Enhanced SVD (ESVD) for Collaborative Filtering

Xin Guan[1], Chang-Tsun Li[1], and Yu Guan[2]

[1] Department of Computer Science, University of Warwick, UK
{x.guan,c-t.li}@warwick.ac.uk
[2] School of Computing Science, Newcastle University, UK
yu.guan@ncl.ac.uk

Abstract. Matrix factorization is one of the most popular techniques for prediction problems in the fields of intelligent systems and data mining. It has shown its effectiveness in many real-world applications such as recommender systems. As a collaborative filtering method, it gives users recommendations based on their previous preferences (or ratings). Due to the extreme sparseness of the ratings matrix, active learning is used for eliciting ratings for a user to get better recommendations. In this paper, we propose a new matrix factorization model called ESVD which combines the classic matrix factorization method with a specific rating elicitation strategy. We evaluate the proposed ESVD method on the Movielens data set, and the experimental results suggest its effectiveness in terms of accuracy and efficiency, when compared with traditional matrix facterization methods and active learning methods.

Keywords: matrix factorization, recommender systems

1 Introduction

Generally speaking, recommender systems provide users with personalized suggestions by predicting the rating or preference that the users would give to an item. They have become increasingly common recently and are used by many internet leaders. Examples include movie recommendation by Netflix [1], web page ranking by Google [2], related product recommendation by Amazon [3], social recommendation by Facebook [4], etc.

Recommender systems are used for generating recommendations to users, usually in one of the two ways: 1) content-based filtering based on the characteristics it has and the item descriptions which could be automatically extracted or manually created, and 2) collaborative filtering that predicts other items the users might like based on the knowledge about preferences (usually expressed in ratings) of users for some items. Due to the limitations of content-based filtering algorithm (sometimes characteristics are unobvious and item descriptions are hard to extract), most recommender systems are based on collaborative filtering.

Collaborative filtering is a technique used to predict the preferences of users according to the same taste or common experiences from others and based on the assumption that people who agreed in the past will also agree in the future over time. There are two primary strategies to deal with collaborative filtering: the neighborhood approaches and latent factor models. Neighborhood methods [5] concentrate on the relationship between items or users, so they are good at detecting localized relationships (e.g., someone who likes Superman also likes Batman), but fall short of power in detecting the user's overall preference. By transforming both items and users to the same latent space, latent factor models try to explain the ratings by items and users, aiming at making them directly comparable. Latent factor models are good at estimating the overall structure (e.g., a user likes comedy movies), but less effective in analyzing associations among small sets of closely related items.

Matrix factorization methods, such as Non-Negative Matrix Factorization (NMF) [6] and Singular Value Decomposition (SVD) [7], are widely used for constructing a feature matrix for users and for items, respectively. Generally, matrix factorization, as one of the most successful realizations of latent factor models, can produce better accuracy than classic nearest neighbor methods when dealing with product recommendations because of the incorporation of additional information such as implicit feedback and temporal effects [8].

In real life situations, when a new user comes in, most recommender systems would only ask the user to rate a certain number of items (which is a small proportion comparing with the whole set). Therefore the ratings matrices are often extremely sparse, which means there is not enough knowledge to form accurate recommendations for the user. To get precise recommendations for this user, active learning in collaborative filtering is often used to acquire more high-quality data in order to improve the precision of recommendations for the target user. However, traditional active learning methods [9] [10] [11] only evaluate each user independently and only consider the benefits of the elicitation to the 'new' user, but pay less attention to the effects of the system. In this work we propose a rating elicitation strategy that improves the accuracy of the whole system by eliciting more ratings for 'existing' users. In some previous works, ratings were elicited one by one per request [9] or user's by user's per request [11]. The consequence is that the model is trained at each request, which is significantly time-consuming. In this paper, we design a series of methods that elicit ratings simultaneously with matrix factorization algorithms. Through this special preprocessing step not only are the computational costs reduced, but also the performance of matrix factorization methods is greatly improved.

The rest of the paper is organized as follows. We start with preliminaries and related work in Section 2. Then in Section 3 we introduce a new way to apply active learning to recommender systems. A more accurate model is proposed in Section 4. Experimental results and analyses are provided in Section 5. Section 6 concludes the work.

2 Preliminaries

2.1 Regularized SVD

Normally, matrix factorization methods are used to deal with the issue of missing values in a given ratings matrix. However, ratings matrices are usually extremely sparse. For example, the density of the famous Netflix and Movielens data sets are 1.18% and 4.61%, respectively, which means that only a few elements are rated while most of them are unknown. Another challenge is that the data set we use in real world recommender systems is typically of high dimensionality. Due to high sparseness and computational complexity, directly applying SVD algorithms to rating matrices is not appropriate.

In [12], Funk proposed an effective method called regularized SVD algorithm for collaborative filtering which decomposes the ratings matrix into two lower rank matrices. Suppose $R \in \mathbb{R}^{m \times n}$ is the ratings matrix of m users and n items. The regularized SVD algorithm finds two matrices $U \in \mathbb{R}^{k \times m}$ and $V \in \mathbb{R}^{k \times n}$ as the feature matrix of users and items:

$$\tilde{R} = U^T V \tag{1}$$

It assumes that each user's rating is composed of the sum of preferences about the various latent factors of that movie. So each rating r_{ij} the ith user to the jth movie in the matrix R can be represented as:

$$\tilde{r}_{ij} = {U_i}^T V_j \tag{2}$$

where U_i, V_j are the feature vectors of the ith user and the jth movie, respectively. Once we get the best approximation of U and V, we can obtain the best prediction accordingly. The optimization of U and V can be performed by minimizing the sum of squared errors between the existing scores and prediction values [12]:

$$E = \frac{1}{2} \sum_{i,j \in \kappa} (r_{ij} - \tilde{r}_{ij})^2 + \frac{k_u}{2} \sum_{i=1}^{m} {U_i}^2 + \frac{k_v}{2} \sum_{j=1}^{n} {V_j}^2 \tag{3}$$

where κ is a set of elements in the ratings matrix R that have been signed values, k_u and k_v are regularization coefficients to prevent over-fitting.

To solve the optimization problem like Equation (3), Stochastic Gradient Descent (SGD) is widely used and has shown to be effective for matrix factorization [12] [13] [14]. SGD loops through all ratings in the training set κ and for each rating it modifies the parameters U and V in the direction of the negative gradient:

$$U_i \leftarrow U_i - \alpha \frac{\partial E_{ij}}{\partial U_i} \tag{4}$$

$$V_j \leftarrow V_j - \alpha \frac{\partial E_{ij}}{\partial V_j} \tag{5}$$

where α is the learning rate.

Unlike traditional SVD, regularized SVD is just a tool for finding those two smaller matrices which minimize the resulting approximation error in the least square sense. By solving this optimization problem, the end result is the same as SVD which just gets the diagonal matrix arbitrarily rolled into the two side matrices, but could be easily extracted if needed.

2.2 SVD++

Since matrix factorization for recommender systems based on regularized SVD was first proposed, several variants have been exploited with extra information on the ratings matrix. For example, Paterek [13] proposed an improved regularized SVD algorithm by adding a user bias and an item bias in the prediction function. Koren [14] extended the model by considering more implicit information about rated items and proposed a SVD++ model with the prediction function:

$$\tilde{r}_{ij} = u + \beta_i + \gamma_j + V_j^T (U_i + |I(i)|^{(-1/2)} \sum_{k \in I(i)} y_k), \tag{6}$$

where u is the global mean, β_i is the bias of the ith user and γ_j is the bias of the jth item. U_i is learnt from the given explicit ratings, *I(i)* is the set of items user i has provided implicit feedback for. $|I(i)|^{(-1/2)} \sum_{k \in I(i)} y_k$ represents the influence of implicit feedback. The implicit information enables SVD++ to produce better performance than regularized SVD model.

2.3 Dataset

Movielens is a classic recommender system data set that recommends films to users through collaborative filtering algorithms. It contains 100,000 ratings from 943 users on 1,682 movies and each rating is an integer ranging from 1 to 5 which represents the interests the user has to this movie. All users were selected at random for inclusion and each user selected had rated at least 20 movies. The whole data set is randomly partitioned into training set and test set with exactly 10 ratings per user in the test set. The quality of results is usually measured by their RMSE:

$$\text{RMSE} = \sqrt{\frac{\sum_{(i,j \in TestSet)} (r_{ij} - \tilde{r}_{ij})^2}{T}} \tag{7}$$

where T is the total number of test samples.

RMSE is one of the common error metrics and often used to evaluate recommender systems especially in the official Netflix contest [1].

3 A System View of Active Learning for Recommendation Systems

It is important to note that the characteristics of the prediction algorithm may influence the prediction accuracy. *Matrix factorization methods learn the model by fitting a limited number of existing ratings, hence we could make an assumption that the more reliable ratings we have, the better accuracy the model can reach.* However, in most recommendation system data sets, the ratings matrices are extremely sparse because a user typically only rates a small proportion of movies while most ratings are unknown, which motivates us to add more high quality data for matrix factorization.

3.1 The Item-oriented Approach

Classic active learning methods focus on different individual rating elicitation strategies for a single user when a new user comes in. These strategies include:

1. *Randomization*: It can be regarded as a baseline method (e.g., [15] [16] [11]).
2. *Popularity-based*: Items with the largest number of ratings are preferred. It is based on the assumption that the more popular the items are, the more likely that they are known by this user (e.g., [15] [17]).
3. *Entropy-based*: Items with the largest entropy are selected [15].
4. *Highest predicted*: Items with the highest predicted ratings are preferred [16].
5. *Lowest predicted*: Items with the lowest predicted ratings are chosen. The items with the lowest ratings are supposed to be the most disliked movies for this user, which also may influence the user to rate them [16].
6. *Hybrid*: This includes Log(popularity)*entropy [15], *Voting*, which considers the overall effect of previous methods [11] [18].

Previous works (e.g., [15] [11] [16]) focused on the accuracy of the recommendations for 'a single user' and the model was trained in each iteration user by user, which incurs high computational costs. While implementing classic active learning strategies, the items selected for different users to elicit are always different. For example, the items with the highest predicted ratings for a user may not be the same as another user's since not all users have the exactly same tastes. Hence strategy has to be applied repeatedly for each user to elicit ratings which are corresponding to different items but with one exception: popularity-based strategy.

On the other hand, the movie popularity may vary significantly. Take Movenlens data set for example, the maximal and minimum number of popularity is 495 and 0, respectively, which means that the most popular movie is rated by 495 users while for some movies there is no one has rated them before. Popularity is based on the number of ratings of each item only and it is irrelevant to users, therefore the popularity remain unchanged no matter which users we choose. Here we only select the top N most popular movies for all the users to form a new block matrix, based on the idea that users tend to rate world-famous movies

Algorithm 1 The item-oriented approach

```
Input: Ratings matrix $R \in \mathbb{R}^{m \times n}$, where $Q_{j \in [1,n]} \in \mathbb{R}^{m \times 1}$ is the column vector, $\kappa$ is a set of elements in the ratings matrix that have been assigned values; the top number of items selected in the block matrix based on popularity $N$;
Output: RMSE of the test set;
    Step 1: Sort items based on the number of ratings each user rate this item (popularity) in descending order $j(1), j(2), ..., j(m)$;
    Step 2: Create a block $M_1$ by selecting the top $N$ of items (columns) in Step 1 based on the popularity. Therefore $M_1 = [Q_{j(1)}, Q_{j(2)}, ...., Q_{j(N)}](N < m)$;
    Step 3: Apply basic matrix factorization (regularized SVD) on matrix $M_1$ to obtain feature matrices $U$ and $V$ according to Equation (1);
    Step 4: Predict every missing value in block $M_1$ to acquire a non-null matrix $M'_1$ according to Equation (2). Then a series of ratings $L_1$ is elicited, such that $L_1 = \{$
    $r_{i_{k(1)},j(1)}, r_{i_{k(2)},j(1)}, ..., r_{i_{k(n)},j(1)},$
    $r_{i_{k(1)},j(2)}, r_{i_{k(2)},j(2)}, ..., r_{i_{k(n')},j(2)},$
    ......,
    $r_{i_{k(1)},j(N)}, r_{i_{k(2)},j(N)}, ..., r_{i_{k(n")},j(N)}\}$
    where $r_{i_k,j} \notin \kappa$;
    Step 5: Fill ratings in the original matrix $R$ with every predicted value by Step 4 to acquire a new ratings matrix $R'$. That means we add the elicited ratings into the set of existing ratings. $\kappa = \{\kappa, L_1\}$;
    Step 6: Apply basic matrix factorization (regularized SVD) on matrix $R'$ to obtain feature matrices $U'$ and $V'$ according to Equation (1);
    Step 7: Predict the target ratings (test set) according to Equation (2) and calculate RMSE according to Euqation (7);
```

than movies in the minority. Then the missing values in this block would be the 'desirable' movies in some sense for the users who missed before. Our strategy is to elicit these specific ratings for all the users at the same time in one iteration through matrix factorization on this block matrix. After adding these ratings to the original rating matrix, a more accurate matrix factorization model could be extracted. The overall procedure of this approach are shown in Algorithm 1.

In summary, Algorithm 1 simultaneously elicit ratings of popular movies for all users, in order to improve the performance of the whole system. Therefore, it reduces the training iteration for as high as the number of users to only 2, and evaluate the benefits of the system instead of each user.

3.2 The User-oriented Approach

Traditional active learning for collaborative filtering is a set of techniques to intelligently select a number of 'items' to rate so as to improve the rating prediction for the user, while Carenini et al. [17] proposed an item-focussed method that elicits ratings by choosing some special users to rate a specific item in order to improve the rating prediction for the item. Because of the feasibility of this item-focussed method we also propose a user-oriented approach as shown in Algorithm 2 to further explore its potential.

Generally, the number of movies each user has rated varies significantly (e.g., in Movielens data set the maximal and minimum number for different user's are 727 and 10, respectively). In fact, though active users who are enthusiastic about movies may watch far more than the ones who are not into movies, there still exist some movies the users have watched but not yet rated. For these reasons, it is easier to accept that they would give ratings to the movies they have not rated yet. In Algorithm 2 we select this kind of special users based on the number of movies they have rated. After these movie enthusiasts are chosen, ratings of the movies they never rate would be elicited by matrix factorization as the missing values in the new block. Then we add these new ratings to the original matrix for matrix factorization again to get the target values we want.

In brief, Algorithm 2 tries to improve the performance of the whole system by eliciting ratings of all movies for only active users simultaneously. Therefore it also has the benefits that Algorithm 1 has. However, both algorithms still may suffer from large computational cost because of the extensively selection of elicitations, especially when the number of popular movies or active users selected in the block matrix is large.

Algorithm 2 The user-oriented approach

```
Input: Ratings matrix $R \in \mathbb{R}^{m \times n}$, where $P_{i \in [1,m]} \in \mathbb{R}^{1 \times n}$ is the row vector, $\kappa$ is a set of elements in the ratings matrix that have been signed values; the top number of users selected in the block matrix based on activity $N'$;
Output: RMSE of the test set;
  Step 1: Sort users based on the number of ratings they rates (activity) in descending order $i(1), i(2), ..., i(n)$;
  Step 2: Create a block $M_2$ by selecting the top $N$ of users(rows) in Step 1 based on the activity. Therefore $M_2 = [P_{i(1)}, P_{i(2)}, ...., P_{i(N)}](N < n)$;
  Step 3: Apply basic matrix factorization (regularized SVD) on matrix $M_2$ to obtain feature matrices $U$ and $V$ according to Equation (1);
  Step 4: Predict every missing value in block $M_2$ to acquire a non-null matrix $M'_2$ according to Equation (2). Then a series of ratings $L_2$ is elicited, such that $L_2 = \{$
  $r_{i(1),j_k(1)}, r_{i(1),j_k(2)}, ..., r_{i(1),j_k(n)},$
  $r_{i(2),j_k(1)}, r_{i(2),j_k(2)}, ..., r_{i(2),j_k(n')}$
  $r_{i(N'),j_k(1)}, r_{i(N'),j_k(2)}, ..., r_{i(N'),j_k(n'')}\}$
  where $r_{i,j_k} \notin \kappa$;
  Step 5: Fill ratings in the original matrix $R$ with every predicted value by Step 4 to acquire a new ratings matrix $R'$. That means we add the elicited ratings into the set of existing ratings. $\kappa = \{\kappa, L_2\}$;
  Step 6: Apply basic matrix factorization (regularized SVD) on matrix $R'$ to obtain feature matrices $U'$ and $V'$ according to Equation (1);
  Step 7: Predict the target ratings (test set) according to Equation (2) and calculate RMSE according to Euqation (7);
```

4 The Proposed ESVD (Density-oriented Approach)

So far we have presented an item-oriented approach and a user-oriented approach, both based on the idea that eliciting a group of reliable and meaningful ratings simultaneously for matrix factorization model to learn. The reason why these new ratings from Agorithm 1 and Agorithm 2 should be reliable is because they are predicted from a denser block which is formed from the top numbers of ratings in either item-view or user-view by a matrix factorization method. Typically the denser the matrix is, the better the matrix factorization model we can obtain. Take the Movielens data set as an example, the density of the original matrix is 5.71%. However, if only the top 5% of the most popular movies are chosen, we can obtain a block of density 29.47% which consists of more ratings that have been already rated by the users. While selecting the top 5% of the most active users, the density of the new block we obtain is 23.33%. Based on this assumption we propose a density-oriented approach which combines previous item-oriented and user-oriented methods in Algorithm 3.

Algorithm 3 The proposed density-oriented approach

```
Input: Ratings matrix $R \in \mathbb{R}^{m\times n}$, where $P_{i\in[1,m]} \in \mathbb{R}^{1\times n}$ is the row vector and
    $Q_{j\in[1,n]} \in \mathbb{R}^{m\times 1}$ is the column vector, $\kappa$ is a set of elements in the ratings ma-
    trix that have been assigned values; The top number of items selected in the block
    matrix based on popularity $N$ and the top number of users selected in the block
    matrix based on activity $N'$;
Output: RMSE of the test set;
    Step 1: Sort both items and users in descending order based on popularity and
    activity respectively. $j(1), j(2), ..., j(m)$; $i(1), i(2), ..., i(n)$;
    Step 2: Create a block $M_3$ by selecting the intersection of top $N$ of items (columns)
    and top $N'$ of users (rows) in Step 1 based on the popularity and activity. Therefore
    $M_3 = M_1 \bigcap M_2$;
    Step 3: Apply basic matrix factorization (regularized SVD) on matrix $M_3$ to obtain
    feature matrices $U$ and $V$ according to Equation (1);
    Step 4: Predict every missing value in block $M_3$ to acquire a non-null matrix $M'_3$
    according to Equation (2). Then a series of ratings $L_3$ is elicited, such that $L_3 =$
    $\{r_{i_{k(1)},j_{t(1)}}, ..., r_{i_{k(n)},j_{t(n')}}\}$where $r_{i_k,j_t} \in (M_3 \bigcap \neg\kappa)$;
    Step 5: Fill ratings in the original matrix $R$ with every predicted value by Step 4.
    That means we add the elicited ratings into the set of existing ratings. $\kappa = \{\kappa, L_3\}$;
    Step 6: Apply basic matrix factorization (regularized SVD) on matrix $R'$ to obtain
    feature matrices $U'$ and $V'$ according to Equation (1);
    Step 7: Predict the target ratings (test set) according to Equation (2) and calculate
    RMSE according to Euqation (7);
```

Because both the popularity of movies and the activity of users depend on the numbers of ratings each user rates or each movie is rated, by choosing the top N popular movies and active users we can obtain the densest block matrix. With the Movielens data set, if we choose the top 5% of the most popular movies and most active users, the density of the newly-formed block matrix would be 77.28%. The missing values in this block matrix can be explained as ratings of the most famous movies but have not been rated by a group of most active users. By applying matrix factorization on this block, a better model could be extracted and the missing values (elicited ratings) could be predicted with higher accuracy. Finally a more accurate matrix factorization model can be learnt by fitting the existing ratings and extra high quality elicited ratings.

5 Experimental Results

We conducted experiments of our proposed item-oriented, user-oriented and density-oriented approach (ESVD) on the classic recommender system data set: Movielens 100K. We also performed some experiments with the larger version and obtained similar results. However, it requires much longer time to perform our experiments since we train and test the models each time for different choice of the number of N. Therefore, we focussed on the smaller data set to be able to run more experiments, in order to explore how this parameter N affects the results of our matrix factorization methods. We use RMSE as the default metric which is widely used in the Netfilx Competition and proved to be effective to measure recommender systems. The number of the latent factors (rank) are set to be 10 for training each matrix factorization model. Although increasing it does raise the performance, the computational cost is proportional to latent factors. For matrix factorization of the comparatively smaller block matrix M_1, M_2 and M_3, the coefficient of the regularization term is 0.01 for both k_u and k_v, and the learning rate α is 0.1 with a decrease by a factor of 0.9 each iteration. For matrix factorization of the ratings matrix (with elicited ratings) R', the coefficient of regularization term is 0.1 for both k_u and k_v, and the learning rate α is 0.01 with decrease by a factor of 0.9 each iteration.

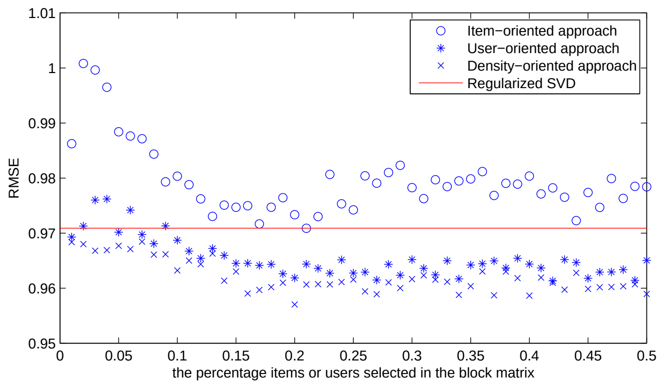

Fig. 1: RMSE comparisons of proposed methods based on SVD

Fig.1 shows the results of proposed methods based on different number of items or users N selected in the block matrix. We see that as the number of items or users N increases, the errors set keeps dropping since more ratings are added into the training matrix, afterwards the errors fluctuate in a small region. Specifically, the results of the item-oriented approach (Algorithm 1) are not promising. Because in the item-oriented approach we only elicit ratings based on the most popular movies, which may lead to a lot of bias and distort the latent factor model. For example, most people prefer happy endings, the consequence is that comedies are more popular than tragedies. Then a lot of comedy movies would be elicited for each user to give ratings which leads the latent factor model (regularized SVD in this case) to put more weights on the factor corresponding to comedies. Although the RMSE of user-oriented approach (Algorithm 2) fluctuates around the baseline method (the red line) at first, it gets lower as N goes up. The best result of Algorithm 2 we obtain is 0.9619 which reduces RMSE by 0.09 when compared with the regularized SVD 0.9709. It is apparent from the figure that our proposed ESVD consistently outperform other methods including the baseline method: regularized SVD.

Table 1: RMSE of ESVD (the density-oriented approach)

Top Items&Users	0%	5%	10%	15%	20%
Block Density	null	77.28%	65.20%	53.90%	45.66%
Elicited Ratings	null	897	5496	16381	34508
RMSE (Test)	0.9709	0.9677	0.9632	0.9630	0.9570

In Table 1, we illustrate the RMSE of proposed density-oriented (ESVD) method which incorporates both item-oriented and user-oriented approach on the same data set. We compare different RMSE based on how many items and users (from 0% to 20%) we select. Note that the basic matrix factorization is a special case of our method when setting $N = 0\%$, which is used as the baseline for comparision. After selecting a certain percentage of items and users, we can obtain a block matrix. We can observe that the more items and users we choose, the sparser the block matrix will be. The missing values in the block are chosen to be elicited ratings. Although sparser matrix may lead to a less accurate matrix factorization model and the quality of elicited ratings may not as good as the ones from the denser matrix, the number of the elicited ratings is increased. Therefore more ratings can be added into the process of learning the target matrix factorization model. At last we compute predictions on the test set and get results. Because the block matrix is the intersection of the top N items and N users, its density is much greater than the one from item-oriented or user-oriented approach. Even with less ratings to be elicited compared with item-oriented and user-oriented, the results are better. In our experiments, we observed that the performance fluctuates as the number of top projects increases (Table 1). The optimal point that balance the quality (density of block matrix) and the quantity (number of elicited ratings) depends on the distribution of ratings. The Algorithm 3 can reach to 0.9570 (when $N = 20\%$) which reduces the RMSE by 0.0139 compared with the regularized SVD 0.9709.

5.1 Extension: ESVD++

Broadly speaking, our proposed ESVD approach can be seen as a preprocessing step and it can be incorporated with other variants of SVD models, such as SVD++ [14]. We conduct ESVD++ by just changing the prediction algorithm from SVD to SVD++. Compared with SVD model, SVD++ improves the prediction accuracy by adding biases and implicit information $I(i)$, and the prediction function is shown in Equation (6). Specifically, $I(i)$ contains all items for which the ith user has provided a rating, even if the value is unknown. Therefore, for prediction of elicited ratings as shown in Step 4 of Algorithm 3, $I(i)$ is set to be the number of existing ratings and the missing values in the block matrix that are also shown in the test set. For prediction of test set as shown in Step 7 of Algorithm 3, $I(i)$ is the same as the one in original matrix without considering the elicited ratings.

As the elicitation strategy is the same as ESVD, the ratings that need to be elicited would also be the same. The results show that the ESVD++ outperforms the state-of-art SVD++ model and greatly reduces the RMSE by 0.0214 (from the baseline SVD++ 0.9601 to 0.9387 when $N = 10\%$).

6 Conclusion

The lack of information is an acute challenge in most recommender systems. In this paper, we present a new matrix factorization method called ESVD which applies SVD with ratings elicitation that best approximates a given matrix with missing values. We conduct corresponding experiments on the Movielens data set. Instead of looking at the active learning from the individual user's point of view, we dealing with the problem from the system's perspective. Although the proposed method cannot deal with the cold start problem where the database keeps growing as new users or items continue to be added, it does reduce the computational costs greatly (the iteration number from as high as the number of users to only 2) since all the ratings are elicited simultaneously. Compared with traditional matrix factorization models, it can be incorporated with different SVD-based algorithms and greatly improve the performance of the whole system. Because the model is learnt by extra high-quality-elicited ratings that are extracted from the densest block matrix we create based on item popularity and user activity. Even using basic SVD as a prediction algorithm, without considering the effect of implicit feedbacks, the performance is better than its counterpart SVD++.

References

1. J. Bennett and S. Lanning, "The netflix prize," in *Proceedings of KDD cup and workshop*, 2007, p. 35.
2. A. S. Das, M. Datar, A. Garg, and S. Rajaram, "Google news personalization: scalable online collaborative filtering," in *Proceedings of the 16th international conference on World Wide Web*. ACM, 2007, pp. 271–280.
3. G. Linden, B. Smith, and J. York, "Amazon.com recommendations: Item-to-item collaborative filtering," vol. 7, no. 1. IEEE, 2003, pp. 76–80.
4. E.-A. Baatarjav, S. Phithakkitnukoon, and R. Dantu, "Group recommendation system for facebook," in *On the Move to Meaningful Internet Systems: OTM 2008 Workshops*. Springer, 2008, pp. 211–219.
5. R. M. Bell and Y. Koren, "Scalable collaborative filtering with jointly derived neighborhood interpolation weights," in *Data Mining, Seventh IEEE International Conference on*. IEEE, 2007, pp. 43–52.
6. Y.-X. Wang and Y.-J. Zhang, "Nonnegative matrix factorization: A comprehensive review," *Knowledge and Data Engineering, IEEE Transactions on*, vol. 25, no. 6, pp. 1336–1353, 2013.
7. G. Golub and W. Kahan, "Calculating the singular values and pseudo-inverse of a matrix," *Journal of the Society for Industrial & Applied Mathematics, Series B: Numerical Analysis*, vol. 2, no. 2, pp. 205–224, 1965.
8. Y. Koren, R. Bell, and C. Volinsky, "Matrix factorization techniques for recommender systems," *Computer*, no. 8, pp. 30–37, 2009.
9. A. S. Harpale and Y. Yang, "Personalized active learning for collaborative filtering," in *Proceedings of the 31st annual international ACM SIGIR conference on Research and development in information retrieval*. ACM, 2008, pp. 91–98.
10. R. Jin and L. Si, "A bayesian approach toward active learning for collaborative filtering," in *Proceedings of the 20th conference on Uncertainty in artificial intelligence*. AUAI Press, 2004, pp. 278–285.
11. M. Elahi, F. Ricci, and N. Rubens, "Active learning strategies for rating elicitation in collaborative filtering: a system-wide perspective," *ACM Transactions on Intelligent Systems and Technology (TIST)*, vol. 5, no. 1, p. 13, 2013.
12. S. Funk, "Netflix update: Try this at home," 2006.
13. A. Paterek, "Improving regularized singular value decomposition for collaborative filtering," in *Proceedings of KDD cup and workshop*, 2007, pp. 5–8.
14. Y. Koren, "Factorization meets the neighborhood: a multifaceted collaborative filtering model," in *Proceedings of the 14th ACM SIGKDD international conference on Knowledge discovery and data mining*. ACM, 2008, pp. 426–434.
15. A. M. Rashid, I. Albert, D. Cosley, S. K. Lam, S. M. McNee, J. A. Konstan, and J. Riedl, "Getting to know you: learning new user preferences in recommender systems," in *Proceedings of the 7th international conference on Intelligent user interfaces*. ACM, 2002, pp. 127–134.
16. B. M. Marlin, R. S. Zemel, S. T. Roweis, and M. Slaney, "Recommender systems, missing data and statistical model estimation." in *IJCAI*, 2011, pp. 2686–2691.
17. G. Carenini, J. Smith, and D. Poole, "Towards more conversational and collaborative recommender systems," in *Proceedings of the 8th international conference on Intelligent user interfaces*. ACM, 2003, pp. 12–18.
18. N. Golbandi, Y. Koren, and R. Lempel, "On bootstrapping recommender systems," in *Proceedings of the 19th ACM international conference on Information and knowledge management*. ACM, 2010, pp. 1805–1808.