Local Latent Space Models for Top-N Recommendation

Evangelia Christakopoulou
Computer Science & Engineering
University of Minnesota
evangel@cs.umn.edu

George Karypis
Computer Science & Engineering
University of Minnesota
karypis@umn.edu

ABSTRACT

Users' behaviors are driven by their preferences across various aspects of items they are potentially interested in purchasing, viewing, etc. Latent space approaches model these aspects in the form of latent factors. Although such approaches have been shown to lead to good results, the aspects that are important to different users can vary. In many domains, there may be a set of aspects for which all users care about and a set of aspects that are specific to different subsets of users. To explicitly capture this, we consider models in which there are some latent factors that capture the shared aspects and some user subset specific latent factors that capture the set of aspects that the different subsets of users care about.

In particular, we propose two latent space models: rGLSVD and sGLSVD, that combine such a global and user subset specific sets of latent factors. The rGLSVD model assigns the users into different subsets based on their rating patterns and then estimates a global and a set of user subset specific local models whose number of latent dimensions can vary.

The sGLSVD model estimates both global and user subset specific local models by keeping the number of latent dimensions the same among these models but optimizes the grouping of the users in order to achieve the best approximation. Our experiments on various real-world datasets show that the proposed approaches significantly outperform state-of-the-art latent space top-N recommendation approaches.

ACM Reference Format:
Evangelia Christakopoulou and George Karypis. 2018. Local Latent Space Models for Top-N Recommendation. In *KDD '18: The 24th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining, August 19–23, 2018, London, United Kingdom.* ACM, New York, NY, USA, 9 pages. https://doi.org/10.1145/3219819.3220112

1 INTRODUCTION

Top-N recommender systems identify a small number of N items that a user will find useful to purchase, view, like, click etc., among a large collection of such items by leveraging historical information from that and other users. They are wildly popular, ranging from Netflix movie recommendations, to Amazon product recommendations, to Facebook friend recommendations etc.

KDD '18, August 19–23, 2018, London, United Kingdom

ACM ISBN 978-1-4503-5552-0/18/08…$15.00
https://doi.org/10.1145/3219819.3220112

The methods developed to tackle the top-N recommendation task broadly fall into two categories: the neighborhood-based (which focus either on users or items) and the latent space ones. The latent space methods [8] compute a low-rank factorization of the user-item matrix into user and item factor matrices, which represent both the users and the items in a common latent space. The neighborhood-based methods [9] (user-based or item-based) focus on identifying similar users/items based on the rating matrix.

Item-item approaches can suffer from inefficient personalization; on the other hand latent space approaches do not face this issue, as an increase of the rank leads to more latent features estimated for every user. However, they assume that users base their behavior on a set of aspects, shared by all of them, which they model by estimating a set of shared latent factors. We believe that this user model is limiting. We instead propose that a user determines his/her preferences based on some aspects shared by all; i.e., *global* aspects, and on some more specific aspects, that are shared by similar subsets of users, i.e., *local* aspects. For example, a young girl can decide on a piece of clothing to purchase, based on some general aspects, such as whether it is in good condition, and also on some more specific aspects, such as whether this item of clothing is fashionable for girls her age. Thus, such a preference model contains both global as well as local elements.

In this paper, we propose to explicitly encode such structure, by estimating both a global low-rank model and multiple user subset specific low-rank models. We propose two approaches: rGLSVD (Global and Local Singular Value Decomposition with varying ranks) and sGLSVD (Global and Local Singular Value Decomposition with varying subsets). The approach rGLSVD considers fixed user subsets but allows for different local models to have varying ranks. The approach sGLSVD solves a joint optimization problem, which estimates local models of fixed ranks, while automatically determining the various subsets of users. The two approaches explore different ways to learn the local low-rank representations that will achieve the best top-N recommendation quality for the users. Estimating such structure with a global latent model can be difficult, since the data at hand are often very sparse.

We evaluated our methods on a variety of real-world datasets and the results showed that our proposed approaches are of similar performance and they outperform competing top-N latent space methods, on average by 13%. Also, the comparison of sGLSVD to the item-item approach GLSLIM, which also estimates a global and multiple local models, shows that while GLSLIM achieves better top-N recommendation results, sGLSVD is an order of magnitude faster and its relative percentage of improvement above baseline global method PureSVD is higher than the improvement of GLSLIM beyond SLIM.

The rest of the paper is organized as follows. Section 2 introduces the notation used. Section 3 describes existing work in literature

that is related to ours. Section 4 presents the proposed approaches: rGLSVD and sGLSVD. Section 5 shows the experimental evaluation we followed. Section 6 presents the experimental results. Finally, Section 7 concludes this work and provides some future directions.

2 NOTATION

We represent all column vectors by bold lower case letters (e.g., $\mathbf{p}$, $\mathbf{q}$). We represent all row vectors by bold lower case letters and having the transpose superscript ^{T}, (e.g., $\mathbf{p}^T$). We represent all matrices by bold upper case letters (e.g., $\mathbf{R}$, $\mathbf{A}$, $\mathbf{U}$). For a given matrix $\mathbf{A}$, we use the notation $\mathbf{a}_i^T$ to refer to its ith row, and $\mathbf{a}_j$ to refer to its jth column. We note the element of matrix $\mathbf{A}$ that corresponds to the ith row and jth column as a_{ij}. We use calligraphic letters to denote sets. We denote a predicted value, by having a $\sim$ over it (e.g., $\tilde{r}$).

We denote the number of users by n and the number of items by m. We use symbols u and i to denote individual users and items, respectively. We use matrix $\mathbf{R}$ to represent the user-item implicit feedback matrix of size $n \times m$, containing the items that the users have purchased/viewed/rated. The vector $\mathbf{r}_u^T$ contains the implicit behavior of user u and the vector $\mathbf{r}_i$ contains the implicit feedback of all users for item i. If user u provided feedback for item i, the r_{ui} entry of $\mathbf{R}$ is 1, otherwise it is 0. We use the term rating to refer to the non-zero entries of $\mathbf{R}$, even though these entries can represent implicit feedback. We also refer to the items that the user has purchased/viewed/rated as rated items and to the rest as unrated items. We denote the set of items that the user u has rated with $\mathcal{R}_u$. The number of items to be recommended is N.

3 RELATED WORK

3.1 Latent Space Approaches for Top-N Recommendation

There are a lot of latent-based approaches used for top-N recommendation [8, 14, 17, 23, 28] that have been shown to have good top-N recommendation quality. The latent-based approaches perform a low-rank factorization of the user-item feedback matrix into two matrices that represent user and item characteristics in a common latent space.

Among the latent-based methods for the top-N recommendation task, a notable one is the **PureSVD** method developed by Cremonesi et al. [8], which performs a truncated Singular Value Decomposition of rank f of the matrix $\mathbf{R}$ to generate the recommendations. In order to do so, the authors proposed to treat the missing entries as zeros.

Many of the latent space approaches developed for the top-N recommendation task focus on ranking the unrated items, instead of accurately estimating the missing values [7, 24, 26]. Among those, a popular one is the Bayesian Personalized Ranking - Matrix Factorization approach (**BPRMF**) [24], which focuses on finding the correct personalized ranking for all items to maximize the posterior probability.

3.2 Local Models for Top-N Recommendation

The idea of using multiple local models is well researched in the literature [5, 10, 18, 19, 22, 25, 29]. Here we discuss these approaches that are most relevant to our work, along with how our work differs from them.

Koren [16] proposes a combined model, which estimates every user-item rating r_{ui} as a combination of a global latent space model and local neighborhood interactions. This method, although utilizing both global and local components, estimates the local components through standard k-NN approaches, thus being very different from our approach which is a latent space approach and estimates multiple local latent space models.

Weston et al. [28] model a user with T latent vectors, each of dimension m, to model the user's latent tastes, while every item has a single latent vector of size m. In order to compute the prediction for each user and item, they compute the maximum possible score after multiplying each of the T user latent vectors to the item one. Although allowing users to have different sets of latent factors, instead of a global one, their method differs from ours, as it does not assume the main user model that we have: that everyone has a set of global latent factors and a set of user subset specific latent factors.

Lee et al. [18, 19] propose a method called Local Low-Rank Matrix Approximation (**LLORMA**). They assume that the rating matrix $\mathbf{R}$ is locally low-rank and is approximated as a weighted sum of low-rank matrices. In their method, they identify neighborhoods surrounding different anchor points of user-item pairs, based on a function that measures distances between pairs of users and items and then they estimate a local low-rank model 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, they are re-estimated, until convergence. The predicted rating of a target user-item pair is calculated as a weighted combination of the estimated local models, where the weight is the similarity of the pair to the anchor points. Lee et al. have tested this approach with both a squared error objective [19] and a pairwise ranking objective [18]. Their approach, although also using multiple local latent space models, does not estimate global factors for all users. Also, the local models are created differently; theirs are based on anchor points, while in our approaches they capture behaviors of user subsets. Also, their method does not allow varying ranks of local models or user subsets to be updated.

Finally, Christakopoulou and Karypis [6] propose a method called **GLSLIM**, which estimates a personalized combination of global and multiple local *item-item* models, allowing for the user subsets to be updated. GLSLIM is an item-item approach, while the method proposed in this paper is a *latent space* approach, thus the underlying user model differs. Also, in this work, we explore both changing user subsets and varying ranks among the local models to allow the best low-rank representation for every user to be estimated.

4 GLOBAL AND LOCAL LATENT MODELS

4.1 Motivation

Latent space approaches assume that every user's behavior can be described by a set of aspects, which are shared by all the users. However, consider the following scenario. When deciding on which restaurant to go to, people generally tend to agree on a set of aspects

Algorithm 1 rGLSVD

1: Assign $g_u = 0.5$ for every user u.
2: Compute the initial assignment of users to subsets.
3: **while** (users whose g_u changed more than 0.01) > 1% of the total users **do**
4: Construct $\mathbf{R}^g$ and $\mathbf{R}^c, \forall c \in \{1, \ldots, k\}$, as discussed in Section 4.3.
5: Compute a truncated SVD of rank f^g on $\mathbf{R}^g$.
6: **for all** user subset c **do**
7: Compute a truncated SVD of rank f^c on $\mathbf{R}^c$.
8: **end for**
9: **for all** user u **do**
10: Compute his user-specific weight g_u with Equation (3).
11: **end for**
12: **end while**

that are important: how clean the restaurant is, how delicious the food is. However, there can be other factors which are important to only a subset of users, such as if vegan options are available and if live music exists. Users of a different subset might be interested in other factors, such as what is the average waiting time, and how big the portions are. We postulate that a user model that assumes that users' preferences can be described by some aspects which are common to all but also some additional user subset specific aspects, can better capture user behavior such as the one described above.

As the available data is generally sparse, estimating the global and user subset specific factors from a global low-rank model could be difficult. Thus, we propose to impose such a structure explicitly, by estimating a global latent space model, and multiple user subset specific latent space models.

4.2 Overview

In order to explicitly model this type of user model described above, we present two approaches: Global and Local Singular Value Decomposition with varying ranks (rGLSVD) and Global and Local Singular Value Decomposition with varying subsets (sGLSVD), which estimate a personalized combination of the global and local low-rank models. Both approaches utilize PureSVD as the underlying model, as it has been shown to have good top-N recommendation performance, while being scalable [8, 16].

The rGLSVD approach assigns the users into different subsets based on their rating patterns, which remain fixed, and then estimates a global model and multiple user subset specific local models whose number of latent dimensions can vary. The sGLSVD model estimates a global model and multiple user subset specific local models by keeping the number of latent dimensions the same among the different local models, but optimizes the grouping of the users in order to achieve the best approximation.

The reason why we do not combine the two methods is because if we allowed in the joint optimization problem we solve for both local models to have varying ranks and user subsets to be updated, we would face overfitting issues. Overfitting would occur because the vast majority of users would always be assigned to the subset with the highest corresponding number of local dimensions, which would result in the smallest training error.

Algorithm 2 sGLSVD

1: Assign $g_u = 0.5$ for every user u.
2: Compute the initial assignment of users to subsets.
3: **while** number of users switching subsets > 1% of the total users **do**
4: Construct $\mathbf{R}^g$ and $\mathbf{R}^c, \forall c \in \{1, \ldots, k\}$, as discussed in Section 4.3.
5: Compute a truncated SVD of rank f^g on $\mathbf{R}^g$.
6: **for all** user subset c **do**
7: Compute a truncated SVD of *the same rank* f^c on $\mathbf{R}^c$.
8: **end for**
9: **for all** user u **do**
10: **for all** user subset c **do**
11: Project user u on user subset c with Equation 4.
12: Compute his user-specific weight g_u for user subset c with Equation 3
13: Compute the training error.
14: **end for**
15: Assign u to the user subset c with the corresponding smallest training error and update his user-specific weight g_u to the corresponding one for user subset c.
16: **end for**
17: **end while**

4.3 Estimating the rGLSVD model

A high-level overview of rGLSVD is shown in Algorithm 1. It follows an alternating optimization approach whose steps consist of estimating the global and user subset specific latent factors, and then estimating the user-specific weights, and repeating these steps until convergence.

We first initialize the user-specific weights g_u with the value 0.5 for all users, so that the global and local components will be of equal contribution (Line 1). The user-specific weights can take values from 0 to 1, where 0 shows that only local models are utilized, and 1 that only a global model is used.

We separate the users into k subsets with a clustering algorithm (Line 2). We use CLUTO by Karypis [15]. Every user can belong to one subset.

In line 4 of the Algorithm 1, we construct the global $n \times m$ training matrix $\mathbf{R}^g$ by stacking the vectors $g_u \mathbf{r}_u^T$, for all users. For every subset $c \in \{1, \ldots, k\}$, we construct the corresponding local training matrix $\mathbf{R}^c$ by stacking the vectors $(1 - g_u)\mathbf{r}_u^T$, for all users u belonging to subset c. Every matrix $\mathbf{R}^c$ has m columns and as many rows as the number of users belonging to subset c, which we note as n^c.

We then compute a truncated singular value decomposition on the global matrix $\mathbf{R}^g$ of rank f^g (line 5), which allows us to estimate the global user factors, in the following way:

$$\tilde{\mathbf{R}}^g = \mathbf{P}\Sigma_{f^g}\mathbf{Q}^T, \tag{1}$$

where $\mathbf{P}$ is an $n \times f^g$ orthonormal matrix showing the global user factors, $\mathbf{Q}$ is an $m \times f^g$ orthonormal matrix showing the global item factors, and Σ_{f^g} is an $f^g \times f^g$ diagonal matrix containing the f^g largest singular values.

For every subset c, we compute a truncated singular value decomposition on $\mathbf{R}^c$ of rank f^c (line 7):

$$\tilde{\mathbf{R}}^c = \mathbf{P}^c \Sigma_{f^c} \mathbf{Q}^{cT}, \quad (2)$$

where $\mathbf{P}^c$ is a $n^c \times f^c$ matrix containing the local user factors which are specific to subset c, and $\mathbf{Q}^c$ is a $m \times f^c$ matrix containing the local item factors of subset c. Note that the ranks f^c can be different for each local subset c.

Overall, we estimate a global user latent factor matrix $\mathbf{P}$, a global item latent factor matrix $\mathbf{Q}$, k user subset specific user latent factor matrices $\mathbf{P}^c$ and k user subset specific item latent factor matrices $\mathbf{Q}^c$. After estimating the latent factor matrices, we proceed to updating the user-specific weights.

We compute the user-specific weights g_u (line 10), by minimizing the squared error for every user u over all items (both rated and unrated ones). After setting the derivative of the squared error to 0, we get:

$$g_u = \frac{\sum_{i=1}^{m} (a-b)(r_{ui}-b)}{\sum_{i=1}^{m} (a-b)^2}, \quad (3)$$

where $a = \frac{1}{g_u} \mathbf{p}_u^T \Sigma_{f^g} \mathbf{q}_i$ and $b = \frac{1}{1-g_u} \mathbf{p}_u^{cT} \Sigma_{f^c} \mathbf{q}_i^c$.

The user-specific weights of the users for rGLSVD can be estimated independently in parallel.

The latent factor estimation and the update of the user-specific weights are repeated until convergence. We consider that the algorithm is converged, when the number of users who have modified personalized weights with respect to the previous iteration is less than or equal to 1% of the total number of users (line 3).

4.4 Estimating the sGLSVD model

A high-level overview of sGLSVD is shown in Algorithm 2. Like rGLSVD, it also follows an alternating optimization approach, with a lot of similar steps. The key differences are that the ranks f^c are the same across the local subsets c (line 5) and that the user subsets can be updated (lines 7-13). The sGLSVD approach is an iterative approach that is reminiscent of k-means-style cluster refinement.

We evaluate the change in the training error resulting by moving each user to each one of the clusters, while computing the weight g_u that the user would have if assigned to that user subset, with Equation (3). If there is a move that will lead to an improvement, then we assign the user to the subset that produced the smallest training error. In order to compute the training error for user u, who is trying to be assigned to a new subset c he/she did not belong to before, we need to *project* him/her to the new subset c, by learning his/her projected user latent factor:

$$\mathbf{p}_u^{cT} = \mathbf{r}_u^T \mathbf{Q}^c \Sigma_{f^c}{}^{-1}. \quad (4)$$

The user-specific weights and user assignments for sGLSVD can be estimated independently in parallel.

4.5 Prediction and Recommendation

The predicted rating of user u, who belongs to subset c, for item i is a combination of the global model and the local model of subset c:

$$\tilde{r}_{ui} = \mathbf{p}_u^T \Sigma_{f^g} \mathbf{q}_i + \mathbf{p}_u^{cT} \Sigma_{f^c} \mathbf{q}_i^c, \quad (5)$$

where $\mathbf{p}_u^T$ is the uth row of $\mathbf{P}$ corresponding to user u, $\mathbf{q}_i$ is the ith column of $\mathbf{Q}^T$ corresponding to item i, $\mathbf{p}_u^{cT}$ is the uth row of $\mathbf{P}^c$ and $\mathbf{q}_i^c$ is the ith column of $\mathbf{Q}^{cT}$. Note that the user-specific weights g_u and $1-g_u$ are enclosed inside the user latent factors $\mathbf{p}_u^T$ and $\mathbf{p}_u^{cT}$ correspondingly.

In order to compute the top-N recommendation list for user u, we estimate the predicted rating $\tilde{r}_{ui}$ with Equation (5) for all his unrated items i, we sort their values in a descending order, and we recommend the N items with the highest corresponding values.

Table 1: Dataset Characteristics.

Name	#Users	#Items	#Transactions	Density
groceries	63,034	15,846	2,060,719	0.21%
ml10m	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%

The 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)).

5 EXPERIMENTAL EVALUATION

5.1 Datasets

We use multiple real-world datasets that span different domains to evaluate the methods in this paper. Table 1 shows their characteristics.

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 *ml10m* dataset corresponds to the MovieLens 10M dataset [13], and contains ratings that users gave on various movies. The *jester* dataset [11] corresponds to an online joke recommender system and contains ratings that users gave on jokes. The *flixster* dataset is a subset of the original Flixster dataset [1], which consists of movie ratings taken from the corresponding social movie site. We created this subset, 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 [3], which contains anonymous movie ratings. We created this subset by keeping the users who have rated between thirty and five hundred items.

Note that some of the datasets originally have ratings, but we converted them to implicit feedback, by transforming the rated entries to ones and the missing entries to zeros. The existence of a rating (1) indicates that the user purchased/rated the item and its absence (0) that he/she did not.

5.2 Evaluation Methodology

We employ leave-one-out cross-validation [27] to evaluate the performance of the developed and competing methods. We randomly select an item rated by each user, and we place it in the test set. The rest of the data comprise the training set.

Algorithm 3 rLSVD

```
Compute the initial assignment of users to subsets.
for all user subset c do
    Construct R^c, as discussed in Section 5.4.
    Compute a truncated SVD of rank f^c on R^c.
end for
```

Algorithm 4 sLSVD

```
Compute the initial assignment of users to subsets.
while number of users switching subsets > 1% of the total users do
    for all user subset c do
        Construct R^c, as discussed in Section 5.4.
        Compute a truncated SVD of the same rank f^c on R^c.
    end for
    for all user u do
        for all user subset c do
            Project user u on user subset c with Equation 4.
            Compute the training error.
        end for
        Assign u to the user subset c with the corresponding smallest training error.
    end for
end while
```

5.3 Performance Metrics

We measure the performance by computing the number of times the single left-out item is 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

$$HR = \frac{\#hits}{\#users}, \tag{6}$$

and ARHR is defined as

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

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. The ARHR is a weighted version of HR, where the position of the test item in the top-N list is taken into account. Both measures have a range from 0 to 1, with 1 being the ideal.

5.4 Proposed Methods

As rGLSVD and sGLSVD estimate multiple components, we propose different variants, to investigate the effect of each component on the top-N recommendation quality:

- **LSVD**, which stands for Local Singular Value Decomposition: We estimate multiple local latent space models of constant rank f^c. The user subsets remain fixed.
- **GLSVD**, which stands for Global and Local Singular Value Decomposition: We estimate a global latent space model along with multiple local latent space models of constant rank f^c. The user subsets are fixed.
- **rLSVD**, which stands for Local Singular Value Decomposition with varying ranks: We estimate multiple latent space models of varying ranks. There is no global model, and the users remain in their original predefined subsets. We compute the predicted rating of user u, who belongs to subset c, for item i as:
$$\tilde{r}_{ui} = \mathbf{p}_u^{cT} \Sigma_{f^c} \mathbf{q}_i^c. \tag{8}$$
After separating the users into k subsets, we construct the corresponding local training matrices $\mathbf{R}^c$ $\forall c \in \{1, \ldots, k\}$ by stacking the vectors $\mathbf{r}_u^T$, for all users u belonging to subset c. We then perform truncated singular value decompositions of varying ranks f^c on each matrix $\mathbf{R}^c$. An overview of rLSVD can be found in Algorithm 3.
- **sLSVD**, which stands for Local Singular Value Decomposition with varying subsets: We estimate multiple latent space models of the same rank; however every user can switch to the subset c, which provides the low-rank representation of u with the smallest training error. There is no global model. We also compute the predicted ratings with Equation (8). An overview of sLSVD can be found in Algorithm 4.

5.5 Comparison Algorithms

We compare our proposed methods against other competing modern top-N recommendation approaches, that span both the item-item-based approaches: SLIM [21], GLSLIM [6], and the latent space approaches: pureSVD [8], BPRMF [24] and LLORMA [19]. The details behind these methods are described in Section 3.

For SLIM, we used the SLIM package.[1] For GLSLIM, we used the GLSLIM package.[2] For PureSVD, we used the SVDLIBC package,[3] which is based on the SVDPACKC library [4]. For BPRMF,we used the LibRec package [12] and for LLORMA we used the PREA toolkit [20].

As LLORMA was developed for rating prediction, but we wanted to use it for top-N recommendation with the evaluation methodology described in Section 5.2, we needed to also utilize the unrated items feedback beyond the rated items. It is of very high complexity to introduce in LLORMA all of the unrated items, making it computationally infeasible. Thus, in this paper, we sampled the unrated items for LLORMA. After experimentation, we concluded that sampling for every user ten times as many unrated items as the items he/she has rated gives overall a good approximation of treating all missing ratings as zeros.

5.6 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 all the methods. We only report the performance corresponding to the parameters that lead to the best results.

For SLIM and GLSLIM, we chose the l_1 and l_2 regularization parameters from the set of values: $\{0.1, 1, 3, 5, 7, 10\}$. The larger the

[1] www-users.cs.umn.edu/~xning/slim/html
[2] https://www-users.cs.umn.edu/~chri2951/code.html
[3] https://tedlab.mit.edu/~dr/SVDLIBC/

regularization parameters are, the stronger the regularization is. For GLSLIM, the number of user subsets examined took on the values: $\{2, 3, 5, 10, 15, \ldots, 90, 95, 100\}$.

For PureSVD, the number of singular values f tried lie in the interval: $\{10, 15, 20, \ldots, 95, 100, 150, 200, \ldots, 1000\}$.

For BPRMF, the number of factors 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: $\{0.0001, 0.001, 0.01, 0.1\}$.

For LLORMA, we followed the parameter methodology of the original paper [19] and we kept fixed the number of iterations $T = 100$, the convergence threshold to $\epsilon = 0.0001$, the number of anchor points to $q = 50$, and used the Epanechnikov kernel with $h1 = h2 = 0.8$. We tried for the regularization values $\lambda_U = \lambda_V$ the values: $\{0.001, 0.01, 0.1\}$. We also tried for the rank of the models the values: $\{1, 2, 3, 5, 7, 10, 15, 20, 25, 30, 35, 40, 45, 50\}$.

For our proposed approaches rGLSVD, sGLSVD and their variants, the number of user subsets examined took on the values: $\{2, 3, 5, 10, 15, \ldots, 95, 100\}$. We varied the rank of the local models f^c among the values: $\{1, 2, 3, 5, 10, 15, \ldots, 90, 95, 100\}$. We did not conduct parameter search on the rank of the global model f^g, instead we fixed it to the value f shown to provide the best results in PureSVD.

The size of the recommendation list N is fixed to the value 10, unless explicitly stated otherwise.

6 EXPERIMENTAL RESULTS

6.1 Performance of rGLSVD and sGLSVD

Table 2 shows the performance of rGLSVD, sGLSVD, and their various simplified variants in terms of HR (Equation (6)) and ARHR (Equation (7)), respectively for every dataset, along with the set of parameters for which this performance was achieved.

We can see that the overall best performing methods are the proposed methods: rGLSVD and sGLSVD. We can also see that the best low-rank representation is achieved in some datasets by varying the rank of local models (rGLSVD), and in others by allowing the user subsets to be updated, while having local models of fixed rank (sGLSVD). We can reach the same conclusion from the pairwise comparison of sLSVD and rLSVD. This shows the merit of both ways to reach the best local low-rank representation.

We can also observe that the global component improves the recommendation quality, by performing a pairwise comparison of LSVD with GLSVD, sLSVD with sGLSVD, and rLSVD with rGLSVD. After performing paired t-tests, the difference in their performance was shown to be statistically significant, with 95% confidence.

Finally, we can see that rLSVD and sLSVD outperform LSVD, both in terms of HR and ARHR, as LSVD is a simpler method than rLSVD and sLSVD: rLSVD with constant rank f^c results in LSVD and sLSVD with fixed user subsets results in LSVD. Also, rGLSVD and sGLSVD outperform GLSVD, which is also expected as GLSVD results from sGLSVD with fixed user subsets, or rGLSVD with constant ranks f^c.

We do not show the ranks of each local model f^c that leads to the best performance of rLSVD and rGLSVD in Table 2 for space reasons. More details can be found in [2].

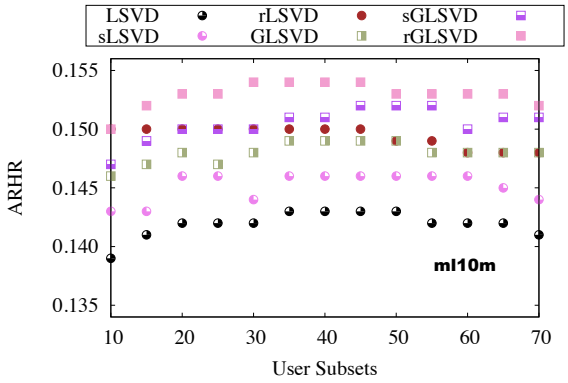

Figure 1: The performance of the proposed methods: LSVD, sLSVD, rLSVD, GLSVD, sGLSVD, and rGLSVD when varying the number of user subsets, in terms of ARHR, for the *ml10m* dataset.

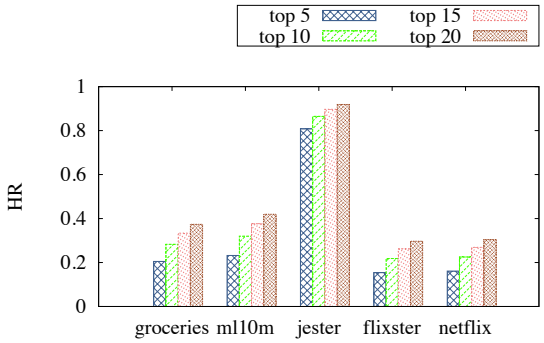

Figure 2: The performance of sGLSVD in terms of HR for different sizes N of the recommendation list.

6.1.1 Sensitivity on the number of user subsets. Figure 1 shows the performance of the proposed methods when varying the number of user subsets, in terms of ARHR for the *ml10m* dataset.

We can see for a wide range of user subsets, and not for just a specific choice, that: (i) rGLSVD and sGLSVD outperform the rest of the approaches, (ii) estimating a global model beyond local models helps the performance, and (iii) allowing update of the user subsets or estimating multiple local models with varying ranks allows for estimation of better low-rank representations than the ones estimated with constant local ranks and fixed user subsets.

The trends are similar for HR and for the rest of the datasets.

6.1.2 Varying the size N of the recommendation list. Figure 2 shows the performance of sGLSVD for different sizes of recommendation list, namely $N = \{5, 10, 15, 20\}$, in terms of HR.

We can see that as the size of the recommendation list increases, the performance of sGLSVD is improved, as a bigger list means that it is more probable for the test item to appear in the recommended list. The same trends hold for ARHR, and for the rest of the proposed approaches.

Table 2: Comparison between our proposed approaches.

Comparison in terms of HR

	LSVD			sLSVD			rLSVD		GLSVD				sGLSVD				rGLSVD		
Dataset	Cls	f^c	HR	Cls	f^c	HR	Cls	HR	Cls	f^g	f^c	HR	Cls	f^g	f^c	HR	Cls	f^g	HR
groceries	100	20	0.192	100	25	0.271	100	0.210	100	25	25	0.204	100	25	25	**0.283**	90	30	0.216
ml10m	25	20	0.300	50	15	0.311	15	0.317	75	65	10	0.311	85	55	10	0.320	30	65	**0.321**
jester	5	3	0.816	2	3	0.816	5	0.895	2	25	2	0.863	2	25	2	0.865	10	15	**0.910**
flixster	5	40	0.202	15	30	0.207	10	0.207	3	80	40	0.214	25	80	30	**0.218**	5	80	0.217
netflix	90	20	0.211	65	20	0.215	90	0.216	65	50	25	0.219	100	50	20	**0.225**	70	50	0.223

Comparison in terms of ARHR

	LSVD			sLSVD			rLSVD		GLSVD				sGLSVD				rGLSVD		
Dataset	Cls	f^c	ARHR	Cls	f^c	ARHR	Cls	ARHR	Cls	f^g	f^c	ARHR	Cls	f^g	f^c	ARHR	Cls	f^g	ARHR
groceries	100	15	0.091	100	15	0.130	100	0.105	100	20	20	0.100	100	20	20	**0.136**	90	30	0.105
ml10m	25	20	0.142	55	15	0.146	20	0.150	35	65	15	0.149	45	55	15	0.152	40	65	**0.154**
jester	5	1	0.693	3	2	0.697	5	0.772	3	15	1	0.746	5	15	1	0.746	3	15	**0.783**
flixster	5	50	0.091	15	30	0.096	10	0.095	3	80	40	0.099	25	80	35	**0.102**	10	80	0.101
netflix	90	20	0.097	95	20	0.100	90	0.099	65	50	20	0.101	100	50	20	**0.105**	85	50	0.104

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 user subsets (Cls), the rank of the global model f^g, and the rank of the local models f^c. Note that the vector of varying local ranks f^c for rGLSVD and rLSVD is not shown, for space reasons. The bold numbers show the best HR/ARHR achieved per dataset.

6.2 Performance against Competing Approaches

6.2.1 Comparison against competing latent space methods. Table 3 compares the performance of our proposed methods rGLSVD and sGLSVD against competitive modern latent space top-N recommendation approaches: LLORMA, PureSVD and BPRMF for all datasets, in terms of HR and ARHR, respectively.

We can see that rGLSVD and sGLSVD outperform the competing latent space top-N approaches, both in terms of HR and in terms of ARHR. We performed paired t-tests of rGLSVD/sGLSVD against the best competing latent space baseline, which was either BPRMF or PureSVD, and the performance difference was shown to be statistically significant. The results of LLORMA being lower in top-N quality than PureSVD surprised us; we believe that the reason is that the original use of LLORMA was for datasets with explicit feedback and for the rating prediction task, thus not necessarily resulting in as good of recommendation quality for performing top-N recommendation on implicit feedback.

From comparing Table 3 with Table 2, we can also see that rLSVD and sLSVD tend to outperform the best competing baseline as well.

6.2.2 Global & local Approaches against standard global approaches. Table 4 compares the use of global and local approaches, against standard global models both in terms of item-based models (SLIM vs GLSLIM) and latent space models (PureSVD vs proposed method sGLSVD). The comparison is shown both in terms of HR and ARHR.

We can see from the pairwise comparison of SLIM with GLSLIM and of PureSVD with sGLSVD that the global and local approaches outperform the standard global models. The paired t-tests we ran showed that the performance different is significant. This showcases their value.

We can also see that GLSLIM performs better than the rest of the approaches. We believe that the reason GLSLIM outperforms sGLSVD is that its underlying model, which is SLIM outperforms PureSVD. Also, even though rGLSVD/sGLSVD does not outperform GLSLIM, we can see that in different cases, its percentage of improvement beyond the underlying global model PureSVD can be higher than the corresponding percentage of improvement of GLSLIM beyond SLIM.

Finally, Table 5 shows the training time needed for GLSLIM versus sGLSVD, for the *ml10m* dataset with 5 user subsets. GLSLIM-warm corresponds to an optimized runtime for GLSLIM, where the estimated model is initialized with a previous model learned, instead of starting from scratch. For SLIM and GLSLIM, the times shown correspond to $\beta_g = \beta_l = 10$ and $\lambda_g = \lambda_l = 1$. For sGLSVD, the times correspond to $f^g = 55$ and $f^c = 10$. Similar timewise comparisons hold for other parameter choices and for the rest of the datasets. The times shown correspond to one node of the supercomputer Mesabi [4], which is equipped with 62 GB RAM and 24 cores. We can see that the time needed to train sGLSVD is only a fraction of the time needed to train GLSLIM. Specifically, sGLSVD is $5-10$ times faster than GLSLIM, which can be of use in cases when faster training is needed, with good top-N recommendation results.

7 CONCLUSION AND FUTURE WORK

In this paper, we propose the following user model: the behavior of a user can be described by a combination of a set of aspects shared by all users, and of a set of aspects which are specific to the subset the user belongs to. This user model is an extension of the model usually employed by the latent space approaches, which assumes

[4] https://www.msi.umn.edu/content/mesabi

Table 3: Comparison with competing latent space approaches.

Comparison in terms of HR

	LLORMA			PureSVD		BPRMF				sGLSVD				rGLSVD		
Dataset	λ	rank	HR	f	HR	factors	lrnrate	reg	HR	Cls	f^g	f^c	HR	Cls	f^g	HR
groceries	0.01	20	0.096	738	0.134	3000	0.01	0.001	0.214	100	25	25	**0.283**	90	30	0.216
ml10m	0.01	35	0.194	64	0.295	5000	0.01	0.01	0.240	85	55	10	0.320	30	65	**0.321**
jester	0.01	30	0.812	25	0.860	300	0.01	0.01	0.903	2	25	2	0.865	10	15	**0.910**
flixster	0.001	35	0.148	90	0.194	4000	0.01	0.001	0.200	25	80	30	**0.218**	5	80	0.217
netflix	0.01	7	0.108	50	0.204	5000	0.01	0.01	0.210	100	50	20	**0.225**	70	50	0.223

Comparison in terms of ARHR

	LLORMA			PureSVD		BPRMF				sGLSVD				rGLSVD		
Dataset	λ	rank	ARHR	f	ARHR	factors	lrnrate	reg	ARHR	Cls	f^g	f^c	ARHR	Cls	f^g	ARHR
groceries	0.01	20	0.046	700	0.059	3100	0.01	0.001	0.099	100	20	20	**0.136**	90	30	0.105
ml10m	0.01	25	0.080	56	0.139	7000	0.01	0.01	0.105	45	55	15	0.152	40	65	**0.154**
jester	0.01	7	0.673	15	0.740	100	0.01	0.01	0.766	5	15	1	0.746	3	15	**0.783**
flixster	0.001	35	0.058	80	0.086	4000	0.01	0.001	0.089	25	80	35	**0.102**	10	80	0.101
netflix	0.01	7	0.043	50	0.091	5000	0.01	0.01	0.100	100	50	20	**0.105**	85	50	0.104

For each method, columns corresponding to the best HR and ARHR and the set of parameters for which they are achieved are shown. For LLORMA, the parameters are the regularization (λ) and the rank of the models (rank). For PureSVD, the parameter is the rank of the model (f). For BPRMF, the parameters are: the number of factors (factors), the learning rate (lrnrate) and the regularization (reg). For sGLSVD and rGLSVD, the parameters are: the number of user subsets (Cls), the rank of the global model (f^g) and the rank of the local models (f^c). Bold numbers indicate the best HR and ARHR across the different algorithms, for every dataset.

Table 4: Comparison of global approaches with global & local approaches.

Comparison in terms of HR.

	SLIM models										PureSVD models						
Dataset	SLIM			GLSLIM						Imp.	PureSVD		sGLSVD				Imp.
	β	λ	HR	Cls	β_g	β_l	λ_g	λ_l	HR	%	f	HR	Cls	f^g	f^c	HR	%
groceries	5	0.1	0.259	100	5	5	1	1	0.304	17.37	750	0.134	100	25	25	0.283	111.19
ml10m	7	5	0.312	10	10	7	1	1	0.345	10.58	65	0.295	85	55	10	0.320	8.47
jester	3	0.1	0.878	10	10	10	10	0.1	0.940	7.06	25	0.860	2	25	2	0.865	0.58
flixster	0.1	2	0.242	3	1	1	1	5	0.255	5.37	90	0.194	25	80	30	0.218	12.37
netflix	0.1	5	0.231	5	1	1	5	5	0.245	6.06	50	0.204	100	50	20	0.225	10.29

Comparison in terms of ARHR

	SLIM models										PureSVD models						
Dataset	SLIM			GLSLIM						Imp.	PureSVD		sGLSVD				Imp.
	β	λ	ARHR	Cls	β_g	β_l	λ_g	λ_l	ARHR	%	f	ARHR	Cls	f^g	f^c	ARHR	%
groceries	3	0.1	0.130	100	5	5	1	1	0.155	19.23	700	0.059	100	20	20	0.136	130.51
ml10m	5	2	0.151	10	10	7	1	1	0.170	12.58	56	0.139	45	55	15	0.152	9.35
jester	7	0.1	0.755	100	1	1	1	1	0.835	10.60	15	0.740	5	15	1	0.746	0.81
flixster	0.1	2	0.116	3	1	1	1	5	0.126	8.62	80	0.086	25	80	35	0.102	18.60
netflix	5	5	0.107	5	1	1	5	5	0.116	8.41	50	0.091	100	50	20	0.105	15.38

The performance of the standard global approaches: SLIM and PureSVD is compared with the performance of the global and local approaches: GLSLIM and sGLSVD. For each method, columns corresponding to the best HR and ARHR and the set of parameters for which they are achieved are shown. The last column of each class of models (Imp.) shows the percentage of improvement of the global and local approach beyond the standard global approach. For SLIM, the parameters are the l_2 and l_1 regularization parameters β and λ. For PureSVD, the parameter is the rank of the model (f). For GLSLIM, the parameters are: the number of user subsets (Cls), the global regularization parameters β_g and λ_g and the local regularization parameters: β_l and λ_l. For sGLSVD, the parameters are: the number of user subsets (Cls), the rank of the global model (f^g) and the rank of the local models (f^c).

Table 5: The training time for *ml10m* dataset with 5 user subsets.

Method	mins
sGLSVD	9.3
GLSLIM	199.2
GLSLIM-warm	53.7

that the behavior of a user can be described by a set of aspects shared by all.

Learning the user model we proposed with a global latent space approach can be difficult, because we often have sparse data. Thus, we propose two methods: rGLSVD and sGLSVD, which explicitly encode this structure, by estimating both a set of global factors and sets of user subset specific latent factors.

The experimental evaluation shows that the proposed approaches estimate better latent representations for the users, outperforming competing latent space top-N recommendation approaches significantly, thus showing the merits of the proposed user model. The performance improvement is on average 13% and up to 37%.

In the future, we plan to combine the two proposed approaches, creating a 'rsGLSVD' approach, using a regularized latent space model as the basis model (such as regularized SVD), so that users would not always switch to the subset of higher dimensions, as this would be penalized. Another potential future direction could be to estimate item-subsets, instead of user subsets, so that the latent factor representation of the items could be also improved. Finally, the proposed approaches can be extended to many levels of local models, instead of one as shown in this paper, thus resulting in a hierarchical model.

8 ACKNOWLEDGEMENTS

This work was supported in part by NSF (IIS-1247632, IIP-1414153, IIS-1447788, IIS-1704074, CNS-1757916), 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.

REFERENCES

[1] [n. d.]. Flixster Dataset. http://http://www.cs.sfu.ca/~sja25/personal/datasets/. ([n. d.]).
[2] 2018. Local Latent Space Models for Top-N Recommendation Appendix - Technical Report. https://www.cs.umn.edu/sites/cs.umn.edu/files/tech_reports/18-010.pdf. (2018).
[3] James Bennett, Stan Lanning, et al. 2007. The netflix prize. In *Proceedings of KDD cup and workshop*, Vol. 2007. New York, NY, USA, 35.
[4] Michael W Berry. 1992. Large-scale sparse singular value computations. *The International Journal of Supercomputing Applications* 6, 1 (1992), 13–49.
[5] Alex Beutel, Ed H Chi, Zhiyuan Cheng, Hubert Pham, and John Anderson. 2017. Beyond Globally Optimal: Focused Learning for Improved Recommendations. In *Proceedings of the 26th International Conference on World Wide Web*. International World Wide Web Conferences Steering Committee, 203–212.
[6] Evangelia Christakopoulou and George Karypis. 2016. Local item-item models for top-n recommendation. In *Proceedings of the 10th ACM Conference on Recommender Systems*. ACM, 67–74.
[7] Konstantina Christakopoulou and Arindam Banerjee. 2015. Collaborative Ranking with a Push at the Top. In *Proceedings of the 24th International Conference on World Wide Web*. International World Wide Web Conferences Steering Committee, 205–215.
[8] Paolo Cremonesi, Yehuda Koren, and Roberto Turrin. 2010. Performance of recommender algorithms on top-n recommendation tasks. In *Proceedings of the fourth ACM conference on Recommender systems*. ACM, 39–46.
[9] Mukund Deshpande and George Karypis. 2004. Item-based top-n recommendation algorithms. *ACM Transactions on Information Systems (TOIS)* 22, 1 (2004), 143–177.
[10] Thomas George and Srujana Merugu. 2005. A scalable collaborative filtering framework based on co-clustering. In *Data Mining, Fifth IEEE international conference on*. IEEE, 4–pp.
[11] Ken Goldberg, Theresa Roeder, Dhruv Gupta, and Chris Perkins. 2001. Eigentaste: A constant time collaborative filtering algorithm. *Information Retrieval* 4, 2 (2001), 133–151.
[12] Guibing Guo, Jie Zhang, Zhu Sun, and Neil Yorke-Smith. 2015. LibRec: A Java Library for Recommender Systems.. In *UMAP Workshops*.
[13] F Maxwell Harper and Joseph A Konstan. 2016. The movielens datasets: History and context. *ACM Transactions on Interactive Intelligent Systems (TiiS)* 5, 4 (2016), 19.
[14] Yifan Hu, Yehuda Koren, and Chris Volinsky. 2008. Collaborative filtering for implicit feedback datasets. In *Data Mining, 2008. ICDM'08. Eighth IEEE International Conference on*. Ieee, 263–272.
[15] George Karypis. 2002. *CLUTO-a clustering toolkit*. Technical Report. MINNESOTA UNIV MINNEAPOLIS DEPT OF COMPUTER SCIENCE.
[16] Yehuda Koren. 2008. Factorization meets the neighborhood: a multifaceted collaborative filtering model. In *Proceedings of the 14th ACM SIGKDD international conference on Knowledge discovery and data mining*. ACM, 426–434.
[17] Yehuda Koren. 2010. Factor in the neighbors: Scalable and accurate collaborative filtering. *ACM Transactions on Knowledge Discovery from Data (TKDD)* 4, 1 (2010), 1.
[18] Joonseok Lee, Samy Bengio, Seungyeon Kim, Guy Lebanon, and Yoram Singer. 2014. Local collaborative ranking. In *Proceedings of the 23rd international conference on World wide web*. ACM, 85–96.
[19] Joonseok Lee, Seungyeon Kim, Guy Lebanon, and Yoram Singer. 2013. Local low-rank matrix approximation. In *International Conference on Machine Learning*. 82–90.
[20] Joonseok Lee, Mingxuan Sun, and Guy Lebanon. 2012. Prea: Personalized recommendation algorithms toolkit. *Journal of Machine Learning Research* 13, Sep (2012), 2699–2703.
[21] Xia Ning and George Karypis. 2011. Slim: Sparse linear methods for top-n recommender systems. In *Data Mining (ICDM), 2011 IEEE 11th International Conference on*. IEEE, 497–506.
[22] Mark OâĂŹConnor and Jon Herlocker. 1999. Clustering items for collaborative filtering. In *Proceedings of the ACM SIGIR workshop on recommender systems*, Vol. 128. UC Berkeley.
[23] Rong Pan, Yunhong Zhou, Bin Cao, Nathan N Liu, Rajan Lukose, Martin Scholz, and Qiang Yang. 2008. One-class collaborative filtering. In *Data Mining, 2008. ICDM'08. Eighth IEEE International Conference on*. IEEE, 502–511.
[24] Steffen Rendle, Christoph Freudenthaler, Zeno Gantner, and Lars Schmidt-Thieme. 2009. BPR: Bayesian personalized ranking from implicit feedback. In *Proceedings of the twenty-fifth conference on uncertainty in artificial intelligence*. AUAI Press, 452–461.
[25] Badrul M Sarwar, George Karypis, Joseph Konstan, and John Riedl. 2002. Recommender systems for large-scale e-commerce: Scalable neighborhood formation using clustering. In *Proceedings of the fifth international conference on computer and information technology*, Vol. 1.
[26] Yue Shi, Alexandros Karatzoglou, Linas Baltrunas, Martha Larson, Nuria Oliver, and Alan Hanjalic. 2012. CLiMF: learning to maximize reciprocal rank with collaborative less-is-more filtering. In *Proceedings of the sixth ACM conference on Recommender systems*. ACM, 139–146.
[27] Pang-Ning Tan et al. 2007. *Introduction to data mining*. Pearson Education India.
[28] Jason Weston, Ron J Weiss, and Hector Yee. 2013. Nonlinear latent factorization by embedding multiple user interests. In *Proceedings of the 7th ACM conference on Recommender systems*. ACM, 65–68.
[29] Bin Xu, Jiajun Bu, Chun Chen, and Deng Cai. 2012. An exploration of improving collaborative recommender systems via user-item subgroups. In *Proceedings of the 21st international conference on World Wide Web*. ACM, 21–30.