Rating Prediction Using Preference Relations Based Matrix Factorization

Maunendra Sankar Desarkar and Sudeshna Sarkar

Department of Computer Science and Engineering,
Indian Institute of Technology Kharagpur,
Kharagpur - 721302,
India

Abstract. *Rating prediction* is an important problem for rating based recommender systems. In Rating Prediction, the task is to predict the rating that a user would give to an item that he/she has not rated in the past. Most of the existing algorithms for the task concentrate on the absolute ratings given to different items by different users in the past. However, there are few recent research work that point out some drawbacks of absolute rating based systems and algorithms, and suggest the use of preference relations between pairs of items to capture the users' interests about the items. In this paper, we propose a rating prediction algorithm that considers the relative ratings given by the users for different pairs of items. The algorithm models the users and items using a matrix factorization framework. The learned model of users and items are first used to predict the personalized utility of an item for a user. This utility is then converted to a valid rating value in a predefined rating scale by employing a *personalized scaling*. Experimental evaluation on a benchmark dataset reveals that better prediction accuracies may be achieved by modeling the users and items using relative rating information.

Keywords: Rating Prediction, Preference Relations, Matrix Factorization.

1 Introduction

The *rating prediction problem* is a widely studied problem in the domain of rating based recommender systems, where the goal is to predict the rating that a user would assign to an item that he/she has not rated in the past. The system may assume that the items with higher predicted rating for a user may be interesting to the user, and those items can be recommended to the user. Hence rating prediction is an important problem for recommender systems.

Although rating based recommender systems are widely popular, several issues associated with absolute ratings have been highlighted in recent literature. To get around some of these issues, several recent research have suggested the use of preference relation information for efficient recommendation. For example, it might be difficult to choose a rating for an item. Whereas, given a pair of items, it might be easier for a user to tell which one he/she likes more, or both are equally preferable. Also, there are some users who always tend to give higher ratings to the items, and there are some who always give low ratings. In other words, the users have different rating biases, and these biases should be carefully handled before using the information for recommendation tasks. When preference relations are used, this type of rating bias is automatically eliminated [1]. It has also been shown by a user survey [2] that users favor providing feedbacks in the form of preference relations. Even for systems where absolute rating are available, use of relative rating information or preference relations may be used to achieve better recommendation performance [3].

In this paper, we present a preference relation based algorithm for the *rating prediction problem* in recommender systems. Given a user and a pair of items, we look at the relative rating (or preference relation) given by the user for this pair of items and use this information for the prediction taskWe perform experiments on a benchmark dataset and show that use of preference relations may help in achieving better prediction performance.

2 Related Work

Rating prediction is a widely studied problem in the domain of recommender systems. One of the most widely used frameworks for the rating prediction problem is the collaborative filtering approach. Here, the system measures the similarity weights between every pair of users by looking at the ratings provided by them to the items that both of them have rated. Pearson Correlation and Vector Similarity [4] are two most common measures for finding the user similarities. Later several researchers have proposed different other measures for calculating user similarities [5, 6, 7]. Weighted average of the *most similar* users' ratings for the test items are output as the predicted rating. There are several algorithms that use probabilistic graphical models for solving the task of rating prediction [8, 9, 10]. Matrix factorization algorithms have also been widely popular. These algorithms model both the user and items as vectors in a low dimensional feature space. Representation of the user and items in the joint feature space is then used to compute the predicted ratings.

However, almost all of the above methods are based on absolute ratings entered by the users. Recently, few researchers have pointed out several problems with actual ratings based systems and algorithms. These observations have encouraged people to look at algorithms that consider the relative ratings between items. Given a user and a pair of items, these algorithms take into account the ratings that the user has given for the item pair. This relative rating indicates which of the two items in the pair is more preferable to the user and can be directly accepted as a feedback in the form of preference relations. Even when preference relations are not directly available as a feedback, algorithms may construct preference relations by looking at the ratings given to the item pair. Few recent algorithms advocate the use of such induced preference relations.

An algorithm that uses only preference relations and ignores absolute rating almost completely for the rating prediction problem is given in [1]. It views the rating profile of each user as a preference graph. Similarity weight between two users is determined by considering the similarities between their preference graphs. First an aggregate preference graph biased to the target user's interest is generated by taking a weighted aggregation of his/her nearest neighbors' preference graphs. Integer ratings for the test items are then picked by minimizing the total weights of the *back edges* in this aggregate graph.

Another algorithm that uses such preference relations for neighborhood based collaborative rating prediction is given in [11]. It measures the similarities between two users by considering the number of item pairs for which both the users have similar relative preferences. Ratings given to the test items by the *nearest neighbors* can then be used for predicting the unknown ratings. The work proposed in [12] motivates the use of *Somers' coefficient* for measuring user similarities. Somers' coefficient between two users is high if there is a large number of item pairs that both of them have rated above (or below) their (the users') mean ratings. Weighted average of the ratings given by the similar users to the test item can be output as the predicted rating.

A preference relation based algorithm for item recommendation is proposed in [3]. It models the items and the users using a matrix factorization framework. The algorithm learns the user and item profile vectors by looking at the relative ratings provided by the users. User's affinity or utility value for a particular item is measured by the inner product of the corresponding user and the item profile vectors. In [13], the authors propose an algorithm that uses preference relations for predicting personalized rankings of items in presence of implicit feedback data. The algorithm learns a model so that the predicted scores of the *seen items* are more than the predicted scores for the *unseen items*. Both these algorithms find the recommendations by using personalized utility values of the items. However, these utility values are real numbers and are not in any fixed range, whereas in rating prediction, the estimated ratings need to conform to the rating scale used by the particular recommender system. Hence these algorithms can not be directly applied for solving the rating prediction problem.

3 Rating Prediction: The Problem Description

We assume that the recommender system has a log of past ratings entered by the users. Given a user u and an item i that u has not rated in the past, the goal of the *rating prediction problem* is to predict the rating that u would have given to i.

In absolute rating based systems, each entry in the log is of the form $\langle u, i, r_{ui} \rangle$, where r_{ui} is the rating that the user u has given to the item i. For a system that has preference relations as feedbacks, the entries are of the form $\langle u, i, j, \pi(u, i, j) \rangle$. Here $\pi(u, i, j)$ denotes the preference relation between the ordered item pair (i, j)

for the user u. If users provide the values of the preference relations, then the values of $\pi(u,i,j)$ are readily available. When only absolute ratings are available, one can induce relative ratings information from it by setting $\pi(u,i,j)$ using the following rule:

$$\pi(u,i,j) = \begin{cases} 0 & \text{if } r_{ui} < r_{uj}, \\ 0.5 & \text{if } r_{ui} = r_{uj}, \\ 1 & \text{if } r_{ui} > r_{uj}, \\ undefined & \text{if either } r_{ui} \text{ or } r_{uj} \text{ are not available.} \end{cases} \tag{1}$$

4 Description of the Algorithm

We have developed a preference relation based matrix factorization algorithm (**PrefNMF-RP**) for the task of rating prediction. An algorithm for preference relation based matrix factorization for item recommendation has been proposed in [3]. The algorithm presented in this paper uses the same framework for modeling the users and the items, but uses the learned model in a different way as it has to output predicted ratings that conform to the integer rating scale used by the underlying recommender system.

The algorithm proposed in this paper represents each user u as a d-dimensional feature vector $p_u \in \mathbb{R}^d$. Similarly, each item i is also represented as a d-dimensional vector $q_i \in \mathbb{R}^d$. The d dimensional representation of the item can be viewed as the item's belongingness into the d hidden categories mined from the data. The d-dimensional vector representation of the user denotes the user's interests in each of these d hidden categories. User's interest in a particular item is estimated by the inner product of the corresponding user and item vectors. The value of d is often chosen beforehand and is much lesser than the number of items (m) and the number of users (n). The user and item vectors are learned from the data in the training phase which is performed offline. Personalized ratings of the items for a test user are generally computed online.

The proposed algorithm proceeds in two phases. The first phase models the users and the items in a low dimensional latent feature space. The second phase finds the unknown ratings using the feature representations of the users and the items. A brief discussion of the two phases is provided in the following subsections.

4.1 Phase 1: Modeling user and item features

The first phase of the proposed algorithm PrefNMF-RP models the users and items in a low dimensional latent feature space. A brief overview of this phase is given below.

Let U be the set of users and I be the set of items. S is the set of preference relations from the training set. Each entry in S is of the form $\langle u, i, j, \pi(u,i,j) \rangle$. It means that for user $u \in U$, the strength of the actual preference relation for the item pair $(i, j) \in I \times I$ is given by $\pi(u, i, j)$. If the user feedback comprises of absolute ratings instead of preference relations, then underlying preference relations can be induced from absolute ratings by using Equation 1.

Given a user u and an item pair (i, j), the strength of the preference relation may be computed as $p_u(q_i - q_j)^T$. However, this value may fall in any arbitrary range. To normalize this value in a range between 0 and 1, we model the strength of the relation using the *inverse-logit* function. So, we define the predicted preference relation for the triplet (u, i, j) as

$$\hat{\pi}(u, i, j) \overset{def}{=} \frac{e^{p_u(q_i - q_j)^T}}{1 + e^{p_u(q_i - q_j)^T}}. \tag{2}$$

For the triplet (u, i, j), the error in prediction is given by: $\frac{1}{2}(\pi(u, i, j) - \hat{\pi}(u, i, j))^2$. Therefore, the total error for the training set can be computed as

$$\mathcal{E} = \sum_{\substack{\langle u,i,j,\pi(u,i,j)\rangle \\ \in S \\ \wedge (i<j)}} \frac{1}{2}(\pi(u, i, j) - \hat{\pi}(u, i, j))^2. \tag{3}$$

The model learning phase aims to minimize this prediction error (given in Equation 3) on the training set. However, in order to avoid overfitting, a regularization term $\mathcal{R}(p, q)$ term is added with this error term, and resultant objective function is of the form:

$$f(p, q) = \frac{1}{2} \sum_{\substack{\langle u,i,j,\pi(u,i,j)\rangle \\ \in S \\ \wedge (i<j)}} (\pi(u, i, j) - \hat{\pi}(u, i, j))^2 + \lambda_p \sum_{u \in U} ||p_u||^2 + \lambda_q \sum_{i \in I} ||q_i||^2. \tag{4}$$

Here, $\mathcal{R}(p, q) = \lambda_p \sum_{u \in U} ||p_u||^2 + \lambda_q \sum_{i \in I} ||q_i||^2$ is the regularization term. λ_p and λ_q are constants. Hence, given a training set S, the model learning phase boils down to solving the optimization function given in Equation 4. Output of this phase are the matrices $p^{|U| \times d}$ and $q^{|I| \times d}$ containing the user and item features respectively.

4.2 Phase 2: Rating Prediction:

The second phase of the algorithm predicts the ratings of the items that are not already rated by the target users. Different recommender systems have different rating scales from which the users have to choose the ratings. Some systems allow the users to give ratings in a scale of 5, some system allow a maximum rating of 10 and so on. Hence, it is expected that the predicted ratings also should be in the rating scale defined by the system.

To find the predicted rating of an item i for the target user u, we first find a personalized *utility* value of the item i for u. This utility value may not be in a fixed range as followed by the recommender systems. So the algorithm has to map these utility values to the different rating levels as dictated by the recommender system.

Estimating Utility: We now consider different ways of measuring the utility of an item for a user. For a user u and the item pair (i, j), the (unnormalized) strength of the preference relation may be predicted as $p_u(q_i - q_j)^T$. Following a similar line of thought as explained in [3], the utility of an item i for the user u, denoted as $x(u, i)$, may be estimated as the item's total preference over all the items in the system. Mathematically,

$$x(u, i) = \sum_{j \in I \setminus i} p_u(q_i - q_j)^T.$$

However, if we use $x(u, i)$ for measuring the item utilities, then we have to first compute the values of $p_u(q_i - q_j)^T$ for each $j \in I \setminus i$. As a result, finding $x(u, i)$ has a time complexity of $O(d|I|)$. Here, d is the number of latent features. Since there are typically many items in the set I, this cost is very high, and may not be suitable for online processing.

To reduce the cost of this operation, we estimate $x(u, i)$ by a function $y(u, i) = p_u q_i^T$. It can be shown that $x(u, i) = c_1 y(u, i) + c_2$, where c_1 and c_2 are constants (independent of u and i). Therefore, the ordering of items produced by the utility functions $x(u, i)$ and $y(u, i)$ are same. i.e. given two items i_1 and i_2, $x(u, i_1) > x(u, i_2) \Leftrightarrow y(u, i_1) > y(u, i_2)$. Complexity of computing $y(u, i)$ is $O(d)$. Hence, $y(u, i)$ has the same expressive power as $x(u, i)$, and can be computed in lesser amount of time.

Mapping utilities to ratings: Once we estimate the utilities using $x(u, i)$ or $y(u, i)$, we need to map the utility values to the appropriate rating scale. One possible way of achieving this is by mapping the utilities to the different possible rating values. However, different users have different levels of leniency while rating items. There are some users who are strict and tend to assign low ratings even for the items they liked very much. On the other hand, there are users who are lenient and assign high ratings even to the items that they have moderately liked. Hence, a set of global mapping parameters that is common for all the users may not be suitable for the rating prediction task. In order to address this issue, we find the parameters of the mapping function for each user separately and use that for finding the unknown ratings.

To obtain these personalized parameters, we look at the ratings already entered by the users to different items. Suppose u has rated l different items $I_u = \{i_1, i_2, \cdots i_l\}$, and the corresponding ratings are $R_u = \{r_{u1}, r_{u2}, \cdots, r_{rl}\}$. We model the mapping by using a linear function given below:

$$r_{u,i_k} = \alpha_u y(u, i_k) + \beta_u. \tag{5}$$

The parameters α_u and β_u can be learned from the training data by solving the following least square objective function:

$$\min_{\alpha_u, \beta_u} \sum_{k=1}^{l} \left(r_{u,i_k} - \alpha_u y(u, i_k) - \beta_u\right)^2. \tag{6}$$

The values α_u and β_u may be stored in the system. The rating that user u would give to a test item j is predicted as $\hat{r}_{u,j} = \alpha_u y(u,j) + \beta_u$. Please note that the mapping can be done using higher order functions also. However, we performed some small experiments and found that linear mapping does better than higher order mappings, and hence we describe only linear mapping here and use the same for our experimental evaluations. Since $y(u,j)$ can be computed in $O(d)$ time and the mapping from $y(u,j)$ to $\hat{r}_{uj}$ can be done in $O(1)$ time (as α_u and β_u are stored in the system), the time complexity of the rating prediction phase is $O(d)$ for each unknown rating, which is reasonably fast for online processing.

5 Experimental Results

5.1 Dataset Used

We use two different samples of the Netflix dataset[1] for experimentation. The first dataset (D1) was created by considering 1500 movies from the Netflix challenge data. The second dataset (D2) was created by considering the next 1500 movies. Some ratings were eliminated so as to have a dataset where each user has rated minimum l_u and maximum r_u movies. The values of l_u and l_r were set to be different for the two datasets to create data with different sparsities. For both the datasets, the data was sorted according to rating timestamps. First 75% of this sorted data were used for training, and the remaining 25% were used for testing. However, as the proposed algorithm has to learn the mapping from the utilities to the bounded rating scale from the training set, we select for test only those users who have rated at least 20 items in the training set.

A brief statistics of the datasets is given below.

Dataset	D1	D2
Number of Ratings	124,637	485,333
Users	3229	22920
Items	1255	1232
Sparsity	96.9%	98.2%
Minimum Number of Ratings by any User	20	10
Maximum Number of Ratings by any User	449	455
Average Number of Ratings by any User	38	21
Minimum Number of Ratings for any Item	1	16
Maximum Number of Ratings for any Item	652	2879
Average Number of Ratings for any Item	99	394

Table 1. Statistics of the datasets

[1] http://www.netflixprize.com/

5.2 Algorithms Compared

We compared the performance of the rating prediction algorithm with five different algorithms from the literature. A brief overview of the algorithms are given below. The first four of them are based on Collaborative Filtering (CF) framework and the remaining one is based on the matrix factorization framework.

1. **CF with User Based Pearson Correlation (PC-CF):** This is a neighborhood based collaborative filtering method introduced in [4]. This is a standard baseline used in several papers proposing rating prediction algorithms. It computes the user similarity weights using Pearson Correlation between the rating values provided by the users. For a given target user u, top-K users according to the similarity weights are considered as u's *neighbors*. Weighted average of the ratings provided to the test item by the neighbors is output as the predicted rating.
2. **CF with Somers' Coefficient (Som-CF):** Use of Somers' coefficient for measuring the similarities between two users was suggested in [12]. Somers' Coefficient between two users is computed as $w = \frac{C-D}{N-T}$. Here, C, D and T are the numbers of concordant, discordant, tied pairs respectively. N is the number of items in the dataset. Once the weights are determined, the unknown ratings can be predicted as it is done in PC-CF.
3. **CF with Preference Relations (Pref-CF):** This is another algorithm that uses relative ratings for measuring user similarities [11]. Similarity weights between two users is determined by computing the fraction of item pairs for which the relative preferences are same for those two users. Predicted rating is calculated by the weighted average technique as used by PC-CF and Som-CF.
4. **CF with Aggregation of Preference Graphs (Pref-GrAgg):** This is a preference relations based algorithm and was proposed in [1]. It views the rating profile of each user as a preference graph. User similarities are measured based on the similarities between their preference graphs. The similarity weights are then used to generate an aggregate preference graph for the target user. Predicted ratings for items are determined by minimizing the total weight of the back edges in the aggregate graph.
5. **Standard Non-negative Matrix Factorization (NMF):** This is a matrix factorization based algorithm that learns the user and item vectors by minimizing a regularized objective function. The objective function represents the prediction error on the training dataset where the predicted rating is computed as the inner product between the corresponding user and item vectors. Description of the method and its working details can be found in [14, 15].

For all the CF based methods, we first select a shortlist of users who have a high number of co-rated items with the target user. We then assign similarity weights to only those users and select the K users with the highest weights as the neighbors.

5.3 Evaluation Metrics

Mean Absolute Error (MAE) and Root Mean Square (RMSE) were used as evaluation metrics. If r_{ui} is the actual rating given by the user u to item i and $\hat{r}_{ui}$ denotes predicted rating, then the metrics MAE and RMSE are defined as:

$$MAE = \frac{\sum_{(u,i)\in\mathcal{T}} |\hat{r}_{ui} - r_{ui}|}{|\mathcal{T}|}$$

and

$$RMSE = \sqrt{\frac{\sum_{(u,i)\in\mathcal{T}} (\hat{r}_{ui} - r_{ui})^2}{|\mathcal{T}|}}.$$

$\mathcal{T}$ is the set of test instances.

5.4 Results and Discussions

Performance Comparison: We compared the performances of PrefNMF-RP with that of the algorithms mentioned in Section 5.2. For PC-CF, Som-CF, Pref-GrAgg and Pref-CF, we experimented with different sizes of the neighborhood: 40, 60 and 80. For each of these algorithms, we report the result of the experiments that achieved the best values of the evaluation metrics. We ran both NMF and PrefNMF-RP algorithms using 40, 60, 80, 100, 120, 150, 200 and 250 features. For NMF and PrefNMF-RP, for each feature size, we run the algorithms 10 times and report the average values of the metrics of those 10 experiments.

The values of MAE and RMSE for the datasets D1 and D2 are shown in Table 2 and Table 3 respectively. For both the datasets and both the evaluation metrics, the PrefNMF-RP algorithm proposed in this paper achieves the best performance. Pref-GrAgg is the closest competitor for both D1 and D2.

Algorithm	MAE	RMSE
PC-CF	1.0765	1.5543
Som-CF	1.2068	1.6678
Pref-CF	1.0579	1.4783
Pref-GrAgg	0.7650	1.0850
NMF	0.8085	1.1278
PrefNMF-RP	**0.7199**	**1.0505**

Table 2. Comparison on dataset D1

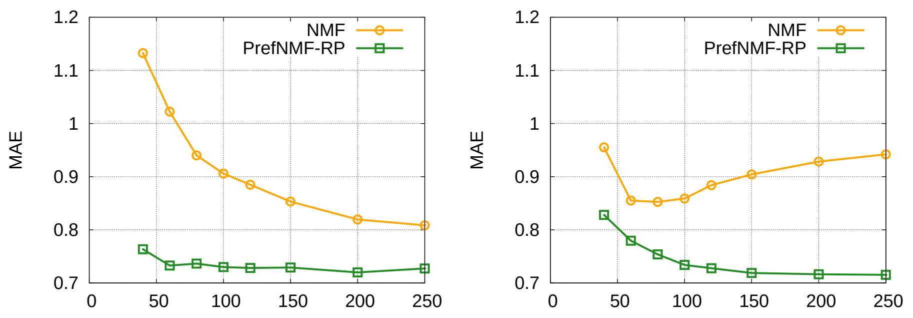
Algorithm	MAE	RMSE
PC-CF	0.9602	1.4001
Som-CF	1.0898	1.5300
Pref-CF	0.9759	1.3665
Pref-GrAgg	0.7623	1.0738
NMF	0.8525	1.1832
PrefNMF-RP	**0.7153**	**1.0267**

Table 3. Comparison on dataset D2

It can be observed from the results that although D2 is sparser than D1, PC-CF, Som-CF and Pref-CF methods perform much better on D2 than on D1. This is because the average number of ratings for an item is more in D2. As a result, it becomes easier to find the set of neighbors for the users. Also, while computing the predicted rating using the weighted average approach, as more number of users are likely to have rated the item, the algorithm can be more confident about the predicted output. On the other hand, Pref-GrAgg uses a graph based approach for finding the ratings from the aggregate graph. If the rating for a heavily-rated item can be predicted confidently by Pref-GrAgg, then that confidence can be passed through the preference edges to other items which are not rated by many users. Hence it does very well even when the dataset is very sparse as in the case of D2.

For the matrix factorization based algorithms, sparsity does affect the performance. For PrefNMF-RP, the results are almost similar for both the datasets. For NMF, The result worsens for D2, but is still much better than the CF based methods. There are newer versions of actual ratings based NMF that use different rating bias terms for the users and items [15]. However, we have not explored those variants of the NMF. We used the basic NMF method so that we have a baseline algorithm parallel with the proposed PrefNMF-RP algorithm.

Effect of the number of features: Here we analyze the effect of the number of features on the prediction accuracies of PrefNMF-RP and NMF. The MAE values of the algorithms obtained by experimenting with different numbers of features are shown in Figure 1. As the number of features is increased, generally the performances of the matrix factorization algorithms improve as they are able to capture different dimensions or aspects about the data. However, after reaching a certain number of features, the performance either saturates or starts to degrade due to overfitting. Overfitting generally happens faster if the data is very sparse.



(a) Dataset D1 (b) Dataset D2

Fig. 1. Changes in MAE with number of features

This trend is clearly visible from the results shown in Figure 1. For dataset D2, the performance of NMF deteriorates after 80 features. This might be due to the reason that as the dataset is very sparse, when NMF attempts to learn more about the data by using higher number of features, the learned model becomes very much tuned for the training data and overfitting occurs. However, the performance of PrefNMF-RP does not degrade or saturate that early for this dataset. This is due to the reason that PrefNMF-RP is able to use more information from the data by using preference relations. If many users rate an item higher than another item, then it is a direct evidence that the former item will have higher utility than the later item, for almost all the users. Also, if user u has rated n_u number of items in the training set, then the preference relations based methods have $\frac{n_u(n_u-1)}{2}$ observations which they can use for learning, as opposed to n_u observations for actual rating based methods. Due to these factors, probably PrefNMF-RP is able to learn the model better as the number of features is increased. As a result, degradation or saturation in performance does not happen that early. In our experiments, performance of PrefNMF-RP improved till 200 features, and then saturated. For dataset D1, which is denser than dataset D2, performance of both the algorithms improve till 200 features. After that, the changes are negligible. Comparison of RMSE values with the number of features show a similar pattern and hence is not shown separately.

6 Conclusions

In this paper, we have proposed a preference relations based matrix factorization algorithm for the rating prediction problem. The algorithm considers the preference relations or relative ratings provided to the items by the different users in the system. It then uses this information to map the users and the items in a joint latent feature space. Utility of an item for a user is estimated by the inner product between the feature vectors for the corresponding user and the item. Higher value of the utility indicate better satisfaction for the user. We then scale this utility to obtain a personalized rating for the item that the user would possibly have assigned to it. As the users have individual rating biases, the scaling parameters are determined separately for each individual user.

Two different samples of different sparsity configurations were used for experimentation. From the experimental results, we see that the proposed algorithm performed best among the different range of methods that we used for comparisons. The results indicate that the use of the preference relations instead of actual ratings can help in achieving better recommendations for the users.

Acknowledgements

Work of the first author is supported by a PhD Fellowship from Microsoft Research India.

References

[1] M. S. Desarkar, S. Sarkar, and P. Mitra, “Aggregating preference graphs for collaborative rating prediction,” in *Proceedings of the fourth ACM conference on Recommender systems*, RecSys ’10, pp. 21–28, ACM, 2010.

[2] N. Jones, A. Brun, and A. Boyer, “Comparisons instead of ratings: Towards more stable preferences,” in *Proceedings of the 2011 IEEE/WIC/ACM International Conferences on Web Intelligence and Intelligent Agent Technology - Volume 01*, WI-IAT ’11, pp. 451–456, IEEE Computer Society, 2011.

[3] M. S. Desarkar, R. Saxena, and S. Sarkar, “Preference relation based matrix factorization for recommender systems,” in *Proceedings of the 20th international conference on Advances in User Modeling*, UMAP ’12, Springer-Verlag, 2012.

[4] J. S. Breese, D. Heckerman, and C. Kadie, “Empirical analysis of predictive algorithms for collaborative filtering,” in *Proceedings of the Fourteenth conference on Uncertainty in artificial intelligence*, UAI’98, pp. 43–52, Morgan Kaufmann Publishers Inc., 1998.

[5] G.-R. Xue, C. Lin, Q. Yang, W. Xi, H.-J. Zeng, Y. Yu, and Z. Chen, “Scalable collaborative filtering using cluster-based smoothing,” in *Proceedings of the 28th annual international ACM SIGIR conference on Research and development in information retrieval*, SIGIR ’05, pp. 114–121, ACM, 2005.

[6] J. Wang, A. P. de Vries, and M. J. T. Reinders, “Unified relevance models for rating prediction in collaborative filtering,” *ACM Trans. Inf. Syst.*, vol. 26, pp. 16:1–16:42, June 2008.

[7] H. Luo, C. Niu, R. Shen, and C. Ullrich, “A collaborative filtering framework based on both local user similarity and global user similarity,” *Mach. Learn.*, vol. 72, pp. 231–245, Sept. 2008.

[8] D. H. Stern, R. Herbrich, and T. Graepel, “Matchbox: large scale online bayesian recommendations,” in *Proceedings of the 18th international conference on World wide web*, WWW ’09, pp. 111–120, ACM, 2009.

[9] M. Harvey, M. J. Carman, I. Ruthven, and F. Crestani, “Bayesian latent variable models for collaborative item rating prediction,” in *Proceedings of the 20th ACM international conference on Information and knowledge management*, CIKM ’11, pp. 699–708, ACM, 2011.

[10] J. Yoo and S. Choi, “Bayesian matrix co-factorization: variational algorithm and cramer-rao bound,” in *Proceedings of the 2011 European conference on Machine learning and knowledge discovery in databases - Volume Part III*, ECML PKDD’11, pp. 537–552, Springer-Verlag, 2011.

[11] A. Brun, A. Hamad, O. Buffet, and A. Boyer, “Towards preference relations in recommender systems,” in *Preference Learning (PL-10) ECML/PKDD-10 Workshop*, 2010.

[12] N. Lathia, S. Hailes, and L. Capra, “Private distributed collaborative filtering using estimated concordance measures,” in *Proceedings of the 2007 ACM conference on Recommender systems*, RecSys ’07, pp. 1–8, ACM, 2007.

[13] S. Rendle, C. Freudenthaler, Z. Gantner, and L. Schmidt-Thieme, “Bpr: Bayesian personalized ranking from implicit feedback,” in *UAI*, pp. 452–461, 2009.

[14] S. Funk, “Netflix update: Try this at home.” `http://sifter.org/~simon/journal/20061211.html`, December 2006. [Online; accessed 02-July-2012].

[15] Y. Koren, R. Bell, and C. Volinsky, “Matrix factorization techniques for recommender systems,” *Computer*, vol. 42, pp. 30–37, Aug. 2009.