Improving Top-N Recommendation with Heterogeneous Loss

Feipeng Zhao and Yuhong Guo
Department of Computer and Information Sciences
Temple University, Philadelphia, PA 19122, USA
{feipeng.zhao, yuhong}@temple.edu

Abstract

Personalized top-N recommendation systems have great impact on many real world applications such as E-commerce platforms and social networks. Most existing methods produce personalized top-N recommendations by minimizing a specific uniform loss such as pairwise ranking loss or pointwise recovery loss. In this paper, we propose a novel personalized top-N recommendation approach that minimizes a combined heterogeneous loss based on linear self-recovery models. The heterogeneous loss integrates the strengths of both pairwise ranking loss and pointwise recovery loss to provide more informative recommendation predictions. We formulate the learning problem with heterogeneous loss as a constrained convex minimization problem and develop a projected stochastic gradient descent optimization algorithm to solve it. We evaluate the proposed approach on a set of personalized top-N recommendation tasks. The experimental results show the proposed approach outperforms a number of state-of-the-art methods on top-N recommendation.

1 Introduction

In the era of Internet, online shopping has become a close part of people’s daily life. To facilitate consumers’ online shopping experience, help consumers to find their interested items from the huge amount of online items and hence encourage online item purchases, effective top-N commercial recommendation systems, which suggest a small list of items that best match each consumer’s taste from the large amount of product items, have become increasingly important. Top-N recommendation systems automatically predict the recommendation scores over each item in the whole product item pool for each consumer, and recommend the items with high scores to the consumers. They can be widely used in many real world applications. For example, Amazon.com has a huge amount of online products, while consumers may not know all the special items they would like. A top-N recommendation system, which can effectively suggest a set of personalized selected products for each consumer, can significantly increase the user-item purchase probability. Similarly, Netflix Inc. has thousands of online movies and TV shows, a personalized recommendation for users can help them to find the videos they may like. Yahoo! and Apple iTunes have millions of musics. Top-N recommendation systems can help users to find the musics they may like and provide a recommendation list to each user. In summary, an effective top-N personalized recommendation system can have significant real world commercial impacts.

Many methods have been developed in the literature to build top-N recommendation systems [Ricci et al., 2011]. One classical technique is collaborative filtering (CF) [Schafer et al., 2007; Su and Khoshgoftaar, 2009], which models the relationships between users and the correlations between items to identify new user-items relationship scores. Two main types of collaborative filtering methods include neighborhood-based CF [Sarwar et al., 2000; 2001; Deshpande and Karypis, 2004; Verstrepen and Goethals, 2014] and model-based CF [Rennie and Srebro, 2005; Rendle et al., 2009; Cremonesi et al., 2010; Sindhwani et al., 2010; Hu et al., 2008; Shi et al., 2012; Liu and Aberer, 2014]. Standard model-based CF methods perform matrix factorization to complete the missing recommendation entries, which exploits the low-dimensional subspace representations of the users and items [Hu et al., 2008; Weimer et al., 2008; Yun et al., 2014b]. Besides CF, sparse aggregation methods that exploit the linear correlations between items have also been explored in a few works [Ning and Karypis, 2011; Cheng et al., 2014; Kabbur et al., 2013; Christakopoulou and Karypis, 2014] to improve top-N recommendation performance. These methods have mainly pursued recommendation score recovery by minimizing the pointwise comparison loss between the reconstructed user-item matrix and the observed incomplete user-item matrix.

Recently, pairwise ranking methods that directly capture users’ pairwise preference structures on product items have demonstrated good performance for top-N recommendation [Weimer et al., 2008; Steck, 2010; Chen and Pan, 2013; Aioli, 2014; Park et al., 2015]. These methods have explored different ranking losses as the optimization targets, including the normalized discounted cumulative gain [Weimer et al., 2008], the AUC scores [Aioli, 2014] and the max-margin loss [Park et al., 2015]. Nevertheless, different types of losses have different strengths in producing the missing recommendation scores. The pointwise losses enforce the entrywise
consistency between the reconstructed recommendation matrix and the original matrix, while the pairwise ranking losses enforce the preference structure consistency. Majority of the existing methods however have focused on using a uniform type of recovery loss, which limits their abilities of integrating the complementary strengths of different types of losses.

In this paper, we propose a novel personalized top-N recommendation approach that uses a combined heterogeneous loss function based on linear self-recovery models. The heterogeneous loss integrates the strengths of pairwise ranking and pointwise recovery loss to enforce both entry-wise consistency and preference structure consistency between the recommendation recovery and the observations, which leads to improved recommendation predictions. We formulate the learning problem with this heterogeneous loss as a constrained convex minimization problem and develop a projected stochastic gradient descent optimization algorithm to solve it. The proposed approach is evaluated on a set of personalized top-N recommendation tasks. The experimental results show the proposed approach outperforms a number of state-of-the-art methods on top-N recommendation.

2 Related Works

Existing top-N recommendation methods have mostly focused on minimizing a uniform type of recommendation recovery loss to predict the recommendation scores. Based on the type of the losses, these recommendation systems can be categories into two main groups. The first group exploits pointwise comparison losses and the second group exploits pairwise ranking losses.

Pointwise comparison methods. Many traditional top-N recommendation algorithms perform learning by minimizing the pointwise (entrywise) divergence of the reconstructed recommendation matrix from the original observation matrix. These include the collaborative filtering methods [Schafer et al., 2007; Su and Khoshgoftaar, 2009; Hu et al., 2008; Weimer et al., 2008; Yun et al., 2014b], and the sparse linear aggregation methods [Ning and Karypis, 2011; Cheng et al., 2014; Kabbur et al., 2013; Christakopoulou and Karypis, 2014]. For example, a weighted regularized matrix factorization (WRMF) model has been developed in [Hu et al., 2008], which minimizes the pointwise difference between the reconstructed matrix produced by the inner product of the user and item latent representations and the training matrix. Similarly, the work in [Yun et al., 2014b] enforces the pointwise similarity between the training matrix and reconstructed matrix. It uses a non-locking parallel computing algorithm to perform matrix completion. In [Ning and Karypis, 2011], a sparse linear method (SLIM) was proposed to recover the missing recommendation entries. It exploits the item-item correlations to reconstruct the incomplete recommendation matrix and minimizes the pointwise reconstruction loss between the reconstructed matrix and the input training matrix.

Pairwise ranking methods. Given a pair of items, a user naturally will prefer one to another, which forms the important pairwise preference structure of the recommendation data. The pointwise comparison methods however ignored this user personalized preference information. Recently, pairwise ranking methods are proposed to produce top-N recommendation systems by optimizing the preference structure consistency between the original matrix and the reconstructed recommendation matrix [Weimer et al., 2008; Steck, 2010; Chen and Pan, 2013; Aiollli, 2014; Park et al., 2015; Rendle et al., 2009]. Pairwise ranking methods treat training data as a set of triplet instances; for example, the triplet \((i, j, k)\) is an instance that encodes the \(i\)-th user’s preference to item \(j\) over item \(k\). Different pairwise ranking losses have been exploited in these works. For example, the pairwise ranking methods in [Rendle et al., 2009; Aiollli, 2014] optimize AUC scores; the work in [Weimer et al., 2008] optimizes a normalized discounted cumulative gain; Yun et al. (2014a) explored the connection between the metric discounted cumulative gain and the binary classification to change the ranking problem into binary classification problems; Park et al. (2015) proposed a large-scale collaborative ranking method that exploits a max-margin hinge ranking loss to minimize the ranking risk in the reconstructed recommendation matrix. Nevertheless, these pairwise ranking methods ignored the entrywise consistency between the reconstructed matrix and the original matrix.

Different from all these existing methods, our proposed approach will perform top-N recommendations by optimizing a heterogeneous loss that integrates the strengths of both a pointwise comparison loss and a pairwise ranking loss.

3 Approach

In this section, we present a novel personalized top-N recommendation approach that minimizes a combined heterogeneous loss within a general learning framework. We assume a partially observed user-item recommendation/purchase matrix \(X \in \mathbb{R}^{n \times m}\) over \(n\) users and \(m\) items is given. For implicit feedbacks, the recommendation/rating values are within \(\{0, 1\}\), where the entry \(X_{ij} = 1\) indicates a transaction or purchase record for the \(i\)-th user on the \(j\)-th item. The entry \(X_{ij} = 0\) on the other hand means either the \(i\)-th user never purchased the \(j\)-th item or the transaction record has been removed and need to be predicted. We aim to identify the most interesting items for each user from his recommended/unpurchased list of items. Below we first introduce the general learning framework for top-N recommendations and then instantiate it with novel objective losses. We finally present a stochastic gradient descent algorithm to solve the recommendation problem formulated.

3.1 A General Learning Framework for Top-N Recommendation

Given the input user-item recommendation/purchase matrix \(X\), a general framework for top-N recommendation performs learning by minimizing a regularized reconstruction loss function:

\[
\min_W \mathcal{L}(X, \hat{X}(W)) + R(W) \tag{1}
\]

where \(\hat{X}\) denotes the reconstructed recommendation matrix with a parametric model with parameter matrix \(W\), \(\mathcal{L}(\cdot)\) denotes a convex reconstruction loss function and \(R(\cdot)\) denotes a regularization function. By substituting \(\mathcal{L}(\cdot)\) with different
loss functions and employing different reconstruction models (denoted by the parameter \( W \)), many existing methods can be produced from this general framework as specific examples. For example, by using a matrix factorization model that reconstructs \( X \) as \( \hat{X} = UV^T \) and a least squares pointwise loss function \( L(X, \hat{X}) = \|X - \hat{X}\|^2_F \), we can produce the simple pure singular value decomposition based (PureSVD) matrix factorization method [Cremonesi et al., 2010] with proper constraints. Similarly, by reconstructing \( X \) with a linear aggregation model \( \hat{X} = XW^T \) and using a least squares pointwise loss function \( L(\cdot) \) and an integrated \( \ell_2 \) and \( \ell_1 \) norm regularization function \( R(W) = \beta \|W\|^2_F + \lambda \|W\|_1 \), the sparse linear method (SLIM) [Ning and Karypis, 2011] can be produced from the framework with additional proper constraints.

Moreover, the state-of-the-art pairwise ranking methods can also be produced as specific examples from this framework. For example, given the set of preference triplets \( \Omega \), by using a matrix factorization model to reconstruct \( X \) as \( \hat{X} = UV^T \), a max-margin hinge loss function \( L_{\text{rank}}(x) = \max(0, 1 - x) \), and \( \ell_2 \) norm regularizers on \( U \) and \( V \), we can produce the large scale pairwise ranking method (AltSVM) in [Park et al., 2015] as below:

\[
\min_{U,V} \sum_{(i,j,k) \in \Omega} L_{\text{rank}}(Y_{ijk}(X) \cdot U_i(V_j - V_k)^T) + \frac{\beta}{2} (\|U\|^2_F + \|V\|^2_F) \quad (2)
\]

where \( Y_{ijk} \) is a function of \( X \) such that \( Y_{ijk} = 1 \) if user \( i \) prefers item \( j \) over item \( k \) in \( X \) and \( Y_{ijk} = -1 \) otherwise; \( \| \cdot \|_F \) denotes the matrix Frobenius norm; \( U_i \) denotes the \( i \)-th row of \( U \) and \( V_j \) denotes the \( j \)-th row of \( V \).

Our proposed approach can be conveniently formulated within this general learning framework as well. We integrate two types of losses into a novel heterogeneous loss based on linear self-recovery models.

### 3.2 Novel Heterogeneous Loss

Pairwise ranking methods have demonstrated great top-N recommendation performance in the literature [Park et al., 2015]. However, with the matrix factorization reconstruction function, the scalable learning problem produced in Eq. (2) is a non-convex optimization problem. To introduce a convenient convex formulation with pairwise ranking losses, we propose to adopt a linear self-recovery model to reconstruct \( X \). In particular, we use a linear reconstruction function \( \hat{X} = XW^T \) with constraints \( W \geq 0 \) and \( \text{diag}(W) = 0 \). This linear function exploits the statistical positive item-item correlations distributed among all the users to reconstruct the missing recommendation scores. The constraints are used to avoid trivial solutions. Then with a pairwise ranking loss function \( L_{\text{rank}}(\cdot) \), we can formulate the following convex pairwise ranking problem to perform top-N recommendation:

\[
\min_{W} \sum_{(i,j,k) \in \Omega} L_{\text{rank}}(Y_{ijk}(X) \cdot X_i(W_j - W_k)^T) + \frac{\beta}{2} \|W\|^2_F \quad \text{s.t.} \quad W \geq 0, \quad \text{diag}(W) = 0 \quad (3)
\]

Note for each user \( i \), if he/she purchased item \( j \) but there is no purchase record for item \( k \) in the training matrix \( X \), we say that user \( i \) prefers item \( j \) over item \( k \) and we have \( X_{ij} > X_{ik} \). The set \( \Omega \) in the formulation above contains all the triplets \((i,j,k)\) where \( X_{ij} > X_{ik} \). To maintain the same pairwise preference structure, we assume that if \( X_{ij} > X_{ik} \) in the original matrix \( X \), one should also have \( \hat{X}_{ij} > \hat{X}_{ik} \) in the reconstructed matrix \( \hat{X} \). The loss function \( L_{\text{rank}}(\cdot) \) aims to encode the pairwise preference ranking divergence between the given matrix \( X \) and the reconstructed matrix \( \hat{X} \). In our implementation, we use the max-margin hinge loss function as the ranking loss, i.e., \( L_{\text{rank}}(x) = \max(0, 1 - x) \). By minimizing such a ranking loss function, the model will increase the consistency of the pairwise preference ranking between the reconstructed matrix \( \hat{X} \) and the original matrix \( X \). However, the consistency is only maintained at the pairwise relative level. It does not necessarily guarantee the good quality of pointwise reconstruction, which might consequently hamper the accurate inference of unseen pairwise relationships.

We nevertheless have an easy solution. The linear self-recovery model \( \hat{X} = XW^T \) naturally allows one to encode the pointwise recovery loss between the entries of the reconstructed matrix \( \hat{X} \) and the original matrix \( X \), e.g., \( L_{\text{point}}(X, \hat{X}) = \|X - \hat{X}\|^2_F \). Previous works [Ning and Karypis, 2011] have also demonstrated good top-N recommendation performance with a least squares pointwise recovery loss. Hence we propose to combine both the pairwise ranking loss and the pointwise recovery loss to produce a new heterogeneous loss function for top-N recommendation. Our learning problem with the heterogeneous loss function can be formulated as below:

\[
\min_{W} \sum_{(i,j,k) \in \Omega} L_{\text{rank}}(Y_{ijk}(X) \cdot X_i(W_j - W_k)^T) + \frac{\alpha}{2} \|X - XW^T\|^2_F + \frac{\beta}{2} \|W\|^2_F \quad \text{s.t.} \quad W \geq 0, \quad \text{diag}(W) = 0 \quad (4)
\]

The heterogeneous loss function used in this formulation integrates two different types of losses, and is expected to capture both the pointwise similarity discrepancy and the pairwise ranking consistency discrepancy to produce a better reconstruction matrix. Moreover, the learning problem (4) remains to be a convex optimization problem.

### 3.3 Optimization Algorithm

Due to the large size of the user-item matrix, standard gradient descent algorithms are not efficient for solving the minimization problem in (4). Moreover, though the learning problem is convex, the hinge loss function for pairwise ranking is non-differentiable and subgradients are typically needed. Hence in this work, we develop a projected stochastic gradient descent (SGD) algorithm to solve our target learning problem. The SGD technique has been widely used in recommendation systems with pairwise ranking [Rendle et al., 2009; Chen and Pan, 2013; Yun et al., 2014b], and it can also be extended to perform parallel computations [Zinkevich et al., 2011; Recht et al., 2011; Yun et al., 2014b].
With the pairwise ranking loss, we use the user-item-item triplet \((i,j,k)\) instances for the stochastic gradient descent procedure. In each iteration, we randomly choose a triplet \((i,j,k)\) \(\in \Omega\) and make a SGD update:

\[
W_j^{+} \leftarrow W_j^{-} - \eta \left\{ \mathcal{L}_{\text{rank}}^\prime(W_j^{+}) + \frac{\beta}{|\Omega|^2} W_j^{+} - \frac{\alpha}{|\Omega|^2} Y_j X_j^T (X W_j - X_{ij}) \right\}
\]

(5)

\[
W_k^{+} \leftarrow W_k^{-} - \eta \left\{ \mathcal{L}_{\text{rank}}^\prime(W_k^{+}) + \frac{\beta}{|\Omega|^2} W_k^{+} - \frac{\alpha}{|\Omega|^2} Y_k X_k^T (X W_k - X_{ik}) \right\}
\]

(6)

where \(|\Omega|\) and \(|\Omega^k|\) denote the number of comparisons in \(\Omega\) that involve item \(j\) and item \(k\) respectively. \(\mathcal{L}_{\text{rank}}^\prime(W_j^{+})\) and \(\mathcal{L}_{\text{rank}}^\prime(W_k^{+})\) denote the derivations of \(\mathcal{L}_{\text{rank}}(\cdot)\) on \(W_j^{+}\) and \(W_k^{+}\) respectively, such that

\[
\mathcal{L}_{\text{rank}}^\prime(W_j^{+}) = Y_{ijk} \cdot 2 X_j^T (X_j W_j - W_j) - 1 \quad \text{if } X_j(W_j - W_k) \leq 1 \quad \text{and } \mathcal{L}_{\text{rank}}^\prime(W_j^{+}) = 0 \quad \text{otherwise.}
\]

Similarly,

\[
\mathcal{L}_{\text{rank}}^\prime(W_k^{+}) = -Y_{ijk} \cdot 2 X_k^T (X_k W_k - W_k) - 1 \quad \text{if } X_j(W_j - W_k) \leq 1 \quad \text{and } \mathcal{L}_{\text{rank}}^\prime(W_k^{+}) = 0 \quad \text{otherwise.}
\]

The \(\eta\) parameter in the SGD update is the stepsize of gradient descent. We used a fixed small stepsize value in our experiments. The overall learning algorithm is given in Algorithm 1.

**Algorithm 1 Projected Stochastic Gradient Descent**

**Input:** \(\alpha > 0, \beta > 0, \eta > 0;\) initialize \(W^0\) as zeros.

**Set** \(W = W^0\)

**for** iter = 1 to MaxIter **do**

1. Randomly select \((i,j,k)\) \(\in \Omega\)
2. Update \(W_j\) by using (5)
3. Project the updated \(W_j\) into the feasible set:
   \[
   W_j = \max(W_j, 0), \quad W_{jj} = 0;
   \]
4. Update \(W_k\) by using (6)
5. Project the updated \(W_k\) into the feasible set:
   \[
   W_k = \max(W_k, 0), \quad W_{kk} = 0;
   \]
6. if converge then break-out **end if**

**return** \(W\)

### 4 Experimental Results

In this section, we first present the experimental setting and then report the empirical results.

**4.1 Experimental Setup**

**Datasets.** We used five datasets in our experiments: Yahoo! music ratings v1.0 dataset, Yahoo! movie ratings v1.0 dataset, MovieLens 100k (ml-100k) dataset, MovieLens 1M (ml-1m) dataset and Netflix dataset. Yahoo!Music contains ratings for songs, the ratings are supplied by users during normal interactions with Yahoo! Music services. Similarly, Yahoo!Movie contains ratings for different movies. In each dataset, we converted the entries with positive values to 1 and converted the user-item matrix to an implicit feedback matrix. Since the original rating matrix is sparse, for Yahoo! Movie and Yahoo! Music we kept the users with more than 20 ratings and items with more than 5 ratings. For ml-100k we used original implicit feedback matrix. For ml-1m and Netflix we used the same datasets as [Aiolli, 2014]. The statistic information of these datasets is reported in Table 1.

**Evaluation Methods.** For each dataset, we split it into a training set and a test set. For each user, we randomly selected ten feedbacks and placed them into the test set and the rest were used as training set. After training, a ranked list of top-N items can be returned for each user according to the reconstruction scores, which were compared to the test set for performance evaluation.

We used two standard performance metrics, the top-N prediction precision (Precision@N) and the mean average precision (MAP@N), to evaluate the test performance. The top-N prediction precision for the \(i\)-th user is defined as

\[
\text{Precision@N}(i) = \frac{1}{N} \sum_{j=1}^{N} T(i,j) \times P_i(j)
\]

(7)

where \(T\) denotes the test matrix and \(P_i\) denotes the index values of the top-N predicted items in the original matrix for the \(i\)-th user. The overall Precision@N value is computed as the average of Precision@N(i) over all users. MAP@N denotes the mean average precision of the top-N predictions. The average precision (AP) of the top-N prediction for the \(i\)-th user can be defined as

\[
\text{AP@N}(i) = \frac{\sum_{j=1}^{N} \text{Precision@j}(i) \times T(i,j)}{\sum_{j=1}^{N} T(i,j)}
\]

(8)

MAP@N can be computed as the mean of the average precision of the top-N predictions for all users, such that

\[
\text{MAP@N} = \frac{1}{n} \sum_i \text{AP@N}(i)
\]

In our experiment, we randomly split each dataset into training and test matrices for five times and report the average test results in terms of these two evaluation metrics.

**Compared Methods.** We compared the proposed method with four methods developed in the literature: WRMF [Hu et al., 2008], SLIM [Ning and Karypis, 2011], RobiRank [Yun et al., 2014a] and AltSVM [Park et al., 2015].

**Table 1: Statistic information of the datasets:** The columns of #user, #item and #transact show the numbers of users, items, and non-zero transactions respectively in each dataset. The columns of #size and #size show the average number of transactions for each user and each item respectively. The column of density shows the density of non-zero transactions in each dataset.

<table>
<thead>
<tr>
<th>Dataset</th>
<th>#user</th>
<th>#item</th>
<th>#transact</th>
<th>density</th>
</tr>
</thead>
<tbody>
<tr>
<td>Yahoo! Music</td>
<td>2689</td>
<td>994</td>
<td>86907</td>
<td>3.95%</td>
</tr>
<tr>
<td>Yahoo! Movie</td>
<td>2382</td>
<td>924</td>
<td>104459</td>
<td>4.75%</td>
</tr>
<tr>
<td>ml-100k</td>
<td>943</td>
<td>1682</td>
<td>100000</td>
<td>6.30%</td>
</tr>
<tr>
<td>ml-1m</td>
<td>3850</td>
<td>2273</td>
<td>315869</td>
<td>3.60%</td>
</tr>
<tr>
<td>Netflix</td>
<td>2979</td>
<td>2544</td>
<td>114865</td>
<td>1.50%</td>
</tr>
</tbody>
</table>

2381
Table 2: Average test results of top-N recommendations for all the comparison methods. The params columns contain the parameter settings for each approach. WRMF, RobiRank and AltSVM have two parameters, the latent factor dimension and the regularization parameter. SLIM has two parameters, $\ell_2$ and $\ell_1$ norm regularization parameters. The proposed approach has two parameters, the weight control parameter and the regularization parameter. Bold-font indicates the best results.

<table>
<thead>
<tr>
<th>method</th>
<th>params</th>
<th>Precision@5</th>
<th>Yahoo! Music</th>
<th>MAP@5</th>
<th>MAP@10</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td>Precision@10</td>
<td></td>
<td></td>
</tr>
<tr>
<td>WRMF</td>
<td>200 100</td>
<td>0.282 ± 0.002</td>
<td>0.226 ± 0.001</td>
<td>0.521 ± 0.002</td>
<td>0.482 ± 0.002</td>
</tr>
<tr>
<td>SLIM</td>
<td>100 1</td>
<td>0.273 ± 0.002</td>
<td>0.213 ± 0.001</td>
<td>0.504 ± 0.003</td>
<td>0.472 ± 0.002</td>
</tr>
<tr>
<td>RobiRank</td>
<td>100 100</td>
<td>0.261 ± 0.001</td>
<td>0.209 ± 0.001</td>
<td>0.461 ± 0.002</td>
<td>0.430 ± 0.002</td>
</tr>
<tr>
<td>AltSVM</td>
<td>500 1000</td>
<td>0.275 ± 0.003</td>
<td>0.219 ± 0.002</td>
<td>0.504 ± 0.003</td>
<td>0.470 ± 0.002</td>
</tr>
<tr>
<td>Proposed</td>
<td>5000 0.01</td>
<td>0.318 ± 0.001</td>
<td>0.244 ± 0.001</td>
<td>0.572 ± 0.002</td>
<td>0.526 ± 0.002</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>method</th>
<th>params</th>
<th>Precision@5</th>
<th>Yahoo! Movie</th>
<th>MAP@5</th>
<th>MAP@10</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td>Precision@10</td>
<td></td>
<td></td>
</tr>
<tr>
<td>WRMF</td>
<td>200 100</td>
<td>0.390 ± 0.001</td>
<td>0.316 ± 0.001</td>
<td>0.595 ± 0.003</td>
<td>0.554 ± 0.003</td>
</tr>
<tr>
<td>SLIM</td>
<td>100 1</td>
<td>0.378 ± 0.002</td>
<td>0.303 ± 0.001</td>
<td>0.577 ± 0.003</td>
<td>0.539 ± 0.003</td>
</tr>
<tr>
<td>RobiRank</td>
<td>100 100</td>
<td>0.358 ± 0.002</td>
<td>0.290 ± 0.002</td>
<td>0.520 ± 0.003</td>
<td>0.483 ± 0.002</td>
</tr>
<tr>
<td>AltSVM</td>
<td>500 1000</td>
<td>0.391 ± 0.002</td>
<td>0.316 ± 0.001</td>
<td>0.608 ± 0.003</td>
<td>0.561 ± 0.003</td>
</tr>
<tr>
<td>Proposed</td>
<td>5000 0.01</td>
<td>0.430 ± 0.001</td>
<td>0.337 ± 0.001</td>
<td>0.657 ± 0.001</td>
<td>0.606 ± 0.001</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>method</th>
<th>params</th>
<th>Precision@5</th>
<th>ml-100k</th>
<th>MAP@5</th>
<th>MAP@10</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td>Precision@10</td>
<td></td>
<td></td>
</tr>
<tr>
<td>WRMF</td>
<td>200 100</td>
<td>0.299 ± 0.001</td>
<td>0.243 ± 0.001</td>
<td>0.515 ± 0.003</td>
<td>0.481 ± 0.002</td>
</tr>
<tr>
<td>SLIM</td>
<td>200 1</td>
<td>0.323 ± 0.002</td>
<td>0.247 ± 0.001</td>
<td>0.553 ± 0.003</td>
<td>0.516 ± 0.003</td>
</tr>
<tr>
<td>RobiRank</td>
<td>200 100</td>
<td>0.317 ± 0.003</td>
<td>0.254 ± 0.001</td>
<td>0.520 ± 0.003</td>
<td>0.477 ± 0.002</td>
</tr>
<tr>
<td>AltSVM</td>
<td>200 1000</td>
<td>0.316 ± 0.002</td>
<td>0.245 ± 0.002</td>
<td>0.552 ± 0.003</td>
<td>0.516 ± 0.003</td>
</tr>
<tr>
<td>Proposed</td>
<td>5000 0.01</td>
<td>0.342 ± 0.001</td>
<td>0.269 ± 0.001</td>
<td>0.574 ± 0.002</td>
<td>0.524 ± 0.002</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>method</th>
<th>params</th>
<th>Precision@5</th>
<th>ml-Im</th>
<th>MAP@5</th>
<th>MAP@10</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td>Precision@10</td>
<td></td>
<td></td>
</tr>
<tr>
<td>WRMF</td>
<td>1000 100</td>
<td>0.193 ± 0.002</td>
<td>0.160 ± 0.002</td>
<td>0.365 ± 0.002</td>
<td>0.355 ± 0.002</td>
</tr>
<tr>
<td>SLIM</td>
<td>200 1</td>
<td>0.193 ± 0.003</td>
<td>0.156 ± 0.002</td>
<td>0.372 ± 0.003</td>
<td>0.359 ± 0.002</td>
</tr>
<tr>
<td>RobiRank</td>
<td>200 1000</td>
<td>0.188 ± 0.002</td>
<td>0.154 ± 0.002</td>
<td>0.358 ± 0.003</td>
<td>0.349 ± 0.002</td>
</tr>
<tr>
<td>AltSVM</td>
<td>200 1000</td>
<td>0.189 ± 0.002</td>
<td>0.159 ± 0.002</td>
<td>0.362 ± 0.002</td>
<td>0.352 ± 0.002</td>
</tr>
<tr>
<td>Proposed</td>
<td>5000 0.1</td>
<td>0.211 ± 0.002</td>
<td>0.167 ± 0.001</td>
<td>0.397 ± 0.003</td>
<td>0.382 ± 0.002</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>method</th>
<th>params</th>
<th>Precision@5</th>
<th>Netfllx</th>
<th>MAP@5</th>
<th>MAP@10</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td>Precision@10</td>
<td></td>
<td></td>
</tr>
<tr>
<td>WRMF</td>
<td>200 100</td>
<td>0.199 ± 0.002</td>
<td>0.163 ± 0.002</td>
<td>0.374 ± 0.002</td>
<td>0.364 ± 0.002</td>
</tr>
<tr>
<td>SLIM</td>
<td>100 0.1</td>
<td>0.192 ± 0.002</td>
<td>0.153 ± 0.002</td>
<td>0.372 ± 0.002</td>
<td>0.363 ± 0.002</td>
</tr>
<tr>
<td>RobiRank</td>
<td>20 1000</td>
<td>0.143 ± 0.002</td>
<td>0.115 ± 0.001</td>
<td>0.261 ± 0.002</td>
<td>0.267 ± 0.001</td>
</tr>
<tr>
<td>AltSVM</td>
<td>500 1000</td>
<td>0.195 ± 0.003</td>
<td>0.159 ± 0.002</td>
<td>0.367 ± 0.002</td>
<td>0.358 ± 0.002</td>
</tr>
<tr>
<td>Proposed</td>
<td>5000 0.1</td>
<td>0.220 ± 0.002</td>
<td>0.173 ± 0.002</td>
<td>0.416 ± 0.003</td>
<td>0.396 ± 0.002</td>
</tr>
</tbody>
</table>

**Parameters Selection.** For the proposed method, we have two parameters, $\alpha$ and $\beta$. We selected the weight control parameter $\alpha$ from $\{10, 100, 1000, 5000, 10000\}$ and selected the regularization parameter $\beta$ from $\{0.001, 0.01, 0.1, 1, 10, 100, 1000\}$. For WRMF, the latent factor dimension $f$ is selected from $\{10, 20, 50, 100, 200, 500, 1000\}$ and the regularization parameter $\lambda$ is selected from $\{0.1, 1, 10, 100, 1000\}$. For SLIM, $\ell_2$ regularization parameter $\beta$ is selected from $\{1, 10, 100, 200, 500, 1000\}$ and $\ell_1$ regularization parameter $\lambda$ is selected from $\{0.001, 0.1, 1, 10\}$. For RobiRank, the latent feature dimension $f$ is selected from $\{10, 20, 50, 100, 200, 500, 1000\}$ and the regularization parameter $\lambda$ is selected from $\{1, 10, 100, 200, 500, 1000\}$. AltSVM also has two similar parameters; the latent dimension is selected from $\{100, 200, 500, 1000\}$ and the regularization parameter $\lambda$ is selected from $\{100, 1000, 10000\}$. For each approach, we report the best results and the corresponding parameter settings.

**4.2 Experimental Results**

The selected parameter settings and the experimental results in terms of Precision@N and MAP@N with $N \in \{5, 10\}$ for all the comparison approaches are reported in Table 2. We can see that among the four comparison methods, \{WRMF, SLIM, RobiRank and AltSVM\}, RobiRank has poor performance and it produces the most inferior results on four datasets, Yahoo!Music, Yahoo!Movie, Netfllx and ml-Im. SLIM performs much better than RobiRank and it even outperforms RobiRank on the remaining dataset ml-100k across three measurements except Precision@10. The pairwise preference ranking method, AltSVM, produces the best results on Yahoo!Movie in terms of all the measurements among the four methods, and has similar performance as SLIM on other datasets. The WRMF method, which uses regularization control and applies
different weights for the entries, outperforms all the other three methods on both Yahoo!Music and Netflix in terms of the four measurements. But it has poor performance on ml-100k. Our proposed approach on the other hand outperforms all these four methods with remarkable margins on all the datasets across the four measurements. These results suggest the proposed approach is indeed an effective method for top-
N recommendation.

4.3 Parameter Sensitivity Analysis

We have also conducted parameter sensitivity analysis for the proposed method on Yahoo!Movie and Yahoo!Music datasets. First, we tested different $\alpha$ values from \{0, 10, 100, 1000, 5000, 10000\} by setting $\beta$ to the fixed value used before. The experimental results are reported in Figure 1. We can see that for Yahoo!Movie dataset, the precision values increase when $\alpha$ increases from 0 to 5000. When $\alpha = 5000$, the model performs the best. Similar results can be observed on Yahoo!Music dataset. According to our objective function in Eq. (4), if $\alpha = 0$, the model will only use the pairwise ranking loss and will not combine the pointwise recovery loss. On the other hand, if $\alpha$ is too big, the contribution of the pairwise ranking loss can be diminished and the model will become a special version of the SLIM method. Our parameter analysis results suggest that we need a parameter in the middle to keep both types of losses, which indicates the two types of losses can complement each other. To further validate this, we have also compared our proposed approach to its variants that only use each individual loss. We call the variant that drops the pointwise recovery loss ($\alpha = 0$) as \textit{pureRanking} and the variant that drops the pairwise ranking loss as variant of SLIM. The comparison results are reported in Figure 2. We can see that the proposed approach with integrated heterogeneous loss outperforms both variants that only use a uniform type of loss component. This validated our algorithm design of using heterogeneous losses.

We also conducted parameter sensitivity analysis on the $\ell_2$-norm regularization parameter $\beta$. We fixed the value of $\alpha$ as 5000 and choose $\beta$ value from \{0, 10$^{-3}$, 1, 10$^3$, 10$^6$\}. The experiment results with different $\beta$ values are reported in Figure 3. We can see that the top-N recommendation performance is not sensitive to the $\beta$ values when $\beta \leq 1000$.

5 Conclusion

In this paper, we proposed a novel personalized top-N recommendation approach that exploits novel heterogeneous loss functions based on linear self-recovery models. The heterogeneous loss integrates the strengths of both pairwise ranking loss and pointwise recovery loss to enforce both entrywise consistency and pairwise preference structure consistency between the reconstructed recommendation matrix and the original observation matrix. We formulated the training problem with the heterogeneous loss as a constrained convex minimization problem and develop a projected stochastic gradient descent optimization algorithm to solve it. The proposed approach was evaluated on a set of real world personalized top-N recommendation tasks. The experimental results showed that the proposed approach not only outperforms its two variants that only used either the comparison recovery loss or the pairwise ranking loss, but also outperforms a number of state-of-the-art methods on top-N recommendation.
References


