Serendipitous Personalized Ranking for Top-N Recommendation

Qiuxia Lu, Tianqi Chen, Weinan Zhang, Diyi Yang, Yong Yu
Department of Computer Science and Engineering
Shanghai Jiao Tong University
800 Dongchuan Road, Shanghai, China
{*luqiuxia,tqchen,wnzhang,yangdiyi,yyu*}*@apex.sjtu.edu.cn*

***Abstract*—Serendipitous recommendation has benefitted both e-retailers and users. It tends to suggest items which are both unexpected and useful to users. These items are not only profitable to the retailers but also surprisingly suitable to consumers' tastes. However, due to the imbalance in observed data for popular and tail items, existing collaborative filtering methods fail to give satisfactory serendipitous recommendations. To solve this problem, we propose a simple and effective method, called serendipitous personalized ranking. The experimental results demonstrate that our method significantly improves both accuracy and serendipity for top-N recommendation compared to traditional personalized ranking methods in various settings.**

***Keywords*-Collaborative Filtering; Recommender Systems; Matrix Factorization; Serendipity**

I. Introduction

Collaborative filtering (CF) based recommender systems are playing an increasingly important role in a variety of Internet services, such as Amazon, Netflix, and Last.fm. They predict users' preferences based on their observed behavior. In recent years, some international contests such as the Netflix $1 Million Challenge and the Yahoo! Music Recommendation Challenge for the KDDCup 2011 have led to the surging interest in CF-based recommender systems in both academia and industry.

Though the widely publicized recommendation contests have greatly promoted research on recommender systems, they have narrowed the horizon of the literature at the same time. The high stakes competition on relevance can reduce the serendipity, which is a key aspect of recommender systems. In fact, serendipitous recommendation has been shown to benefit both e-retailers and users by suggesting both unexpected and useful items, which are always located in the tail of the popularity distribution. On one hand, the aggregated profit on tail items is a large proportion of the total profit [1]. For example, the proportion of tail item profit is 25% on Amazon and 20% on Netflix. On the other hand, the consumers tend to enjoy the serendipity of stumbling upon something that turns out to be fascinating [2].

However, there is still no satisfactory CF solution for serendipitous item recommendation. The serious imbalance of observed preferences between popular (lowly unexpected) and tail (highly unexpected) items results in unbalanced training in a supervised learning model. For a rating prediction task, supervised learning models tend to have more training cases for popular items than for tail ones. Therefore, the state-of-the-art CF models always give high performance on popular items but relatively low performance on tail ones. For a personalized ranking task, pairwise ranking models [3] always select popular items as positive examples (because they have more observed preferences) and tail items as negative examples (because they have fewer observed preferences). As a result, popular items are much more likely to rank higher than tail items, making the recommendation list lack of serendipity. A good list should have high accuracy to match users' tastes and should contain a not-too-low proportion of unexpected items to give serendipity. Unfortunately, due to the low performance on tail items, there is usually a trade-off between accuracy and serendipity in the top-N recommendation list. To the best of our knowledge, there is no previous work that successfully improves both accuracy and serendipity for top-N item recommendation.

We also investigate users' preferences. Popular items have probably already been recommended to (or noticed by) the users. For example, Netflix presents recently released movies on the home page and Yahoo! Music shows popular albums or artists on its portal page. Thus if a popular item is not rated (or consumed) by a user, it is likely that the user has seen it but has no interest in it, while if a tail item is not rated by a user, it is likely that he or she has not noticed it yet rather than that he or she dislikes it. Based on this, popular items without observed preferences are more likely to be the true negative items to the user than tail ones.

Previous work by Cremonesi et al. [4] suggests that CF algorithms optimized for minimizing root mean squared error (RMSE) do not necessarily perform as expected in top-N recommendation task. On the other hand, some recently proposed ranking oriented CF approaches show good performance in tackling the top-N recommendation problem, which constitute the foundation of our algorithm.

Based on the facts above, we propose a simple and effective method for serendipitous item recommendation, which makes the ranking sensitive to the popularity of negative examples. We compare our method, which we call serendipitous personalized ranking, with traditional collaborative ranking on three popular latent factor models using

logistic loss and hinge loss respectively. Two groups of experiments are conducted on the Netflix dataset and the Yahoo! Music dataset, which demonstrate that our method significantly improves both accuracy and serendipity in top-N recommendation.

The paper is organized as follows: in Section II we briefly review some related works. Section III describes our serendipitous personalized ranking method in detail. We then show the experimental results and analysis in Section IV. Finally, we conclude the paper in Section V.

II. Related Work

The two primary approaches in the field of collaborative filtering are the neighborhood approaches [5], [6] and the latent factor approaches [7], [8]. The neighborhood approaches are based on the similarity among users or items, called user-based approach and item-based approach respectively. Most latent factor models are based on the factorization of the user-item rating matrix [9]. Named after the related Singular Value Decomposition, these latent factor models are known as SVD models. According to [10], there are several variants of SVD models which have high quality in rating prediction tasks.

Ranking oriented collaborative filtering approaches have been proposed recently in order to tackle the top-N recommendation task more practically. Pessiot et al. [11] proposed a pairwise preference error minimization framework to optimize the item ranking. Liu and Yang [12] presented an EigenRank algorithm based on user-user similarity defined by their item ranking. Moreover, Rendle et al. [3] proposed a generic optimization criterion BPR-Opt and Weimer et al. [13] proposed Ordinal Loss for maximum margin matrix factorization (MMMF), both of which directly optimize for ranking, and we adopt them as the baselines to compare against.

It has been widely acknowledged that serendipity is a key aspect in recommender systems [14]. Since serendipity is difficult to study and largely depends on subjective characteristics, there are various definitions and evaluations proposed. Herlocker et al. [15] defined serendipity as a measure of how often the recommended items are both surprising and interesting to the users. Celma and Herrera [16] presented two methods, item centric and user centric, to evaluate the quality of serendipitous recommendations. [17] introduced TANGENT, which is based on a graph mining technique to provide "Surprise-me" recommendations. In [18], the author proposed an algorithm which focused on the search time each user would need to find a desirable and novel item by him/herself. Murakami et al. [19] and Ge et al. [20] captured the two essential aspects of serendipity: unexpectedness and usefulness. Adamopoulos and Tuzhilin [21] further revised this definition of serendipity and proposed a recommendation approach which took both quality and unexpectedness into account. However, their approach is more like a two-stage procedure, which requires to predict the quality and unexpectedness first and then combine them to make a optimal decision. Other works such as [22], [23] focused on the metric of novelty while [24], [25] dealt with the trade off between accuracy and diversity. Though overlapping with the concept of serendipity, they did not capture the unexpectedness and usefulness together. In this paper, we follow the definition of serendipity in [19], [20] and propose a serendipitous ranking model which optimizes accuracy and serendipity in a single learning procedure. Thus, our method can improve both recommendation accuracy and serendipity, while in the previous works such as [20], [24], serendipity or diversity always indicated a loss of accuracy.

Some work has proposed to leverage the problem of the long tail. Park and Tuzhilin [26] used clustering ideas to deal with the small number of ratings of the items in the long tail. They clustered the items in the tail into various groups and applied several models such as kNN and SVM to predict ratings. The experimental results showed that the clustered tail method solved the long tail problem in the sense that the error rates for the items in tail were lower than with the non-clustered method. However, [26] concentrated on the evaluation metric of RMSE and did not address the top-N recommendation scenario, which constitutes the focus of this paper. Besides, [26] did not explore their clustering methods on SVD models, which are state-of-the-art collaborative filtering approaches. Steck [27] proposed a popularity-stratified training method to examine the trade-off between item popularity and recommendation accuracy. However, this work was focused on rating regression and did not directly optimize rank; it was evaluated by the measures of item popularity and recommendation accuracy which are different from serendipity.

To sum up, experimental studies of serendipity are still rare; how to implement serendipitous recommender systems is still an open question.

III. Serendipitous Personalized Ranking

The real objective of top-N recommendation is to optimize the ranking order, so the traditional squared error loss function is actually not solving the right problem in this scenario. Recently, some recommendation algorithms which directly optimize for ranking have been proposed and they demonstrate good recommendation accuracy. However, these ranking-based approaches still work badly in suggesting serendipitous items. There seems to be a trade-off between recommendation accuracy and serendipity. In this section we propose a Serendipitous Personalized Ranking (SPR) method to explore the relation between accuracy and serendipity.

We first describe the notational framework. $U = \{u_1, u_2, ..., u_m\}$ denotes the set of all users and $I = \{i_1, i_2, ..., i_n\}$ denotes the set of all items. The ratings of U on I are stored in a matrix $R(m \times n)$, and the entry r_{ui} represents the

rating user u assigns to item i. Let S and $\bar{S}$ denote the set of (u, i) pairs whose ratings are observed and unobserved respectively. Further, we define I_u^+ as the set of items that user u likes:

$$I_u^+ := \{i \in I : (u, i) \in S, r_{ui} >= r^+\}$$

where r^+ denotes a high rating value which depends on the rating range of the dataset, following the methodology used in [4]. Different from the traditional approach, most ranking models use item pairs as training data and optimize for correctly ranking the user-dependent item pairs instead of approximating the single item ratings. As suggested by [28], there is a much larger proportion of negative examples than positive examples in the unobserved part. Both [4] and [28] show that considering unobserved as negative can improve top-N recommendation accuracy. So in our work we assume that a user prefers all observed items that he/she gives high ratings to ($>= r^+$) over the non-observed items or the observed items that he/she gives low ratings to ($< r^+$). Thus the training data $D_s \in U \times I \times I$ is formalized as follows:

$$D_s := \{(u, i, j)|i \in I_u^+ \wedge j \in I \setminus I_u^+\}$$

The triple $(u, i, j) \in D_s$ means that user u prefers item i to item j.

A. Traditional Personalized Ranking Approach

Many traditional personalized ranking methods maximize the metric of area under the ROC curve (AUC). The AUC of a user on a given model is defined as [29]:

$$AUC(u) := \frac{1}{|I_u^+||I \setminus I_u^+|} \sum_{i \in I_u^+} \sum_{j \in I \setminus I_u^+} \delta(\hat{r}_{ui} - \hat{r}_{uj})$$

where $\hat{r}_{ui}$ is a real-valued function of the model parameter (omitted) which captures the preference of user u on item i. A value of $\hat{r}_{ui} - \hat{r}_{uj} > 0$ means the model predicts that user u prefers item i to item j. The average AUC can be obtained by :

$$AUC := \frac{1}{|U|} \sum_{u \in U} AUC(u)$$

With the notation of D_s it can be formalized as:

$$AUC = \sum_{(u,i,j) \in D_s} z_u \delta(\hat{r}_{ui} - \hat{r}_{uj}) \quad (1)$$

Here z_u is a normalizing term:

$$z_u = \frac{1}{|U||I_u^+||I \setminus I_u^+|}$$

$\delta(x)$ is a 0-1 function, which corresponds to 0-1 loss:

$$\delta(x) = \begin{cases} 1 & x > 0 \\ 0 & otherwise \end{cases} \quad (2)$$

Since a 0-1 loss function is non-differentiable, some surrogate loss functions $l(x)$ are used for optimization. There are two main kinds of surrogate functions. One is logistic loss, which corresponds to Bayesian personalized ranking [3]:

$$l(x) = \ln(1 + e^{-x}) \quad (3)$$

Another is hinge loss, which leads to maximum margin matrix factorization (MMMF) model [13]:

$$l(x) = \max(0, 1 - x) \quad (4)$$

These loss functions can be easily solved by using stochastic gradient descent.

B. Serendipitous AUC Optimization

The AUC metric evaluates the predicted probability that a user's preference values on his/her liked items are greater than that on his/her disliked items (low rated or unrated). However, users are more likely to provide feedback on popular items because they are easily found. In other words, the observed feedback is biased towards popular items compared to users' real interests. There is a much larger proportion of ratings on the popular items than on the tail items. Thus a model which is learned by optimizing the AUC metric defined in Equation 1 will result in a large popularity bias. That is, the popular items which are easily expected will be much more favored and the unexpected items will rarely appear in the top-N list. In order to suggest more serendipitous items to the user, we propose a Serendipitous AUC (SAUC) metric by adding a term of popularity weight in Equation 1:

$$SAUC = \sum_{(u,i,j) \in D_s} z_u \delta(\hat{r}_{ui} - \hat{r}_{uj})(Pop(j))^\alpha \quad (5)$$

where $Pop(j)$ denotes the popularity of item j. Since the observation that a user prefers an item to a popular item provides more information to us in capturing his/her interest than the observation that he/she prefers an item to an unpopular item. Thus we multiply $(Pop(j))^\alpha$ to promote the triple (u, i, j) where j is popular.

In the experiments, we assign the popularity weight $Pop(j) \propto N_j$, where N_j is the number of ratings of item j in the observed set S. The exponent α is a parameter which can be tuned for optimal performance. Note that SAUC with $\alpha = 0$ is equivalent to AUC and a larger α means the model will tend to suggest more tail items which may be highly unexpected. By tuning the value of α we can improve both accuracy and serendipity of the recommendation result.

SAUC can also be optimized using logistic loss (Equation 3) or hinge loss (Equation 4). We conduct experiments to compare the two loss functions.

C. Collaborative Filtering Models

In order to explore the effectiveness of our SPR, we apply it to various collaborative filtering algorithms. In this section, we will first introduce the CF models that we use and then show how to apply SPR to these models.

The latent factor model has good predictive accuracy and is one of the main algorithms in collaborative filtering. Most latent factor models are based on factorizing the user-item rating matrix [9]. Named after the related Singular Value Decomposition, these models are also known as SVD models. According to [10], there are several variants of SVD models which have high quality in the rating prediction tasks. Thus we adopt them to explore the effectiveness of our SPR in recommendation accuracy and serendipity[1]. SVD models factorize the user-item rating matrix to the product of two low rank matrices, namely the user factor p_u and the item factor q_i, where p_u and q_i are both of f-dimension. The rating of user u on item i is predicted as follows in the basic SVD model:

$$\hat{r}_{ui} = b_{ui} + q_i^T p_u \tag{6}$$

where b_{ui} is a bias term which accounts for the user and item effects. One extension to SVD is known as SVD++ [9]; it has a much higher quality in rating prediction than the basic SVD:

$$\hat{r}_{ui} = b_{ui} + q_i^T \left(p_u + |R(u)|^{-\frac{1}{2}} \sum_{j \in R(u)} y_j \right) \tag{7}$$

where $R(u)$ denotes the set of items rated by user u, and $\sum_{j \in R(u)} y_j$ represents the implicit preference of user u.

The neighborhood model is another important approach in CF; it has an advantage over the latent factor model in detecting local information. In the work of [10] they integrate a neighborhood model with a latent factor model to enrich each other:

$$\hat{r}_{ui} = b_{ui} + q_i^T p_u + |R(u,i;k)|^{-\frac{1}{2}} \sum_{j \in R(u,i;k)} c_{ij} \tag{8}$$

Here $R(u,i;k)$ stands for the k-nearest neighbors of item i which are rated by user u. In Equation 8, the first two terms represent the basic SVD model while the last term represents the neighborhood model. We denote this integrated model as SVDNbr.

These models are traditionally learned by optimizing RMSE in the rating prediction problem. However, in top-N recommendation, they can still work perfectly by optimizing AUC and SAUC. This can be easily achieved by replacing $\hat{r}_{ui}$ and $\hat{r}_{uj}$ in Equation 1 and Equation 5 with model parametrization in Equation 6,7 and 8 respectively.

IV. Experiments and Discussion

In this section, we present the experiments to evaluate the performance of our serendipitous personalized ranking method in both recommendation accuracy and serendipity. We have chosen three popular latent factor models, namely SVD, SVD++ and SVDNbr, which are known to give high performance in the rating prediction tasks. In our evaluation, the three models are learned by optimizing AUC or SAUC, which are denoted as method PR and SPR respectively in the following sections. Two groups of experiments are conducted to compare PR and SPR. The first group is aimed to evaluate recall while the second group concentrates on precision and serendipity in the top-N recommendation list.

[1]SPR can also be applied to adaptive neighborhood models in the similar way

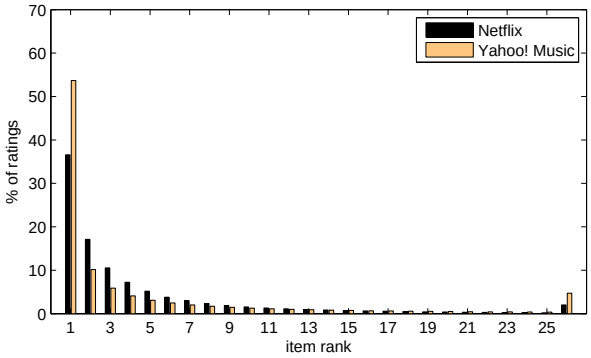


Figure 1. Rating distribution for Netflix and Yahoo! Music

A. Dataset

We use two popular datasets in our study: the Netflix data [30] and the Yahoo! Music data [31]. The Netflix dataset contains about 100 million ratings on a scale of 1 to 5 from $480,189$ users on $17,770$ movies. The provided dataset is already split into a training set and a probe set. The Yahoo! Music dataset contains about 250 million ratings on a scale of 0 to 100 from $1,000,990$ users on $624,961$ music items. The provided Yahoo! Music dataset has been split into two disjoint sets: a training set and a validation set.

Figure 1 plots the rating distributions of the Netflix and Yahoo! Music datasets. Item bars in the horizontal axis are ordered according to their popularity, most popular at the left. Each bar contains 2% of items except for the last two bars which represent the last 50% of less popular items in Netflix and Yahoo! Music respectively. About 33% of ratings in Netflix involve only the most popular 1.7% of items. Similar to [4], we define this small set of popular items as the short head and the rest as the long tail. The rating distribution of Yahoo! Music is even more long tailed, where the short head involves only the most popular 0.4% of items.

B. Experimental Setup

In the first group of experiments we follow the testing methodology of [4] to evaluate recall. For each dataset, known ratings are randomly split into two subsets: the training set M and the test set T. The test set contains only high ratings, 5-star for Netflix by following [4] and 80 or higher for Yahoo! Music according to the official definition

of users' liked music[2]. So we can reasonably state that T contains items liked by the respective users. The creation of M and T are based on their original splitting. For the Netflix dataset, the training set M is the original training set excluding the probe set, while the test set T consists of all the high ratings in the probe set. A similar splitting procedure is applied to the Yahoo! Music dataset, i.e., the training set M is the original training set and the test set T contains all the high ratings in the validation set.

We train our models over the training set M. Then for each (u, i) pair in T, we randomly sample 1000 additional items unrated by user u. Then we generate a ranking list for all the 1001 items. In the top-N recommendation task, if the rank of the test item i is smaller than or equal to N, we get a hit. Thus the overall recall is defined as follows:

$$recall@N = \frac{\#hits}{|T|}$$

where $|T|$ is the number of test ratings. In order to evaluate the performance of recommending methods on the aspect of serendipity, the test set T is further split into two subsets, T_{head} and T_{tail}. T_{head} consists of items in the short head while T_{tail} contains items in the long tail. Thus $recall@N$ on T_{tail} can reflects the ability of the method in suggesting serendipitous items.

In the second group of experiments, we randomly split the original training set into a new training set Tr and a ground truth set Gt. Gt contains $1/5$ high rating items of each user (5-star for Netflix and scored 80 or higher for Yahoo! Music), while Tr contains all the remaining data.

In order to explore the performance of our method in suggesting serendipitous items, we follow the metric proposed by Murakami et al. [19] and Ge et al. [20] which captures the two essential aspects of serendipity: unexpectedness and usefulness, and adapt their measures to our method. Particularly, according to [20], an unexpected set of recommendations for user u ($UNEXP(u)$) is defined as:

$$UNEXP(u) = RS(u) \setminus PM \tag{9}$$

where PM is a set of recommendations generated by a primitive model which is assumed of low unexpectedness. $RS(u)$ denotes the top-N recommendations generated by a recommender system for user u. When an element of $RS(u)$ does not belong to PM, they consider it to be unexpected. In our experiments, we follow the methodology of [21] using the top-K items with the largest number of ratings to form the PM recommendation list (where $K = 100$).

Based on the definition of unexpectedness, they introduce serendipity measure as :

$$Srdp(u) = \frac{|UNEXP(u) \cap USEFUL(u)|}{N} \tag{10}$$

[2]http://kddcup.yahoo.com

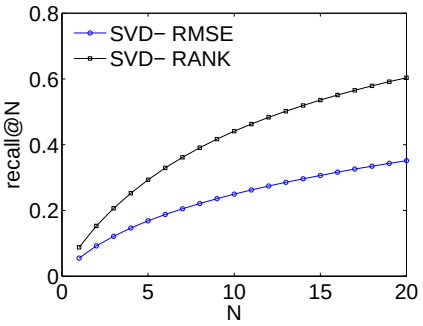


Figure 2. Recall of SVD-RMSE and SVD-RANK on Netflix

where $USEFUL(u)$ denotes the useful (relevant) items for user u and N is the size of recommendation set $RS(u)$. In our experiments, we consider an item to be useful to an user if it is high rated by the user ($USEFUL(u) = \{i|(u, i) \in Gt\}$). Thus, the average serendipity for top-N recommendation is defined as follows:

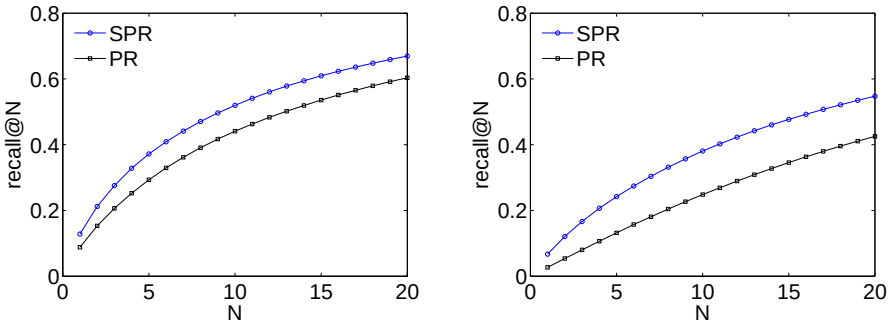
$$Srdp@N = \frac{1}{|U|} \sum_{u \in U} \frac{|UNEXP(u) \cap USEFUL(u)|}{N} \tag{11}$$

Besides, the well-known precision at N ($P@N$) metric is used to evaluate the precision performance. Through our comparison, the number of latent factors is set to 100 in both groups of experiments.

C. Recall

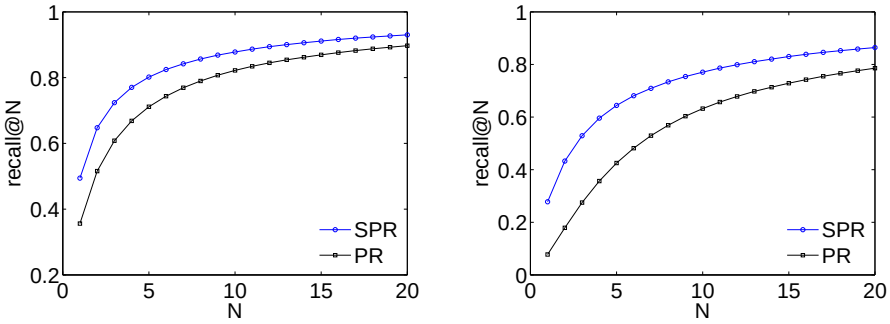
It has been pointed out by previous work [4] that algorithms optimizing RMSE do not optimize rank in top-N recommendation. We use experiments to confirm this assumption at the beginning of our exploration. We apply the RMSE optimized basic SVD (denoted by method SVD-RMSE) and the rank optimized SVD (denoted by method SVD-RANK) in the same Netflix setting. Figure 2 illustrates the performance comparison between SVD-RMSE and SVD-RANK, from which we can see that the ranking method has a significant advantage over the RMSE optimized method. For instance, the recall of SVD-RANK at $N = 10$ is about 0.44 which has an improvement of 78% over the recall of SVD-RMSE. The results demonstrate that a ranking method is more appropriate for top-N scenario, thus we adopt it in our serendipitous recommendation.

Figure 3 provides a first assessment of recommendation accuracy and serendipity of PR and SPR on the Netflix dataset. Figure 3(a) shows recall on the test set while Figure 3(b) shows recall on the long tail set (with the most rated 1.7% of items removed from the test set). In both figures we can see that our proposed SPR outperforms PR regarding recall at all values of $N \in [1, 20]$. For instance, in Figure 3(a) the recall of SPR at $N = 10$ is about 0.52, i.e., it has a probability of 52% to place an appealing movie in the top-10 recommendation list, which is higher than the recall



(a) recall at N (b) recall at N on long tail (98.3% of items)

Figure 3. Recall of PR and SPR on Netflix



(a) recall at N (b) recall at N on long tail (99.6% of items)

Figure 4. Recall of PR and SPR on Yahoo! Music

of PR (about 44% at $N = 10$). This result demonstrates that our considering of popularity weight in personalized ranking models will not hurt the overall recommendation accuracy, instead, it can greatly improve it. Noting that the gap between line PR and SPR in Figure 3(b) is wider than that in Figure 3(a), which shows that our serendipitous personalized ranking method has a greater advantage over the traditional personalized ranking method in suggesting serendipitous items.

Figure 4 shows a performance comparison of recall between PR and SPR on the Yahoo! Music dataset, with results similar to Figure 3. This further proves that SPR outperforms PR in recommendation accuracy and serendipity. In both Figure 3 and Figure 4, we use the basic SVD model with logistic loss. However, a similar improvement of SPR over PR is found when applied to the extension models SVD++ and SVDNbr. Here we set the popularity weight α of SPR to 0.5, which is optimal for recommendation accuracy. The impact of α will be further discussed in a later section.

D. Precision and Serendipity

In this section, we will look at the precision and serendipity of the recommendation lists generated by the methods PR and SPR. Precision is evaluated by the well-known precision at N ($P@N$) metric while serendipity is evaluated by $Srdp@N$, which is defined in Equation 11. The popularity weight α is also set to 0.5 here.

Table I shows the average precision and serendipity of the recommended top-5 lists across all users on the Netflix dataset. In this table, the results of PR and SPR applied to the three latent factor models with two loss functions are all listed. In the metric of $P@5$, SPR has an improvement over PR of about 10% in all three models with either loss function. That is to say our proposed serendipitous personalized ranking method can give better overall recommendation precision than the traditional personalized ranking method. This observation informs us that precision and serendipity may no longer be a trade-off in SPR. Now let us turn to the results of $Srdp@5$ for further confirmation. Averagely, SPR outperforms PR for about 55% with regard to $Srdp@5$. Taking the model SVD++ with logistic loss as an example, the $Srdp@5$ of SPR is 0.3036, i.e., about 30% of the recommended items are both unexpected and useful, which has an improvement of 58% over PR.

Table II presents the recommendation precision and serendipity of PR and SPR on the Yahoo! Music dataset. The same models and evaluation metrics are used as on the Netflix dataset. From Table II we can obviously see that our proposed serendipitous personalized ranking method

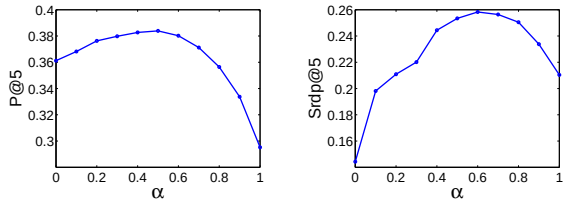
Table I
PRECISION AND SERENDIPITY ON NETFLIX

Model	Loss	Method	P@5	Srdp@5
SVD	Logistic	PR	0.3612	0.1443
		SPR	**0.3839**	**0.2534**
	Hinge	PR	0.3339	0.1500
		SPR	**0.3778**	**0.2460**
SVD++	Logistic	PR	0.3958	0.1916
		SPR	**0.4316**	**0.3036**
	Hinge	PR	0.3561	0.2458
		SPR	**0.4110**	**0.2932**
SVDNbr	Logistic	PR	0.3751	0.1521
		SPR	**0.4080**	**0.2962**
	Hinge	PR	0.3435	0.2522
		SPR	**0.3781**	**0.2978**

Table II
PRECISION AND SERENDIPITY ON YAHOO! MUSIC

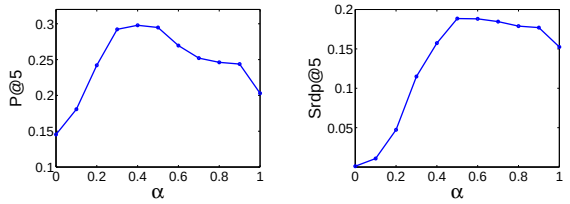
Model	Loss	Method	P@5	Srdp@5
SVD	Logistic	PR	0.1454	0.0010
		SPR	**0.2803**	**0.1909**
	Hinge	PR	0.1534	0.0122
		SPR	**0.2751**	**0.1995**
SVD++	Logistic	PR	0.1761	0.0113
		SPR	**0.3605**	**0.2445**
	Hinge	PR	0.1805	0.0348
		SPR	**0.3568**	**0.2771**
SVDNbr	Logistic	PR	0.1513	0.0026
		SPR	**0.4626**	**0.3499**
	Hinge	PR	0.2322	0.1286
		SPR	**0.4374**	**0.3834**

has an advantage over the traditional personalized ranking method with regard to both recommendation precision and serendipity, as is the case on the Netflix dataset. However, the average improvement in precision of SPR over PR on Yahoo! Music is 110%, which is much larger than that on Netflix. For instance, the precision of SPR applied to the model SVD++ with logistic loss is 0.3605, which has an improvement of 100% over PR applied to the same model, while the precision improvement of the same model on the Netflix dataset is only about 10%. What's more, on the aspect of serendipity, the gap between PR and SPR on Yahoo! Music is even wider. PR has very low performance on serendipity, with a $Srdp@5$ of 0.0010 in the basic SVD model with logistic loss, which is not comparable to that of SPR in the same model. The reason is that the Yahoo! Music dataset has a larger popularity bias than the Netflix dataset (see Figure 1). The traditional personalized ranking method will bias towards the popular items because most of the observed items are popular, making the recommendations lowly unexpected (i.e., most recommended items are contained in PM). By adding a larger popularity weight to those training cases (u, i, j) in which j is popular, the popular (easily expected) items are penalized, thus SPR can recommend more serendipitous items in the top-N list.



(a) impact of α on P@5 (b) impact of α on Srdp@5

Figure 5. Impact of popularity weight α on accuracy and serendipity on Netflix



(a) impact of α on P@5 (b) impact of α on Srdp@5

Figure 6. Impact of popularity weight α on accuracy and serendipity on Yahoo! Music

We conclude that our proposed serendipitous personalized ranking method is a win-win, improving both recommendation accuracy and serendipity without increasing computing complexity.

E. Impact of Popularity Weight

In this section we will discuss the impact of popularity weight α in Equation 5 on both recommendation accuracy and serendipity. In our experiments, α is tuned in the range $[0, 1]$. Figure 5 illustrates the impact of α on the Netflix dataset while Figure 6 shows the impact on the Yahoo! Music dataset. From Figure 5 and Figure 6 we can see that the optimal value of α with respect to $P@5$ is around 0.5 on both datasets. Too high or too low a value of α will lead to a decrease in precision. On the Netflix dataset, the poorest $P@5$ is achieved when α is set to 1 while on the Yahoo! Music dataset $P@5$ is worst when α is set to 0. This difference again reflects the different popularity bias in the two datasets.

Figure 5(b) illustrates the change of serendipity with α on the Netflix dataset, from which we can see that $Srdp@5$ increases with the growth of α in the range of $[0, 0.5]$ and reaches its best around 0.5. A larger popularity weight α will lead to a poorer performance on serendipity. Similar results are observed on the Yahoo! Music dataset in Figure 6(b). This observation informs us that there is a trade-off between popularity weight and serendipity. If the popularity weight α is too high, the model will be optimized to favor long tail items which may be not useful to the user, thus the usefulness will be reduced. If α is too low, the model will bias towards the popular items which are lowly

unexpected, thus the unexpectedness will be reduced. So in both situations, the recommendation serendipity will drop.

Noting that both $P@5$ and $Srdp@5$ achieve their best when α is around 0.5 , this states that our SPR method can optimize accuracy and serendipity simultaneously. Further, Figure 5 and Figure 6 clearly demonstrate the improvement on accuracy and serendipity when α is set to the value of 0.5 over the value of 0. Since an α value of 0 represents the performance of the traditional personalized ranking method; this again shows the effectiveness of our serendipitous personalized ranking method.

V. Conclusions

In this paper we propose a serendipitous personalized ranking method to explore the relation between recommendation accuracy and serendipity. Our method extends traditional personalized ranking methods by considering item popularity in AUC optimization. We apply it to the various variants of SVD models and conduct experiments to compare it with the traditional personalized ranking methods. The results show that our method improves both recommendation accuracy and serendipity significantly.

VI. Acknowledgement

Research of this paper is supported by grants from NSFC-RGC joint research project 60931160445.

References

[1] C. Anderson, "The long tail: Why the future of business is selling less of more," *Hyperion*, 2006.

[2] E.G., "It's still good to have gatekeepers," *The Economist*, 2011.

[3] S. Rendle, C. Freudenthaler, Z. Gantner, and S.-T. Lars, "Bpr: Bayesian personalized ranking from implicit feedback," in *Proceedings of the Twenty-Fifth Conference on Uncertainty in Artificial Intelligence*, ser. UAI '09, 2009, pp. 452–461.

[4] P. Cremonesi, Y. Koren, and R. Turrin, "Performance of recommender algorithms on top-n recommendation tasks," in *Proceedings of the fourth ACM conference on Recommender systems*, ser. RecSys '10, 2010, pp. 39–46.

[5] J. Herlocker, J. A. Konstan, and J. Riedl, "An empirical analysis of design choices in neighborhood-based collaborative filtering algorithms," *Inf. Retr.*, vol. 5, no. 4, pp. 287–310, Oct. 2002.

[6] B. Sarwar, G. Karypis, J. Konstan, and J. Reidl, "Item-based collaborative filtering recommendation algorithms," in *Proceedings of the 10th international conference on World Wide Web*, ser. WWW '01, 2001, pp. 285–295.

[7] T. Hofmann, "Latent semantic models for collaborative filtering," *ACM Trans. Inf. Syst.*, vol. 22, pp. 89–115, January 2004.

[8] A. Paterek, "Improving regularized singular value decomposition for collaborative filtering," in *Proc. KDD Cup Workshop at SIGKDD'07, 13th ACM Int. Conf. on Knowledge Discovery and Data Mining*, pp. 39–42.

[9] Y. Koren, R. Bell, and C. Volinsky, "Matrix factorization techniques for recommender systems," *Computer*, vol. 42, no. 8, pp. 30 –37, aug. 2009.

[10] 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*, ser. KDD '08, 2008, pp. 426–434.

[11] J. Pessiot, T. Truong, N. Usunier, M. Amini, and P. Gallinari, "Learning to rank for collaborative filtering," in *9th International Conference on Enterprise Information Systems*. Citeseer, 2007.

[12] 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*, ser. SIGIR '08, 2008, pp. 83–90.

[13] M. Weimer, A. Karatzoglou, and A. Smola, "Improving maximum margin matrix factorization," *Mach. Learn.*, vol. 72, pp. 263–276, September 2008.

[14] S. M. McNee, J. Riedl, and J. A. Konstan, "Being accurate is not enough: how accuracy metrics have hurt recommender systems," in *CHI '06 extended abstracts on Human factors in computing systems*, ser. CHI EA '06, 2006, pp. 1097–1101.

[15] J. L. Herlocker, J. A. Konstan, L. G. Terveen, and J. T. Riedl, "Evaluating collaborative filtering recommender systems," *ACM Trans. Inf. Syst.*, vol. 22, pp. 5–53, January 2004.

[16] O. Celma and P. Herrera, "A new approach to evaluating novel recommendations," in *Proceedings of the 2008 ACM conference on Recommender systems*, ser. RecSys '08, 2008, pp. 179–186.

[17] K. Onuma, H. Tong, and C. Faloutsos, "Tangent: a novel, 'surprise me', recommendation algorithm," in *Proceedings of the 15th ACM SIGKDD international conference on Knowledge discovery and data mining*, ser. KDD '09, 2009, pp. 657–666.

[18] N. Kawamae, "Serendipitous recommendations via innovators," in *Proceedings of the 33rd international ACM SIGIR conference on Research and development in information retrieval*, ser. SIGIR '10, 2010, pp. 218–225.

[19] T. Murakami, K. Mori, and R. Orihara, "Metrics for evaluating the serendipity of recommendation lists," in *Proceedings of the 2007 conference on New frontiers in artificial intelligence*, ser. JSAI'07, 2008, pp. 40–46.

[20] M. Ge, C. Delgado-Battenfeld, and D. Jannach, "Beyond accuracy: evaluating recommender systems by coverage and serendipity," in *Proceedings of the fourth ACM conference on Recommender systems*, ser. RecSys '10, 2010, pp. 257–260.

[21] P. Adamopoulos and A. Tuzhilin, "On unexpectedness in recommender systems: Or how to expect the unexpected," in *Workshop on Novelty and Diversity in Recommender System at RecSys 2011*, 2011.

[22] N. Hurley and M. Zhang, "Novelty and diversity in top-n recommendation – analysis and evaluation," *ACM Trans. Internet Technol.*, vol. 10, no. 4, pp. 14:1–14:30, Mar. 2011.

[23] S. Vargas and P. Castells, "Rank and relevance in novelty and diversity metrics for recommender systems," in *RecSys*, ser. RecSys '11. New York, NY, USA: ACM, 2011, pp. 109–116.

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

[25] C. nicolas Ziegler and S. M. Mcnee, "Improving recommendation lists through topic diversification," in *Proceedings of the 14th international conference on World Wide Web*. ACM Press, 2005, pp. 22–32.

[26] Y.-J. Park and A. Tuzhilin, "The long tail of recommender systems and how to leverage it," in *Proceedings of the 2008 ACM conference on Recommender systems*, ser. RecSys '08, 2008, pp. 11–18.

[27] H. Steck, "Item popularity and recommendation accuracy," in *Proceedings of the fifth ACM conference on Recommender systems*, ser. RecSys '11, 2011, pp. 125–132.

[28] ——, "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*, ser. KDD '10. New York, NY, USA: ACM, 2010, pp. 713–722.

[29] A. P. Bradley, "The use of the area under the roc curve in the evaluation of machine learning algorithms," *Pattern Recognition*, vol. 30, pp. 1145–1159, 1997.

[30] J. Bennett and S. Lanning, "The Netflix Prize," in *Workshop at SIGKDD-07, ACM Conference on Knowledge Discovery and Data Mining*, 2007.

[31] G. Dror, N. Koenigstein, Y. Koren, and M. Weimer, "The Yahoo! Music Dataset and KDD-Cup'11," in *KDD-Cup Workshop 2011*, 2011.