Local Item-Item Models for Top-N Recommendation

Evangelia Christakopoulou and George Karypis
Computer Science & Engineering
University of Minnesota, Minneapolis, MN
{evangel,karypis}@cs.umn.edu

ABSTRACT

Item-based approaches based on SLIM (Sparse LInear Methods) have demonstrated very good performance for top-N recommendation; however they only estimate a single model for all the users. This work is based on the intuition that not all users behave in the same way – instead there exist subsets of like-minded users. By using different item-item models for these user subsets, we can capture differences in their preferences and this can lead to improved performance for top-N recommendations. In this work, we extend SLIM by combining global and local SLIM models. We present a method that computes the prediction scores as a user-specific combination of the predictions derived by a global and local item-item models. We present an approach in which the global model, the local models, their user-specific combination, and the assignment of users to the local models are jointly optimized to improve the top-N recommendation performance. Our experiments show that the proposed method improves upon the standard SLIM model and outperforms competing top-N recommendation approaches.

1. INTRODUCTION

Top-N recommender systems [3] are everywhere from online shopping websites to video portals. They provide users with a ranked list of N items they will likely be interested in, in order to encourage views and purchases.

Several algorithms for the top-N recommendation problem have been developed [18], including approaches that use latent-space models and approaches that rely on neighborhoods. The latent space methods [7] factorize the user-item matrix into lower rank user factor and item factor matrices, which represent both the users and the items in a common latent space. The neighborhood-based methods [8] (user-based or item-based) identify similar users or items. The latent-based methods have been shown to be superior for solving the rating prediction problem, whereas the neigborhood methods are shown to be better for the top-N recommendation problem [4, 8, 13, 16]. Among them, the item-based methods, which include item k-NN [8] and Sparse LInear Methods (SLIM) [16] have been shown to outperform the user-based schemes for the top-N recommendation task.

However, item-based methods have the drawback of estimating only a single model for all users. In many cases, there are differences in users' behavior, which cannot be captured by a single model. For example, there could be a pair of items that are extremely similar for a specific user subset, while they have low similarity for another user subset. By using a global model, the similarity between these items will tend to be towards some average value; thus, losing the high correlation of the pair for the first user subset.

In this paper we present a top-N recommendation method that extends the SLIM model in order to capture the differences in the preferences between different user subsets. Our method, which we call GLSLIM (Global and Local SLIM), combines global and local SLIM models in a personalized way and automatically identifies the appropriate user subsets. This is done by solving a joint optimization problem that estimates the different item-item models (global and local), their user-specific combination, and the assignment of the users to these models. Our experimental evaluation shows that GLSLIM outperforms competing top-N recommendation methods, reaching up to 17% improvement in recommendation quality.

The rest of the paper is organized as follows. Section 2 introduces the notation. Section 3 presents the related work. Section 4 presents the proposed model. Section 5 presents the evaluation methodology and the datasets. Section 6 presents the performance of the method. Finally, Section 7 provides some concluding remarks.

2. NOTATION

All vectors are represented by bold lower case letters and are column vectors (e.g., $\mathbf{p}$, $\mathbf{q}$). All matrices are represented by upper case letters (e.g., R, W). For a given matrix A, its ith row is represented by $\mathbf{a}_i^T$ and its jth column by $\mathbf{a}_j$. A predicted value is denoted by having a $\sim$ over it (e.g., $\tilde{r}$).

The number of users is denoted by n and the number of items is denoted by m. Matrix R is used to represent the *user-item implicit feedback* matrix of size $n \times m$. It shows which items the users have purchased/viewed/rated. Symbols u and i are used to denote individual users and items, respectively. If user u provided feedback for item i, the r_{ui} entry of R is 1, otherwise it is 0. We will use the term *rating* to refer to the non-zero entries of R, even though these entries can represent implicit feedback. We also refer to the items that the user has purchased/viewed/rated as

RecSys '16, September 15-19, 2016, Boston , MA, USA
 ISBN 978-1-4503-4035-9/16/09…$15.00
DOI: http://dx.doi.org/10.1145/2959100.2959185

rated items and to the rest as *unrated* items. The set of items that the user u has rated will be denoted by $\mathscr{R}_u$. We will use the symbol $\odot$ to denote the Hadamart product (element-wise multiplication).

3. RELATED WORK

3.1 Top-N Recommendation Methods

There has been extensive work in the area of top-N recommendation. Here we present a few notable works in the area that have advanced the state-of-the-art. Deshpande et. al. [8] developed a nearest neighbor item-based approach, which showed that item-based models lead to better top-N recommendation than user-based. Cremonesi et. al. [7] developed the pureSVD method, which uses a truncated SVD decomposition of matrix R to generate the top-N recommendations. Their work demonstrated that treating the missing entries as zero leads to better results than the matrix completion approaches. There is also the point of view of the learning-to-rank formulation [17].

3.1.1 Sparse LInear Method for top-N Recommendation (SLIM)

Ning et. al. [16] introduced SLIM, which was the first method to compute the item-item relations using statistical learning and has been shown to be one of the best approaches for top-N recommendation. SLIM estimates a sparse $m \times m$ aggregation coefficient matrix S. The recommendation score on an urated item i for user u is computed as a sparse aggregation of all the user's past rated items:

$$\tilde{r}_{ui} = \mathbf{r}_u^T \mathbf{s}_i, \tag{1}$$

where $\mathbf{r}_u^T$ is the row-vector of R corresponding to user u and $\mathbf{s}_i$ is the ith column vector of matrix S, that is estimated by solving the following optimization problem:

$$\begin{array}{ll} \underset{\mathbf{s}_i}{\text{minimize}} & \frac{1}{2}||\mathbf{r}_i - R\mathbf{s}_i||_2^2 + \frac{\beta}{2}||\mathbf{s}_i||_2^2 + \lambda||\mathbf{s}_i||_1, \\ \text{subject to} & \mathbf{s}_i \geq 0, \text{and} \\ & s_{ii} = 0. \end{array} \tag{2}$$

The constants β and λ are regularization parameters. The non-negativity constraint is used so that the vector estimated contains positive coefficients. The $s_{ii} = 0$ constraint makes sure that when computing the weights of an item, that item itself is not used as this would lead to trivial solutions.

3.2 Local Models for Recommendation

The idea of estimating multiple local models has been proposed in the work by O'connor et. al. [6], who performed rating prediction by clustering the rating matrix item-wise and estimating a separate local model for each cluster with nearest neighbor collaborative filtering.

Xu et al. [19] developed a method that co-clusters users and items and estimates a separate local model on each cluster, by applying different collaborative filtering methods; including the item-based neighborhood method. The predicted rating for a user-item pair is the prediction from the subgroup with the largest weight for the user.

Lee et al. [14,15] proposed a method that relies on the idea that the rating matrix is locally low-rank. First, neighborhoods are identified surrounding different anchor points of user-item pairs, based on a function that measures distances between pairs of users and items and then a local low-rank model is estimated for every neighborhood. The estimation is done in an iterative way where first the latent factors representing the anchor points are estimated and then based on the similarities of the observed entries to the anchor points, the latent factors are re-estimated, until convergence. A prediction is computed as a convex combination of local models, weighted by the similarity of the corresponding local anchor point to the user–item pair whose rating needs to be predicted.

GLSLIM differs from the earlier work in the following ways: (i) In all of the above-mentioned works, only local models are considered; while GLSLIM also computes a global model and has a personalization factor for each user determining the interplay between the global and the local information. (ii) GLSLIM updates the assignment of the users to subsets, allowing better local models to be estimated. (iii) Lee et. al. [14, 15] use user and item latent factors, while GLSLIM focuses on item-item models. (iv) In [6] the authors use item clusters, in [19] the authors use co-clusters and in [14, 15] they use user-item anchor points. Instead, GLSLIM uses user subsets.

4. PROPOSED APPROACH

4.1 Motivation

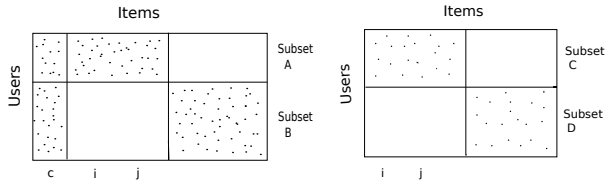
A global item-item model may not be sufficient to capture the preferences of a set of users, especially when there are user subsets with diverse and sometimes opposing preferences. An example of when local item-item models (item-item models capturing similarities in user subsets) will be beneficial and outperform the item-item model capturing the global similarities is shown in Figure 1. It portrays the training matrix R of two different datasets that both contain two distinct user subsets. Item i is the target item for which we will try to compute predictions. The predictions are computed by using an item-item cosine similarity-based method, in this motivation example.

In the left dataset, (Figure 1a) there exist some items which have been rated only by the users of one subset, but there is also a set of items which have been rated by users in both subsets. Items c and i will have different similarities when estimated for user-subset A, than when estimated for user-subset B, than for the overall matrix. Specifically, their similarity will be zero for the users of subset B (as item i is not rated by the users of that subset), but it will be non-zero for the users of subset A – and we can further assume without loss of generality that in this example it is high. Then, the similarity between i and c will be of average value when computed in the global case. So, estimating the local item-item similarities for the user subsets of this dataset will help capture the diverse preferences of user-subsets A and B, which would otherwise be missed if we only computed them globally.

However, when using item j to make predictions for item i, their similarity will be the same, either globally estimated, either locally for subset A, as they both have been rated only by users of subset A. The same holds for the dataset pictured in Figure 1b, as this dataset consists of user subsets who have no common rated items between them.

Although datasets like the one in Figure 1b cannot benefit from using local item-item similarity models, datasets such as the one pictured in Figure 1a can greatly benefit as they

can capture item-item similarities, which could be missed in the case of just having a global model.



(a) Overlapping rated items between user subsets (b) No common rated items between user subsets

Figure 1: (a) Local item-item models improve upon global item-item model. (b) Global item-item model and local models yield the same results.

4.2 Overview

In this work, we present our method GLSLIM, which computes top-N recommendations that utilize user–subset specific models and a global model. These models are jointly optimized along with computing the user assignments for them. We use SLIM for estimating the models. Thus, we estimate a global item-item coefficient matrix S and also k local item-item coefficient matrices S^{p_u}, where k is the number of user subsets and $p_u \in \{1, \ldots, k\}$ is the index of the user subset, for which we estimate the local matrix S^{p_u}. Every user can belong to one user subset.

The predicted rating of user u, who belongs to subset p_u, for item i will be estimated by:

$$\tilde{r}_{ui} = \sum_{l \in \mathscr{R}_u} g_u s_{li} + (1 - g_u) s_{li}^{p_u}. \tag{3}$$

The meanings of the various terms are as follows: The term s_{li} shows the global item-item similarity between the lth item rated by u and the target item i. The term $s_{li}^{p_u}$ depicts the item-item similarity between the lth item rated by u and target item i, corresponding to the local model of the user-subset p_u, to which target user u belongs. Finally, the term g_u is the personalized weight per user, which controls the interplay between the global and the local part. It lies in the interval $[0, 1]$, with 0 showing that the recommendation is affected only by the local model and 1 showing that the user u will use only the global model.

In order to perform top-N recommendation for user u, we compute the estimated rating $\tilde{r}_{ui}$ for every unrated item i with Equation 3. Then, we sort these values and we recommend the top-N items with the highest ratings to the user.

The estimation of the item-item coefficient matrices, the user assignments and the personalized weight is done with alternating minimization, which will be further explained in the following subsections.

4.3 Estimating the item-item models

We first separate the users into subsets with either a clustering algorithm (we used CLUTO [1]) or randomly. We initially set g_u to be 0.5 for all users, in order to have equal contribution of the global and the local part and we estimate the coefficient matrices S and S^{p_u}, with $p_u \in \{1, \ldots, k\}$. We use two vectors $\mathbf{g}$ and $\mathbf{g}'$ each of size n, where the vector $\mathbf{g}$ contains the personalized weight g_u for every user u and the vector $\mathbf{g}'$ contains the complement of the personalized weight $(1 - g_u)$ for every user u. When assigning the users into k subsets, we split the training matrix R into k training matrices R^{p_u} of size $n \times m$, with $p_u \in \{1, \ldots, k\}$. Every row u of R^{p_u} will be the uth row of R, if the user u who corresponds to this row belongs in the p_uth subset. If the user u does not belong to the p_uth subset, then the corresponding row of R^{p_u} will be empty, without any ratings. When estimating the local model S^{p_u}, only the corresponding R^{p_u} will be used. Following SLIM, the item–item coefficient matrices can be calculated per column, which allows for the different columns (of both the global and the local coefficient matrices) to be estimated in parallel. In order to estimate the ith column of S ($\mathbf{s}_i$) and S^{p_u} ($\mathbf{s}_i^{p_u}$) where $p_u \in \{1, \ldots, k\}$, GLSLIM solves the following optimization problem:

$$\begin{aligned}
\underset{\mathbf{s}_i, \{\mathbf{s}_i^1, \ldots, \mathbf{s}_i^k\}}{\text{minimize}} \quad & \tfrac{1}{2}||\mathbf{r}_i - \mathbf{g} \odot R\mathbf{s}_i - \mathbf{g}' \odot \textstyle\sum_{p_u=1}^{k} R^{p_u}\mathbf{s}_i^{p_u}||_2^2 + \\
& \tfrac{1}{2}\beta_g||\mathbf{s}_i||_2^2 + \lambda_g||\mathbf{s}_i||_1 + \\
& \textstyle\sum_{p_u=1}^{k} \tfrac{1}{2}\beta_l||\mathbf{s}_i^{p_u}||_2^2 + \lambda_l||\mathbf{s}_i^{p_u}||_1, \\
\text{subject to} \quad & \mathbf{s}_i \geq 0, \\
& \mathbf{s}_i^{p_u} \geq 0, \forall p_u \in \{1, \ldots, k\}, \\
& s_{ii} = 0, \\
& {s^{p_u}}_{ii} = 0, \forall p_u \in \{1, \ldots, k\},
\end{aligned} \tag{4}$$

where $\mathbf{r}_i$ is the ith column of R. β_g and β_l are the l_2 regularization weights corresponding to S and S^{p_u} $\forall p_u \in \{1, \ldots, k\}$ respectively. Finally λ_g and λ_l are the l_1 regularization weights controlling the sparsity of S and S^{p_u} $\forall p_u \in \{1, \ldots, k\}$, respectively.

By having different regularization parameters for the global and the local sparse coefficient matrices, we allow flexibility in the model. In this way, we can control through regularization which of the two components will play a more major part in the recommendation.

The constraint $s_{ii} = 0$ makes sure that when computing r_{ui}, the element r_{ui} is not used. If this constraint was not enforced, then an item would recommend itself. For the exact same reason, we enforce the constraint ${s^{p_u}}_{ii} = 0$, $\forall p_u \in \{1, \ldots, k\}$ for the local sparse coefficient matrices too.

The optimization problem of Equation 4 can be solved using coordinate descent and soft thresholding [9].

4.4 Finding the optimal assignment of users to subsets

After estimating the local models (and the global model), GLSLIM fixes them and proceeds with the second part of the optimization: updating the user subsets. While doing that, GLSLIM also determines the personalized weight g_u. We will use the term *refinement* to refer to finding the optimal user assignment to subsets.

Specifically, GLSLIM tries to assign each user u to every possible cluster, while computing the weight g_u that the user would have if assigned to that cluster. Then, for every cluster p_u and user u, the training error is computed. The cluster for which this error is the smallest is the cluster to which the user is assigned. If there is no difference in the training error, or if there is no cluster for which the training error is smaller, the user u remains at the initial cluster. The training error is computed for both the user's rated and unrated items.

Algorithm 1 GLSLIM

Assign $g_u = 0.5$, to every user u.
Compute the initial clustering of users with CLUTO [1].
while number of users who switched clusters $> 1\%$ of the total number of users **do**
  Estimate S and S^{p_u}, $\forall p_u \in \{1, \ldots, k\}$ with Equation 4.
  for all user u **do**
    for all cluster p_u **do**
      Compute g_u for cluster p_u with Equation 5.
      Compute the training error.
    end for
    Assign user u to the cluster p_u that has the smallest training error and update g_u to the corresponding one for cluster p_u.
  end for
end while

Table 1: Dataset characteristics.

Name	#Users	#Items	#Transactions	Density
groceries	63,034	15,846	2,060,719	0.21%
ml	69,878	10,677	10,000,054	1.34%
jester	57,732	150	1,760,039	20.32%
flixster	29,828	10,085	7,356,146	2.45%
netflix	274,036	17,770	31,756,784	0.65%

Columns corresponding to #users, #items and #transactions show the number of users, number of items and number of transactions, respectively, in each dataset. The column corresponding to density shows the density of each dataset (i.e., density=#transactions/(#users×#items)).

In order to compute the personalized weight g_u, we minimize the squared error of Equation 3 for user u who belongs to subset p_u, over all items i. By setting the derivative of the squared error to 0, we get:

$$g_u = \frac{\sum_{i=1}^{m} (\sum_{l \in \mathscr{R}_u} s_{li} - \sum_{l \in \mathscr{R}_u} s_{li}^{p_u})(r_{ui} - \sum_{l \in \mathscr{R}_u} s_{li}^{p_u})}{\sum_{i=1}^{m} (\sum_{l \in \mathscr{R}_u} s_{li} - \sum_{l \in \mathscr{R}_u} s_{li}^{p_u})^2}. \quad (5)$$

The overview of GLSLIM as well as the stopping criterion are shown in Algorithm 1.

5. EXPERIMENTAL EVALUATION

5.1 Datasets

We evaluated the performance of our method on different datasets, whose characteristics are shown in Table 1.

The *groceries* dataset corresponds to transactions of a local grocery store. Each user corresponds to a customer and the items correspond to the distinct products purchased over a period of one year. The *ml* dataset corresponds to MovieLens 10M dataset [12], which represents movie ratings. The *jester* dataset [10] corresponds to an online joke recommender system. The *flixster* dataset is a subset of the original Flixster dataset [2], which consists of movie ratings taken from the corresponding social movie site that allows users to share movie ratings and meet friends. The subset was created by keeping the users who have rated more than thirty items and the items that have been rated by at least twenty-five users. The *netflix* dataset is a subset of the original Netflix dataset [5], which contains anonymous movie ratings. The subset was created by keeping the users who have rated between thirty and five hundred items. All the ratings were converted to binary ratings, showing whether a user purchased/rated an item or not.

Algorithm 2 LSLIM

Compute the initial clustering of users with CLUTO.
while number of users who switched clusters $> 1\%$ of the total number of users **do**
  Estimate S^{p_u}, $\forall p_u \in \{1, \ldots, k\}$ with Equation 8.
  for all user u **do**
    for all cluster p_u **do**
      Compute the training error.
    end for
    Assign user u to the cluster p_u that has the smallest training error.
  end for
end while

Algorithm 3 GLSLIMr0

Assign $g_u = 0.5$, to every user u.
Compute the initial clustering of users with CLUTO.
while $diff > 0.01\%$ **do**
  Estimate S and S^{p_u}, $\forall p_u \in \{1, \ldots, k\}$ with Equation 4.
  $\forall$ user u compute g_u with Equation 5.
  Compute difference in the objective function ($diff$) between subsequent iterations.
end while

5.2 Evaluation Methodology

We employed leave-one-out cross-validation to evaluate the performance of the proposed model. For each user, we randomly selected an item, which we placed in the test set. The rest of the data comprised the training set.

We measure the performance by computing the number of times the single left-out item was in the top-N recommended items for this user and its position in that list. The quality measures used are the hit-rate (HR) and average-reciprocal hit rank (ARHR). HR is defined as

$$HR = \frac{\#hits}{\#users}, \quad (6)$$

and ARHR is defined as

$$ARHR = \frac{1}{\#users} \sum_{i=1}^{\#hits} \frac{1}{p_i}, \quad (7)$$

where "#users" is the total number of users (n), p is the position of the item in the list, where $p = 1$ specifies the top of the list, and "#hits" is the number of users whose item in the test set is present in the size-N recommendation list.

5.3 Proposed Methods

As our method contains multiple elements, we want to investigate how each of them impacts the recommendation performance. Thus, beyond GLSLIM, we also investigate the following methods:

- LSLIMr0, which stands for Local SLIM without refinement. In LSLIMr0, a separate item-item model is

estimated for each of the k user subsets. No global model is estimated; so there is no personalized weight g_u either. Specifically, the ith column of the p_uth local model S^{p_u} ($\mathbf{s}_i^{p_u}$) is estimated by solving the optimization problem:

$$\begin{aligned} \underset{\{\mathbf{s}_i^1,\ldots,\mathbf{s}_i^k\}}{\text{minimize}} \quad & \tfrac{1}{2}||\mathbf{r}_i - \textstyle\sum_{p_u=1}^{k} R^{p_u}\mathbf{s}_i^{p_u}||_2^2 + \\ & \textstyle\sum_{p_u=1}^{k} \tfrac{1}{2}\beta_l||\mathbf{s}_i^{p_u}||_2^2 + \lambda_l||\mathbf{s}_i^{p_u}||_1, \\ \text{subject to} \quad & \\ & \mathbf{s}_i^{p_u} \geq 0, \forall p_u \in \{1,\ldots,k\}, \\ & {s^{p_u}}_{ii} = 0, \forall p_u \in \{1,\ldots,k\}, \end{aligned} \tag{8}$$

where the meanings of the different terms are identical to those used in Equation 4.

In LSLIMr0, the initial assignment of users to subsets using CLUTO is the one used and never gets updated. The predicted rating for user u, who belongs to subset p_u, and item i is estimated by:

$$\tilde{r}_{ui} = \sum_{l \in \mathscr{R}_u} s_{li}^{p_u}. \tag{9}$$

The overall recommendation quality is computed as the weighted average of the item-item model performance in every subset.

- LSLIM, which stands for Local SLIM with refinement. In LSLIM, the predicted rating for user u and item i is also estimated by Equation 9 and the local models are estimated in the same way as in LSLIMr0. However, the users switch subsets and the local models get updated accordingly, until convergence. The algorithm for LSLIM is shown in Algorithm 2.
- GLSLIMr0, which stands for Global and Local SLIM without refinement. In GLSLIMr0, both a global model and separate item-item local models are estimated along with the per user weight g_u. However the assignment of users to subsets remains fixed. The algorithm for this method is shown in Algorithm 3.

5.4 Comparison Algorithms

The top-N recommendation algorithms that we compare against are: PureSVD [7], BPR-MF [17] and SLIM (described in Section 3.1.1).

PureSVD [7] is a popular top-N recommendation algorithm, which estimates the user–item matrix R by the factorization:

$$\tilde{R} = U\Sigma Q^T, \tag{10}$$

where U is an $n \times f$ orthonormal matrix, Q is an $m \times f$ orthonormal matrix and Σ is an $f \times f$ diagonal matrix containing the first f singular values.

BPR-MF [17] (Bayesian Personalized Ranking – Matrix Factorization) is a well-known top-N recommendation method, which uses the bayesian personalized ranking optimization criterion on matrix factorization. The BPR criterion focuses on finding the correct personalized ranking for all items to maximize the posterior probability.

For PureSVD, we used the SVDLIBC package[1], for BPR-MF, we used the LibRec package [11] and for SLIM, we used the SLIM package[2].

[1]https://tedlab.mit.edu/~dr/SVDLIBC/

[2]www-users.cs.umn.edu/~xning/slim/html

5.5 Model Selection

We performed an extensive search over the parameter space of the various methods, in order to find the set of parameters that gives us the best performance for each method.

We only report the performance corresponding to the parameters that lead to the best results. The l_1 and l_2 regularization parameters were chosen from the set of values: $\{0.1, 1, 3, 5, 7, 10\}$. The number of clusters examined took on the values: {2, 3, 5, 10, 15, 20, 25, 30, 35, 40, 45, 50, 60, 70, 80, 90, 100 and 150}.

For PureSVD, the number of singular values f tried lie in the interval $[10, 5000]$. For BPR-MF, the number of factors we used in order to get the best results lie in the interval $[1, 10000]$. The values of the learning rate that we tried are: $\{0.0001, 0.001, 0.01, 0.1\}$. The values of the regularization we tried are: $\{0.0001, 0.001, 0.01, 0.1\}$.

6. EXPERIMENTAL RESULTS

In this section we present the results of our experiments. The following questions will be answered: (i) How do the proposed methods compare between them? (ii) How does our method compare against competing top-N recommendation methods? (iii) What is the time complexity of our method?

6.1 Performance of the proposed methods

The comparison of our proposed approaches (LSLIMr0, LSLIM, GLSLIMr0 and GLSLIM) in terms of HR and ARHR is shown in Table 2. Overall, we can see that the general pattern is that GLSLIM is the best-performing method, followed by GLSLIMr0 and LSLIM, while LSLIMr0 is the approach with the lowest performance.

By comparing these methods, we can see the relative benefits provided by the different components of GLSLIM. The comparisons of LSLIMr0 with GLSLIMr0 and also of LSLIM with GLSLIM show the benefit of adding a global model with a personalized weight g_u. The comparisons of LSLIMr0 with LSLIM, and also between GLSLIMr0 and GLSLIM demonstrate the benefit of allowing users to switch subsets. We can see that both these components improve the performance. However, in all of the datasets but *ml*, the relative gain of considering a global model beyond the local item-item models and also computing a personalized weight g_u is higher than the gain of allowing users to switch subsets. When all of the components are combined, as in the case of GLSLIM, we get the best performance, both in terms of HR and ARHR.

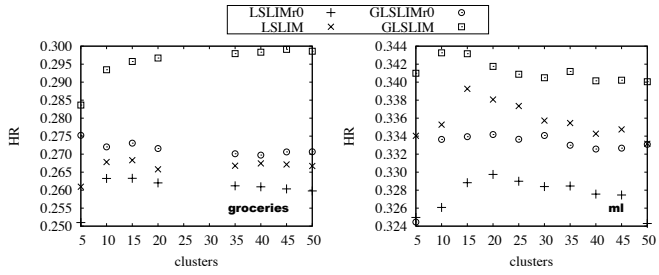


Figure 2: The effect of the number of clusters on the performance

Table 2: Comparison between our proposed approaches.

Comparison in terms of Hit Rate (HR)

	LSLIMr0				LSLIM				GLSLIMr0						GLSLIM					
Dataset	Cls	β_l	λ_l	HR	Cls	β_l	λ_l	HR	Cls	β_g	β_l	λ_g	λ_l	HR	Cls	β_g	β_l	λ_g	λ_l	HR
groceries	15	5	0.1	0.263	15	5	1	0.268	3	5	5	1	1	0.280	100	5	5	1	1	**0.304**
ml	20	5	1	0.329	15	5	3	0.339	15	7	3	1	5	0.335	10	10	7	1	1	**0.345**
jester	5	5	0.1	0.898	10	0.1	0.1	0.916	20	7	1	10	1	0.929	10	10	10	10	0.1	**0.940**
flixster	3	1	2	0.248	3	0.1	3	0.250	3	1	1	5	5	0.254	3	1	1	1	5	**0.255**
netflix	10	1	5	0.238	10	1	5	0.241	20	1	1	5	5	0.243	5	1	1	5	5	**0.245**

Comparison in terms of Average Reciprocal Hit Rate (ARHR)

	LSLIMr0				LSLIM				GLSLIMr0						GLSLIM					
Dataset	Cls	β_l	λ_l	ARHR	Cls	β_l	λ_l	ARHR	Cls	β_g	β_l	λ_g	λ_l	ARHR	Cls	β_g	β_l	λ_g	λ_l	ARHR
groceries	3	3	0.1	0.133	15	3	3	0.135	3	5	5	1	1	0.144	100	5	5	1	1	**0.155**
ml	25	7	2	0.163	15	7	3	0.167	15	7	7	1	3	0.166	10	10	7	1	1	**0.170**
jester	10	0.1	0.1	0.775	10	10	5	0.804	150	1	1	1	1	0.820	100	1	1	1	1	**0.835**
flixster	3	0.1	2	0.121	3	1	3	0.122	3	5	5	1	1	0.125	3	1	1	1	5	**0.126**
netflix	20	0.1	5	0.113	10	3	10	0.114	20	1	1	5	5	0.115	5	1	1	5	5	**0.116**

For each method, the columns correspond to the best HR and ARHR and the parameters for which they are achieved. The parameters are: the number of clusters, the global l_2 regularization parameter β_g, the local l_2 regularization parameter β_l, the global l_1 regularization parameter λ_g and the local l_1 regularization parameter λ_l. The bold numbers show the best HR/ARHR achieved, per dataset.

6.1.1 *Sensitivity on the number of Clusters*

Figure 2 shows how the number of clusters affects the HR for GLSLIM and its variants in the *groceries* and *ml* datasets. The trends are the same for the rest of the datasets and for the metric ARHR. We can see that GLSLIM outperforms the rest of the methods for all clusters. Also, we should note that for all datasets GLSLIM can achieve at least 95% of its best performance for only ten clusters, outperforming its closest competing method. This is the case even for the datasets where the best performance occured at a much bigger number of clusters.

6.1.2 *Initializing with random user subsets*

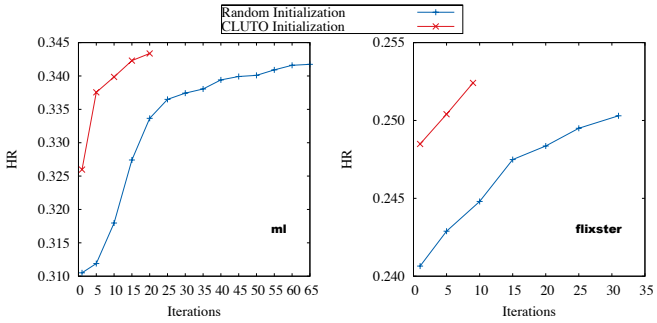


Figure 3: Comparing CLUTO initialization with random initialization of user subsets

The results presented up to this point have been obtained by initializing the user subsets with a user clustering algorithm (we used CLUTO[3] [1]). In order to show that the good performance of GLSLIM is not dependent on the clustering algorithm used, we present the performance when the initialization of the user subsets is random.

In Figure 3, we can see for the *ml* and *flixster* datasets, the HR achieved across iterations with the two different ways of initialization, for the same regularization and for ten clusters. The same trends hold for ARHR and for different regularizations, clusters and the rest of the datasets.

We can see that the HR of the first iteration with random initialization is lower than the HR of the first iteration when initializing with CLUTO. This is expected, as in the first iteration, only the global and local models are estimated; no personalization nor cluster refinement has been done yet. Thus, the local models estimated from CLUTO are more meaningful than the local models on random user subsets.

As the iterations progress and cluster refinement is done, we see that the HR increases. In the converged state, the final HR achieved is very similar with both initializations. However, when starting from random user subsets, more iterations are needed until convergence. We can then conclude that our method is able to estimate the local models and reach convergence, even with random initialization.

6.1.3 *The interplay between the global and the local part of the model*

In order to see how the local models affect the recommendation performance, we look at the l_1 norm of the global model S and the local models S^{p_u} in the beginning of the algorithm and when the algorithm has converged.

Figure 4 shows these l_1 norms for 5, 50 and 100 clusters, for the *groceries* and *ml* datasets. We can see that the l_1 norm of the global model S is small and it remains small for all possible clusters and throughout the iterations of the algorithm. For the local models S^{p_u}, their l_1 norm is larger than the l_1 norm of the global model. As the number of clusters increases, the l_1 norm of the local models increases. In addition, the l_1 norm of the local models in the converged state is larger than the l_1 norm of the local models in the

[3]We used the clustering program *vcluster*. The similarity function considered was the cosine similarity. All the other parameters were the default ones.

beginning. This shows that the effect of local information on the models is major and it becomes greater as the iterations progress and as the number of clusters increases.

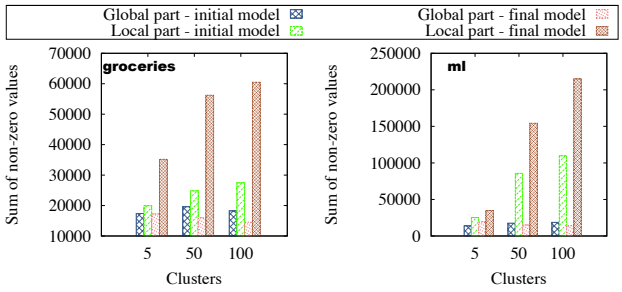


Figure 4: How the l_1 norm of the global model S and local models S^{p_u} changes from the beginning of the algorithm until convergence.

6.1.4 Top-N

The results presented throughout the paper show the performance of our algorithms for a list of size 10. The recommendation list can be of different sizes. In this section, we describe how the performance of our method is affected by using lists of sizes 5, 15 and 20 as well. We choose N to be quite small because users do not look past the very top presented recommendations in a list, anyway.

In Figure 5, we can see the HR of GLSLIM, while using the parameters with the best results as presented in Table 2 for the different sizes of top-N list. We can see that as N

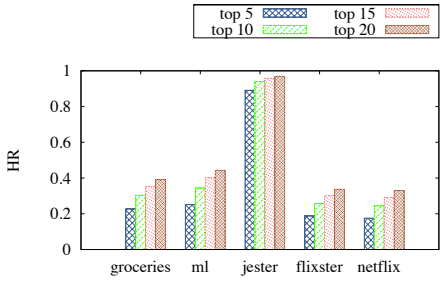


Figure 5: Varying the size of the top-N list.

increases, the performance of our method increases as well, which is expected, as there is higher probability that the hidden item of our test set will be in the top-N list. The impact of the size of the recommendation list N on ARHR is similar to the one shown in Figure 5.

6.2 Performance against Competing Approaches

Table 3 presents the performance of the competing algorithms PureSVD, BPR-MF and SLIM versus the performance of our best method, which is GLSLIM. The abovementioned table presents the best HR and ARHR achieved, along with the set of parameters for which they were achieved.

We can see that GLSLIM outperforms all competing approaches for all datasets, as seen in Table 3. By comparing Tables 2 with 3, we can also see that LSLIMr0, which is our simplest method, still outperforms the best competing approach, which shows that using multiple item-item models helps top-N recommendation quality.

6.3 Time Complexity

We will use $O(SLIM_i(R))$ to denote the computational cost of estimating the ith column of S. Then, the complexity of estimating the ith column of S and $S^1, \ldots, S^k$, is $O(SLIM_i(R)) + O(SLIM_i(R^1)) + \ldots + O(SLIM_i(R^k))$, where $R^1, \ldots, R^k$ are non-overlapping submatrices of R. Since in order to estimate the ith column of S, we need to touch every non-zero in the input matrix R, the complexity $O(SLIM_i(R))$ is at least linear in the number of non-zeros (nnz). We can then say that the complexity of estimating the ith column for the submatrices $R^1, \ldots, R^k$ is less than or equal to the complexity of solving it on the matrix R: $O(SLIM_i(R)) \geq O(SLIM_i(R^1)) + \ldots + O(SLIM_i(R^k))$. As a result, the complexity of Equation 4 is the dominant term $O(SLIM_i(R))$. Since the regression problem of Equation 4 needs to be solved for all m columns (items), the complexity of estimating the global and local models is $O(m \times SLIM_i(R))$. The complexity of updating the cluster assignment for each of the n users, after trying to assign them to each of the k clusters (lines $5-11$ of Algorithm 1), is $O(nmk)$, since both the computation of the training error and g_u is $O(m)$. Thus, the per iteration cost of GLSLIM is $O(m(SLIM_i(R) + nk))$. The number of iterations until GLSLIM converges is typically small, as can be seen in Section 6.1.2.

7. CONCLUSION

In this paper, we proposed a method to improve upon top-N recommendation item-based schemes, by capturing the differences in the preferences between different user subsets, which cannot be captured by a single model. For this purpose, we estimate a separate local item-item model for every user subset, in addition to the global item-item model. The proposed method allows cluster refinement, in the context of users being able to switch the subset they belong to, which leads to updating the local model estimated for this subset, as well as the global model. The method is personalized, as we compute for all users their own personal weight, defining the degree to which their top-N recommendation list will be affected from global or local information. We performed different experiments, which show that our method outperforms competing top-N recommender methods, indicating the value of using multiple item-item models.

Acknowledgments

This work was supported in part by NSF (OCI-1048018, IIS-1247632, IIP-1414153, IIS-1447788), Army Research Office (W911NF-14-1-0316), Intel Software and Services Group, and the Digital Technology Center at the University of Minnesota. Access to research and computing facilities was provided by the Digital Technology Center and the Minnesota Supercomputing Institute.

8. REFERENCES

[1] Cluto clustering toolkit. http://glaros.dtc.umn.edu/gkhome/cluto/cluto/overview.

[2] Flixster dataset. http://http://www.cs.sfu.ca/~sja25/personal/datasets/.

[3] G. Adomavicius and A. Tuzhilin. Toward the next generation of recommender systems: A survey of the state-of-the-art and possible extensions. *Knowledge and Data Engineering, IEEE Transactions on*, 17(6):734–749, 2005.

Table 3: Comparison with competing approaches.

Comparison in terms of Hit Rate (HR)

	PureSVD		BPR-MF				SLIM			GLSLIM					
Dataset	f	HR	factors	lrnrate	reg	HR	β	λ	HR	Cls	β_g	β_l	λ_g	λ_l	HR
groceries	738	0.134	3000	0.01	0.001	0.214	5	0.1	0.259	100	5	5	1	1	**0.304**
ml	64	0.295	5000	0.01	0.01	0.240	7	5	0.312	10	10	7	1	1	**0.345**
jester	25	0.860	300	0.01	0.01	0.903	3	0.1	0.878	10	10	10	10	0.1	**0.940**
flixster	90	0.194	4000	0.01	0.001	0.200	0.1	2	0.242	3	1	1	1	5	**0.255**
netflix	50	0.204	5000	0.01	0.01	0.210	0.1	5	0.231	5	1	1	5	5	**0.245**

Comparison in terms of Average Reciprocal Hit Rate (ARHR)

	PureSVD		BPR-MF				SLIM			GLSLIM					
Dataset	f	ARHR	factors	lrnrate	reg	ARHR	β	λ	ARHR	Cls	β_g	β_l	λ_g	λ_l	ARHR
groceries	700	0.059	3100	0.01	0.001	0.099	3	0.1	0.130	100	5	5	1	1	**0.155**
ml	56	0.139	7000	0.01	0.01	0.105	5	2	0.151	10	10	7	1	1	**0.170**
jester	15	0.740	100	0.01	0.01	0.766	7	0.1	0.755	100	1	1	1	1	**0.835**
flixster	80	0.086	4000	0.01	0.001	0.089	0.1	2	0.116	3	1	1	1	5	**0.126**
netflix	50	0.091	5000	0.01	0.01	0.100	5	5	0.107	5	1	1	5	5	**0.116**

For each method, columns corresponding to the best HR and ARHR and the set of parameters with which they are achieved are shown. For PureSVD the parameter is the number of singular values (f). For BPR-MF, the parameters are: the number of factors, the learnrate and the regularization. For SLIM, the parameters are the $l2$ regularization parameter β and the $l1$ regularization parameter λ. For GLSLIM, the parameters are the number of clusters, global β (β_g), local β (β_l), global λ (λ_g) and local λ (λ_l). Bold numbers indicate the best HR and ARHR across the different algorithms, for every dataset.

[4] F. Aiolli. A preliminary study on a recommender system for the million songs dataset challenge. *PREFERENCE LEARNING: PROBLEMS AND APPLICATIONS IN AI*, page 1, 2012.

[5] J. Bennett and S. Lanning. The netflix prize. In *Proceedings of KDD cup and workshop*, volume 2007, page 35, 2007.

[6] M. Connor and J. Herlocker. Clustering items for collaborative filtering. In *Proceedings of the ACM SIGIR workshop on recommender systems*, volume 128. Citeseer, 1999.

[7] P. Cremonesi, Y. Koren, and R. Turrin. Performance of recommender algorithms on top-n recommendation tasks. In *Proceedings of the fourth ACM conference on Recommender systems*, pages 39–46. ACM, 2010.

[8] M. Deshpande and G. Karypis. Item-based top-n recommendation algorithms. *ACM Transactions on Information Systems (TOIS)*, 22(1):143–177, 2004.

[9] J. Friedman, T. Hastie, and R. Tibshirani. Regularization paths for generalized linear models via coordinate descent. *Journal of statistical software*, 33(1):1, 2010.

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

[11] G. Guo, J. Zhang, Z. Sun, and N. Yorke-Smith. Librec: A java library for recommender systems. In *Posters, Demos, Late-breaking Results and Workshop Proceedings of the 23rd International Conference on User Modeling, Adaptation and Personalization*, 2015.

[12] J. L. Herlocker, J. A. Konstan, A. Borchers, and J. Riedl. An algorithmic framework for performing collaborative filtering. In *Proceedings of the 22nd annual international ACM SIGIR conference on Research and development in information retrieval*, pages 230–237. ACM, 1999.

[13] S. Kabbur, X. Ning, and G. Karypis. Fism: Factored item similarity models for top-n recommender systems. 2013.

[14] J. Lee, S. Bengio, S. Kim, G. Lebanon, and Y. Singer. Local collaborative ranking. In *Proceedings of the 23rd international conference on World wide web*, pages 85–96. ACM, 2014.

[15] J. Lee, S. Kim, G. Lebanon, and Y. Singer. Local low-rank matrix approximation. In *Proceedings of The 30th International Conference on Machine Learning*, pages 82–90, 2013.

[16] X. Ning and G. Karypis. Slim: Sparse linear methods for top-n recommender systems. In *Data Mining (ICDM), 2011 IEEE 11th International Conference on*, pages 497–506. IEEE, 2011.

[17] S. Rendle, C. Freudenthaler, Z. Gantner, and L. Schmidt-Thieme. Bpr: Bayesian personalized ranking from implicit feedback. In *Proceedings of the Twenty-Fifth Conference on Uncertainty in Artificial Intelligence*, pages 452–461. AUAI Press, 2009.

[18] F. Ricci and B. Shapira. *Recommender systems handbook*. Springer, 2011.

[19] B. Xu, J. Bu, C. Chen, and D. Cai. An exploration of improving collaborative recommender systems via user-item subgroups. In *Proceedings of the 21st international conference on World Wide Web*, pages 21–30. ACM, 2012.