Hindawi Publishing Corporation
Mathematical Problems in Engineering
Volume 2015, Article ID 380472, 13 pages
http://dx.doi.org/10.1155/2015/380472

Research Article

Improving Top-N Recommendation Performance Using Missing Data

Xiangyu Zhao,[1,2] Zhendong Niu,[2] Kaiyi Wang,[1] Ke Niu,[2] and Zhongqiang Liu[1]

[1]*Beijing Research Center for Information Technology in Agriculture, Beijing 100097, China*
[2]*School of Computer Science and Technology, Beijing Institute of Technology, Beijing 100081, China*

Correspondence should be addressed to Xiangyu Zhao; zhaoxy@nercita.org.cn

Received 23 April 2015; Accepted 26 August 2015

Academic Editor: Jean-Charles Beugnot

Recommender systems become increasingly significant in solving the information explosion problem. Data sparse is a main challenge in this area. Massive unrated items constitute missing data with only a few observed ratings. Most studies consider missing data as unknown information and only use observed data to learn models and generate recommendations. However, data are missing not at random. Part of missing data is due to the fact that users choose not to rate them. This part of missing data is negative examples of user preferences. Utilizing this information is expected to leverage the performance of recommendation algorithms. Unfortunately, negative examples are mixed with unlabeled positive examples in missing data, and they are hard to be distinguished. In this paper, we propose three schemes to utilize the negative examples in missing data. The schemes are then adapted with SVD++, which is a state-of-the-art matrix factorization recommendation approach, to generate recommendations. Experimental results on two real datasets show that our proposed approaches gain better top-N performance than the baseline ones on both accuracy and diversity.

1. Introduction

In the current age of information overload, it is becoming increasingly hard for people to find relevant content. Recommender systems have been introduced to help people in retrieving potentially useful information in a huge set of choices. Conventional recommendation methods are based on users' rating values. These rating values are considered as indications of users' preference level towards the rated items. Recommender systems estimate the ratings of items that have not been rated by the target user based on the rating history and recommend top-N items with highest predicted ratings. This kind of rating prediction approaches has gain significant success. Recently, there is a growing interest in improving recommender systems in terms of ranking performance as it seems to better approximate the true task [1, 2]. As a result, some researchers consider the recommendation problem as a ranking prediction problem and directly optimize a ranking goal to learn their recommendation algorithms.

Most of these approaches, either rating prediction ones or ranking prediction ones, are trained and tested on observed ratings only. The effectiveness of these approaches is based on an implicit underlying assumption that the ratings in the available data are missing at random. If the assumption is not satisfied, the missing data mechanism cannot be ignored in general and has to be modeled precisely so as to obtain correct results. Indeed, some recent works find that the data are missing not at random [3–5]. Marlin et al. [3] provide evidence that low ratings are much more likely to be missing from the observed data than high ratings in the Yahoo!LaunchCast data. This may be a consequence of the fact that users are free to choose which items to rate. Steck [4] works on training and testing recommender systems on data missing not at random and illustrates that accounting for missing ratings can improve the top-N performance of simple matrix factorization model. Therefore, missing data, which have not been rated by the active users, carry useful information of user preferences.

We agree with the idea that data are missing not at random. However, our assumption is different with these ones in [3–5], which consider missing data mainly as negative ratings. In our opinion, there are lots of negative examples of users' preference in missing data while observed data are positive ones. We have found that the rating behaviors themselves are evidences of user preferences no matter whether the rating values are high or low in the previous work [6]. This indicated that users are free to choose which items to rate in the context of recommender systems. Therefore, lots of missing data are because users choose not to rate them. For example, one who dislikes horror film will not watch "Evil Dead" and also will not rate it. To a certain user, the entire item data can be split into two sets: one is the item set (c^+) that the user chooses to rate and the other is the item set (c^-) that the user does not deliberately choose to rate. Observed data are part of c^+. The rest of c^+ combines with c^- to be missing data. The goal of recommender systems is to identify the items in c^+ but not been rated yet and to recommend them to users. As a result, it is important to distinguish which item set an item belongs to. Unfortunately, there are only positive examples to be used for classification; negative examples are mixed with some positive ones in missing data.

An intuitive approach to solving this problem is to distinguish the negative examples and use them together with the positive ones to learn recommendation model. In this paper, we propose three kinds of schemes to get negative examples from missing data and adapt them with an existing recommendation approach to a unique model. It is expected that the new model can better distinguish between positive items and negative ones compared to the original model and recommend more items that users may rate. As a result, the new model is expected to gain improvement in top-N recommendation performance.

The first scheme considers that all missing data are negative examples with different confidence towards the positive ones. The other two sample some negative examples from missing data with a stochastic method or a neighbor-based method. All the schemes utilize the information from missing data. To verify the effectiveness, we adapt the schemes with SVD++ [7] (a state-of-the-art rating prediction approach using a matrix factorization model) to new models. Our experiments demonstrate that the new models gain significant improvement of SVD++ in top-N recommendation.

The remainder of the paper is organized as follows. We review related literature in Section 2. The schemes to deal with missing data are introduced in Section 3. The improvements of SVD++ are proposed in Section 4. Section 5 introduces some popular evaluation metrics. Experiments are carried out on MovieLens and EachMovie datasets in Section 6 to compare the proposed approaches with existing ones. Finally, we conclude the paper in Section 7.

2. Related Work

In this section, the review of literatures is divided into four parts. The first one is about conventional rating prediction recommendation algorithms. The second one includes some studies on ranking prediction recommendation approaches. The third one is about some recent works on nonrandom missing data. The last one focuses on one-class collaborative filtering, the idea which is similar to our proposed schemes.

2.1. Rating Prediction Approaches.

2.1. Rating Prediction Approaches. Recommendation techniques have been studied for several years. Conventional recommendation approaches are based on rating prediction. They are used for providing personalized recommendations to help people in solving the information explosion problem. Collaborative filtering (CF) is a very popular technique, since it is not necessary to analyze the content of the candidate items using swarm intelligence instead. Furthermore, it can be easily adapted from one domain to another. CF algorithms can be divided into two classes: memory-based and model-based [8, 9].

Memory-based algorithms are heuristic methods that make rating predictions based on the entire collection of items previously rated by users [10, 11]. They are based on a basic assumption that people who agreed about their preferences to certain items in the past tend to agree again in the future [12]. The level of agreement can be measured by similarity. Based on the similarity calculation, recommender systems predict ratings for unknown items using adjusted weighted sum of known ratings and recommend items with high predicted values [11].

Model-based CF is another kind of typical CF methods. Model-based algorithms use the collection of ratings to learn a model, typically using some statistical machine-learning methods, which are then used to make rating prediction. These approaches always design appropriate loss functions and optimization procedure to learn their model by minimizing the error between predicted ratings and actual ones. Examples of such techniques include Bayesian clustering [9], matrix factorization [7], and topic model [13].

SVD++ [7] is a model-based CF using matrix factorization technique. It considers implicit feedbacks as complement of explicit feedbacks and utilizes them together to build recommendation models by minimizing prediction errors. This approach is a state-of-the-art rating prediction approach, which is used as the foundation of our improvement.

2.2. Ranking Prediction Approaches. Different from those rating prediction approaches, some researches directly consider the recommendation problem as a ranking problem. They propose models for ranking predictions by directly modeling user preferences with respect to a set of items rather than the rating scores on individual items.

Weimer et al. [14] present a method (CofiRank) which uses Maximum Margin Matrix Factorization and considers maximum NDCG as the optimizing target. The approach is adaptable to different scores. Since the optimizing target of CofiRank is a listwise one, the approach scales well on collaborative filtering tasks.

Liu and Yang [15] measure the similarity between users based on the correlation between their rankings of the items rather than the rating values. Based on the preferences of similar users, they propose collaborative filtering algorithms for ranking items with either a greedy strategy or a random walk model.

Liu et al. [2] propose a probabilistic latent preference analysis (pLPA) model to make ranking predictions. From a user's observed ratings, they extract his/her preferences in the form of pairwise comparisons of items which are modeled by a mixture distribution based on Bradley-Terry model. An EM algorithm for fitting the corresponding latent class model as well as a method for predicting the optimal ranking is described.

Koren and Sill [16] propose a collaborative filtering recommendation framework (OrdRec), which is based on viewing user feedback on products as ordinal, rather than the more common numerical view. Their approach is based on a pointwise ordinal model, which allows it to linearly scale with data size. OrdRec is also an improvement of SVD++. It is used as a comparing approach in our experiments to verify the effectiveness of our proposed approaches in the top-N recommendation task.

2.3. Nonrandom Missing Data. Most of conventional collaborative filtering approaches use observed ratings only, and they expect that the model optimizing with observed ratings only is an unbiased estimating of using the entire data. These approaches are based on an implicit assumption that the ratings not in observed data are missing at random. However, this may not be satisfied. Some recent works have found that data are not missing at random [3–5].

Marlin et al. [3] find that low ratings are much more likely to be missing from observed data than high ratings in the Yahoo!LaunchCast data. This is an evidence of data missing not at random. Steck [4, 5] works on training and testing recommender systems on data missing not at random. He assumes that the relevant rating values are missing at random, and the other ratings are missing with higher probability. Based on the assumption, he presents two performance measures that can be estimated, under mild assumptions, without bias from data even when ratings are missing not at random. In addition, he also propose an appropriate surrogate measure for training models which is captured as AllRank. In this measure, both observed and missing data are considered. It improves the top-N performance of a simple matrix factorization model by accounting for missing ratings.

Cremonesi et al. [1] propose an improvement of matrix factorization by considering all missing values in the user rating matrix as 0, which is captured as PureSVD. This approach gets better top-N performance even than more detailed and sophisticated latent factor models. The result demonstrates that considering missing data as 0 value is much more effective than just ignoring them, which is also an evidence of data missing not at random.

2.4. One-Class Collaborative Filtering. In some recommendation context, the training data usually consist simply of binary data reflecting a user's action. Researchers consider these problems as one-class collaborative filtering problems (OCCF). In these problems, users' action data are usually extremely sparse (a small fraction are positive examples); therefore ambiguity arises in the interpretation of the non-positive examples. Negative examples and unlabeled positive examples are mixed together, and they always are unable to be distinguished. Pan et al. [17] propose two frameworks to solve the OCCF problems. One is based on weighted low rank approximation; the other is based on negative example sampling. Li et al. [18] exploit the rich user information to improve recommendation accuracy in the OCCF problems. They propose two ways to incorporate such user information into the OCCF models: one is to linearly combine scores from different sources and the other is to embed user information into collaborative filtering. Rendle et al. [19] consider missing data as a mixture of real negative feedback and missing positive values and present a generic optimization criterion (BPR) for personalized ranking that is the maximum posterior estimator derived from a Bayesian analysis of OCCF problem.

The schemes, which we will propose in the next section to deal with missing data of recommender systems by weighting or sampling, are similar to the idea in OCCF. However, the context of recommendation is different between our schemes and the schemes in OCCF. OCCF focuses on the binary recommendation problems with implicit feedbacks while our proposed ones focus on the classical recommendation problems with explicit feedbacks. Furthermore, the neighbor-based sampling scheme proposed in Section 3.3 can utilize the advance of kNN methods while sampling negative examples.

3. Schemes to Deal with Missing Data

In the context of recommender systems, users are free to choose which items to rate. As a result, the observed rating data can indicate users' preferences. In the survey of Marlin et al. [3] using Yahoo!LanchCast data, there are 93.9% users report that they rate an item which they love very often, while only 36.5% users report that they rate an item for which they are "neutral" with the same frequency. The survey is a collection of ratings for songs, which is a little time-consuming context. If the context changes to a very time-consuming or cost-consuming, such as movie or e-commerce, the ratio of users choosing to rate an item for which they are "neutral" should be less. Therefore, there are two types of items for a certain user. One is the items that the certain user wants to rate. They are partitioned to a set c^+. The other is the items that the user does not care and does not want to rate. They are partitioned to c^-. The observed data contains the rated items. It is a part of c^+. The rest part of c^+ combines with c^- to be missing data. In this paper, we consider the items in c^+ as positive examples, and the items in c^- as negative examples.

Based on the partition, Steck considers that the rating distribution is different between c^+ and c^- [5]. He tries to model the difference to improve a simple matrix factorization approach in top-N recommendation task [4]. In his opinion, the negative ratings with low value get high probabilities to be missing. Therefore, he imputes a small value (r_m) for all missing data, and uses a weighting parameter (w_m) to control the effect of missing data. In this way, the improved models using missing data can gain better top-N performance than the original matrix factorization model using observed data only (in the work, AllRank-Regression with $w_m = 0.05$ and

$r_m = 2$ gains the best top-N performance. It is used as a comparing approach in our experiment).

The main idea of Steck [4, 5] is that most of missing data are negative ratings. The difference between c^+ and c^- is rating distribution. Different from them, in our opinion, most of missing data are negative examples. Positive examples in c^+ and negative examples in c^- are two different item sets. The goal of the recommender systems is to identify the unrated positive examples. As a result, it is necessary to distinguish an item belong to which item set. Unfortunately, only positive examples are explicit in recommendation context, negative examples are mixed with some positive examples in missing data. In order to solve the problem, we try to distinguish the negative examples, and use them together with the positive ones to learn recommendation model. Like the idea in [4], we use an imputed value (r_m) for negative examples in order to model both positive and negative examples in one unique model. Different from Steck [4], our r_m is used to represent negative examples, which are actually in a different item set from positive examples. Therefore, the value of r_m should out of the range of rating scale in order to distinguish negative examples with positive ones using rating value (The typical value of r_m is 0. The impact of different r_m is experimented in Section 6 even with the value in the range of rating scale.).

In the rest of this section, three schemes are introduced to deal with missing data.

3.1. Weighting Scheme. Weighting Scheme (WS) considers that all missing data are negative examples with different confidence levels towards the positive ones. The weighting value indicates the confidence level, which determines how much missing data are considered as negative examples. The weighting function can be written as:

$$w_{ui} = \begin{cases} 1, & \langle u, i \rangle \in R \\ \delta, & \langle u, i \rangle \notin R, \end{cases} \tag{1}$$

where w_{ui} is the weighting value for user u on item i, R is the observed data, δ is a uniform confidence threshold for all missing data. If user u has rated item i, it is a positive example, and the weighting value is set to 1. Otherwise, $\langle u, i \rangle$ is considered as a negative example with a confidence level δ. In this scheme, all missing data are imputed with r_m. It can be formalized as:

$$r^*_{ui} = \begin{cases} r_{ui}, & \langle u, i \rangle \in R \\ r_m, & \langle u, i \rangle \notin R, \end{cases} \tag{2}$$

where r^*_{ui} is the data for learning recommendation models, and r_{ui} is the rating value in observed data.

With WS, a recommendation approach aims at finding a prediction model to minimizing the objective of a weighted Frobenius loss function as:

$$\mathscr{L}(R^*) = \sum_{\langle u,i \rangle} w_{ui} \left(r^*_{ui} - \hat{r}(u, i)\right)^2, \tag{3}$$

where R^* is the re-construct matrix which contains both observed data and imputed ratings, while $\hat{r}(u, i)$ is the rating predicted by recommender systems.

Broadly speaking, WS can be considered the same as AllRank-Regression in [4]. The main difference is that their opinion about missing data is negative ratings (AllRank-Regression) or negative examples (WS). In addition, PureSVD, which is proposed in [1] is a special case of using WS in SVD approach with $\delta = 1$.

3.2. Random Sampling Scheme. WS considers all missing data as negative examples. This assumption is roughly held in most cases. However, the main drawback is that the computational costs are very high especially when the target problem of recommender systems is information overload, which contains a massive set of missing data. Sampling scheme could solve this problem in a certain degree by considering some missing data as negative examples, which is much different from WS.

In this subsection, we propose a random sampling scheme (RSS) which samples some negative examples from missing data with a stochastic method. In RSS, θ percentage of missing data is randomly selected as negative examples (*RandomN*). These negative examples are combined with rating matrix R to be the re-construct matrix R^* for RSS. It can be formalized as:

$$r^*_{ui} = \begin{cases} r_{ui}, & \langle u, i \rangle \in R \\ r_m, & \langle u, i \rangle \in RandomN. \end{cases} \tag{4}$$

RSS uses R^* to optimize the recommendation model and generate recommendations. Therefore, the size of R^* is a major aspect of the computational cost for different recommendation approaches. As the size of R is a constant, the computational cost is mainly relevant to the size of *RandomN*. When θ is 1, *RandomN* is the entire set of missing data, the computational cost of RSS is similar to WS. When θ is 0, *RandomN* is an empty set, the computational cost of RSS is similar to the original recommendation approach which does not consider the effect of missing data. When θ is between 0 and 1, the computational cost of RSS is reduced with a decrease in θ. The experimental results will show that RSS gains the best performance when θ is 0.2. This indicates that the computational cost of RSS is much less than WS.

As RSS mainly focuses on utilizing missing data without improvement of training process of recommendation approaches. Therefore, RSS learns the prediction model by minimizing the objective of an unweighted Frobenius loss function as most recommendation approaches do. It can be written as:

$$\mathscr{L}(R^*) = \sum_{\langle u,i \rangle} \left(r^*_{ui} - \hat{r}(u, i)\right)^2. \tag{5}$$

It is notable that the *RandomN* should be re-built in each learning step, as the sampling scheme is a stochastic one, in order to reduce the randomness.

3.3. Neighbor-Based Sampling Scheme. Sampling scheme can reduce the computational costs of weighting scheme. However, using a stochastic method leads that both the missing positive examples and the negative ones have the same

```
Input: the rating matrix $R$, the random ratio $\theta$, the neighbor size $k$
Output: the neighbor-based sampling matrix $NeighborN$
(1) for each user $u \in U$ do
(2)     Find $N(u)$: the top-$k$ most similarity users of $u$;
(3)     Find $OUT(u)$: the item set, in which items have not been rated by user $u$;
(4)     Find $C(u)$: the candidate item set, a sub set of $OUT(u)$, in which items have not
        been rated by all users in $N(u)$;
(5)     Random select $\theta$ percentage of items in $C(u)$ into $NeighborN$;
(6) end for
```

Algorithm 1: The neighbor-based sampling scheme.

chance to be selected as negative examples. In this subsection, we propose a neighbor-based sampling scheme (NSS) to increase the chance of negative examples to be selected and to decrease the selected chance of positive ones. Different from RSS sampling with a stochastic method, NSS samples some negative examples from missing data using swarm intelligence.

NSS is based on an assumption that similar users have similar tendency about negative examples. Like the idea of neighbor-based CF, in NSS, for a certain user, items that have rarely been rated by his/her neighbors are very likely to be negative examples. As a result, NSS searches the k most similar users as neighbors for individual users and then selects the items which have not been rated by users' neighbors. In this case, negative examples have bigger chance to be selected than positive ones. After that, NSS randomly samples some items as negative examples from the selected items. The sampled result is a negative example set (*NeighborN*). The detail of NSS is described in Algorithm 1.

With the sampling result, the elements of the reconstruct matrix for NSS can be written as

$$r^*_{ui} = \begin{cases} r_{ui}, & \langle u,i\rangle \in R \\ r_m, & \langle u,i\rangle \in NeighborN. \end{cases} \tag{6}$$

Recommendation approaches with NSS learn their models with the same loss function as RSS. Since the similar users in NSS are used to share the opinions about which items (not) to rate, the similarity functions which consider two users who often (do not) rate the same items as similar users are suitable. Jaccard index is a such kind of similarity function, and it is used for measuring users' similarities in this paper.

4. Recommendation Approaches

The proposed schemes in Section 3 can be adapted with many recommendation approaches to utilize missing data. In this section, we take a matrix factorization approach which is known as SVD++ [7] as the basic model and adapt the schemes with it in order to improve its top-N recommendation performance by using missing data.

The SVD++ approach is demonstrated to yield superior accuracy by considering implicit feedbacks (including rating behaviors) as complement of explicit feedbacks and use them together to build recommendation models by minimizing prediction errors. The prediction model of SVD++ is

$$\hat{r}(u,i) = \mu + b_u + b_i + q_i^T \left(p_u + |N(u)|^{-1/2} \sum_{j \in N(u)} y_j \right), \tag{7}$$

where μ is the average rating value of the known data. In addition, b_u and b_i indicate the observed deviations of user u and item i, respectively, from the average. p_u and q_i are the factorized user and item factor, respectively. $N(u)$ represents the item set rated by user u. y_j is an item factor which is according to the impact of implicit feedbacks.

The approach learns the values of involved parameters with a stochastic gradient descent technique by minimizing the regularized squared error function. The loss function can be written as

$$\begin{aligned} \min_{p^*,q^*,b^*,y^*} \sum_{\langle u,i\rangle} \Bigg(r_{ui} - \mu - b_u - b_i \\ - q_i^T \left(p_u + |N(u)|^{-1/2} \sum_{j \in N(u)} y_j \right) \Bigg)^2 + \lambda_6 \left(b_u^2 + b_i^2 \right) \\ + \lambda_7 \left(\|q_i\|^2 + \|p_u\|^2 + \sum_{j \in N(u)} \|y_j\|^2 \right), \end{aligned} \tag{8}$$

where λ_6 and λ_7 are the regularizing terms. The learning process runs in the rating matrix R, which contains all the rating values proposed by users. The predicted ratings can be calculated by (7) using the learned parameters.

SVD++ learns parameters from R. It considers all missing data as unknown information and ignores them.

However, as mentioned above, missing data should not be ignored simply. Therefore, we adapt the proposed scheme with SVD++ to deal with missing data in order to improve the recommendation performance. The improvement approaches are introduced in the rest part of this section, respectively.

4.1. Improvement of SVD++ with Weighting Scheme. In this subsection, WS is adapted with SVD++ to deal with missing data. This approach is captured as WSVD++. In WSVD++, we just change the learning process of SVD++ without any change of model structure. As a result, the prediction model of WSVD++ is consistent with SVD++, as shown in (7). On the other hand, the loss function should be converted to a weighted function as shown in (3). It can be written as

$$\min_{p^*,q^*,b^*,y^*} \sum_{\langle u,i\rangle} w_{ui} \left[\left(r^*_{ui} - \mu - b_u - b_i - q_i^T \left(p_u + |N(u)|^{-1/2} \sum_{j\in N(u)} y_i \right) \right)^2 + \lambda_6 \left(b_u^2 + b_i^2 \right) + \lambda_7 \left(\|q_i\|^2 + \|p_u\|^2 + \sum_{j\in N(u)} \|y_j\|^2 \right) \right]. \quad (9)$$

In WSVD++, the learning process runs through all the user-item pairs with different weighting values as defined in (1). Let us denote the prediction error, $r^*_{ui} - \hat{r}_{ui}$, by e_{ui}. WSVD++ loops through all the user-item pairs. For a given case r^*_{ui}, we modify the parameters by moving in the opposite direction of the gradient, yielding

(i) $b_u \leftarrow b_u + w_{ui} \cdot \gamma_1 \cdot (e_{ui} - \lambda_6 \cdot b_u)$,

(ii) $b_i \leftarrow b_i + w_{ui} \cdot \gamma_1 \cdot (e_{ui} - \lambda_6 \cdot b_i)$,

(iii) $q_i \leftarrow q_i + w_{ui} \cdot \gamma_2 \cdot (e_{ui} \cdot (p_u + |N(u)|^{-1/2} \sum_{j\in N(u)} y_j) - \lambda_7 \cdot q_i)$,

(iv) $p_u \leftarrow p_u + w_{ui} \cdot \gamma_2 \cdot (e_{ui} \cdot q_i - \lambda_7 \cdot p_u)$,

(v) $\forall j \in N(u) : y_j \leftarrow y_j + w_{ui} \cdot \gamma_2 \cdot (e_{ui} \cdot |N(u)|^{-1/2} \cdot q_i - \lambda_7 \cdot y_j)$.

In this way, the learned parameters can converge to minimize the loss function. However, b_u and b_i are two parameters which indicate user and item rating bias, respectively. They should not been influenced by missing data, which are not explicit feedbacks from users for items. In addition, y_j is an indicator of implicit feedbacks, and the negative examples are negative evidences for implicit feedbacks. It should not been influenced by missing data, either. Therefore, we use a different weighting function for these three parameters, which can be written as

$$w'_{ui} = \begin{cases} 1, & \langle u,i\rangle \in R \\ 0, & \langle u,i\rangle \notin R. \end{cases} \quad (10)$$

Correspondingly, the learning process can be revised to

(i) $b_u \leftarrow b_u + w'_{ui} \cdot \gamma_1 \cdot (e_{ui} - \lambda_6 \cdot b_u)$,

(ii) $b_i \leftarrow b_i + w'_{ui} \cdot \gamma_1 \cdot (e_{ui} - \lambda_6 \cdot b_i)$,

(iii) $q_i \leftarrow q_i + w_{ui} \cdot \gamma_2 \cdot (e_{ui} \cdot (p_u + |N(u)|^{-1/2} \sum_{j\in N(u)} y_j) - \lambda_7 \cdot q_i)$,

(iv) $p_u \leftarrow p_u + w_{ui} \cdot \gamma_2 \cdot (e_{ui} \cdot q_i - \lambda_7 \cdot p_u)$,

(v) $\forall j \in N(u) : y_j \leftarrow y_j + w_{ui}\prime \cdot \gamma_2 \cdot (e_{ui} \cdot |N(u)|^{-1/2} \cdot q_i - \lambda_7 \cdot y_j)$.

There is an important parameter (δ) in WSVD++. It determines the confidence of missing data to be negative examples. We will discuss the impact in Section 6.2.

4.2. Improvement of SVD++ with Sampling Scheme. In this subsection, the sampling scheme is adapted with SVD++ to deal with missing data. The scheme can be either RSS or NSS. Depending on the choosing scheme, the approach is captured as RSSVD++ (with RSS) or NSSVD++ (with NSS), respectively. The same as WSVD++, we only change the learning process of SVD++ without any changing of model structure. The prediction models of RSSVD++ and NSSVD++ are consistent with SVD++, since they are all unweighted functions. Furthermore, the loss functions are still consistent with SVD++.

In addition, the negative examples are not relevance to the parameter b_u, b_i, and y_j like WSVD++. Therefore, in order to guarantee the irrelevance, an indicator function is defined. It can be written as

$$I_{ui} = \begin{cases} 1, & \langle u,i\rangle \in R \\ 0, & \langle u,i\rangle \in R^* - R. \end{cases} \quad (11)$$

The learning process of RSSVD++ or NSSVD++ runs in the reconstruct matrix R^*. It can be defined as

(i) $b_u \leftarrow b_u + I_{ui} \cdot \gamma_1 \cdot (e_{ui} - \lambda_6 \cdot b_u)$,

(ii) $b_i \leftarrow b_i + I_{ui} \cdot \gamma_1 \cdot (e_{ui} - \lambda_6 \cdot b_i)$,

(iii) $q_i \leftarrow q_i + \gamma_2 \cdot (e_{ui} \cdot (p_u + |N(u)|^{-1/2} \sum_{j\in N(u)} y_j) - \lambda_7 \cdot q_i)$,

(iv) $p_u \leftarrow p_u + \gamma_2 \cdot (e_{ui} \cdot q_i - \lambda_7 \cdot p_u)$,

(v) $\forall j \in N(u) : y_j \leftarrow y_j + I_{ui} \cdot \gamma_2 \cdot (e_{ui} \cdot |N(u)|^{-1/2} \cdot q_i - \lambda_7 \cdot y_j)$.

Both RSSVD++ and NSSVD++ are sampling schemes. They share similar loss function and similar learning process. The main difference between them is the way in which they are sampling negative examples from missing data. There is an important parameter (θ) which is the ratio of the negative examples randomly selected from missing data in RSSVD++. The impact of it will be discussed in Section 6.3. For NSSVD++, there are two important parameters, for example, θ and k. The former one is the ratio of a negative example randomly selected from candidate item set. The latter one determines the size of the neighbors for users. We will discuss the impact of them in Section 6.4.

5. Evaluation Metrics

In this paper, we focus on top-N recommendation task, where a recommender system is trying to pick the best N items for people [1, 6, 8]. The Normalized Discounted Cumulative Gain (NDCG) [20] metric is a popular metric for evaluating the relevance of top-N results in information retrieval where the documents are assigned graded rather than binary relevance judgments. As the rating values can indicate the levels of users' preferences on items in recommender systems, the NDCG metric is suitable for evaluating recommendation quality in the top-N recommendation task. In this paper, we use NDCG as our main evaluation metric. It is an accuracy metric, which can be written as

$$\mathrm{DCG@}N(u) = \sum_{p=1}^{N} \frac{2^{R(u,p)} - 1}{\log(1+p)}, \tag{12}$$

$$\mathrm{NDCG@}N = \frac{1}{|U|} \sum_{u \in U} \frac{\mathrm{DCG@}N(u)}{\mathrm{IDCG@}N(u)}, \tag{13}$$

where $R(u,p)$ is the rating value of user u rating the item at the pth position of the recommendation list, $\mathrm{DCG@}N(u)$ represents the Discounted Cumulative Gain value at the Nth position for user u, and IDCG is the maximum possible DCG which is used for normalizing the NDCG value. However, in recommender systems, the recommended items are always unknown ones (the target user has not rated the items yet). Therefore, some researchers consider the recommendation problem as a ranking problem and use NDCG to evaluate the algorithms while ranking the items in the test set [2]. In this paper, we recognize it as NDCG+ metric and consider it to be a comparative evaluation metric. The main difference between these two metrics is that NDCG+ ranks the items contained in the test set of each user, whereas NDCG ranks all the possible items (all items except the ones that the current user has rated in the training set) for each user.

In addition, 1-call at top-N recommendations is used as another accuracy metric. It reflects the ratio of users who have at least one relevant item in their top-N recommendation lists [21]. This paper is in the context of data missing not at random. Recall is a popular evaluation metric in this context, as it can be estimated without bias from observed data, whether or not the relevant ratings are missing at random [5]. Recall is defined as

$$\mathrm{R@}N = \frac{1}{|U|} \sum_{u \in U} \frac{\mathrm{REL}(u,N)}{\mathrm{REL}(u)}, \tag{14}$$

where $\mathrm{REL}(u,N)$ is the number of relevant items among the top-N items for user u and $\mathrm{REL}(u)$ is the number of all relevant items for the user. As used in [4, 5], the relevant items are the items rated by the current user with value 5.

Furthermore, a number of recent studies find that beyond accuracy there are other quality factors, which are also important to users, for example, diversity and novelty [8, 22]. Diversity in recommender systems refers to how different the recommended items are from each other. It is an important complement of accuracy since a recommender system which recommends relevant items has little value to a user if the recommendation cannot expand his/her interest. Coverage is one of the most popular diversity metrics. It measures the percentage of items that an algorithm is able to recommend to users in the system. Denoting the total number of distinct items in top-N places of all recommendation lists as N_d, the N-dependent coverage is defined as

$$\mathrm{COV}(N) = \frac{N_d}{N}. \tag{15}$$

Furthermore, just recommending popular items is not sufficient for users, which is considered as lack of novelty. Therefore, the coverage of recommendations in the long tail of the items is also a significant evaluation metric, which indicates the novelty of recommendations to a certain degree. It can be written as

$$\mathrm{CIL}(N) = \frac{NL_d}{N}, \tag{16}$$

where NL_d represents the intersection of N_d and the long tail item set. In this paper, we consider that the long tail item set contains all the items which are not in the top 20% popular item set.

In summary, 6 evaluation metrics are used to evaluate our proposed approach. NDCG, 1-call, and Recall are used to evaluate the top-N recommendation quality. COV is used to evaluate the diversity of recommendations, whereas CIL is mainly for evaluating novelty. NDCG+ is a metric to evaluate the ranking prediction quality.

6. Experiment

6.1. Experiment Setup. The proposed recommendation approaches are evaluated on MovieLens and EachMovie datasets, which are both widely used in the field of recommender systems. MovieLens dataset, denoted as ML, contains 100 thousand ratings which are assigned by 943 users on 1682 movies. Collected ratings are in 1-to-5 scale. EachMovie dataset, denoted as EM, contains 2.8 million ratings from 72916 users on 1628 movies. The original ratings from EM are in 0-to-1 scale. In order to be consistent with ML, it is converted to 0-to-5 scale, and then ratings with 0 value are excluded. In addition, EM dataset is very sparse, some users only rate a few items, and some items are only rated by a few users. These data may reduce the recommendation quality. Therefore, we exclude the users who rated no more than 20 items and the items which are rated no more than 10 times.

We use 5-fold cross validation for the evaluation. Starting from the initial data set, five distinct splits of training and test data are generated. For each data split, 80% of the original set is included in the training data and 20% of it is included in the test data. Users' rating history in the training set is used to generate recommendations according to different algorithms. The test set is then used to evaluate the recommendation results. We further split the test set randomly into two disjoint sets of equal size. One of them is used to determine the tuning parameters. The other is for final evaluation of the trained model.

In order to demonstrate the effectiveness of our proposed approaches, we compare them with the original SVD++ approach and other benchmark approaches. User-based CF

(UserCF) and Slope-one are two classic rating prediction approaches. UserCF [11] is a classic neighbor-based CF approach, which is based on an assumption that users always like the items liked by similar users. Slope-one [23] is a memory-based CF approach based on average rating differential. The original SVD++ is also a rating prediction approach. OrdRec [16] is a ranking prediction approach based on a pointwise ordinal model. As an improvement of SVD++, it is used to compare the improvement level between using missing data and optimizing ranking performance. In addition, PureSVD [1] and AllRank-Regression [4] are two existing approaches dealing with missing data, which have been introduced in Section 2. They are both used for comparison. In order to minimize the impact of different original models, their idea of using missing data are adapted with SVD++, captured as Pure and AllRank, respectively. All of the above-mentioned approaches are evaluated by NDCG, Recall, 1-call, COV, and CIL, compared with NDCG+.

In addition, some approaches need user-specific parameters. The details of parameter assignments for different approaches are as follows: the size of nearest neighbors for UserCF is 50; SVD++, Pure, and AllRank have 50 features and 25 iteration steps with $\lambda_6 = \lambda_7 = 0.05$ and $\gamma_1 = \gamma_2 = 0.002$; the r_m and w_m for AllRank are 2 and 0.05, respectively, as suggested in [4]; OrdRec has 50 features and 60 iteration steps with $\lambda_6 = 0.0005$, $\lambda_7 = 0.0001$, $\gamma_1 = \gamma_2 = 0.05$, and $\gamma_3 = \gamma_4 = 0.006$. Our proposed approaches are improvements of SVD++. The parameters, which have been included in SVD++, are set the same value as in SVD++. Therefore, the effectiveness of the proposed approaches is irrelevance with the impact of these parameters. The impact of other new added parameters is analyzed in the next subsections.

6.2. Parameters of WSVD++. WSVD++ is an improvement of SVD++ with a weighting scheme. There are two new added parameters, for example, δ and r_m. One is δ, which determines the weight of the negative examples. The other is r_m, which is the imputed value of missing data. In order to analyze the impact of them, experiments are carried out to analyze the performances of WSVD++ with different values of δ and r_m. When one of them is analyzed, the other one remains unchanged. Here, we only use NDCG value, which is our main evaluation matric, to analyze the performances.

Figure 1 shows the performances of WSVD++ as a function of the value of δ with $r_m = 0$ on ML (the trends of performances on EM are similar. Therefore, we will not show them here. Furthermore, we use the same data in the experiments in this and the next two subsections while analyzing the performances of the proposed approaches with different parameters and only show the results on ML). The value of δ is changed from 0 to 1, step by 0.1.

When δ is 0, WSVD++ degenerates to SVD++. When δ is from 0.1 to 0.2, WSVD++ gets better performance than SVD++. This indicates that improving SVD++ with a weighting scheme could increase the capability to recommend relevance items. When δ is bigger than 0.2, the performance of WSVD++ is not better than SVD++. This indicates low confidence of missing data being negative examples. If the weighting of missing data as negative examples is considered

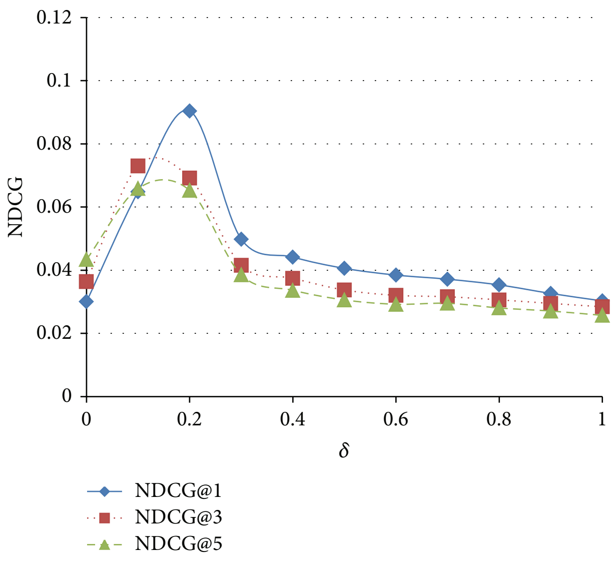

FIGURE 1: NDCG of WSVD++ as a function of the value of δ with $r_m = 0$ on ML.

too much, the performance may decrease. When δ is 1, WSVD++ can be considered as Pure. The performance of it is similar to original SVD++. When δ is 0.2, WSVD++ gets the best performances. As a result, the best value of δ is chosen as 0.2 in WSVD++.

Figure 2 shows the performance of WSVD++ as a function of the value of r_m with δ as 0.2, which is chosen as the best value. The value of r_m is changed from −5 to 5, step by 1. It is obvious that when r_m is less than 1 (out of the range of rating scale), the performance of WSVD++ is better than it with r_m between 1 and 5. The result verifies the effectiveness of using an imputed value out of rating scale to model negative examples with a weighting scheme. In addition, it demonstrates that considering missing value as negative examples is more appropriate than considering it as negative ratings. The best performance is located at $r_m = 0$. When the value of r_m is smaller than 0, the performance declines. It may be because of the low confidence of the weighted examples. Too small r_m introduces bias of the trained model.

6.3. Parameters of RSSVD++. RSSVD++ is an improvement of SVD++ with a random sampling scheme. There are two new added parameters, for example, θ and r_m. θ is the ratio of the negative examples randomly selected from missing data, while r_m is the imputed rating for missing data. In order to analyze the impact of them, experiments are carried out to analyze the performances of RSSVD++ with different values of θ and r_m, respectively. Similar to the above subsection, we only use NDCG value to analyze the performances.

Figure 3 shows the performances of RSSVD++ as a function of the value of θ with $r_m = 0$. The value of θ is changed from 0 to 1, step by 0.1.

When θ is 0, RSSVD++ degenerates to SVD++. When θ is from 0.1 to 0.2, RSSVD++ gets better performance than SVD++. This indicates that improving SVD++ with a random sampling scheme could increase the capability to

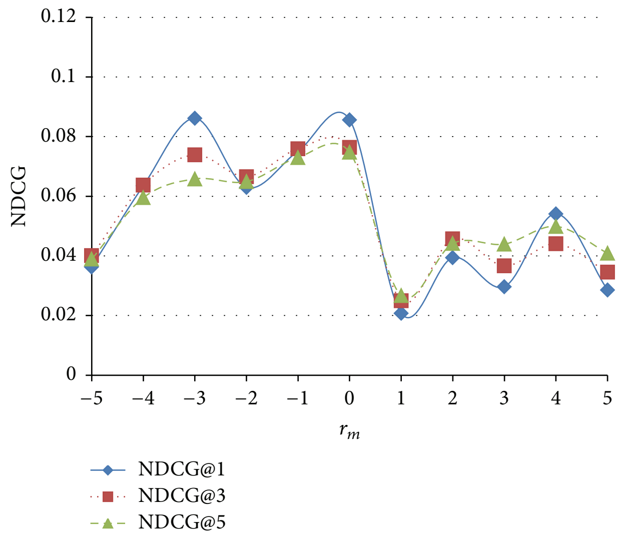

FIGURE 2: NDCG of WSVD++ as a function of the value of r_m with $\delta = 0.2$ on ML.

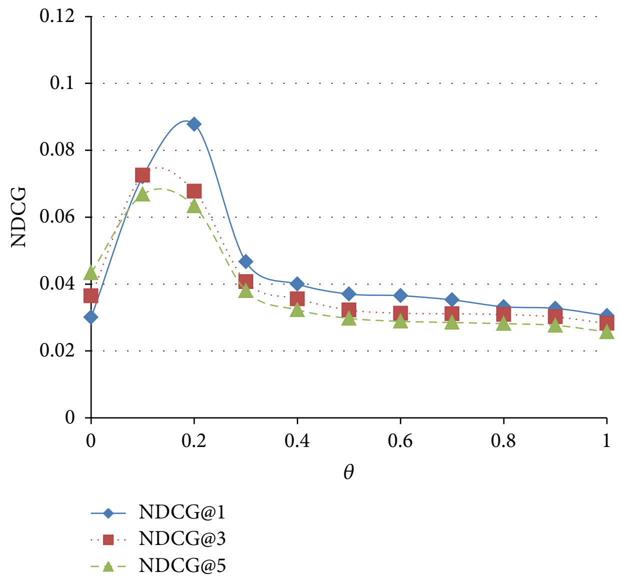

FIGURE 3: NDCG of RSSVD++ as a function of the value of θ with $r_m = 0$ on ML.

recommend relevance items. When θ is bigger than 0.2, the performance of RSSVD++ is not better than SVD++. This indicates that considering too much missing data as negative examples is not a good solution to improve recommendation quality, since some missing data are not negative examples. As a result, a small value of θ can lead RSSVD++ getting good performances. When θ is 0.2, RSSVD++ gets the best performances. Therefore, the value of θ is chosen as 0.2 in RSSVD++.

Figure 4 shows the performance of RSSVD++ as a function of the value of r_m with θ as 0.2. The value of r_m is changed from −5 to 5, step by 1. It can be found that when r_m is less than 1, the performance of RSSVD++ is better than it with r_m between 1 and 5. When $r_m = 0$, RSSVD++ gains

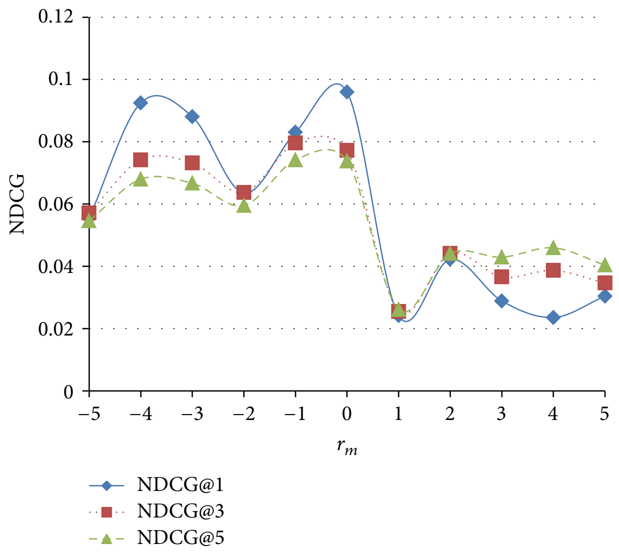

FIGURE 4: NDCG of RSSVD++ as a function of the value of r_m with $\theta = 0.2$ on ML.

the best performance. This verifies again the effectiveness of considering missing value as negative examples and using an imputed value out of rating scale to model the negative examples.

6.4. Parameters of NSSVD++. NSSVD++ is an improvement of SVD++ with a neighbor-based sampling scheme. There are three new added parameters of it, for example, k, θ, and r_m. k is the size of the nearest neighbor. Large k indicates big size of $N(u)$ and small size of $C(u)$, since more users have less unrated items. Therefore, the value of k determines the candidate item set. θ is the ratio of the negative example randomly selected from the candidate item set. The negative examples for NSSVD++ are determined by both k and θ. In order to analyze the impact of them, we use three typical values of k and carry out experiments to analyze the performances of NSSVD++ with different values of θ and certain value of k. r_m is the imputed value of missing data. The impact of it is analyzed with the optimized k and θ. Similar to the above subsections, we only use NDCG value to analyze the performances.

Figures 5–7 show the performances of NSSVD++ as a function of the value of θ with $k = 20, 50, 80$, respectively. r_m remains 0 in these three figures. The value of θ is changed from 0 to 1, step by 0.1.

When θ is 0, NSSVD++ degenerates to SVD++. When θ is bigger than 0, NSSVD++ gains better performance than SVD++ no matter what value θ is. This is very different from WSVD++ and RSSVD++. It is because the candidate item set is heuristic selected by a neighbor-based algorithm, and most items in the set are likely to be negative examples. Furthermore, there is a similar phenomenon in these three figures. It is that the NDCG performances keep stable when θ gets up to a certain value. For $k = 20$, this value is 0.2; for $k = 50$, the value is 0.5; and for $k = 80$, the value is 0.6. All in all, when θ is 0.5 and k is 50, NSSVD++ gets the

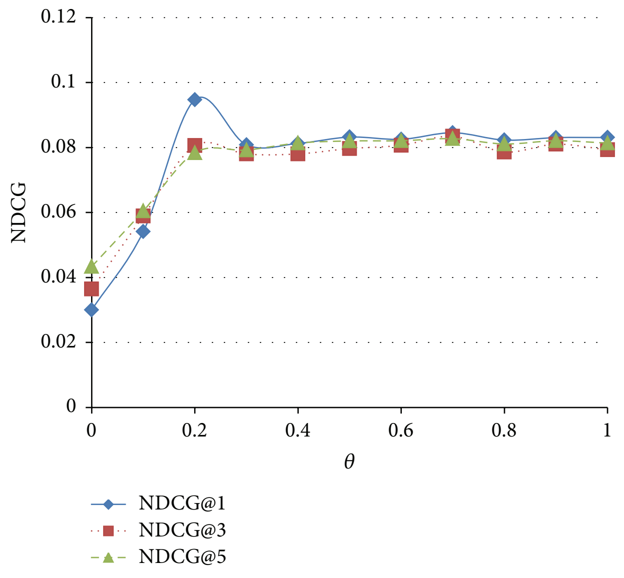

Figure 5: NDCG of NSSVD++ as a function of the value of θ with $k = 20$ and $r_m = 0$ on ML.

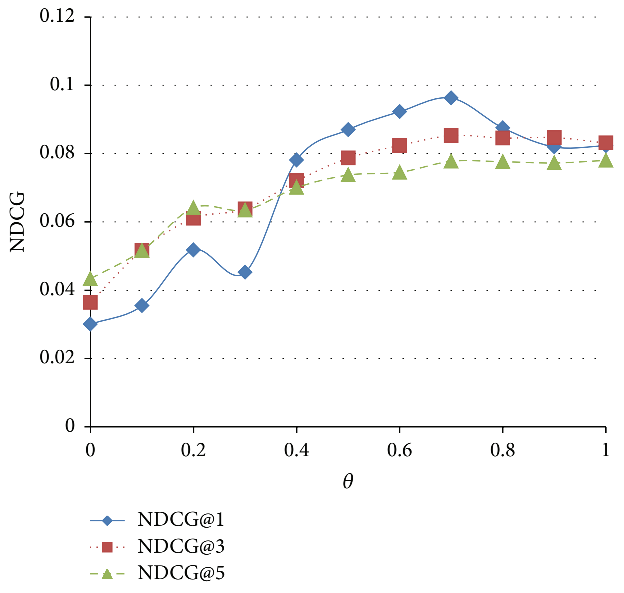

Figure 7: NDCG of NSSVD++ as a function of the value of θ with $k = 80$ and $r_m = 0$ on ML.

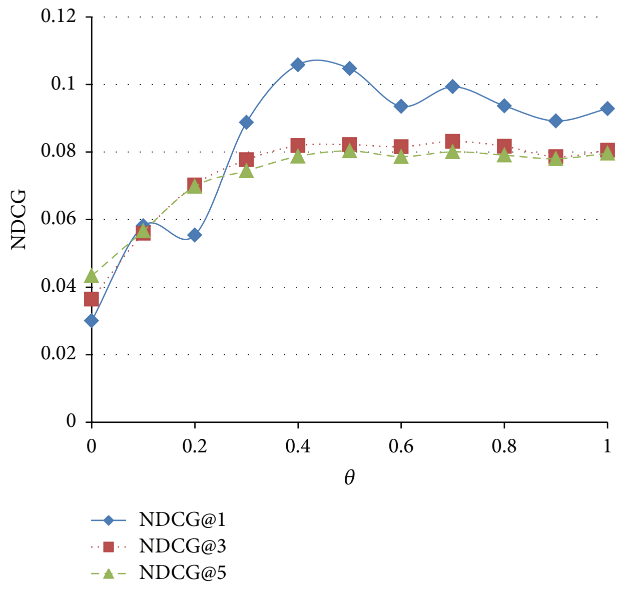

Figure 6: NDCG of NSSVD++ as a function of the value of θ with $k = 50$ and $r_m = 0$ on ML.

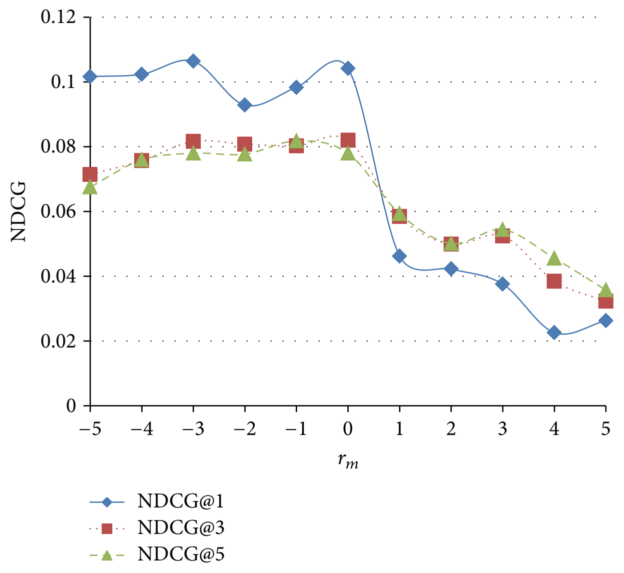

Figure 8: NDCG of NSSVD++ as a function of the value of r_m with $\theta = 0.5$ and $k = 50$ on ML.

best recommendation quality. This pair of parameter values is chosen as the best one in NSSVD++.

Figure 8 shows the performance of NSSVD++ as a function of the value of r_m with $\theta = 0.5$ and $k = 50$. The value of r_m is changed from -5 to 5, step by 1. Similar to WSVD++ and RSSVD++, when r_m is less than 1, the performance of NSSVD++ is better than it with r_m between 1 and 5. However, when r_m is between -5 and 0, the performance is almost the same. It is very different from WSVD++ and RSSVD++. This may be because the negative examples of NSSVD++ are generated by a neighbor-based strategy, which introduces high confidence of the negative examples. Small value of r_m does not introduce bias of the trained model. To be consistent with WSVD++ and RSSVD++, the chosen value of r_m for NSSVD++ is still 0.

6.5. Comparison with Baselines. In this subsection, we present a performance comparison of both accuracy and diversity between our proposed approaches and the baseline ones on ML and EM datasets. For each approach, we report the NDCG and Recall values at the 1st, 3rd, and 5th positions in the recommendation list and 1-call, COV, and CIL at the 5th position, comparing with NDCG+ values at the 5th position.

Table 1: Performance of different approaches on ML.

	NDCG			Recall			1-call	COV	CIL	NDCG+
	1	3	5	1	3	5				
UserCF	0.019	0.019	0.020	0.000	0.002	0.003	0.12	**129.4**	**69.8**	0.65
Slope-one	0.026	0.033	0.033	0.001	0.003	0.005	0.18	19.6	5.8	0.68
SVD++	0.030	0.036	0.043	0.001	0.003	0.006	0.19	20.8	7.6	**0.71**
OrdRec	0.080	0.065	0.062	0.002	0.005	0.008	0.23	19.0	4.6	0.52
Pure	0.037	0.033	0.032	0.001	0.002	0.004	0.11	34.2	13.0	0.66
AllRank	0.063	0.058	0.055	0.002	0.004	0.006	0.19	21.2	4.4	0.70
WSVD++	0.104	**0.083**	**0.079**	0.004	0.008	0.014	0.28	23.6	8.8	0.70
RSSVD++	0.095	0.074	0.073	**0.006**	0.007	0.013	0.27	24.0	9.2	0.69
NSSVD++	**0.108**	**0.083**	0.078	**0.006**	**0.013**	**0.019**	**0.30**	38.2	16.2	0.68

Table 1 illustrates the results on ML (the bold cell indicates the best result for that metric). In terms of NDCG measures, NSSVD++ outperforms all other approaches, followed by WSVD++ and RSSVD++. These proposed approaches improve the original SVD++ by 68% at least. This indicates their good performance in top-N recommendation task. These proposed approaches are improvements of SVD++ with utilizing negative examples in missing data. As a result, the existence of negative examples in missing data is demonstrated. In addition, NSSVD++ performs better than WSVD++ and RSSVD++. This is because the former one distinguishes negative examples from real missing data with a heuristic algorithm and makes the selected negative examples having a high confidence.

Pure is an improvement of SVD++ by considering all missing data as rating 0. It has similar NDCG performance as SVD++. This indicates that simply considering all missing data as negative examples is not feasible. AllRank considers missing data as negative ratings. The result shows that it outperforms original SVD++. However, its improvement is less than the improvement of our proposed approaches. This verifies that missing data are more likely to be negative examples rather than negative ratings. Furthermore, OrdRec improves SVD++ by ranking prediction. Its improvement is much less than our proposed approaches. This verifies that using the negative examples from missing data is a more effective method than considering recommendation problem as a ranking prediction one.

Recall is a metric for evaluation of the capability of recommending relevant items, which is the optimizing goal of AllRank. However, our proposed approaches still gain the best performance. This indicates that utilizing the negative examples in missing data can leverage the capability of recommending relevant items.

The proposed approaches improve SVD++ on 1-call about 42% at least. In addition, the proposed approaches gain better performance than other baselines about 17% at least. This indicates the contribution to provide valuable recommendations at top-N positions. It raises the chance that users receive at least one relevant recommendation among just a few top-ranked items.

UserCF gets the best diversity performance. If we only focus on the matrix factorization approaches, NSSVD++ outperforms others. The proposed approaches improve the original SVD++ at least 13% on COV and 16% on CIL. Furthermore, NSSVD++ outperforms the other two proposed approaches about 59% on COV and 76% on CIL. This illustrates that our proposed approaches can improve the baseline ones on diversity and novelty as well as on accuracy.

NDCG+ is a comparative evaluation metric, and we only consider the value at the 5th position. Unfortunately, the proposed approaches almost get the worst performances on this metric. It is mainly because negative examples introduce some biases of user and item factors. The prediction of the proposed approaches considers whether or not users will rate items as well as the predicted rating values. However, this drawback is not very much, and the worst approach (NSSVD++) gets only 4% less than SVD++, which gains the best NDCG+ performance.

All in all, the experiment results on ML show that the proposed recommendation approaches outperforms baseline ones on both accuracy and diversity. In order to further demonstrate the effectiveness of the proposed approaches, a similar experiment is carried out on EM dataset.

Table 2 shows the performance of different approaches on EM, which is a much larger dataset than ML. Similar observations can be found. The proposed approaches still gain better performance than the original SVD++ and other ones. However, the advantages of WSVD++ and RSSVD++ decrease when the dataset changes from ML to EM. This may be because EM is a much sparser dataset; missing positive examples take higher proportion of missing data. In this circumstance, treating all missing data in a unique way is not a suitable solution. This problem does not exist in NSSVD++, which uses a heuristic method to sample negative examples from missing data. Therefore, NSSVD++ still improves SVD++ by 36% on NDCG, 35% on Recall, 78% on 1-call, 90% on COV, and 88% on CIL at least. The experiment results on EM further verify the effectiveness of our proposed approaches, especially NSSVD++.

In addition, it is obvious that there is no conversion relationship between NDCG and NDCG+. This illustrates that good performance on ranking prediction does not indicate good quality of top-N recommendation. Therefore, we cannot easily consider recommendation problem as a ranking prediction one.

Table 2: Performance of different approaches on EM.

	NDCG			Recall			1-call	COV	CIL	NDCG+
	1	3	5	1	3	5				
UserCF	0.010	0.014	0.019	0.002	0.008	0.017	0.12	**174.0**	**116.2**	0.65
Slope-one	0.035	0.034	0.034	0.008	0.019	0.027	0.15	11.0	7.0	0.71
SVD++	0.112	0.091	0.088	0.031	0.058	0.080	0.28	30.6	12.6	**0.83**
OrdRec	0.063	0.057	0.049	0.015	0.040	0.044	0.19	10.6	6.8	0.66
Pure	0.074	0.071	0.071	0.018	0.039	0.052	0.27	38.8	15.0	0.69
AllRank	0.111	0.104	0.103	0.031	0.073	0.103	0.37	13.4	4.4	**0.83**
WSVD++	0.081	0.098	0.100	0.020	0.065	0.090	0.36	56.0	26.0	0.80
RSSVD++	0.095	0.096	0.100	0.027	0.061	0.086	0.34	52.8	23.6	0.79
NSSVD++	**0.153**	**0.148**	**0.150**	**0.042**	**0.102**	**0.143**	**0.50**	58.2	23.8	0.81

7. Conclusions

In recommender systems, the rating data, which are used for generating recommendations, are only a small part of the whole dataset. The rest part of the dataset is considered as missing data. Existing recommendation approaches always use observed rating data to learn models and to generate recommendations. However, part of unknown ratings is because users choose not to rate them. This kind of unknown ratings is the negative examples of user preference in our point of view. Therefore, we propose three schemes to deal with missing data. The first one is based on a weighting scheme, which considers all missing data as negative examples with different confidence towards positive ones. The second one is based on a random sampling scheme, which samples some missing data as negative examples with a stochastic approach. The third one is also a sampling scheme, which uses a neighbor-based algorithm to generate negative examples with a heuristic idea. In addition, we adapt the schemes with SVD++ in order to improve the top-N recommendation performance by using missing data. Experiment results show that all the proposed approaches outperform baseline ones in top-N recommendation on both accuracy and diversity.

Conflict of Interests

The authors declare that there is no conflict of interests regarding the publication of this paper.

Acknowledgments

This work is supported by the National Key Technology R&D Program of China (Project no. 2014BAD10B08) and the National Natural Science Foundation of China (Project no. 61250010). The EachMovie dataset is by courtesy of Digital Equipment Corporation and was generously provided by Paul McJones.

References

[1] P. Cremonesi, Y. Koren, and R. Turrin, "Performance of recommender algorithms on top-n recommendation tasks," in *Proceedings of the 4th ACM Conference on Recommender Systems*, pp. 39–46, ACM, September 2010.

[2] N. N. Liu, M. Zhao, and Q. Yang, "Probabilistic latent preference analysis for collaborative filtering," in *Proceedings of the ACM 18th International Conference on Information and Knowledge Management, (CIKM '09)*, pp. 759–766, November 2009.

[3] B. M. Marlin, R. S. Zemel, R. Sam, and M. Slaney, "Collaborative filtering and the missing at random assumption," in *Proceedings of the 23rd Conference on Uncertainty in Artificial Intelligence*, pp. 267–275, July 2007.

[4] H. Steck, "Training and testing of recommender systems on data missing not at random," in *Proceedings of the 16th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pp. 713–722, ACM, July 2010.

[5] H. Steck, "Evaluation of recommendations: rating-prediction and ranking," in *Proceedings of the 7th ACM Conference on Recommender Systems (RecSys '13)*, pp. 213–220, ACM, October 2013.

[6] X. Zhao, Z. Niu, and W. Chen, "Interest before liking: two-step recommendation approaches," *Knowledge-Based Systems*, vol. 48, pp. 46–56, 2013.

[7] 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 (KDD '08)*, pp. 426–434, August 2008.

[8] G. Adomavicius and Y. Kwon, "Improving aggregate recommendation diversity using ranking-based techniques," *IEEE Transactions on Knowledge and Data Engineering*, vol. 24, no. 5, pp. 896–911, 2012.

[9] J. S. Breese, D. Heckerman, and C. Kadie, "Empirical analysis of predictive algorithms for collaborative filtering," in *Proceedings of the 14th Conference on Uncertainty in Artificial Intelligence*, pp. 43–52, Morgan Kaufmann, 1998.

[10] W. Chen, Z. Niu, X. Zhao, and Y. Li, "A hybrid recommendation algorithm adapted in e-learning environments," *World Wide Web*, vol. 17, no. 2, pp. 271–284, 2014.

[11] J. Delgado and N. Ishii, "Memory-based weighted majority prediction," in *Proceedings of the Workshop on Recommender Systems (SIGIR '99)*, ACM, Berkeley, Calif, USA, 1999.

[12] J. L. Herlocker, J. A. Konstan, L. G. Terveen, and J. T. Riedl, "Evaluating collaborative filtering recommender systems," *ACM Transactions on Information Systems*, vol. 22, no. 1, pp. 5–53, 2004.

[13] T. Hofmann, "Latent semantic models for collaborative filtering," *ACM Transactions on Information Systems*, vol. 22, no. 1, pp. 89–115, 2004.

[14] M. Weimer, A. Karatzoglou, Q. V. Le, and A. J. Smola, "Cofi rank-maximum margin matrix factorization for collaborative ranking," in *Advances in Neural Information Processing Systems*, pp. 1593–1600, 2007.

[15] N. N. Liu and Q. Yang, "Eigenrank: a ranking-oriented approach to collaborative filtering," in *Proceedings of the 31st Annual International ACM SIGIR Conference on Research and Development in Information Retrieval (SIGIR '08)*, pp. 83–90, 2008.

[16] Y. Koren and J. Sill, "Ordrec: an ordinal model for predicting personalized item rating distributions," in *Proceedings of the 5th ACM Conference on Recommender Systems*, pp. 117–124, ACM, 2011.

[17] R. Pan, Y. Zhou, B. Cao et al., "One-class collaborative filtering," in *Proceedings of the 8th IEEE International Conference on Data Mining (ICDM '08)*, pp. 502–511, IEEE, 2008.

[18] Y. Li, J. Hu, C. Zhai, and Y. Chen, "Improving one-class collaborative filtering by incorporating rich user information," in *Proceedings of the 19th ACM International Conference on Information and Knowledge Management*, pp. 959–968, ACM, October 2010.

[19] S. Rendle, C. Freudenthaler, Z. Gantner, and L. Schmidt-Thieme, "BPR: Bayesian personalized ranking from implicit feedback," in *Proceedings of the 25th Conference on Uncertainty in Artificial Intelligence (UAI '09)*, pp. 452–461, Montreal, Canada, June 2009.

[20] K. Järvelin and J. Kekäläinen, "Cumulated gain-based evaluation of IR techniques," *ACM Transactions on Information Systems*, vol. 20, no. 4, pp. 422–446, 2002.

[21] Y. Shi, A. Karatzoglou, L. Baltrunas, M. Larson, N. Oliver, and A. Hanjalic, "Climf: learning to maximize reciprocal rank with collaborative less-is-more filtering," in *Proceedings of the 6th ACM Conference on Recommender Systems (RecSys '12)*, pp. 139–146, ACM, September 2012.

[22] Y.-J. Park and A. Tuzhilin, "The long tail of recommender systems and how to leverage it," in *Proceedings of the 2nd ACM Conference on Recommender Systems (RecSys '08)*, pp. 11–18, ACM, Lausanne, Switzerland, October 2008.

[23] D. Lemire and A. Maclachlan, "Slope one predictors for online rating-based collaborative filtering," *Society for Industrial Mathematics*, vol. 5, pp. 471–480, 2005.