating matrix $V = \{v_{\alpha,i}\}$ recording all ratings from users to items;

Output: the rating set $\{p_{\beta,j}|j \in I - I_\beta\}$ needed to be predicted for the active user β.

Steps:

(1) For each $j \in I - I_\beta$, calculate the quality score of item j based on Eq. (6);

(2) Calculate the active user β's rating preference according to formula Eq. (7);

(3) For each $j \in I - I_\beta$, calculate the predicted rating of $p_{\beta,j}$ by Eq. (8).

rated records once to get its quality score, thus complexity is $O(NM)$ for all j; then we combine these quality scores with the active user's rating preference, with complexity of $O(N)$. So the complexity of the QP algorithm for one user is $O(NM)$, and the basic operation is addition. For UCF, it computes the similarity between the active user and other users, thus with complexity of $O(NM)$; then predicts the ratings of user β for all the items, thus with complexity of $O(NM)$. So the complexity of UCF is also $O(NM)$ with the basic operations of multiplication. Similarly, the complexity of ICF is also $O(MN)$ with the multiplication as the basic operation. For TB, the average score of both user and item needs to be computed, as well as their tendencies, thus its complexity is also $O(NM)$ but with twice computation cost of QP. Besides, complicated comparison operations and extra parameter are involved in TB when predicting the ratings. In other words, the proposed algorithm has minimal computational complexity. Table 2 is a concise comparison of all the algorithms' complexities with their basic operations.

Table 2 Algorithm complexity comparison

Algorithms	Complexity	Basic operation
UCF	$O(MN)$	Multiplication
ICF	$O(MN)$	Multiplication
TB	$O(MN)$	Addition
QP	$O(MN)$	Addition

Moreover, we have recorded the CPU time required to generate predictions for all algorithms on three data sets. Figure 1 gives an example of the comparative running time for different algorithms when the ratio of training set to probe set is 9:1. From Fig. 1, we can see that TB and QP have the least running time, but QP runs slightly faster.

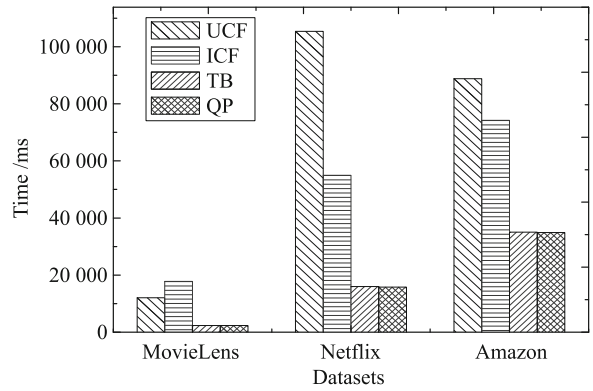


Fig. 1 The running time of the four algorithms on three data sets

4 Experiment

4.1 Experimental design

Here we justify the rationality of our QP algorithm through empirical analysis. Many algorithms mention the four rating features: the average user rating score, the average item score, item score tendency, and the user rating tendency. We have shown the computation of item average score and user rating tendency (Eqs. (6) and (7)). The other two features can be computed correspondingly. However, it is possible that not all of them are necessary for rating prediction.

We first consider the case when all the four features simultaneously participate in predicting the missing ratings. We expect to pick out the essential features from the four. Let $\boldsymbol{Y} = \{v_{\alpha,i}\}_{(\alpha,i)\in E_T}$ be the dependent variable denoting past known rating scores from all users and items. Let $\boldsymbol{X_1} = \{\overline{v_{\alpha\cdot}}\}$, $\boldsymbol{X_2} = \{\overline{v_{\cdot i}}\}$, $\boldsymbol{X_3} = \{\tau_i\}$, and $\boldsymbol{X_4} = \{\tau_\alpha\}$ be the four explanatory variables denoting user α's average rating score, item i's average score, the score tendency of item i, the rating tendency of user α respectively associated with each element $v_{\alpha,i}$ in $\boldsymbol{Y}$. We define a linear regression approach $\boldsymbol{Y} = B_1\boldsymbol{X_1} + B_2\boldsymbol{X_2} + B_3\boldsymbol{X_3} + B_4\boldsymbol{X_4}$ to model the relationship between $\boldsymbol{Y}$ and $\boldsymbol{X} = (\boldsymbol{X_1}, \boldsymbol{X_2}, \boldsymbol{X_3}, \boldsymbol{X_4})$. A least square regression process is then applied to get the four corresponding regression coefficients denoted as $\boldsymbol{B}^* = (B_1, B_2, B_3, B_4)^{\mathrm{T}}$, where

$$\boldsymbol{B}^* = \operatorname{argmin} \|\boldsymbol{Y} - \boldsymbol{X}\boldsymbol{B}\|. \tag{9}$$

Moreover, the value of the regression coefficients can quantify the strength of the relationships between $v_{\alpha,i}$ and the corresponding features. Features associated with small values of coefficients may have less relationship with $v_{\alpha,i}$, while others associated with large coefficients may have strong relationships with $v_{\alpha,i}$. The above regression process minimizes the sum of squared vertical distances between the observed responses and the responses derived by the linear approximation. We expect that applying this liner model can also achieve the minimum distances for the unobserved data. So the predicted value of user α to item i can be formulated as:

$$p_{\alpha,i} = (\overline{v_{\alpha\cdot}}, \overline{v_{\cdot i}}, \tau_i, \tau_\alpha) \times \boldsymbol{B}^*. \tag{10}$$

Meanwhile, we can select the features that are more important based on their associated regression coefficients. We exclude the features whose corresponding regression coefficient is relatively smaller, and the remaining features are expected to be more important.

Then the same regression approach is conducted according to Eq. (9) again, only using the relatively important features. In the same way, the strength of the relationship between $v_{\alpha,i}$ and the remaining features can be qualified. Again, the feature whose regression coefficient is apparently smaller than others is excluded until the coefficient values of all remaining features are close. And the remaining features are determined to be the essential factors for making predictions. Once we get the key features for making predictions, we can predict the unknown ratings only based on these key features. We considered adding an additional constant to the linear regression model. However, the constant introduces disturbances, so we have not adopted it.

Note that QP algorithm comes from the above screening process. In the results section, we will compare QP with two typical CF algorithms and the TB algorithm.

4.2 Datasets and metrics

We use three datasets, namely MovieLens, Netflix, and Amazon, to evaluate algorithms' performance. They are commonly used in previous literatures. The MovieLens[1)] data set consists of 943 users and 1 682 items from its movie recommendation website. Netflix[2)] provides an online DVD and Blu-ray Disc rental service in the US, and the data set we used here is a random sample which consists of 3 000 users and 2 779 movies. Amazon[3)] is a multinational electronic commerce company. The original data were collected from July 28, 2005 to September 27, 2005, and what we use here is also a random sample. Users of the three data sets can rate items with five level ratings from 1 to 5. Let M and N be the number of users and items respectively. Denote $|E|$ as the number of observed ratings. The sparsity of dataset is defined as $|E|/(M \times N)$. Table 3 summarizes the basic statistics.

Table 3 Basic statistics of the three data sets

Datasets	M	N	$\|E\|$	Sparsity
MovieLens	943	1 682	100 000	6.30×10^{-2}
Netflix	3 000	2 779	197 248	2.37×10^{-2}
Amazon	3 604	4 000	134 679	9.34×10^{-3}

To test the algorithm's accuracy, the observed ratings set, $|E|$, is randomly divided into two parts: the training set, E_T, is treated as known information, while the probe set, E_P, is used for testing and no information in this set is allowed to be used for prediction. Clearly, $E_T \bigcup E_P = E$ and $E_T \bigcap E_P = \varnothing$. In the experiment, we randomly select $p\%$ of the ratings as the training set, and the remaining $(100 - p)\%$ constitute the probe set. One can control the data sparsity by tuning p, with larger p corresponding to denser data. We observe the performance of algorithms varying p from 10 to 90 with an interval of 10 to test its robustness. We test five times for each p and report the average performance.

Many metrics have been proposed to evaluate the performances of recommender systems [17]. Two frequently-used measures for quantifying the accuracy of predictions are mean absolute error (MAE) and root-mean-square error (RMSE). We use RMSE [17] to evaluate the accuracy of predictions, where RMSE is defined as

$$\mathrm{RMSE} = \sqrt{\frac{\sum_{(\alpha,i)\in E_P} (v_{\alpha,i} - p_{\alpha,i})^2}{|E_P|}}. \tag{11}$$

4.3 Results

Figure 2 gives an example of the distributions of the four features on the MovieLens dataset. Each subplot represents a feature, plotted by the feature's value range on the horizontal axis and value's distribution probability on the vertical. We can see from the results that the average rating score of a user and the average score of item present a similar distribution, as well as the distribution of item's score tendency and user's rating tendency. The average user rating measures the mean

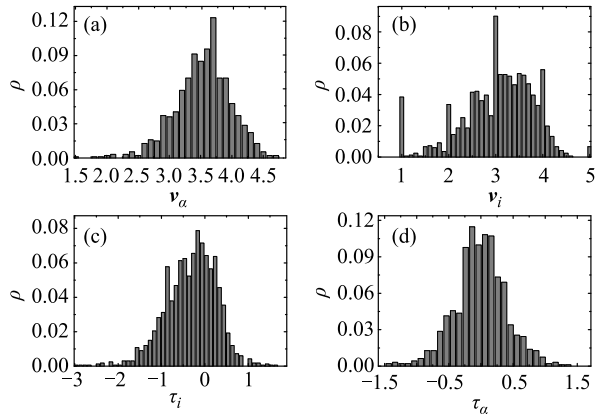


Fig. 2 The distribution of four features on the MovieLens dataset. (a) average user rating; (b) average item score; (c) item rating tendency; (d) user rating tendency

[1)] http:// movielens.umn.edu
[2)] https://signup.netflix.com/global
[3)] http://www.amazon.com

score the user gives to items. The distribution center for average user rating is approximately 3.5 (Fig. 2(a)). Whereas, the average item score measures the mean score an item gets from all users. The distribution center is approximately 3 for average item score (Fig. 2(b)).

Item score tendency reflects the extent that the item score deviates from the average user rating score. Also, user's rating tendency reflects the extent to which the user's rating deviates from the item's average score. Both tendency features can be positive or negative. The distribution difference between the average score of user and item is not great, so as the distribution of the tendency. However, significant difference may exist when they are used to make predictions.

We study the values of regression coefficients when four features are involved in the regression model. As mentioned above, the value of coefficient can reflect the corresponding feature's contribution weight for rating predictions. Figure 3 shows the contributions of the four features when making predictions in three data sets. From the figure, we can infer that the average item score and the user rating tendency seem to have greater contribution for rating predictions than the other two features, because the value of B_2, B_4 are larger than the value of B_1, B_3.

For the sake of brevity, we next exclude user's average rating score and the item's score tendency at the same time. In the same way, we again present the contribution coefficients of the remaining features when only the item's average score and user's rating tendency are involved in the regression model. The case that the features are excluded one by one is also tested, but negligible differences occur. So here, we only present the performance of recommendation with four features and two features involved, omitting the case when three features remain. Figure 4 reveals the contribution coefficients of item average score and user rating tendency. What we can learn is that the two features have almost the same weights, both close to 1.

Let QP* be the recommendation method with the two most important features involved. QP is the regular algorithm we propose in this paper. Figure 5 presents the relative recommendation accuracy of QP* and QP while the recommendation method with four features involved serves as a baseline. The best recommendation accuracy is obtained when all the four features participate in the regression. But the advantage is slight.

As for the recommendation performance of QP* and QP methods, however, slightly differences exist. So we highlight QP algorithm, which indicates that item quality and user rating preferences are the key factors for rating predictions. Moreover, the contribution weights of the two features are close to 1, which to some extent has justified the rationality of QP algorithm. Besides, we can clearly see that when the training set percent is less than 30%, QP performs better than

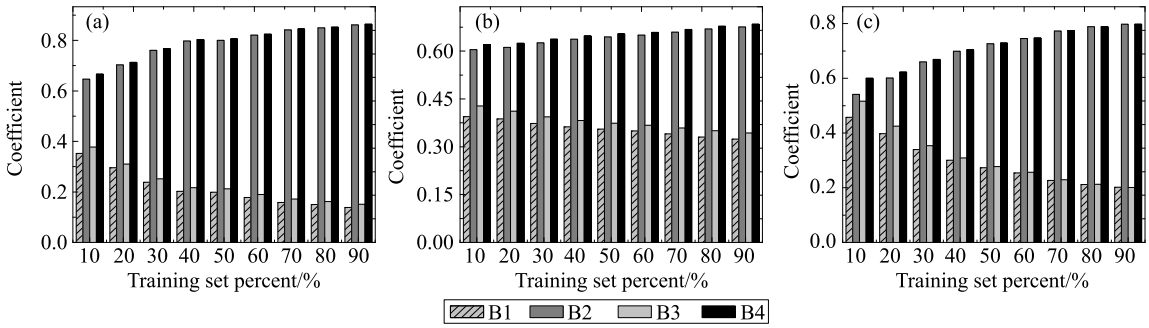


Fig. 3 The values of the regression coefficients with four features involved in the regression. (a) MovieLens, (b) Netflix, (c) Amazon

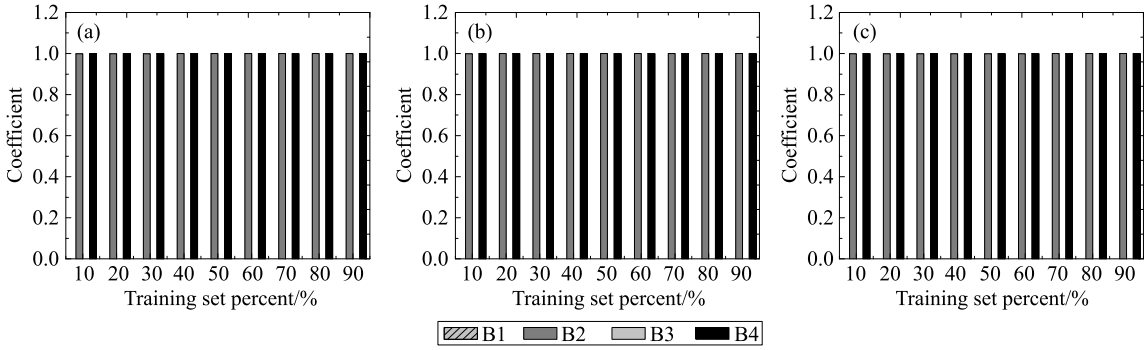


Fig. 4 The value of regression coefficients of the remaining two features. (a) MovieLens, (b) Netflix, (c) Amazon

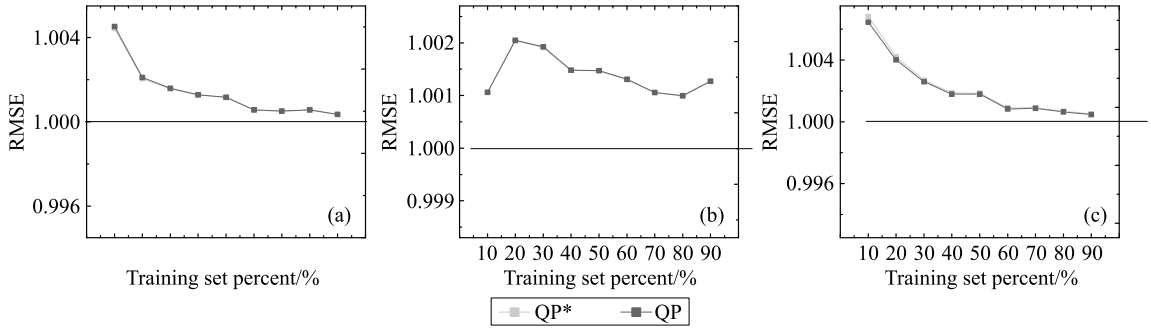


Fig. 5 The relative accuracy performance of QP* and QP compared with the algorithm with four features involved. (a) MovieLens, (b) Netflix, (c) Amazon

QP*. This is reasonable because the results are conducted on the probe set rather than the training set.

Up to now, we have verified that the QP algorithm is effective for rating predictions. Next, we will compare QP with two conventional CF algorithms and TB algorithm. In Fig. 6, the horizontal axis stands for the "training set percent", while the vertical axis represents RMSE. As for TB, we set its tunable parameter to be fixed at 0.5 for all three data sets. From Fig. 6, we can see that QP outperforms almost all the other algorithms on three datasets. More importantly, it performs better when the available data is sparser. Besides, we also put into comparison the algorithm of ItemMean, which simply uses the average score of an item to be the predicted rating. The difference between ItemMean and QP lies in the consideration of user rating preference, the value of which is around 0. But it does have an important influence on rating predictions, because QP performs significantly better than ItemMean.

Table 4 lists the numerical results of the enhancement of QP compared with TB. We can get an average higher recommendation accuracy of 2.99%, 3.18%, and 3.53% respectively on three datasets through all the training percent. On the other hand, if we rethink UCF from the perspective of QP, UCF is actually the addition of user's average rating score

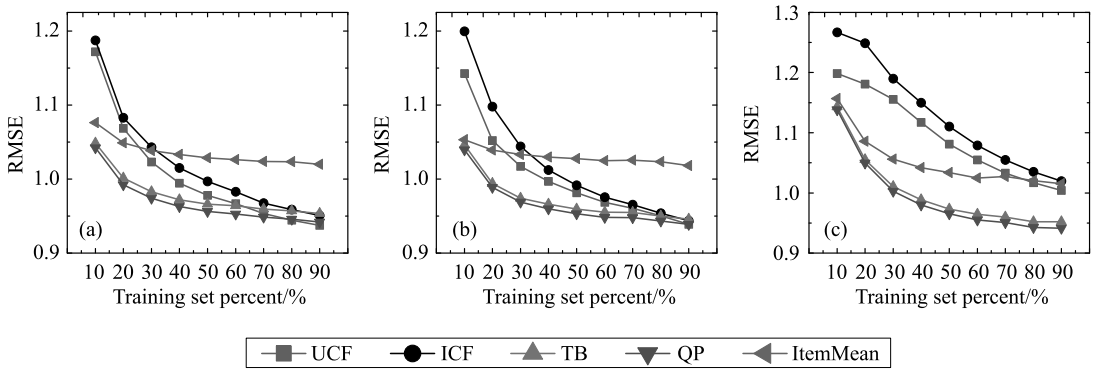


Fig. 6 The accuracy performance of QP compared to UCF, ICF, TB, and ItemMean algorithms. (a) MovieLens, (b) Netflix, (c) Amazon

Table 4 The RMSE enhancement of QP compared with TB in MovieLens, Netflix, and Amazon dataset

Training set percent /%	RMSE								
	MovieLens			Netflix			Amazon		
	TB	QP	Enhancement/%	TB	QP	Enhancement/%	TB	QP	Enhancement/%
10	1.08	1.04	3.25	1.07	1.04	2.84	1.19	1.14	4.46
20	1.02	0.99	3.14	1.02	0.99	3.00	1.09	1.05	3.83
30	1.00	0.97	2.90	1.00	0.97	3.13	1.04	1.00	3.60
40	0.99	0.96	2.97	0.99	0.96	3.27	1.02	0.98	3.47
50	0.99	0.96	2.93	0.99	0.95	3.27	1.00	0.97	3.28
60	0.98	0.95	3.00	0.98	0.95	3.31	0.99	0.96	3.36
70	0.98	0.95	2.94	0.98	0.95	3.30	0.98	0.95	3.27
80	0.97	0.95	2.97	0.97	0.94	3.20	0.97	0.94	3.28
90	0.97	0.94	2.82	0.97	0.94	3.32	0.97	0.94	3.20
Average enhancement/%			2.99			3.18			3.53

with the weighted score tendency of items. Since the related two features are not the key factors for making predictions, it is inevitable that UCF performs not as well as QP.

5 Conclusion

We have presented a simpler yet more efficient recommendation algorithm to explore a scalable solution for recommender systems. We propose QP, an algorithm based on item quality and user rating preference. The experimental results on regression analysis of four features indicate the validity of our approach. Furthermore, comparisons between QP and contemporary algorithms suggest that the QP achieves a considerable improvement in accuracy. In particular, QP increases recommendation accuracy by an average of 3.2% and reduce computational cost by 50% compared with the genetic TB algorithm.

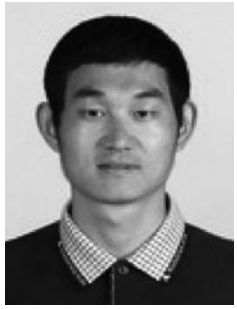
The proposed algorithm can be further improved in several ways. Firstly, other metrics such as diversity and novelty are worth to be considered as performance comparisons. Secondly, item's quality score is currently calculated simply by averaging all its scores, which may be deceptive due to the existence of unusual user rating behavior. We plan to evaluate the quality score of items by building a reputation system for users [18]. Thirdly, the user rating preference also needs to be further improved since it has a significant impact. Other useful user behavioral patterns [19], are also worth exploring. Finally if the data set contains temporal information, it may be better to use the current (rather than historical) quality of item and the current rating preferences of users to make predictions.

Acknowledgements This work was supported by the National Natural Science Foundation of China (Grant Nos. 61370150 and 91324002) and the Sichuan Provincial Science and Technology Department (2012FZ0120). S-MC specially appreciates the Fundamental Research Funds for the Center Universities (WK2100230004, ZYGX2012J075).

References

1. Adomavicius G, Tuzhilin A. Toward the next generation of recommender systems: a survey of the state-of-the-art and possible extensions. IEEE Transactions on Knowledge and Data Engineering, 2005, 17(6): 734–749
2. Ricci F, Rokach L, Shapira B, Kantor P B. Recommender Systems Handbook. Springer, 2010
3. Lü L Y, Medo M, Yeung C H, Zhang Y C, Zhang Z K, Zhou T. Recommender systems. Physics Reports. 2012, 519: 1–49
4. Ekstrand M D, Riedl J, Konstan J A. Collaborative filtering recommender systems. Foundations and Trends in Human-Computer Interaction, 2011, 4(2): 175–243
5. Dror G, Koenigstein N, Koren Y. Web-scale media recommendation systems. Proceedings of the IEEE, 2012, 100(9): 2722–2736
6. Takács G, Pilászy I, Németh B, Tikk D. Scalable collaborative filtering approaches for large recommender systems. Journal of Machine Learning Research, 2009, 10: 626–656
7. Koren Y, Bell R, Volinsky C. Matrix factorization techniques for recommender systems. Computer, 2009, 42(8): 30–37
8. Zhou T, Ren J, Medo M, Zhang Y C. Bipartite network projection and personal recommendation. Physical Review E. 2007, 76(4): 046115
9. Zhang Y C, Blattner M, Yu Y K. Heat conduction process on community networks as a recommendation model. Physical Review Letters, 2007, 99(15): 154301-1–154301-4
10. Zhou T, Kuscsik Z, Liu J G, Medo M, Wakeling J R, Zhang Y C. Solving the apparent diversity-accuracy dilemma of recommender systems. Proceedings of National Academy of Sciences, 2010, 107(10): 4511–4515
11. Huang Z, Chen H, Zeng D. Applying associative retrieval techniques to alleviate the sparsity problem in collaborative filtering. ACM Transactions on Information Systems, 2004, 22(1): 116–142
12. Yang S H, Long B, Smola A, Sadagopan N, Zheng Z, Zha H. Like like alike-joint friendship and interest propagation in social networks. In: Proceedings of the 20th International Conference on World Wide Web. ACM Press. 2011: 537–546
13. Shang M S, Zhang Z K, Zhou T, Zhang Y C. Collaborative filtering with diffusion-based similarity on tripartite graphs. Physica A, 2010, 389(6): 1259–1264
14. Cacheda F, Carneiro V, Fernĺćndez, D, Formoso V. Comparison of collaborative filtering algorithms: Limitations of current techniques and proposals for scalable, high-performance recommender systems. ACM Transactions on the Web. 2011, 5(1), Article 2
15. Resnick P, Iacovou N, Suchak M, Bergstrom P, Riedl J. GroupLens: An Open Architecture for Collaborative Filtering of Netnews. In: Proceedings of the ACM Conference on Computer Supported Cooperative Work. 1994, 175–186
16. Linden G, Smith B, York J. Amazon.com recommendations: item-to-item collaborative filtering. IEEE Internet Computing. 2003, 7(1): 76–80
17. Herlocker J L, Konstan J A, Terveen L G, Riedl J T. Evaluating collaborative filtering recommender systems. ACM Transactions on Information Systems, 2004, 22(1): 5–53
18. Yu Y K, Zhang Y C, Laureti P, Moret L. Decoding information from noisy, redundant, and intentionally distorted sources. Physica A, 2006, 371(2): 732–744
19. Yang Z M, Zhang Z K, Zhou T. Anchoring bias in online voting. Europhysics Letters. 2012, 100(6): 68002-1–68002-6

Yuan Guan received her BS in Computer Science and Engineering from University of Electronic Science and Technology of China in 2011, and is now pursuing her MS in the Web Sciences Center, University of Electronic Science and Technology of China, China. Her research interests include data mining and recommender systems.



Shimin Cai received his BS in Electrical Engineering from Hefei University of Technology in 2004 and his PhD in Circuit and Systems from the University of Science and Technology of China in 2009. He currently serves as an associate professor of the University of Electronic Science and Technology of China. He is interested in complex network theory and its application for mining and modeling of real large-scale networked systems, time series analysis, and personalized recommendation systems.

Mingsheng Shang received his PhD in Computer Science from the University of Electronic Science and Technology of China. He is a professor of UESTC. His research interests include data mining, complex networks, and cloud computing and their applications.