CCCF: Improving Collaborative Filtering via Scalable User-Item Co-Clustering

Yao Wu[†*], Xudong Liu[‡*], Min Xie[♯], Martin Ester[†], Qing Yang[‡]
[†]School of Computing Science, Simon Fraser University, Burnaby, BC, Canada
[‡]Institute of Automation, Chinese Academy of Sciences, Beijing, China
[♯]Walmart Labs, San Bruno, CA, USA
[†]{wuyaow, ester}@sfu.ca [‡]{xudong.liu, qyang}@nlpr.ia.ac.cn [♯]mxie@walmartlabs.com

ABSTRACT

Collaborative Filtering (CF) is the most popular method for recommender systems. The principal idea of CF is that users might be interested in items that are favorited by similar users, and most of the existing CF methods measure users' preferences by their behaviors over all the items. However, users might have different interests over different topics, thus might share similar preferences with different groups of users over different sets of items. In this paper, we propose a novel and scalable method CCCF which improves the performance of CF methods via user-item co-clustering. CCCF first clusters users and items into several subgroups, where each subgroup includes a set of like-minded users and a set of items in which these users share their interests. Then, traditional CF methods can be easily applied to each subgroup, and the recommendation results from all the subgroups can be easily aggregated. Compared with previous works, CCCF has several advantages including scalability, flexibility, interpretability and extensibility. Experimental results on four real world data sets demonstrate that the proposed method significantly improves the performance of several state-of-the-art recommendation algorithms.

Categories and Subject Descriptors

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

Keywords

Collaborative Filtering; Recommender Systems; Co-Clustering

1. INTRODUCTION

The goal of recommender systems is to deliver proper items to users in order to improve user engagement and merchant revenue. Collaborative Filtering (CF) is the most popular method for recommender systems [16]. The underlying assumption of CF is that if a user u has similar preferences with another user v on many items, it is more likely for them to share preferences over the remaining items than a randomly chosen pair of users.

Typical CF methods measure users' preferences by their historical behaviors over the entire item space. For example, the user-based nearest neighbors method measures the similarity between pairs of users by their preferences on all the items [16]; Matrix Factorization decomposes the whole user-item matrix into two low-rank sub-matrices where the collective intelligences are represented as latent factors in the model [8]. However, this assumption does not always hold, especially when the underlying system includes a large set of items of different types. For example, an Amazon user share similar tastes on books with a certain group of users, while having similar preferences with another group of users on movies.

It is more natural to model the users and the items using subgroups, where each subgroup includes a set of like-minded users and the subset of items that they are interested in, and each user/item can be assigned to multiple subgroups so that users can share their interests with different subsets of users on different subsets of items. Partitioning users and items into subgroups for CF has been studied by several previous works, where user clustering [26], item clustering [13] and user-item co-clustering [6] methods have been proposed to boost the performance of collaborative filtering. However, in these methods each user/item is only allowed to be assigned to a single subgroup, so that they are not able to model the case where users have multiple interests. To address this problem, several other papers [18, 20, 21] extend the Mixed Membership Stochastic Blockmodels (MMSB) [2] to allow mixed memberships. However, these methods optimize the accuracy of rating/link prediction instead of the practically important problem of top-N recommendation.

In [25], the authors propose a unified framework for improving collaborative filtering via overlapping co-clustering, which can be employed for top-N recommendation. It works as follows: 1) Users and items are first grouped into multiple overlapping subgroups with different weights. 2) After obtaining the subgroups, any traditional CF method can be applied to each subgroup separately to make the predictions for the users on their unrated items in the same subgroup. 3) The final recommendation list for a user is aggregated from the results in the subgroups that she/he is involved in. The authors of [31] adopt a similar framework, and propose to partition the user-item matrix by permuting the rows and columns of the user-item matrix to form (approximate) Bordered Block Diagonal matrices.

However, the co-clustering objective function of [25] is based on a weighted graph cut problem, which partitions the underlying graph into non-overlapping subgroups and minimizes the total costs associated with the cut edges. Therefore, [25] does not *intrinsically* learn overlapping subgroups. Also, solving the weighted graph cut problem has a high computational complexity. In [31], assignments of users and items to subgroups are deterministic and have equal

*The first two authors contributed equally.

WSDM'16, February 22–25, 2016, San Francisco, CA, USA.

ISBN 978-1-4503-3716-8/16/02...$15.00

DOI: http://dx.doi.org/10.1145/2835776.2835836

weight, *i.e.*, users are forced to have equal amount of interest in each subgroup that they are assigned to, thus affiliation strengths are not modeled.

We argue that in order to model users' shared interests in different subsets of items, we have to resort to overlapping co-clustering methods which have the following three properties:

- The proposed method should be able to model the strength of affiliations between users, items and subgroups by assigning weights to each user and item for each subgroup. As we shall see later in this work, these weights are critical when aggregating the results from subgroups to make the final recommendation.
- The proposed method should be able to model a user's interest in an item through her/his affiliations with different subgroups.
- The proposed method should have the property that the more subgroups that a user u and an item i share, and the higher the affiliation strengths of u and i with the subgroups they share, the more likely u should be interested in i.

In this paper, we propose a novel co-clustering method, called CCCF (CO-CLUSTERING FOR COLLABORATIVE FILTERING). In CCCF, each user/item can belong to multiple subgroups, and we use an affiliation strength score to denote the strength of the relationship between the user/item and the corresponding subgroup. CCCF models the probability of a user liking an item as a function based on subgroup affiliation strengths between users, items and their shared subgroups. If a user and an item share multiple subgroups, we assume that each such shared subgroup has an independent chance to trigger the link[1] between them [27]. As will be discussed in Section 3, the resulting CCCF model satisfies the three desired properties stated above.

To estimate the parameters, such as the affiliation strengths, of the proposed model, we formulate the problem as a probabilistic model and leverage a scalable Markov Chain Monte Carlo method, of which the complexity is linear *w.r.t.*the size of the dataset, thus making the underlying algorithm scalable. After we get the subgroups and the affiliation strengths of the users and items, any traditional CF method can be applied to each subgroup to make the predictions for the users on the items within the same subgroup. Finally, we leverage the affiliation strengths to aggregate the results from all the subgroups to make the final recommendation.

CCCF has the following four benefits (see details in Section 3.5): 1) The structure of the obtained subgroups can be very flexible, and is not restricted to be of certain shapes as in previous works. 2) It is easy to interpret the relationships between user/item and the obtained subgroups through the affiliation strengths. 3) CCCF can be easily modified to leverage content features of users/items which is challenging for previous works. 4) The computational complexity of the proposed model is linear *w.r.t.*the number of data points, thus making CCCF scalable to large datasets.

We evaluate the proposed method on four real world data sets. Our experimental results demonstrate that the proposed method outperforms state-of-the-art methods on the Top-N recommendation task. A qualitative analysis also shows that the obtained subgroups are semantically meaningful.

In the rest of this paper, we first introduce the problem definition and some useful notations in Section 2. In Section 3, we describe the CCCF model and present a scalable inference method. Experimental results on four real world data sets are shown in Section 4. We finally discuss some related works in Section 5 and conclude this paper in Section 6.

[1] Here a link between a user and an item means the user likes the item. If there is no link between them, we treat it as missing.

Table 1: Mathematical Notations

Symbol	Value	Description
y_{ui}	$\{0,1\}$	user u's feedback on item i
ϕ_{uk}	$[0,1]$	latent membership of user u in subgroup k
ϕ_{ik}	$[0,1]$	latent membership of item i in subgroup k
θ_k	$[0,1]$	in-group preference strength of subgroup k
$z_{u\to i,k}$	$\{0,1\}$	membership indicator of user u in subgroup k when interacting with item i
$z_{i\to u,k}$	$\{0,1\}$	membership indicator of item i in subgroup k when interacting with user u

2. PROBLEM DEFINITION

Given a set of users $\mathcal{U} = \{u|u = 1, ..., U\}$, a set of items $\mathcal{I} = \{i|i = 1, ..., I\}$, as well as the log of users' historical preferences $\mathcal{O} = (u, i, y_{ui})$, the goal of recommender systems is to recommend to each user u a list of items that will maximize her/his satisfaction. Here, y_{ui} can be numeric ratings in the scale of, say, $[1, 5]$ or binary values $\{0, 1\}$.

In this paper, we mainly focus on the case of *implicit* feedback, namely, we only have a partial information of the items that users have viewed/liked, and users' feedback on other items is missing. This is the most common scenario in practice. However, the proposed method can also be applied to other cases with slight modifications. For implicit feedback, all y_{ui} in $\mathcal{O}$ are 1; and for those $(u', i', y_{u'i'})$ triples not belonging to $\mathcal{O}$, the corresponding $y_{u'i'}$'s are missing. We use $\mathcal{O}'$ to denote the set of missing triples. The goal of top-N recommendation is to recommend each user a list of N items that she/he is most likely to like, *i.e.*, to pick the list of missing $y_{u'i'}$-s for every user which are most likely to be 1.

Some important notations used in this paper are listed in Table 1. We use u to index a user, and i and j to index items, and use normal font symbol x to denote scalars and bold math symbols $\boldsymbol{x}$ to denote vectors. The k-th element of vector $\boldsymbol{x}_i$ is denoted by x_{ik}.

3. METHODOLOGY

3.1 Overview

The overall procedure of our method, also used by [25, 31], is as follows:

1. Cluster users and items into subgroups, where users and items can be assigned to multiple subgroups and each subgroup includes a group of like-minded users and a set of items that these users are particularly interested in. We present a scalable co-clustering method called CCCF in Section 3.2.

2. In each subgroup, we apply traditional collaborative filtering methods (*e.g.*, ITEMCF, MATRIX FACTORIZATION) to learn users' preferences over the items within this subgroup. Since the learning of CF models in each subgroup is independent from each other, this step can be easily done in parallel (See Section 3.3).

3. For each user, we aggregate the recommendation results from all the subgroups that she/he is involved in. We discuss several aggregation strategies in Section 3.4 and study their performance in Section 4.3.4.

The main differences from previous works [25, 31] lie in step 1 and 3, which will be explained in detail in Section 3.2 and 3.4, respectively. In Section 3.5, we elaborate the benefits of the proposed method compared to those related works.

3.2 Scalable Co-Clustering for CF

In this subsection, we present Co-Clustering for Collaborative Filtering (CCCF), a co-clustering model which assigns users and items into overlapping subgroups. As discussed in Section 1, our goal is to find several latent subgroups, where each subgroup includes a set of like-minded users and the subset of items they are interested in. We assume that each user/item has a latent membership vector over subgroups. The memberships reveal users' latent interests on different types of items and items' latent properties that would be consumed by users, respectively. The links between users and items are generated based on the interactions between users, items and subgroups, *i.e.*, a user has larger chance to like an item if they both have strong affiliations to the same subgroups.

The two principles of designing CCCF are:

- A user has multiple types of interests so that she/he should be assigned to multiple subgroups with different strengths. Analogously, items are also allowed to be affiliated to multiple subgroups.
- The reason that a user likes an item is explained by the subgroup affiliations, *i.e.*, a user is more likely to like an item *if and only if* they belong to same subgroups. And we assume that the probability of a user liking an item depends on two ingredients: 1) The more overlapping subgroups a user shares with an item, the higher the probability that the user likes the item is. 2) A link between a user and an item can be explained by a dominant reason, *i.e.*, if a user and an item both have large affiliation strengths with a single subgroup, it is sufficient to form the link between them.

We assume there are K subgroups and each user/item has a probability of belonging to each subgroup. We denote by $\phi_{uk} \in [0, 1]$ the affiliation strength of user u to subgroup k, and $\phi_{ik} \in [0, 1]$ the affiliation strength of item i to subgroup k. We sample ϕ_{uk} and ϕ_{ik} from Beta distributions parameterized by predefined constants α_{k1} and α_{k2}.

$$\begin{aligned} \phi_{uk} &\sim \text{Beta}\left(\alpha_{k1}, \alpha_{k2}\right) \\ \phi_{ik} &\sim \text{Beta}\left(\alpha_{k1}, \alpha_{k2}\right) \end{aligned} \tag{1}$$

Intuitively, ϕ_{uk} represents the probability that user u likes the items in subgroup k. Analogously, ϕ_{ik} is the probability that item i is liked by the users in subgroup k. Items having large memberships in subgroup k share some latent properties recognized by a group of like-minded users.

The reasons why we use Beta distribution for ϕ_{uk} and ϕ_{ik} are: 1) In our work, we define $\boldsymbol{\phi}$ as membership variables with values in $[0, 1]$, and we do not enforce $\boldsymbol{\phi}_u$ to form a probabilistic distribution over the set of all the subgroups as the Mixed Membership models (*e.g.*, [2, 18]) do. Instead, users and items can have large affiliation strengths with multiple subgroups, which has been shown to be more suitable for modeling overlapping subgroups [27, 28]. 2) Beta distribution is the conjugate prior of Bernoulli distribution, which we use for the indicator variable z later. This enables us to perform sampling steps more efficiently (see Section 3.2.1).

We assume that each subgroup k has an independent chance θ_k to trigger the link between user u and item i if both of them belong to this subgroup, and the overall probability relies on the rate that at least one of the K event succeeds.

Here θ_k acts as a parameter to model the strength of connectivity within each subgroup k, meaning that the larger θ_k is, the more likely it is to form links between users and items within this subgroup. This parameter can model the fact that in practice, some subgroups are more active than others. Also, if a user belongs to this subgroup, there is a large chance of her/him liking an item shared by users within this subgroup, whereas for the less active subgroups, the chance of her/him liking an item shared by users within the subgroup is lower.

We also place a Beta distribution as the prior of θ_k:

$$\theta_k \sim \text{Beta}\left(\beta_1, \beta_2\right). \tag{2}$$

Since our goal in this paper is to find several focused subgroups, we'd like all the learned θ_k-s being close to 1. This prior knowledge can be incorporated into the model by, *e.g.*, setting $\beta_1 = 10$ and $\beta_2 = 1$.

For a pair of user u and item i, we use a parameter $z_{u \to i,k}$[2] to denote whether user u belongs to subgroup k or not when forming the link with item i. $z_{u \to i,k}$ is an indicator parameter drawn from a Bernoulli distribution parameterized by user u's group affiliation weight ϕ_{uk}. The parameter $z_{i \to u,k}$ which represents item i's affiliation with subgroup k for the link with user u is defined analogously.

$$\begin{aligned} z_{u \to i,k} &\sim \text{Bernoulli}\left(\phi_{uk}\right) \\ z_{i \to u,k} &\sim \text{Bernoulli}\left(\phi_{ik}\right) \end{aligned} \tag{3}$$

Then, the probability p_{uik} that subgroup k triggers a link between user u and item i is defined as

$$p_{uik} = \begin{cases} \theta_k, & \text{if } z_{u \to i,k} = 1 \text{ and } z_{i \to u,k} = 1 \\ 0, & \text{otherwise.} \end{cases} \tag{4}$$

Note that, if either the user or the item do not belong to subgroup k, the probability of forming a link via this subgroup is 0.

The overall probability p_{ui} that user u likes item i is modeled as the overall success rate of the K independent events, each of which represents the probability of a subgroup k to trigger the link between u and i.

$$p_{ui} = 1 - \textstyle\prod_k (1 - p_{uik}) = 1 - \prod_k (1 - \theta_k)^{z_{u \to i,k} z_{i \to u,k}} \tag{5}$$

The link between user u and item i is sampled as follows:

$$y_{ui} \sim \text{Bernoulli}\left(p_{ui}\right), \tag{6}$$

where $y_{ui} = 1$ means user u has a link with item i and $y_{ui} = 0$ means the link between them is missing.

We show the generative process in Algorithm 1.

Note that in addition to using the subgroup affiliations to explain the observed links with $y_{ui} = 1$, the CCCF model also needs to consider the missing links (where $y_{ui} = 0$) properly to avoid non-meaningful subgroup affiliations. However, the number of missing links is usually almost as large as $U \cdot I$ due to the data sparsity, which makes it impractical for large data sets. To address this issue, we randomly sample a subset of the missing links during the inference, and alternate different subsets in different iterations. Specifically, we randomly select N_u missing links for each user u, where N_u is proportional to the number of u's observed links.

3.2.1 Parameter Estimation

Maximum likelihood estimation is intractable for this model due to the existence of hidden variables. We use a Markov Chain Monte

[2] Here the notation $u \to i$ is used to denote user u's assignment when forming the link with the item i. Similar notations are also used in, *e.g.*, [2].

Algorithm 1 Generative Process of CCCF.

Input: $\mathcal{O}$, $\mathcal{O}'$, $\boldsymbol{\alpha}$ and $\boldsymbol{\beta}$
for all $k \in \{1, ..., K\}$ **do**
  for all $u \in \mathcal{U}$ **do**
    Draw $\phi_{uk} \sim \text{Beta}(\alpha_{k1}, \alpha_{k2})$
  end for
  for all $i \in \mathcal{I}$ **do**
    Draw $\phi_{ik} \sim \text{Beta}(\alpha_{k1}, \alpha_{k2})$
  end for
  Draw $\theta_k \sim \text{Beta}(\beta_1, \beta_2)$
end for
for all $(u, i, y_{ui}) \in \mathcal{O} \cup \mathcal{O}'$ **do**
  for all $k \in \{1, ..., K\}$ **do**
    Draw $z_{u \to i,k} \sim \text{Bernoulli}(\phi_{uk})$
    Draw $z_{i \to u,k} \sim \text{Bernoulli}(\phi_{ik})$
    if $z_{u \to i,k} = 1$ and $z_{i \to u,k} = 1$ **then**
      Set $p_{uik} = \theta_k$
    else
      Set $p_{uik} = 0$
    end if
  end for
  Set $p_{ui} = 1 - \prod_k (1 - p_{uik})$
  Draw $y_{ui} \sim \text{Bernoulli}(p_{ui})$
end for

Carlo (MCMC) method to estimate the parameters where we iteratively sample unknown variables from their conditional distributions given all the observations and the other variables fixed.

Denoting the set of items that user u has links with in the training data (including the observed links and randomly sampled negative links) by $\mathcal{T}_u$, we have the following sampling equation for $z_{u \to i,k}$ (latent variables $\boldsymbol{\phi}$ have been integrated out):

$$\begin{aligned} & p(z_{u \to i,k} = 1 | z_{\neg(u \to i,k)}, \boldsymbol{y}, \boldsymbol{\theta}, \alpha_{k1}, \alpha_{k2}) \\ \propto & \left(n_{uk}^{\neg i} + \alpha_{k1}\right) p(y_{ui} | \boldsymbol{z}_{\neg(u \to i,k)}, z_{u \to i,k} = 1, \boldsymbol{\theta}), \\ & p(z_{u \to i,k} = 0 | z_{\neg(u \to i,k)}, \boldsymbol{y}, \boldsymbol{\theta}, \alpha_{k1}, \alpha_{k2}) \\ \propto & \left(n_u - 1 - n_{uk}^{\neg i} + \alpha_{k2}\right) p(y_{ui} | \boldsymbol{z}_{\neg(u \to i,k)}, z_{u \to i,k} = 0, \boldsymbol{\theta}), \end{aligned} \tag{7}$$

where $n_{uk}^{\neg i} = \sum_{j \in \mathcal{T}_u \setminus \{i\}} \mathbf{1}(z_{u \to j,k} = 1)$ and $n_u = |\mathcal{T}_u|$

Since we sample different subsets of missing links during different iterations, a missing link might be new at iteration t. In this case, we do not have previous statistics of $z_{u \to i,k}$ and $z_{i \to u,k}$ to sample their new values according to Equation 7. Inspired by the stochastic collapsed variational Bayesian inference method for Latent Dirichlet Allocation [5], we use some additional *burn-in* steps to update these statistics before doing the sampling.

Since it is intractable to directly sample the posterior of each θ_k, we update these parameters using Metropolis-Hasting steps, where in each iteration a candidate for the next sample is generated from a *"proposal"* distribution. The proposed sample is accepted with some probability, otherwise the previous sample is duplicated. To be specific, in each iteration we propose each θ_k's new value from $\theta_k \sim \text{Beta}(\beta_1, \beta_2)$ and accept this new value with probability

$$\min\left(1, \frac{P(\boldsymbol{y}|\boldsymbol{z}, \boldsymbol{\theta}^{(t)}) P(\boldsymbol{\theta}^{(t)}|\boldsymbol{\beta})}{P(\boldsymbol{y}|\boldsymbol{z}, \boldsymbol{\theta}^{(t-1)}) P(\boldsymbol{\theta}^{(t-1)}|\boldsymbol{\beta})}\right). \tag{8}$$

If the new value is rejected, the previous value is retained.

The overall inference procedure is shown in Algorithm 2. After we get the optimized parameters $\boldsymbol{\theta}$ and $\boldsymbol{z}$, the group memberships

Algorithm 2 Inference Procedure for CCCF

Input: Randomly initialize $\boldsymbol{z}$. Set each $\theta_k = 1$.
Output: $\boldsymbol{z}$, $\boldsymbol{\theta}$, $\boldsymbol{\phi}$
while not converged or maximum number of iterations **do**
  Randomly sample a subset of missing values $\mathcal{O}''$ and set the training set as $\mathcal{T} = \mathcal{O} \cup \mathcal{O}''$.
  for all $(u, i, y_{ui}) \in \mathcal{T}$ **do**
    for all $k \in \{1, ..., K\}$ **do**
      Sample $z_{u \to i,k}$ using Equation 7.
      Sample $z_{i \to u,k}$ similarly.
    end for
  end for
  while not converged **do**
    for all $k \in \{1, ..., K\}$ **do**
      Update θ_k using Equation 8
    end for
  end while
end while
Compute $\boldsymbol{\phi}_u$ and $\boldsymbol{\phi}_i$ for all u and i using Equation 9 and 10.

$\boldsymbol{\phi}$ can be computed by

$$\phi_{uk} = \frac{\sum_{i \in \mathcal{T}_u} z_{u \to i,k} + \alpha_{k1}}{n_u + \alpha_{k1} + \alpha_{k2}}, \tag{9}$$

$$\phi_{ik} = \frac{\sum_{i \in \mathcal{T}_i} z_{i \to u,k} + \alpha_{k1}}{n_i + \alpha_{k1} + \alpha_{k2}}. \tag{10}$$

3.3 Collaborative Filtering in Subgroups

After getting the group affiliation strengths, we assign to each subgroup k a threshold ϵ_k. If user u's (or item i's) affiliation strength ϕ_{uk} (ϕ_{ik}) is larger than ϵ_k, we assume that this user/item belongs to this subgroup. Then, *any* Collaborative Filtering algorithm can be applied to a subgroup to compute the prediction $\hat{y}_{ui}^k$, which represents user u's preference on item i based on subgroup k.

The choice of the underlying CF methods depends on a lot of factors, such as data sparsity, complexity, model interpretability and latency of recommendation. One benefit of the framework is that the method does not rely on a specific CF algorithm. Any CF method of interest can be used and the framework will likely improve their performances. In our experiments, we choose four most representative CF models as the base methods applied to the subgroups, and show that the framework improves their recommendation performances by a significant margin on four real world data sets.

To make this paper self-contained, here we discuss the four CF methods which we use in our experiments. The reason why we choose these methods is that they cover a diverse set of methods ranging from the simplest solution to state-of-the-art models.

Popularity (POP). Items are scored by the percentage of users who like them.

$$y_{ui} = \sum_{u'} \frac{y_{u'i}}{|\mathcal{U}|} \tag{11}$$

Though simple, POP is widely used in today's industry as it is extremely easy to implement. The disadvantage is that it is not a personalized method and all the users receive the same recommendations. This problem can be solved by the proposed framework. CCCF first partitions users and items into subgroups. The item popularity within a specific subgroup indicates the popularity among a small group of people. Using the recommendation method

that we will discuss in Section 3.4, CCCF provides users personalized recommendations by aggregated ranking lists of items that are popular in the subgroups that users are particularly interested in.

ITEMCF [17]. We use Jaccard correlation as the similarity measure and set the number of nearest neighbors to 50.

$$\hat{y}_{ui} = \sum_{j \in \mathcal{O}_u} s(i,j), \tag{12}$$

where $s(i,j)$ is the Jaccard similarity between the set of users who have liked item i and j, respectively.

MF (with negative sampling) [8, 12]. Matrix Factorization, or Latent Factor Model is the most popular Collaborative Filtering method for recommender systems. The preference of user u on item i is the dot product of the latent feature vectors of the user and the item, with some bias terms.

$$\hat{y}_{ui} = \alpha + b_u + b_i + \boldsymbol{u}_u^\top \boldsymbol{v}_i \tag{13}$$

The model parameters, latent factors and bias term, are learned by optimizing the following objective function.

$$\sum_{(u,i) \in \mathcal{T}} \ell(y_{ui}, \hat{y}_{ui}) + \lambda \mathcal{R}(\boldsymbol{b}, \boldsymbol{u}, \boldsymbol{v}), \tag{14}$$

where $\mathcal{R}$ is the regularization term.

For implicit feedback, we sample a subset of missing values as negative samples, and re-sample the subset for each learning iteration.

WARP [22, 23]. Matrix factorization with the WARP loss [22] is the state-of-the-art method for top-n recommendation on implicit feedback data, which has been used by many recent works [7, 24]. The main difference between WARP and MF is the objective function. WARP models the relative orders between items, which is the goal of top-N recommendation.

Another way of making recommendation in the subgroup is just setting $\hat{y}_{ui}^k = \theta_k$. However, this makes a strong assumption that users in a subgroups have the same preferences for all the items within this subgroup, which will result in suboptimal recommendation results. We will discuss this case in Section 3.4 and compare it with other recommendation methods in Section 4.3.4.

3.4 Recommendation

The last but not the least question is to how to compute the final recommendation list for each user. In [25], the authors just keep several subgroups with the largest weights for each user and sum up the recommendation scores from the selected subgroups. In [31], the proposed algorithm gets the final prediction of user u on an item i by averaging the predicted scores of u on i in all subgroups they share.

However, users and items belong to the subgroups with different weights, and the final recommendation should not ignore the weights. In this paper, we propose to use an affiliation strength weighted aggregation function to compute the final recommendations:

$$\hat{y}_{ui} = \sum_k \phi_{uk} \cdot \phi_{ik} \cdot \theta_k \cdot \hat{y}_{ui}^k \tag{15}$$

This aggregation function matches our motivation of designing the co-clustering method: 1) The prediction $\hat{y}_{ui}^k$ has a large value if the user and the item both have large affiliation strengths with this subgroup; 2) The more subgroups the user and the item share, the larger their predicted score is. The comparison with other aggregation methods is discussed in Section 4.3.4.

We note that for some CF methods (*e.g.*, WARP [22], BPR [15]), the prediction score $\hat{y}_{ui}^k$ is only a relative measurement (*i.e.*, $\hat{y}_{ui} > \hat{y}_{uj}$ means user u prefers item i than j) and unbounded. Before aggregating the scores, we first scale all the predictions of a user to $[0, 1]$.

In this following we also present three other strategies to aggregate the results.

CCCF-PR. As discussed in Section 3.3, the most straightforward way is just using the in-group strength θ_k-s as the prediction scores for the subgroups and calculating the overall score as follows:

$$\hat{y}_{ui} = \sum_k \phi_{uk} \cdot \phi_{ik} \cdot \theta_k \tag{16}$$

Comparing with Equation 15, this strategy sets all $\hat{y}_{ui}^k$ to 1. From another point of view, it can be thought as applying the *random* prediction (all the items have the same prediction score 1) base method in the subgroups.

CCCF-AVG. This is the strategy used in [32], where the average predicted score is taken as the overall score, ignoring the affiliation strengths of users and items.

$$\hat{y}_{ui} = \sum_k \hat{y}_{ui}^k / K \tag{17}$$

CCCF-MAX. As discussed in Section 3.2, a user would like an item as long as the probability on one of the subgroup is large enough. Another strategies is to make the prediction using the maximum score among these subgroups.

$$\hat{y}_{ui} = \max_k \hat{y}_{ui}^k \tag{18}$$

3.5 Discussion

In this subsection, we discuss several properties of the proposed model.

- **Scalability.** The complexity of the proposed CCCF method is linear in the number of observed links. On the other side, the objective function of the method proposed in [25] is NP-hard, and their optimization method for the relaxed objective involves a step of solving the top eigenvectors of a squared matrix with size $M = U + I$. To our best knowledge, the computational complexity of the fastest eigenvalue decomposition solver is $O(M^{2.367})$ when the matrix has some special structures [14].
- **Flexibility.** The structure of the subgroups in CCCF is flexible to model any kind of overlapping subgroups where these subgroups can be densely overlapping, hierarchically nested or non-overlapping. The methods in [32] and [31] are not so flexible since they assume the subgroups to form (approximate) Bordered Block Diagonal matrices. The method proposed in [25] uses an objective function of a weighted graph cut problem, which is designed to partition the graph into non-overlapping parts, thus it does not intrinsically discover overlapping subgroups.
- **Interpretability.** CCCF is a probabilistic model where the resulting affiliation strengths have semantic meanings. The strengths are particularly important when aggregating the results from different subgroups to make final recommendation (see Section 3.4). Also, as shown in Section 4, we analyze the top items with the largest affiliation strengths of each subgroup and find that these items have similar properties (*e.g.*, locations, categories in the Yelp data).
- **Extensibility.** When we are able to get some useful features for users and items, it is easy for CCCF to leverage the fea-

Table 2: Data Statistics

	#users	#items	#dyads	#density(%)
Lastfm	1.8K	1.5K	50K	1.76
MovieLens	35K	5.6K	3.4M	1.73
Netflix	46K	13K	8M	1.33
Yelp	2.8K	2.8K	80K	1.02

tures in the learning procedure as prior knowledge or supervisions (*e.g.*, [1], [29]). On one hand, it makes the subgroups even more interpretable by enforcing that the users/items in the same subgroup share some common features. On the other hand, features can help dealing with the cold-start problem – some users/items may have too few connections to be assigned to the right subgroups. It is not straightforward for the methods of [25] and [31] to achieve this goal.

4. EXPERIMENTS

Our experiments are designed to answer the following two questions: 1) Can the proposed method improve the accuracy of recommendation? 2) How meaningful are the discovered subgroups?

4.1 Data Sets and Preprocessing

We use four publicly available data sets: Last.fm[3], MovieLens 10M[4], Netflix[5] and Yelp[6]. As discussed in Section 2, we are more interested in the *implicit* feedback case. We follow the preprocessing steps that have been used in many recent works (*e.g.*, [15, 30], *et al.*.). For the data sets with explicit ratings, we remove the ratings of less than 4 stars and convert the remaining ratings to 1. We remove users and items with less than 20 ratings. The statistics of the resulting data sets are shown in Table 2. For each user, we randomly hold 20% of her/his feedback in the test data and use the rest of her/his ratings as training data.

4.2 Evaluation Measures

In the case of top-N recommendation, we present each user with N items that have the highest predicted values and have not been adopted by the user in the training data. We evaluate different approaches based on which of the items are actually adopted by the user in the test data.

Precision and Recall. Given a top-N recommendation list $C_{N,\text{rec}}$, precision and recall are defined as

$$\begin{aligned} \text{P@}N &= \frac{|C_{N,\text{rec}} \bigcap C_{\text{adopted}}|}{N} \\ \text{R@}N &= \frac{|C_{N,\text{rec}} \bigcap C_{\text{adopted}}|}{|C_{\text{adopted}}|}, \end{aligned} \tag{19}$$

where C_{adopted} are the items that a user has adopted in the test data. The precision and recall for the entire recommender system are computed by averaging the precision and recall over all the users, respectively.

F-measure. The F-measure represents a trade-off between precision and recall. We consider the F_β metric, which is defined as

$$\text{F}_\beta\text{@}N = (1+\beta^2) \cdot \frac{\text{P@}N \times \text{R@}N}{\beta^2 \cdot \text{P@}N + \text{R@}N}. \tag{20}$$

where β is a weight to control the balance. In our experiments, we use $F1$ metric with $\beta = 1$. We also calculate the $F1$ score for each user and report the average score.

[3] https://grouplens.org/datasets/hetrec-2011/
[4] https://grouplens.org/datasets/movielens/
[5] http://www.netflixprize.com/
[6] http://www.yelp.com/dataset_challenge/

Mean Average Precision (MAP). Average precision (AP) is a ranked precision metric that gives larger credit to correctly recommended items in higher positions. AP@N is defined as the average of precisions computed at all positions with an adopted item, namely,

$$\text{AP@}N = \frac{\sum_{k=1}^{N} \text{P@}k \times \text{rel}(k)}{\min\{N, |C_{\text{adopted}}|\}}, \tag{21}$$

where $\text{Precision}(k)$ is the precision at cut-off k in the top-N list $C_{N,\text{rec}}$, and $\text{rel}(k)$ is an indicator function equaling 1 if the item at rank k is adopted, otherwise it equals zero. Finally, Mean Average Precision (MAP@N) is defined as the mean of the AP@N scores of all users.

4.3 Accuracy of Top-N Recommendation

4.3.1 *Implementation Details*

We choose 4 popular CF methods as the comparison partners, and these 4 methods also serve as the base methods applied to each subgroup (as discussed in Section 3.3). We first train CCCF on the training data, and then apply the above CF methods in each subgroup. At last, for each user, we predict her/his preferences on all the unrated items by aggregating the predictions from the subgroups the user is involved in, and pick the Top-N items with largest predicted values to form the recommendation list. We set the hyperparameters of CCCF by cross validation on the training data. We found that the performance is not so sensitive to the choices of α and β. We empirically set them as: $\alpha_{k1} = 1$, $\alpha_{k2} = 1$, $\beta_1 = 10$, $\beta_2 = 1$ and $\epsilon = 0.1$. We will study how the choice of K impacts the performance in Section 4.3.5.

4.3.2 *CCCF vs. No Co-Clustering*

We first study the performance of CCCF comparing with the CF methods that are directly applied on the training data without co-clustering.

The results are shown in Table 3, 4 and 5. The overall observation is that CCCF improves the base methods on most of the cases – the best scores on all the data sets are all from CCCF. For most of the cases, CCCF improves the corresponding base method by at least 10%. Especially, combining CCCF with simple recommendation methods like POP can produce fairly good results.

We note that the performance of these CF methods varies a lot on different data sets. For example, on the Yelp data set, we observe that ItemCF performs better than MF and WARP; while WARP is the best method on the Netflix data set. This means that no single CF method can beat all others at all times. The advantage of the proposed framework that we do not rely on the underlying base method, and users of our method are free to configure their favorite predictors in order to adapt to different scenarios.

4.3.3 *CCCF vs. Other Co-Clustering Methods*

We compare CCCF against two other co-clustering based methods MCoC [25] and BBDF [31], which are the most recent co-clustering methods for recommender systems, and adopt the same three-step framework. For MCoC and BBDF, we carefully choose the best parameters (*e.g.*, the number of top eigenvectors for MCoC, the density level in BBDF) to ensure fair comparisons. For both CCCF and MCoC, we set the number of subgroups to 10. The number of subgroups in BBDF is determined by the density parameter, which is chosen by cross validation.

Figure 1 shows the results for F1@10 and MAP@10 on the four data sets. The general observation is that CCCF outperforms the other two co-clustering methods in most cases. The only exception is on the Yelp data set, where both MCoC and CCCF achieve a

Table 3: CCCF vs. No Co-Clustering on the Lastfm data

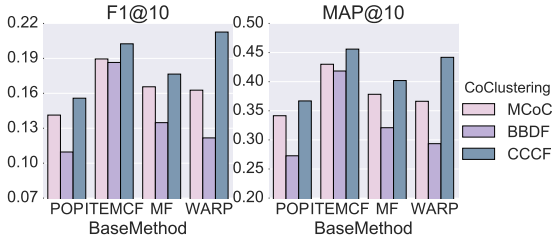
	POP		ITEMCF		MF		WARP	
	NONE	CCCF	NONE	CCCF	NONE	CCCF	NONE	CCCF
P@10	0.070	0.134	0.173	0.174	0.152	0.156	0.178	0.187
R@10	0.090	0.176	0.223	0.250	0.204	0.217	0.253	0.264
F1@10	0.078	0.149	0.189	0.198	0.170	0.177	0.203	0.213
MAP@10	0.182	0.355	0.446	0.456	0.399	0.402	0.428	0.442

Table 4: CCCF vs. No Co-Clustering on the Netflix data

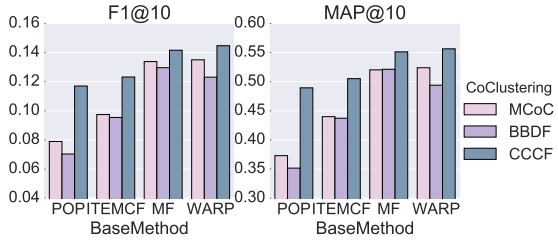
	POP		ITEMCF		MF		WARP	
	NONE	CCCF	NONE	CCCF	NONE	CCCF	NONE	CCCF
P@10	0.189	0.283	0.308	0.312	0.330	0.354	0.347	0.364
R@10	0.047	0.075	0.085	0.085	0.091	0.098	0.095	0.100
F1@10	0.070	0.109	0.123	0.123	0.132	0.142	0.138	0.145
MAP@10	0.348	0.471	0.493	0.505	0.528	0.551	0.538	0.556

Table 5: CCCF vs. No Co-Clustering on the Yelp data

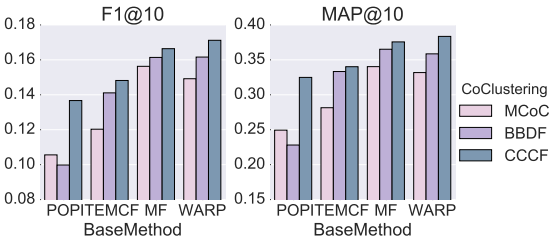
	POP		ITEMCF		MF		WARP	
	NONE	CCCF	NONE	CCCF	NONE	CCCF	NONE	CCCF
P@10	0.018	0.046	0.048	0.057	0.043	0.046	0.047	0.056
R@10	0.024	0.058	0.062	0.072	0.055	0.058	0.058	0.068
F1@10	0.018	0.046	0.048	0.058	0.044	0.046	0.047	0.055
MAP@10	0.050	0.124	0.142	0.159	0.124	0.131	0.137	0.152



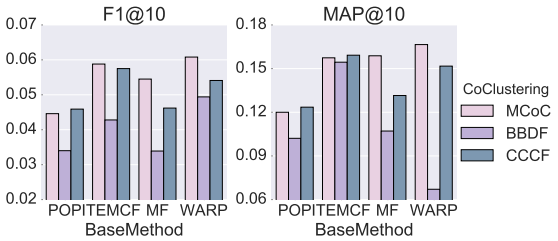
(a) Results on the Lastfm data



(b) Results on the Netflix data



(c) Results on the MovieLens data



(d) Results on the Yelp data

Figure 1: Comparison with other two co-clustering methods on the three data sets.

large performance gain compared to BBDF, and MCoC obtains the best results on this data set. This is likely due to the fact that the users and items of the Yelp data are isolated by the geographical distance, *i.e.*, they are more likely to form subgroups by the cities they are in.

Surprisingly, in some cases, MCoC and BBDF even decrease the performance of some of the base methods on the Lastfm, Netflix and MovieLens data. As discussed in Section 3.5, MCoC is actually optimizing a non-overlapping clustering objective function. So on the data sets that do not have obvious clustering structures, the clustering result of MCoC fails to boost recommendation accuracy. The performance of BBDF is not as good as expected on all the data sets except MovieLens. One reason may be that it assumes the user-item matrix follows some specific structure, and not all the data sets have this property. For example, for the Yelp data set, users and items are more likely to form non-overlapping clusters, due to the geographical separations of users and items. In this case, it is impossible for BBDF to find a reasonable bordered-block sub-matrix, which produces unsatisfactory results.

4.3.4 On the Choice of Aggregation Method

As discussed in Section 3.4, there are several alternative ways to aggregate recommendations from subgroups to make final recommendations: CCCF-PR, CCCF-MAX, CCCF-AVG. In this subsection, we compare these three alternatives with the method we proposed in Equation 15 (referred to as CCCF).

The MAP@10 results are shown in Figure 2. For the base method ITEMCF, CCCF-MAX can provide similar performance as CCCF,

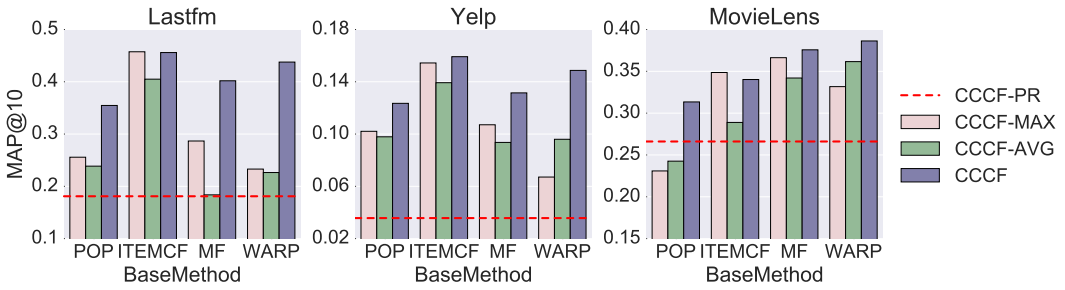
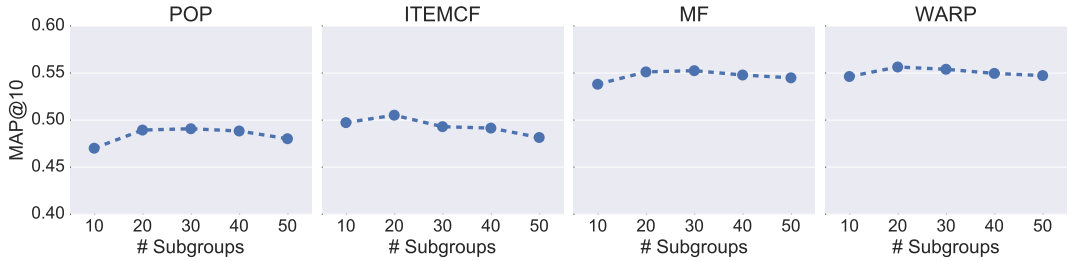
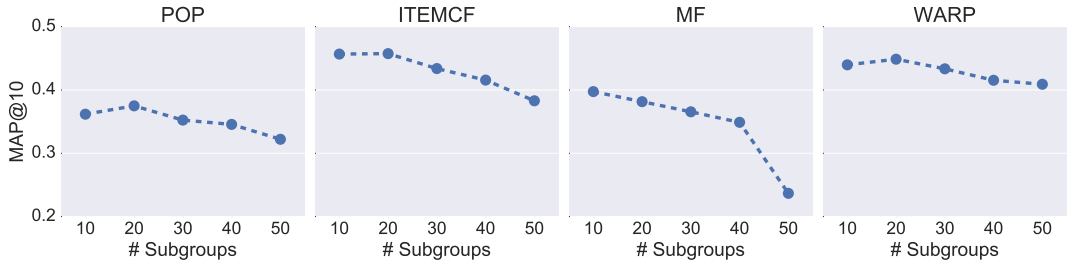


Figure 2: MAP@10 on the Lastfm and Yelp data with different strategies to aggregate the results from subgroups. Here we use the same co-clustering results and CF base methods. The only difference between the comparisons is the recommendation strategy. CCCF-PR does not need a base method and we plot it using the dashed line.



(a) MAP@10 on the Netflix data



(b) MAP@10 on the Lastfm data

Figure 3: MAP@10 on the Lastfm and Netflix data with different numbers of subgroups.

but its results for other methods are not competitive. For the Yelp data, CCCF-PR is outperformed by other aggregation methods by a large margin. This means that the step of applying a CF method in the subgroup is necessary. For almost all the cases, CCCF outperforms alternatives, which indicates that the steps of applying CF methods in each subgroup and considering the affiliation strengths are both essential when aggregating the results.

4.3.5 On the Number of Subgroups

In this subsection, we study how the choice of the number of subgroups K impacts the recommendation result. Due to space limits, we only show the results on the Lastfm and Netflix data. The results on MovieLens and Yelp data show similar trends. The results are shown in Figure 3.

We observe that for the Netflix data, setting K to 20 is a good choice where all the four base methods get best results. The Netflix data is not so sensitive to the number of subgroups. For the Lastfm data, 10 and 20 are the best choices. Increasing the number decreases the performance.

We note that increasing K would not necessarily increase the recommendation performance. The user-item matrix is very sparse, and increasing the number of subgroups would keep splitting large subgroups into several more focused small overlapping groups. Taking an extreme case for example, we might learn a lot of small subgroups that all of the users and items have links with each other. This case produces perfect co-clusters, but we do not have *new* items to recommend to users, which would hurt the recommendation performance.

4.4 Analysis of the Co-Clustering Results

In this section, we conduct a qualitative analysis of the co-clustering results to provide anecdotal evidence that the discovered subgroups are meaningful.

The goal of our method is to find focused subgroups, where the items in the same subgroup share some common latent properties that attract a group of like-minded users. Users' interests are usually not explicit, and we only have limited information about users in the data sets due to privacy issues. It is easier and more straightforward to study the item properties because we have more detailed information about items.

Table 6 shows the items with largest affiliation strengths from 3 sample subgroups from 10 total subgroups. We also present their locations and categories to study what these items are. We can see that the items from each subgroup share similar geographical and semantic properties. The top items from the first subgroups are all businesses from the city *Phoenix*, and most of them are restaurants. We can regard this subgroup as "Restaurants in *Phoenix*". Similarly, the top items from the second subgroup are also restaurants,

Table 6: Items with largest affiliation strengths from 3 out of 10 subgroups on the Yelp data. We show their locations and the most representative categories that these items belong to.

Sample Cluster 1			Sample Cluster 2			Sample Cluster 3		
Business Name	**City**	**Description**	**Business Name**	**City**	**Description**	**Business Name**	**City**	**Description**
Gallo Blanco	Phoenix	Restaurants	Bread and Butter	Henderson	Restaurants	Las Vegas North Premium Outlets	Las Vegas	Shopping
Lux	Phoenix	Restaurants	Snow Ono Shave Ice	Las Vegas	Restaurants	Orleans Hotel & Casino	Las Vegas	Casinos & Hotels
Postino Central	Phoenix	Restaurants	Buldogis Gourmet Hot Dogs	Las Vegas	Restaurants	Fremont Street Experience	Las Vegas	Entertainment
Maizie's Cafe & Bistro	Phoenix	Restaurants	Sweet Tomatoes	Las Vegas	Restaurants	Conservatory & Botanical Garden	Las Vegas	Entertainment
Cherryblossom Noodle Cafe	Phoenix	Restaurants	Strip N Dip Chicken Strips	Las Vegas	Restaurants	Planet Hollywood Las Vegas Resort & Casino	Las Vegas	Casinos & Hotels
Pizza a Metro	Phoenix	Restaurants	Patisserie Manon	Las Vegas	Restaurants	Flamingo Las Vegas Hotel & Casino	Las Vegas	Casinos & Hotels
SideBar	Phoenix	Bars	Island Flavor	Las Vegas	Restaurants	New York - New York	Las Vegas	Casinos & Hotels
Churn	Phoenix	Coffee & Tea	Japanese Curry Zen	Las Vegas	Restaurants	Miracle Mile Shops	Las Vegas	Shopping
Carly's Bistro	Phoenix	Restaurants	Slidin' Thru	Las Vegas	Restaurants	Palms Casino Resort	Las Vegas	Casinos & Hotels
Federal Pizza	Phoenix	Restaurants	Art of Flavors	Las Vegas	Food	The Forum Shops	Las Vegas	Shopping

but they are all in the greater Las Vegas area (*Henderson* is a city adjacent to *Las Vegas*). The third subgroup contains hotels, casinos or attractions in *Las Vegas*. The users from this subgroup are more likely to be tourists of *Las Vegas*, while the users of the second subgroup may also be the *Las Vegas* locals.

5. RELATED WORK

Using co-clustering based methods to improve CF has been studied by several previous works. [26] uses the K-means algorithm to cluster users into several subgroups and uses the subgroup assignments to smooth the unrated data for each individual user. [6] proposes a scalable CF framework based on a weighted co-clustering method.

Bayesian Co-Clustering (BCC) [18] adopts the Mixed Membership Stochastic Blockmodels (MMSB) [2] to the user-item bipartite graph. They assume that there are several latent user clusters and item clusters, and each user/item has mixed memberships to the latent clusters. The link between a user and an item is generated by the sum of the coefficient weights between the clusters they belong to. [20] proposes a collapsed Gibbs sampling and a collapsed variational Bayesian algorithm for BCC. [21] further extends BCC to a nonparametric model that can automatically learn the number of clusters from the data. In [4], the user and item latent factors are described by a nonparametric mixture of Gaussians based on their cluster allocations. [3] uses additive co-clustering to build concise model for matrix approximation. In these works, users and items are *separately* partitioned into user clusters and item clusters, which is totally different from co-clustering method used in this paper which groups users and items into the same clusters in a holistic manner. Also, these works optimize either the likelihood of the probabilistic mixed membership model on all the user-item links [18, 20, 21], or the accuracy of the point-wise rating prediction [4, 3], which can not guarantee good top-N recommendation results as they are optimizing a different goal [15, 19]. On the other hand, our method can utilize state-of-the-art top-N recommendation models as the base CF method in subgroups.

The idea of divide-and-conquer has also been used in other related works for collaborative filtering [11, 10, 9]. Divide Factor Combine (DFC) [11] divides a large-scale matrix factorization task into smaller subproblems by random row/column selections, solves each subproblem in parallel using an arbitrary base matrix factorization algorithm, and combines the subproblem solutions using techniques from randomized matrix approximation. The authors of [10] propose a model called Local Low-Rank Matrix Approximation (LLORMA), leading to a representation of the observed matrix as a weighted sum of several local low-rank matrices. They further extend LLORMA to the Local Collaborative Ranking (LCR) model [9] to consider the relative orders of items. The main differences from ours are: 1) They partition the user-item matrix by either random row/column sampling [11] or random anchor points [10, 9], while our goal is to find locally focused subgroups. 2) They focus on the rating prediction [11, 10] and collaborative ranking [9] problems on observed ratings[7] and ignore the fact that ratings are missing not at random. It leads to unsatisfactory top-N recommendation results [15, 19]. 3) They rely on the matrix factorization techniques, while ours can apply any CF method in the subgroups. This is important because matrix factorization is not always the best choice for all the cases, which has also been shown in our experiments (see Section 4.3.2).

The closest works are [25, 31, 32], which propose models that cluster users and items into the same subgroups. The authors of [25] propose an overlapping user-item co-clustering method based on a weighted graph-cut optimization problem. The objective function is NP-Hard, and they propose to optimize a relaxed problem. However, the learning involves solving the top eigenvectors of a large squared matrix so it cannot scale to large data sets. In [31] and [32], the authors permute rows and columns of the user-item rating matrix to form (approximate) Bordered Block Diagonal (BBD) matrices, and then apply traditional CF to each BBD matrix.

The framework used in our paper, as well as in [25, 31], has several benefits. 1) It partitions the matrix into several overlapping sub-matrices, which are denser than the original matrix, thus making the CF methods more feasible. 2) Any CF method can be used in each subgroup. Since the performance of different methods varies under different scenarios, this allows users to select their favorite CF base methods for the subgroups. 3) The training of the CF methods in subgroups can be trivially parallelized.

As discussed in Section 3.5, compared with previous works [25, 31], CCCF has various benefits including scalability, flexibility, interpretability and extensibility.

[7] Although LCR [9] models the orders of items, it still only considers the ranking between *observed* ratings.

6. CONCLUSION

In this paper, we proposed the scalable co-clustering method CCCF to improve the performance of CF-based methods for Top-N recommendation. The intuition of our model is that users may have different preferences over different subsets of items, where these subsets of items may overlap, and we explore subgroups of users who share interests over similar subsets of items through overlapping clustering. Experimental results on four real world data sets demonstrate that CCCF can improve four representative CF base methods (POP, ITMECF, MF, WARP) by co-clustering, and it also outperforms other co-clustering based methods (MCoC and BBDF). Qualitative analysis on the data sets also shows that the discovered subgroups are semantically meaningful.

A potential future work is to combine the idea of CCCF with Locally Collaborative Ranking (LCR) [9], where when applying the CF methods to the subgroups, the group affiliation strengths could also considered. Our model is complementary to LCR in the sense that it can replace the randomized anchor points selection and define the distance function using the joint user-item affiliation strengths. Also, it would be interesting to incorporate user and item features into the CCCF model in order to further improve the clustering results and to make them more interpretable. Besides, in this paper, the number of subgroups is a pre-defined hyperparameter. How to automatically determine the best number of subgroups using a Bayesian nonparametric method is also worth exploring.

7. REFERENCES

[1] D. Agarwal and B.-C. Chen. Regression-based latent factor models. In *Proceedings of the 15th ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 19–28. ACM, 2009.

[2] E. M. Airoldi, D. M. Blei, S. E. Fienberg, and E. P. Xing. Mixed membership stochastic blockmodels. In *Advances in Neural Information Processing Systems*, pages 33–40, 2009.

[3] A. Beutel, A. Ahmed, and A. J. Smola. Accams: Additive co-clustering to approximate matrices succinctly. In *Proceedings of the 24th International Conference on World Wide Web*, pages 119–129, 2015.

[4] A. Beutel, K. Murray, C. Faloutsos, and A. J. Smola. Cobafi: collaborative bayesian filtering. In *Proceedings of the 23rd international conference on World wide web*, pages 97–108, 2014.

[5] J. Foulds, L. Boyles, C. DuBois, P. Smyth, and M. Welling. Stochastic collapsed variational bayesian inference for latent dirichlet allocation. In *Proceedings of the 19th ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 446–454. ACM, 2013.

[6] T. George and S. Merugu. A scalable collaborative filtering framework based on co-clustering. In *ICDM'05*. IEEE, 2005.

[7] L. Hong, A. S. Doumith, and B. D. Davison. Co-factorization machines: modeling user interests and predicting individual decisions in twitter. In *Proceedings of the sixth ACM international conference on Web search and data mining*, pages 557–566. ACM, 2013.

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

[9] J. Lee, S. Bengio, S. Kim, G. Lebanon, and Y. Singer. Local collaborative ranking. In *Proceedings of the 23rd international conference on World wide web*, pages 85–96, 2014.

[10] J. Lee, S. Kim, G. Lebanon, and Y. Singer. Local low-rank matrix approximation. In *Proceedings of The 30th International Conference on Machine Learning*, pages 82–90, 2013.

[11] L. W. Mackey, M. I. Jordan, and A. Talwalkar. Divide-and-conquer matrix factorization. In *Advances in Neural Information Processing Systems*, pages 1134–1142, 2011.

[12] A. Mnih and R. Salakhutdinov. Probabilistic matrix factorization. In *Advances in neural information processing systems*, pages 1257–1264, 2007.

[13] M. O'Connor and J. Herlocker. Clustering items for collaborative filtering. In *Proceedings of the ACM SIGIR workshop on recommender systems*, volume 128, 1999.

[14] V. Y. Pan and Z. Q. Chen. The complexity of the matrix eigenproblem. In *Proceedings of the Thirty-first Annual ACM Symposium on Theory of Computing*, STOC '99, pages 507–516. ACM, 1999.

[15] S. Rendle, C. Freudenthaler, Z. Gantner, and L. Schmidt-Thieme. Bpr: Bayesian personalized ranking from implicit feedback. In *Proceedings of the Twenty-Fifth Conference on Uncertainty in Artificial Intelligence*, pages 452–461, 2009.

[16] F. Ricci, L. Rokach, and B. Shapira. *Introduction to recommender systems handbook*. Springer, 2011.

[17] B. Sarwar, G. Karypis, J. Konstan, and J. Riedl. Item-based collaborative filtering recommendation algorithms. In *Proceedings of the 10th International Conference on World Wide Web*, pages 285–295. ACM, 2001.

[18] H. Shan and A. Banerjee. Bayesian co-clustering. In *ICDM'08*, pages 530–539. IEEE, 2008.

[19] H. Steck. Training and testing of recommender systems on data missing not at random. In *Proceedings of the 16th ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 713–722. ACM, 2010.

[20] P. Wang, C. Domeniconi, and K. B. Laskey. Latent dirichlet bayesian co-clustering. In *Machine Learning and Knowledge Discovery in Databases*, pages 522–537. Springer, 2009.

[21] P. Wang, K. B. Laskey, C. Domeniconi, and M. I. Jordan. Nonparametric bayesian co-clustering ensembles. In *SDM*. SIAM, 2011.

[22] J. Weston, S. Bengio, and N. Usunier. Wsabie: Scaling up to large vocabulary image annotation. In *IJCAI*, pages 2764–2770, 2011.

[23] J. Weston, C. Wang, R. Weiss, and A. Berenzweig. Latent collaborative retrieval. In *Proceedings of the 29th International Conference on Machine Learning*, pages 9–16, 2012.

[24] J. Weston, R. J. Weiss, and H. Yee. Nonlinear latent factorization by embedding multiple user interests. In *Proceedings of the 7th ACM conference on Recommender systems*, pages 65–68. ACM, 2013.

[25] B. Xu, J. Bu, C. Chen, and D. Cai. An exploration of improving collaborative recommender systems via user-item subgroups. In *Proceedings of the 21st International Conference on World Wide Web*, pages 21–30. ACM, 2012.

[26] G.-R. Xue, C. Lin, Q. Yang, W. Xi, H.-J. Zeng, Y. Yu, and Z. Chen. Scalable collaborative filtering using cluster-based smoothing. In *Proceedings of the 28th annual international ACM SIGIR conference on Research and development in information retrieval*, pages 114–121. ACM, 2005.

[27] J. Yang and J. Leskovec. Community-affiliation graph model for overlapping network community detection. In *ICDM'12*, pages 1170–1175. IEEE, 2012.

[28] J. Yang and J. Leskovec. Overlapping community detection at scale: A nonnegative matrix factorization approach. In *Proceedings of the Sixth ACM International Conference on Web Search and Data Mining*, pages 587–596. ACM, 2013.

[29] J. Yang, J. McAuley, and J. Leskovec. Community detection in networks with node attributes. In *ICDM'13*, pages 1151–1156. IEEE, 2013.

[30] S.-H. Yang, B. Long, A. J. Smola, H. Zha, and Z. Zheng. Collaborative competitive filtering: learning recommender using context of user choice. In *Proceedings of the 34th international ACM SIGIR conference on Research and development in Information Retrieval*, pages 295–304. ACM, 2011.

[31] Y. Zhang, M. Zhang, Y. Liu, and S. Ma. Improve collaborative filtering through bordered block diagonal form matrices. In *Proceedings of the 36th international ACM SIGIR conference on Research and development in information retrieval*, pages 313–322. ACM, 2013.

[32] Y. Zhang, M. Zhang, Y. Liu, S. Ma, and S. Feng. Localized matrix factorization for recommendation based on matrix block diagonal forms. In *Proceedings of the 22Nd International Conference on World Wide Web*, pages 1511–1520. ACM, 2013.