Probabilistic Latent Preference Analysis for Collaborative Filtering

Nathan N. Liu
Department of Computer Science and Engineering
Hong Kong University of Science and Technology,
Hong Kong, China
nliu@cse.ust.hk

Min Zhao
NEC Labs China
Beijing, China
zhaomin@research.nec.c-om.cn

Qiang Yang
Department of Computer Science and Engineering
Hong Kong University of Science and Technology,
Hong Kong, China
qyang@cse.ust.hk

ABSTRACT

A central goal of collaborative filtering (CF) is to rank items by their utilities with respect to individual users in order to make personalized recommendations. Traditionally, this is often formulated as a rating prediction problem. However, it is more desirable for CF algorithms to address the ranking problem directly without going through an extra rating prediction step. In this paper, we propose the probabilistic latent preference analysis (pLPA) model for ranking predictions by directly modeling user preferences with respect to a set of items rather than the rating scores on individual items. From a user's observed ratings, we extract his preferences in the form of pairwise comparisons of items which are modeled by a mixture distribution based on Bradley-Terry model. An EM algorithm for fitting the corresponding latent class model as well as a method for predicting the optimal ranking are described. Experimental results on real world data sets demonstrated the superiority of the proposed method over several existing CF algorithms based on rating predictions in terms of ranking performance measure NDCG.

Categories and Subject Descriptors

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

General Terms

Algorithms, Experimentation

Keywords

Collaborative Filtering, Ranking, Latent Class Model

1. INTRODUCTION

With the explosive growth of easily accessible information on the web, technologies for helping people efficiently sift through huge amount of information is becoming indispensable in order to overcome the resulted information overload problem. Recommender system is a promising technology that aims to automatically generate item recommendations from a huge collection of items based on users' past feedback. Broadly speaking, existing technologies underlying recommender systems fall into either of the following two categories: *content-based filtering* versus *collaborative filtering.* Content-based filtering approach analyzes the content information associated with the items and users such as product descriptions, user profiles etc., in order to represent users and items using a set of features. To recommend new items to a user, content-based filters match their representations to those items the user has expressed interests on. In contrast, the collaborative filtering(CF) approach does not require any content information about the items, it works by collecting ratings on the items by a large number of users and make recommendations to a user based on the preference patterns of other users. The CF approach is based on the assumption that a user is often interested in those items that have been selected by some users with similar tastes. Besides avoiding the need for collecting extensive information about items or users, the CF approach does not require domain knowledge and can be easily adopted in different recommender systems.

A very important function of most recommender systems is the generation of the Top-N item list for each user in order to make personalized recommendations, which essentially involves solving a ranking problem. To rank items, most collaborative filtering algorithms formulate this as a rating prediction problem in which a user's potential ratings on the items are first predicted and then used to order the items. However, there are several drawbacks with such rating prediction based framework. Firstly, the rating prediction accuracy, which has been optimized in most existing methods, is not always consistent with ranking effectiveness. Suppose the true ratings on a pair of items are 3 and 4 respectively and the predicted ratings by method A are 2 and 5 while method B's predictions are 4 and 3. In terms of rating prediction accuracy as measured by the absolute deviation from the true ratings, the two methods are indifferent. However, the resulted ranking of the two items based on method B's predictions is incorrect while method A can still ensure the correct order of the two items. So models that solely focus on improving rating prediction accuracies may be suboptimal for ranking applications. Secondly, ratings are often predicted independently for each item while rankings are inherently about relations among multiple items. For example, one common form of such relations is a pairwise preference which indicates which one of a pair of items is more preferred by the user. This form of relational information is clearly more relevant to task of ranking than the rating scores. Therefore, it is more desirable for a model to explicitly take such interdependencies into account and conduct collective inference which computes the ranking as a function of the whole set of items. Finally, in many interactive applications such as web search, explicit user feedbacks in the form of ratings are often unavailable while it is very easy to collect abundant implicit feedbacks such as user click-throughs, from which one can often extract pairwise preferences regarding items[12, 19]. Therefore models for preferences are much more general and can handle both implicit and explicit user feedbacks.

In this paper, we propose a ranking prediction based approach to collaborative filtering named probabilistic latent preference analysis (pLPA), which directly addresses the item-ranking problem without going through the intermediate step of rating prediction. The pLPA model is a latent class model that simultaneously captures user preferences as well as the user community structures. Unlike previous memory based methods[14], the pLPA model is based on a compact statistical model of the data, which allows learning and inference to be efficiently conducted on large scale data sets.

2. RELATED WORKS

Generally speaking, there are two common approaches to collaborative filtering. One is the memory-based approach and the other is the model-based approach.

2.1 Memory-based Approach

Memory-based approach conducts certain forms of nearest neighbor search in order to predict the rating for particular user-item pair. A most common method is the user-based model, which estimate the unknown ratings of a target user based on the ratings by a set of neighboring users that tend to rate similarly to the target user. A crucial component of the user-based model is the user-user similarity for determining the set of neighbors. Popular similarity measures include the Pearson Correlation Coefficient(PCC)[21, 6]and the vector similarity(VS)[2].

One difficulty in measuring the user-user similarity is that the raw ratings may contain biases caused by the different rating behaviors of different users. For example, some users may tend to give high ratings. To correct such biases, different methods have been proposed to normalize or center the data prior to measuring user similarities. [21, 2] showed that by correcting for user-specific means the prediction quality could be improved. Later, Jin et al. proposed a technique for normalizing the user ratings based on the halfway accumulative distribution[11].

Another difficulty in user-based models arises from the fact that the known user-item ratings data is typically highly sparse, which makes it very hard to find highly similar neighbors for making accurate predictions. To alleviate such sparsity problem, different techniques have been proposed to fill in some of the unknown ratings in the matrix such as dimensionality reduction[5] and data-smoothing methods[24, 15].

An alternative form of the neighborhood-based approach is the item-based model[22, 13]. Here the item-item similarity is used to select a set of neighboring items that have been rated by the target user and the ratings on the unrated items are predicted based on his ratings on the neighboring items. Since the number of items is usually much less than the number of users in most applications, item-item similarities are less sensitive to the data sparsity problem. Sarwar et al.[22] recommended using the adjusted cosine similarity to compute the item-item similarity and found that the item-based model could obtain higher accuracy than the user-based model, while allowing more efficient computations.

EigenRank[14] is a recently proposed memory based method which also views CF as a ranking problem rather than a regression problem(i.e., rating prediction). It adapts traditional user-based model by measuring user similarity using rank correlations and using user dependent rank aggregation methods to produce rankings by fusing the preferences of a user's neighbors. It was empirically demonstrated that such ranking oriented approach can produce better rankings than traditional rating prediction methods. However, like any memory based methods, it requires nearest neighbor search in the entire rating database at prediction time, which is very time consuming on large data sets.

2.2 Model-based Approach

Memory based methods became popular because they are conceptually simple and intuitive. However, there are several shortcomings with these methods. First, the accuracies of memory-based methods are often suboptimal. Secondly, while memory-based methods can produce prediction, they do not involve much learning, thus little knowledge or insights about the users or items can be gained from the data. Thirdly, memory-based methods often require direct manipulation the full ratings database at prediction time, which can be very computationally expensive. Finally, without a proper model, it is hard to adapt memory-based methods to optimize different objectives associated with specific tasks or application domains.

The model-based approach to CF uses the observed user-item ratings to train a compact model that explains the given data so that ratings could be predicted via the model instead of directly searching in the rating database as the memory-based approach does. Models in this category include matrix factorization[17], probabilistic latent semantic analysis[8] and Bayesian networks[18].

Model based methods can effectively address the limitations of memory based methods. They are often found to yield superior performance. By compressing the full data into a very compact statistical model, predictions can be made in constant time although there involves an additional model building stage that can be done off-line. Besides making predictions, a model also has well defined semantics and structures that can capture useful information such as user communities. Moreover, by designing appropriate loss functions and optimization procedures, models can be systematically tuned to suit task-dependent objectives.

In this work, we also take the model based approach considering its many attractive properties described above. In the following sections, we will first review several existing model based methods for rating prediction and then describe our probabilistic latent preference analysis model for ranking prediction.

3. MODEL-BASED RATING PREDICTION

The general problem setting for collaborative filtering is as follows. Suppose we are given a set of m users and a set of n items. Each observed rating record is a triple (u, i, r_{ui}) denoting the rating assigned to item i by user u, where the user index $u \in \{1, 2, ..., m\}$, the item index $i \in \{1, 2, ..., n\}$ and the rating values $r_{ui} \in \mathbb{R}$. We assume each user can rate each item only once so all the ratings can be arranged into an $m \times n$ matrix $\mathbf{R}$ whose ui-th entry equals r_{ui}. If a user has not rated an item, the corresponding entry in R is unknown. We let $\mathcal{R}$ denote the set of all (u, i) pairs associated with the observed ratings. Typically, $|\mathcal{R}|$ is much smaller than $m \times n$ since each user only rate very few items, this is commonly known as the *sparsity* problem

3.1 Matrix Factorization

The goal of matrix factorization (MF) techniques is to approximate the observed rating matrix $\mathbf{R}$ as the product of two low-rank matrices:

$$\mathbf{R} \approx \mathbf{U}^\top \mathbf{V}, \tag{1}$$

where $\mathbf{U}$ is an $k \times m$ matrix, $\mathbf{V}$ is an $k \times n$ matrix. The parameter k controls the number of latent features for each user and movie which is typically much smaller than m and n.

Under this model, each user u is represented by a k-dimensional feature vector $\mathbf{u}_u \in \mathbb{R}^k$, the u-th column of $\mathbf{U}$, while each item i is modeled by $\mathbf{v}_i \in \mathbb{R}^k$, the i-th column of $\mathbf{V}$. The predicted rating on item i by user u is equal to the inner product of the corresponding user and item features:

$$\widehat{r}_{ui} = \sum_{k=1}^{K} u_{ku} v_{ki} = \mathbf{u}_u^\top \mathbf{v}_i \tag{2}$$

Once the matrices $\mathbf{U}$ and $\mathbf{V}$ have been learnt, each prediction can be made in just $O(k)$ time. The parameter matrices $\mathbf{U}$ and $\mathbf{V}$ can be found by solving the following optimization problem:

$$\min_{\mathbf{U} \in \mathbb{R}^{k \times m}, \mathbf{V} \in \mathbb{R}^{k \times n}} \sum_{(u,i) \in \mathcal{R}} (r_{ui} - \mathbf{u}_u^\top \mathbf{v}_i)^2 + \lambda(\|\mathbf{U}\|_F^2 + \|\mathbf{V}\|_F^2), \tag{3}$$

where $\|.\|_F^2$ denotes the Frobenius 2-norm, (i.e., $\|\boldsymbol{U}\|_F^2 = \sum_{ui} u_{ui}^2$) and λ is the parameter controlling the strength of regularization in order to avoid over-fitting. The problem (3) can be solved using an EM like algorithm[23], where we update $\mathbf{U}$ and $\mathbf{V}$ alternately while holding the other matrix fixed. When $\mathbf{U}$ (or $\mathbf{V}$) is fixed, minimizing $\mathcal{L}$ over $\mathbf{V}$ (or $\mathbf{U}$) simply involves solving an unconstrained quadratic optimization problem which can be done very efficiently.

3.2 Probabilistic Latent Semantic Analysis

Compared with matrix factorization methods, statistical models are able to capture the complex dependencies among different factors such as user clusters, item clusters and ratings using well-defined probabilistic semantics. They are thus much easier to interpret in order to gain insights about the data and can also rely on rich existing statistical techniques for doing learning and inferences.

Probabilistic latent semantic analysis (pLSA)[7] is a widely used latent variable model for co-occurrence data based on mixture distributions. To apply the pLSA framework to collaborating filtering applications, Hofmann [8] proposed to introduce a hidden state variable z for each observed rating r_{ui} the distribution of which is decomposed as:

$$P(r_{ui}|u, i) = \sum_{z=1}^{k} P(r_{ui}|i, z) P(z|u) \tag{4}$$

where the sum over z runs over all latent classes and k is a parameter controlling the number of latent classes. The latent classes can capture the user communities while the mixing proportions $P(z|u)$ capture the strength of a user's membership in each community.

For rating values that are of numeric scale, the Gaussian distribution can be used to model the rating value for each item i in each latent class z by introducing a mean parameter $\mu_{ki} \in \mathbb{R}$ and a variance parameter $\sigma_{ki}^2 \in \mathbb{R}^+$, so the model defined in (4) becomes a gaussian mixture model:

$$\begin{aligned} P(r_{ui}|u, i) &= \sum_{z=1}^{k} P(z|u) P(r_{ui}; \mu_{zi}, \sigma_{zi}) && (5) \\ P(r_{ui}; \mu_{zi}, \sigma_{zi}) &= \frac{1}{\sqrt{2\pi}\sigma_{zi}} \exp\left[-\frac{(r_{ui} - \mu_{zi})^2}{\sigma_{zi}^2}\right] && (6) \end{aligned}$$

To make predictions, we can compute the expected rating that would be assigned to item i by user u:

$$\begin{aligned} \widehat{r}_{ui} = \int r P(r|u, i) &= \sum_{z=1}^{k} P(z|u) \int r P(r; \mu_{zi}, \sigma_{zi}) dr \\ &= \sum_{z} P(z|u) \mu_{zi} && (7) \end{aligned}$$

which again can be computed in $O(k)$ time.

The model parameters in (5) thus consist of $n \times k$ means μ_{zi}, $n \times k$ variances σ_{zi}^2 and the $m \times k$ user dependent mixing proportions $P(z|u)$. Using the maximum likelihood approach, we can estimate these parameters by maximizing the log-likelihood of all the observed ratings:

$$\begin{aligned} \mathcal{L} &= \sum_{(u,i) \in \mathcal{R}} \log(P(r_{ui}|u, i)) && (8) \\ &= \sum_{(u,i) \in \mathcal{R}} \log \left\{ \sum_{z=1}^{k} P(r_{ui}; \mu_{zi}, \sigma_{zi})\, P(z|u, i) \right\} \end{aligned}$$

where the expansion follows equation (5) and (6).

The above loss function can be optimized using the EM algorithm which alternates between a expectation step and a maximization step. Let there be a latent variable z associated with each observed rating $(u, i) \in \mathcal{R}$. In the expectation step, the posterior probabilities of the latent variables are computed:

$$P(z|u, i) = \frac{P(z|u) P(r_{ui}; \mu_{zi}, \sigma_{zi})}{\sum_{z'=1}^{k} P(z'|u) P(r_{ui}; \mu_{z'i}, \sigma_{z'i})} \tag{9}$$

In the maximization step, the parameters μ_{zi}, σ_{zi}^2 and $P(z|u)$ can be obtained by maximizing the expected complete log-likelihood $\mathbf{E}[\mathcal{L}^c]$:

$$\mathbf{E}[\mathcal{L}^c] = \sum_{(u,i) \in \mathcal{R}} \sum_{z=1}^{k} P(z|u, i) \log \left[P(r_{ui}; \mu_{zi}, \sigma_{zi}) P(z|u) \right] \tag{10}$$

When class conditional rating distributions are modeled by gaussian distributions, the above optimization problem can be readily solved by introducing an additional Lagrange multiplier to enforce that user specific mixing proportions $P(z|u)$ sum to 1 and solve the resulting constrained optimization problem, which has the following closed form solutions:

$$P(z|u) = \frac{\sum_{(u',i)\in\mathcal{R}:u'=u} P(z|u,i)}{\sum_{z'=1}^{k}\sum_{(u',i)\in\mathcal{R}:u'=u} P(z'|u,i)} \quad (11)$$

$$\mu_{zi} = \frac{\sum_{(u,i')\in\mathcal{R}:i'=i} r_{ui}P(z|u,i)}{\sum_{(u,i')\in\mathcal{R}:i'=i} P(z|u,i)} \quad (12)$$

$$\sigma_{zi}^2 = \frac{\sum_{(u,i')\in\mathcal{R}:i'=i}(r_{ui}-\mu_{zi})^2 P(z|u,i)}{\sum_{(u,i')\in\mathcal{R}:i'=i} P(z|u,i)} \quad (13)$$

The expectation and maximization are alternated until a termination condition is met, for example, by checking the convergence of the log-likelihood defined in (4) is reached. In the experiments, we found that 30-50 iterations are often sufficient.

4. PROBABILISTIC LATENT PREFERENCE ANALYSIS

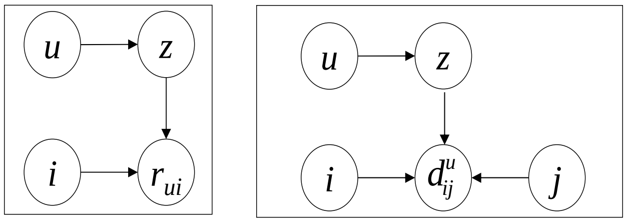


Figure 1: Graphical model representation of (a) probabilistic latent semantic analysis and (b) probabilistic latent preference analysis

Both matrix factorization (MF) and probabilistic latent semantic analysis (pLSA) are based on models of the underlying mechanism of how a user would assign a particular rating score to an item. However as we argued in the previous sections, a more important goal of collaborative filtering is to rank items correctly. Therefore we propose a new latent class model named probabilistic latent preference analysis (pLPA) for modeling user's preferences regarding the items rather than the ratings scores. In particular, we focus on modeling user preferences in the form of paired comparisons which are binary responses that indicate whether a user likes an item more than another. In contrast to pLSA, which employs a Gaussian distribution for the rating on each item for each latent class, pLPA uses the Bradley-Terry model for each latent class's preferences on pairs of items.

4.1 Bradley-Terry Model

Consider the situation when there is a set of n items which are compared to each other in pairs to yield binary outcomes δ_{ij} which is equal to 1 if i is preferred to j and 0 otherwise. The Bradley-Terry model for the probability distribution of δ_{ij} with parameters $\boldsymbol{\gamma} \in \mathbb{R}_n^+$:

$$P(\delta_{ij}=1;\boldsymbol{\gamma}) = \frac{\gamma_i}{\gamma_i+\gamma_j} \quad (14)$$

where the γ_i's are nonnegative item parameters indicating the utility of an item such that the higher γ_i is compared to γ_j the more likely item i will be preferred to j.

The Bradley-Terry model for paired comparisons can be used to define a probability distribution over rankings[9, 16]. Let $\mathbf{P}_n$ denote the set of all possible permutations on the set of integers from 1 to n. A vector $\boldsymbol{\pi} \in \mathbf{P}_n$ defines a ranking for a set of n items such that $\pi_i = 1$ means that item i is ranked first. Given a ranking $\boldsymbol{\pi}$, we can define a variable $\delta_{ij}^{\boldsymbol{\pi}}$ with respect to $\boldsymbol{\pi}$ such that $\delta_{ij}^{\boldsymbol{\pi}} = 1$ if $\pi_i < \pi_j$ and 0 otherwise. The probability distribution over the set of all possible rankings $\mathbf{P}_n$ is therefore:

$$P(\boldsymbol{\pi};\boldsymbol{\gamma}) = \frac{1}{C(\boldsymbol{\gamma})}\prod_{i=1}^{n-1}\prod_{j=i+1}^{n} P(\delta_{ij}^{\boldsymbol{\pi}};\boldsymbol{\gamma}) \quad (15)$$

where $C(\boldsymbol{\gamma}) = \sum_{\boldsymbol{\pi}\in\mathbf{P}_n}\prod_{i=1}^{n-1}\prod_{j=1+1}^{n} P(\delta_{ij}^{\boldsymbol{\pi}};\boldsymbol{\gamma})$ is a normalization constant that guarantees it is a probability measure on $\mathbf{P}_n$, the set of valid ranking with no ties. Inserting (14) into (15) yields

$$P(\boldsymbol{\pi};\boldsymbol{\gamma}) = \frac{1}{C^*(\boldsymbol{\gamma})}\prod_{i=1}^{n}\gamma_i^{n-\pi_i} \quad (16)$$

where $C^*(\boldsymbol{\gamma}) = C(\boldsymbol{\gamma})\prod_{i=1}^{n-1}\prod_{j=i+1}^{n}(\gamma_i+\gamma_j)$ is a constant factor not depending on $\boldsymbol{\pi}$. From (16), we can easily see that under a Bradley-Terry model there is a single most probable ranking $\boldsymbol{\pi}^* = \arg\max_{\boldsymbol{\pi}} P(\boldsymbol{\pi};\boldsymbol{\gamma})$ that is equal to $\arg\operatorname{sort}(\boldsymbol{\gamma})$, which orders $\{1,2,...,n\}$ by the corresponding γ_i's into decreasing order.

4.2 Latent Class Models for User Preferences

In collaborative filtering, given a user's observed ratings r_{ui}, we can extract his preferences on item pairs by defining variables δ_{ij}^u such that $\delta_{ij}^u = 1$ if $r_{ui} > r_{uj}$ and 0 otherwise. Note that δ_{ij}^u is unknown if either r_{ui} or r_{uj} is unknown or $r_{ui} = r_{uj}$. We let $\mathcal{Q}$ denote the set of (u,i,j) triples for which the preference is δ_{ij}^u is observed.

Similar to pLSA, we model each observed pairwise preference by a mixture distribution:

$$P(\delta_{ij}^u|u,i,j) = \sum_{z=1}^{k} P(\delta_{ij}^u|i,j,z)P(z|u) \quad (17)$$

The graphical model representations for both pLSA and pLPA are depicted in Figure 1. In pLPA, we designate a Bradley-Terry Model with parameter $\boldsymbol{\gamma}_z \in \mathbb{R}_n^+$ to model $P(\delta_{ij}^u|i,j,z)$:

$$P(\delta_{ij}^u=1|z,i,j) = P(\delta_{ij}^u=1;\boldsymbol{\gamma}_z) = \frac{\gamma_{zi}}{\gamma_{zi}+\gamma_{zj}} \quad (18)$$

The parameters of the pLPA model thus consist of $n\times k$ item parameters γ_{zi} and $m\times k$ user mixing proportions $P(z|u)$. Like pLSA, we can use the EM algorithm to obtain maximum likelihood estimates of the parameters by maximizing the log-likelihood of the observed user pairwise preferences δ_{ij}^u's. In the expectation step, the posterior probability of the latent variable for each observed pairwise preference is computed:

$$P(z|u,i,j) = \frac{P(z|u)P(\delta_{ij}^u;\boldsymbol{\gamma}_z)}{\sum_{z'=1}^{k}P(z'|u)P(\delta_{ij}^u;\boldsymbol{\gamma}_{z'})} \quad (19)$$

For the maximization step, we first derive the expected complete log-likelihood $\mathbf{E}[\mathcal{L}^c]$:

$$\sum_{(u,i,j)\in\mathcal{Q}} \sum_{z=1}^{k} P(z|u,i,j)\big[\log P(\delta_{ij}^u;\boldsymbol{\gamma}_z) + \log P(z|u)\big] \quad (20)$$

Optimizing $\mathbf{E}[\mathcal{L}^c]$ with respect to $P(z|u)$ while enforcing the normalization constraint $\sum_{z=1}^{k} P(z|u) = 1$ yields the following update equations for $P(z|u)$:

$$P(z|u) = \frac{\sum_{(u',i,j)\in\mathcal{Q}:u'=u} P(z|u,i,j)}{\sum_{z'=1}^{k}\sum_{(u',i,j)\in\mathcal{Q}:u'=u} P(z'|u,i,j)} \quad (21)$$

Unfortunately, unlike in pLSA, there is no closed form solutions for the parameters γ_{zi} in the maximization step for pLPA. We therefore has to resort to numeric methods to obtain maximum likelihood estimates of γ_{zi}'s. To estimate the parameter vector $\boldsymbol{\gamma}_z$ for each latent class z, the only relevant part in (20) is:

$$\begin{aligned}\mathcal{L}_z^c(\boldsymbol{\gamma}_z) &= \sum_{(u,i,j)\in\mathcal{Q}} \sum_{z=1}^{k} P(z|u,i,j)\log P(\delta_{ij}^u;\boldsymbol{\gamma}_z) \\ &= \sum_{i=1}^{n}\sum_{j=1}^{n} w_{ij}^z\big[\log\gamma_{zi} - \log(\gamma_{zi}+\gamma_{zj})\big] \quad (22)\end{aligned}$$

where $w_{ij}^z = \sum_{(u,i,j)\in\mathcal{Q}} P(z|u,i,j)$ is the expected number of times item i is preferred to item j in latent class z. Each $\mathcal{L}_z^c(\boldsymbol{\gamma}_z)$ could be maximized separately to obtain estimates of $\boldsymbol{\gamma}_z$ in the current maximization step.

We use a recently proposed iterative algorithm for obtaining maximum likelihood estimates of parameters in Bradley-Terry models known as the minorization-maximization (MM) algorithm[9], which has guaranteed convergence to the unique maximum likelihood estimator. Suppose $\boldsymbol{\gamma}_z^{(t)}$ is the estimate at the t-th iteration. Fix $\boldsymbol{\gamma}_z^{(t)}$, we can define the function:

$$\mathcal{Q}_t(\boldsymbol{\gamma}_z) = \sum_{i=1}^{n}\sum_{j=1}^{n} w_{ij}^z\left[\log\gamma_i - \frac{\gamma_i+\gamma_j}{\gamma_i^{(t)}+\gamma_j^{(t)}} - \log(\gamma_i^{(t)}+\gamma_j^{t}) + 1\right] \quad (23)$$

Utilizing the strict concavity of the logarithm function, which implies for positive x and y that $-\log x \geq 1 - \log y - (x/y)$. We can show that the function $\mathcal{Q}_t(\boldsymbol{\gamma}_z)$ is a *minorization* of $\mathcal{L}_z^c(\boldsymbol{\gamma}_z)$ at point $\boldsymbol{\gamma}_z^{(t)}$ such that $\mathcal{Q}_t(\boldsymbol{\gamma}_z) \leq \mathcal{L}_z^c(\boldsymbol{\gamma}_z)$ with equality only if $\boldsymbol{\gamma}_z = \boldsymbol{\gamma}_z^{(t)}$, As a result, we can easily verify that $\mathcal{Q}_t(\boldsymbol{\gamma}_z) \geq \mathcal{Q}_t(\boldsymbol{\gamma}_z^{(t)})$ implies $\mathcal{L}_z^c(\boldsymbol{\gamma}_z) \geq \mathcal{L}_z^c(\boldsymbol{\gamma}_z^{(t)})$. This property suggests an iterative algorithm in which we maximize $\mathcal{Q}_t(\boldsymbol{\gamma}_z)$ at each iteration and let $\boldsymbol{\gamma}_z^{t+1}$ be the maximizer of $\mathcal{Q}_t(\boldsymbol{\gamma}_z)$, which yields the following update equation:

$$\gamma_{zi}^{(t+1)} = W_i^z\left[\sum_{j\neq i}\frac{N_{ij}^z}{\gamma_{zi}^{(t)}+\gamma_{zj}^{(t)}}\right] \quad (24)$$

where $W_i^z = \sum_{j=1}^{n} w_{ij}^z$ and $N_{ij}^z = w_{ij}^z + w_{ji}^z$.

The above algorithm for fitting Bradley-Terry Models does not require any second-order information like the Newton algorithm does, thus each iteration can be computed very efficiently. In the experiments, we have found it normally takes 30-50 iteration to converge to the maximum likelihood estimations.

4.3 Ranking Prediction

Following the pairwise decomposition approach in (15), the pLPA model can be used to obtain the following probability distribution over rankings $\boldsymbol{\pi} \in \mathbf{P}_n$:

$$\begin{aligned}P(\boldsymbol{\pi}|u) &= \frac{1}{C(u)}\prod_{i=1}^{n-1}\prod_{j=i+1}^{n} P(\delta_{ij}^{\boldsymbol{\pi}}|u) \quad (25)\\ &= \frac{1}{C(u)}\prod_{i=1}^{n-1}\prod_{j=i+1}^{n}\sum_{z=1}^{k} P(\delta_{ij}^{\boldsymbol{\pi}};\boldsymbol{\gamma}_z)P(z|u)\end{aligned}$$

where $C(u)$ is a user-dependent normalization constant. Unlike rating scores, a ranking $\boldsymbol{\pi}$ is a discrete combinatorial structure, thus we could not take expectations of $\boldsymbol{\pi}$ as in (7). Instead we need to find the ranking with the maximum a posteriori (MAP) probabilities under (26):

$$\widehat{\boldsymbol{\pi}}_u = \arg\max_{\boldsymbol{\pi}\in\mathbf{P}_n} P(\boldsymbol{\pi}|u) \quad (26)$$

However, unlike the case of a single Bradley-Terry model, (26) is a mixture of Bradley-Terry models each of which has its respective MAP ranking determined by $\boldsymbol{\gamma}_z$. So the MAP ranking has to be obtained by performing combinatorial search in the set $\mathbf{P}_n$. Taking the logarithm of $P(\boldsymbol{\pi}|u)$ and remove all factors not dependent on $\boldsymbol{\pi}$, the MAP ranking $\boldsymbol{\pi}^*$ is the ranking that maximizes the following function:

$$V_u(\boldsymbol{\pi}) = \sum_{(i,j):\pi_i<\pi_j} \omega_{ij}^u \quad (27)$$

where $\omega_{ij}^u = \log P(\delta_{ij} = 1|u)$. Roughly speaking, we want to obtain the ranking that maximizes the probabilities of a higher ranked item being more preferred to those items ranked lower. Unfortunately, finding the optimal ranking to maximize a function of the form in (27) turn out to be a NP-complete problem, which can be shown via reduction from the cyclic-ordering problem[4].

In the following, we describe a strategy for efficiently producing a ranking for the pLPA model inspired by the rank aggregation problem in meta search. In the meta search scenario, there are multiple judges each of which provides a list of ranked results and the goal of rank aggregation is to effectively combine the individual judges' scores to produce a good overall ranking. Most methods for rank aggregation rely on the following information: (i) the scores assigned to each item by different judges; and/or (ii) the ordinal ranks of each item in different result lists. Different methods have been proposed to compute an overall score for each item based on the original scores, ranks or some transformation of these values so that items could be ordered by the overall scores[20]. To apply rank aggregation to produce rankings from the pLPA model, we consider the Bradley-Terry model of each latent class, $P(\delta_{ij};\boldsymbol{\gamma}_z)$, as a judge which assigns the scores $\gamma_{z1},...,\gamma_{z2}$ to the n items accordingly. Here the item parameters γ_{zi} are a natural measure for ranking since items with higher γ_{zi} values are more likely to be more preferred under the Bradley-Terry model. To produce user dependent rankings, we also consider user-dependent weights on each of the judges using the mixing proportions $P(z|u)$. This leads to the following formula for item scores used for ranking:

$$\theta_{ui} = \sum_{z=1}^{k} P(z|u)\gamma_{zi} \quad (28)$$

which can be computed in $O(k)$ time, the same as the time complexity of making predictions using the rating oriented pLSA model.

Note that we can also first perform normalization on the raw values of the item parameters γ_{zi} in different Bradley-Terry models before combining them. However, in the experiments, this was not found to yield any improvement over just using the raw scores. This is probably because in the pLPA models all the parameters γ_{zi} are learnt jointly based on the data so the different scales of the γ_{zi}'s are actually capturing certain useful properties of the data rather than being a kind of bias associated with different judges that needs to be corrected like in the meta search setting.

4.4 Comparison with Other Models

The pLPA and pLSA models are closely related as both are latent class models with different choice of the mixture component distributions. pLSA focuses on modeling the rating scores, which makes gaussian distribution a natural use. pLPA is designed to capture relative ordering of items, for which Bradley-Terry model is more appropriate. One advantage for the pLPA model over the pLSA model is that it can handle both explicit feedback in the form of ratings as well as implicit feedback such as click-through, etc. Many previous works have shown how pairwise preferences regarding item-pairs may be derived from implicit user feedbacks such as click-throughs[12]. The pLPA model can be naturally applied to analyze such pairwise preference data. In contrast, the pLSA model as well as most existing CF algorithms are not able to handle implicit feedback data which contains no numeric ratings.

The training time complexity of pLPA is higher than pLSA. The E-step of pLPA involves doing the computation in 19 on every pair of rated items by the same users, which requires $O(knm^2)$ time while for pLSA the E-step only takes $O(knm)$. The M-step of pLSA has closed form solutions while pLPA involves running numerical optimization to find the maximum likelihood parameters. At prediction time, both pLSA and pLPA can efficiently generate scores for all items in $O(mk)$ time.

EigenRank[14] is another ranking oriented algorithm for collaborative filtering. Unlike pLPA, it is memory based and have high complexity at prediction time. After determining the n nearest neighbors of a target user, it first needs to compare every pair of rated items by each neighbor to estimate the user's pairwise preferences on the set of m items and then applies a random walk model to obtain the item scores, which would requires $O(m^2)$ time complexity, much higher than the $O(mk)$ complexity of pLPA since $m \gg k$. This makes EigenRank impractical for applications that requires computing recommendation list in real time.

5. EXPERIMENTS

5.1 Data Sets

We evaluated the proposed model on two real world movie ratings data sets: EachMovie and Netflix. The EachMovie data set consists of about 2.8 million ratings made by more than 72 thousand users on 1628 movies. The Netflix data set contains over 100 million ratings from over 480 thousand users on around 18000 titles. The EachMovie data contains 6 rating classes from 1 to 6 while the Netflix ratings fall into 5 rating classes from 1 to 5. For the experiments in this work, we only used the ratings on the 1000 and 2000 most frequently rated movies in the EachMovie and Netflix datasets respectively.

5.2 Baseline Methods

We compared the proposed pLPA algorithm with 5 existing algorithms. For memory based methods we implemented the item based model using vector similarity (IVS)[22] and the user based model using Pearson Correlation Coefficient (UPCC)[2]. We also implemented the two model based methods in section 3 including matrix factorization (MF)[17] and the probabilistic latent semantic analysis using gaussian rating model (pLSA)[8]. We also compare pLPA with another recently proposed algorithm, EigenRank (ER)[14], which adapted the memory-based methods for ranking oriented collaborative filtering.

5.3 Evaluation Metric

Traditionally, collaborative filtering algorithms are evaluated by the accuracy of their predicted ratings. One commonly used performance metric for rating accuracy is the Mean Absolute Error (MAE):

$$\text{MAE} = \frac{\sum_{(u,i)\in\mathcal{R}} |r_{ui} - \hat{r}_{ui}|}{|\mathcal{R}|}$$

However, as we argued previously, ranking effectiveness is a more suitable criterion for evaluating CF algorithms. Therefore, we choose the Normalized Discounted Cumulative Gain (NDCG) [10] as the performance measure, which is widely used for evaluating ranked results in information retrieval when the documents are assigned graded rather than binary relevance judgements. To use it for collaborative filtering, we simply treat the ratings on the items assigned by users as graded relevance judgements.

The NDCG is evaluated over some number k of the top items on a ranked item list. Let Q be the set of users included in the test data and $R(u,p)$ be the rating assigned by u to the item at the p-th position on the ranked list produced for user u. The NDCG at the k-th rank position with respect to the set of users Q is:

$$\text{NDCG}(Q,k) = \frac{1}{|Q|}\sum_{u\in Q} Z_u \sum_{p=1}^{k} \frac{2^{R(u,p)}-1}{\log(1+p)}$$

where Z_u is a normalization factor calculated so that the NDCG of the optimal ranking has a value of 1. The value of NDCG ranges from 0 to 1 with a higher value indicates better ranking effectiveness. An attractive property of NDCG is that it is more sensitive to the ratings of the items at higher rank positions, which is consistent with requirements in real recommender systems where only few items can be recommended each time.

5.4 Evaluation Protocol

From both data sets, we randomly picked a set of 10600 users with the most ratings from both data sets. In order the better assess ranking effectiveness, we computed the variance of the observed ratings for each user and chose the 500 users with highest rating variances from both data sets to form the testing data since the NDCG on these data are more sensitive to item ordering. For the remaining 10100 users, we use 10000 as the training data and the other 100 for parameter tuning. For the 500 active users, we use 5-fold cross-validation to evaluate different algorithm's performance, where in each run a user's model is trained using 80% of his ratings and tested on the remaining 20%.

5.5 Results and Discussion

5.5.1 *Performance Comparison between pLPA and other algorithms*

The evaluation results on both EachMovie and Netflix data sets are shown in Table 2 (a) and (b) respectively, where all the performances are calculated via 5-fold cross validation. On both data sets, we vary the amount of training users from 2,000 to 10,000 to study the performances of different algorithms under different sparsity conditions. For each method, we measure the NDCG values at the 5th, 10th and 20th positions respectively. We also report the MAE for the 4 rating prediction algorithms.

Based on the results, we have made the following interesting observations:

1. For the 4 rating prediction algorithms, it is interesting to note that the two metrics MAE and NDCG often lead to inconsistent judgements on the algorithms. For example, on EachMovie 2000 users dataset, the item based method IVS, despite having worst MAE, have achieved best NDCG10 and NDCG20 scores out of all the rating prediction methods. Similarly, on the netflix datasets, while the MAE of UPCC and IVS are very close to each other, a much bigger difference in NDCG scores were observed between the two algorithms. These results provide support to our belief in designing and evaluating CF algorithms from the ranking perspective.

2. The two ranking oriented methods ER and pLPA generally outperform the other rating prediction methods in terms of NDCG scores. On the eachmovie dataset, pLPA achieves best NDCG scores at all different positions in most settings. On the netflix dataset, ER appeared to be highly effective at finding top items as evidenced by its high NDCG@5 scores. On the other hand, the pLPA method was more effective for longer ranked list as shown by its high NDCG@20 scores. Overall, the results indicate that the proposed pLPA model is very effective for ranking items in collaborative filtering.

5.5.2 *Efficiency Comparison between EigenRank and pLPA*

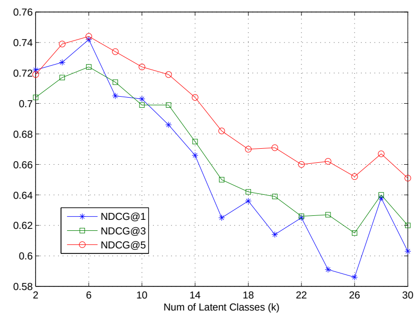
As noted in the analysis in section 4.4, the time complexity of the prediction operation using the EigenRank algorithm is much higher than that of pLPA. In this set of experiments, we measured the amount of time that it takes to generate rankings for all the 500 test users in both EachMovie and Netflix datasets under different parameter settings. The results are shown in Table 1. We have varied the number of nearest neighbors, denoted by n, in EigenRank from 100 to 400 and the number of latent classes, denoted by k, in pLPA model from 10 to 40, which are the typical ranges for n and k. We can clearly see that making predictions using pLPA is several orders of magnitude more efficient than using EigenRank.

5.5.3 *Effect of the Number of Latent Classes in pLPA*

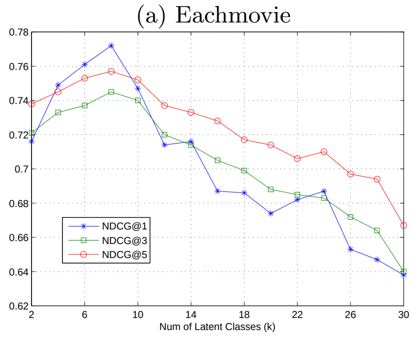
Figure 2 shows the ranking performances of pLPA on test data as a function of the number of latent classes when trained on the 5000 users datasets for EachMovie and Netflix. The graphs show that the optimal performances are obtained with k equals 6 and 8 on the EachMovie and Netflix data sets respectively after which the performances on test data tend to degrade, which indicates that the model suffers from overfitting when the number of parameters increases. To address over-fitting, one possible solution is to introduce hyper-prior distributions over model parameters for regularization as in the latent Dirichlet allocation model[1]. We will leave this for our future work.

	EachMovie	Netflix
ER(n=100)	223.8	426.8
ER(n=200)	289.3	573.5
ER(n=400)	345.9	806.2
pLPA(k=10)	0.4	0.9
pLPA(k=20)	1.1	2.1
pLPA(k=40)	2.0	5.3

Table 1: Prediction Time Comparison between EigenRank and pLPA (in seconds)



(a) Eachmovie



(b) Netflix

Figure 2: Performance of pLPA as a function of the number of latent classes

6. CONCLUSIONS AND FUTURE WORK

In this paper, we propose the probabilistic latent preference analysis model for ranking-oriented collaborative filtering. The pLPA model addresses the item ranking problem directly by modeling pairwise preferences on items using a latent class model based on the Bradley-Terry Model for paired comparisons. Experimental results show that our approach can effectively improve the ranking performance.

For future work, we would like to investigate several extensions of the current pLPA model. Firstly, in addition to modeling pairwise preferences, we would also consider probability models on listwise preferences such as the Plackett-Luce models for ranked data[3]. Thirdly, we will consider introducing hyper-prior distributions over model parameters for regularization. Finally, we also plan to test the pLPA

Table 2: Cross Validated Performances of Different Algorithms

	2000 Training Users				5000 Training Users				10000 Training Users			
	mae	ndcg5	ndcg10	ndcg20	mae	ndcg5	ndcg10	ndcg20	mae	ndcg5	ndcg10	ndcg20
UPCC	1.450	0.648	0.688	0.752	1.410	0.679	0.713	0.782	1.381	0.682	0.716	0.781
IVS	1.491	0.669	0.712	0.770	1.494	0.669	0.714	0.781	1.494	0.668	0.712	0.780
pLSA	1.261	0.688	0.700	0.749	1.222*	0.701	0.719	0.789	1.210	0.705	0.710	0.788
MF	1.233*	0.691*	0.708	0.750	1.279	0.703	0.725	0.794	1.187*	0.711	0.720	0.793
ER	–	0.675	0.696	0.762	–	0.704	0.727	0.792	–	0.713	0.723	0.797
pLPA	–	0.690	0.728*	0.792*	–	0.738*	0.748*	0.809*	–	0.743*	0.755*	0.815*

(a) Results on EachMovie Data Set (* indicates best performance)

	2000 Training Users				5000 Training Users				10000 Training Users			
	mae	ndcg5	ndcg10	ndcg20	mae	ndcg5	ndcg10	ndcg20	mae	ndcg5	ndcg10	ndcg20
UPCC	0.916	0.651	0.660	0.698	0.902	0.670	0.651	0.690	0.884	0.711	0.704	0.728
IVS	0.888*	0.621	0.625	0.681	0.890	0.627	0.631	0.686	0.891	0.628	0.634	0.689
pLSA	0.906	0.698	0.672	0.707	0.826	0.704	0.674	0.719	0.802	0.704	0.706	0.738
MF	0.891	0.705	0.675	0.708	0.812*	0.703	0.693	0.732	0.796*	0.703	0.711	0.745
ER	–	0.718*	0.689*	0.720	–	0.747*	0.704	0.730	–	0.760*	0.725	0.741
pLPA	–	0.706	0.684	0.728*	–	0.730	0.709*	0.741*	–	0.743	0.728*	0.759*

(b) Results on Netflix Data Set (* indicates best performance)

model on implicit user feedback data such as query logs from which pairwise preferences of items can be extracted.

7. REFERENCES

[1] D. M. Blei, A. Y. Ng, and M. I. Jordan. Latent dirichlet allocation. *Journal of Machine Learning Research*, 3:993–1022, 2003.

[2] J. S. Breese, D. Heckerman, and C. Kadie. Empirical analysis of predictive algorithms for collaborative filtering. In *Proceedings of UAI 1998*, 1998.

[3] Z. Cao and T. yan Liu. Learning to rank: From pairwise approach to listwise approach. In *In Proceedings of the 24th International Conference on Machine Learning*, pages 129–136, 2007.

[4] W. W. Cohen, R. E. Schapire, and Y. Singer. Learning to order things. *Journal of Artificial Intelligence Research*, 5:243–270, 1999.

[5] K. Y. Goldberg, T. Roeder, D. Gupta, and C. Perkins. Eigentaste: A constant time collaborative filtering algorithm. *Information Retrieval*, 4(2):133–151, 2001.

[6] J. Herlocker, J. A. Konstan, and J. Riedl. An empirical analysis of design choices in neighborhood-based collaborative filtering algorithms. *Information retrieval*, 5(4), 2002.

[7] T. Hofmann. Probabilistic latent semantic analysis. In *UAI*, pages 289–296, 1999.

[8] T. Hofmann. Latent semantic models for collaborative filtering. *ACM Trans. Inf. Syst.*, 22(1):89–115, 2004.

[9] D. R. Hunter. MM algorithms for generalized Bradley-Terry models. *Annals of Statistics*, 32(1):384–406, 2004.

[10] K. Järvelin and J. Kekäläinen. Cumulated gain-based evaluation of ir techniques. *ACM Trans. Inf. Syst.*, 20(4):422–446, 2002.

[11] R. Jin, L. Si, C. Zhai, and J. Callan. Collaborative filtering with decoupled models for preferences and ratings. In *Proceedings of CIKM 2003*, 2003.

[12] T. Joachims. Optimizing search engines using clickthrough data. In *Proceedings of SIGKDD 2002*, pages 133–142, 2002.

[13] G. Linden, B. Smith, and J. York. Amazon.com recommendations: Item-to-item collaborative filtering. *IEEE Internet Computing*, 7(1):76–80, 2003.

[14] N. N. Liu and Q. Yang. Eigenrank: a ranking-oriented approach to collaborative filtering. In *SIGIR*, pages 83–90, 2008.

[15] H. Ma, I. King, and M. R. Lyu. Effective missing data prediction for collaborative filtering. In *Proc. of SIGIR 2007*, pages 39–46, 2007.

[16] J. I. Marden. *Analyzing and Modeling Rank Data.* Chapman & Hall, New York, 1995.

[17] A. Paterek. Improving regularized singular value decomposition for collaborative filtering. *Proceedings of KDD Cup and Workshop*, 2007.

[18] D. M. Pennock, E. Horvitz, S. Lawrence, and C. L. Giles. Collaborative filtering by personality diagnosis: A hybrid memory and model-based approach. In *Proc. of UAI*, pages 473–480, 2000.

[19] F. Radlinski and T. Joachims. Query chains: learning to rank from implicit feedback. In *KDD*, pages 239–248, 2005.

[20] M. E. Renda and U. Straccia. Web metasearch: Rank vs. score based rank aggregation methods. In *Proceedings of the 2003 ACM Symposium on Applied Computing (SAC)*, pages 841–846, 2003.

[21] P. Resnick, N. Iacovou, M. Suchak, P. Bergstrom, and J. Riedl. Grouplens: An open architecture for collaborative filtering of netnews. In *Proc. of ACM Conference on Computer Supported Cooperative Work*, pages 175–186, 1994.

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

[23] N. Srebro and T. Jaakkola. Weighted low-rank approximations. In *ICML*, pages 720–727, 2003.

[24] 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 *Proc. of SIGIR 2005*, pages 114–121, 2005.