Iterative Smoothing Technique for Improving Stability of Recommender Systems

Gediminas Adomavicius
University of Minnesota
gedas@umn.edu

Jingjing Zhang
Indiana University
jjzhang@indiana.edu

ABSTRACT

We focus on the measure of recommendation stability, which reflects the consistency of recommender system predictions. Stability is a desired property of recommendation algorithms and has important implications on users' trust and acceptance of recommendations. Prior research has reported that some popular recommendation algorithms can suffer from a high degree of instability. In this study we propose a scalable, general-purpose iterative smoothing approach that can be used in conjunction with different traditional recommendation algorithms to improve their stability. Our experimental results on real-world rating data demonstrate that the proposed approach can achieve substantially higher stability as compared to the original recommendation algorithms. Importantly, the proposed approach not only does not sacrifice the predictive accuracy in order to improve recommendation stability, but is actually able to provide additional accuracy improvements at the same time.

1. INTRODUCTION

Recommender systems represent technologies that assist users in finding a set of interesting or relevant items [1]. In order to provide good recommendations, recommender systems employ users' feedback on consumed items. This input can include explicitly provided feedback in the form of ratings or tags, as well as feedback that can be implicitly inferred by monitoring users' behavior such as browsing, linking, or buying patterns. The most common approach to modeling users' preferences for items is via numeric *ratings*. The recommendation algorithm then analyzes patterns of users' past ratings and predicts users' preference ratings for new, not yet consumed items. Once ratings for the new items are estimated, the item(s) with the highest estimated rating(s) can be recommended to the user.

In the recommender systems literature, evaluating performance of recommendation algorithms has always been a key issue, and recommendation accuracy has been the major focus in developing evaluation metrics [11,23]. As a result, much of the research in the recommender systems area has focused on proposing new techniques to enhance the *accuracy* of recommendation algorithms in predicting what users will like, as exemplified by the recent $1M Netflix prize competition. Prediction accuracy metrics typically compare the rating values estimated by a recommendation algorithm against the actual rating values and reflect the closeness of the system's predictions to users' true ratings. In addition to recommendation accuracy, researchers have proposed a number of alternative types of measures, including recommendation coverage, diversity, novelty, serendipity, and several others, to evaluate the performance of recommender systems [11,23]. Of special interest to us is the recently introduced measure of recommendation *stability* [2], which reflects the level of consistency among the predictions made by the system.

Copyright is held by the author/owner(s). Workshop on Recommendation Utility Evaluation: Beyond RMSE (RUE 2012), held in conjunction with ACM RecSys 2012. September 9, 2012, Dublin, Ireland.

According to the definition, stability is the consistent agreement of predictions made on the same items by the same algorithm, when any new incoming ratings are in complete agreement to system's prior estimations [2]. As has been discussed in prior work, stability is an important and desired property of recommender systems, and has a number of potential implications related to users' trust and acceptance of such systems [2].

While providing stable and consistent recommendations is important in many contexts, prior research has demonstrated that some popular collaborative filtering recommendation algorithms can suffer from high degree of instability [2]. This is particularly true for the widely used item- and user-based nearest-neighbor collaborative filtering approaches. It has also been shown that stability does not necessarily correlate with predictive accuracy [2], i.e., different recommendation algorithms can exhibit different levels of stability, even though they may have similar prediction accuracy. Thus, maximizing accuracy may not necessarily help to improve stability, and vice versa. For instance, a simple heuristic that predicts any unknown user rating as an average of all known ratings of that user is perfectly stable [2]; however, in most real-world settings this heuristic is outperformed by more sophisticated recommendation algorithms in terms of predictive accuracy. Therefore, the main objective of this study is to develop an approach that can improve stability of recommendation algorithms without sacrificing their accuracy.

In this paper, we propose a general *iterative smoothing* approach to improve stability of any given recommendation technique. The approach serves as a *meta-algorithm*, i.e., it can be used in conjunction with any traditional recommendation technique. Accordingly, the paper evaluates the performance of the proposed approach in conjunction with a number of popular and widely-used recommendation algorithms in terms of their stability as well as accuracy on several real-world movie rating datasets. The results show that this meta-algorithmic approach provides substantial improvements in recommendation stability as compared to the original recommendation algorithms, while providing some additional accuracy benefits as well.

2. RELATED WORK

Based on how unknown ratings are predicted, recommendation techniques can be classified into three general categories: content-based, collaborative filtering, and hybrid [1,3]. Among different recommendation approaches, collaborative filtering techniques have been most widely used, largely because they are domain independent, require minimal, if any, information about user and item features, yet can still achieve accurate predictions [13,19].

In a typical setting of collaborative filtering recommender systems, users' preferences for items are modeled via numeric ratings. Thus, the recommendation problem is reduced to the problem of estimating ratings for the items that have not been seen by a user, and this estimation is usually based on the other available ratings given by this and/or other users. More formally, given a set of users U and a set of items I, the entire user-item

space is denoted as $S = U \times I$. Let R_{ui} represent rating that user u gave to item i, where R_{ui} is typically known only for a limited subset of all possible (u,i) pairs. Let D be the set of known ratings, and $S\backslash D$ be the set of unknown ratings. Therefore, the recommendation task is to estimate unknown R_{ui} values for all $(u,i) \in S\backslash D$ pairs, given U, I, and known $R(u,i)$ values for $(u,i) \in D$.

As mentioned earlier, predictive accuracy has been a major focus in recommender systems literature. One of the most widely used predictive accuracy metrics for recommender systems is *root mean squared error* (RMSE), which we will use in this paper to report the recommendation accuracy results. However, there is a growing understanding that good recommendation accuracy alone may not give users a satisfying experience using the recommender systems, and that some other (complementary) measures are also important in evaluating the effectiveness of the system [23]. Our focus in this paper is one of these important complementary measures, recommendation *stability* [2].

Recommendation stability measures the inherent consistency among the different predictions made by the system. Consider an example where the system makes predictions for two movies, i_1 and i_2, for user u. Let's denote the two rating predictions as $R^*(u,i_1)$ and $R^*(u,i_2)$. Let's assume that prediction $R^*(u,i_1)$ is precisely accurate and, after user u consumes item i_1, it gets added to the system as part of the known ratings, i.e., $R(u,i_1) = R^*(u,i_1)$. The recommendation algorithm re-computes all other predictions in light of the new data. Would this change the value of the other prediction, $R^*(u,i_2)$, and to what extent, even though the newly incoming rating data was exactly as predicted by the system? In other words, two predictions $R^*(u,i_1)$ and $R^*(u,i_2)$ of the same recommender system can be viewed as "inconsistent" with each other, if adding one of them to the training data for the system changes the other prediction. As discussed in [2], the degree of the change in predictions reflects the (in)stability of the recommendation algorithm.

Specifically, the stability of a recommender system is defined as the extent to which the system's predictions for the same items stay the same/similar (i.e., are stable), when any and all new incoming ratings submitted to the system are in complete agreement with system's prior predictions. Based on this idea, a two-phase approach (illustrated in Figure 1) for computing stability of a recommendation algorithm has been introduced in prior literature [2]. In Phase 1, given a set of known ratings, a predictive model is built, and predictions for all unknown ratings are made. Then, a random subset of system's predictions is added to the original set of known ratings as hypothetical new incoming ratings. In Phase 2, based on expanded training data, a second predictive model is built using the same recommendation technique, and predictions on remaining unknown ratings are made. Stability is then measured by comparing the two sets of predictions to compute their root mean squared difference, which is called *root mean squared shift* (RMSS), and is computed in a similar fashion as RMSE:

$$\mathrm{RMSS} = \sqrt{\sum_{(u,i)\in P_1 \cap P_2} (P_1(u,i) - P_2(u,i))^2 \;/\; |P_1 \cap P_2|}$$

where P_1 and P_2 are the predictions made in Phase 1 and 2, respectively, i.e., RMSS captures the shift in predictions made by the same recommendation algorithm with new ratings that are in complete agreement with algorithm's own prior estimations.

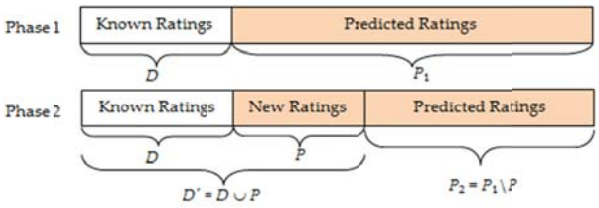

Figure 1. Illustration of stability computation, adapted from [2].

Providing consistent predictions has important implications on users' trust and acceptance of recommendations. In the consumer psychology literature, it has been shown that online advice-giving agents with greater fluctuations in past opinions are considered less informative than those with a more uniform pattern of opinions [9], and advice inconsistent with past recommendations is considered less helpful than consistent advice [7,24]. Inconsistent recommendations may be discredited by users and, as a result, the decrease in users' trust will further reduce their perceptions of the recommender system's competence [12,17]. In contexts where consistency is vital, the instability of a recommender system is likely to have a negative impact on users' acceptance and, thus, harm the success of the system.

Prior research has provided a comprehensive investigation into the stability of popular recommendation algorithms [2] and showed that some widely used algorithms can be highly unstable. For example, it has been shown that RMSS for user-based collaborative filtering approach can be as high as 0.47 on the Netflix movie rating data, meaning that on average *every* predicted rating will shift by 0.47 stars (i.e., by about a half-star) after adding to the training data some new ratings that are identical to the system's current predictions [2]. This is a very significant shift in prediction, considering the length of the rating scale for the dataset is only 4 stars (i.e., ratings go from 1 to 5). Thus, techniques for stability improvement are needed. In this paper, we propose a general-purpose meta-algorithmic approach that can be used to improve stability of traditional recommendation algorithms.

3. ITERATIVE SMOOTHING APPROACH

3.1 General Idea

High instability results from predictions that are inconsistent with each other. We propose an iterative smoothing approach, which involves multiple iterations for repeatedly and collectively adjusting the rating predictions of a recommendation algorithm based on its other predictions and, thus, is explicitly aimed at improving consistency of predicted ratings. The key idea of iterative smoothing is that the rating predictions computed during current iteration will be fed back into the data to predict other ratings in subsequent iterations.

Figure 2 provides an overview of the proposed iterative smoothing algorithm, and Figure 3 gives a high-level illustration of the overall process. Given rating space S and training set D where ratings are known, predictions on unknown ratings $S\backslash D$ are first estimated using some standard recommendation algorithm T. These predictions are denoted as P_0. The procedure of iterative smoothing then starts to iteratively adjust estimations for each rating in $S\backslash D$ based on all other ratings in the rating space S (i.e., both known as well as predicted) in order to proactively improve consistency between different predicted ratings.

More specifically, during the k-th iteration (for any k = 1, 2, …), for each unknown user-item pair $(u,i) \in S\backslash D$, a model $f_{k,u,i}$ is built based on training dataset $D_{k,u,i}$. This dataset is constructed to contain all known ratings combined with all predictions on user-

item pairs in S\D computed in the previous (k-1) iteration, except for the prediction on (u,i) as indicated in Figure 2. As the result of the k-th iteration, each model $f_{k,u,i}$ produces a new prediction for (u,i) that is stored as $P_k(u,i)$, i.e.:

$$P_k(u,t) = \begin{cases} R(u,i), & for\ (u,i) \in D \\ f_{k,u,i}(u,i), & for\ (u,i) \in S\backslash D \end{cases}$$

Here $R(u,i)$ represents the known rating that user u gave item i, and $f_{k,u,i}$ is the prediction model built based on $D_{k,u,i}$. Figure 4 illustrates the process within each iteration of the iterative smoothing approach. The set of predictions made in the current iteration is compared with predictions made in the previous iteration to compute the deviation between the two sets (measured in root mean squared difference). The iterative smoothing process stops either after a fixed, pre-determined number of iterations or when predictions on unknown ratings do not change.

By definition, in each iteration, a separate model is built for every user-item pair with unknown rating. Thus, in total, $|S\backslash D|\cdot K$ models need to be constructed over the course of K iterations, each using $|S|-1$ ratings as a training dataset. If $t(x)$ is the time needed to build a predictive model on data sample of size x using recommendation algorithm T, then the time complexity of the iterative smoothing approach is $O(|S\backslash D|\cdot K\cdot t(|S|))$.

Iterative Smoothing Algorithm:

Inputs: known ratings data D, # of iterations K, algorithm T

Process:

1. Build model f_0 on known ratings D using some standard recommendation algorithm T, i.e., $f_0 \leftarrow T(D)$
2. Apply model f_0 to compute predictions P_0 for unknown ratings $S\backslash D$, i.e., $P_0(u,i) = f_0(u,i)$ for $(u,i) \in S\backslash D$
3. For each iteration $k \in \{1, \ldots, K\}$
   For each unknown rating pair $(u, i) \in S\backslash D$
   - **a.** Construct dataset $D_{k,u,i}$ by including all known ratings D and all predicted ratings P_{k-1} from the previous iteration, except for rating $P_{k-1}(u,i)$, i.e.,
     $$D_{k,u,i} = D \cup P_{k-1} \backslash \{P_{k-1}(u,i)\}$$
   - **b.** Build model $f_{k,u,i}$ on dataset $D_{k,u,i}$ using T, i.e.,
     $$f_{k,u,i} \leftarrow T(D_{k,u,i})$$
   - **c.** Make prediction on (u, i) and store in P_k, i.e.,
     $$P_k(u, i) = f_{k,u,i}(u, i)$$
4. Output predictions made in the final iteration P_K

Output: P_K

Figure 2. Iterative smoothing approach.

In other words, the complexity of iterative smoothing algorithm is proportional to the size of unknown ratings in the rating space. This computational requirement is likely to be prohibitively expensive for many real-world scenarios, i.e., anytime the user-item rating space is large and rating data is sparse. For example, the Movielens 100K dataset is a publicly available movie rating dataset [10], which is considered to be relatively small. This dataset contains 100,000 movie ratings from 943 users on 1682 movies. Thus, the total number of possible ratings is about 1.6M (i.e., 1682 x 943), of which 1.5M is unknown. If we apply iterative smoothing algorithm on the Movielens 100K dataset, in total 1.5 million predictive models (each built on 1.6 million ratings) would need to be constructed within each iteration. Therefore, applying the full iterative smoothing approach may not be feasible even for datasets that are not very large. In order to overcome this complexity issue, in the next section we propose a variation of the iterative smoothing algorithm that offers significant scalability improvements while retaining most of the stability benefits (as will be shown in the Results section).

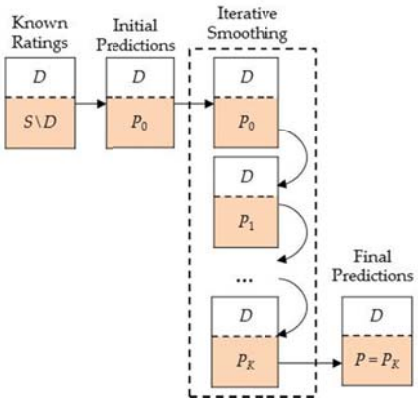

Figure 3. Illustration of the general iterative smoothing process.

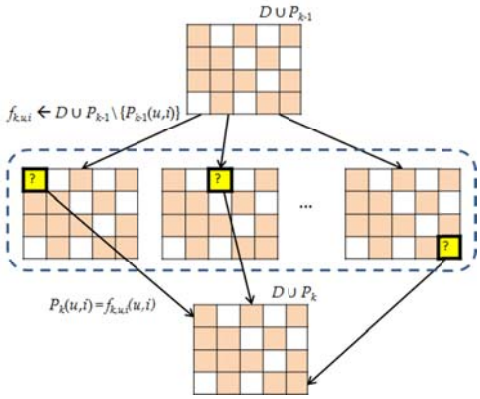

Figure 4. Illustration of one smoothing iteration.

3.2 Scalable Iterative Smoothing

We propose a variation of iterative smoothing approach that substantially simplifies the original approach and significantly reduces its computational requirements.

Figure 5 provides an overview of the proposed scalable iterative smoothing algorithm. Similarly to the original iterative smoothing approach, the proposed scalable version also involves multiple iterations for repeatedly adjusting estimations for each unknown rating. However, in each iteration, instead of building one model for each unknown rating, *only one single* model is built upon all known ratings and predicted ratings in previous iteration.

Specifically, during k-th iteration (for any k = 1, 2, …), model f_k is built based on training dataset D_k which contains all known ratings D combined with all predictions P_{k-1} on user-item pairs in $S\backslash D$ computed in the previous iteration. $P_k(u,i)$ denotes the predicted rating for (u,i) in the k iteration by the collective inference approach, defined as follows:

$$P_k(u,t) = \begin{cases} R(u,i), & for(u,i) \in D \\ f_k(u,i), & for\ (u,i) \in S\backslash D \end{cases}$$

Here $R(u,i)$ represents the known rating that user u gave item i. As the result of the k-th iteration, for each $(u,i) \in S\backslash D$, the model f_k produces a new prediction for (u,i) that is stored as $P_k(u,i)$. Similarly to the original approach, new predictions are compared with predictions made in the previous iteration and the procedure ends either after a fixed number of iterations or when predictions on unknown ratings converge (i.e., do not change).

The key difference between this simplified variation and the original iterative smoothing algorithm is the number of predictive models built within each iteration k. The original algorithm builds a separate model for every unknown rating in the rating space, in order to properly adjust the predicted rating $P_k(u,i)$ using *all other*

ratings from previous iteration, i.e., all ratings from D as well all ratings $P_{k-1}(u',i')$ where $(u',i') \neq (u,i)$.

Scalable Iterative Smoothing Algorithm:

Inputs: known ratings data D, # of iterations K, algorithm T

Process:

1. Build model f_0 on known ratings D using some standard recommendation algorithm T, i.e., $f_0 \leftarrow T(D)$
2. Apply model f_0 to compute predictions P_0 for unknown ratings $S\backslash D$, i.e., $P_0(u,i) = f_0(u,i)$ for $(u,i) \in S\backslash D$
3. For each iteration $k \in \{1, \ldots, K\}$
   a. Construct dataset D_k by including all known ratings D and all predicted ratings P_{k-1} from the previous iteration, i.e.,
   $$D_k = D \cup P_{k-1}$$
   b. Build model f_k on dataset D_k using T, i.e.,
   $$f_k \leftarrow T(D_k)$$
   c. For each unknown rating pair $(u, i) \in S\backslash D$, make prediction on (u, i) and store in P_k, i.e.,
   $$P_k(u, i) = f_k(u, i)$$
4. Output predictions made in the final iteration P_K

Output: P_K

Figure 5. Overview of scalable iterative smoothing approach.

In contrast, the simplified algorithm builds only one predictive model in each iteration, based on the entire rating matrix. In other words, predicted rating $P_k(u,i)$ is adjusted using *all* ratings from previous iteration, i.e., all ratings from D as well as from P_{k-1}, including $P_{k-1}(u,i)$. Thus, in the simplified algorithm, for any given rating prediction $P_1(u,i)$ in the first iteration, the predictive model is built on a rating data (i.e., D_1) that only differs from the rating data used in the original algorithm by one additional rating (i.e., $D_1 \backslash \{P_0(u,i)\}$). Because the influence of one additional rating is often subtle, especially when entire rating space is large (i.e., in settings with large numbers of users and items), the single overall model build in the simplified algorithm should produce outcomes similar to the ones produced by individual models built in the original algorithm, especially in the first iteration. While the difference between the original and simplified versions of the iterative smoothing may slowly increase as the number of iterations grows, the simplified approach still provides significant performance improvements (both in stability and accuracy), as demonstrated by the experimental results later in the paper.

Moreover, the runtime complexity of the simplified algorithm is much lower, making it much more practical from the scalability perspective. In particular, as only one overall model is built on all available ratings (i.e., dataset of size $|S|$) within each iteration, in total K models are constructed over the course of K iterations. Thus, the time complexity of the simplified variation is $O(K \cdot t(|S|))$. Comparing this to the complexity of the original algorithm, $O(|S\backslash D| \cdot K \cdot t(|S|))$, the scalable heuristic offers huge computational improvements (i.e., by roughly $|S\backslash D|$ times). In this paper, we use scalable iterative smoothing in our experiments.

4. EXPERIMENTAL RESULTS

4.1 Overall Process

Our experiments test the stability improvements achieved by the proposed meta-algorithmic approach in conjunction with several popular collaborative filtering techniques.

The experiments follow the two-phase stability computation from prior literature [2], discussed in Section 2. We used the standard train-test data splitting approach and divided known ratings data D into two sets: training data D_T (80%) and validation data D_V (20%), where $D = D_T \cup D_V$ and $D_T \cap D_V = \varnothing$. Training set D_T was used for building rating prediction models, while validation set D_V was reserved exclusively for evaluating the predictive accuracy of the final predictions. Similarly, a randomly chosen half of the unknown rating space E_T was dedicated for the stability evaluation of rating prediction models during the training phase, and the other half of the unknown rating space E_V was reserved exclusively for proper evaluation of the stability of the final predictions. Here, $S\backslash D = E_T \cup E_V$ and $E_T \cap E_V = \varnothing$.

Step 0: Create training and test datasets, i.e., D_T, D_V, E_T, E_V.

Step 1: Find the best model parameters:

i. Use a portion (e.g., 75%) of training set D_T to build rating prediction models using iterative smoothing.
ii. Compute model accuracy on other portion (25%) of D_T.
iii. Compute model stability on predictions made on E_T.
iv. Repeat steps i-iii with various parameter settings for iterative smoothing (i.e., number of iterations).
v. Find the best parameters for iterative smoothing, i.e., the specifications that result in best stability and accuracy.

Step 2: Evaluate accuracy and stability of the final predictions

i. Use the best parameter settings for iterative smoothing.
ii. Train the system on the entire training set D_T using iterative smoothing.
iii. Evaluate predictive accuracy on the reserved validation rating set D_V.
iv. Evaluate recommendation stability on the reserved unknown rating space E_V.

Step 3: Report parameter setting(s) as well as the accuracy and stability of the final predictions.

Figure 6. Overall experimental process.

Additionally, the process of iterative smoothing involves multiple iterations to adjust the predictions of unknown ratings. One of the goals in this study is to find whether predictions converge during the process of iterative smoothing and, if so, when. In addition, there is possibility that iterative smoothing models can "over-adjust" rating predictions after a number of iterations, in their attempt to maximize the performance on training data. Over-fitting is a well-known phenomenon which occurs when a predictive model is fine-tuned to fit the training data (including the random errors, outliers, and noise in the data) too well, which typically leads to diminished predictive performance on test data.

Therefore, for a given algorithm, it is necessary to find the best number of iterations to use on a given dataset in the final iterative smoothing procedure. In order to find the optimal parameter settings, we further used the standard train-test data splitting approach internally *within* the training data to identify the best parameter values for the proposed approach, i.e., that result in best accuracy and stability. The overall experiment process is summarized in Figure 6.

4.2 Recommendation Algorithms

In our experiments, we test the proposed approach in conjunction with four popular recommendation algorithms: the simple baseline method, classic user- and item-based variations of neighborhood-based CF approaches, and the matrix factorization technique. A brief overview of each technique is provided below.

Baseline. In real-world settings, some users may systematically tend to give higher ratings than others, and some universally liked items might receive higher ratings than others. Without normalization, such user and item effects could bias system's predictions. Hence, recommender systems often involve a pre-processing step to remove these "global effects". One common practice is to estimate and remove three effects: the overall mean, the main effect of an item, and the main effect of a user [4]. Such

"global effects" can serve as a *baseline estimate* for unknown rating of corresponding user and item, i.e.,

$$b_{ui} = \mu + b_u + b_i ,$$

where μ is the overall average rating, b_u is the average observed deviation from μ on ratings provided by user u, and b_i is the average observed deviation from μ on ratings given to item i. Note that, in all of our experiments (i.e., with all other recommendation algorithms), the ratings data were normalized by removing these global effects. Moreover, this estimate is often used as a baseline recommendation technique for comparison with other recommendation algorithms, i.e., $R^*(u,i) = b_{ui}$, and we investigate its performance in our experiments as well.

User-Based Collaborative Filtering (CF_User). The user-based nearest-neighbor collaborative filtering approach is a heuristic that makes predictions of unknown ratings for a user based on the ratings previously rated by this user's "nearest neighbors", i.e., other users who have similar rating patterns [6,20]. That is, the value of the unknown rating for user u and item i is usually computed as an aggregate of the neighbors' ratings for the same item i. The most common aggregation approach is the weighted sum of the neighbors' ratings, where the similarity of two users is used as a weight. I.e., the more similar user u' and target user u are, the more weight will be carried by the rating provided by user u' on item i in the weighted sum when computing the prediction. Predicted rating for user u on item i is computed as:

$$R^*(u,i) = b_{ui} + \frac{\sum_{v \in N(u,i)} sim_{uv} * (R(v,i) - b_{vi})}{\sum_{v \in N(u,i)} |sim_{uv}|}$$

where $N(u,i)$ is a set of "neighbors" with similar rating patterns to user u and that have provided ratings for item i, sim_{uv} is the similarity between users u and v, and b_{ui} is the baseline estimate for user u on item i. In our implementation, two users must have rated at least 3 items in common to allow computation of similarity between them. The similarity between two users is calculated as Pearson correlation between rating vectors (based on the commonly rated items) of the two users. Prediction of each unknown rating is formulated by combining the preferences of 20 most similar users who have rated the same item.

Item-Based Collaborative Filtering (CF_Item). The user-based collaborative filtering technique also has an analogous item-based version, where the ratings of the nearest-neighbor items are used to predict unknown ratings for a given item. Several studies have presented empirical evidence that item-based algorithms often provide better predictive accuracy than user-based methods (e.g., [22]). Thus, our experiments also test the standard item-based collaborative filtering in conjunction with the proposed approach. Similarly to the settings employed in user-based CF, in our experiments, two items are required to have been rated by 3 common users to allow similarity evaluation between them, and 20 nearest-neighbor items are used to formulate a prediction.

Matrix factorization (SVD). Matrix factorization technique is a model-based (as opposed to heuristic-based) collaborative filtering approach that characterizes items and users via a number of latent factors inferred from known ratings [8,15]. This technique models the $U \times I$ rating space as a product of two sub-matrices: user preference matrix ($U \times L$) and item feature matrix ($L \times I$). Each user and item is described by a vector of L latent variables. In our experiments L is set to be 20. The user vector indicates the preference of the user for several latent features, and the item vector represents an item's importance weights for the same latent features. *Singular value decomposition* (SVD) techniques are used to decompose original rating matrix into the two sub-matrices in an optimal way that minimizes the resulting approximation error. After the two sub-matrices are learned using known ratings, each unknown rating is estimated as a dot-product of the corresponding user- and item-factors vectors. Many variations of matrix factorization techniques have been developed during the recent Netflix Prize competition (e.g., [14,15,18,21]). Our experiments focus on the basic underlying version of the matrix factorization [8]; however, the proposed meta-algorithmic approach can be applied with any variation of this technique.

4.3 Results: Comparing Iterative Smoothing with Standard Recommendation Techniques

The objective of the experiment is to compare the performance of the proposed iterative smoothing approach with standard single-model recommendation techniques on several real world datasets.

The first dataset we used is the Movielens 100K dataset [10], which contains 100,000 known ratings on 1682 movies from 943 users (6.3% data density). Our second dataset is a sample extracted from the Movielens 1M dataset. The original Movielens 1M dataset consists of 1,000,000 ratings for 6040 movies by 3952 users (4.2% data density) [10]. From this dataset we extracted a random sample of 3000 users and 3000 movies. Resulted dataset contains 400,627 known ratings (i.e., 4.45% data density). Our third dataset is sampled from the Netflix 100M dataset used in the recent Netflix Prize competition [5]. Similarly to the second dataset, we sub-sampled 3000 random users and 3000 random movies from the original data file. The result data sample consists of 105,256 known ratings (i.e., 1.17% data density). The three datasets used in our experiments come from different sources and have different data characteristics (i.e., size and sparsity). All movie ratings in the Movielens and Netflix datasets are integer values between 1 and 5, where 1 represents the least liked movies, and 5 represents the most liked movies. The datasets used in this experiment are summarized in Table 1.

Table 1. Summary of Experimental Datasets.

DataSet	Description	Users	Items	Density
Movielens 100K	Movie ratings from Movielens movie recommender system.	943	1682	6.30%
Movielens 1M		3000	3000	4.45%
Netflix	Movie ratings distributed by Netflix company.	3000	3000	1.17%

The procedure of this experiment followed the general experimental process described in Figure 6. We examined the prediction accuracy and stability of the final predictions on the reserved validation datasets (as described in Figure 6, Step 2). For each recommendation algorithm used in our study, we compare two approaches: the standard (original) single-model approach and the scalable version of iterative smoothing approach. Accuracy and stability numbers (measured by RMSE and RMSS) of the two approaches on real-world movie rating datasets are provided in Table 2.

Experimental results are consistent across different datasets. On all three datasets, our proposed meta-algorithmic iterative smoothing approach outperformed the original recommendation techniques in both stability and accuracy in the vast majority of cases. In particular, on average (i.e., across all datasets), iterative smoothing provided a dramatic 55% improvement over the original recommendation algorithms in stability (as measured by RMSS) for CF_User and CF_Item algorithms. Even for the fairly stable SVD and baseline techniques, on average, iterative smoothing was able to further improve RMSS by 14%. In terms of predictive accuracy, on average, iterative smoothing provided

1.4% improvements in RMSE across different algorithms.

Table 2. Iterative Smoothing vs. Standard Techniques.

	Method	Approach	Accuracy (RMSE)	Stability (RMSS)
Movielens 100K	SVD	Standard	0.9437	0.0866
		Smoothing	0.9378	0.0779
	CF_User	Standard	0.9684	0.4023
		Smoothing	0.9586	0.2020
	CF_Item	Standard	0.9560	0.3234
		Smoothing	0.9320	0.1347
	Baseline	Standard	0.9758	0.1148
		Smoothing	0.9609	0.0569
Movielens 1M	SVD	Standard	0.8846	0.0804
		Smoothing	0.8798	0.0719
	CF_User	Standard	0.9393	0.3294
		Smoothing	0.9182	0.1145
	CF_Item	Standard	0.9135	0.2755
		Smoothing	0.8929	0.1089
	Baseline	Standard	0.9425	0.0910
		Smoothing	0.9346	0.0923
Netflix	SVD	Standard	0.9372	0.1208
		Smoothing	0.9363	0.1137
	CF_User	Standard	0.9608	0.4610
		Smoothing	0.9407	0.2462
	CF_Item	Standard	0.9579	0.4394
		Smoothing	0.9381	0.2291
	Baseline	Standard	0.9622	0.1465
		Smoothing	0.9556	0.1351

5. CONCLUSIONS AND FUTURE WORK

This paper introduces a general-purpose, practical meta-algorithmic approach – based on iterative smoothing – for improving stability of a variety of traditional recommendation algorithms. The iterative smoothing approach uses multiple iterations to repeatedly and explicitly adjust predictions of a recommendation algorithm based on its other predictions in order to make them more consistent with each other. We examined the performance of iterative smoothing approach on several real world datasets. Our experiments show that the proposed approach demonstrates effectiveness in their ability to improve stability for several widely used recommendation algorithms. Perhaps as importantly, the proposed approach not only does not sacrifice the predictive accuracy to obtain these stability improvements, but actually is able to provide some additional accuracy improvements at the same time.

This work provides several interesting directions for future research. This study shows that iterative smoothing can improve stability for different recommendation algorithms, providing larger improvements for some algorithms and smaller improvements for some others. Providing some additional theoretical understanding of what algorithmic and data characteristics can lead to larger vs. smaller improvements in recommendation stability for the proposed approach is an important direction for future work. Another interesting direction would be to perform user behavior studies to investigate the value of stable (i.e., as opposed to unstable) recommendations on users' usage patterns and acceptance of recommender systems.

ACKNOWLEDGMENTS

This research was supported in part by the National Science Foundation grant IIS-0546443.

REFERENCES

[1] Adomavicius, G., and Tuzhilin, A. 2005. "Toward the Next Generation of Recommendation Systems: A Survey of the State-of-the-Art and Possible Extensions," *IEEE Transactions on Knowledge and Data Engineering*, 17, (6), 734-749.

[2] Adomavicius, G., and Zhang, J. 2010. "On the Stability of Recommendation Algorithms," *ACM Conference on Recommender systems*, Barcelona, Spain: ACM New York, NY, USA 47-54.

[3] Balabanovic, M., and Shoham, Y. 1997. "Fab: Content-Based, Collaborative Recommendation," *Comm. of ACM*, 40, (3), 66-72.

[4] Bell, R.M., and Koren, Y. 2007. "Scalable Collaborative Filtering with Jointly Derived Neighborhood Interpolation Weights " *Seventh IEEE International Conference on Data Mining*, Omaha, NE, USA.

[5] Bennett, J., and Lanning, S. 2007. "The Netflix Prize," *KDD-Cup and Workshop*, San Jose, CA.

[6] Breese, J.S., Heckerman, D., and Kadie, C. 1998. "Empirical Analysis of Predictive Algorithms for Collaborative Filtering," *14th Conf. on Uncertainty in Artificial Intelligence*, Madison, WI.

[7] D'astous, A., and Touil, N. 1999. "Consumer Evaluations of Movies on the Basis of Critics' Judgments," *Psychology & Marketing*, 16, 677–694.

[8] Funk, S. 2006. "Netflix Update: Try This at Home."

[9] Gershoff, A., Mukherjee, A., and Mukhopadhyay, A. 2003. "Consumer Acceptance of Online Agent Advice: Extremity and Positivity Effects," *J. of Consumer Psychology*, 13, (1&2), 161-170.

[10] Grouplens 2011. "Movielens Data Sets."

[11] Herlocker, J.L., Konstan, J.A., Terveen, K., and Riedl, J.T. 2004. "Evaluating Collaborative Filtering Recommender Systems," *ACM Transactions on Information Systems*, 22, (1), Jan, 5-53.

[12] Komiak, S., and Benbasat, I. 2006. "The Effects of Personalization and Familiarity on Trust and Adoption of Recommendation Agents," *MIS Quarterly*, 30, (4), 941-960.

[13] Koren, Y. 2009. "The Bellkor Solution to the Netflix Grand Prize." from http://www.netflixprize.com/assets/GrandPrize2009_BPC_BellKor.pdf.

[14] Koren, Y. 2008. "Factorization Meets the Neighborhood: A Multifaceted Collaborative Filtering Model," *ACM Intl. Conf. on Knowledge Discovery and Data Mining (KDD'08)*, 426-434.

[15] Koren, Y., Bell, R., and Volinsky, C. 2009. "Matrix Factorization Techniques for Recommender Systems," *IEEE Computer*, 42, 30-37.

[16] Macskassy, S.A., and Provost, F. 2007. "Classification in Networked Data: A Toolkit and a Univariate Case Study," *Journal of Machine Learning R*, 8, 935-983.

[17] O'Donovan, J., and Smyth, B. 2005. "Trust in Recommender Systems," *10th Intl. Conf. on Intelligent User Interfaces*.

[18] Paterek, A. 2007. "Improving Regularized Singular Value Decomposition for Collaborative Filtering," *KDDCup*, 39-42.

[19] Pilaszy, I., and Tikk, D. 2009. "Recommending New Movies: Even a Few Ratings Are More Valuable Than Metadata,"*Third ACM Conf. on Recommender systems (RecSys '09)*, 93-100.

[20] Resnick, P., Iacovou, N., Suchak, M., Bergstrom, P., Riedl, J. 1994. "Grouplens: An Open Architecture for Collaborative Filtering of Netnews.," *Conf. on Comp. Supported Cooperative Work*, 175–186.

[21] Salakhutdinov, R., and Mnih, A. 2008. "Bayesian Probabilistic Matrix Factorization Using Markov Chain Monte Carlo," *25th Intl. Conf. on Machine Learning*, Helsinki, Finland: ACM, 880-887.

[22] Sarwar, B., Karypis, G., Konstan, J.A., and Riedl, J. 2001. "Item-Based Collaborative Filtering Recommendation Algorithms," *10th International WWW Conference*, Hong Kong, 285 - 295.

[23] Shani, G., and Gunawardana, A. 2011. "Evaluating Recommendation Systems," in *Recommender Systems Handbook*, F. Ricci, L. Rokach, B. Shapira and P.B. Kantor (eds.). 257-294.

[24] Van Swol, L.M., and Sniezek, J.A. 2005. "Factors Affecting the Acceptance of Expert Advice," *British Journal of Social Psychology*, 44, (3), 443-461.