Proceedings of the Twenty-Ninth AAAI Conference on Artificial Intelligence

Collaborative Topic Ranking: Leveraging Item Meta-Data for Sparsity Reduction

Weilong Yao* [1,3], **Jing He**[2], **Hua Wang**[2], **Yanchun Zhang**[2,3], **Jie Cao**[4]
[1]University of Chinese Academy of Sciences, Beijing, China
[2]Centre for Applied Informatics, College of Engineering & Science, Victoria University, Australia
[3]Fudan University, Shanghai, China
[4] Nanjing University of Finance and Economics, Nanjing, China
yaoweilong12@mails.ucas.ac.cn
{Jing.He,Hua.Wang,Yanchun.Zhang}@vu.edu.au, tcaojie690929@163.com

Abstract

Pair-wise ranking methods have been widely used in recommender systems to deal with implicit feedback. They attempt to discriminate between a handful of observed items and the large set of unobserved items. In these approaches, however, user preferences and item characteristics cannot be estimated reliably due to overfitting given highly sparse data. To alleviate this problem, in this paper, we propose a novel hierarchical Bayesian framework which incorporates "bag-of-words" type meta-data on items into pair-wise ranking models for one-class collaborative filtering. The main idea of our method lies in extending the pair-wise ranking with a probabilistic topic modeling. Instead of regularizing item factors through a zero-mean Gaussian prior, our method introduces item-specific topic proportions as priors for item factors. As a by-product, interpretable latent factors for users and items may help explain recommendations in some applications. We conduct an experimental study on a real and publicly available dataset, and the results show that our algorithm is effective in providing accurate recommendation and interpreting user factors and item factors.

Introduction

Recommender systems, as a subclass of information filtering systems, attempt to model user preferences by analysing user feedback. Explicit user feedback, e.g., numeric rating has been well studied in the literature (Koren 2008; Rendle 2010). In most applications, however, the collected data of user behaviors, e.g., assigning tags (Pan and Chen 2013b), purchasing products (Rendle et al. 2009) and watching videos (Rendle and Freudenthaler 2014), are usually implicit. Collaborative Filtering (CF) with implicit feedback is referred as One-Class Collaborative Filtering (OCCF) (Pan et al. 2008).

Existing approaches to OCCF can be classified into two categories: (1) point-wise preferences methods and (2) pair-wise preferences methods. The main idea of these methods is to learn user factors and item factors for modeling user preferences and item characteristics, respectively. Point-wise methods assume that observed feedback denotes "like" and unobserved feedback denotes "dislike", and propose to train a recommendation model by approximating the absolute rating scores (Hu, Koren, and Volinsky 2008; Wang and Blei 2011), while pair-wise methods assume that a user prefers an observed item to an unobserved item, and convert the problem to a pair-wise ranking problem (Rendle et al. 2009; Takács and Tikk 2012). Empirically, the latter assumption of pair-wise preferences over two items yields better recommendation accuracy than the point-wise assumption (Lee et al. 2014).

However, the data sparsity problem has become a major bottleneck for pair-wise ranking methods. In real world scenarios, a user typically interacts with a limited number of items out of possibly thousands or millions of items, which leads to extremely sparse user implicit feedback. These latent factors based methods are too flexible and are very likely to suffer from overfitting given sparse data, even for small dimensionality of latent factors. That is, the amount of information for learning user factors and item factors is far from enough.

In many practical situations, though, we have more information than the simple implicit feedback. Typically the item in a recommender system is equipped with "bag-of-words" type meta-data. Please note that "word" in this context is a general term used to denote elements like *phrases*, *tags* and *movie actors* in the applications of article recommendation, tag-based picture recommendation, and movie recommendation, respectively. The meta-data is highly informative in terms of identifying item characteristics to differentiate the item from others. This is especially crucial for addressing the OCCF problem, as we are mostly interested in distinguishing those items which are of potential interest to users from others. For example, given a user who likes "machine learning" articles and two articles j_1 and j_2, we would recommend him/her the article j_1 rather than j_2 if we have the prior knowledge that j_1 is about "machine learning" while j_2 is not. Thus, it is natural to consider item meta-data for alleviating data sparsity. Nevertheless, existing pair-wise ranking approaches fail to provide a natural way to consider such

*This work was done when the first author was visiting Victoria University, Australia.

meta-data available as free-form text.

In this paper, we propose a novel hierarchical Bayesian framework, called Collaborative Topic Ranking (CTRank), which incorporates meta-data on items and implicit feedback simultaneously for collaborative filtering with highly sparse implicit feedback. By extending pair-wise ranking models with probabilistic topic models, CTRank can utilize item meta-data to tackle the data sparsity problem and further to enhance recommendation accuracy. In CTRank, topic models serve to identify topic proportions from item meta-data, while pair-wise ranking models are to fit a ranked list of items.

We also note that, by discovering the latent topics behind both users and items, CTRank can reveal how an item relates to a user's preferences. The topic representation provides interpretability and may help in explaining recommendations to users. Specifically, the entities of user factors reveal his/her preferences towards the specific topics, while the entities of item factors represent its affinity to the topics. With topics serving as a bridge between users and items, there could be many ways to construct effective explanations for recommendations, e.g., word clouds (Vig, Sen, and Riedl 2009).

To summarize, the contributions of this paper are as follows: (1) We propose a hierarchical Bayesian framework, CTRank, which leverages item meta-data to remedy the data sparsity for OCCF problem by bridging the gap between pair-wise ranking and probabilistic topic modeling. Further, in the proposed framework, two alternative models are developed based on different assumptions. (2) We develop an efficient EM-style algorithm to learn the parameters in CTRank. (3) We conduct an experimental study on a large and real-world dataset. The results empirically show that CTRank outperforms other recommendation methods, in terms of prediction accuracy. Our method is also able to interpret user factors and item factors.

Preliminaries

We start with formulating the problem discussed in this paper, and then review some related works on point-/pair-wise methods for OCCF, Latent Dirichlet Allocation (LDA) (Blei, Ng, and Jordan 2003) and sparsity reduction.

Problem Definition Let $\mathcal{U}$ and $\mathcal{I}$ denote sets of users and items, respectively. The set of observed implicit feedback $\mathcal{R}^+ = \{(i,j) : r_{ij} = 1, i \in \mathcal{U}, j \in \mathcal{I}\}$ from n users and m items is available, where r_{ij} denotes the preferences from user i for item j. The item meta-data $\mathbf{w}_j$ is in the form of "bag-of-words" representation. Needless to say, this is pervasive in web applications.

Let $u_i, v_j \in \mathbb{R}^K$ be user i's factor vector and item j's factor vector respectively, where K is the dimensionality of latent space. We also have the corresponding representation in matrix form, i.e.,$\mathbf{U} = u_{1:n}$ and $\mathbf{V} = v_{1:m}$. The goal is to learn user factors $\mathbf{U}$ and item factors $\mathbf{V}$ from implicit feedback and to generate a personalized ranked list of items for each user i.

Point-wise Ranking for OCCF Point-wise approaches (Hu, Koren, and Volinsky 2008; Pan et al. 2008; Wang and Blei 2011) assume unobserved feedback as a rating 0 and learn latent factors by solving the following regression problem:

$$r_{ij} = 1, r_{ik} = 0, j \in \mathcal{I}_i^+, k \in \mathcal{I} \backslash \mathcal{I}_i^+, \quad (1)$$

where $\mathcal{I}_i^+ = \{j : (i,j) \in \mathcal{R}^+\}$.

Pair-wise Ranking for OCCF Pair-wise ranking approaches capture the pair-wise preferences over two items. Specifically, it is assumed that item j is preferred over item k for a given user i, if and only if j but not k is observed:

$$r_{ij} \succ r_{ik}, j \in \mathcal{I}_i^+, k \in \mathcal{I} \backslash \mathcal{I}_i^+ \quad (2)$$

where $\succ$ denotes a total order regarding user preferences. Our model falls into this category.

The foremost of pair-wise ranking models is Bayesian Personalized Ranking (BPR) (Rendle et al. 2009) which maximizes the likelihood of pair-wise preferences over observed items and unobserved items. More works that extend BPR are detailed in (Krohn-Grimberghe et al. 2012; Pan and Chen 2013a; 2013b). Given sparse data, such latent factor models are too flexible and would overfit if they are not regularized appropriately. Also, all of the above pair-wise methods ignore additional item meta-data while generating recommendations.

Based on pair-wise learning to rank techniques (Liu 2009), several approaches have been proposed to consider additional textual information in some applications (Chen et al. 2012; Qiang, Liang, and Yang 2013). However, directly adopting these ranking approaches is not suitable for OCCF due to that they only learn a single ranking function, i.e., they are non-personalized. The other key difference between those works and ours is how to use textual information. Specifically, previous models utilize textual information to manually craft application-oriented features while we focus on leveraging meta-data, e.g., textual information, for the purpose of automatic feature learning.

LDA for CF In this work, we use LDA (Blei, Ng, and Jordan 2003) to discover topics from meta-data available as free form text. For LDA and Matrix Factorization (MF) techniques, both of which are methods to reduce original data into latent space, it is natural to combine them together. Typical works include (Agarwal and Chen 2010; Shan and Banerjee 2010; Wang and Blei 2011; McAuley and Leskovec 2013). With point-wise preference assumption, these methods focus on squared error minimization and are suitable for rating prediction. However, it is the relevancy of the top items on the list that is crucial to the success of recommender systems, which just has been ignored by point-wise approaches. To overcome this limitation, we propose to directly approximate the order of items. In the experimental study, we use CTR (Wang and Blei 2011) as our major comparison method.

Sparsity Reduction for CF Recently, several research works have been conducted on sparsity reduction. In (Li, Yang, and Xue 2009; Pan et al. 2010), transfer learning techniques are used to transfer knowledge from auxiliary domain to the target domain for sparsity reduction while in practice an auxiliary domain is difficult to identify. Our work

Table 1: Some notations used in this paper.

Notation	Description
K	the number of topics
α	Dirichlet prior
I_K	K-dimensional identity matrix
w_{jt}	the t-th word in item j
$\mathbf{w}_j$	item j's word vector (meta-data)
$\mathbf{W} = \mathbf{w}_{1:m}$	item-word matrix
z_{jt}	the topic assigned to word w_{jt}
θ_j	item j's topic proportions
β_k	topic k's distribution over words
δ_{ijk}	response variable
$\boldsymbol{\delta} = (\delta_{ijk})_{i=1,j=1,k=1}^{n\times m\times m}$	response variable matrix
$\mathcal{P}_i = \{(j,k) : r_{ij} \succ r_{ik}\}$	user i's pair-wise preferences
T_j	number of words in item j
λ_u, λ_v	precision parameter
c_{ijk}	confidence parameter

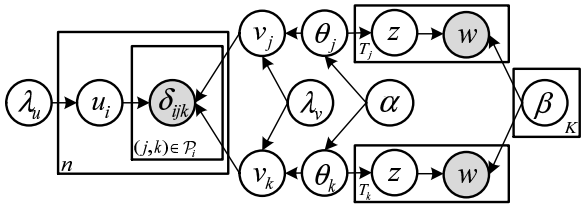


Figure 1: The CTRank model in plate notation.

is also related to hybrid recommendation approaches which combines content-based models with CF models in different ways (Kim et al. 2006; Gunawardana and Meek 2009; Tang and Zhou 2013). However, these methods are devised for explicit feedback while we focus on implicit feedback.

Collaborative Topic Ranking

In this section, we present the proposed model, CTRank, which integrates pair-wise ranking models with LDA into a unified generative model for collaborative filtering with highly sparse implicit feedback.

Model Description

Overview Fig. 1 shows the graphical representation of CTRank, with the notations described in Table 1. The basic idea here is to use two correlated generative processes to model item meta-data and pair-wise preferences together. The first process is to model words associated with items and to specify the relationship between item factors and item topic proportions. Consequently, meta-data is introduced into our generative model. Specifically, we follow the same process as in LDA to generate words (meta-data), and we regularize item factors v_j through the item's topic proportions θ_j. This is a key difference between CTRank and existing pair-wise models that attach latent factors with zero-mean Gaussian priors to each item. The second process models the interaction among user i, observed item j and unobserved item k. In particular, δ_{ijk} is observed response variable which denotes the response from user i for item j over k. With different assumptions on δ_{ijk}, we develop two alternative models for capturing the relative order of pair-wise items.

Bernoulli Assumption First, a natural choice is to directly model the order of the pair-wise items. Specifically, we assume δ_{ijk} follows Bernoulli distribution, then the generative process of CTRank is as follows:

1. For each item j,
   (a) Draw topic proportions $\theta_j \sim \text{Dirichlet}(\alpha)$.
   (b) Draw item factor vector $v_j \sim \mathcal{N}(\theta_j, \lambda_v^{-1} I_K)$.
   (c) For each word w_{jt} in $\mathbf{w}_j$,
      i. Draw a topic $z_{jt} \sim \text{Mult}(\theta_j)$.
      ii. Draw word $w_{jt} \sim \text{Mult}(\beta_{z_{jt}})$.
2. For each user i,
   (a) Draw user factor vector $u_i \sim \mathcal{N}(0, \lambda_u^{-1} I_K)$.
   (b) For each pair-wise preferences $(j,k) \in \mathcal{P}_i$, draw the response variable $\delta_{ijk} = 1 \sim \text{Bernoulli}(\rho_{ijk}^{c_{ijk}})$,

where ρ_{ijk} is the individual probability that user i really prefers item j over k, and is given by:

$$\begin{aligned}\rho_{ijk} &= P(r_{ij} \succ r_{ik} | u_i, v_j, v_k, ...) \\ &= P(\delta_{ijk} = 1 | u_i, v_j, v_k, ...) \\ &= (1 + exp^{-(u_i^{\mathrm{T}} v_j - u_i^{\mathrm{T}} v_k)})^{-1}.\end{aligned} \tag{3}$$

c_{ijk} is the confidence parameter for user i's preferences for item j over k. For simplicy, we set $c_{ijk} = 1$ in our experiments. For $(j,k) \in \mathcal{P}_i$, j is an observed item and k is an unobserved item which is sampled from user i's unobserved items using bootstrap sampling (Rendle et al. 2009).

Based on the generative process above, we have the following conditional probability: $P(\boldsymbol{\delta}|\mathbf{U},\mathbf{V}) = \prod_{i=1}^{n}\prod_{(j,k)\in\mathcal{P}_i} P(\delta_{ijk} = 1|\mathbf{U},\mathbf{V})^{c_{ijk}}$. User factor matrix and item factor matrix are given by $P(\mathbf{U}|\lambda_u) = \prod_{i=1}^{n}\mathcal{N}(u_i|0,\lambda_u^{-1}I_k)$ and $P(\mathbf{V}|\theta,\lambda_v) = \prod_{j=1}^{m}\mathcal{N}(v_i|\theta_j,\lambda_v^{-1}I_k)$, respectively. As in LDA (Blei, Ng, and Jordan 2003), the likelihood of the meta-data under the topic model is a product of likelihood of each $\mathbf{w}_j$, $P(\mathbf{W},\theta|\alpha,\beta) = \prod_{j=1}^{m}\prod_{t=1}^{T_j}\sum_{k=1}^{K}\theta_{jk}\beta_{k,w_{jt}}$. Here we fix the hyperparameter $\alpha = 1$ to keep the computation simple.

Parameters can be learned by using the maximum a posteriori probability (MAP) estimator. Through Bayesian inference, we have the following posteriori probability of $\mathbf{U}$, $\mathbf{V}$ and θ given the model parameters:

$$\begin{aligned}&P(\mathbf{U},\mathbf{V},\theta|\boldsymbol{\delta},\mathbf{W},\beta,\alpha,\lambda_u,\lambda_v) \propto \\ &P(\mathbf{U}|\lambda_u)P(\mathbf{V}|\theta,\lambda_v)P(\boldsymbol{\delta}|\mathbf{U},\mathbf{V})P(\mathbf{W},\theta|\alpha,\beta).\end{aligned} \tag{4}$$

That is, in our model, it is equivalent to maximize the following log-likelihood:

$$\begin{aligned}\mathcal{L}_1 = &-\sum_{ijk} c_{ijk}\log\left(1 + exp^{-(u_i^{\mathrm{T}} v_j - u_i^{\mathrm{T}} v_k)}\right) - \frac{\lambda_u}{2}\sum_i u_i^{\mathrm{T}} u_i \\ &- \frac{\lambda_v}{2}\sum_j (v_j - \theta_j)^{\mathrm{T}}(v_j - \theta_j) + \sum_{jt}\log\left(\sum_k \theta_{jk}\beta_{k,w_{jt}}\right) \\ &+ \mathcal{C},\end{aligned} \tag{5}$$

where $\mathcal{C}$ is a constant and can be omitted.

Gaussian Assumption Unlike in the first model, we use δ_{ijk} to model the preference difference between a pair of items, i.e., $\delta_{ijk} = r_{ij} - r_{ik}$: when $\delta_{ijk} > 0$, item j ranks higher than k, otherwise lower. Specifically, we draw the response variable as follows:

$$\delta_{ijk} \sim \mathcal{N}(u_i^{\mathrm{T}} v_j - u_i^{\mathrm{T}} v_k, c_{ijk}^{-1}). \tag{6}$$

The complete log-likelihood is denoted as $\mathcal{L}_2$:

$$\begin{aligned}\mathcal{L}_2 = &- \sum_{ijk} \frac{c_{ijk}}{2}(\delta_{ijk} - (u_i^{\mathrm{T}} v_j - u_i^{\mathrm{T}} v_k))^2 - \frac{\lambda_u}{2}\sum_i u_i^{\mathrm{T}} u_i \\ &- \frac{\lambda_v}{2}\sum_j (v_j - \theta_j)^{\mathrm{T}}(v_j - \theta_j) + \sum_{jt} \log\,(\sum_k \theta_{jk}\beta_{k,w_{jt}})\end{aligned} \tag{7}$$

Discussion By defining $\Delta_{ijk} = u_i^{\mathrm{T}} v_j - u_i^{\mathrm{T}} v_k$ as predicted preference difference, we derive a general framework by rewriting $\mathcal{L}_1$ and $\mathcal{L}_2$ as follows:

$$\begin{aligned}\mathcal{L} = &- \sum_{ijk} c_{ijk}\ell(\delta_{ijk}, \Delta_{ijk}) - \frac{\lambda_u}{2}\sum_i u_i^{\mathrm{T}} u_i \\ &- \frac{\lambda_v}{2}\sum_j (v_j - \theta_j)^{\mathrm{T}}(v_j - \theta_j) + \sum_{jt} \log\,(\sum_k \theta_{jk}\beta_{k,w_{jt}}),\end{aligned} \tag{8}$$

where $\ell(\delta_{ijk}, \Delta_{ijk})$ is pair-wise loss function, such as log-loss in $\mathcal{L}_1$, and squared loss in $\mathcal{L}_2$. With this framework, more pair-wise loss functions can be adapted for collaborative topic ranking, such as hinge loss (Joachims 2002) and exponential loss (Freund et al. 2003). From this point of view, we refer the first model as CTRank-log and the second model CTRank-squared.

As Eq. 8 shows, the framework CTRank mainly consists of two key components: a ranking modeling component and a meta-data modeling component, between which a topic regularization term $\sum_j (v_j - \theta_j)^{\mathrm{T}}(v_j - \theta_j)$ serves as a bridge. In CTRank, topic proportions guide the learning of latent factors for ranking while the latent factors further improve the meta-data modeling. As with previous pair-wise ranking approaches, the ranking component is too flexible and needs to be regularized appropriately. In our model, topic proportions θ_j are used as priors to specific constraints on item factors v_j, reducing the effective degrees of freedom and yielding good performance. The parameter λ_v controls the extent of topic regularization, and thus plays a vital role in our model. In the extreme case, if we use a very small value of λ_v, we almost lose the effect from topic information, then CTRank reduces to a general pair-wise ranking approach. On the other side, if we employ a very large value of λ_v, topic proportions θ_j will dominate the learning process, and then CTRank benefits little from the ranking component.

Taking a deep insight into Eq. 8, we find that maximizing $-\sum_j (v_j - \theta_j)^{\mathrm{T}}(v_j - \theta_j)$ will make v_j close to its topic proportions parameter θ_j. If items j and k are similar in terms of topic distributions, we are actually minimizing the distance between item factors v_j and v_k. That is, knowledge can be transferred from items with sufficient feedback to items with little feedback. Consequently, we can still learn an accurate representation for items even though the data is extremely sparse.

Parameter Learning

Since the two proposed models can be optimized in a similar way, we only demonstrate how parameters of CTRank-squared are learned in this subsection. We optimize the objective function $\mathcal{L}_2$ using coordinate ascent which alternatively optimizes latent factor variables u_i, v_j and topic proportions θ_j.

We first fix θ_j and update u_i, v_j and v_k using the following stochastic Newton-Raphson rules:

$$u_i = u_i - \eta \frac{\lambda_u u_i - c_{ijk}\mathcal{E}_{ijk}(v_j - v_k)}{\lambda_u + c_{ijk}(v_j - v_k)^{\mathrm{T}}(v_j - v_k)}, \tag{9}$$

$$v_j = v_j - \eta \frac{\lambda_v (v_j - \theta_j) - c_{ijk}\mathcal{E}_{ijk} u_i}{\lambda_v + c_{ijk} u_i^{\mathrm{T}} u_i}, \tag{10}$$

$$v_k = v_k - \eta \frac{\lambda_v (v_k - \theta_k) + c_{ijk}\mathcal{E}_{ijk} u_i}{\lambda_v + c_{ijk} u_i^{\mathrm{T}} u_i}, \tag{11}$$

where η is learning rate, and $\mathcal{E}_{ijk} = \delta_{ijk} - (u_i^{\mathrm{T}} v_j - u_i^{\mathrm{T}} v_k)$.

Given updated $\mathbf{U}$ and $\mathbf{V}$, we now optimize topic proportions θ_j. We define $q(z_{jt} = k) = \phi_{jtk}$, where $\phi_j = (\phi_{jtk})_{t=1,k=1}^{T\times K}$. We then separate the terms that contain θ_j and apply Jensen's inequality to obtain a tight lower bound of $\mathcal{L}_2(\theta_j)$:

$$\begin{aligned}\mathcal{L}_2(\theta_j) \geq &\sum_t \sum_k \phi_{jit}(\log \theta_{jk}\beta_{k,w_{jt}} - \log \phi_{jtk}) \\ &- \frac{\lambda_v}{2}(v_j - \theta_j)^{\mathrm{T}}(v_j - \theta_j) = \mathcal{L}_2(\theta_j, \phi_j).\end{aligned} \tag{12}$$

The optimal ϕ_{jtk} satisfies $\phi_{jtk} \propto \theta_{jk}\beta_{k,w_{jt}}$. Thus, projection gradient (Bertsekas 1999) can be applied to update θ_j.

For the parameter β, we follow the same M-step update as in LDA (Blei, Ng, and Jordan 2003), i.e.,

$$\beta_{kt'} \propto \sum_j \sum_t \phi_{jtk} \mathbb{1}(w_{jt} = t'), \tag{13}$$

where $\mathbb{1}(x)$ is an indicator function that is equal to 1, if x is true, otherwise 0.

In practice, we iteratively update the parameters until the likelihood does not increase (by 0.01%) or the maximum iteration limit, say 100, is reached.

Top-N Recommendation

After we learn the optimal parameters $\mathbf{U}$, $\mathbf{V}$, and θ, we predict r_{ij} from its expectation, based on which a ranked list of N items is generated, as follows:

$$\mathbb{E}(r_{ij} | u_i, v_j, \theta, ...) = u_i^{\mathrm{T}} v_j. \tag{14}$$

Complexity Analysis

The complexity for updating $\mathbf{U}$ and $\mathbf{V}$ is in the order of $O(n\bar{n}K)$, where $\bar{n}$ is the average number of items a user likes. Optimizing θ requires time $O(m(\bar{T}K + \tilde{i}K))$, where $\bar{T}$ is the average number of words in an item and $\tilde{i}$ is the number of iterations for projection gradient algorithm. The final complexity for one full iteration of CTRank is $O(n\bar{n}K + m(\bar{T}K + \tilde{i}K))$ which is much lower than that of CTR, our major counterpart, $O(2n\bar{n}K^2 + (n+m)K^3 + m(\bar{T}K + \tilde{i}K))$.

Experiments

Experiment Settings

Dataset CiteULike[1] is an academic social network, which allows users to create individual reference libraries for the articles they like. In this work, we use a large dataset collected by Wang (Wang and Blei 2011) from CiteULike.

Articles in a user's reference library are considered as observed items. In this dataset, 5551 users expressed 204,986 observed ratings for 16,980 items (articles) with a high sparseness of 99.78%. On average, each user has 37 articles in the library, ranging from 1 to 321, and each article appears in 12 users' libraries, ranging from 1 to 321. By randomly removing observed feedback, we obtain 4 datasets with different sparsity levels of 99.80%, 99.85%, 99.90% and 99.95% correspondingly. For each article, its title and abstract are abstracted to form the bag-of-word representation of metedata. Removing stop words and choosing top words based on tf-idf (Manning, Raghavan, and Schütze 2008) construct a corpus with 1.6M terms from a vocabulary of size 8000.

Evaluation Following one-class CF literature (Rendle et al. 2009; Hariri, Mobasher, and Burke 2012; Paquet and Koenigstein 2013), we adopt the popular 10-fold leave-one-out cross validation to evaluate the performance of recommendation models. That is, we randomly remove one example from CiteULike dataset $\mathcal{R}^+$ to form test set $\mathcal{R}_{test}$, and the remaining constitutes the training set $\mathcal{R}_{train}$. The models are learned on $\mathcal{R}_{train}$ and their predicted ranking is evaluated on $\mathcal{R}_{test}$ by averaging HitRatio@N, the probability that the removed article is recommended as part of the top-N recommendations:

$$HitRatio@N = \frac{1}{|\mathcal{R}_{test}|} \sum_{(i,j)\in\mathcal{R}_{test}} \mathbb{1}(rank_{ij} < N),$$

where $rank_{ij}$ denotes the rank position of item j in user i's recommendation list. We repeat this process 10 times and report the average results. As the definition shows, a higher value of HitRatio@N indicates a better performance.

Parameter Effect on CTRank

We vary the dimensionality of latent space K in CTRank models in the range $\{100, 150, 200, 250\}$, and the precision parameter λ_v in the range $\{0.0025, 0.025, 0.25, 2.5\}$ with fixed $\lambda_u = 0.0025$. How proposed models behave with different combinations of parameters are shown in Fig. 2.

Effect of Dimensionality In Fig. 2(a), we investigate the effect of K on the performance of CTRank, measured by HitRatio@10. As expected, HitRatio@10 increases monotonically with the growing number of iterations. When the dimensionality grows from 100 to 200, CTRank achieves higher HitRatio@10. We also notice that CTRank with $K = 250$ yields slightly lower HitRatio@10 than $K = 200$ even though the former is the quickest approach leading to convergence.

[1]http://www.citeulike.org

Effect of Parameter λ_v In Fig. 2(b), we observe that the value of λ_v affects the recommendation performance significantly, which demonstrates that exploiting topic information can greatly improve recommendation accuracy. It is also observed that, as λ_v decreases, the HitRatio@10 value increases at first, but when λ_v goes beyond a certain threshold like 0.025, the HitRatio@10 value drops with further decrease of λ_v. This shows that purely using implicit feedback or purely using topic information cannot generate better performance than appropriately fusing these two sources together.

Effect of Loss Function The loss function determines how the ranking component impacts latent factors. As shown in Fig. 2(c), CTRank-log converges very early but only converges to a low value of HitRatio@10, whereas CTRank-squared obtains increasing HitRatio@10 as the growth of the number of iteration. The poor performance of CTRank-log arises from the limitation that the approaches minimizing log-loss function focus on incorrectly ordered pairs, and benefit little from pairs that are currently correctly ordered. In the scenarios where the majority of items are irrelevant, a sampled unobserved example is likely to be ranked correctly below an observed item, and thus these models do not benefit from the training examples, i.e., the gradient vanishes.

Performance Comparison

To evaluate the proposed model, we compare it with other models in datasets with varying sparsity levels.

Comparison Methods To study the effect of topic information discovered from meta-data, we consider the following general pair-wise ranking approaches which do not consider topic information.

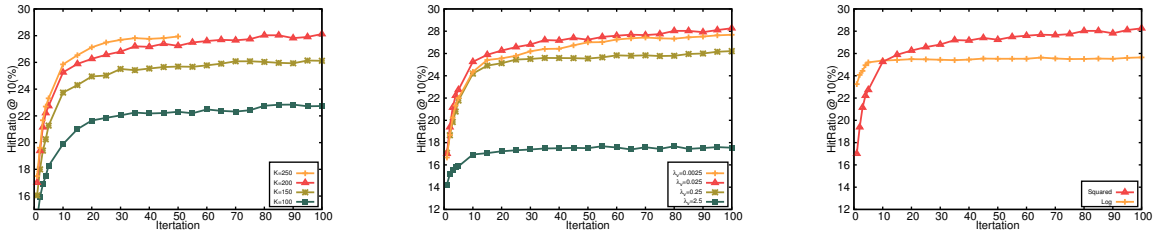
BPR: Bayesian Personalized Ranking optimization for MF (Rendle et al. 2009), which is the state-of-the-art approach for implicit feedback data. It can be absorbed as a special case of CTRank-log without considering topic information.

PALS: Pair-wise Alternating Least Squares, a variant of CTRank-squared without considering topic information.

To investigate the advantage of pair-wise methods over point-wise approaches, we compare CTRank with **CTR** (Wang and Blei 2011), a point-wise model, which links MF and LDA together.

Parameter Settings For all comparison methods, we set respective optimal parameters either according to corresponding references or based on our validation experiments. Specifically, we randomly select one test example per user from training set and use these examples as validation set to determine the trade-off parameters $\{K, \lambda_v, \lambda_u\}$. For all these latent factor based methods, we set the dimensionality of latent space $K = 200$ and the learning rate $\eta = 0.005$. For BPR, parameters λ_u and λ_v are 0.0025. For PALS $\lambda_v = \lambda_u = 0.0025$. For CTR, λ_u and λ_v are set to 0.01 and 100, respectively. For CTRank models, $\lambda_v = 0.025$, $\lambda_u = 0.0025$.

Comparison Results We report the comparison results in Table 2. The key conclusions are summarized as follows: (1)



(a) Effect of the dimensionality K. (b) Effect of the parameter λ_v. (c) Effect of the loss function.

Figure 2: Effect of parameters on the performance of CTRank. The sparsity level of the dataset used is 99.80%. Parameter settings: (a) $\lambda_v = 0.025$, $\lambda_u = 0.0025$ and squared loss is used. (b) $\lambda_u = 0.0025$, $K = 200$ and squared loss is used. (c) $\lambda_v = 0.025$, $\lambda_u = 0.0025$, $K = 200$.

Table 2: HitRatio (%) comparison over datasets with different sparsity levels. Numbers in boldface (e.g., **21.13**) and in Italic (e.g., *18.27*) are the best and second best results among all methods, respectively. N is the number of recommended items.

Methods	Sparsity											
	99.80%			99.85%			99.90%			99.95%		
	N=5	N=10	N=15	N=5	N=10	N=15	N=5	N=10	N=15	N=5	N=10	N=15
BPR	16.59	22.75	27.49	10.37	16.33	20.55	5.40	9.71	13.64	4.77	7.94	10.59
PALS	18.04	25.62	29.94	11.69	17.29	21.17	6.70	10.61	13.80	3.31	5.56	7.44
CTR	17.51	24.84	*30.23*	12.43	*19.04*	23.01	6.12	10.23	13.13	4.99	9.01	12.47
CTRank-log	*18.27*	*25.65*	29.43	*13.01*	18.84	*23.61*	*7.30*	*11.71*	*15.18*	*6.10*	*9.76*	*12.56*
CTRank-squared	**21.13**	**28.33**	**32.73**	**14.80**	**20.52**	**25.40**	**8.90**	**14.39**	**18.52**	**6.11**	**9.97**	**12.83**

The proposed CTRank-squared outperforms all other baselines at all sparsity levels while CTRank-log achieves the second best results in most cases. (2) The improvement from CTRank-squared over PALS and that from CTRank-log over BPR coincide with our assumption that meta-data can be used to differentiate the relevant items from the irrelevant ones. In particular, the relative improvement is more significant when the data is more sparse, which confirms the effectiveness of CTRank in alleviating data sparsity. (3) The pair-wise approaches directly optimizing ranking loss yields better performance than point-wise approaches, as revealed by CTRank and CTR.

Table 3: An example user from CiteULike dataset. We show some typical articles he/she likes, top 3 topics in u_i and top 5 recommended articles. $\surd$ means the target article.

Articles in his/her library	1. Parallel human genome analysis microarray based expression monitoring of genes 2. Gene expression data analysis 3. Experimental design for gene expression microarrays 4. The nature of statistical learning theory 5. On the convergence properties of the em algorithm 6. A tutorial on support vector machines for pattern recognition ...	
CTRank		
Top 3 topics	1. gene-expression-microarray-expressed-profiling-... 2. algorithms-class-optimization-output-binary-... 3. learning-classification-machine training-vector-...	
Top 5 Articles	1. Distinct types of diffuse large bcell lymphoma identified by gene expression profiling 2. An introduction to support vector machines and other kernelbased learning methods 3. Interpreting patterns of gene expression with selforganizing maps 4. The elements of statistical learning 5. Computational cluster validation in postgenomic data analysis	$\surd$

Interpretability

Besides promising recommendation performance, an important advantage of our model is that it can interpret fine-grained levels of latent spaces: (1) user preferences space (u_i) and (2) item characteristics space (v_j) using the topics identified by CTRank. In Table 3, we show an example user from CiteULike dataset. From the user's library, we may observe that he/she is interested in the topics "gene expression" and "machine learning". We also list the user's top-3 topics (found by $k = \arg\max_k u_{ik}$) along with top-5 articles recommended by CTRank. It is obvious that user preferences can be well explained by the learned topics, and the recommended articles are closely linked to these topics. Likewise, we can also interpret item characteristics using top topics discovered by CTRank. And the overlapping topics between users and items may serve to generate explanations for recommendations.

Conclusion

In this paper, we propose a novel hierarchical Bayesian framework CTRank that incorporates "bag-of-words" type meta-data on items into pair-wise ranking models for alleviating the data sparsity problem. By regularizing item factors through item topic proportions, CTRank extends pair-wise ranking models with topic models. This design allows to learn expressive latent factors for users and items from extremely sparse implicit feedback. As an additional benefit, CTRank is able to provide interpretable user factors and item factors. This property may help in generating explanations for recommendations in some applications.

Acknowledgment

This work is partially supported by the National Natural Science Foundation of China (Grant No. 61272480, 61332013 and 71372188) and National Center for International Joint Research on E-Business Information Processing under Grant 2013B01035.

References

Agarwal, D., and Chen, B.-C. 2010. flda: matrix factorization through latent dirichlet allocation. In *WSDM'10*, 91–100. ACM.

Bertsekas, D. P. 1999. Nonlinear programming.

Blei, D. M.; Ng, A. Y.; and Jordan, M. I. 2003. Latent dirichlet allocation. *JMLR* 3:993–1022.

Chen, K.; Chen, T.; Zheng, G.; Jin, O.; Yao, E.; and Yu, Y. 2012. Collaborative personalized tweet recommendation. In *SIGIR'12*, 661–670. ACM.

Freund, Y.; Iyer, R.; Schapire, R. E.; and Singer, Y. 2003. An efficient boosting algorithm for combining preferences. *JMLR* 4:933–969.

Gunawardana, A., and Meek, C. 2009. A unified approach to building hybrid recommender systems. In *RecSys'09*, 117–124. ACM.

Hariri, N.; Mobasher, B.; and Burke, R. 2012. Context-aware music recommendation based on latenttopic sequential patterns. In *RecSys'12*, 131–138. ACM.

Hu, Y.; Koren, Y.; and Volinsky, C. 2008. Collaborative filtering for implicit feedback datasets. In *ICDM'08*, 263–272. IEEE.

Joachims, T. 2002. Optimizing search engines using clickthrough data. In *KDD'02*, 133–142. ACM.

Kim, B. M.; Li, Q.; Park, C. S.; Kim, S. G.; and Kim, J. Y. 2006. A new approach for combining content-based and collaborative filters. *Journal of Intelligent Information Systems* 27(1):79–91.

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

Krohn-Grimberghe, A.; Drumond, L.; Freudenthaler, C.; and Schmidt-Thieme, L. 2012. Multi-relational matrix factorization using bayesian personalized ranking for social network data. In *WSDM'12*, 173–182. ACM.

Lee, J.; Bengio, S.; Kim, S.; Lebanon, G.; and Singer, Y. 2014. Local collaborative ranking. In *WWW'14*, 85–96. International World Wide Web Conferences Steering Committee.

Li, B.; Yang, Q.; and Xue, X. 2009. Can movies and books collaborate? cross-domain collaborative filtering for sparsity reduction. In *IJCAI'09*, volume 9, 2052–2057.

Liu, T.-Y. 2009. Learning to rank for information retrieval. *Foundations and Trends in Information Retrieval* 3(3):225–331.

Manning, C. D.; Raghavan, P.; and Schütze, H. 2008. *Introduction to information retrieval*, volume 1. Cambridge university press Cambridge.

McAuley, J., and Leskovec, J. 2013. Hidden factors and hidden topics: understanding rating dimensions with review text. In *RecSys'13*, 165–172. ACM.

Pan, W., and Chen, L. 2013a. Cofiset: Collaborative filtering via learning pairwise preferences over item-sets. In *ICDM'13*, 180–188. SIAM.

Pan, W., and Chen, L. 2013b. Gbpr: group preference based bayesian personalized ranking for one-class collaborative filtering. In *IJCAI'13*, 2691–2697. AAAI Press.

Pan, R.; Zhou, Y.; Cao, B.; Liu, N. N.; Lukose, R.; Scholz, M.; and Yang, Q. 2008. One-class collaborative filtering. In *ICDM'08*, 502–511. IEEE.

Pan, W.; Xiang, E. W.; Liu, N. N.; and Yang, Q. 2010. Transfer learning in collaborative filtering for sparsity reduction. In *AAAI'10*, volume 10, 230–235.

Paquet, U., and Koenigstein, N. 2013. One-class collaborative filtering with random graphs. In *WWW'13*, 999–1008. International World Wide Web Conferences Steering Committee.

Qiang, R.; Liang, F.; and Yang, J. 2013. Exploiting ranking factorization machines for microblog retrieval. In *CIKM'13*, CIKM '13, 1783–1788. New York, NY, USA: ACM.

Rendle, S., and Freudenthaler, C. 2014. Improving pairwise learning for item recommendation from implicit feedback. In *WSDM'14*, 273–282. ACM.

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

Rendle, S. 2010. Factorization machines. In *ICDM'10*, 995–1000. IEEE.

Shan, H., and Banerjee, A. 2010. Generalized probabilistic matrix factorizations for collaborative filtering. In *ICDM'10*, 1025–1030. IEEE.

Takács, G., and Tikk, D. 2012. Alternating least squares for personalized ranking. In *RecSys'12*, 83–90. ACM.

Tang, X., and Zhou, J. 2013. Dynamic personalized recommendation on sparse data. *IEEE Transactions on Knowledge and Data Engineering* 25(12):2895–2899.

Vig, J.; Sen, S.; and Riedl, J. 2009. Tagsplanations: explaining recommendations using tags. In *IUI'09*, 47–56. ACM.

Wang, C., and Blei, D. M. 2011. Collaborative topic modeling for recommending scientific articles. In *KDD'11*, 448–456. ACM.