Improving Latent Factor Models via Personalized Feature Projection for One Class Recommendation

Tong Zhao[1,2], Julian McAuley[3], Irwin King[1,2]
[1]Shenzhen Key Laboratory of Rich Media Big Data Analytics and Applications, Shenzhen Research Institute, The Chinese University of Hong Kong, Shenzhen, China
[2]Department of Computer Science and Engineering, The Chinese University of Hong Kong, Shatin, N.T., Hong Kong
[3]Department of Computer Science and Engineering, UC San Diego, La Jolla, CA, USA
{tzhao, king}@cse.cuhk.edu.hk, julian.mcauley@gmail.com

ABSTRACT

Latent Factor models, which transform both users and items into the same latent feature space, are one of the most successful and ubiquitous models in recommender systems. Most existing models in this paradigm define both users' and items' latent factors to be of the same size and use an inner product to represent a user's 'compatibility' with an item. Intuitively, users' factors encode 'preferences' while item factors encode 'properties', so that the inner product encodes how well an item matches a user's preferences. However, a user's opinion of an item may be more complex, for example each dimension of each user's opinion may depend on a combination of multiple item factors simultaneously. Thus it may be better to view each dimension of a user's preference as a personalized projection of an item's properties so that the preference model can capture complex relationships between items' properties and users' preferences.

Therefore, in this paper we propose a novel *personalized feature projection* method to model users' preferences over items. Specifically, for each user, we define a personalized projection matrix, which takes the place of user-specific factors from existing models. This matrix describes a mapping between items' factors and users' preferences in order to build personalized preference models for each user and item. The proposed personalized feature projection method is quite general and existing latent factor models, for example, can be cast as a special case. We present three objective functions to optimize predictions in the form of ranked lists of users' preferences over items, and demonstrate how each can be used to improve one-class recommendation performance. Experiments are conducted on four real-world datasets and our results show that our personalized feature projection method outperforms several state-of-the-art methods on various evaluation metrics.

Categories and Subject Descriptors

H.4 [**Information Systems Applications**]: Miscellaneous

CIKM'15, October 19–23, 2015, Melbourne, Australia.

DOI: http://dx.doi.org/10.1145/2806416.2806511.

General Terms

Algorithms, Experimentation, Performance

Keywords

Collaborative filtering; Personalized feature projection; One-class recommendation

1. INTRODUCTION

Modeling people's opinions and identifying which items are relevant to each person plays a critical role in the online marketplace, and is the basic task of a *recommender system*. A major and ongoing thrust of research on recommender systems is concerned with improving the performance of personalized recommendation. For example, a significant amount of research has been proposed that aims to improve the performance of Latent Factor (LF) models [10, 27, 28, 29], one of the most successful and ubiquitous approaches in recommender systems. Latent factor models assume that each user and each item is associated with some K-dimensional latent factor vector, such that the inner product of a user's and an item's latent factors represents the preference score or compatability between the user and the item.

Given a user-item matrix consisting of the rating scores that users give to items, latent factor models essentially consist of identifying the low-dimensional structure in this matrix. The goal is to identify K-dimensional embeddings of users and items such that their product best recovers this rating matrix in terms of the mean-squared error (MSE). Besides fitting rating values, some works [8, 32, 37, 39] also adapt latent factor models to optimize a ranking loss with the intuition that higher scoring user-item pairs should have larger inner products than lower-scoring items for the given user, e.g., for a user u, their inner product with a 5-star item should be higher than that of a 2-star item, though only the *relative* rather than the exact ratings are modeled.

Another popular research direction using latent factor models is so-called one-class recommendation, which is the focus of this paper. In many real applications, explicit numerical ratings might not be available and one must instead try to model some form of implicit feedback from users, such as the media they consume, the pages they browse, the music they listen to, or whom they befriend [6, 42]. This setting is called "one-class" recommendation and a variety of solutions have been proposed to solve it by directly modeling relative preferences, or *rankings*, of items for personalized recommendation [13, 19, 26, 43]. However, most existing works are still using an inner product of the latent factor vectors to predict users' preferences on items, which essentially assumes that each

dimension of a user's preferences is associated with one and only one of an item's factors.

In this paper, we generalize this linear structure and assign a projection matrix to each user instead of a latent factor vector. Intuitively, traditional latent factor models can be interpreted as assuming that a user's latent factors represent how the user thinks about the latent features of an item and more restrictively, that each dimension of their opinion is related to a single latent factor of a product. However, a more expressive model would allow each dimension of a user's opinion to be a function of a *combination* of an item's latent factors. Therefore, it is better to learn a projection between user preferences and item properties so that the model is capable of expressing complex relationships between the two. Moreover since in latent factor models, the inner product of a user's and an item's latent feature vectors is always used to represent the user's preference toward the item, a user's preference on an item is modeled only by a real number, a one-dimensional value, and thus only numerical preference modeling can be used to design objective functions. As we argued above, each dimension of a user's preference is related to a *combination* of their tastes toward an item's properties, so a simple inner product may not be enough. To address the drawbacks mentioned above, we propose a *personalized feature projection* (PFP) method that learns users' latent features as a personalized projection matrix instead of a vector. Figure 1 depicts the idea of our PFP method. Based on items' latent feature vectors and users' projection matrices, our PFP method models users' preferences over items in terms of projected latent feature vectors instead of a real number. Thus, we can leverage metric learning methods to design the objective function for one-class recommendation. In addition, vector-based (rather than numerical) objectives, can be formulated, which provides more flexible structures to describe users' preferences.

The most similar work to our PFP method is the method called *max interest latent factors* [39], which also proposes to model multiple user latent vectors to enrich users' preference representations. In [39], the predicted preference between a user and an item is given by the *maximum* match between the user vectors and the item vector, and omits other tastes of the user. Different from their work, when modeling user's preference on items, the proposed PFP method tries to consider all tastes of the user simultaneously by employing a matrix projection approach.

We summarize our contributions as follows:

1. We develop Personalized Feature Projection methods, that employ users' projection matrices and items' factors to solve one-class recommendation problems using three different optimization criteria.

2. We evaluate the proposed method on four real-world datasets. Empirical results show that the proposed model can significantly improve one-class recommendation performance compared to state-of-the-art alternatives.

3. We conduct a detailed analysis of the effect that various model parameters have on the performance of our method.

The remainder of this paper is organized as follows. Section 2 describes preliminary and related work; Section 3 details the proposed PFP methods and introduces three criteria used to optimize them; Section 4 shows experimental results and verifies the improvements of PFP methods for one-class recommendation problems; Finally, Section 5 concludes the paper.

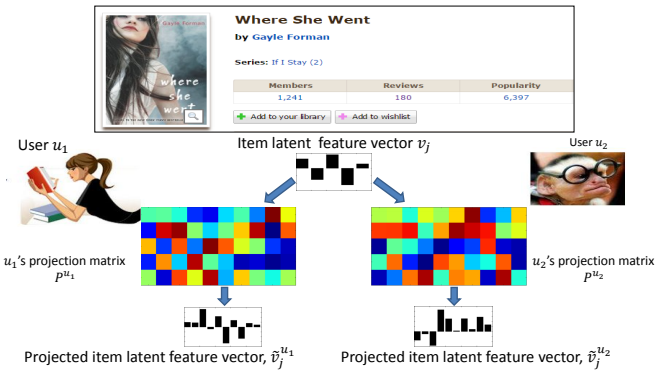


Figure 1: Illustration of the key idea behind the PFP method. Unlike traditional latent factor models, we use a more flexible transform that associates each item's latent factor with a combination of users' latent factors. The above figure is based on a model trained on our *LibraryThing* dataset. Here we set the number of item latent factors to 5 and fit user projection matrices of size 5×10. In this case, the preferences of different users toward the item can be represented by a set of 10-dimensional personalized projection vectors.

2. PRELIMINARY AND RELATED WORK

We consider the *one-class recommendation problem*, where the model is trained on implicit feedback (e.g., a set of items a user purchased), rather than explicit rating scores. Before presenting our *Personalized Feature Projection* method, we first give an overview of existing latent factor models and describe how they can be applied to solve one-class recommendation problems.

2.1 Latent Factor Models for One-class Recommendation

Latent Factor models can be described in terms of matrix factorization. Matrix factorization aims to decompose a matrix into two (or more) low-rank terms whose product reconstructs or approximates the original matrix as closely as possible. Typically, we are given a matrix $R \in \mathbb{R}^{M \times N}$ representing the observed feedback (say, in the form of numeric ratings) from M users and N items.

This feedback matrix can then be factorized into one user-specific matrix $U \in \mathbb{R}^{M \times K}$ and one item-specific matrix $V \in \mathbb{R}^{N \times K}$:

$$R \approx UV^T, \tag{1}$$

where K is the dimensionality of the latent factor vector which characterizes the features of a user or an item. Accordingly, $U_u V_i^T$ captures user u's preference toward item i.

For 'one-class' recommendation problems, there is no numeric feedback matrix R, but rather for each user we observe two signals in the form of *positive feedback* and *negative feedback*:

Positive feedback: positive feedback $D_u = \{i\}$ is defined as the set of items i toward which user u explicitly shows positive feedback (e.g., products they purchased, pages they viewed, etc.).

Negative feedback: negative feedback is denoted by a set of user-item pairs $N_u = \{j\}$, which includes those items j toward which user u has not shown positive feedback.

Then, in lieu of an explicit rating score, we follow the assumption of [24] that users' preferences toward positive feedback items should be higher than those of negative feedback items, which is

captured by the following objective function:

$$\max \sum_{u,i,j} \ln \sigma(u_u \cdot v_i - u_u \cdot v_j), \quad (2)$$

where $\sigma(\cdot)$ is the logistic (sigmoid) function, $i \in D_u$, and $j \in N_u$. Here $u_u \cdot v_i$ and $u_u \cdot v_j$ represent user u's preferences towards items i and j modeled by a latent factor model. Eq. (2) can be optimized by using stochastic gradient ascent, which iteratively updates the sampled user-specific latent factors U_u and the sampled positive (and negative) feedback-specific latent factors V_i and V_j.

2.2 Existing Methods for One-Class Recommendation

Several works have studied one-class collaborative filtering and can be mainly divided into two branches: pointwise methods [7, 19, 21] and pairwise methods [22, 24, 25, 26, 36, 40, 43].

Pointwise methods aim to fit a numeric value associated with each evaluated item. These methods model positive feedback as high preference scores and use several strategies to sample negative feedback as low preference scores. Then existing matrix factorization methods can be used to fit the preference scores. Pan et al. [19] solve the one-class recommendation problem in two ways: negative example *weighting* and negative example *sampling*. Hu et al. [7] introduced a novel concept called a "confidence level," associated with positive and negative feedback, and propose an efficient optimization method for confidence-based matrix factorization.

Different from pointwise methods, pairwise methods focus on modeling the order, or ranking of the feedback. Pairwise methods always consider implicit feedback as relative relationships indicating that users show higher preference on positive feedback than on negative feedback items. In [24], Rendle et al. propose a Bayesian personalized ranking (*BPR*) framework. Following this, various ideas have been proposed that incorporate different types of contextual information into the *BPR* framework. [13] extends the *BPR* framework to model both users' feedback on items and on their social relations. In [26], Rendle et al. extend the *BPR* framework from matrix factorization to tensor factorization for tag recommendation. Pan et al. [20] aggregate the features of a group of related users to reduce the uncertainty of the selected training instances. Du et al. [3] improve one-class recommendation performance by incorporating a social regularization term into the *BPR* framework. Zhao et al. [45] leverage social connections to improve the performance of one-class recommendation. In [8], Kabbur et al. use an item-based method for generating top-N recommendations that learns the item-item similarity matrix as the product of two low dimensional latent factor matrices. All of these existing pairwise methods for one-class recommendation use a numerical value to model user preferences, while our model generalizes this setting and describes each user's preferences toward items as a personalized projection vector.

2.3 Metric Learning

Since the method proposed in the following section also relates to metric learning problems with pairwise constraints [2, 11, 12, 23, 31, 35, 41], here we briefly review existing works in this field. Metric learning, whose goal is to model the similarity/dissimilarity between instances, is one of the most fundamental problems of machine learning and has been widely used for clustering and classification. [33] learns a Mahalanobis metric to study the semantic similarity between music songs. In [23], the authors propose a metric learning algorithm to improve the prediction performance for multi-task learning by learning one metric that is shared amongst all the tasks and one specific metric unique to each task. Kusner et al. [15] uses distance metric learning to improve the efficiency and accuracy of k-nearest neighbor (kNN) classification. Yue et al. [44] proposes a personalized collaborative clustering algorithm that leverages the idea of metric learning to investigate how existing users cluster or group items in order to predict similarity models for other users' clustering tasks. Lee et al. [17] assumes the observed user-item feedback matrix is locally low-rank within certain neighborhoods of the metric space defined by some anchor (user, item) pairs and employs the Epanechnikov kernel function to measure the local metric. However, most of the existing work on metric learning problems focus on solving clustering-related problems and few works consider the issue of personalized one-class recommendation.

Table 1: Notation.

Symbol	Discription
M	number of users
N	number of items
K	number of latent factors
K^*	number of projected user latent factors
$C_u, C_i \in \mathbb{R}$	projection constraint in PFP model
$\{\alpha_i\}$	parameters used in WARP-loss function
$U \in \mathbb{R}^{M \times K}$	user latent factor matrix in LF models
$V \in \mathbb{R}^{N \times K}$	item latent factor matrix
$P^u \in \mathbb{R}^{K \times K^*}$	projection matrix of user u in PFP model
$u_u \in \mathbb{R}^K$	u_{th} row vector in U
$v_j \in \mathbb{R}^K$	j_{th} row vector in V
$p_f^u \in \mathbb{R}^K$	f_{th} column vector in P^u

2.4 Non-linear Latent Factorization

Another line of latent factor models are called non-linear latent factor models [16, 30, 39], which break the mold of scoring each item by a linear embedding of users' and items' latent factor vectors. The authors of [16] proposed a nonlinear LF approach with Gaussian processes. The authors of [30] proposed a Restricted Boltzmann machine-based method for collaborative filtering. Baltrunas et al. [1] splits user profiles into several sub-profiles and models each sub-profile in a particular context. In [39], the authors propose a *MaxMF* method that models a user with multiple latent factor vectors representing their different latent tastes and associate each item with a single latent feature vector. The preference score between a user and a given item is then defined as the maximum match between the user tastes and the given item. This work is the most similar one compared to our method. However, there are explicit differences between their method and ours: our proposed PFP method utilizes all latent factor vectors of the user and models the user's preference on the item as a projected vector rather than the maximum inner product in [39].

3. PERSONALIZED FEATURE PROJECTION

We first introduce our *Personalized Feature Projection* (PFP) method. The proposed method inherits the benefits of both metric learning and latent factor models. Then, in Section 3.2 we show how this method can be trained to optimize three different evaluation criteria. Our notation is shown in Table 1.

3.1 Personalized Feature Projection and Preference Modeling

In the PFP method, for each item j, we use a vector v_j to represent its latent features, much as is done using existing latent factor models. However, for each user u, we employ a matrix P^u to represent their latent features. Specifically, let K be the number of latent factors (for each item) and let K^* be the number of *projected* user factors. Then we have the following definition:

Personalized Projection Matrix : for each user u, we define a personalized projection matrix $P^u = \mathbb{R}^{K \times K^*}$. The column vector $p_f^u = \mathbb{R}^K$ represents column f in matrix P^u.

Based on the Personalized Projection Matrix, we can use the following formula to personalize items' latent feature vectors and obtain the projected item-feature vector.

Personalized Feature Projection : for a specific user u, item j's feature vector v_j can be projected by multiplying u's personalized projection matrix. In this way, the projected feature vector of item j, $\widetilde{v}_j^u$ ($\widetilde{v}_j$ for simplicity), can be represented as

$$\widetilde{v}_j = v_j P^u. \tag{3}$$

One can easily check that, when $K^* = 1$, the personalized projection matrix for each user reduces to a latent feature vector of size K, which is equivalent to the user latent feature vector in a latent factor model so that the projected item-feature vector reduces to the dot product of users' and items' latent factors. In this way, we can say that the latent factor model is a special case of the personalized feature projection method. When $K^* > 1$, the projected item-feature vector provides a more flexible way to model users' complex preferences.

Preference Modeling : Based on the personalized projection matrix, we propose a factorized model to describe users' preferences by summarizing all the projected feature vectors from their positive feedback. In particular, for an item j, we consider the preference model as

$$f_u(j) = \frac{1}{|D_u|} \sum_{i \in D_u} \widetilde{v}_i \widetilde{v}_j^T = \frac{1}{|D_u|} \sum_{i \in D_u} v_i P^u (P^u)^T v_j^T, \tag{4}$$

where D_u represents user u's positive feedback. Actually, if we normalize both $\widetilde{v}_i$ and $\widetilde{v}_j$ to be unit vectors, the above equation represents the average cosine similarity of the projected feature vectors between item j and the user's positive feedback items. Compared with [8] and [10], instead of using only item latent vectors, this formula incorporates user's personalized projection matrix into preference modeling, thus, for different users, Eq. (4) can model his/her personalized preference on items. Based on Eq. (4), we propose an assumption to derive the optimization criterion for personalized ranking that is based on pairs of items: for an item i that has been viewed by user u, the preference score of it calculated by Eq. (4) should be larger than other unviewed items j, $f_u(i) \succ f_u(j)$. Actually, the proposed assumption indicates that the projected feature vectors of users' positive feedback items should be closer to users' average taste than are the negative feedback items. Similar assumptions are commonly used in various pairwise-based recommendation methods ([24], [26], [20], [13]). However, unlike existing works, the personalized feature projection method models users' preferences toward items as a projected vector rather than as an inner product, which provides flexible structure for preference representation. Another advantage of our approach is that it summarizes all the projected vectors from users' positive feedback items to reduce the uncertainty caused by training instance selection. More specifically, when we uniformly sample a positive feedback item from a user, the average similarity of Eq. (4) makes the approach insensitive to the choice of which positive feedback items should be selected. $P^u(P^u)^T$ here can be viewed as the personalized metric used to measure the preference difference between items.

3.2 PFP for One-class Recommendation

According to the pairwise assumption proposed above, we will introduce how the PFP model can be trained to optimize three different evaluation criteria to solve the one-class recommendation problem.

3.2.1 AUC-loss

In this section, we choose the area under the ROC curve (AUC) as the ranking statistic and explain how to extend the Bayesian Personalized Ranking [24] framework to incorporate our personalized feature projection matrix for recommendation.

The AUC per user is usually defined as

$$AUC(u) = \frac{1}{|D_u||N_u|} \sum_{i \in D_u} \sum_{j \in N_u} I(f_u(i) > f_u(j)), \tag{5}$$

where $I(\cdot)$ is an indicator function. Following [24], maximizing the AUC value can be approximated by maximizing the following likelihood function:

$$\ln \prod_u AUC(u) = \ln \prod_{(u,i,j) \in U \times I \times I} P(f_u(i) > f_u(j))^{i \in D_u \wedge j \in N_u} (1 - P(f_u(i) > f_u(j)))^{j \in D_u \wedge i \in N_u}. \tag{6}$$

Due to the totality and antisymmetry of a pair-wise ordering scheme as in [24], the objective function for a particular user u can be simplified to

$$AUC(u) = \frac{1}{|D_u||N_u|} \sum_{i \in D_u} \sum_{j \in N_u} P(f_u(i) > f_u(j)). \tag{7}$$

By incorporating Eq. (4), the above objective can be rewritten as:

$$\begin{aligned} AUC(u) &= \frac{1}{|D_u||N_u|} \sum_{i \in D_u} \sum_{j \in N_u} \ln \sigma(f_u(i) - f_u(j)) \\ &= \frac{1}{|D_u||N_u|} \sum_{i \in D_u} \sum_{j \in N_u} \ln \sigma(\frac{1}{|D_u|} \sum_{i' \in D_u} \widetilde{v}_{i'} (\widetilde{v}_i - \widetilde{v}_j)^T), \end{aligned} \tag{8}$$

where σ is the logistic (sigmoid) function: $\sigma(x) = \frac{1}{1+e^{-x}}$. We use the gradient-based algorithm to optimize the objective function as Eq. (8). The main process is to randomly pick a (positive, negative) feedback pair and iteratively update parameters based on the sampled feedback pairs.

3.2.2 WARP-Loss

In this subsection, we describe how to adapt our projected feature representation to design a WARP-Loss (Weighted Approximated Ranking Pairwise Loss) function.

Based on Eq. (4), we can calculate users' preferences for each item $j \in N$ and the *Weighted Approximate-Rank Pairwise* (WARP) Loss function [38, 39, 40] can be defined as

$$L_{WARP}(f_u, D_u) = \sum_{i \in D_u} \Theta(rank_i(f_u)), \tag{9}$$

where $rank_i(f_u)$ is the margin-based rank of item i, i.e.,

$$rank_i(f_u) = \sum_{j \notin D_u} I(f_u(j) \geq f_u(i)), \tag{10}$$

where $I(.)$ is the indicator function. $\Theta(.)$ is a loss function, which transforms the predicted rank of an item into a loss value,

$$\Theta(k) = \sum_{t=1}^{k} \alpha_t, \alpha_1 \geq \alpha_2 \geq \cdots \geq 0. \tag{11}$$

Different settings of α_t allow the loss function to optimize different objectives. Minimizing Θ with $\alpha_t = \frac{1}{N}$ would optimize the mean rank and minimizing Θ with $\alpha_t > \alpha_{t+1}$ would assign higher importance to the top-ranked items.

In this paper, we use a hinge loss to replace the indicator function so that a margin can be added and stochastic gradient descent (SGD) can be adopted to learn the parameters. Thus, the loss function for a specific item $i \in D_u$ in Eq. (9) can be approximated by Eq. (12) as

$$\begin{aligned} \overline{\Theta}(rank_i(f_u)) &= \sum_{j \notin D_u} \Theta(rank_i(f_u)) \frac{I(f_u(j) \geq f_u(i))}{rank_i(f_u)} \\ &\approx \sum_{j \notin D_u} \Theta(rank_i(f_u)) \frac{|1 - f_u(i) + f_u(j)|_+}{rank_i(f_u)}, \end{aligned} \tag{12}$$

where $|q|_+$ represents the positive part of q.

To minimize Eq. (12), we first uniformly sample a positive training instance $i \in D_u$. Then we need to calculate the margin rank of the item i among all items. However, exactly computing the rank of the item i is highly time-consuming when the number of items is very large. Therefore, following [38], at each iteration, we uniformly sample a negative feedback instance $j \notin D_u$ until a pairwise violation is found, that is, until $1 - f_u(i) + f_u(j) > 0$. If Q steps are required to find such a pairwise violation, then the rank of item i can be approximated as

$$rank_i(f_u) \approx \lfloor \frac{N-1}{Q} \rfloor. \tag{13}$$

Now the loss from the chosen item pair $\{f_u(.), i, j\}$ becomes

$$L_{WARP}(f_u, i, j) = \Theta(\lfloor \frac{N-1}{Q} \rfloor) |1 - f_u(i) + f_u(j)|_+. \tag{14}$$

To optimize Eq. (14), we can simply use gradient descent to perform updates, i.e.,

$$\theta_{t+1} = \theta_t - \eta \frac{\partial L_{WARP}(f_u, i, j)}{\partial \theta_t}, \tag{15}$$

where η is the learning rate.

3.2.3 Vector-based Objective Function

Since most existing latent factor models use the inner product of users' and items' latent factors to represent a user's preference or compatibility toward an item, only numerical loss functions can be used to optimize the ranking objective. However, only a one-dimensional number cannot precisely describe user's complex preference on the item, and moreover, numerical loss functions seem to be too arbitrary when comparing users' preferences on different items. Therefore, we propose to model users' preferences on items via a projected vector and propose a method that measures users' preference differences on items in vector space. Here, we describe how to model users' preferences by directly using the projected vectors. Recalling our *AUC* and *WARP*-Loss objective functions, we find that the basic idea behind them is to let the projected feature vectors of positive instances be 'more similar' to users' average preferences than are negative instances. Therefore, we can follow this intuition and directly model it via vector similarity. Actually, there are various proposed similarity measures that are applicable to compare two vectors to solve classification, clustering and retrieval problems. For example, Sørensen distance [34] uses the L_1 norm to measure similarity, *Wave Hedges* [5] uses an intersection-based measure to model similarity, and the *KL-divergence* [14] measures the similarity based on Shannon's concept of probabilistic uncertainty. In this work, we build our objective function based on the *KL-divergence*, which is a popular similarity measure in machine learning and data mining. The *KL-divergence* between two vectors can be represented as

$$d_{KL}(T, W) = \sum_{q=1}^{K^*} W(q) \ln \frac{W(q)}{T(q)}, \tag{16}$$

where T and W are vectors with size K^* and $W(q)$ represents the q^{th} entry in vector W.

Based on Eq. (16), for a particular user u, our goal is to maximize the *KL-divergence* between the projected vectors from users' positive and negative instances. More specifically, when given a positive instance $i \in D_u$ and a negative instance $j \in N_u$, the objective function can be defined in terms of maximizing

$$\begin{aligned} \mathcal{O}_u(i, j) &= \sum_{f}^{K^*} (\widetilde{v}_j(f) \ln \frac{\widetilde{v}_j(f)}{\widetilde{v}_i(f)}) \\ &= \sum_{f}^{K^*} ((v_j P^u)(f) \ln \frac{(v_j P^u)(f)}{(v_i P^u)(f)}) \\ &= \sum_{f}^{K^*} ((v_j p_f^u) \ln \frac{v_j p_f^u}{v_i p_f^u}). \end{aligned} \tag{17}$$

To maximize Eq. (17), we uniformly sample a positive instance $i \in D_u$ and a negative instance $j \in N_u$. Then we can simply perform updates via gradient ascent. This process is repeated until the model converges. Specifically, the detailed gradients of the corresponding latent variables in the matrix factorization are as follows:

$$\begin{aligned} \frac{\partial \mathcal{O}_u(i,j)}{\partial P_{kf}^u} &= v_{jk}(1 + \ln \frac{v_j p_f^u}{v_i p_f^u}) - v_{ik} \frac{v_j p_f^u}{v_i p_f^u}, \\ \frac{\partial \mathcal{O}_u(i,j)}{\partial v_{ik}} &= \sum_{f}^{K^*} [-\frac{v_j p_f^u}{v_i p_f^u} p_{kf}^u], \\ \frac{\partial \mathcal{O}_u(i,j)}{\partial v_{jk}} &= \sum_{f}^{K^*} [(\ln \frac{v_j p_f^u}{v_i p_f^u} + v_i p_f^u) p_{kf}^u], \end{aligned} \tag{18}$$

where P_{kf}^u and p_f^u are as defined above. To avoid overfitting, we also constrain the parameters using the following inequalities and project the parameters back on the constraints when they violate the constraints at each step:

$$\begin{aligned} &\|P^u\|_2 \leq C_u, u \in 1, 2, ..., M, \\ &\|v_j\|_2 \leq C_i, j \in 1, 2, ..., N, \end{aligned} \tag{19}$$

where C_u and C_i are called projection constraint terms.

We note that although KL-divergence is asymmetric, due to the experimental results, this asymmetry has little impact on model performance. Thus, we just define the objective function as Eq. 17. Our proposed personalized feature projection method is quite general and can be easily extended to other vector similarity measures

Table 2: Statistics of the Datasets.

	Ciao	BeerAdvocate	Ratebeer	Lthing
#Users	1,705	33,387	29,264	73,882
#Item	12,252	63,324	107,381	337,561
#Feedback	22,839	1,424,957	2,578,594	979,053
#Avg. feedback	13	42.67	88.11	13.25
Sparsity	0.11%	0.06%	0.082%	0.004%

to define new objective functions. We do not discuss other similarity measures further, since the focus of this paper is to illustrate how to we can model users' preferences as a personalized projected vector rather than as an inner product.

3.3 Complexity Analysis

The computational complexity for *PFP* (KL) method is $\mathcal{O}(M \cdot N \cdot K \cdot K^*)$. Since we apply stochastic gradient descent methods for optimization, the complexity for each iteration is $\mathcal{O}(K \cdot K^*)$. One solution to scale this algorithm is to alternatively optimize the parameters based on multi-threading and parallelization: fist optimize item-latent factor vectors, then train users' projection matrices. In fact, our proposed method has the same complexity as *MaxMF* [39], therefore, we could use similar MapReduce algorithms to train all user parameters in parallel: when a training triple is sampled, the mapper collects and emits each column vector of the user's projection matrix and the item's latent vector, and the reducer calculates the dot product of each column vector and the item's vector independently. The reducer would then emit the results to build up a final projection vector for training.

4. EXPERIMENTAL RESULTS

In this section, we conduct experiments on the real-world datasets to evaluate the effectiveness of the proposed methods.

4.1 Experimental Setup

4.1.1 Datasets and Evaluation Metrics

The data sets [45] used in this paper are collected from four popular web sites: Ciao, BeerAdvocate, Ratebeer and LibraryThing (Lthing). Statistics of the four datasets are summarized in Table 2. Since this paper focuses on solving the one-class recommendation problem, we filter out explicit negative feedback (rating scores below 4 out of 5 stars) and use the remaining instances as positive feedback for model learning. All are available online.[1] [2]

For each dataset, we randomly split it into a training part, used for model training, and a test part, used for model evaluation. For each user u, we randomly select 90% of their observed feedback items as D_u and leave the remainder as T_u for testing. Grid search is applied to find projection constraint terms and learning rate is set as 0.01.

Our experiments are intended to address the following questions:

1. How does our approach compare with related personalized ranking methods for item recommendation?
2. Can vector-based recommendation models outperform traditional numerical models?
3. How does the number of projected factors affect the results?

[1] http://www.public.asu.edu/jtang20/datasetcode/truststudy.htm
[2] http://cseweb.ucsd.edu/~jmcauley/

We use two popular metrics, *NDCG* (Normalized Discounted Cumulative Gain) and *Area Under the Curve* (AUC), to measure the recommendation quality of our proposed approach in comparison to baseline methods. The average AUC statistic is defined as

$$AUC = \frac{1}{M} \sum_{u \in M} \frac{1}{|E(u)|} \sum_{(i,j) \in E(u)} \delta(x_{ui} > x_{uj}), \qquad (20)$$

where $E(u) = \{(i,j) | (u,i) \in T_u \wedge (u,j) \notin (D_u \cup T_u)\}$.

$DCG@X$ considers the ranking of the recommended items by discounting the importance and is defined as

$$DCG@X = \sum_{i=1}^{X} \frac{2^{rel_i} - 1}{\log_2(i+1)}, \qquad (21)$$

where rel_i represents the relevance score of the item i (we use a binary value for this quantity). $NDCG$ is the ratio of the DCG value to the ideal DCG value for that user. The ideal value of DCG comes from the best ranking function for the user. Here we set $X = N - D_u$ where N is the total number of items.

4.1.2 Comparison Methods

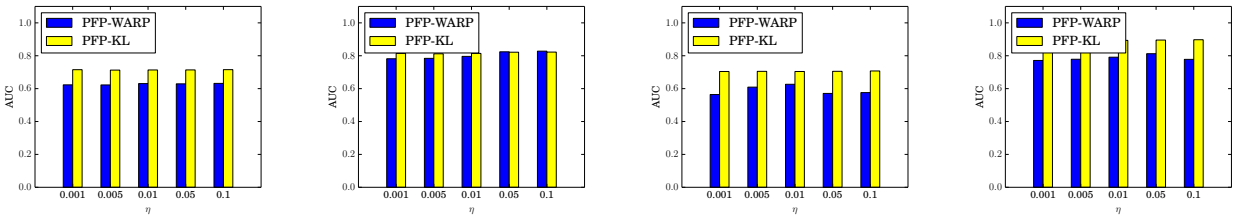
In order to demonstrate the benefits of our approach, we compare our model with the following methods for item recommendation. Since the problem we solve in this paper is one-class recommendation (without rating scores), it is unsuitable to compare our methods with rating estimation methods. Instead, we consider some state-of-the-art one-class recommendation methods as baselines.

- **CofiRank:** This method [37] is an extension of Maximum Margin Matrix Factorization for item recommendation optimized by a soft hinge ranking loss.
- **WRMF:** The weighted matrix factorization method is proposed by [19], which uses a point-wise strategy for solving one-class recommendation problems.
- **BPR-MF:** This method [24] proposes a pairwise assumption for item ranking and it is a well-known method for one-class recommendation.
- **GBPR:** This work [20] relaxes *BPR*'s assumption to a group pairwise preference assumption in order to smooth the individual positive feedback model and increase the confidence for pairwise classification. Here we fix the number of grouped users to 5 by cross-validation.
- **FISM-AUC:** This work [8] uses an item-based method for generating top-N recommendations. Since our work solves the one-class recommendation problem, here we choose their AUC-based loss function for comparison.
- **MaxMF:** This method is proposed in [39]. It is a non-linear matrix factorization method we introduced in Section 2.4. Note that *MaxMF* has the same number of parameters as PFP methods in all experiments.
- **PFP (WARP):** This method is proposed in Section 3.2.2 and uses the WARP-loss as an objective function.
- **PFP (KL):** This method is proposed in Section 3.2.3 and the KL-diversity is used for optimization.

Most of the above baseline methods can be found in [4].

Table 3: Recommendation performance of different methods on four real-world datasets. The last column shows the improvement of the proposed method compared with the best baseline method.

Dataset	Metrics	CofiRank	WRMF	BPR	GBPR	FISM-AUC	MaxMF	PFP-WARP	PFP-KL	Improv.
Ciao	NDCG	0.1692	**0.1881**	0.1672	0.1683	0.1570	0.1589	0.1865	0.1831	-0.85%
	AUC	0.6079	0.6948	0.5894	0.5909	0.5446	0.5669	0.6587	**0.7212**	+3.77%
Lthing	NDCG	0.1255	0.1665	0.1220	0.1322	0.1107	0.1132	0.1689	**0.1774**	+6.54%
	AUC	0.6035	0.7255	0.5816	0.6904	0.5926	0.6244	0.6587	**0.7880**	+8.61%
BeerAdvocate	NDCG	0.2180	0.2207	0.2390	0.2449	0.1888	0.1874	0.2354	**0.2501**	+2.12%
	AUC	0.8544	0.8523	0.8389	0.8649	0.7523	0.7812	0.8589	**0.8893**	+2.82%
RateBeer	NDCG	0.2600	0.2708	0.2715	0.2801	0.1728	0.2123	0.2614	**0.2877**	+2.71%
	AUC	0.8879	0.8758	0.8785	0.8881	0.7734	0.8120	0.8364	**0.9048**	+1.88%



(a) Ciao (b) BeerAdvocate (c) Lthing (d) Ratebeer

Figure 2: Impact of learning rate η.

4.2 Performance Comparison

Results in terms of the AUC and NDCG are shown in Table 3. Although we can assign different values to the number of latent factors and the number of projected factors, here we just report the results by setting the same value for both parameters and leave the discussion of different parameter settings to Section 4.3.

On most datasets, we find that PFP methods outperform baselines. In particular, three observations can be drawn from the results: First, the proposed PFP methods achieve the best performance in most cases; from Table 2, we observe that users in these two datasets have more positive feedback items compared to other datasets (the average RateBeer user reviews close to *ninety* beers in their lifetime as a reviewer). These results corroborate that PFP can better describe users' preferences when we have sufficiently many positive feedback items.

Second, compared to BPR and GBPR, where the AUC is optimized via maximizing the difference of two inner products, PFP (KL) succeeds in enhancing the ranking performance by directly optimizing the difference between two projected vectors. As also can be seen from the results, FISM-AUC performs worse than other AUC-based methods (BPR, GBPR, PFP (WARP) and PFP (KL)) in most datasets. The reason might be that the objective function used in FISM-AUC ignores the influence of user bias for personalized ranking though it should be noted that FISM-AUC is designed for datasets with rating information such that it may be not be suitable for scenarios in which we have only binary feedback. The MaxMF method also uses multiple user latent factors to model users' preferences and this method has the same amounts of parameters as our proposed methods. The better performance of our method validates that modeling users' preferences by a combination of multiple item factors is more accurate than just using a maximal value as MaxMF does.

Third, from Table 3, we find that PFP (KL) always performs better than PFP (WARP). The improvements suggest that vector-based optimization criteria (such as the one used by PFP (KL)) are more plausible to model users' preferences and that modeling users' preferences in terms of a single value may be insufficient.

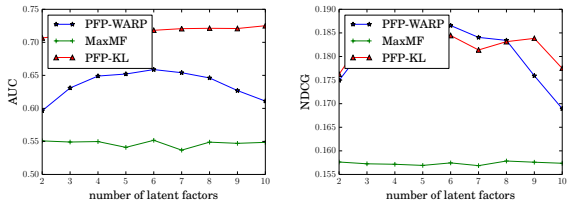
Note that when setting α_t as a constant in the WARP-Loss function, PFP (WARP) is equivalent to optimizing the AUC loss; moreover, from the results, we find that the proposed method based on the WARP-Loss always achieves similar results with that on the AUC loss, therefore, we omit AUC-based results from Table 3.

4.3 Parameter Analysis

Projection Factor Analysis. Different from existing latent factor models, PFP models require us to set not only the number of latent factors per item, but also the number of users' projection factors as hyperparameters. Therefore in this section, we experimentally investigate how different settings of the two parameter values influence the performance of PFP models. To make the results clear, we also show the performance of the most similar baseline method, MaxMF for comparison. The results are shown in Figure 3 and 4.

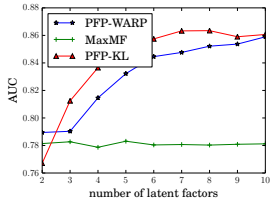
We find that in most cases, our proposed methods show better performance than MaxMF. Since MaxMF has the same number of parameters as our proposed methods, these results clearly verify that employing user's all tastes for preference modeling is better than only considering his/her maximum preference score. Moreover, we can also observe that even when PFP methods has a smaller value of K than MaxMF, the former's performance is still better than latter. This fact verifies our assumption that a user's opinion of an item may be complex and taking more parameters for modeling users' latent factors is more effective to improve the recommendation accuracy. As introduced before, K and K^* are two important parameters in PFP methods, we also give some detailed analysis about them below.

First, we examine the case where the number of item latent factors matches the number of users' projection factors. Results are

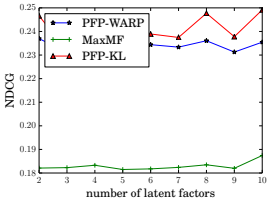


(a) Ciao(AUC)

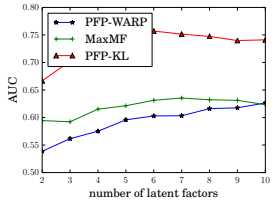
(b) Ciao(NDCG)



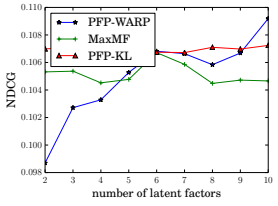
(c) BeerAdvocate(AUC)



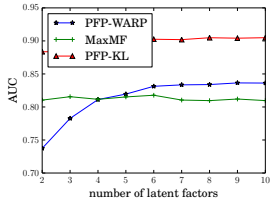
(d) BeerAdvocate(NDCG)



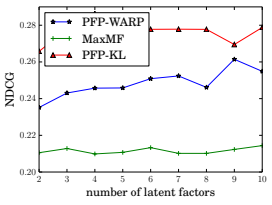
(e) Lthing(AUC)



(f) Lthing(NDCG)

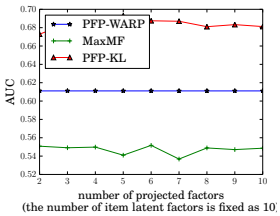


(g) Ratebeer(AUC)

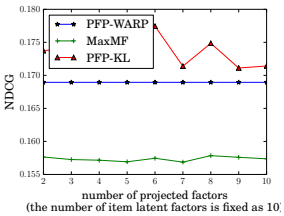


(h) Ratebeer(NDCG)

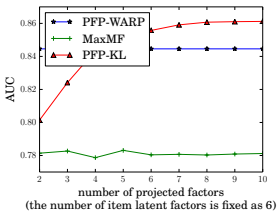
Figure 3: Impact of #latent factors, k (here we set the number of projected factors value, k^* as same as k).



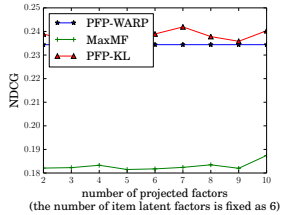
(a) Ciao(AUC)



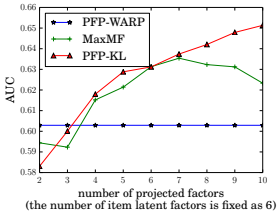
(b) Ciao(NDCG)



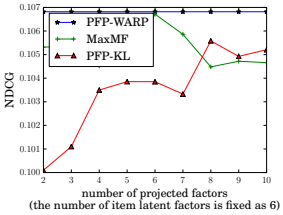
(c) BeerAdvocate(AUC)



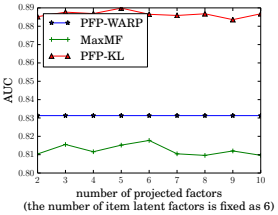
(d) BeerAdvocate(NDCG)



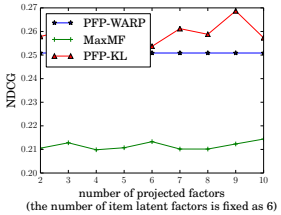
(e) Lthing(AUC)



(f) Lthing(NDCG)



(g) Ratebeer(AUC)



(h) Ratebeer(NDCG)

Figure 4: Impact of #projected factors.

shown in Figure 3. We vary K from 2 to 10 and compare the performance of PFP (KL) with PFP (WARP). AUC and NDCG are used as evaluation metrics. We find that in all datasets, PFP (KL) always performs better than PFP (WARP). In particular, compared with NDCG results, PFP (KL)'s AUC performance shows stable improvement as the number of latent factors increases. On the Ciao dataset, the NDCG performance of both methods show poor performance when $K = K^* = 10$. The reason we for this might be that when the number of positive feedback items per user is limited for training, a model with many parameters may suffer from overfitting. This conclusion is verified by our results on the Ratebeer dataset. Since Ratebeer contains sufficient positive feedback items for each user in the training set, the models' performance improves as we increase the number of latent factors in most cases.

We also conduct experiments to show how PFP methods perform when setting different values of K and K^*. Figure 4 illustrates the behavior of PFP models on different datasets. We find that different settings of the two parameters lead to varying results. On the Ciao dataset, fixing K to 10 and varying K^* from 2 to 10, PFP (KL) always achieves better performance than PFP (WARP). While on the BeerAdvocate dataset, we find that when we fix K to 6 and gradually increase K^* from 2 to 10, the performance of PFP (KL) improves significantly. This result indicates that when K^* is small, PFP (KL) cannot make use of such a low-dimensional projection space to describe users' preferences, and this also validates our motivation to model a user's preference toward an item as a combination of multiple item factors simultaneously. Similar results can be found on most other datasets. We also find an interesting result on Lthing: although PFP (KL)'s AUC outperforms PFP (WARP) when K^* is larger than 3, its NDCG performance is always unsatisfactory. One reason may be that when $K = 6$, PFP (KL) cannot project items' latent factors properly and moreover, the optimization criterion used in PFP (KL) does not focus on top-N ranking compared with the WARP-loss used in PFP (WARP). Based on the experimental results, we conclude that when we have sufficiently enough observed feedback for training, we can assign a large projection number for better user taste modeling, while when

C_u \ C_i	0.1	1	10	50	100	200
0.1	0.537	0.537	0.536	0.536	0.546	0.537
1	0.537	0.566	0.573	0.572	0.584	0.580
10	0.537	0.672	0.724	**0.725**	0.724	0.722
50	0.537	0.647	0.681	0.681	0.680	0.683
100	0.537	0.633	0.674	0.677	0.677	0.675
200	0.537	0.634	0.670	0.675	0.675	0.682

Table 4: Impact of projection constraints terms on Ciao dataset (AUC metric).

C_u \ C_i	0.1	1	10	50	100	200
0.1	0.503	0.503	0.503	0.503	0.503	0.503
1	0.503	0.878	0.868	0.870	0.867	0.871
10	0.503	0.531	**0.916**	0.909	0.907	0.906
50	0.503	0.871	0.888	0.882	0.881	0.881
100	0.503	0.871	0.887	0.884	0.881	0.880
200	0.503	0.869	0.886	0.883	0.882	0.881

Table 5: Impact of projection constraints terms on BeerAdvocate dataset. (AUC metric).

C_u \ C_i	0.1	1	10	50	100	200
0.1	0.486	0.486	0.486	0.486	0.486	0.486
1	0.486	0.884	0.877	0.874	0.871	0.876
10	0.486	0.351	**0.905**	0.897	0.892	0.891
50	0.486	0.858	0.891	0.885	0.884	0.885
100	0.486	0.856	0.891	0.888	0.886	0.883
200	0.486	0.856	0.892	0.886	0.886	0.885

Table 6: Impact of projection constraints terms on Ratebeer dataset (AUC metric).

C_u \ C_i	0.1	1	10	50	100	200
0.1	0.763	0.763	0.763	0.763	0.763	0.763
1	0.763	0.769	0.741	0.742	0.739	0.738
10	0.763	0.456	0.740	0.744	0.738	0.739
50	0.763	0.611	**0.788**	0.784	0.781	0.780
100	0.763	0.608	0.787	0.782	0.780	0.779
200	0.763	0.608	**0.788**	0.780	0.780	0.778

Table 7: Impact of projection constraints terms on Lthing dataset (AUC metric).

we have limited training data, we may decrease the projection number and use a simple model to avoid overfitting. When K^* is set to 1, the projected preference vector will degenerate to a dot product, as used in existing methods [8, 32, 37, 39].

Projection Constraint Analysis. In order to avoid overfitting, we constrain the parameters C_u and C_i using Eq. (19). Here we analyze the impact of constraint terms on model performance. The experimental results are shown in Tables 4 to 7. These results provide many hints about how to find the proper projection constraint terms for better modeling. On all datasets, we fix $K = K^* = 10$. In particular, we find that when $C_u = 10$ and $C_i = 50$, PFP (KL) exhibits the best performance on the Ciao dataset. Such results indicate that in order to avoid overfitting, we have to constrict the constraint of user's projection matrix, especially when the number of items is not particularly large (less than 20K). On the Lthing dataset, we obtain the best performance when C_u is above 50 and C_i is 10. Such results suggest that when the number of candidate items is large, it is better to reduce the constraint on users' projection matrices so that the model can describe the subtleties of users' preferences. On the BeerAdvocate and Ratebeer datasets, since the average number of positive feedback instances per user is large, the observed information is enough to model users' preferences toward items, so that the projection constraint terms do not affect the model performance on these two datasets as significantly as on the others. The best performance on both BeerAdvocate and Ratebeer datasets occur when setting $C_u = 10$ and $C_i = 10$.

Impact of η. We also perform experiments to investigate the impact of the learning rate η. The results are shown in Figure 2. We consider $\eta \in \{0.001, 0.005, 0.01, 0.05, 0.1\}$ and find that PFP models are insensitive to the value of η. Therefore, we conclude that selecting proper values for K and K^* are much more important to obtain good performance than adjusting the learning rate when training. These results also indicate that we could use relatively large learning rates to improve the convergence time of the model.

5. CONCLUSION AND FUTURE WORK

In this paper, we proposed a *Personalized Feature Projection* method in order to improve recommendation accuracy on one-class recommendation problems. Our model learns a projection matrix for each user that is able to capture the complexities of their preferences towards certain items over others. In contrast to existing methods that assume a one-to-one relationship between users' preferences and item properties, we assume that each dimension of a user's preference is related to a combination of item factors simultaneously. We formulated three optimization criteria for one-class recommendation, and performed experiments on four real-world datasets, finding that our method effectively improves the recommendation accuracy for one-class recommendation problems.

For future work, we are interested in extending PFP method in three ways: (1) Investigating how to incorporate social network information into the PFP model, in order to model the similarity between users' and their friends' preferences via the PFP method. Possible alternatives include modeling the similarity between projection matrices, or modeling the similarity between projected vectors. (2) Employing contextual information (such as text) in order to understand the meaning behind each projected latent factor; similar to [9, 18], we might allow text or time to explain the particular meanings of projected factors. (3) Exploring how to set the number of projected latent factors automatically.

6. ACKNOWLEDGEMENTS

This research was in part supported by grants from the National Grand Fundamental Research 973 Program of China (No. 2014CB340405), the Research Grants Council of the Hong Kong Special Administrative Region, China (Project No. CUHK 413213), and Microsoft Research Asia Regional Seed Fund in Big Data Research (Grant No. FY13-RES-SPONSOR-036).

References

[1] L. Baltrunas and X. Amatriain. Towards time-dependant recommendation based on implicit feedback. *In Workshop on context-aware recommender systems*, 2009.

[2] J. Davis, B. Kulis, P. Jain, S. Sra, and I. Dhillon. Information-theoretic metric learning. *In Proceedings of ICML*, 2007.

[3] L. Du, X. Li, and Y.-D. Shen. User graph regularized pairwise matrix factorization for item recommendation. *In Proceedings of ADMA*, pages 372–385, 2011.

[4] Z. Gantner, S. Rendle, C. Freudenthaler, and L. Schmidt-Thieme. Mymedialite: a free recommender system library. *In Proceedings of RecSys*, pages 305–308, 2011.

[5] Hedges. An empirical modication to linear wave theory. *In Proceedings of ICE*, pages 575 –579, 1976.

[6] L. Hong, R. Bekkerman, J. Adler, and B. D. Davison. Learning to rank social update streams. *In Proceedings of SIGIR*, 2012.

[7] Y. Hu, Y. Koren, and C. Volin-sky. Collaborative filtering for implicit feedback datasets. *In Proceedings of ICDM*, pages 263–272, 2008.

[8] S. Kabbur, X. Ning, and G. Karypis. Fism: factored item similarity models for top-n recommender systems. *In Proceedings of KDD*, pages 659–667, 2013.

[9] A. Karatzoglou, X. Amatriain, N. Oliver, and L. Baltrunas. Multiverse recommendation: n-dimensional tensor factorization for context-aware collaborative filtering. *In Proceedings of RecSys*, 2010.

[10] Y. Koren and R. Bell. Advances in collaborative filtering. *In Recommender Systems Handbook*, pages 145–186, 2011.

[11] W. K.Q. *Metric Learning with Convex Optimization*. PhD thesis, University of Pennsylvania, July 2007.

[12] W. K.Q. and L. Saul. Distance metric learning for large margin nearest neighbor classification. *The Journal of Machine Learning Research*, 10:207–244, 2009.

[13] A. Krohn-Grimberghe, L. Drumond, C. Freudenthaler, and L. Schmidt-Thieme. Multi-relational matrix factorization us-ing bayesian personalized ranking for social network data. *In Proceedings of WSDM*, pages 173–182, 2012.

[14] S. Kullback and R. Leibler. On info rmation and sufficiency. *Ann. Math.Statist*, pages 79 –86, 1951.

[15] M. Kusner, S. Tyree, K. Q. Weinberger, and K. Agrawal. Stochastic neighbor compression. *In Proceedings of ICML*, 2014.

[16] N. D. Lawrence and R. Urtasun. Non-linear matrix factorization with gaussian processes. *In Proceedings of ICML*, 2009.

[17] J. Lee, S. Bengio, S. Kim, G. Lebanon, and Y. Singer. Local collaborative ranking. *In Proceedings of WWW*, 2014.

[18] J. McAuley and J. Leskovec. Hidden factors and hidden topics: understanding rating dimensions with review text. *In Proceedings of RecSys*, 2013.

[19] R. Pan, Y. Zhou, B. Cao, N. N. Liu, R. Lukose, M. Scholz, and Q. Yang. One-class collaborative filtering. *In Proceedings of ICDM*, pages 502–511, 2008.

[20] W. Pan and L. Chen. Gbpr: Group preference based bayesian personalized ranking for one-class collaborative filtering. *In Proceedings of IJCAI*, 2013.

[21] N. Pappas and A. Popescu-Belis. Sentiment analysis of user comments for one-class collaborative filtering over ted talks. *In Proceedings of SIGIR*, 2013.

[22] U. Paquet and N. Koenigstein. One-class collaborative filtering with random graphs. *In Proceedings of WWW*, pages 999–1008, 2013.

[23] S. Parameswaran and K. Weinberger. Large margin multi-task metric learning. *In Proceedings of NIPS*, 2010.

[24] S. Rendle, C. Freuden-thaler, Z. Gantner, and L. Schmidt-Thieme. Bpr:bayesian personalized ranking from implicit feedback. *In Proceedings of UAI*, pages 452–461, 2009.

[25] S. Rendle, C. Freuden-thaler, and L. S. Thieme. Factorizing personalized markov chains for next-basket recommendation. *In Proceedings of WWW*, pages 811–820, 2010.

[26] S. Rendle and L. Schmidt-Thieme. Pairwise interaction tensor factorization for personalized tag recommendation. *In Proceedings of WSDM*, pages 81–90, 2010.

[27] J. D. M. Rennie and N.Srebro. Fast maximum margin matrix factorization for collaborative prediction. *In Proceedings of ICML*, 2005.

[28] R.Salakhutdinov and A.Mnih. Bayesian probabilistic matrix factorization using markov chain monte carlo. *In Proceedings of ICML*, 2008.

[29] R.Salakhutdinov and A.Mnih. Probabilistic matrix factorization. *In Proceedings of NIPS*, 2008.

[30] R. Salakhutdinov, A. Mnih, and G. Hinton. Restricted boltzmann machines for collaborative filtering. *In Proceedings of ICML*, 2007.

[31] M. Schultz and T. Joachims. Learning a distance metric from relative comparisons. *In Proceedings of NIPS*, 2003.

[32] Y. Shi, A. Karatzoglou, L. Baltrunas, M. Larson, and A. Hanjalic. Climf: Learning to maximize reciprocal rank with collaborative less-is-more filtering. *In Proceedings of RecSys*, 2012.

[33] M. Slaney, K. Weinberger, and W. White. Learning a metric for music similarity. *In Proceedings of ISMIR*, 2008.

[34] Srensen. A method of establishing groups of equal amplitude in plant sociology based on similarity of species and its application to analyses of the vegetation on danish commons. *Kongelige Danske Videnskabernes Selskab*, pages 1–34, 1948.

[35] K. Wagstaff and C. Cardie. Clustering with instance-level constraints. *In Proceedings of AAAI*, 2000.

[36] H. Wang, X. He, M.-W. Chang, Y. Song, R. W. White, and W. Chu. Personalized ranking model adaptation for web search. *In Proceedings of SIGIR*, 2013.

[37] M. Weimer, A. Karatzoglou, and A. Smola. Improving maximum margin matrix factorization. *Machine Learning Journal*, 2008.

[38] J. Weston, S. Bengio, and N. Usunier. Large scale image annotation: learning to rank with joint word-image embeddings. *In Proceedings of Machine Learning*, 2010.

[39] J. Weston, R. J. Weiss, and H. Yee. Nonlinear latent factorization by embedding multiple user interests. *In Proceedings of RecSys*, 2013.

[40] J. Weston, H. Yee, and R. J. Weiss. Learning to rank recommendations with the k-order statistic loss. *In Proceedings of RecSys*, pages 245–248, 2013.

[41] E. Xing, A. Ng, M. Jordan, and S. Russell. Distance metric learning, with application to clustering with side-information. *In Proceedings of NIPS*, 2002.

[42] S.-H. Yang, B. Long, A. Smola, N. Sadagopan, Z. Zheng, and H. Zha. Like like alike: joint friendship and interest propagation in social networks. *In Proceedings of WWW*, 2011.

[43] S.-H. Yang, B. Long, A. J. Smola, H. Zha, and Z. Zheng. Collaborative competitive filtering: learning recommender using context of user choice. *In Proceedings of SIGIR*, pages 295–304, 2011.

[44] Y. Yue, C. Wang, K. El-Arini, and C. Guestrin. Personalized collaborative clustering. *In Proceedings of WWW*, 2014.

[45] T. Zhao, J. McAuley, and I. King. Leveraging social connections to improve personalized ranking for collaborative filtering. *In Proceedings of CIKM*, 2014.