A Monte Carlo Algorithm for Cold Start Recommendation

Yu Rong
The Chinese University of Hong Kong
yrong@se.cuhk.edu.hk

Xiao Wen
The Chinese University of Hong Kong
wenx@se.cuhk.edu.hk

Hong Cheng
The Chinese University of Hong Kong
hcheng@se.cuhk.edu.hk

ABSTRACT

Recommendation systems have been widely used in E-commerce sites, social networks, etc. One of the core tasks in recommendation systems is to predict the users' ratings on items. Although many models and algorithms have been proposed, how to make accurate prediction for new users with extremely few rating records still remains a big challenge, which is called the cold start problem. Many existing methods utilize additional information, such as social graphs, to cope with the cold start problem. However, the side information may not always be available. In contrast to such methods, we propose a more general solution to address the cold start problem based on the observed user rating records only. Specifically we define a random walk on a bipartite graph of users and items to simulate the preference propagation among users, in order to alleviate the data sparsity problem for cold start users. Then we propose a Monte Carlo algorithm to estimate the similarity between different users. This algorithm takes a precomputation approach, and thus can efficiently compute the user similarity given any new user for rating prediction. In addition, our algorithm can easily handle dynamic updates and can be parallelized naturally, which are crucial for large recommendation systems. Theoretical analysis is presented to demonstrate the efficiency and effectiveness of our algorithm, and extensive experiments also confirm our theoretical findings.

Categories and Subject Descriptors

H.3.3 [**Information Search and Retrieval**]: Information filtering

General Terms

Algorithms, Experimentation

Keywords

Preference propagation; random walk on bipartite graph; Monte Carlo simulation; cold start

WWW'14, April 7–11, 2014, Seoul, Korea.
ACM 978-1-4503-2744-2/14/04.
http://dx.doi.org/10.1145/2566486.2567978.

1. INTRODUCTION

With the huge volume of information available on the Web, it is necessary to help users filter overloaded data and extract relevant information. As a result, recommendation systems have been widely used in E-commerce sites, social networks, etc. An important recommendation task is to predict the missing ratings according to users' historical rating records. One of the most popular recommendation frameworks to address this problem is Collaborative Filtering (CF). Generally, CF methods can be divided into memory-based approach [21, 22] and model-based approach [14, 15, 20]. Memory-based approach, such as user-based method [21] and item-based method [22], is now widely used in practice. In recent years, model-based approach, especially the latent factor model based on matrix factorization, has attracted a lot of attention in research. Many matrix factorization (MF) methods have been proposed [14, 15, 20].

However, challenges still remain in recommendation systems, such as the cold start problem, dynamic model update and the scalability issue. Cold start is one of the most common but crucial problems in the real-world recommendation tasks, which is defined as *how to provide accurate recommendations for new users.* Because new users often have very few records in the system, it is hard to guess their preferences given the insufficient information. While many MF methods [14, 15, 20] achieve very good performance in the warm start situation, they cannot obtain satisfactory prediction accuracy in the cold start situation due to data sparsity. What's more, they may cause overfitting. In order to solve the data sparsity problem, many models have been proposed [27, 8, 10, 17] which utilize additional information, such as social relationships, to help improve the prediction accuracy. However, in many applications, especially the E-commerce sites, lack of such side information limits the successful application of these models.

Furthermore, any recommendation systems should not be static. Users are constantly added into the system, so are the new ratings made by existing users. It is desirable that a recommendation system can efficiently deal with the dynamic changes for model update. However, the MF methods have to re-factor the whole rating matrix given any changes, which is time-consuming and impractical.

Motivated by the above issues, we propose a novel Monte Carlo algorithm which utilizes the preference propagation between users to enhance the recommendation performance. We model the ratings as a bipartite graph between users and items and construct a random walk on this graph. This random walk process can effectively simulate the propagation

of user's rating preference to other users, providing an estimate of the user similarity. We design an efficient Monte Carlo algorithm on this random walk process, which can precompute the similarity between existing users in a recommendation system as the model. Given any target user, the model can be easily applied for predicting his missing ratings. Our algorithm can solve the cold start problem very well, especially for the extremely cold start situation since the propagated preference of users can alleviate the data sparsity issue. What's more, our algorithm can easily handle dynamic updates and can be parallelized naturally, which are crucial for handling large recommendation systems.

The main contributions of this work are summarized as follows.

- We define a random walk process to simulate the preference propagation among users based on the rating information only. This can alleviate the data sparsity problem for cold start users.
- A Monte Carlo algorithm based on the random walk process is designed to estimate the user similarity. Our algorithm works independently of new target users, thus allows us to separate the model training phase and the rating prediction phase. In particular, the model can be trained offline first and applied subsequently for rating prediction in an efficient way.
- We provide theoretical analysis on the accuracy and convergence of our algorithm. Extensive experiments on real-world large data sets show the superiority of our method and confirm our theoretical findings.

The rest of the paper is organized as follows. We formally define the rating prediction problem and introduce preliminary concepts in Section 2. Our random walk model is introduced in Section 3. In Section 4 we propose a Monte Carlo algorithm to efficiently estimate the user similarity and provide theoretical analysis. Experimental results are presented in Section 5. We discuss related work in Section 6 and conclude our work in Section 7.

2. PRELIMINARY CONCEPTS

2.1 Problem Definition

In a recommendation system we have a user set $U = \{u_1, \ldots, u_m\}$ and an item set $O = \{o_1, \ldots, o_n\}$. For any user $u \in U$, the set of items rated by u is denoted as $O_u \subseteq O$. For any item $o \in O$, the set of users who have rated o is denoted as $U_o \subseteq U$. The rating of user u_i on item o_j is denoted by r_{ij}, which can be any real number, but is often represented by an integer in the range $[1, 5]$. We use a rating matrix R of size $m \times n$ to represent the ratings between the user set U and the item set O. We set $r_{ij} = 0$ in R if user u_i has not rated item o_j.

Definition 1. **(Rating Prediction)** Given the rating matrix R and the set of items O_{u_τ} rated by a target user u_τ, predict the ratings $r_{\tau j}$ of the other items $o_j \in O \setminus O_{u_\tau}$ by u_τ. The predicted rating is denoted by $\hat{r}_{\tau j}$.

There have been many approaches for solving the rating prediction problem in the literature. In this paper we adopt the user-based collaborative filtering approach [21] which uses the similarity between users to make the rating prediction. Specifically, for any user u, we define a similarity vector c_u to indicate the preference similarity of u to other users, the i-th element of which, denoted as $c_u(i)$, quantifies the degree that user u is similar to user $u_i \in U$. Then we use the weighted mean with user preference to make the prediction as follows:

$$\hat{r}_{\tau j} = \bar{r}_\tau - \bar{r}_a + \frac{\sum_{u_i \in U_{o_j}} c_\tau(i) \cdot r_{ij}}{\sum_{u_i \in U o_j} c_\tau(i)}. \quad (1)$$

In Equation (1), $\bar{r}_\tau$ is the average rating of user u_τ in O_{u_τ}, and $\bar{r}_a$ is the average of all available ratings in R. We observe that different users have different rating biases, for example, a critical user may give consistently low ratings to all items. Thus we remove the common rating bias $\bar{r}_a$ and add the individual bias $\bar{r}_\tau$ to improve the accuracy of rating prediction.

In the user-based collaborative filtering recommendation framework, the key issue is how to compute the similarity vector c_u for any user u for accurate rating prediction. Many existing studies rely on additional information, for example, social network [23, 9] or voting graph [2], to estimate the trust between users. However, such additional information may not always be available for the concerned user set. Thus we propose to estimate the user similarity vector from the available ratings without any side information. So our task is defined as follows.

Definition 2. **(Similarity Estimation)** Given the rating matrix R and the set of items O_{u_τ} rated by a target user u_τ, estimate the similarity vector c_τ of u_τ from R.

For a target user u_τ who has rated a large enough set of items O_{u_τ}, we can make accurate predictions for u_τ. However, for a new user who has very limited rating information available, making accurate predictions for such a cold start user remains a big challenge. In this work we focus on predicting the missing ratings of the cold start users.

2.2 Challenges and Intuitions

In this section we analyze the challenges of the cold start problem and describe the intuitions behind our approach.

The basic idea of the collaborative filtering framework is to find the most similar user/item as the recommendation based on the observed information from the target user u_τ [21, 22]. Unfortunately, this approach only utilizes a very small portion of information embedded in the rating matrix and may lead to poor prediction performance. The situation becomes worse for the cold start users who have very limited rating information. The essence of the cold start problem is the data sparsity. So the key issue to address the cold start problem is *how to make the most use of the available rating information*.

Instead of considering only the users who have common rated items with the target user u_τ, we can utilize more data to estimate similarity based on the preference propagation. Consider the following illustrative example as depicted in Figure 1. For the target user and two existing users A, B, the numbers in the circles represent their ratings on corresponding items. The task is to predict the target user's missing ratings on items $o_4, \ldots, o_{10}$. Traditional recommendation methods will regard user A as a similar user based

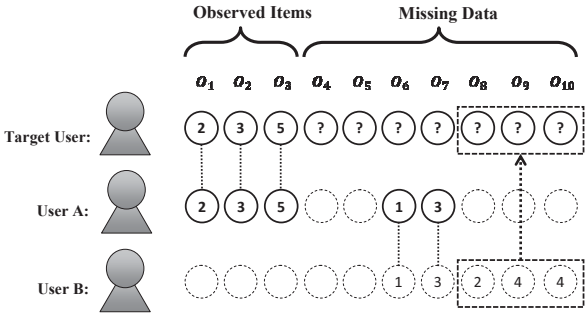

Figure 1: An example of rating prediction

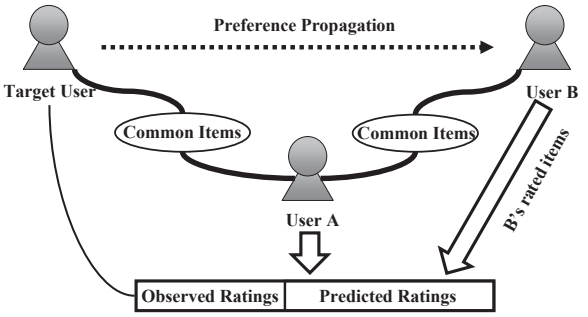

Figure 2: The user-item graph constructed from Figure 1

on his ratings on items o_1, o_2, o_3, and then predict the target user's ratings on o_6, o_7 by A's rating profile. However, the target user's ratings on items $o_8, \ldots, o_{10}$ still remain unknown under this simple similarity based recommendation.

From this example, a crucial observation we make is that besides these available ratings, we can infer the structural information of users and items if we model them as a bipartite graph. Consider the graph constructed between users and items as depicted in Figure 2. We can observe that the target user and user A have similar ratings on a subset of items, and users A, B also have similar ratings on another subset of items. Although the target user and user B may not have any connections in the rating system, it is reasonable to propagate the preference from the target user to B, and exploit B's ratings to make predictions for the target user on items $o_8, \ldots, o_{10}$. That is, we can estimate the similarity between the target user and user B by the preference propagation over the user-item graph. This example shows that we can infer more information from the bipartite graph for more effective rating prediction, especially for cold start users with very few rating records.

3. RANDOM WALK ON BIPARTITE GRAPH

We model the users, the items and the ratings as an *undirected weighted bipartite graph* $G = (U, O, E_R)$, where the users U and the items O form two sets of vertices, and E_R denotes the edge set. There exists an edge between user u_i and item o_j if $r_{ij} \neq 0$. We define a random walk on the bipartite graph G to model the preference propagation. Analogous to personalized PageRank [11] in which the stationary distribution represents the importance of web pages, in our model the stationary distribution reflects the similarity between users, which is expressed by the similarity vector.

3.1 Random Walk Construction

As the stationary distribution in a bipartite graph is related to the parity of number of steps [16], we force our random walk to start from and end at user nodes in G, assuring that we obtain the stationary distribution over the user set. Thus we construct an even-length random walk process $\{Z_t\}_{t\geq 0}$ on G from user to user to estimate the similarity vector for any single user. $\{Z_t\}_{t\geq 0}$ is comprised of two types of walks on G:

Type 1. From user u_i to item o_j. Type 1 walk reflects a user's preference over different items. Similar to the random walk on the homogeneous graph, we assume that a user chooses his rated items uniformly. Thus, the transition probability of type 1 walk is defined as:

$$P(u_i \to o_j) = \frac{1}{|O_{u_i}|}, \tag{2}$$

where $o_j \in O_{u_i}$.

Type 2. From item o_j to user u_k. Type 2 walk indicates the transition from an item to a user. Based on the idea of *preference propagation*, there is a larger chance to jump to a user u_k who gives a more similar rating on o_j as the user u_i in the previous step of the random walk process. Thus we define the transition probability of type 2 walk as:

$$P(o_j \to u_k) = \frac{sim(r_{ij}, r_{kj})}{\sum_{u_p \in U_{o_j}} sim(r_{ij}, r_{pj})}, \tag{3}$$

where $o_j \in O_{u_k}$, and $sim(r_{ij}, r_{kj})$ measures the similarity between ratings r_{ij} and r_{kj}. In this paper, we measure the rating similarity by

$$sim(r_{ij}, r_{kj}) = MAXSCORE - |r_{ij} - r_{kj}|, \tag{4}$$

where $MAXSCORE$ is the maximum rating score in the system.

Next we define the transition probability between two users based on length-2 walk. For a length-2 walk from user u_i to user u_k connected by item o_j, we can write the transition probability from u_i to u_k as:

$$p(u_k|u_i, o_j) = P(u_i \to o_j)P(o_j \to u_k). \tag{5}$$

As there can be more than one length-2 path between two users, we sum up the probability of each such path as the transition probability from u_i to u_k:

$$\begin{aligned} p_{i,k} &= \sum_{o_j \in \xi} p(u_k|u_i, o_j) \\ \xi &= O_{u_i} \cap O_{u_k}. \end{aligned} \tag{6}$$

Based on this definition of transition probability, we denote the transition matrix between users as P. Then we define the random walk process $\{Z_t\}_{t\geq 0}$ as follows:

Definition 3. **(Random Walk on Bipartite Graph)** $\{Z_t\}_{t\geq 0}$ begins with the user u_i, i.e., $u_{t=0} = u_i$. At each step this random walk terminates with probability α, or makes the transition between two users according to the matrix P with probability $1 - \alpha$. Each step contains two types of transitions: a type 1 walk from a user to an item according to (2), followed by a type 2 walk from the item to another user according to (3).

4. SAMPLING ALGORITHM

In this section we present the sampling algorithm to estimate the stationary distribution of $\{Z_t\}_{t\geq 0}$. For convenience, we permute the target user u_τ to the first place in matrix P and denote the stationary distribution (similarity vector) as c_τ. We call $u \in U \setminus \{u_\tau\}$ a training user.

4.1 Online Sampling Algorithm

First we propose an online sampling algorithm for a target user. In $\{Z_t\}_{t\geq 0}$, the transition matrix $\tilde{P}$ is:

$$\tilde{P} = (1-\alpha)P + \alpha \vec{1}_{m\times 1} c_0. \tag{7}$$

The stationary distribution c_τ satisfies the following equality:

$$c_\tau = c_\tau \tilde{P}. \tag{8}$$

According to (7) and (8), we have:

$$c_\tau = (1-\alpha)c_\tau P + \alpha c_0. \tag{9}$$

In (7), $c_0 = [1, \vec{0}_{1\times(m-1)}]$ is the initial distribution determined by the target user u_τ, because $\{Z_t\}_{t\geq 0}$ always starts from u_τ. Inspired from [3], we design the basic Monte Carlo algorithm to estimate the similarity vector c_τ of u_τ as follows.

ALGORITHM 1. *Simulate N runs of random walk process $\{Z_t\}_{t\geq 0}$ starting from the target user u_τ. Evaluate $c_\tau(j)$ as the fraction of N random walks which end at user $j = 1, \ldots, m$.*

Let $\hat{c}_{\tau,N}(j)$ be the estimator of $c_\tau(j)$ obtained by Algorithm 1. According to [3]:

$$\begin{aligned} E(\hat{c}_{\tau,N}(j)) &= c_\tau(j) \\ Var(\hat{c}_{\tau,N}(j)) &= N^{-1} c_\tau(j)(1-c_\tau(j)). \end{aligned} \tag{10}$$

To guarantee the precision of the estimation, N should be of the order m^2 [3]. But in practice, m^2 rounds of simulations are not necessary. In most cases $O(m)$ rounds are enough for accurate estimation.

4.2 Monte Carlo Algorithm with Precomputation

In the previous section we have introduced the online Monte Carlo algorithm (Algorithm 1) to estimate the similarity vector for the target user. Although this algorithm is quite intuitive and easy to implement, it is impractical in reality, because we need to simulate N runs of random walk process for every target user to estimate his similarity vector. When the number of target users is very large, the random walk simulation is very time-consuming. Thus we design another Monte Carlo algorithm with a precomputation approach to build a model to estimate any target users' similarity vector.

First, we divide the matrix P into four blocks in the following form:

$$P = \begin{bmatrix} \pi_{\tau\tau} & \pi_{\tau\cdot} \\ \pi_{\cdot\tau}^T & P_* \end{bmatrix}. \tag{11}$$

In (11), $\pi_{\tau\cdot}$ is a $1\times(m-1)$ transition vector from the target user u_τ to the training users, and $\pi_{\cdot\tau}$ is a $1\times(m-1)$ transition vector from the training users to the target user u_τ, $\pi_{\tau\tau}$ is the transition probability from u_τ to himself, P_* is the transition matrix of the training user set. Here we set $\pi_{\tau\tau} = 0$, $\pi_{\cdot\tau} = \vec{0}_{1\times(m-1)}$ by approximation. With this, we simply do not jump back to the target user in a random walk. This approximation allows us to separate the training users from the target user, and thus we can precompute the similarity between the training users independently of the target user. This approximation is also reasonable in the sense that we do not need the similarity of user u_τ to himself for rating prediction.

From (9), we can obtain the stationary distribution c:

$$c = \alpha c_0 (\tilde{I} - (1-\alpha)P)^{-1}, \tag{12}$$

where $\tilde{I}$ is an $m\times m$ identity matrix. According to (11), we can get a closed form of $(\tilde{I} - (1-\alpha)P)^{-1}$:

$$(\tilde{I} - (1-\alpha)P)^{-1} = \begin{bmatrix} 1 & (1-\alpha)\pi_{\tau\cdot}(I-(1-\alpha)P_*)^{-1} \\ \vec{0}^T & (I-(1-\alpha)P_*)^{-1} \end{bmatrix}, \tag{13}$$

where I is an $(m-1)\times(m-1)$ identity matrix. So the stationary distribution can be written as:

$$c = \alpha[1, (1-\alpha)\pi_{\tau\cdot}(I-(1-\alpha)P_*)^{-1}]. \tag{14}$$

Since the first component of c is the similarity to the target user himself, we only need the last $(m-1)$ components of c, which correspond to the target user's similarity c_τ to the $(m-1)$ training users.

After normalization, we have

$$c_\tau = \alpha \pi_{\tau\cdot}(I-(1-\alpha)P_*)^{-1}. \tag{15}$$

In (15), how to estimate $(I-(1-\alpha)P_*)^{-1}$ is the key problem. Note that for all $i, j = 1, \ldots, m-1$, the element w_{ij} of the matrix

$$W = (I-(1-\alpha)P_*)^{-1} = \sum_{k=0}^{\infty}(1-\alpha)^k (P_*)^k \tag{16}$$

can be regarded as the average number of times that the random walk $\{Z_t\}_{t\geq 0}$ visits a user j given that this random walk starts at user i. Denote

$$w_{\cdot j} = \sum_{i=1}^{m-1} \pi_{\tau i} w_{ij}. \tag{17}$$

So

$$c_\tau(j) = \alpha w_{\cdot j}. \tag{18}$$

Thus we can propose an estimator based on a complete path of the random walk $\{Z_t\}_{t\geq 0}$.

ALGORITHM 2. ***MC Complete Path Algorithm*** *This algorithm consists of three stages:*

1. ***Precomputation stage:*** *Simulate the random walk $\{Z_t\}_{t\geq 0}$ exactly N times from each training user. For any user i, evaluate w_{ij} as the average number of visits to user j given that the random walk starts from i, denoted as $\bar{w}_{ij}$.*

2. ***Similarity estimation stage:*** *For any target user u_τ, we calculate $\pi_{\tau\cdot}$ by enumerating all the paths from u_τ to the training users. Then we estimate $c_\tau(j)$ as*

$$\hat{c}_\tau(j) = \alpha \sum_{i=1}^{m-1} \pi_{\tau i} \bar{w}_{ij}. \tag{19}$$

3. ***Rating prediction stage:*** *For the target user u_τ, the estimated rating on item o_j is*

$$\hat{r}_{\tau j} = \bar{r}_\tau - \bar{r}_a + \frac{\sum_{u_i \in U_{o_j}} \hat{c}_\tau(i) \cdot r_{ij}}{\sum_{u_i \in U_{o_j}} \hat{c}_\tau(i)}. \quad (20)$$

Different from Algorithm 1, Algorithm 2 precomputes the matrix $(I-(1-\alpha)P_*)^{-1}$ in (15) by Monte Carlo simulation on the training set in stage 1. This precomputation can be done offline before any target user arrives for rating prediction. Given any target user, we only need to compute $\pi_{\tau\cdot}$ to estimate the similarity vector c_τ. The precomputation result in stage 1 can be reused for all target users for rating prediction. Thus the computational cost can be reduced remarkably.

4.2.1 Theoretical Analysis

The key issue in Algorithm 2 is how many rounds to simulate for each training user in stage 1 to guarantee the estimation accuracy. In this section we will provide a detailed theoretical analysis on: (1) the accuracy of the estimator, and (2) the simulation round to bound the error of the estimator.

Expectation and Variance of $\hat{c}_\tau(j)$.

Definition 4. Let W_{ij} be a random variable distributed as the average number of visits to user $j = 1, \ldots, m-1$ by the random walk $\{Z_t\}_{t\geq 0}$ given that the random walk initiates at user $i = 1, \ldots, m-1$. Formally,

$$P(W_{ij} = x) = P\left(\left[\sum_{t=0}^{\infty} \mathbf{1}_{\{Z_t = j\}}\right] = x | Z_0 = i\right), x = 0, 1, \ldots,$$

where $\mathbf{1}_{\{\cdot\}}$ is the indicator function.

Then let $W_{ij}^{(l)}, l \geq 1$, be independent random variables distributed as W_{ij}. The estimator produced by Algorithm 2 can be written as

$$\hat{c}_\tau(j) = \frac{\alpha}{N} \sum_{l=1}^{N} \sum_{i=1}^{m-1} \pi_{\tau i} W_{ij}^{(l)}.$$

For $i, j = 1, \ldots, m-1$, define

$$\begin{aligned} \bar{W}_{ij} &= \frac{1}{N} \sum_{l=1}^{N} W_{ij}^{(l)} \\ \bar{W}_{\cdot j} &= \sum_{i=1}^{m-1} \pi_{\tau i} \bar{W}_{ij}, \end{aligned}$$

then the estimator can be rewritten as

$$\hat{c}_\tau(j) = \alpha \bar{W}_{\cdot j}.$$

Similar to the analysis in [3], we have

$$\begin{aligned} E(W_{ij}) &= w_{ij} = \frac{q_{ij}}{1-q_{jj}} \\ Var(W_{ij}) &= \frac{1+q_{jj}}{1-q_{jj}} w_{ij} - w_{ij}^2, \end{aligned}$$

where q_{ij} is the probability that starting from user i, the random walk $\{Z_t\}_{t\geq 0}$ reaches user j, and it is obvious $q_{jj} \leq 1-\alpha$ for all j.

Assuming that all W_{ij}'s are independent, we obtain

$$\begin{aligned} E(\hat{c}_\tau(j)) &= E(\alpha \bar{W}_{\cdot j}) = \alpha E(\sum_{i=1}^{m-1} \pi_{\tau i} \bar{W}_{ij}) \\ &= \alpha \sum_{i=1}^{m-1} \pi_{\tau i} E(\bar{W}_{ij}) = \alpha \sum_{i=1}^{m-1} \pi_{\tau i} w_{ij} \\ &= c_\tau(j) \end{aligned} \quad (21)$$

and

$$\begin{aligned} Var(\hat{c}_\tau(j)) &= Var(\alpha \bar{W}_{\cdot j}) \\ &= \alpha^2 Var(\sum_{i=1}^{m-1} \pi_{\tau i} \bar{W}_{ij}) \\ &= \alpha^2 \sum_{i=1}^{m-1} \pi_{\tau i}^2 Var(\bar{W}_{ij}) \\ &= \frac{\alpha^2}{N} \sum_{i=1}^{m-1} \pi_{\tau i}^2 (\frac{1+q_{jj}}{1-q_{jj}} w_{ij} - w_{ij}^2) \\ &\leq \frac{\alpha^2}{N} \max_i(\pi_{\tau i}) \cdot \frac{1+q_{jj}}{1-q_{jj}} w_{\cdot j} \\ &\leq \frac{\alpha^2}{N} \max_i(\pi_{\tau i}) \cdot \frac{2-\alpha}{\alpha} w_{\cdot j} \\ &\leq \frac{2-\alpha}{N} c_\tau(j) \end{aligned} \quad (22)$$

(21) shows that our estimator by Algorithm 2 is unbiased. In the following we will use (22) to bound the simulation round N by the central limit theorem.

Estimation of Simulation Round N.
We have the following theorem with proof to bound the simulation round N.

THEOREM 1. *If we run $N = (2-\alpha)\frac{x_{1-\beta/2}^2}{\epsilon^2 c_\tau(j)}$ rounds of random walk for each training user, then with probability $1-\beta$, the estimator $\hat{c}_\tau(j)$ output by Algorithm 2 satisfies*

$$|\hat{c}_\tau(j) - c_\tau(j)| < \epsilon c_\tau(j),$$

where α is the restart probability, $c_\tau(j)$ is the target user's similarity to training user u_j, and $x_{1-\beta/2}$ is $(1-\beta/2)$-quantile of the standard normal distribution.

PROOF. Consider the confidence interval for $\hat{c}_\tau(j)$ defined as

$$P(|\hat{c}_\tau(j) - c_\tau(j)| < \epsilon c_\tau(j)) \geq 1-\beta.$$

Since $\hat{c}_\tau(j)$ is a sum of a large number of terms, the random variable $[\hat{c}_\tau(j) - c_\tau(j)]/\sqrt{Var(\hat{c}_\tau(j))}$ has approximately a standard normal distribution. Thus, we deduce

$$\epsilon c_\tau(j)/\sqrt{Var(\hat{c}_\tau(j))} \geq x_{1-\beta/2}$$

which results in

$$N \geq (2-\alpha)\frac{x_{1-\beta/2}^2}{\epsilon^2 c_\tau(j)},$$

where $x_{1-\beta/2}$ is $(1-\beta/2)$-quantile of the standard normal distribution. □

According to Theorem 1, given ϵ and β, the simulation round N is related to $c_\tau(j)$ and the restart probability α.

With a larger α, we can achieve the same accuracy with fewer simulation rounds. From (14), we know that the stationary distribution will be affected by α. Especially when $\alpha = 1$, W becomes an identity matrix according to (16), so no simulation is needed.

On the other hand, $c_\tau(j)$ affects N in a way that smaller $c_\tau(j)$ leads to more simulations. It takes more simulations to estimate $c_\tau(j)$ when its value is small. However, from (20), the users who are more similar to the target user have more influence on the rating prediction, so it is important for us to obtain an accurate estimation for those users who have very high similarity value. A reasonable assumption is that we should care more about those users whose similarity is larger than the average, i.e., $c_\tau(j) \geq \frac{1}{m-1}$. Under this circumstance we have

$$\begin{aligned} N &= (2-\alpha)\frac{x^2_{1-\beta/2}}{\epsilon^2 c_\tau(j)} \\ &\leq \frac{(2-\alpha)x^2_{1-\beta/2}(m-1)}{\epsilon^2}, \end{aligned}$$

which means we only conduct $O(m)$ rounds of simulations for each training user to achieve the given accuracy.

4.3 Extensions

4.3.1 Parallel Implementation

One of the advantages of the Monte Carlo simulation is that simulations are independent of each other, thus we can easily parallelize our precomputation algorithm in a shared-memory environment. Assume that we have t processors, and the number of training users is m. The training users are evenly distributed across the processors and each processor is assigned m/t training users for the Monte Carlo simulation. Each processor performs $O(T_{seq}/t)$ amount of computation, where T_{seq} is the time needed to solve the same problem on a single processor. Thus Algorithm 2 can achieve a speedup of t using t processors in parallel.

4.3.2 Dynamic Updates

Owing to the Monte Carlo method, our approach can be easily extended to handle dynamic updates in the recommendation system. A common scenario is that new users along with their ratings are added who can serve as additional training users. Instead of recomputing the model from scratch, we treat a new training user as a target user and compute the stationary distribution based on the original model to approximate the new model. This approximation treats the new training users as having no connections with each other. For a small number of new training users, this approximation works well.

5. EXPERIMENTS

5.1 Data Set Description

To evaluate the performance and robustness of our algorithm, we conduct a series of experiments on five real-world data sets: MovieLens-1M[1] (ML-1M), Epinions [18], Bookcrossing [28] (Bc), Amazon[2] and Yahoo! Music R1[3]. The ratings in the first four data sets are real numbers in the range $[1, 5]$, while the ratings in Yahoo! Music are integers in the range $[1, 100]$. Table 1 shows the statistics of these data sets.

[1]http://www.grouplens.org/datasets/movielens/
[2]http://memetracker.org/
[3]http://webscope.sandbox.yahoo.com/

Table 1: Statistics of The Data Sets

	#users	#items	#ratings	avg density
ML-1M	6040	3952	1,000,209	4.19%
Epinions	40,163	139,738	664,823	0.012%
Bc	105,274	340,545	1,149,766	0.0032%
Amazon	1,555,170	402,724	6,359,182	0.001%
Yahoo!	1,948,882	1,101,749	81,113,494	0.0038%

5.1.1 Experimental Configuration

We perform the test using 4-fold cross validation to reduce the influence of sampling. For each test user, we split his ratings into two parts: *observed items* and *held-out items*. Our task is to use the ratings of the observed items to predict the ratings of the held-out items. We use the split ratio of 10 : 90, i.e., for any test user, we use 10% of the items rated by the user to predict the ratings of the remaining 90% items. Since we are particularly interested in the cold start problem, we divide the test users into three subsets based on the number of their observed items. Table 2 shows the configuration in the experiment, and Table 3 reports the percentage of test users in different settings for the first four smaller data sets. The experiment on the largest data set, Yahoo! Music, is reported in Section 5.2.3.

Table 2: Experimental Configuration

	# of observed item
Extremely cold start users (D_1)	≤ 3
Cold start users (D_4)	$[4, 9]$
Warm start users (D_{10})	≥ 10

Table 3: Percentage of Different Types of Test Users

	D_1	D_4	D_{10}
ML-1M	13.36%	35.04%	51.60%
Epinions	44.42%	46.75%	8.83%
Bc	32.30%	41.98%	25.72%
Amazon	45.23%	41.62%	13.15%

5.1.2 Evaluation Metrics

We use *Mean Absolute Error* as the evaluation metric to measure the prediction quality. This metric has been widely used in the recommendation problem [7]:

$$MAE_i = \sum_{j \in O_{u_i}} \frac{|r_{ij} - \hat{r}_{ij}|}{|O_{u_i}|}, \tag{23}$$

where r_{ij} and $\hat{r}_{ij}$ are the true rating and predicted rating of item o_j by user u_i, respectively. We report the average MAE on the test set.

5.1.3 Candidate Methods for Comparison

Our method Monte Carlo Complete Path algorithm is denoted as **MCCP**. We compare it with the following state-of-the-art recommendation methods.

Table 4: MAE(Smaller is Better) Comparison with Other Algorithms

	ML-1M				Epinions			
	D_1	D_4	D_{10}	Overall	D_1	D_4	D_{10}	Overall
MCCP	**0.7823**	**0.7668**	**0.7027**	**0.7117**	**0.8116**	**0.7983**	**0.7749**	**0.7946**
SVD++	0.8067	0.7793	0.7272	0.7346	0.8859	0.8764	0.8626	0.8746
CDTF	0.8716	0.8182	0.7624	0.7708	0.9509	0.9434	0.9174	0.9375
LFL	0.8100	0.7892	0.7609	0.7649	1.3435	1.3222	1.2600	1.3091
	Bookcrossing				Amazon			
	D_1	D_4	D_{10}	Overall	D_1	D_4	D_{10}	Overall
MCCP	**1.3682**	**1.2928**	**1.0026**	**1.0938**	**0.8013**	**0.7670**	**0.6700**	**0.7234**
SVD++	1.4664	1.4093	1.0488	1.1585	0.9324	0.8876	0.7605	0.8301
CDTF	1.4741	1.4172	1.1571	1.2375	-	-	-	-
LFL	1.5884	1.6457	1.6816	1.6666	1.3730	1.3410	1.1526	1.2455

- **SVD++**. This is a latent factor model which combines the matrix factorization technique with implicit feedback from the users [13].
- **CDTF**. This is a generalized Cross Domain Triadic Factorization over the triadic relation user-item-domain [8].
- **LFL**. This is a latent feature log-linear model for the dyadic prediction task [19].

To provide the sufficient samples to do the evaluation, in each data set, we select users who have rated more than 20 items, and items which have been rated by at least 5 users in the test set.

5.2 Experimental Results

5.2.1 Rating Prediction Performance

Table 4 presents the MAE results of all methods on the first four data sets. The column "Overall" reports the MAE performance on the whole test user set, i.e., $D_1 \cup D_4 \cup D_{10}$. Note that CDTF cannot terminate in 48 hours on Amazon. For our method we set the restart probability $\alpha = 0.8$ and the number of simulations $N = m$ according to our theoretical analysis on convergence in Section 4.2.1. As shown in Table 4, our method MCCP consistently outperforms the other methods on all four data sets in both the cold start and warm start settings. This indicates that our method is very effective for recommendation on sparse data sets. Among the three comparison methods, SVD++ performs better than CDTF and LFL, and LFL has the worst prediction performance.

5.2.2 Impact of Parameters

Our method has two parameters: the restart probability α and the number of simulations N. In this section we will investigate the sensitivity of the rating prediction performance w.r.t. α and N, respectively.

The restart probability α: We use ML-1M to study the impact of the restart probability α. We fix the simulation round $N = m$, and vary α from 0.1 to 1.0 linearly. For each configuration, we run 4-fold cross validation. Figure 3 presents the average MAE on the whole test user set, i.e., $D_1 \cup D_4 \cup D_{10}$, under different restart probability α. When α increases from 0.1 to 0.9, the MAE becomes lower, which means a better recommendation performance. This result

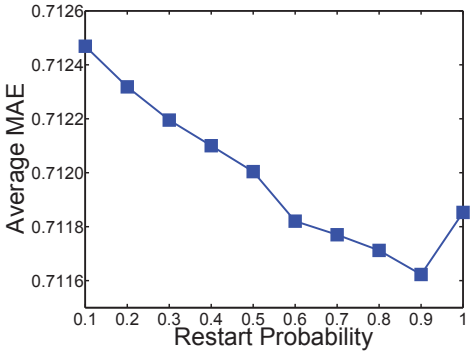

Figure 3: Parameter sensitivity w.r.t. α

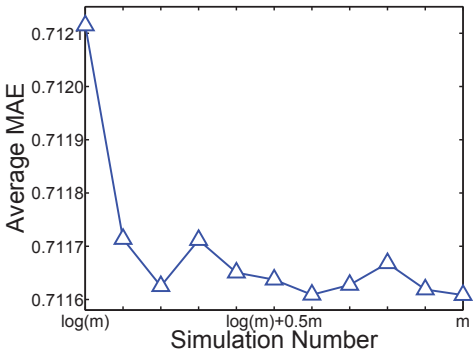

Figure 4: Parameter sensitivity w.r.t. N

indicates that in reality, the influence of preference propagation for rating prediction decays quickly in a small number of hops. This also validates the conclusion that a larger restart probability leads to fewer simulation rounds for a given error tolerance in our theoretical analysis. But when $\alpha = 1.0$, the MAE increases again indicating a worse performance. This is because when $\alpha = 1.0$, there is no random walk simulation between training users to propagate the preference among them. Thus the algorithm performs rating prediction only based on the transition from the target user to training users through their commonly rated items. The performance deterioration is most notable on the extremely cold start situation (D_1), when a target user has no more than 3 observed ratings. This confirms the effectiveness of the propagated

preference for cold start recommendation from a different perspective. Finally, we observe that the maximum MAE difference under different restart probability is only 0.0008, which indicates that our approach is very robust to the variation of restart probability.

The number of simulations N: We evaluate the impact of the simulation round on the prediction performance. We fix the restart probability $\alpha = 0.8$ and vary the number of simulation from $N = \log(m)$ to m linearly. Figure 4 shows the average MAE under different rounds of simulation. We can conclude that our method performs already well even with $N = \log(m)$ rounds of simulation. More simulation rounds yield a little improvement on the prediction accuracy. This result shows that the theoretical bound on convergence we analyze in Section 4.2.1 is quite conservative in the real application.

5.2.3 Scalability Test

As mentioned in Section 4.3.1, our method can be parallelized naturally, which makes our algorithm suitable for handling big data. We conduct an experiment on the Yahoo! Music data set, one of the largest open data sets on user-item ratings, to evaluate the scalability of our approach. Our precomputation algorithm (MCCP) is implemented in Matlab and C++. We run 4 Matlab processes and each process runs 10 threads using the Parallel Computing Toolbox in Matlab. The experiment is conducted on a Windows server with an Intel Xeon 2.4GHz CPU and 384GB memory. For the testing process, we randomly select 2000 users in the test set, and set the split ratio as 5 : 95 to divide the ratings into *observed items* and *held-out items*. We follow the configuration in Table 2 to divide the test users into D_1, D_4, D_{10} based on the number of observed items. Due to the scale of this data set, most comparison methods cannot finish in reasonable time. So we only report the performance of SVD++ and our method MCCP.

Table 5: MAE on Yahoo! Music

	D_1	D_4	D_{10}	Overall
MCCP	14.0961	14.6828	17.4747	14.9501
SVD++	13.6275	14.8153	15.7766	15.0996

Table 6: Total Training Time on Yahoo! Music

	Training Time (seconds)
MCCP	21,564
SVD++	85,918

Table 5 presents the MAE of the predicted ratings in the range $[1, 100]$. We can see that the MAE of SVD++ is slightly lower than that of MCCP in extremely cold start users (D_1) and warm start users (D_{10}), while MCCP achieves a better performance in cold start users (D_4). MCCP is superior to SVD++ in the overall test set as well.

Table 6 shows the total model training time of MCCP and SVD++ for the three test configurations. MCCP performs model training by Monte Carlo simulation independently of the test set. For SVD++, we embed the three test sets D_1, D_4, D_{10} into the training set and train the model once. The training time of SVD++ is 4 times longer than that of MCCP. Importantly, MCCP is a precomputation algorithm which is independent of test users. Given a training set, it only needs to train a model once which can be subsequently applied on different test users. In contrast, SVD++ needs to embed the test set into the training process for matrix factorization. For the real world application, we need to make the recommendation for any new user when he enters the system. Our precomputation method can handle this situation much better than SVD++.

5.2.4 Handling Dynamic Updates

In this experiment we test the performance of our algorithm in a dynamic environment. To simulate the situation where new training users are added to the system, we divide the training set of the ML-1M data evenly into 10 parts. We use 1, 2, 3, 4, or 5 parts as *the new training set* respectively, and the remaining as *the original training set*. The new data ratio, defined as the percentage of new training users w.r.t. the original training users, varies from 11.1% to 100%. We evaluate the rating prediction performance of our incremental computing approach introduced in Section 4.3.2 given new training users. For comparison, we use the recomputing approach as a baseline, which simply combines the original training set and the new training set and re-runs the random walk simulation. Figure 5 reports the MAE scores for different test configurations.

We can see that MAE increases very little using the incremental computing approach, which demonstrates that the incremental approach can achieve almost the same result with the recomputing approach. Another observation is that, under all test configurations, the MAE of the incremental computing approach increases slightly with the increasing of the new data ratio. This is because the incremental approach simply treats the new training users as having no connections with each other, and thus ignores the preference propagation among them. When the user similarity among a large number of training users is not considered, the rating prediction performance will be compromised, especially for the cold start users. On the other hand, the prediction performance of the recomputing approach is not affected by the varying ratio of new training data, as it combines the original and new training sets (always equivalent to the 10 parts of training users) for model training.

6. RELATED WORK

In this section, we will review some existing works on recommendation and the cold start problem.

Memory based model. The memory based model is the classical model in Collaborative Filtering. Two early representatives are user-based [21] and item-based [22] models, which calculate the similarity between users or items. Since memory based models cannot handle the cold start problem well, a lot of follow-up studies try to refine the model to address the cold start problem. Some additional information, such as item taxonomy information [24], is used to refine the prediction results. Some other methods [4] define a new similarity measure to gain more reliable recommendation from users. The modification of the basic model, such as an error reflection model [12] can also enhance the prediction accuracy.

Latent factor model. Another group of very popular model in the past decade is the latent factor model based on matrix factorization, such as Singular Value Decomposition (SVD) [14], Non-negative Matrix Factorization (NMF) [15], Probabilistic Matrix Factorization (PMF) [20] and so on. There are a lot of studies addressing the cold start prob-

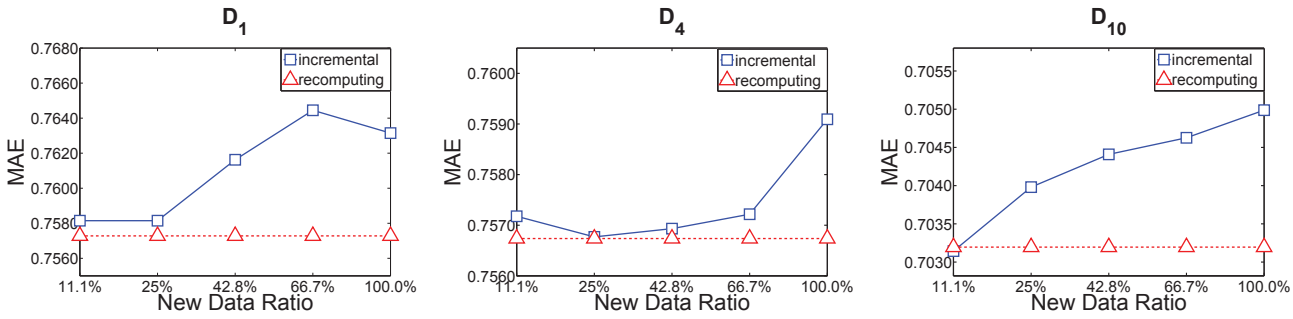

Figure 5: The performance comparison between dynamic update algorithm and original algorithm

lem based on this model. Some methods consider the side information such as user/item profiles [27], cross domain information [8] and social network [10, 17] to improve the prediction accuracy. Another way to enhance the prediction accuracy is combining the memory-based approach with the latent factor model [13]. Since learning the latent factor is very crucial in the latent factor model, some methods adopt different learning schema, such as LDA model [1], log-linear model [19] to pursue better and more reasonable learning results. Moreover, some permutation techniques are used for generating denser sub-matrix to make the matrix factorization more accurate and efficient [26].

Random sampling model. The most related work to our study is the random sampling model for recommendation. Some studies employ the random sampling technique on the side information, such as the trust network [9], item similarity graph [25], contextual graph [5], to refine prediction result. Another way is to use the random walk technique to measure similarities between users/items, which can overcome the sparsity problem [6] compared with the traditional measure in memory-based model. The major difference between our method and these random sampling methods is that, we do not use any side information for rating prediction. Instead we use the preference propagation solely based on the ratings to address the cold start problem.

7. CONCLUSION

In this paper, we study the cold start problem in the recommendation system. To overcome the data sparsity issue, we designed a random walk process on the bipartite graph to model the preference propagation among users. We proposed an efficient algorithm which learns the preference similarity between users in a precomputation approach, which can be efficiently applied for rating prediction on any new user. The nature of Monte Carlo simulation enables the parallel implementation of our algorithm. Theoretical analysis is provided to guarantee the efficiency and effectiveness of our algorithm. Extensive experimental results on real-world data sets also confirm our theoretical findings.

8. ACKNOWLEDGMENTS

This work is supported by the Hong Kong Research Grants Council (RGC) General Research Fund (GRF) Project No. CUHK 411211, 411310, and the Chinese University of Hong Kong Direct Grant No. 4055015.

9. REFERENCES

[1] D. Agarwal and B.-C. Chen. fLDA: Matrix factorization through latent dirichlet allocation. In *WSDM*, pages 91–100, 2010.

[2] R. Andersen, C. Borgs, J. Chayes, U. Feige, A. Flaxman, A. Kalai, V. Mirrokni, and M. Tennenholtz. Trust-based recommendation systems: An axiomatic approach. In *WWW*, pages 199–208, 2008.

[3] K. Avrachenkov, N. Litvak, D. Nemirovsky, and N. Osipova. Monte carlo methods in pagerank computation: When one iteration is sufficient. *SIAM Journal on Numerical Analysis*, 45(2):890–904, 2007.

[4] J. Bobadilla, F. Ortega, A. Hernando, and J. Bernal. A collaborative filtering approach to mitigate the new user cold start problem. *Knowledge-Based Systems*, 26:225–238, 2012.

[5] T. Bogers. Movie recommendation using random walks over the contextual graph. In *CARS Workshop*, 2010.

[6] F. Fouss, A. Pirotte, J.-M. Renders, and M. Saerens. Random-walk computation of similarities between nodes of a graph with application to collaborative recommendation. *IEEE Transaction on Knowledge and Data Engineering*, 19(3):355–369, 2007.

[7] J. L. Herlocker, J. A. Konstan, L. G. Terveen, and J. T. Riedl. Evaluating collaborative filtering recommender systems. *ACM Transactions on Information Systems*, 22(1):5–53, 2004.

[8] L. Hu, J. Cao, G. Xu, L. Cao, Z. Gu, and C. Zhu. Personalized recommendation via cross-domain triadic factorization. In *WWW*, pages 595–606, 2013.

[9] M. Jamali and M. Ester. Trustwalker: A random walk model for combining trust-based and item-based recommendation. In *KDD*, pages 397–406, 2009.

[10] M. Jamali and M. Ester. A matrix factorization technique with trust propagation for recommendation in social networks. In *RecSys*, pages 135–142, 2010.

[11] G. Jeh and J. Widom. Scaling personalized web search. In *WWW*, pages 271–279, 2003.

[12] H.-N. Kim, A. El-Saddik, and G.-S. Jo. Collaborative error-reflected models for cold-start recommender systems. *Decision Support Systems*, 51(3):519–531, 2011.

[13] Y. Koren. Factorization meets the neighborhood: a multifaceted collaborative filtering model. In *KDD*, pages 426–434, 2008.

[14] Y. Koren, R. Bell, and C. Volinsky. Matrix factorization techniques for recommender systems. *Computer*, 42(8):30–37, 2009.

[15] D. Lee and H. Seung. Algorithms for non-negative matrix factorization. In *NIPS*, pages 556–562, 2001.

[16] L. Lovász. Random walks on graphs: A survey. *Combinatorics, Paul erdos is eighty*, 2(1):1–46, 1993.

[17] H. Ma, H. Yang, M. R. Lyu, and I. King. Sorec: social recommendation using probabilistic matrix factorization. In *CIKM*, pages 931–940, 2008.

[18] P. Massa and P. Avesani. Trust-aware recommender systems. In *RecSys*, pages 17–24, 2007.

[19] A. K. Menon and C. Elkan. A log-linear model with latent features for dyadic prediction. In *ICDM*, pages 364–373, 2010.

[20] A. Mnih and R. Salakhutdinov. Probabilistic matrix factorization. In *NIPS*, pages 1257–1264, 2007.

[21] P. Resnick, N. Iacovou, M. Suchak, P. Bergstrom, and J. Riedl. Grouplens: an open architecture for collaborative filtering of netnews. In *CSCW*, pages 175–186, 1994.

[22] B. Sarwar, G. Karypis, J. Konstan, and J. Riedl. Item-based collaborative filtering recommendation algorithms. In *WWW*, pages 285–295, 2001.

[23] F. E. Walter, S. Battiston, and F. Schweitzer. A model of a trust-based recommendation system on a social network. *Autonomous Agents and Multi-Agent Systems*, 16(1):57–74, 2008.

[24] L.-T. Weng, Y. Xu, Y. Li, and R. Nayak. Exploiting item taxonomy for solving cold-start problem in recommendation making. In *ICTAI*, volume 2, pages 113–120, 2008.

[25] H. Yildirim and M. S. Krishnamoorthy. A random walk method for alleviating the sparsity problem in collaborative filtering. In *RecSys*, pages 131–138, 2008.

[26] Y. Zhang, M. Zhang, Y. Liu, S. Ma, and S. Feng. Localized matrix factorization for recommendation based on matrix block diagonal forms. In *WWW*, pages 1511–1520, 2013.

[27] K. Zhou, S.-H. Yang, and H. Zha. Functional matrix factorizations for cold-start recommendation. In *SIGIR*, pages 315–324, 2011.

[28] C.-N. Ziegler, S. M. McNee, J. A. Konstan, and G. Lausen. Improving recommendation lists through topic diversification. In *WWW*, pages 22–32, 2005.