JMLR: Workshop and Conference Proceedings 45:1–16, 2015 ACML 2015

Preference Relation-based Markov Random Fields for Recommender Systems

Shaowu Liu S.LIU@DEAKIN.EDU.AU
School of Information Technology
Deakin University, Geelong, Australia

Gang Li GANG.LI@DEAKIN.EDU.AU
School of Information Technology
Deakin University, Geelong, Australia

Truyen Tran TRUYEN.TRAN@DEAKIN.EDU.AU
Pattern Recognition and Data Analytics
Deakin University, Geelong, Australia

Yuan Jiang JIANGYUAN@NJU.EDU.CN
National Key Laboratory for Novel Software Technology
Nanjing University, China

Abstract

A *preference relation*-based Top-N recommendation approach, *PrefMRF*, is proposed to capture both the second-order and the higher-order interactions among users and items. Traditionally Top-N recommendation was achieved by predicting the item ratings first, and then inferring the item rankings, based on the assumption of availability of *explicit* feedbacks such as ratings, and the assumption that optimizing the ratings is equivalent to optimizing the item rankings. Nevertheless, both assumptions are not always true in real world applications. The proposed *PrefMRF* approach drops these assumptions by explicitly exploiting the preference relations, a more practical user feedback. Comparing to related work, the proposed *PrefMRF* approach has the unique property of modeling both the second-order and the higher-order interactions among users and items. To the best of our knowledge, this is the first time both types of interactions have been captured in *preference relation*-based method. Experiment results on public datasets demonstrate that both types of interactions have been properly captured, and significantly improved Top-N recommendation performance has been achieved.

Keywords: Preference Relation, Pairwise Preference, Markov Random Fields, Collaborative Filtering, Recommender Systems

1. Introduction

Recommender Systems (RecSys) aim to recommend users with some of their potentially interesting items, which can be virtually anything ranging from *movies* to *tourism attractions*. To identify the appropriate items, RecSys attempts to exploit various information including *user preferences* (Koren et al., 2009) and *side information* (Balabanović and Shoham, 1997). By far, *Collaborative Filtering* (Koren et al., 2009) is one of the most popular RecSys techniques, which exploits user preferences, especially in form of *absolute ratings*.

Recently, a considerable literature (Liu et al., 2009; Rendle et al., 2009; Desarkar et al., 2012; Brun et al., 2010; Shi et al., 2010) has grown up around the theme of *relative* preferences. The underlying motivation is that *relative* preferences are often easier to collect and more reliable as a measure of user preferences. For example, it can be easier for users to tell which item is preferable than expressing the precise degree of liking. Furthermore, studies (Koren and Sill, 2011; Brun et al., 2010) have reported that absolute ratings may not be completely trustworthy. For example, rating 4 out of 5 may in general indicate high quality, but it can mean just OK for critics. In fact, users' quantitative judgment can be affected by irrelevant factors such as the *mood* when rating. While users are not good at making consistent quantitative judgment, the *preference relation* (PR), as a kind of relative preference, has been considered as more consistent across like-minded users (Brun et al., 2010; Desarkar et al., 2012). By measuring the relative order between items, the *PR* is usually invariant to irrelevant factors. For example, a user in bad *mood* may give lower ratings but the relative ordering between items remains the same. In addition, as the ultimate goal of *RecSys*, obtaining the ranking of items by itself is to obtain the relative preferences, a more natural input than absolute ratings (Koren and Sill, 2011).

While the *PR* captures the user preferences in the *pairwise* form, most existing works (Koren and Sill, 2011; Liu et al., 2014) take the *pointwise* approach to exploiting ordinal properties possessed by absolute ratings. To accept the *PR* as input and output item rankings, *pairwise* approaches have emerged in two forms: memory-based (Brun et al., 2010) and model-based (Liu et al., 2009; Rendle et al., 2009; Desarkar et al., 2012). These studies show the feasibility of *PR*-based methods, and demonstrated competitive performance comparing to their underlying models, such as memory-based *K-Nearest Neighbor* (KNN) (Brun et al., 2010) and model-based *Matrix Factorization* (MF) (Desarkar et al., 2012).

However, the limitations of these underlying models have constrained the potentials of their *PR* extensions. Specifically, both *KNN* and *MF* based methods can only capture one type of information at a time, while both the *local* and the *global* information are essential in achieving good performance (Tran et al., 2009; Koren, 2008; Liu et al., 2014):

Local Structure The *local structure* (LS) refers to the second-order interactions between similar users (Resnick et al., 1994) or items (Sarwar et al., 2001). *LS*-based approaches ignore the majority of preferences in making predictions, but are effective when the users/items correlations are highly localized.

Global Structure The *global structure* (GS) refers to the weaker but higher-order interactions among all users and items (Koren et al., 2009). *GS*-based approaches are often competitive in terms of accuracy and computational efficiency (Koren et al., 2009).

Previous studies have suggested that these two structures are complementary since they address different aspects of the preferences (Tran et al., 2009; Koren, 2008; Liu et al., 2014). However, to the best of our knowledge, there is yet no *PR*-based method that can capture both *LS* and *GS*. All the above reasonings lead to the desired model with the following properties: *1*) Accept *PR* as input; *2*) Capture both *LS* and *GS*; *3*) Output item rankings.

Recent advances in *Markov Random Fields*-based RecSys (Tran et al., 2009; Defazio and Caetano, 2012; Liu et al., 2014) have made it possible to achieve the above objectives. *MRF*-based RecSys was first developed in (Tran et al., 2009) to capture both *LS* and *GS*.

Later on, it has been extended in (Liu et al., 2014) to exploit ordinal properties possessed by absolute ratings. Nevertheless, all of these attempts rely on absolute ratings.

This paper aims to push the *MRF*-based RecSys one step further by fitting it into the *PR* framework, namely the *Preference Relation-based Markov Random Fields* (PrefMRF). The remaining part of this paper is organized as follows. Section 2 introduces the concepts of *PR*-based RecSys and formalizes the problem, followed by a review of related work. Section 3 is devoted to the proposed *PrefMRF* model. Benchmark results on Top-N recommendation are presented in Section 4. Finally, Section 5 concludes this paper by summarizing the main contributions and envisaging future works.

2. Preliminaries and Related Work

Recommender Systems (RecSys) aim at predicting users' future interest in items, and the recommendation task can be considered as a *preference learning* problem, which aims to construct a predictive preference model from observed preference information (Mohri et al., 2012). Existing *preference learning* methods are based on different *learning to rank* approaches (Fürnkranz and Hüllermeier, 2010). Among them, the *pointwise* approach is the choice of most RecSys (Sarwar et al., 2001; Koren, 2008), which exploit absolute ratings, though *pairwise* approach that exploits *PR* has been largely overlooked until recently. The rest of this section describes the basic concepts and formalizes the *PR*-based RecSys.

2.1. Preference Relation

A *preference relation* (PR) encodes user preferences in form of pairwise ordering between items. This representation is a useful alternative to absolute ratings for three reasons.

Firstly, *PR* is more consistent across like-minded users (Brun et al., 2010; Desarkar et al., 2012) as it is invariant to many irrelevant factors, such as *mood*. Secondly, *PR* is a more natural and direct input for Top-N recommendation, as both the input and the output are relative preferences. Finally, and perhaps most importantly, *PR* can be obtained implicitly rather than asking the users explicitly. For example, the *PR* over two *Web pages* can be inferred by the *stayed time*, and consequently applies to the displayed items. This property is important as not all users are willing to rate their preferences, where collecting feedbacks implicitly delivers a more user-friendly RecSys. With these potential benefits, we shall take a closer look at the *PR*, and investigate how they can be utilized in RecSys.

We formally define the *PR* as follows. Let $\mathcal{U} = \{u\}^n$ and $\mathcal{I} = \{i\}^m$ denote the set of n users and m items, respectively. The preference of a user $u \in \mathcal{U}$ between items i and j is encoded as π_{uij}, which indicates the strength of user u's *PR* for the ordered item pair (i, j). A higher value of π_{uij} indicates a stronger preference on the first item over the second item.

Definition 1 (Preference Relation) *The preference relation is defined as*

$$\pi_{uij} = \begin{cases} (\frac{2}{3}, 1] & \textit{if } i \succ j \ (u \textit{ prefers } i \textit{ over } j) \\ [\frac{1}{3}, \frac{2}{3}] & \textit{if } i \simeq j \ (i \textit{ and } j \textit{ are equally preferable to } u) \\ [0, \frac{1}{3}) & \textit{if } i \prec j \ (u \textit{ prefers } j \textit{ over } i) \end{cases} \tag{2.1}$$

where $\pi_{uij} \in [0, 1]$ *and* $\pi_{uij} = 1 - \pi_{uji}$.

This definition is similar to (Desarkar et al., 2012), however, we allocate an interval for each preference category, i.e., *preferred*, *equally preferred*, and *less preferred*. Indeed, each preference category can be further break down into more intervals. Similar to (Brun et al., 2010), the *PR* can be converted into user-wise preferences over items.

Definition 2 (User-wise Preference) *The user-wise preference is defined as*

$$p_{ui} = \frac{\sum_{j \in \mathcal{I}_u} [\![\pi_{uij} > \frac{2}{3}]\!] - \sum_{j \in \mathcal{I}_u} [\![\pi_{uij} < \frac{1}{3}]\!]}{|\Pi_{ui}|} \tag{2.2}$$

where $[\![\cdot]\!]$ *gives* 1 *for* true *and* 0 *for* false, Π_{ui} *is the set of user* u*'s PR related to item* i.

The user-wise preference p_{ui} falls in the interval $[-1, 1]$, where -1 and 1 indicate that item i is the least or the most preferred item for u, respectively. The user-wise preference measures the relative position of an item for a particular user.

2.2. Problem Statement

Generally, the task of *PR*-based RecSys is to take *PR* as input and output Top-N recommendations. Specifically, let $\pi_{uij} \in \Pi$ encode the *PR* of each user $u \in \mathcal{U}$. Each π_{uij} is defined over an ordered item pair (i, j), denoting $i \prec j$, $i \simeq j$, or $i \succ j$. The goal is to estimate the value of each unknown $\pi_{uij} \in \Pi_{unknown}$, such that $\hat{\pi}_{uij}$ approximates π_{uij}. This can be considered as an optimization task performs directly on the *PR*

$$\hat{\pi}_{uij} = \underset{\hat{\pi}_{uij} \in [0,1]}{\arg\min} (|\pi_{uij} - \hat{\pi}_{uij}|) \tag{2.3}$$

However, it can be easier to estimate the $\hat{\pi}_{uij}$ by the difference between the two user-wise preferences p_{ui} and p_{uj}, i.e., $\hat{\pi}_{uij} = \phi(\hat{p}_{ui} - \hat{p}_{uj})$, where $\phi(\cdot)$ is a function that bounds the value into $[0, 1]$ and ensures $\phi(0) = 0.5$. For example, the *inverse-logit* function $\phi(x) = \frac{e^x}{1+e^x}$ can be used when user-wise preferences involve large values. Therefore, the objective of this paper is to solve the following optimization problem:

$$(\hat{p}_{ui}, \hat{p}_{uj}) = \underset{\hat{p}_{ui}, \hat{p}_{uj}}{\arg\min} (|\pi_{uij} - \phi(\hat{p}_{ui} - \hat{p}_{uj})|) \tag{2.4}$$

which optimizes the user-wise preferences directly, and Top-N recommendations can be obtained by simply sorting the estimated user-wise preferences.

2.3. Related Work

User preferences can be modeled in three types: *pointwise*, *pairwise*, and *listwise*. Though RecSys is not limited to *pointwise* absolute ratings, the recommendation task is usually considered as a rating prediction problem. Recently, a considerable literature (Liu et al., 2009; Rendle et al., 2009; Desarkar et al., 2012; Brun et al., 2010; Shi et al., 2010) has grown up around the theme of *relative* preferences, especially the *pairwise* *PR*. Meanwhile, recommendation task is also shifting from rating prediction to item ranking (Weimer et al., 2007; Shi et al., 2010), in which the ranking itself is also *relative* preferences.

The use of *relative* preferences has been widely studied in the field of *Information Retrieval* for *learning to rank* tasks. Recently, *PR*-based (Liu et al., 2009; Rendle et al., 2009; Desarkar et al., 2012; Brun et al., 2010) and *listwise*-based (Shi et al., 2010) RecSys have been proposed. Among them, the *PR*-based approach is the most popular, which can be further categorized as *memory-based* methods (Brun et al., 2010) that capture *local structure* and *model-based* methods (Liu et al., 2009; Rendle et al., 2009; Desarkar et al., 2012) that capture *global structure*. To the best of our knowledge, there is yet no *PR*-based method that can capture both *LS* and *GS*.

Advances in *Markov Random Fields* (MRF) have made it possible to utilize both *LS* and *GS* by taking advantages of MRF's powerful representation capability. Nevertheless, exploiting the *PR* is not an easy task for *MRF* (Tran et al., 2009; Liu et al., 2014). This observation leads to a natural extension of unifying the MRF method with the *PR*-based methods, to complement their strengths. We summarize the capabilities of the existing and our proposed *PrefMRF* methods in Table 1.

Table 1: Capabilities of Different Methods

Method	Input	Output	LS	GS
Pointwise Memory-based	Ratings	Ratings	✓	
Pointwise Model-based	Ratings	Ratings		✓
Pointwise Hybrid	Ratings	Ratings	✓	✓
Pairwise Memory-based	Preference Relations	Item Rankings	✓	
Pairwise Model-based	Preference Relations	Item Rankings		✓
PrefMRF	**Preference Relations**	**Item Rankings**	✔	✔

3. Preference Relation-based Markov Random Fields

In this section, we propose the *Preference Relation-based Markov Random Fields* (PrefMRF) to model the *PR* and capture both *LS* and *GS*. In this work, we exploit *LS* in terms of the item-item correlations only, while the user-user correlations can be modeled similarly. The rest of this section introduces the concept of the *PrefNMF* (Desarkar et al., 2012) that will be our underlying model, and then followed a detailed discussion of the *PrefMRF* on issues such as feature design, parameter estimation, and predictions.

3.1. Preference Relation-based Matrix Factorization

Matrix Factorization (MF) (Koren et al., 2009) is a popular approach to *RecSys* that has mainly been applied to absolute ratings. Recently, the *PrefNMF* (Desarkar et al., 2012) model was proposed to adopt *PR* input for *MF* models. The *PrefNMF* model discovers the latent factor space shared between users and items, where the latent factors describe both the *taste* of users and the *characteristics* of items. The attractiveness of an item to a user is then measured by the inner product of their latent feature vectors.

Formally, each user u is associated with a latent feature vector $\mathbf{u}_u \in \mathbb{R}^k$ and each item i is associated with a latent feature vector $\mathbf{v}_i \in \mathbb{R}^k$, where k is the dimension of the latent factor space. The attractiveness of items i and j to the user u are $\mathbf{u}_u^\top \mathbf{v}_i$ and $\mathbf{u}_u^\top \mathbf{v}_j$, respectively. When $\mathbf{u}_u^\top \mathbf{v}_i > \mathbf{u}_u^\top \mathbf{v}_j$ the item i is said to be more preferable to the user u than the item j, i.e., $i \succ j$. The strength of this preference relation π_{uij} can be estimated by $\mathbf{u}_u^\top(\mathbf{v}_i - \mathbf{v}_j)$, and the *inverse-logit* function is applied to ensure $\hat{\pi}_{uij} \in [0, 1]$:

$$\hat{\pi}_{uij} = \frac{e^{\mathbf{u}_u^\top(\mathbf{v}_i - \mathbf{v}_j)}}{1 + e^{\mathbf{u}_u^\top(\mathbf{v}_i - \mathbf{v}_j)}} \tag{3.1}$$

The latent feature vectors $\mathbf{u}_u$ and $\mathbf{v}_i$ are learned by minimizing regularized squared error with respect to the set of all known preference relations Π:

$$\min_{\mathbf{u}_u, \mathbf{v}_i \in \mathbb{R}^k} \sum_{\pi_{uij} \in \Pi \wedge (i<j)} (\pi_{uij} - \hat{\pi}_{uij})^2 + \lambda(\|\mathbf{u}_u\|^2 + \|\mathbf{v}_i\|^2) \tag{3.2}$$

where λ is the regularization coefficient. The optimization can be done with *Stochastic Gradient Descent* for the favor of speed on sparse data, or with *Alternating Least Squares* for the favor of parallelization on dense data.

3.2. Markov Random Fields

Markov Random Fields (MRF) (Tran et al., 2007; Defazio and Caetano, 2012) model a set of random variables having Markov property with respect to an undirected graph $\mathcal{G}$. The undirected graph $\mathcal{G}$ consists a set of vertices $\mathcal{V}$ connected by a set of edges $\mathcal{E}$ without orientation, where two vertices are neighborhood of each other when connected. Each vertex in $\mathcal{V}$ encodes a random variable, and the Markov property implies that a variable is conditionally independent of others given its neighborhoods.

In this work, we use *MRF* to model user-wise preference and their interactions respect to a set of undirected graphs. Specifically for each user u, there is a graph $\mathcal{G}_u$ with a set of vertices $\mathcal{V}_u$ and a set of edges $\mathcal{E}_u$. Each vertex in $\mathcal{V}_u$ represents a preference p_{ui} of user u on the item i. Note that the term *preference* is used instead of *rating* because in the new model the *preference* is not interpolated as absolute ratings but user-wise ordering of items. Each edge in $\mathcal{E}_u$ captures a relation between two preferences by the same user.

As we consider only the item-item correlations in this work, two preferences are connected by an edge if they are given by the same user. Fig. 1 shows an example of two graphs for users u and v. Note that vertices of different graphs are not connected directly, however, the edges between the same pair of items are associated to the same item-item correlation. For example, the edge between p_{ui} and p_{uj} and the edge between p_{vi} and p_{vj} are associated to the same item-item correlation ψ_{ij} between items i and j.

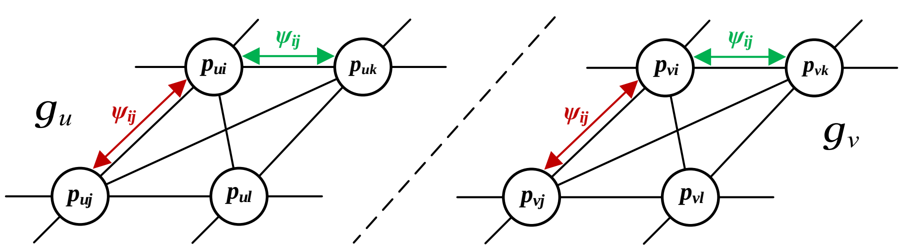


Figure 1: Example of undirected graphs for users u and v

Formally, let $\mathcal{I}_u$ be the set of all items evaluated by the user u and $\mathbf{p}_u = \{p_{ui} \mid i \in \mathcal{I}_u\}$ be the joint set of all preferences (the variables) expressed by user u, then the *MRF* defines a distribution $P(\mathbf{p}_u)$ over the graph $\mathcal{G}_u$:

$$P(\mathbf{p}_u) = \frac{1}{Z_u}\Psi(\mathbf{p}_u) \tag{3.3}$$

$$\Psi(\mathbf{p}_u) = \prod_{(ui,uj)\in\mathcal{E}_u} \psi_{ij}(p_{ui}, p_{uj}) \tag{3.4}$$

where Z_u is the normalization term that ensures $\sum_{\mathbf{p}_u} P(\mathbf{p}_u) = 1$, and $\psi(\cdot)$ is a positive function known as *potential*.

The potential $\psi_{ij}(p_{ui}, p_{uj})$ captures the correlation between items i and j

$$\psi_{ij}(p_{ui}, p_{uj}) = \exp\{w_{ij} f_{ij}(p_{ui}, p_{uj})\} \tag{3.5}$$

where $f_{ij}(\cdot)$ is the feature function and w_{ij} is the corresponding weight. The correlation features capture the *LS*, while the weights realize the importance of each correlation feature. With the weights estimated from data, the unknown preference p_{ui} can be predicted as

$$\hat{p}_{ui} = \operatorname*{arg\,max}_{p_{ui}\in[-1,1]} P(p_{ui} \mid \mathbf{p}_u) \tag{3.6}$$

where $P(p_{ui} \mid \mathbf{p}_u)$ measures the confidence of the prediction.

3.3. Ordinal Logistic Regression

The original *PrefNMF* (Desarkar et al., 2012) computes the attractiveness of an item to a user by the product of their latent feature vectors which results a scalar value. Instead of computing these point estimates, we wish to have the distributions over ordinal values. Therefore the *Random Utility Models* (McFadden, 1980) and the *Ordinal Logistic Regression* (McCullagh, 1980) are applied to perform the conversion.

Random Utility Models (McFadden, 1980) assume the existence of a *latent utility* $x_{ui} = \mu_{ui} + \epsilon_{ui}$ that captures how much the user u is interested in the item i, where μ_{ui} captures the interest and ϵ_{ui} is the random noise, and here assumed to follow the logistic distribution (Koren and Sill, 2011).

The *Ordinal Logistic Regression* (McCullagh, 1980) is then used to convert the user-wise preferences p_{ui} into ordinal values, which assumes that the preference p_{ui} is chosen based on the interval to which the latent utility belongs

$$p_{ui} = l \text{ if } x_{ui} \in (\theta_{l-1}, \theta_l] \text{ for } l < L \text{ and } p_{ui} = L \text{ if } x_{ui} > \theta_{L-1} \tag{3.7}$$

where L is the number of ordinal levels and θ_l are the threshold values of interest. The probability of receiving a preference l is therefore

$$Q(p_{ui} = l \mid u, i) = \int_{\theta_{l-1}}^{\theta_l} P(x_{ui} \mid \theta)\, \mathrm{d}\theta = F(\theta_l) - F(\theta_{l-1}) \tag{3.8}$$

where $F(\theta_l)$ is the cumulative logistic distribution evaluated at θ_l with standard deviation s_{ui}

$$F(x_{ui} \leq l \mid \theta_l) = \frac{1}{1 + \exp(-\frac{\theta_{uil} - \mu_{ui}}{s_{ui}})} \tag{3.9}$$

The thresholds θ_l can be parameterized to depend on user or item. This paper employs the user-specific thresholds parameterization described in (Koren and Sill, 2011). Therefore a set of thresholds $\{\theta_{ul}\}_{l=1}^{L}$ is defined for each user u to replace the thresholds θ_{uil} in Eq. 3.9, and is learned from data.

Given the learned ordinal distribution $Q(p_{ui} \mid u, i)$, not only the preferences can be predicted but also the *confidence* for each prediction.

3.4. PrefMRF: Unifying PrefNMF and MRF

The standard *MRF* approach captures the *LS* by modeling item-item correlations under the framework of probabilistic graphical models. However, it employs the log-linear modeling as shown in Eq. 3.5, and therefore does not enable a simple treatment of *PR*. *PrefNMF*, on the other hand, can nicely model the preference relations but is weak in capturing the *LS*. The complementary between these two techniques calls for the unified *PrefMRF* model to take all of the advantages.

Essentially, the proposed *PrefMRF* model promotes the agreement between the *GS* discovered by the *PrefNMF* and the *LS* discovered by the *MRF*. More specifically, the *PrefMRF* model combines the item-item correlations (Eq. 3.5) and the ordinal distributions $Q(p_{ui} \mid u, i)$ over user-wise preferences obtained from Eq. 3.8.

$$P(\mathbf{p}_u) \propto \Psi_u(\mathbf{p}_u) \prod_{p_{ui} \in \mathbf{p}_u} Q(p_{ui} \mid u, i) \tag{3.10}$$

where Ψ_u is the potential function capturing the interaction among items evaluated by user u. The potentials can be further factorized as follows:

$$\Psi_u(\mathbf{p}_u) = \exp\left(\sum_{p_{ui}, p_{uj} \in \mathbf{p}_u} w_{ij} f_{ij}(p_{ui}, p_{uj}) \right) \tag{3.11}$$

where $f_{ij}(\cdot)$ is the correlation feature to be defined shortly in Section 3.4.1, and w_{ij} is the corresponding weight. Put all together, the joint distribution $P(\mathbf{p}_u)$ for each user u can be modeled as

$$P(\mathbf{p}_u) \propto \exp\left(\sum_{p_{ui}, p_{uj} \in \mathbf{p}_u} w_{ij} f_{ij}(p_{ui}, p_{uj}) \right) \prod_{p_{ui} \in \mathbf{p}_u} Q(p_{ui} \mid u, i) \tag{3.12}$$

where there is a graph for each user but the weights are optimized by all users.

3.4.1. Feature Design

A feature is essentially a function f of $n > 1$ arguments that maps the (n-dimensional) input onto the unit interval $f : \mathbb{R}^n \to [0, 1]$, where the input can be preference or auxiliary information such as *content* (Tran et al., 2007).

The item-item correlation is captured by the feature

$$f_{ij}(p_{ui}, p_{uj}) = g(|(p_{ui} - \bar{p}_i) - (p_{uj} - \bar{p}_j)|) \tag{3.13}$$

where $g(\alpha) = 1 - \alpha/L$ does normalization with α acts as the deviation. $\bar{p}_i$ and $\bar{p}_j$ are the average user-wise preference for items i and j, respectively. This correlation feature captures the intuition that correlated items should be ranked similarly by the same user after offsetting the goodness of each item.

Since one correlation feature exists for each possible pair of co-rated items, the number of correlation features can be large, and this makes the estimation slow to converge and less robust. Therefore we only keep the correlation features if strong correlation exists between two items i and j. Specifically, the *strong correlation features* $\mathbf{f}_{\text{strong}}$ are extracted based on the *Pearson Correlation* and a user-specified *minimum correlation threshold*. Note that the correlation is calculated based on the user-wise preferences generated from *PR*.

3.4.2. Parameter Estimation

In general, *MRF* models cannot be determined by standard maximum likelihood approaches, instead, approximation techniques such as *Pseudo-likelihood* (Besag, 1974) and *Contrastive Divergence* (CD) (Hinton, 2002) are often used in practice. The *pseudo-likelihood* leads to exact computation of the loss function and its gradient with respect to parameters, and thus faster. The CD-based methods may, on the other hand, lead to better estimation given enough time. As the experiments involve different settings and large number of features, this study employs the *pseudo-likelihood* technique to perform efficient parameter estimation by maximizing the regularized sum of log local likelihoods

$$log\mathcal{L}(\mathbf{w}) = \sum_{p_{ui}} \log P(p_{ui}|\mathbf{p}_u \backslash p_{ui}) - \frac{1}{2\sigma^2}\mathbf{w}^\top \mathbf{w} \tag{3.14}$$

where $1/2\sigma^2$ controls the regularization, and $\mathbf{w}_u$ is the subset of weights related to user u.

The local likelihood is defined as

$$P(p_{ui}|\mathbf{p}_u \backslash p_{ui}) = \frac{1}{Z_{ui}} \exp\left(\sum_{p_{uj}\in\mathbf{p}_u\backslash p_{ui}} w_{ij} f_{ij}(p_{ui}, p_{uj})\right) Q(p_{ui} \mid u, i) \tag{3.15}$$

where Z_{ui} is the normalization term.

$$Z_{ui} = \sum_{p_{ui}=l_{min}}^{l_{max}} \exp\left(\sum_{p_{uj}\in\mathbf{p}_u\backslash p_{ui}} w_{ij} f_{ij}(p_{ui}, p_{uj})\right) Q(p_{ui} \mid u, i) \tag{3.16}$$

where l_{min} is the first and l_{max} is the last interval.

To optimize the parameters, we use the stochastic gradient ascent procedure that updates the parameters by passing through the set of ratings of each user:

$$\mathbf{w}_u \leftarrow \mathbf{w}_u + \eta \nabla log\mathcal{L}(\mathbf{w}_u) \tag{3.17}$$

where η is the learning rate. More specifically, for each p_{ui} and its neighbor $p_{uj} \in \mathbf{p}_u$, update the weight w_{ij} using the gradient of the regularized log pseudo-likelihood

$$\frac{\partial log\mathcal{L}}{\partial w_{ij}} = f_{ij}(p_{ui}, p_{uj}) - \sum_{p_{ui}=l_{min}}^{l_{max}} P(p_{ui} \mid \mathbf{p}_u \backslash p_{ui}) f_{ij}(p_{ui}, p_{uj}) - \frac{w_{ij}}{\sigma^2} \tag{3.18}$$

3.4.3. Item Recommendation

The ultimate goal of *RecSys* is often to rank the items and recommend the Top-N items to the user. To obtain the item rankings, *PrefMRF* estimates distributions over user-wise preferences which can be converted into point estimate by computing the expectation:

$$\hat{p}_{ui} = \sum_{p_{ui}=l_{min}}^{l_{max}} p_{ui} P(p_{ui} \mid \mathbf{p}_u) \tag{3.19}$$

where l refers to the intervals of user-wise preferences: from least to most preferred.

Given the predicted user-wise preferences, the items can be sorted and ranked accordingly. Finally, Alg. 1 summarizes the learning and prediction procedures for the *PrefMRF*.

Algorithm 1 *PrefMRF* Algorithm

```
Input: PR Π inferred from explicit or implicit feedbacks.
Step 1: Predict user-wise preferences p̂_ui using Eq. 3.1 and Eq. 2.2.
Step 2: Predict distribution for each p̂_ui using Eq. 3.8.
Step 3: Repeat
for each u ∈ U do
  for each p_ui ∈ p_u do
    Compute normalization term Z_ui using Eq. 3.16
    Compute local likelihood using Eq. 3.15
    for each p_uj ∈ p_u, i ≠ j ∧ f_ij ∈ f_strong do
      Compute correlation feature f_ij using Eq. 3.13
      Compute gradient for correlation feature f_ij using Eq. 3.18
      Update w_ij with the gradient using Eq. 3.17
    end for
  end for
end for
Until stopping criteria met
Predictions:
* Predict user-wise preferences using Eq. 3.19.
* Select Top-N items according to estimated user-wise preferences.
```

3.4.4. Computational Complexity

We perform a quick analysis on the computational complexity w.r.t. number of users, items, and ratings. Given n users and m items each has d_u and d_i preferences, respectively. Let us temporarily ignore the user-specified latent factors. Then the complexity of both *PrefNMF* and *PrefMRF* is $O(nd_u^2)$. However, in practice few item co-rated by the same user are strong neighbors of each other due to the correlation threshold defined in Section 3.4.1. As a result, the computation time of *PrefMRF* tends to be $O(nd_uc)$ where c is a factor of correlation threshold.

4. Experiment and Analysis

To study the performance of the proposed *PrefMRF* model, comparisons were done with the following representative algorithms: *a*) *K-Nearest Neighbors* (KNN) (Resnick et al., 1994), which represents the methods exploiting the *LS* from absolute ratings; *b*) *Non-negative Matrix Factorization* (NMF) (Koren et al., 2009), which represents the methods exploiting the *GS* from absolute ratings; *c*) *Preference Relation-based KNN* (PrefKNN) (Brun et al., 2010), which exploits the *LS* from *PR*; *d*) *Preference Relation-based NMF* (PrefNMF) (Desarkar et al., 2012), which exploits the *GS* from *PR*.

4.1. Experimental Settings

4.1.1. Datasets

Ideally, the experiments should be conducted on datasets that contain user preferences in two forms: *PR* and *absolute ratings*. Unfortunately no such a dataset is publicly available at the moment, therefore we choose to compile the rating-based datasets into the form of *PR*. We use the same conversion method as in (Desarkar et al., 2012) by comparing the ratings of each ordered pair of items co-rated by the same user. For example, 1 is assigned to the *PR* π_{uij} if $p_{ui} > p_{uj}$; 0 is assigned if $p_{ui} < p_{uj}$, and 0.5 is assigned if $p_{ui} = p_{uj}$.

Experiments were conducted on two datasets: the *MovieLens*-1M [1] and the *EachMovie* [2] datasets. The *MovieLens*-1M dataset contains more than 1 million ratings by $6,040$ users on $3,900$ movies. The *EachMovie* dataset contains 2.8 million ratings by $72,916$ users on $1,628$ movies. The minimum rating is 1 and we cap the maximum at 5 for both datasets.

For a reliable and fair comparison, each dataset is split into train and test sets, and the following settings are aligned to related work (Weimer et al., 2007). As the sparsity levels differ between the *MovieLens*-1M and the *EachMovie* datasets, different number of ratings are reserved for training and the rest for testing. Specifically, for each user in the *MovieLens*-1M we randomly select $N = 30$, 40, 50, 60 ratings for training, and put the rest for testing. Some users do not have enough ratings thus were excluded from experiments. The *EachMovie* has less items but much more users comparing to *MovieLens*-1M, therefore it is safe to remove some less active users and we set $N = 70$, 80, 90, 100 to investigate the performance on dense dataset.

4.1.2. Evaluation Metrics

Traditional recommender systems aim to optimize *RMSE* or *MAE* which emphasizes on absolute ratings. However, the ultimate goal of recommender systems is usually to obtain the ranking of items (Koren and Sill, 2011), where good performance on *RMSE* or *MAE* may not be translated into good ranking results (Koren and Sill, 2011). Therefore, we employ two evaluation metrics: *Normalized Cumulative Discounted Gain@T* (NDCG@T) (Järvelin and Kekäläinen, 2002) which is popular in academia, and *Mean Average Precision@T* (MAP@T) (Chapelle et al., 2009) which is popular in contests [3]. Among them, the

1. http://grouplens.org/datasets/movielens
2. http://grouplens.org/datasets/eachmovie
3. KDD Cup 2012 and Facebook Recruiting Competition

NDCG@T metric is defined as

$$\text{NDCG}@T = \frac{1}{K(T)} \sum_{t=1}^{T} \frac{2^{r_t} - 1}{\log_2 (t+1)} \tag{4.1}$$

where r_t is the relevance judgment of the item at position t, and $K(T)$ is the normalization constant. The MAP@T metric is defined as

$$\text{MAP}@T = \frac{1}{|\mathcal{U}_{test}|} \sum_{u \in \mathcal{U}_{test}} \sum_{t=1}^{T} \frac{P_u(t)}{min(m_u, t)} \tag{4.2}$$

where m_u is the number relevant items to user u, and $P_u(t)$ is user u's precision at position t. Both metrics are normalized to $[0, 1]$, and a higher value indicates better performance.

These metrics, together with other ranking-based metrics, require a set of relevant items to be defined in the test set such that the predicted rankings can be evaluated against. The relevant items can be defined in different ways. In this paper, we follow the same selection criteria used in the related work (Koren, 2008; Brun et al., 2010) to consider items with the highest ratings as relevant.

4.1.3. Parameter Setting

For a fair comparison, we fix the number of latent factors to 50 for all algorithms, the same as in related work (Cremonesi et al., 2010). The number of neighbors for *KNN* algorithms is set to 50. We vary the minimum correlation threshold to examine the performances with different number of features. Different values of regularization coefficient are also tested.

4.2. Results and Analysis

4.2.1. Comparison on Top-N Recommendation

Implementations of the benchmark algorithms including ours are publicly available in *GitHub* repository. Comparison of these algorithms is conducted by measuring the *NDCG* and the *MAP* metrics on Top-N recommendation tasks. Each experiment is repeated ten times with different random seeds and we report the mean results with standard deviation on *MovieLens*-1M dataset in Table 2 and on *EachMovie* dataset in Table 3. We also report the *NDCG* and *MAP* values by varying the position T in Fig. 2. The following observations can be made based on the results.

Firstly, the *KNN* and the *PrefKNN* methods didn't perform well on *MovieLens*-1M comparing with *Matrix Factorization* based methods. One possible reason is that predictions are made based only on the neighbors, and as a result too much information has been ignored especially when the dataset is large. However, the performance of *KNN*-based methods has improved on the *EachMovie* dataset as we reserved more ratings for training, i.e., better neighbors can be found for prediction.

Secondly, *PrefNMF* outperforms *NMF* on *MovieLens*-1M dataset which is consistent to the results reported in (Desarkar et al., 2012). However, *PreNMF* does not perform well on *EachMovie* where its performance is only slightly better than user-based *KNN*. The reason behind could be the *EachMovie* is much denser than the *MovieLens*-1M dataset,

Table 2: Mean results and standard deviation over ten runs on *MovieLens*-1M dataset.

	Given 30				Given 40			
Algorithm	NDCG@5	NDCG@10	MAP@5	MAP@10	NDCG@5	NDCG@10	MAP@5	MAP@10
UserKNN	0.3969 ± 0.0020	0.4081 ± 0.0029	0.2793 ± 0.0021	0.2744 ± 0.0025	0.4108 ± 0.0040	0.4252 ± 0.0036	0.2936 ± 0.0036	0.2877 ± 0.0034
NMF	0.5232 ± 0.0057	0.5195 ± 0.0040	0.3866 ± 0.0055	0.3549 ± 0.0037	0.5323 ± 0.0050	0.5291 ± 0.0034	0.3976 ± 0.0045	0.3631 ± 0.0035
PrefKNN	0.3910 ± 0.0044	0.4048 ± 0.0038	0.2745 ± 0.0043	0.2720 ± 0.0037	0.4122 ± 0.0024	0.4283 ± 0.0024	0.2944 ± 0.0023	0.2904 ± 0.0023
PrefNMF	0.5729 ± 0.0049	0.5680 ± 0.0041	0.4387 ± 0.0046	0.3992 ± 0.0033	0.5773 ± 0.0037	0.5732 ± 0.0028	0.4437 ± 0.0041	0.4019 ± 0.0032
PrefMRF	$\mathbf{0.5970 \pm 0.0050}$	$\mathbf{0.5864 \pm 0.0039}$	$\mathbf{0.4622 \pm 0.0050}$	$\mathbf{0.4194 \pm 0.0036}$	$\mathbf{0.6125 \pm 0.0029}$	$\mathbf{0.6020 \pm 0.0023}$	$\mathbf{0.4784 \pm 0.0025}$	$\mathbf{0.4316 \pm 0.0020}$

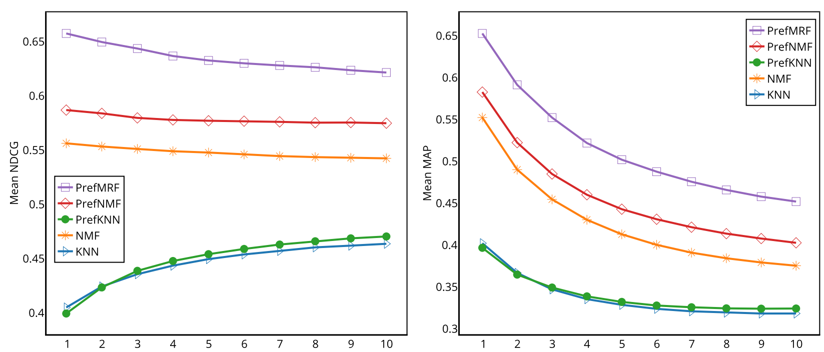
	Given 50				Given 60			
Algorithm	NDCG@5	NDCG@10	MAP@5	MAP@10	NDCG@5	NDCG@10	MAP@5	MAP@10
UserKNN	0.4273 ± 0.0040	0.4424 ± 0.0027	0.3078 ± 0.0038	0.3015 ± 0.0026	0.4480 ± 0.0044	0.4622 ± 0.0035	0.3266 ± 0.0036	0.3163 ± 0.0027
NMF	0.5360 ± 0.0041	0.5326 ± 0.0036	0.4010 ± 0.0040	0.3669 ± 0.0025	0.5462 ± 0.0068	0.5409 ± 0.0063	0.4109 ± 0.0069	0.3734 ± 0.0055
PrefKNN	0.4326 ± 0.0027	0.4483 ± 0.0030	0.3125 ± 0.0024	0.3070 ± 0.0022	0.4526 ± 0.0062	0.4689 ± 0.0039	0.3301 ± 0.0051	0.3223 ± 0.0033
PrefNMF	0.5761 ± 0.0067	0.5745 ± 0.0035	0.4424 ± 0.0064	0.4019 ± 0.0033	0.5756 ± 0.0062	0.5733 ± 0.0048	0.4409 ± 0.0059	0.4007 ± 0.0037
PrefMRF	$\mathbf{0.6148 \pm 0.0053}$	$\mathbf{0.6082 \pm 0.0032}$	$\mathbf{0.4806 \pm 0.0053}$	$\mathbf{0.4360 \pm 0.0027}$	$\mathbf{0.6311 \pm 0.0037}$	$\mathbf{0.6200 \pm 0.0037}$	$\mathbf{0.5012 \pm 0.0035}$	$\mathbf{0.4500 \pm 0.0026}$

Table 3: Mean results and standard deviation over ten runs on *EachMovie* dataset.

	Given 70				Given 80			
Algorithm	NDCG@5	NDCG@10	MAP@5	MAP@10	NDCG@5	NDCG@10	MAP@5	MAP@10
UserKNN	0.7088 ± 0.0020	0.7115 ± 0.0015	0.6012 ± 0.0027	0.5767 ± 0.0017	0.7146 ± 0.0018	0.7168 ± 0.0017	0.6070 ± 0.0021	0.5825 ± 0.0019
NMF	0.7581 ± 0.0022	0.7577 ± 0.0017	0.6524 ± 0.0026	0.6225 ± 0.0020	0.7636 ± 0.0021	0.7638 ± 0.0018	0.6583 ± 0.0025	0.6286 ± 0.0018
PrefKNN	0.7260 ± 0.0022	0.7307 ± 0.0018	0.6197 ± 0.0020	0.5990 ± 0.0016	0.7337 ± 0.0028	0.7377 ± 0.0018	0.6271 ± 0.0029	0.6057 ± 0.0021
PrefNMF	0.7408 ± 0.0033	0.7348 ± 0.0039	0.6330 ± 0.0035	0.5800 ± 0.0038	0.7422 ± 0.0036	0.7319 ± 0.0040	0.6329 ± 0.0039	0.5774 ± 0.0033
PrefMRF	$\mathbf{0.8217 \pm 0.0032}$	$\mathbf{0.8095 \pm 0.0029}$	$\mathbf{0.7312 \pm 0.0039}$	$\mathbf{0.6824 \pm 0.0034}$	$\mathbf{0.8264 \pm 0.0036}$	$\mathbf{0.8132 \pm 0.0030}$	$\mathbf{0.7353 \pm 0.0038}$	$\mathbf{0.6861 \pm 0.0032}$

	Given 90				Given 100			
Algorithm	NDCG@5	NDCG@10	MAP@5	MAP@10	NDCG@5	NDCG@10	MAP@5	MAP@10
UserKNN	0.7191 ± 0.0022	0.7279 ± 0.0028	0.6120 ± 0.0021	0.5933 ± 0.0013	0.7279 ± 0.0028	0.7277 ± 0.0015	0.6238 ± 0.0032	0.5973 ± 0.0021
NMF	0.7712 ± 0.0039	0.7692 ± 0.0033	0.6663 ± 0.0043	0.6431 ± 0.0034	0.7741 ± 0.0030	0.7717 ± 0.0028	0.6719 ± 0.0034	0.6411 ± 0.0030
PrefKNN	0.7418 ± 0.0028	0.7421 ± 0.0015	0.6357 ± 0.0030	0.6192 ± 0.0020	0.7505 ± 0.0019	0.7511 ± 0.0012	0.6478 ± 0.0020	0.6231 ± 0.0014
PrefNMF	0.7456 ± 0.0031	0.7358 ± 0.0038	0.6357 ± 0.0040	0.5819 ± 0.0036	0.7391 ± 0.0033	0.7298 ± 0.0034	0.6318 ± 0.0039	0.5761 ± 0.0039
PrefMRF	$\mathbf{0.8294 \pm 0.0035}$	$\mathbf{0.8149 \pm 0.0032}$	$\mathbf{0.7374 \pm 0.0037}$	$\mathbf{0.6946 \pm 0.0032}$	$\mathbf{0.8308 \pm 0.0031}$	$\mathbf{0.8167 \pm 0.0030}$	$\mathbf{0.7436 \pm 0.0038}$	$\mathbf{0.6961 \pm 0.0036}$

which makes the number of *PR* huge and difficult to tune optimal parameters. Besides, we observe that *PrefNMF* in general only achieves a slight improvement with more training data and even drops a bit with *Given* 60. Similarly for the *EachMovie* dataset. With these observations, it appears that for a given number of users, the *PrefNMF* can be trained reasonably well with fewer data.



(*a*) Mean NDCG@T (*b*) Mean MAP@T

Figure 2: Performance for different position T (MovieLens-1M, *Given 60*).

Finally, the proposed *PrefMRF* has made further improvement on both datasets upon the *PrefNMF* through capturing both *LS* and *GS*. From Fig. 2 we can see that the algorithms stabilized around position 10 and *PrefMRF* consistently delivers better performance than others. It should be noted that the performance of *PrefMRF* relies on its underlying model that captures the *GS*. In other words, the performance may vary when the *PrefNMF* is replaced with other alternative methods such as (Liu et al., 2009).

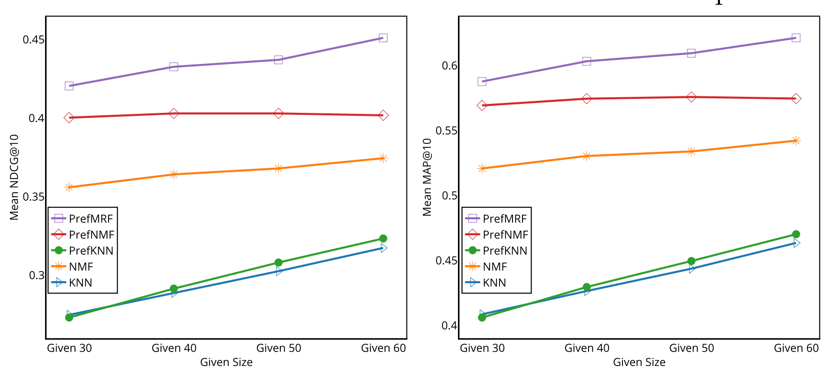
Table 4: Paired t-test for *PrefMRF* and *PrefNMF*.

Settings			t-test statistics		
Dataset	Sparsity	Metric	df	t	p-value
MovieLens	*Given* 60	*NDCG*@10	9	15.6998	< 0.00001
MovieLens	*Given* 60	*MAP*@10	9	23.1577	< 0.00001
EachMovie	*Given* 100	*NDCG*@10	9	70.4189	< 0.00001
EachMovie	*Given* 100	*MAP*@10	9	71.7146	< 0.00001

To confirm the improvements, a paired t-test (two-tailed) with a significance level of 95% has been applied.Results shown in Table 4 confirm that the performance of methods with and without capturing the *LS* is statistically significant.

4.2.2. Performance on Various Data Sparsity Levels

To thoroughly examine the performance of these algorithms, we compare their performances under different settings of training set sizes: *Given* 30, *Given* 50, and *Given* 70. Results are plotted in Fig. 3. It can be observed that in general more training data result in better performance. However, *PrefNMF* does not gain much benefit from more data and even perform slightly worse in *Given* 60. The *PrefMRF* on the other hand consistently gains performance from more data as the *LS* information can be better captured.



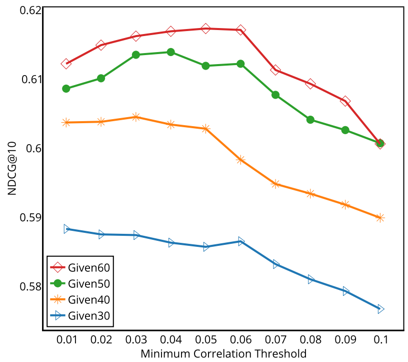
(a) NDCG@10 by Given Size (b) MAP@10 by Given Size

Figure 3: Impact of Sparsity Levels (MovieLens-1M).

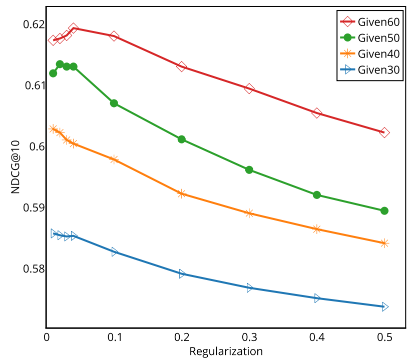
4.2.3. Impact of Minimum Correlation Threshold

As described in Section 3.4.1, a *minimum correlation threshold* is required to control the number of features in the *PrefMRF* model. By default, each pair of co-rated items has a feature which results in a large number of features. However, many of these features are useless if the item-item correlation are weak. To make the model more robust and with faster convergence, a *minimum correlation threshold* is applied to remove weak features. Specifically, the feature is removed if two items has a correlation measured by *Pearson* correlation less than the threshold. Results are plotted in Fig. 4(a).

It can be observed that a smaller correlation threshold delivers better performance, however, the number of features will also increase. To balance the performance and computation time, it is wise to select a moderate level of threshold depending on the dataset.



(a) NDCG@10 by Threshold



(b) NDCG@10 by Reg.

Figure 4: Impact of Parameters (MovieLens-1M)

4.2.4. Impact of Regularization Coefficient

As the number of features in *PrefMRF* can be large, the model might be prone to over-fitting. Therefore, we investigate the impact of regularization settings as plotted in Fig. 4(b).

We observe that the performance is better when a small regularization penalty applies. In other words, the *PrefMRF* can generalize reasonable well without too much regularization. This can be explained as the weights of item-item correlations are not user-specific but shared by all users, thus they cannot over-fit every user perfectly.

5. Conclusions and Future Works

In this paper we presented the *PrefMRF* model, which is capable of modeling both *LS* and *GS*. Experiment results on public datasets demonstrate that types of interactions have been properly captured, resulting improved Top-N recommendation performance. For future work, we would like to see how the proposed model performs on real *PR*-based dataset generated from from implicit feedbacks such as *activity logs*.

Acknowledgement

This work was partially supported by the National Science Foundation of China (61273301) and the Collaborative Innovation Center of Novel Software Technology and Industrialization.

References

M. Balabanović and Y. Shoham. Fab: content-based, collaborative recommendation. *Commun. ACM*, 40(3):66–72, 1997.

J. Besag. Spatial interaction and the statistical analysis of lattice systems. *Journal of the Royal Statistical Society, Series B*, 36(2):192–236, 1974.

A. Brun, A. Hamad, O. Buffet, and A. Boyer. Towards preference relations in recommender systems. In *Preference Learning (PL 2010) ECML/PKDD*, 2010.

O. Chapelle, D. Metlzer, Y. Zhang, and P. Grinspan. Expected reciprocal rank for graded relevance. In *CIKM'09*, pages 621–630. ACM, 2009.

P. Cremonesi, Y. Koren, and R. Turrin. Performance of recommender algorithms on top-n recommendation tasks. In *RecSys'10*, pages 39–46. ACM, 2010.

A. Defazio and T. Caetano. A graphical model formulation of collaborative filtering neighbourhood methods with fast maximum entropy training. In *ICML'12*, 2012.

M. S. Desarkar, R. Saxena, and S. Sarkar. Preference relation based matrix factorization for recommender systems. In *UMAP'12*, pages 63–75. Springer, 2012.

J. Fürnkranz and E. Hüllermeier. *Preference learning*. Springer, 2010.

G. Hinton. Training products of experts by minimizing contrastive divergence. *Neural Computation*, 14(8):1771–1800, 2002.

K. Järvelin and J. Kekäläinen. Cumulated gain-based evaluation of ir techniques. *ACM TOIS*, 20(4):422–446, 2002.

Y. Koren. Factorization meets the neighborhood: a multifaceted collaborative filtering model. In *KDD'08*, pages 426–434. ACM, 2008.

Y. Koren and J. Sill. Ordrec: an ordinal model for predicting personalized item rating distributions. In *RecSys'11*, pages 117–124. ACM, 2011.

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

N. N. Liu, M. Zhao, and Q. Yang. Probabilistic latent preference analysis for collaborative filtering. In *CIKM'09*, pages 759–766. ACM, 2009.

S. Liu, T. Tran, G. Li, and Y. Jiang. Ordinal random fields for recommender systems. In *ACML'14*, pages 283–298. JMLR: workshop and conference proceedings, 2014.

P. McCullagh. Regression models for ordinal data. *Journal of the Royal Statistical Society, Series B*, 42(2):109–142, 1980.

D. McFadden. Econometric models for probabilistic choice among products. *Journal of Business*, 53(3):S13–S29, 1980.

M. Mohri, A. Rostamizadeh, and A. Talwalkar. *Foundations of machine learning*. MIT press, 2012.

S. Rendle, C. Freudenthaler, Z. Gantner, and L. Schmidt-Thieme. Bpr: Bayesian personalized ranking from implicit feedback. In *UAI'09*, pages 452–461. AUAI Press, 2009.

P. Resnick, N. Iacovou, M. Suchak, P. Bergstrom, and J. Riedl. An open architecture for collaborative filtering of netnews. In *CSCW'94*, pages 175–186. ACM, 1994.

B. Sarwar, G. Karypis, J. Konstan, and J. Riedl. Item-based collaborative filtering recommendation algorithms. In *WWW'10*, pages 285–295. ACM, 2001.

Y. Shi, M. Larson, and A. Hanjalic. List-wise learning to rank with matrix factorization for collaborative filtering. In *RecSys'10*, pages 269–272. ACM, 2010.

T. Tran, D. Q. Phung, and S. Venkatesh. Preference networks: Probabilistic models for recommendation systems. In *AusDM'07*, pages 195–202. ACS, 2007.

T. Tran, D. Q. Phung, and S. Venkatesh. Ordinal boltzmann machines for collaborative filtering. In *UAI'09*, pages 548–556. AUAI Press, 2009.

M. Weimer, A. Karatzoglou, Q. V. Le, and A. Smola. Maximum margin matrix factorization for collaborative ranking. In *NIPS'07*, pages 1593–1600, 2007.