Iterative Similarity Inference via Message Passing in Factor Graphs for Collaborative Filtering

Jun Zou*, Arash Einolghozati*, Erman Ayday†, and Faramarz Fekri*
*School of Electr. & Comp. Engineering, Georgia Institute of Technology, Atlanta, GA 30332, USA
†School of Comp. & Commun. Sciences, Ecole Polytechnique Federale de Lausanne, Lausanne, Switzerland
Email: *{junzou, einolghozati, fekri}@ece.gatech.edu, †erman.ayday@epfl.ch

***Abstract*—In this paper, we develop a Belief Propagation (BP) algorithm for similarity computation to improve the recommendation accuracy of the neighborhood method, which is one of the most popular Collaborative Filtering (CF) recommendation algorithms. We formulate a probabilistic inference problem as to compute the marginal posterior distributions of similarity variables from their joint posterior distribution given the observed ratings. However, direct computation is prohibitive in large-scale recommender systems. Therefore, we introduce an appropriate chosen factor graph to express the factorization of the joint distribution function, and utilize the BP algorithm that operates in the factor graph to exploit the factorization for efficient inference. In addition, since the high degree at the factor node incurs an exponential increase in computational complexity, we also propose a complexity-reduction technique. The overall complexity of the proposed BP algorithm on a factor graph is linear in the number of variables, which ensures scalability. Finally, through experiments on the MovieLens dataset, we show the superior prediction accuracy of the proposed BP-based similarity computation algorithm for recommendation.**

I. Introduction

With the thriving of the Internet and online services, users now can easily have access to huge amount of product information they never experienced in traditional real-entity consuming activities. However, this poses a problem when users have to find the items, e.g., books and movies, interest them. Recommender systems are powerful tools for providing personalized recommendations of items according to user profiles and their past behaviors. Many online service providers have already adopted recommender systems to suggest items that meet user preferences, so as to improve user satisfaction and meanwhile reduce user efforts. There are increasing demands for scalable and accurate recommender systems from large-scale e-commerce websites such as Amazon.com and eBay.com. The databases associated with those websites usually hold millions of transaction and rating records. Meanwhile, accurate recommendations are critical to attract users with products they like, so that they spend more and continue using the website and its recommendation service.

One of the major approaches for designing recommender systems is Collaborative Filtering (CF) [1]. The CF approach takes as input the historic ratings on items given by users, and predicts ratings on unseen items for each active user. It includes model-based methods and memory-based methods. The model-based method generates rating predictions from a model learned from the collected historic ratings, but the learning process is often time-consuming, which is not suitable for systems with frequent updates. The memory-based method, *a.k.a.* the neighborhood method, can be further divided into user-based and item-based methods. The user-based method recommends to an active user new items favorably rated by other users with similar tastes to the active user [2]. The item-based method on the other hand analyzes the similarity between items using the aggregated user ratings, and recommends to an active user new items that are similar to the items he liked in the past [3].

The key component for neighborhood methods is the computation of similarities between users or items. In this paper, we formulate the similarity computation as a probabilistic inference problem of computing the marginal distributions of similarity variables from their joint posterior distribution given the observed ratings. To efficiently solve this problem, we introduce a factorization of the joint distribution in an appropriate chosen factor graph, and apply the Belief Propagation (BP) algorithm [4] that operates in the factor graph to exploit the factorization for efficient inference. In addition, a complexity reduction technique is proposed to contain the exponential increase in computational complexity caused by the high degree at the factor node in our setup of recommender systems. This BP-based approach was motivated by the successful application of BP for iterative decoding algorithms in error-control systems [5]. BP is very efficient in computing marginal functions from global functions of many variables. Moreover, BP can even be applied in situations where exact solutions for marginal functions are computationally intractable, such as the iterative decoding of Low-Density Parity-Check (LDPC) codes and turbo codes. As we shall see, the concerned problem in this work also belongs to such intractable cases.

The recent applications of BP to CF systems were also introduced in [6]–[9]. [6] and [7] proposed message-passing algorithms for performing probabilistic low-rank matrix factorization to represent users and items with low-dimension vectors. [8] and [9] proposed to directly predict ratings in the factor graph, where probabilistic messages on unknown ratings are iteratively exchanged, whereas in this work we focus on inferring the similarities, based on which the unknown ratings are predicted by the neighborhood method.

II. Background on Collaborative Filtering

We assume a set of M users, denoted by $\mathbb{U} = \{1, \ldots, M\}$, and a set of N items, denoted by $\mathbb{I} = \{1, \ldots, N\}$, in the

This material is based upon work supported by the National Science Foundation under Grant No. IIS-1115199, and a gift from the Cisco University Research Program Fund, an advised fund of Silicon Valley Community Foundation.

recommender system. Specifically, user u provides feedback on item i in the form of rating r_{ui}. Let U_i denote the set of users who have rated item i, and I_u denote the set of items rated by user u. We arrange the collection of all ratings in an incomplete $M \times N$ matrix $\mathbb{R}$, with r_{ui} at the intersection of u-th row and i-th column. The entries of unknown ratings are unfilled. The neighborhood method takes as input the historic ratings in $\mathbb{R}$, and generates rating predictions for an active user u on unseen items in $\mathbb{I} \backslash I_u$.

The neighborhood method can be either user-based [2] or item-based [3]. To predict the rating r_{ui} for user u on an unseen item i, the user-based algorithm sorts the users in U_i according to their similarity to user u in descending order, and finds a subset of top K most similar users, denoted by $\mathcal{N}_{ui}$, where $|\mathcal{N}_{ui}| = K$. We refer to $\mathcal{N}_{ui}$ as the neighbourhood of user u for predicting rating on item i, and K as the neighborhood size. Then r_{ui} is predicted by

$$\hat{r}_{ui} = \bar{r}_u + \frac{\sum_{v \in \mathcal{N}_{ui}} s_{uv} \times (r_{vi} - \bar{r}_v)}{\sum_{v \in \mathcal{N}_{ui}} |s_{uv}|}, \tag{1}$$

where s_{uv} is the similarity between users u and v, and $\bar{r}_u$ is the average rating by user u on all items it has rated in the past. Alternatively, the item-based algorithm predicts r_{ui} as

$$\hat{r}_{ui} = \frac{\sum_{j \in \mathcal{I}_{ui}} s_{ij} \times r_{uj}}{\sum_{j \in \mathcal{I}_{ui}} |s_{ij}|}, \tag{2}$$

where s_{ij} denotes the similarity between items i and j, and $\mathcal{I}_{ui}$ represents a subset of K items in I_u that are deemed most similar to item i.

The similarity computation plays a pivotal role in determining the accuracy of the neighborhood methods. There are several well-known simlarity computation methods including Vector Space Similarity (VSS) [10] and Pearson Correlation Coefficient (PCC) [3]. It is observed that PCC performs better than VSS, as PCC takes into account the difference in average ratings. In the case of user similarity, PCC computes s_{uv} between users u and v as

$$s_{uv} = \frac{\sum_{i \in I_{uv}} (r_{ui} - \bar{r}_u)(r_{vi} - \bar{r}_v)}{\sqrt{\sum_{i \in I_{uv}} (r_{ui} - \bar{r}_u)^2} \sqrt{\sum_{i \in I_{uv}} (r_{vi} - \bar{r}_v)^2}}, \tag{3}$$

where $I_{uv} = I_u \cap I_v$ denotes the subset of common items rated by both users u and v. The item similarity can be similarly computed using PCC. The PCC algorithm gains popularity as it is easy to implement and its computational complexity is low, while providing reasonable recommendation performance. In the following, we will develop a BP-based algorithm for similarity computation to improve the accuracy of the neighborhood method, yet with a computational complexity comparable to that of PCC.

III. Proposed Similarity Computation Algorithm

We focus on the similarity computation for the user-based neighborhood method, which generates rating predictions via (1). Its extension to the item-based neighborhood method can be likewise developed and thus is not discussed here for conciseness. To predict ratings for active user u using (1), we need to first compute the user similarities between user u and other users. Let $\mathbb{S}_u$ denote the set of concerned user similarities for user u, $\mathbb{S}_u = \{s_{uv} : 1 \leq v \leq M, v \neq u\}$.

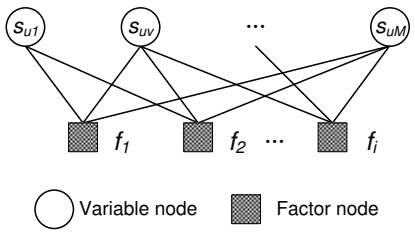


Fig. 1: The factor graph $\mathcal{G}_u$ for similarity computation.

We model s_{uv} as a discrete random variable that takes values from a predefined alphabet set $\mathcal{S}$ with size $L = |\mathcal{S}|$. We denote $P(\mathbb{S}_u|\mathbb{R})$ as the joint posterior probability distribution of all variables in $\mathbb{S}_u$ given the evidence of observed ratings in $\mathbb{R}$. Then to compute the individual user similarity s_{uv}, we need to find its marginal posterior distribution $P(s_{uv}|\mathbb{R})$, which can be derived as

$$P\left(s_{uv}|\mathbb{R}\right) = \sum_{s_{u1} \in \mathcal{S}} \cdots \sum_{s_{u(v-1)} \in \mathcal{S}} \sum_{s_{u(v+1)} \in \mathcal{S}} \cdots \sum_{s_{uM} \in \mathcal{S}} P\left(\mathbb{S}_u|\mathbb{R}\right). \tag{4}$$

For notational convenience, we rewrite (4) for the sum over all variables in $\mathbb{S}_u$ except s_{uv} as

$$P\left(s_{uv}|\mathbb{R}\right) = \sum_{\mathbb{S}_u \backslash s_{uv}} P\left(\mathbb{S}_u|\mathbb{R}\right). \tag{5}$$

Similar notations are used for this type of sum throughout this paper. Unfortunately, the complexity of direct computation using (5) is $\mathcal{O}\left(L^M\right)$, which is exponential in the number of users. We therefore propose to factorize the joint distribution $P(\mathbb{S}_u|\mathbb{R})$ into local functions on a factor graph, which allows the application of the efficient BP algorithm for inferring marginal distributions.

A. Modeling Similarity via Factor Graphs

A factor graph is a bipartite graph that expresses the factorization structure of a function, where variable nodes and factor nodes represent variables and local functions, respectively, and an edge connects a variable node to a factor node if and only if the variable is an argument of the local function represented by the factor node [4]. To construct a factor graph for $P(\mathbb{S}_u|\mathbb{R})$, we first find a proper factorization of $P(\mathbb{S}_u|\mathbb{R})$. We notice that dependencies among user similarities are induced by user ratings on the common items. Hence, for each item $i \in I_u$, we let $\mathbb{S}_{ui} = \{s_{uv} : v \in U_i \backslash u\}$ be the subset of user similarities between user u and the users who have rated item i, and use a local function $f_i\left(\mathbb{S}_{ui}\right)$ to model the dependencies among variables in $\mathbb{S}_{ui}$ based on observed ratings on item i. Then the factorization of $P(\mathbb{S}_u|\mathbb{R})$ can have the following form

$$P(\mathbb{S}_u|\mathbb{R}) = \frac{1}{Z} \prod_{i \in I_u} f_i\left(\mathbb{S}_{ui}\right), \tag{6}$$

where Z is a normalization constant.

Now we construct a factor graph $\mathcal{G}_u$ for $P(\mathbb{S}_u|\mathbb{R})$ according to (6) as illustrated in Fig. 1. Each variable s_{uv} is represented by a variable node v, and each local function $f_i\left(\mathbb{S}_{ui}\right)$ is represented by factor node i. The variable nodes in $\mathcal{V}_i = U_i \backslash u$ are connected to factor node i via edges, and thus the degree at each factor node i is $|\mathcal{V}_i|$.

The local function f_i at factor node i is designed specifically for the eventual goal, which is to predict ratings using (1). Assuming the rating r_{ui} on item i in U_i is unknown, we predict r_{ui} as

$$\hat{r}_{ui}\left(\mathbb{S}_{ui}\right) = \bar{r}_u + \frac{\sum_{v\in\mathcal{V}_i} s_{uv}(r_{vi} - \bar{r}_v)}{\sum_{v\in\mathcal{V}_i} |s_{uv}|}. \tag{7}$$

Note that (7) has a similar form to (1), except that the observed ratings on item i from all users in $\mathcal{V}_i$ are used in (7), since user similarities are not determined yet. We then define the factor node function f_i as

$$f_i\left(\mathbb{S}_{ui}\right) = \frac{1}{Z_i}\exp\left\{-\frac{1}{\sigma^2}\left(\hat{r}_{ui}\left(\mathbb{S}_{ui}\right) - r_{ui}\right)^2\right\}, \tag{8}$$

where Z_i is a normalization constant, and σ^2 is a designing parameter which controls the sensitivity of the function to the discrepancy between $\hat{r}_{ui}\left(\mathbb{S}_{ui}\right)$ and the actual rating r_{ui}. The factor node function f_i here can be seen as a soft check constraint, which checks the weighted average rating in (7) against the true rating r_{ui} for a given configuration of user similarities in $\mathbb{S}_{uv}$. f_i decreases as the discrepancy between $\hat{r}_{ui}$ and r_{ui} increases. It is insightful to compare our factor graph formulation to iterative BP decoding of LDPC codes. Indeed, the factor node in our setup plays a similar role as the check node in LDPC decoding.

B. Iterative Message Passing for Similarity Computation

Our goal is to compute the marginal posterior probability distribution $P\left(s_{uv}|\mathbb{R}\right)$ from $P(\mathbb{S}_u|\mathbb{R})$ for each s_{uv} in $\mathbb{S}_u$. We tap into the BP algorithm for efficient inference on the factor graph. Due to loops in the constructed factor graph $\mathcal{G}_u$, we need to apply the "loopy" BP algorithm, which performs iterative message exchanging between variable nodes and factor nodes. Although its computed results are not exact, "loopy" BP has shown great success in applications such as LDPC decoding, in which the underlying factor graph also has loops, as it achieves near optimal performance with linear complexity.

We define two types of messages following the principle of the sum-product BP algorithm [4]: (i) The λ-message $\lambda_{i,v}(s_{uv})$ sent from a factor node i to a variable node v, and (ii) the μ-message $\mu_{v,i}(s_{uv})$ sent from a variable node v to a factor node i. The λ-messages and μ-messages are iteratively updated and passed along edges in the factor graph. In the n-th iteration, to compute $\lambda_{i,v}^{(n)}(s_{uv})$, factor node i multiplies local function f_i with all μ-messages received in the last iteration except the one from the recipient variable node v, and sums out variables in $\mathbb{S}_{ui}$ excluding s_{uv} as below

$$\lambda_{i,v}^{(n)}(s_{uv}) \propto \sum_{\mathbb{S}_{ui}\backslash s_{uv}} f_i\left(\mathbb{S}_{ui}\right) \prod_{w\in\mathcal{V}_i\backslash v} \mu_{w,i}^{(n-1)}(s_{uw}). \tag{9}$$

The λ-message $\lambda_{i,v}^{(n)}(s_{uv})$ tells user v the likelihoods of $s_{uv} = s, \forall s \in \mathcal{S}$. It says given how similar other users' ratings on item i are to user u's, how similar user v is to user u from item i's point of view.

Then the algorithm continues to update μ-messages using the λ-messages generated in the current iteration. To obtain $\mu_{v,i}^{(n)}(s_{uv})$, variable node v computes the product of all incoming λ-messages except the one from the recipient factor node i as follows

$$\mu_{v,i}^{(n)}(s_{uv}) \propto \prod_{j\in F_v\backslash i} \lambda_{j,v}^{(n)}(s_{uv}) \tag{10}$$

where F_v denotes the set of factor nodes connected to variable node v. Here, $F_v = I_v \cap I_u$. The μ-message $\mu_{v,i}^{(n)}(s_{uv})$ tells item i the probabilities of $s_{uv} = s, \forall s \in \mathcal{S}$. It says given how similar user v is to user u from other items' point of view, how similar user v's rating is to user u's on item i.

In each iteration, the messages are computed for each node in the factor graph. To check convergence of the algorithm, we compute the marginal probability distributions after each iteration as follows

$$P(s_{uv}|\mathbb{R}) = \frac{1}{Z_v}\prod_{i\in F_v} \lambda_{i,v}^{(n)}(s_{uv}), \tag{11}$$

where Z_v is a normalization constant. The algorithm exits iteration when $P(s_{uv}|\mathbb{R})$'s converge. Finally, we can estimate user similarity s_{uv} from (11) according to various criteria. We consider the minimum mean squared error criterion here, and thus the optimal estimation of user similarity s_{uv} is given by its expectation

$$\bar{s}_{uv} = \sum_{s\in\mathcal{S}} sP(s_{uv} = s|\mathbb{R}). \tag{12}$$

C. Complexity Reduction

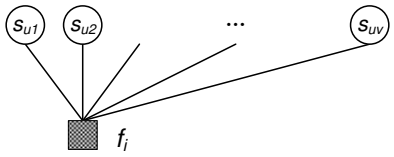
We analyze the complexity of the proposed BP-based algorithm in terms of number of multiplications. The complexity to update a μ-message using (10) is only $\mathcal{O}\left(|I_u|\right)$, but the complexity to update a λ-message using (9) is $\mathcal{O}\left(|\mathcal{V}_i|L^{|\mathcal{V}_i|}\right)$, which is exponential in the degree of the factor node. From the construction process of the factor graph $\mathcal{G}_u$ in Sec. III, we know that $\mathcal{V}_i = U_i\backslash u$. Unfortunately, in recommender systems, an item can be rated by over hundreds of users. Therefore, direct application of the BP algorithm is not feasible. We thus propose to construct a new factor graph $\hat{\mathcal{G}}_u$ from $\mathcal{G}_u$ for complexity reduction as follows.

The key here is to reduce the degree of factor nodes. For a high-degree factor node i, the variable nodes in $\mathcal{V}_i$ are divided into small groups of size D, resulting in $G_i = \lceil\frac{|\mathcal{V}_i|}{D}\rceil$ groups. We denote the subset of variable nodes in group k of factor node i as $\mathcal{V}_i^{(k)}$, $1 \le k \le G_i$. For each variable node $v \in \mathcal{V}_i$, we set an indicator $v_i^{(k)} = 1$ if $v \in \mathcal{V}_i^{(k)}$ and $v_i^{(k)} = 0$ otherwise. Note that $\sum_{k=1}^{G_i} v_i^{(k)} = 1$. Further, we connect a separated factor node $i^{(k)}$ to variable nodes in $\mathcal{V}_i^{(k)}$. We do the same for all other factor nodes in the original factor graph $\mathcal{G}_u$. The complexity-reduction technique on a factor graph is illustrated in Fig. 2. Let $\mathbb{S}_{ui}^{(k)} = \{s_{uv} : v \in \mathcal{V}_i^{(k)}\}$. The factor function $f_i^{(k)}$ of factor node $i^{(k)}$ is derived from (8) as

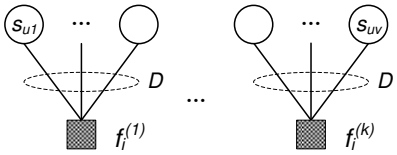
$$f_i^{(k)}\left(\mathbb{S}_{ui}^{(k)}\right) = \frac{1}{Z_i^{(k)}}\exp\left\{-\frac{1}{\sigma^2}\left(\hat{r}_{ui}^{(k)}(\mathbb{S}_{ui}^{(k)}) - r_{ui}\right)^2\right\}, \tag{13}$$

where $Z_i^{(k)}$ is a normalization constant, and

$$\hat{r}_{ui}^{(k)}(\mathbb{S}_{ui}^{(k)}) = \bar{r}_u + \frac{\sum_{v\in\mathcal{V}_i^{(k)}} s_{uv}(r_{vi} - \bar{r}_v)}{\sum_{v\in\mathcal{V}_i^{(k)}} |s_{uv}|}. \tag{14}$$



(a) A high-degree factor node in $\mathcal{G}_u$



(b) Multiple low-degree factor nodes in $\hat{\mathcal{G}}_u$

Fig. 2: Illustration of complexity-reduction in the factor graph.

To better understand the proposed technique for complexity reduction, we provide an intuitive explanation using user-item relations. We divide the users of item i into multiple groups, and create a separated virtual item $i^{(k)}$ for each group k. Users in $\mathcal{V}_i^{(k)}$ give the same ratings on item $i^{(k)}$ as the original item i, i.e., item $i^{(k)}$ can be seen as a copy of item i, but only rated by group k. As in Sec. III-A, now the factorization of $P(\mathbb{S}_u|\mathbb{R})$ can be expressed by

$$P(\mathbb{S}_u|\mathbb{R}) = \frac{1}{Z}\prod_{i\in I_u}\prod_{k=1}^{G_i} f_i^{(k)}\left(\mathbb{S}_{ui}^{(k)}\right). \tag{15}$$

And from (15), we can construct a new factor graph which is exactly the complexity reduction factor graph $\hat{\mathcal{G}}_u$.

While the new factor graph $\hat{\mathcal{G}}_u$ is not mathematically equivalent to the original factor graph $\mathcal{G}_u$, they share the same underlying principle, that is, they properly assign probabilities for each configuration of user similarities by checking the weighted rating against the true item rating via factor node functions (or, soft check node constraints). Moreover, we can apply the similar iterative BP algorithm described in Sec. III-B with minor modifications. The new λ-message $\lambda_{i^{(k)},v}^{(n)}(s_{uv})$ sent from factor node $i^{(k)}$ to variable node v is given by

$$\lambda_{i^{(k)},v}^{(n)}(s_{uv}) \propto \sum_{\mathbb{S}_{ui}^{(k)}\setminus s_{uv}} f_i^{(k)}\left(\mathbb{S}_{ui}^{(k)}\right)\prod_{w\in\mathcal{V}_i^{(k)}\setminus v}\mu_{w,i^{(k)}}^{(n-1)}(s_{uw}). \tag{16}$$

And the new μ-message $\mu_{v,i^{(k)}}^{(n)}(s_{uv})$ sent from variable node v to factor node $i^{(k)}$ is given by

$$\mu_{v,i^{(k)}}^{(n)}(s_{uv}) \propto \prod_{j\in\hat{F}_v\setminus i^{(k)}}\lambda_{j,v}^{(n)}(s_{uv}), \tag{17}$$

where $\hat{F}_v = \left\{j^{(k)} : v_j^{(k)} = 1, j\in I_v\cap I_u, 1\le k\le G_j\right\}$.

Now the complexity of updating one λ-message is effectively reduced to $\mathcal{O}\left(DL^D\right)$ from $\mathcal{O}\left(|\mathcal{V}_i|L^{|\mathcal{V}_i|}\right)$ by using (16), where the group size D is a small integer and is adjustable, while the total number of λ-messages need to be updated remains the same. The computational complexity with regard to μ-messages does not change. Then to solve for $\mathbb{S}_u$ on $\hat{\mathcal{G}}_u$, the overall complexity for one iteration of the iterative BP algorithm becomes $\mathcal{O}\left(\bar{M}\bar{N}DL^D + M\bar{N}^2\right)$, where $\bar{M}$ is the average number of users of one item, and $\bar{N}$ is the average number of items rated by one user. Since the number of items a user can consume is limited by his time and money, we assume $\bar{N}$ is much smaller than N, and we also assume $\bar{M}$ grows in the order of $M^{1-\epsilon}$, where $0<\epsilon<1$. Then we can rewrite the computation complexity on $\hat{\mathcal{G}}_u$ as $\mathcal{O}\left(M^{1-\epsilon}\bar{N}DL^D + M\bar{N}^2\right)$, and when M is large, it is dominated by the second term, so the complexity becomes $\mathcal{O}\left(M\bar{N}^2\right)$.

IV. Evaluation

We evaluate the performance of the proposed BP-based similarity computation algorithm using the 100K MovieLens dataset[1]. The dataset contains $100,000$ ratings, all integers from 1 to 5, on 1682 items (movies) by 943 users, and each user has rated at least 20 items. We randomly divide the users into two disjoint sets, 80% for training and 20% for testing. Specifically, for each user in the test set, we only keep a subset of its ratings as known (15 ratings in our setup), and use the rest of its ratings as test ratings. The known ratings are used as memory for the user-based neighborhood method to compute user similarity and to predict ratings using (1).

We compare the prediction performances of the user-based neighborhood method when the user similarities are computed using our proposed algorithm and the PCC algorithm [3], in terms of both Mean Absolute Error (MAE) and Root Mean Squared Error (RMSE) metrics. Specifically, we compute the MAE and RMSE as below

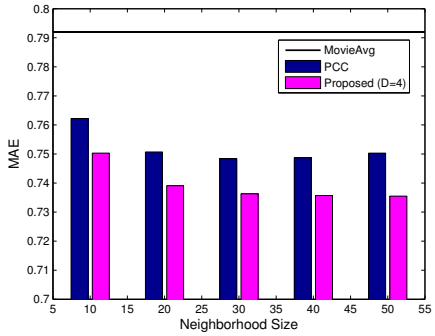
$$\text{MAE} = \frac{1}{\mathbb{T}}\sum_{r_{ui}\in\mathbb{T}}|r_{ui}-\hat{r}_{ui}|,$$

$$\text{RMSE} = \sqrt{\frac{1}{\mathbb{T}}\sum_{r_{ui}\in\mathbb{T}}(r_{ui}-\hat{r}_{ui})^2},$$

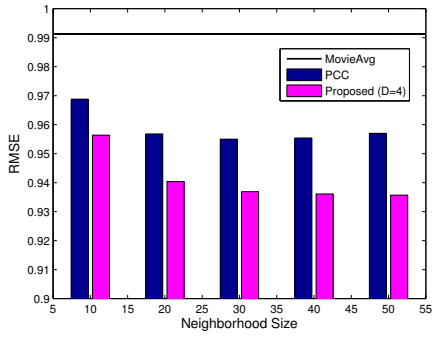
where $\mathbb{T}$ is the set of all test ratings for users in the test dataset, r_{ui} is the actual value of the rating provided by user u on the item i in the test dataset, and $\hat{r}_{ui}$ is the predicted rating value. Smaller RMSE and MAE errors mean better prediction accuracy. Note that the RMSE metric is more sensitive to large errors than MAE.

The results for the proposed algorithm and the PCC algorithm are presented in Fig. 3, where for the proposed algorithm, the similarity values of s_{uv} are taken from $\mathcal{S}=\{1,2\}$, σ in (8) is set as $\sigma=0.5$, and the group size D (i.e., the maximum allowed degree of factor node) is set as $D=4$. We also show the results of the MovieAvg algorithm, which simply predicts ratings as the average of past ratings of each item. It can be observed from Fig. 3 that our proposed algorithm outperforms PCC in both MAE and RMSE under different neighborhood sizes. When the neighborhood size increases from 10 to 30 the performances of both algorithms improve, but the gain from increasing neighborhood size quickly diminishes and finally the performance even starts to degrade when K reaches 50. This is because as neighborhood size increases, ratings from users with smaller similarity to the active user are also included to predict the rating, which corrupts the prediction from other users with higher similarity. The computational complexity of the proposed algorithm to solve for the similarities between the active user u and other users is $\mathcal{O}\left(M\bar{N}^2\right)$ as discussed

[1]Available at: http://www.grouplens.org/node/73.



(a) MAE versus the effective neighborhood size.



(b) RMSE versus the effective neighborhood size.

Fig. 3: Rating prediction performance comparison.

TABLE I: Proposed algorithm with varying group size D when $\mathcal{S} = \{1, 2\}$.

Group size	MAE		RMSE	
	$K = 10$	$K = 30$	$K = 10$	$K = 30$
$D = 3$	0.7533	0.7373	0.9593	0.9380
$D = 4$	0.7487	0.7358	0.9548	0.9362
$D = 5$	0.7522	0.7370	0.9587	0.9379

in Sec. III-C, while the complexity for the PCC algorithm is $\mathcal{O}\left(M\bar{N}\right)$, and thus the complexity of both algorithms is linear in the number of users.

We also investigate the impacts of different parameters on the proposed algorithm. In Table I, we show the results of the proposed algorithm for varying group size D, where we set $\mathcal{S} = \{1, 2\}$, and $\sigma = 0.5$. It can be seen that $D = 4$ achieves the best performance. It is also interesting to notice that the performances under different D's are close, meaning the algorithm is not sensitive to the choice of D. Since a large D dose not necessarily improve the performance and it exponentially increases computational complexity for generating λ-messages as discussed in Sec. III-C, it is wise to start from a small integer when searching for a good D for the algorithm.

In Table II, we show the results for different $\mathcal{S}$, where $D = 3$ and $\sigma = 0.5$. We denote $\mathcal{S}_L = \{s : 1 \leq s \leq L\}$, where s only takes integer values and thus $|\mathcal{S}_L| = L$. It can be observed that the performances of the proposed algorithm for different $\mathcal{S}$'s are quite close, and $\mathcal{S}_5$ actually has slightly better

TABLE II: Proposed algorithm with varying $\mathcal{S}$ when $D = 3$.

$\mathcal{S}$	MAE		RMSE	
	$K = 10$	$K = 30$	$K = 10$	$K = 30$
$\mathcal{S}_2$	0.7533	0.7373	0.9593	0.9380
$\mathcal{S}_5$	0.7524	0.7375	0.9584	0.9381
$\mathcal{S}_{10}$	0.7532	0.7379	0.9590	0.9386

performance than $\mathcal{S}_{10}$. This is because a large alphabet set $\mathcal{S}$ can causes overfitting, that is the computed user similarity is biased towards the training ratings and dose not generalize well on the test ratings.

V. Conclusion

In this paper, we proposed a BP-based similarity computation algorithm for the neighborhood method in recommender systems. In order to take advantage of BP to efficiently compute the marginal distributions of similarity variables from their joint posterior distribution, we introduced a proper factorization of the joint distribution function, which was expressed by an appropriate chosen factor graph. Since the resulting factor graph has loops, the "loopy" BP algorithm using iterative message passing was applied. We also proposed a complexity-reduction technique to contain the exponential increase in computational complexity due to the high degree at the factor node. The experimental results on 100K MovieLens dataset showed that the proposed similarity computation algorithm for the user-based neighborhood method achieves improved accuracy over the popular PCC algorithm in terms of both MAE and RMSE. Meanwhile, the computational complexity of the BP-based algorithm is comparable to that of PCC, growing linear in the number of users.

References

[1] G. Adomavicius and A. Tuzhilin, "Toward the next generation of recommender systems: A survey of the state-of-the-art and possible extensions," *IEEE Trans. Knowl. and Data Eng.*, vol. 17, pp. 734–749, Jun. 2005.

[2] P. Resnick, N. Iakovou, M. Sushak, P. Bergstrom, and J. Riedl, "Grouplens: An open architecture for collaborative filtering of netnews," in *Proc. 1994 Computer Supported Cooperative Work Conf.*, 1994.

[3] B. Sarwar, G. Karypis, J. Konstan, and J. Riedl, "Item-based collaborative filtering recommendation algorithms," in *Proc. WWW*, 2001.

[4] F. R. Kschischang, B. J. Frey, and H.-A. Loeliger, "Factor graphs and the sum-product algorithm," *IEEE Transactions on Information Theory*, vol. 47, no. 2, pp. 498–519, Feb. 2001.

[5] F. R. Kschischang and B. J. Frey, "Iterative decoding of compound codes by probability propagation in graphical models," *IEEE J. Select. Areas Commun.*, vol. 16, no. 2, pp. 219–230, Feb. 1998.

[6] D. H. Stern, R. Herbrich, and T. Graepel, "Matchbox: Large scale online bayesian recommendations," in *Proc. 18th International Conference on World Wide Web (WWW)*, 2009, pp. 111–120.

[7] B.-H. Kim, A. Yedla, and H. D. Pfister, "IMP: A message-passing algorithm for matrix completion," in *Proc. 6th Int. Symp. on Turbo Codes and Iterative Inform. Proc. (ISTC)*, 2010, pp. 462–466.

[8] E. Ayday and F. Fekri, "A belief propagation based recommender system for online services," in *Proc. 4th ACM Conference on Recommender Systems (RecSys)*, 2010, pp. 217–220.

[9] E. Ayday, A. Einolghozati, and F. Fekri, "BPRS: Belief propagation based iterative recommender system," in *Proc. IEEE International Symposium on Information Theory (ISIT)*, 2012, pp. 1992–1996.

[10] J. S. Breese, D. Heckerman, and C. Kadie, "Empirical analysis of predictive algorithms for collaborative filtering," in *Proc. UAI*, 1998.