Article

JIS

Journal of Information Science
1–13

DOI: 10.1177/0165551516654354
jis.sagepub.com
SAGE

Collaborative filtering using non-negative matrix factorisation

Mehdi Hosseinzadeh Aghdam
Payame Noor University, Iran
Iran University of Science and Technology, Iran

Morteza Analoui
Iran University of Science and Technology, Iran

Peyman Kabiri
Iran University of Science and Technology, Iran

Abstract

Collaborative filtering is a popular strategy in recommender systems area. This approach gathers users' ratings and then predicts what users will rate based on their similarity to other users. However, most of the collaborative filtering methods have faced problems such as sparseness and scalability. This paper presents a non-negative matrix factorisation method to alleviate these problems via decomposing rating matrix into user matrix and item matrix. This method tries to find two non-negative user matrix and item matrix whose product can well estimate the rating matrix. This approach proposes updated rules to learn the latent factors for factorising the rating matrix. The proposed method can estimate all the unknown ratings and its computational complexity is very low. Empirical studies on benchmark datasets show that the proposed method is more tolerant of the sparseness and scalability problems.

Keywords

collaborative filtering; latent factors; matrix factorisation; model-based recommender systems

1. Introduction

Recently, recommendation applications have dramatically increased to detect interesting items for users. This approach helps users to find their items among several other items. These kinds of recommender systems (RSs) have been used at large online stores. They also have created interest in the research communities in information systems, marketing and computer science [1]. Data are about the items to recommend and the users who will receive recommendations. The recommendation challenge truly arose as an independent field of research in the mid-1990s. It has roots in several fields such as information retrieval and cognitive science. Strategies for this problem are normally considered into two categories: content-based and collaborative filtering (CF) methods [2].

The content-based method detects the features of items that have received a favourable rating from a user and propose to the user new items that share these features. These methods generally suffer from the problems of limited content and over-specialisation [2]. Limitation in content comes from the fact that the recommender systems may have only limited information on their users. Unlike the content-based method, the CF method relies on the ratings of users [2]. CF approaches overcome some of the problems of the content-based method. For example, items for which the content is not available can still be recommended to users through the ratings of other users. Also, CF recommendations are based on the quality of items as rated by similar users instead of depending on content. These methods can recommend items

Corresponding author:
Mehdi Hosseinzadeh Aghdam, School of Computer Engineering, Iran University of Science and Technology, Tehran, 16846-13114, Iran.
Email: mhaghdam@iust.ac.ir

with very different content, as long as neighbours have already shown a preference for these different items. Research on CF can be grouped into two categories: instance-based methods and model-based methods [3].

Instance-based methods collect the ratings in the system and use them directly to estimate ratings for new items. Instance-based methods can be grouped into two categories: user-based or item-based recommendation. Instance-based approaches compute similarities between users or between items and use them to recognise the top-most similar neighbours. Then, the unknown rating is predicted using the combination of known rating of the neighbours. However, there exist problems in terms of sparseness and scalability. When the rating matrix is sparse, users or items are unlikely to have common ratings and, consequently, instance-based approaches will estimate unknown ratings using a small number of neighbours. Moreover, similarities between users or between items may be calculated using only a small number of ratings, resulting in biased predictions. This is worsened when users or items newly added to the system may have no ratings at all. This problem is called the cold start problem. Cold start is a subproblem of coverage because it evaluates the system coverage over a specific set of users and items [2].

Unlike instance-based methods, model-based methods use these ratings to learn a predictive model. The main idea is to model the user-item scores with features showing latent factors of the users and items in the system. Then the model is trained using the available scores and later used to predict unknown ratings [4–7]. However, generating a model and keeping it up-to-date are often time-consuming because there are usually many free parameters to tune [8].

Matrix factorisation (MF) is one of the most successful types of approach to collaborative filtering [9–11]. MF assumes that there are a small number of latent features that can describe preferences of users; users and items are shown as unknown feature vectors whose dimensions are assumed as latent factors. These feature vectors are achieved from the known scores and then predict unknown scores for users using the inner products of the corresponding feature vector pairs. Various approaches differ in the approximation criteria or the cost function they apply, and variants may be derived by appending different kinds of regularisation to avoid over-fitting [12]. Singular value decomposition (SVD) is a well-known approach which is based on MF [13]. More recently, several MF approaches have been successfully applied to implementing RSs, including the maximum margin MF [14], the probabilistic latent semantic analysis [15] and the expectation maximisation for MF [16]. During the Netflix prize, regularised MF is reported which is accurate, more efficient and easy to implement. Inspired by regularised MF, many papers have further studied MF for RSs. They have reported sophisticated MF for RSs [2]. We use non-negative matrix factorisation (NMF) to propose a method to learn the latent factors of users and items, and use them to create personalised recommendations. The ability to capture latent factors of users, representing attributes of users, enables our approach to generate more diverse recommendations. We evaluated our approach using an online cross-validation strategy. In our experiments, we used MovieLens datasets and Book-Crossing datasets. Our evaluation results indicate that the NMF achieves better performance compare to other baseline methods.

The rest of this work is organised as follows. Section 2 explains NMF for CF. The time complexity of the proposed method is described in section 3. Section 4 explains the recommendation algorithms. Section 5 reports computational experiments. It also includes a brief discussion of the recommendation algorithms and the results. Finally, we conclude the paper in the last section.

2. Non-negative matrix factorization for collaborative filtering

NMF has been studied by many researchers, but it has achieved more attention due to the research by Lee and Seung [17]. For the argument that non-negativity is critical in human perception, they presented simple methods for finding non-negative representations of non-negative data. The NMF method decomposes a matrix $R(R \geq 0)$ to find two non-negative user matrix (U) and item matrix (I), that is $R \approx UI^T$.

RSs contain a set of users and a set of items. The rating matrix $R_{n \times m}$ shows the scores that expressed by users on items. The entry R_{ui} indicates the score of user u on item i. Scores can be any real number, but often they are integers. The task of a RS is to predict the unknown ratings.

This paper employs NMF to learn latent features of users and items and predicts the unknown scores using these latent factors. Let $U_{n \times k}$ and $I_{m \times k}$ be user and item latent factor matrices, with row vectors U_u and I_i representing k-dimensional user and item latent feature vectors of user u and item i, respectively. The proposed approach tries to learn these latent factors and exploit them for the prediction. The user matrix can be considered as containing a basis that is optimised for the linear estimation of the rating matrix. Since few basis vectors are used to represent many vectors, accepted prediction can only be achieved if the basis vectors discover a structure that is latent in the rating matrix. The non-negative constraints on user matrix and item matrix only allow additive combinations among different bases. This is the main difference between NMF and the other MF approaches [17].

In order to estimate user matrix and item matrix, the cost function is required to quantify a difference between the original rating matrix and the estimated rating matrix, $\hat{R} = UI^T$. The choice of the cost function mostly depends on the probability distribution of data. The simple way is to use Frobenius-norm and Kullback–Leibler measures:

$$D_F(R||UI^T) = \frac{1}{2}\left\|R - UI^T\right\|_F^2 = \sum\nolimits_{u,\,i}\left(R_{ui} - \sum\nolimits_{f=1}^{k} U_{uf} I_{fi}^T\right)^2 \tag{1}$$

$$D_{KL}(R||UI^T) = \sum\nolimits_{u,\,i}\left(R_{ui}\ln\frac{R_{ui}}{\left[UI^T\right]_{ui}} - R_{ui} + \left[UI^T\right]_{ui}\right) \tag{2}$$

These are lower bounded by zero and disappear if, and only if, $R = UI^T$. KL-divergence cannot be named a distance because it is not symmetric in R and UI^T. Therefore, it will be referred as the divergence of R from UI^T. The objective of these cost functions is the minimisation of the difference between original rating matrix and estimated rating matrix with respect to user matrix and item matrix. So, we can use both Eq. (1) and Eq. (2). In this paper, the combination of these cost functions is used for the proposed approach:

$$costfunction = \omega D_F(R||UI^T) + \varphi D_{KL}(R||UI^T) \tag{3}$$

where ω and φ are two parameters that control the relative weight of Frobenius-norm and divergence, $\omega \in [0,1]$ and $\varphi = 1 - \omega$. In our experiment, they were set as $\omega = 0.5$, $\varphi = 0.5$.

This cost function is convex with respect to either user matrix or item matrix, but not both. So, it is infeasible to find the global minimum for this problem. However, there are many approaches from numerical optimisation that can be applied to find a local minimum. Gradient descent is the simplest to implement, but convergence can be slow. Other approaches like conjugate gradients have faster convergence, at least in the neighbourhood of local minimum, but are more complex to implement than gradient descent. The convergence of gradient-based approaches also has the disadvantage of being very sensitive to the choice of step size [17].

The initialisation step has more effect on the performance of MF approaches. Therefore, it is important to have efficient strategies for initialising user matrix and item matrix. Poor initialisation often yields low convergence, and in certain cases may even lead to an irrelevant or incorrect solution. The problem of initialisation becomes even more difficult for large MF problems and when certain constraints are applied on the factored matrices involved [18, 19]. The proposed approach uses the similarity weights for initialising user matrix and item matrix.

For factorising the rating matrix, we need updated rules to train user matrix and item matrix. In this paper, the updated rules are derived from cost function by using gradient descent. The updated rules for user matrix and item matrix can be written as:

$$\begin{aligned} U_{uf} &\leftarrow U_{uf} + \omega \times \tau_{uf}[(RI)_{uf} - (UI^TI)_{uf}] \\ &+ \phi \times \eta_{uf}[\sum\nolimits_{\mu=1}^{m} I_{f\mu}^T \frac{R_{u\mu}}{(UI^T)_{u\mu}} - \sum\nolimits_{\mu=1}^{m} I_{f\mu}^T] \end{aligned} \tag{4}$$

$$\begin{aligned} I_{fi}^T &\leftarrow I_{fi}^T + \omega \times \tau_{fi}[(U^TR)_{fi} - (U^TUI^T)_{fi}] \\ &+ \varphi \times \eta_{fi}[\sum\nolimits_{\mu=1}^{n} U_{\mu f} \frac{R_{\mu i}}{(UI^T)_{\mu i}} - \sum\nolimits_{\mu=1}^{n} U_{\mu f}] \end{aligned} \tag{5}$$

If τ and η are all set to an equal small positive number, these update rules should reduce cost function. Now if we set:

$$\tau_{uf} = \frac{U_{uf}}{(UI^TI)_{uf}} \tag{6}$$

$$\tau_{fi} = \frac{I_{fi}^T}{(U^TUI^T)_{fi}} \tag{7}$$

$$\eta_{uf} = \frac{U_{uf}}{\sum_{\mu=1}^{m} I_{f\mu}^T} \tag{8}$$

$$\eta_{fi} = \frac{I_{fi}^T}{\sum_{\mu=1}^{n} U_{\mu f}} \tag{9}$$

Non-negative matrix factorisation algorithm for collaborative filtering

Input: original rating matrix $R_{n \times m}$
Output: estimated rating matrix $\hat{R}_{n \times m}$

Initialising:
Generate two non-negative matrices $U_{n \times k}$ and $I_{m \times k}$ using similarity between users and items, respectively.

Training:

for each *iteration*=1,…,*max_iteration* **do**
  for each user *u*=1,…,*n* **do**
    for each feature *f*=1,…,*k* **do**
      update U_{uf} using equation (10)
    end
  end
  for each item *i*=1,…,*m* **do**
    for each feature *f*=1,…,*k* **do**
      update I_{fi}^T using equation (11)
    end
  end
  $\hat{R} = UI^T$
  if $\hat{R} = R$ **then**
    return $\hat{R}$
  end
end for
return $\hat{R}$

Then we obtain the update rules for user matrix and item matrix. The following updated rules are a good agreement between speed and ease of implementation to estimate user and item factor matrices. The cost function is non-increasing under these update rules.

$$U_{uf} \leftarrow U_{uf} \times [\omega \frac{(RI)_{uf}}{(UI^T I)_{uf}} + \varphi \frac{\sum_{\mu=1}^{m} I_{f\mu}^T R_{u\mu} / (UI^T)_{u\mu}}{\sum_{\mu=1}^{m} I_{f\mu}^T}] \tag{10}$$

$$I_{fi}^T \leftarrow I_{fi}^T \times [\omega \frac{(U^T R)_{fi}}{(U^T UI^T)_{fi}} + \varphi \frac{\sum_{\mu=1}^{n} U_{\mu f} R_{\mu i} / (UI^T)_{\mu i}}{\sum_{\mu=1}^{n} U_{\mu f}}] \tag{11}$$

At each iteration of the proposed method, the new values of user matrix and item matrix are found by multiplying the current values by some features that depend on the quality of the estimations. The quality of the estimated scores increases monotonically with the application of these multiplicative update rules. In other words, repeated iteration of these rules is guaranteed to converge to a locally optimal MF. Figure 1 illustrates this process by providing a simple example.

3. Scalability and complexity analysis

We assume the average number of items that are rated by two specific users is $\bar{p}$ and the average number of users that rated two specific items is $\bar{h}$. The time complexity of the proposed approach consists of two parts: initialisation and training time complexity. In the initialisation step, the complexity of computing the similarity matrices is $O\left(n^2\bar{p} + m^2\bar{h}\right)$, and the complexity of equalisation of the feature vectors is $O(n^2 + m^2)$. In the training step, the most significant complexity is in updating the feature vectors. The complexity of learning each user feature vector and each item feature vector are $O(km)$ and $O(kn)$, respectively. So, the complexity of the training step is $O(iteration \times (nkm + mkn))$. We can rewrite this as $O(iteration \times nkm)$, which denotes that the computational complexity of the proposed NMF is linear with respect to either the number of users or items. This complexity analysis shows that the proposed method is efficient and can scale to very large datasets. The architecture of the proposed NMF is shown in Figure 2.

In order to maintain high prediction accuracy, the recommender systems must be kept up-to-date. The drawback of matrix factorization methods is that once the matrices are calculated, the model is static. For real world problems, updating a model is important. Particularly, when ratings on new users or new items come in, updating the feature vectors is

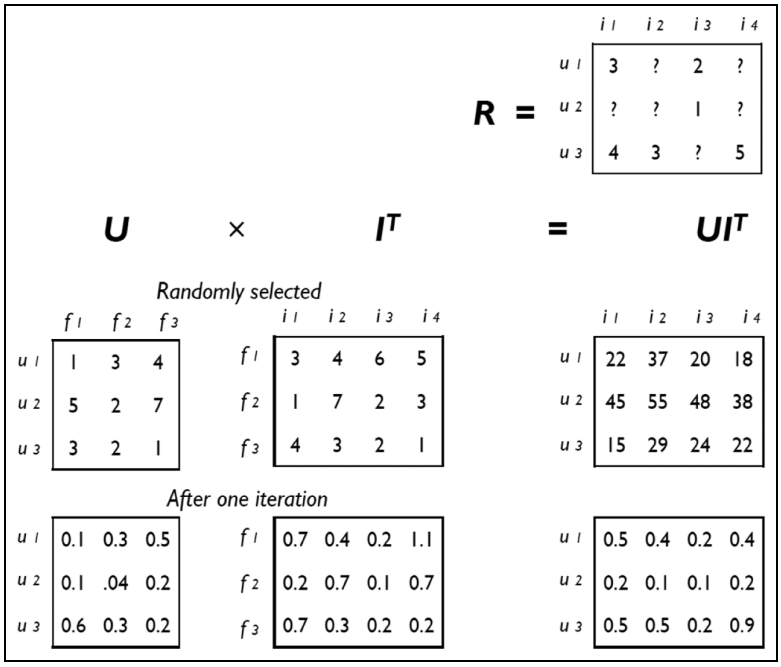

Figure 1. Example for the proposed NMF.

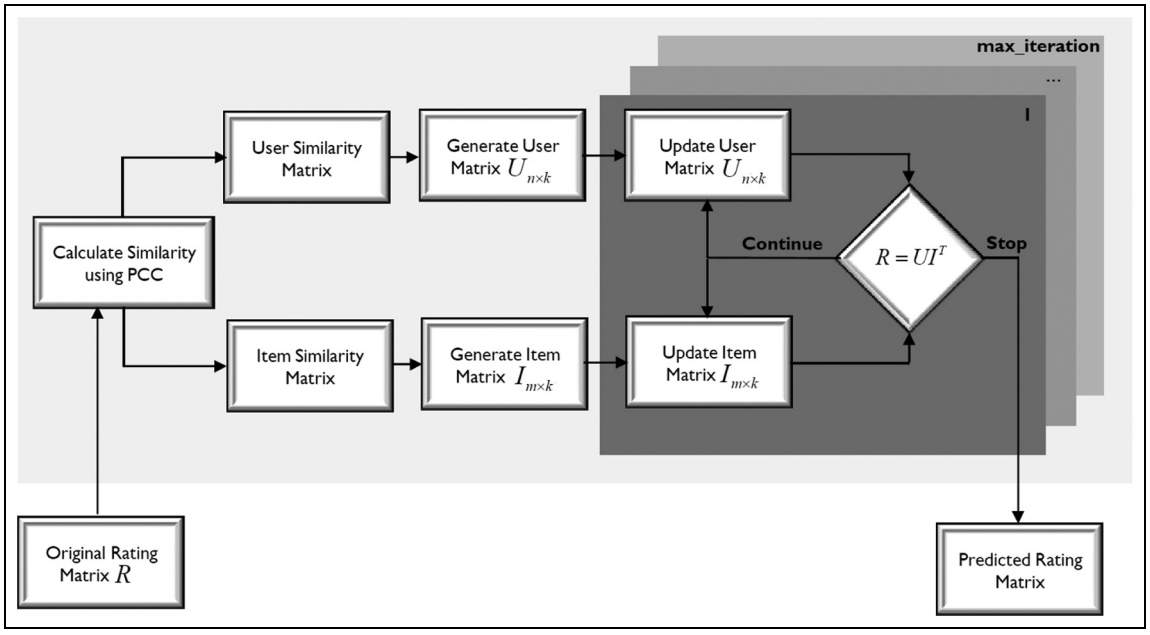

Figure 2. The proposed NMF architecture.

important. When new rate come in, the proposed method updates the only feature vectors that depend on. This operation has very low complexity $O(kn + km)$. On the other hand, both empirical and theoretical results for runtime complexity of the proposed NMF make it feasible for huge datasets.

4. Recommendation algorithms

In this paper, we tested the five popular recommendation methods, including simple averaging methods, user mean and item mean, variations of neighbourhood-based CF methods and the SVD method.

4.1. User mean and item mean

Simple average methods predict an unknown rating R_{ui} with an average rating provided by user u (user mean) or with an average rating received by item i (item mean) [2]. More specifically, for user mean simple average, the system estimates all unknown ratings with the average rating of each user, $\hat{R}_{ui} = \bar{R}_u$. Where $\hat{R}_{ui}$ is the estimated rating for current user u and item i. $\bar{R}_u$ denotes the average rating of user u. For item mean, the system estimates all unknown ratings with average ratings of each item, $\hat{R}_{ui} = \bar{R}_i$. Where $\bar{R}_i$ denotes the average rating received by item i.

4.2. User-based collaborative filtering

User-based CF operates by finding most similar users to the target user and formulates a prediction by combing the preference of similar users. The predicted rating of the target user for the item is a weighted sum of similar users' ratings on this item. One of the popular similarity measures that is used in recommendation algorithms is Pearson's correlation coefficient (PCC) in which the similarity weight of users u and v is calculated as follows [19]:

$$PCC(u, v) = \frac{\sum_{i \in P} (R_{ui} - \bar{R}_u)(R_{vi} - \bar{R}_v)}{\sqrt{\sum_{i \in P} (R_{ui} - \bar{R}_u)^2} \sqrt{\sum_{i \in P} (R_{vi} - \bar{R}_v)^2}} \tag{12}$$

where P denotes the set of items that are rated by both users u and v. R_{vi} is the ratings by other users v for item i. CF methods can predict a rating of the item for target user using a combination of the ratings of the similar users that are already familiar with item [19]. The classical CF formula is:

$$\hat{R}_{ui} = \bar{R}_u + \frac{\sum_v PCC(u, v)(R_{vi} - \bar{R}_v)}{\sum_v PCC(u, v)} \tag{13}$$

4.3. Item-based collaborative filtering

Similar to user-based CF, item-based CF computes the similarity between two items by comparing ratings made by the same users on these items [2]. The similarity is the PCC between two items and is calculated as:

$$PCC(i, j) = \frac{\sum_{u \in H} (R_{ui} - \bar{R}_u)(R_{uj} - \bar{R}_u)}{\sqrt{\sum_{u \in H} (R_{ui} - \bar{R}_u)^2} \sqrt{\sum_{u \in H} (R_{uj} - \bar{R}_u)^2}} \tag{14}$$

where H is the set of users that rated both items i and j. After computing the similarity between items, we can generate the predicted score using the formula below:

$$\hat{R}_{ui} = \bar{R}_u + \frac{\sum_j PCC(i, j)(R_{uj} - \bar{R}_u)}{\sum_j PCC(i, j)} \tag{15}$$

4.4. Singular value decomposition (SVD)

SVD is one of the most successful realisations of latent factor methods in RSs, as evidenced by the solutions to the recent Netflix prize competition [20]. MF attempts to map both users and items to a joint latent vector space, such that user-item scores are modelled as inner products in that space. Estimations on unknown ratings are then made by matching corresponding item factors with user factors [21]. More specifically, each user u is associated with a user vector U_u and each item i is associated with an item vector I_i. The prediction is done by taking the inner product of user vector and item vector. To learn the factor vectors, the system minimises the regularised squared error on the set of known ratings:

$$\min_{U, I} \sum_{(u, i) \in R} (R_{ui} - U_u I_i^T)^2 + \lambda(\|U_u\|^2 + \|I_i\|^2) \tag{16}$$

The constant λ controls the extent of regularisation of learned factors in order to avoid over-fitting the observed data.

Table 1. Statistics of datasets.

	MovieLens 100K	MovieLens 1M	Book-Crossing
Users	943	6040	3156
Items	1682	3952	7485
Ratings	100,000	1,000,209	253,902
Density	6.30%	4.19%	1.07%

5. Experiments

5.1. Dataset description

There are several kinds of datasets for recommender systems. This paper has used MovieLens and Book-Crossing datasets to evaluate the performance of the NMF method. GroupLens research was collected and made available rating datasets from the MovieLens website. MovieLens consists of three datasets: 100K, 1M, and 10M. We used 100K and 1M in our evaluation. The Book-Crossing dataset was collected by Cai-Nicolas Ziegler in a 4-week crawl from the Book-Crossing community. The Book-Crossing dataset is extremely sparse. We edit the dataset in order to achieve more meaningful results from CF methods when estimating recommendations. Hence, the book titles with fewer than 20 ratings are deleted, along with all ratings referring to them. Only users with at least 20 ratings were kept. The resulting dataset's dimensions were greatly moderate. The edited dataset consists of book ratings on an integer scale from 0 to 10. Their statistics are listed in Table 1.

5.2. Evaluation metrics

There are several types of metrics for evaluating the performance of recommendation algorithms [22].

5.2.1. Mean absolute error (MAE).

$$MAE = \frac{1}{|R_{test}|} \sum_{u,i} |R_{ui} - \hat{R}_{ui}| \tag{17}$$

Where $\hat{R}_{ui}$ denotes the rating user u gave to item i as predicted by a method, and R_{test} denotes the tested ratings.

5.2.2. Root mean square error (RMSE).

$$RMSE = \sqrt{\frac{1}{|R_{test}|} \sum_{u,i} (R_{ui} - \hat{R}_{ui})^2} \tag{18}$$

From the definition, a smaller MAE or RMSE value means a better accuracy [2].

5.2.3. Coverage. The MAE and RMSE do not fully express the usefulness of the RSs. We also need to consider the effectiveness of the RS by calculating the total coverage of the system. Coverage is the metric of the percentage of items for which a recommendation algorithm can provide recommendations [23]. A RS may not be able to create recommendations on every item. For instance, if an item has been rated by very few users, or if a user has rated very few items. A recommendation algorithm which has high prediction accuracy, but only on a small set of items, would not be very useful. Calculating coverage will give further insight into the effectiveness of the RS. We compute coverage as a number of items for which the RS can create recommendations, over the total number of item predictions that are requested.

$$\text{coverage} = \frac{|\mathrm{R_i}|R_i \in R_{test}|}{|R_{test}|} \tag{19}$$

Where R_i denotes the recommendation that the recommendation algorithm created on item i.

5.2.4. Stability. Stability of recommendation algorithms measures the consistency of RS predictions. Stability has a positive effect on the users' view to accept recommendations. In this paper, stability is measured using the following steps that are already reported by Adomavicius and Zhang [24]:

(1) Train the recommendation algorithms based on the known ratings and predict all the unknown ratings.
(2) Select and add a subset of the predicted ratings as the new incoming ratings to the original dataset.
(3) Re-train the recommendation algorithms based on the new data and make new predictions for unknown ratings.
(4) Compare predictions from steps (1) and (3) and calculate stability by using following measures:
    - Mean absolute shift (MAS)

$$MAS = \frac{1}{R_{test2}} \sum_{(u,\, i) \in R_{test2}} |R_{ui1} - R_{ui2}| \tag{20}$$

Where R_{ui1} and R_{ui2} denote the ratings that user u gave to item i as predicted by a method in steps (1) and (3), respectively. R_{test2} denotes the number of tested ratings in step (3).

- Root mean squared shift (RMSS)

$$RMSS = \sqrt{\frac{1}{R_{test2}} \sum_{(u,\, i) \in R_{test2}} (R_{ui1} - R_{ui2})^2} \tag{21}$$

5.3. Experimental results

This section focuses on comparing the performance of the proposed method and other methods. The tested methods included the proposed NMF method and other baseline methods. All tested methods were implemented on MATLAB R2011b. All experiments were conducted on a Microsoft Windows computer with two 2.67 GHz dual-core CPUs and 4 GB RAM.

In this work, we used the five fold cross-validation in experiments. Each fold contains 80% of the dataset as the training set and the remaining as the test set. MAE and RMSE are shown in Table 2. We can see that on average, NMF obtained a higher accuracy value than the other algorithms. If the standard deviation is high, the user mean and item mean have poor accuracy. The standard deviation of rating on the MovieLens datasets is higher than 1 for each user and each item. Likewise, the standard deviation of ratings on the Book-Crossing dataset is higher than 3.

As mentioned earlier, in the MovieLens and Book-Crossing datasets, each user has rated at least 20 items. As a result, the user mean approach has 100% coverage, but it is not true in all datasets with the cold start problem. On the other hand, MF methods have 100% coverage in all datasets. Table 3 shows the coverage of recommendation algorithms for all datasets.

For conveniently comparing with other methods reported in [8, 25], for MovieLens 100K, we also extracted a subset of 500 users with more than 40 ratings. The first 100, 200 and 300 users in the dataset are considered into three different training user sets, which are indicated as ML-100, ML-200 and ML-300, respectively. But for different training sizes, the test user set is fixed, i.e. the last 200 users. The available ratings of each test user are equally split into an observed

Table 2. The accuracy of recommendation algorithms.

	MovieLens 100K		MovieLens 1M		Book-Crossing	
	MAE	RMSE	MAE	RMSE	MAE	RMSE
User mean	0.8362	1.0437	0.8385	1.0361	5.7227	6.6088
Item mean	0.8188	1.0278	0.7810	0.9856	5.7788	6.5464
CF user	0.7047	0.9896	0.7312	0.9263	5.6654	6.5679
CF item	0.7010	0.9847	0.7025	0.9052	5.7260	6.5740
SVD	0.7366	0.9654	0.6873	0.8683	3.2512	5.2312
Proposed NMF	0.6941	0.9563	0.6423	0.7823	2.2513	4.1931

Table 3. The coverage of recommendation algorithms.

	MovieLens 100K	MovieLens 1M	Book-Crossing
User mean	100%	100%	100%
Item mean	97%	96%	99%
CF user	97%	98%	73%
CF item	98%	97%	75%
SVD	100%	100%	100%
Proposed NMF	100%	100%	100%

Table 4. Comparing results from the proposed NMF against the results listed in [8, 25].

	ML-100			ML-200			ML-300		
	Given5	Given10	Given20	Given5	Given10	Given20	Given5	Given10	Given20
PCC	0.874	0.836	0.818	0.859	0.829	0.813	0.849	0.841	0.820
PD	0.849	0.817	0.808	0.836	0.815	0.792	0.827	0.815	0.789
AM	0.963	0.922	0.887	0.849	0.837	0.815	0.820	0.822	0.796
MMMF	0.945	0.861	0.803	0.930	0.849	0.786	0.929	0.843	0.773
SCBPCC	0.848	0.819	0.789	0.831	0.813	0.784	0.822	0.810	0.778
SF2	0.847	0.774	0.791	0.827	0.773	0.783	0.804	0.761	0.769
CFONMTF	0.838	0.801	0.804	0.827	0.791	0.787	0.801	0.780	0.782
Proposed NMF	0.8012	0.7856	0.7642	0.7732	0.7528	0.7319	0.7605	0.7424	0.7253

set and a held-out set. The observed ratings are used to predict the held-out ratings. Furthermore, we randomly chose 5, 10 and 20 items rated by test users in the observed set, which were termed Given5, Given10 and Given20, respectively.

In order to show the performance of the proposed NMF, we compare it versus the state-of-art methods: PCC [26], personality diagnosis (PD) [27], aspect model (AM) [7], scalable cluster-based Pearson correlation coefficient (SCBPCC) [8], similarity fusion (SF2) [28] and CF using orthogonal non-negative matrix Tri-factorization (CFONMTF) [25]. A comparison between the results for the proposed NMF versus other methods tested on the MovieLens 100K is presented in Table 4. It can be stated that all the methods tested on this dataset offered an acceptable level of performance, but the proposed method outperforms all the other methods and is just a little worse than similarity fusion in Given10. Similarity fusion suffers from the scalability problem while the proposed NMF resolves this problem.

5.3.1. Impact of data sparseness. The graphs in Figures 3–6 display the comparisons of the accuracy of recommendation algorithms with different levels of data density on MovieLens datasets. The results are consistent with the prior studies in that the recommendation methods typically demonstrate higher predictive accuracy on denser rating datasets.

The sparseness of a rating matrix can have a great effect on the performance of recommendation algorithms. To evaluate the accuracy of the proposed NMF, this paper conducted an experiment to simulate the sparseness of rating matrix and compare the accuracy of five methods: user mean, item mean, CF user, CF item and SVD. This paper empirically analyses how MAE and RMSE evolve with the density of rating matrix.

Figures 3 and 4 show the accuracy of NMF when the sparseness of MovieLens 100K varies. We randomly selected different percentages of the known ratings from the whole dataset to represent different degrees of the sparseness of the rating matrix. Also, Figures 5 and 6 present this setting for MovieLens 1M. The results indicate that the sparseness has an important impact on the accuracy of recommendation algorithms. When the rating matrix becomes dense, all algorithms tend to obtain higher accuracy. As seen in Figures 3–6, the MAE and RMSE curves of NMF are below that of the other algorithms, which means that the sparseness has the least impact on the NMF.

In order to compare the NMF with other algorithms in terms of stability, this paper has considered five algorithms which are mentioned earlier on the MovieLens datasets. The total number of ratings is 100K in MovieLens 100K. For initial stability calculations, we used 80K ratings as the input to predict the remaining 20K unknown ratings. From this predicted rating matrix, we drew a random sample of 10K ratings and treated them as new incoming ratings and the

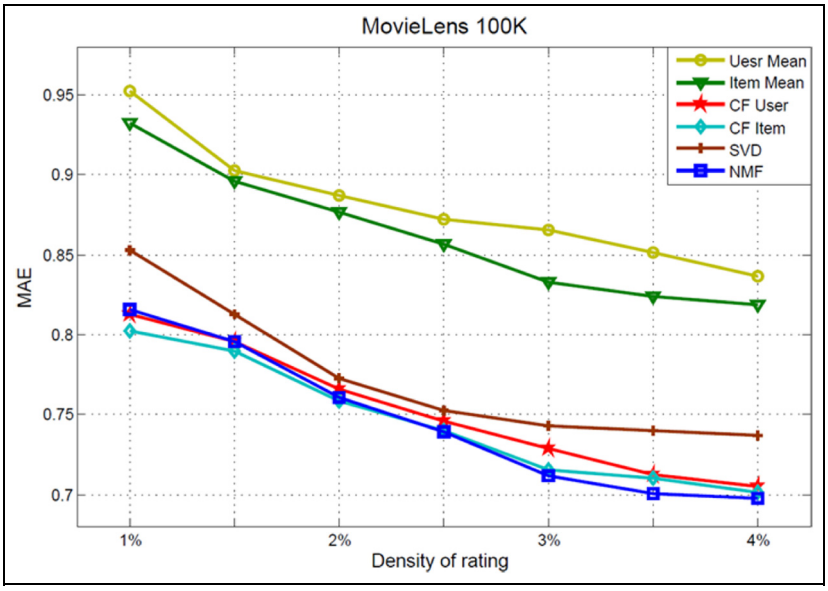

Figure 3. Comparison of MAE for different data density levels on MovieLens 100K.

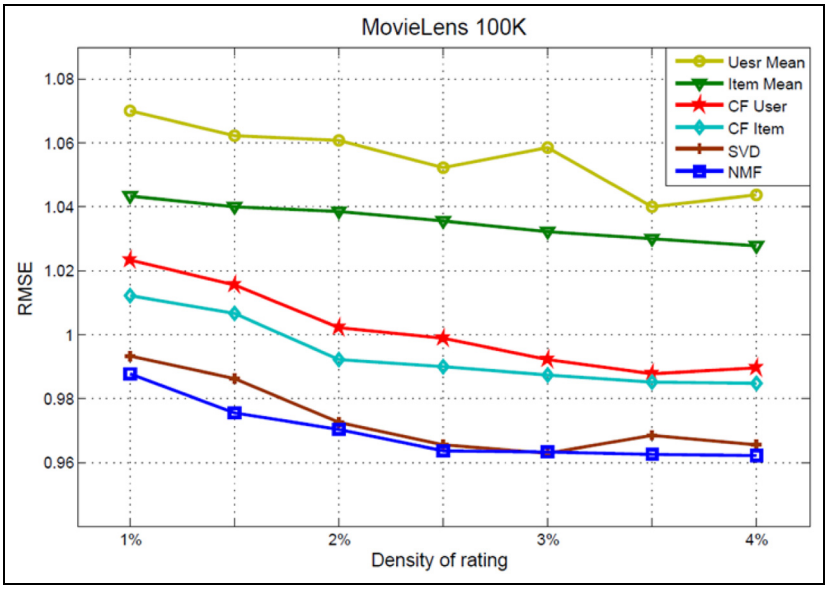

Figure 4. Comparison of RMSE for different data density levels on MovieLens 100K.

remaining ratings were used to compute the prediction shift. In order to obtain robust empirical results, we ran the experiments three times for each algorithm and reported the average stability of the three runs in Table 5. We used the same approach with MovieLens 1M.

Both user-mean and item-mean methods had lower prediction shifts than other algorithms. In other words, adding new ratings does not significantly change the user/item average. Furthermore, while the NMF, SVD, CF user and CF item were comparable in term of predictive accuracy, NMF, which is based on the global optimisation, represented higher stability than the SVD, CF user and CF item that are based on the local, nearest neighbour heuristics.

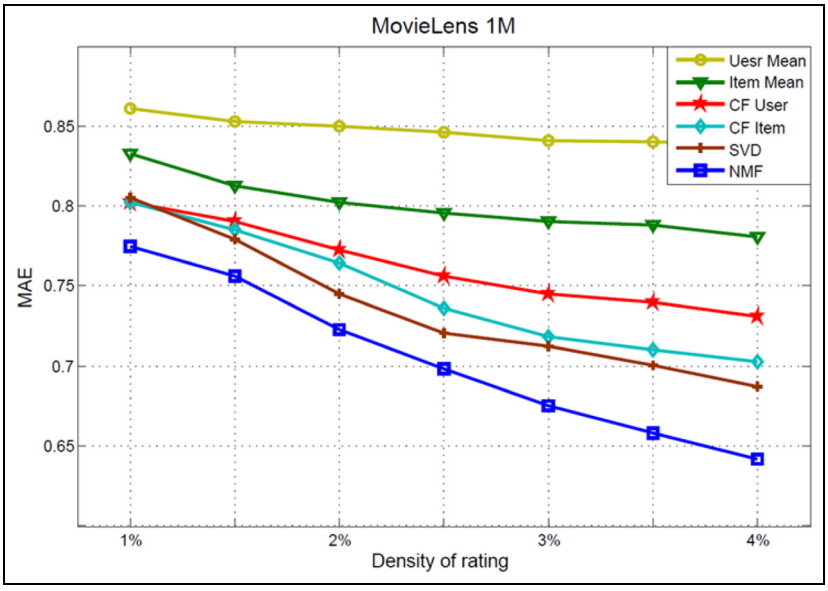

Figure 5. Comparison of MAE for different data density levels on MovieLens 1M.

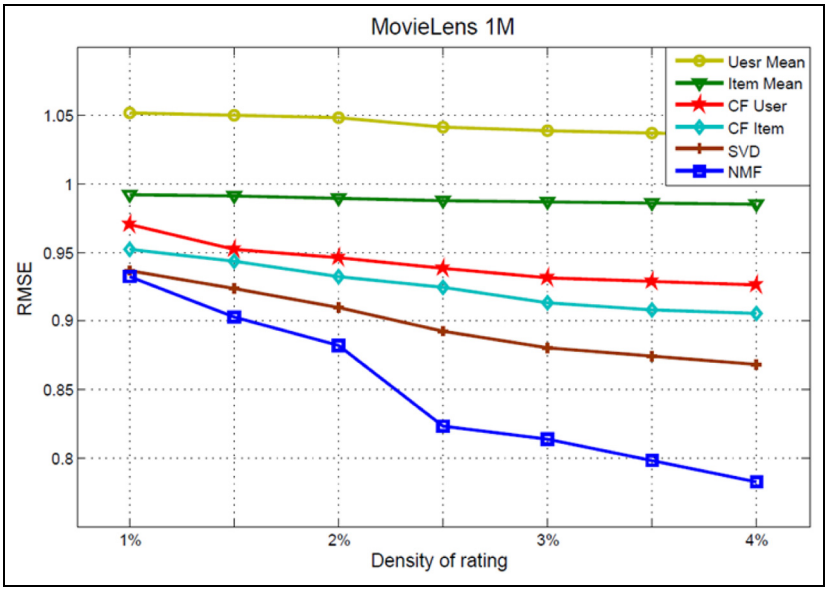

Figure 6. Comparison of RMSE for different data density levels on MovieLens 1M.

Table 5. Stability of the recommendation algorithms on the MovieLens datasets.

	MovieLens 100K		MovieLens 1M	
	MAE	RMSE	MAE	RMSE
User mean	0.0045	0.0065	0.0021	0.0042
Item mean	0.0021	0.0042	0.0014	0.0038
CF user	0.2825	0.4110	0.2385	0.3234
CF item	0.2089	0.3218	0.1724	0.2785
SVD	0.0982	0.1463	0.0824	0.1264
Proposed NMF	0.0421	0.0862	0.0275	0.0528

6. Conclusions

CF methods attempt to predict what the most interested items are based on the user's previous ratings in the system. In this paper, we focused on the application of NMF for CF. The basic idea is to assume that there exist a latent low dimensional representation of users' preferences and items' preferences where users and items can be modelled accurately. MF methods are a class of latent factor models that try to find weighted low-rank approximations from original ratings, where weights are used to hold out unknown ratings in the system.

Instance-based methods suffer from the lack of available ratings and cold start problems. Cold start is a problem common to most CF methods due to the fact that users usually rate only a small number of the items. Sparseness has an important effect on the accuracy and coverage of RSs. As RSs help users to find interested items in huge datasets, one of the goals is to scale up to real datasets. With the increase of the dataset, many approaches are either slowed down or require additional resources such as computation power or memory. So, dimensionality reduction comes in naturally. The proposed approach mitigates sparseness and scalability by applying the NMF. Our evaluation results verified that the proposed NMF effectively improves the prediction accuracy of CF and resolves the sparseness and scalability challenges. The updated rules are easy to implement computationally and will hopefully be utilised by others for a wide range of applications. As for future work, the intention is to combine the proposed NMF with clustering to improve the performance of CF.

Funding

This research was supported by the Iran Telecommunication Research Center (ITRC).

References

[1] Aghdam MH, Analoui M and Kabiri P. Modelling trust networks using resistive circuits for trust-aware recommender systems. *Journal of Information Science*. DOI: 10.1177/0165551516628733.

[2] Ricci F, Rokach L and Shapira B. Introduction to recommender systems handbook. In: Ricci F, Rokach L, Shapira B and Kantor PB. *Recommender Systems Handbook*. Amsterdam: Springer, 2011, pp. 1–35.

[3] Rafeh R and Bahrehmand A. An adaptive approach to dealing with unstable behaviour of users in collaborative filtering systems. *Journal of Information Science* 2012; 38: 205–221.

[4] Liu Y, Lin Z, Zheng X and Chen D. Incorporating social information to perform diverse replier recommendation in question and answer communities. *Journal of Information Science*. DOI: 10.1177/0165551515592093.

[5] Movahedian H and Khayyambashi MR. Folksonomy-based user interest and disinterest profiling for improved recommendations: An ontological approach. *Journal of Information Science* 2014; 40: 594–610.

[6] Hofmann T and Hartmann D. Collaborative filtering with privacy via factor analysis. In: *Proceedings of the ACM Symposium on Applied Computing* 2005, pp. 791–795.

[7] Hofmann T and Puzicha J. Latent class models for collaborative filtering. *IJCAI* 1999; 99: 688–693.

[8] Xue GR, Lin C, Yang Q, Xi W, Zeng HJ, Yu Y and Chen Z. Scalable collaborative filtering using cluster-based smoothing. In: *Proceedings of the 28th annual international ACM SIGIR Conference on Research and Development in Information Retrieval* 2005, pp. 114–121.

[9] Koren Y, Bell R and Volinsky C. Matrix factorization techniques for recommender systems. *Computer* 2009; 42: 30–37.

[10] Aghdam MH, Analoui M and Kabiri P. Application of nonnegative matrix factorization in recommender systems. In: *Proceeding of the Sixth International Symposium on Telecommunications (IST)* 2012, pp. 873–876.

[11] Aghdam MH, Analoui M and Kabiri P. A novel non-negative matrix factorization method for recommender systems. *Applied Mathematics & Information Sciences* 2015; 9(5): 2721–2732.

[12] Sindhwani V, Bucak S, Hu J and Mojsilovic A. A family of non-negative matrix factorizations for one-class collaborative filtering problems. In: *Proceedings of the ACM Recommender Systems Conference (RecSys)* 2009.

[13] Sarwar B, Karypis G, Konstan J and Riedl J. Application of dimensionality reduction in recommender system-a case study. Technical report, DTIC Document 2000.

[14] Srebro N, Rennie J and Jaakkola TS. Maximum-margin matrix factorization. In: *Advances in Neural Information Processing Systems* 2004, pp. 1329–1336.

[15] Hofmann T. Latent semantic models for collaborative filtering. *ACM Transactions on Information Systems (TOIS)* 2004; 22(1): 89–115.

[16] Kurucz M, Benczur AA and Csalogany K. Methods for large scale SVD with missing values. In: *Proceedings of KDD Cup and Workshop* 2007, pp. 31–38.

[17] Lee DD and Seung HS. Algorithms for non-negative matrix factorization. In: *Advances in Neural Information Processing Systems* 2001, pp. 556–562.

[18] Cichocki A, Zdunek R, Phan AH and Amari S. *Nonnegative Matrix and Tensor Factorizations: Applications to Exploratory Multi-Way Data Analysis and Blind Source Separation*. Oxford: John Wiley & Sons, 2009.

[19] Jannach D, Zanker M, Felfernig A and Friedrich G. *Recommender Systems: an Introduction*. Cambridge: Cambridge University Press, 2010.
[20] Bennett J and Lanning S. The netflix prize. In: *Proceedings of KDD cup and workshop* 2007, p. 35.
[21] Koren Y. Factor in the neighbors: Scalable and accurate collaborative filtering. *ACM Transactions on Knowledge Discovery from Data (TKDD)* 2010; 4(1): 1–24.
[22] Sarwar B, Karypis G, Konstan J and Riedl J. Item-based collaborative filtering recommendation algorithms. In: *Proceedings of the 10th international Conference on World Wide Web, ACM* 2001, pp. 285–295.
[23] Herlocker JL, Konstan JA, Terveen LG and Riedl JT. Evaluating collaborative filtering recommender systems. *ACM Transactions on Information Systems (TOIS)* 2004; 22(1): 5–53.
[24] Adomavicius G and Zhang J. Stability of recommendation algorithms. *ACM Transactions on Information Systems (TOIS)* 2012; 30(4): 23.
[25] Chen G, Wang F and Zhang C. Collaborative filtering using orthogonal nonnegative matrix tri-factorization. *Information Processing & Management* 2009; 45(3): 368–379.
[26] Breese JS, Heckerman D and Kadie C. Empirical analysis of predictive algorithms for collaborative filtering. In: *Proceedings of the Fourteenth conference on Uncertainty in artificial intelligence* 1998, pp. 43–52.
[27] Pennock DM, Horvitz E, Lawrence S and Giles CL. Collaborative filtering by personality diagnosis: A hybrid memory-and model-based approach. In: *Proceedings of the Sixteenth conference on Uncertainty in artificial intelligence* 2000, pp. 473–480.
[28] Wang J, De Vries AP and Reinders MJ. Unifying user-based and item based collaborative filtering approaches by similarity fusion. In: *Proceedings of the 29th annual international ACM SIGIR conference on Research and development in information retrieval* 2006, pp. 501–508.