NLMF: NonLinear Matrix Factorization Methods for Top-N Recommender Systems

Santosh Kabbur and George Karypis
Department of Computer Science, University of Minnesota
Twin Cities, USA
{skabbur,karypis}@cs.umn.edu

***Abstract*—Many existing state-of-the-art top-N recommendation methods model users and items in the same latent space and the recommendation scores are computed via the dot product between those vectors. These methods assume that the user preference is consistent across all the items that he/she has rated. This assumption is not necessarily true, since many users can have multiple personas/interests and their preferences can vary with each such interest. To address this, a recently proposed method modeled the users with multiple interests. In this paper, we build on this approach and model users using a much richer representation. We propose a method which models the user preference as a combination of having global preference and interest-specific preference. The proposed method uses a nonlinear model for predicting the recommendation score, which is used to perform top-N recommendation task. The recommendation score is computed as a sum of the scores from the components representing global preference and interest-specific preference. A comprehensive set of experiments on multiple datasets show that the proposed model outperforms other state-of-the-art methods for top-N recommendation task.**

***Keywords*-Database Applications, Data mining, Personalization, Mining methods and algorithms**

I. Introduction

Recommender Systems are prevelant and are widely used in many applications. Specifically, top-N recommender systems are widely used in e-commerce applications to recommend ranked list of items to users in order to identify the items that best fit their personal tastes obtained from their feedback. Over the years many algorithms and methods have been developed to address the top-N recommender problem (1; 2). These algorithms make use of the user feedback data available in the form of purchase, rating or review. The existing methods can be broadly classified into two groups: collaborative filtering (CF) based methods and content based methods. User/Item co-rating is used in collaborative filtering methods to build models. Typically these methods represent the user ratings on items in a user-item rating matrix and act on it. One class of state-of-the-art methods in top-N recommender problem is based on learning latent factors for users and items. In these methods, users and items are represented as vectors in common latent space and the recommendation score for a given user and item pair is computed as the dot product of the corresponding user and item latent vectors. Most notable methods rely on matrix factorization (MF) (2) or singular value decomposition (SVD) (3) to learn the user and item latent factors. Some extensions and variations to SVD are also proposed (like SVD++ (4)). In content based methods, users/items features are used to build models (5; 6). In this work, we limit our focus to only CF based methods.

One of the recently developed methods called MaxMF (7), extends the traditional Matrix Factorization (MF) based approaches by representing the user with multiple latent vectors, each corresponding to a different "taste" associated with the user. These different tastes associated with each user representation are termed as *interests*. The assumption behind this approach is that, by letting the users to have multiple interests, it helps to capture user preferences better, especially when the itemsets or user's interests is diverse. The authors then propose a $\max$ function based non linear model, which takes the maximum scoring interest as the final recommendation score for a given user item pair. It was shown that MaxMF achieves better recommendation performance than other state-of-the-art methods. However, one of the limitations of this method is, it models the users with only interest-specific component. This can potentially dilute the learnt latent factors for users who have not provided enough preferences or who do not have enough diversity in their itemsets due to lack of support (in terms of number of rated items) for each of the interests.

In this paper, we propose a new method called NLMF (Non Linear Matrix Factorization), which models the user as a combination of global preference and interest-specific latent factors. This representation of user allows NLMF to effectively capture both the global preference and multiple interest-specific preference. This approach implicitly allows the model to strike a balance between the global and interest-specific components. Our experimental evaluation on multiple datasets show that NLMF performs better than MaxMF and other state-of-the-art methods.

The key contributions of the work presented in this paper are the following:

(i) proposes a new nonlinear method, which models the users multiple interests as a combination of global and interest-specific preferences;
(ii) proposes two different approaches based on shared and independent item factors between the global preference and interest-specific preferences; and
(iii) compares the performance of the proposed method with other state-of-the-art methods for top-N recommendation task, and investigates the impact of various parameters as they relate to number of latent factors and number of

interests.

The rest of the paper is organized as follows. Section II introduces the notations used in this paper. In Section III, we present the relavant existing methods. Section IV motivates the need for a better model and constrasts the proposed method with the existing ones. In Section V, we present the details of the NLMF methods. Section VI presents the evaluation methodology, the data sets used along with their characteristics and the details of the baseline algorithms that we will be comparing the proposed approach with. In Section VII, we present the experimental evaluation with the discussion. Finally, Section VIII provides the concluding remarks.

II. Notations

In this paper, all vectors are represented by bold lower case letters and they are row vectors (e.g., $\mathbf{p}, \mathbf{q}$). All matrices are represented by bold upper case letters (e.g., $\mathbf{R}$, $\mathbf{W}$). The ith row of a matrix $\mathbf{A}$ is represented by $\mathbf{a_i}$. We use calligraphic letters to denote sets (e.g., $\mathcal{C}$, $\mathcal{D}$). A predicted and an estimated value are denoted by having a ˆ (hat) over it (e.g., $\hat{r}$).

$\mathcal{C}$ and $\mathcal{D}$ are used to denote the sets of users and items, respectively, whose respective cardinalities are n and m (i.e., $|\mathcal{C}| = n$ and $|\mathcal{D}| = m$). Matrix $\mathbf{R}$ will be used to represent the user-item implicit feedback (purchase/review) matrix of size $n \times m$, i.e., $\mathbf{R} \in \mathbb{R}^{n \times m}$. Symbols u and i are used to denote individual users and items, respectively. An entry (u, i) in $\mathbf{R}$, denoted by r_{ui}, is used to represent the rating on item i by user u. $\mathbf{R}$ is a binary matrix. That is, if the user has provided feedback for a particular item, then the corresponding entry in $\mathbf{R}$ is 1, otherwise it is 0. We will refer to the entries for which the user has provided feedback as *rated* items and those for which the user has not provided feedback as *unrated* items. For quick reference, all the important symbols used in this paper, along with their definition is summarized in Table I.

III. Review of Relevant Research

UserKNN (8; 9) is a classical user based CF method, which computes k-nearest neighbors for each user, based on their rating profiles. These nearest neighbors are then used to predict the rating for a user on an unrated item as the weighted average of the rating of the nearest neighbors of the user. This method is nonlinear in terms of the preferences of the user, which are implicitly captured via the nearest neighbors. However, this method relies on the co-rating information between the users to compute the similarity. Thus, it suffers from data sparsity issue and fails to capture relations between users who do not have enough co-rated items.

In the recent user-item factorization methods based on MF (3), the rating matrix $\mathbf{R}$ is approximated as a product of two low-rank matrices $\mathbf{P}$ and $\mathbf{Q}$, where $\mathbf{P} \in \mathbb{R}^{n \times k}$ is the users latent vector matrix, $\mathbf{Q} \in \mathbb{R}^{m \times k}$ is the items latent vector matrix, k is the number of latent factors and $k \ll n, m$. The recommendation score of a user u for item i is predicted as,

$$\hat{r}_{ui} = \mathbf{p}_u \mathbf{q}_i^{\mathsf{T}}, \tag{1}$$

where $\mathbf{p}_u$ is the latent vector associated with the user u and $\mathbf{q}_i$ is the latent vector associated with the item i.

TABLE I: Symbols used and definitions.

Symbol	Definition
$\mathcal{C}$	Set of users.
$\mathcal{D}$	Set of items.
u	Individual user u.
i	Individual item i.
n	Number of users.
m	Number of items.
k	Number of latent factors.
l	Number of latent factors for interest-specific component in NLMFi.
T	Number of user interests.
$\mathbf{R}$	Binary Rating Matrix, $\mathbf{R} \in \mathbb{R}^{n \times m}$.
r_{ui}	Rating by user u on item i.
$\hat{r}_{ui}$	Predicted rating for user u on item i.
$\mathbf{P}$	User Latent Factor Matrix, $\mathbf{P} \in \mathbb{R}^{n \times k}$.
$\mathbf{Q}$	Item Latent Factor Matrix, $\mathbf{Q} \in \mathbb{R}^{m \times k}$.
$\mathbf{W}$	User Latent Factor Tensor, $\mathbf{W} \in \mathbb{R}^{n \times k \times T}$.
$\mathbf{Y}$	Item Latent Factor Matrix, for interest-specific component in NLMFi, $\mathbf{Y} \in \mathbb{R}^{m \times k}$.
λ	ℓ_F regularization weight.
ρ	Sampling factor for learning algorithm.
η	Learning Rate for learning algorithm.

A recent method for top-N recommendation task proposed by Weston et. al. called MaxMF (7) defines T interest latent vectors per user. The user factors are thus represented by a tensor $\mathbf{P}$, where $\mathbf{P} \in \mathbb{R}^{n \times k \times T}$. The items factors, $\mathbf{Q}$ remains similar to MF based approaches. Thus, each user u is represented by $\mathbf{p}_u$, where $\mathbf{p}_u \in k \times T$. For a given user u and item i pair, the predicted recommendation score is calculated by computing T dot products between each of the T user vectors and the corresponding item vector. The highest scoring dot product is taken as the estimated/predicted rating. That is,

$$\hat{r}_{ui} = \max_{t=1,\ldots,T} \mathbf{p}_{ut} \mathbf{q}_i^{\mathsf{T}}, \tag{2}$$

where the $\max$ function computes the maximum of the set of dot products between each of $\mathbf{p}_{ut}$ and $\mathbf{q}_i$.

The main intuition behind this approach is that, the user is represented with T different interests and the interest which matches the best with the given item is captured using the $\max$ function. In other words, the set of items is partitioned into T partitions for each user, and this partitioning process is personalized on the user. For each such item partition, a different scoring function is used to estimate the rating.

IV. Motivation

Users typically provide ratings to only a handful of items out of possible thousands or millions of items. Due to limited preferences given out by users, the user-item rating matrix becomes sparse. Methods like MaxMF learns only interest-specific user preferences, by implicitly partitioning the items rated by the user into multiple subsets and learning a separate user latent preference vector for each partition. In case of users who have not provided sufficient ratings, learning only interest-specific preferences will result in lesser support (in terms of number of items) for each interest. This can potentially affect the learning process and can result in learning less meaningful (latent) factors for all the item partitions corresponding to that user.

To overcome this problem, our proposed approach NLMF learns the user preferences as a combination of global preference and interest-specific preference components. The global preference is learned using all the ratings provided by the user. Thus, it helps to better estimate the user's preferences when the available data is limited. With regularization, this method allows the model to be flexible, i.e., it implicitly allows the learning process to strike a balance between the global preference and interest-specific preference components. Hence this model is expected to perform better than MaxMF.

V. NLMF - Nonlinear Methods for CF

In NLMF, given a user u and an item i, the estimated rating $\hat{r}_{ui}$ is given by the sum of the estimations from global preference and interest-specific preference components. That is,

$$\hat{r}_{ui} = \mathbf{p}_u\mathbf{q}_i^\mathsf{T} + \max_{t=1,\ldots,T} f(u,i,t), \tag{3}$$

where $\mathbf{p}_u$ is the latent vector associated with user u and $\mathbf{q}_i$ is the latent vector associated with item i. Thus, $\mathbf{p}_u\mathbf{q}_i^\mathsf{T}$ gives the prediction score from global preference component of the model and $f(u,i,t)$ is the prediction score from interest-specific preference component. The final prediction score is the sum of the predictions from global preference and interest-specific preference components. Figure 1 illustrates the overview of the NLMF method.

The selection of the best interest t^* is done by choosing the interest which results in the maximum score from the multiple interests model. The max function is used to compute the maximum recommendation score for the item amongst all the interests of the user in the multiple interest function. The intuition behind this idea is that, for an item to be ranked higher in the top-N list of the user, at least one of the interests of the user must provide a high score for that item.

We use squared error loss function to compute and minimize the loss. That is,

$$\mathcal{L}(\cdot) = \sum_{i\in\mathcal{D}}\sum_{u\in\mathcal{C}}(r_{ui} - \hat{r}_{ui})^2, \tag{4}$$

where r_{ui} is the ground truth value and $\hat{r}_{ui}$ is the estimated value.

We propose two different methods to represent the interest-specific preference component, $f(u,i,t)$. First one has independent item factors in $f(u,i,t)$ compared to that of global preference component, whereas the second one shares the item factors of $f(u,i,t)$ with the global preference component. These two methods are described in the next two sections.

A. NLMFi - Independent Item Factors

In NLMFi, the interest-specific preference component $f(u,i,t)$ is given by,

$$f(u,i,t) = \mathbf{w}_{ut}\mathbf{y}_i^\mathsf{T}, \tag{5}$$

where $\mathbf{w}_{ut}$ is the user latent vector for u in the interest-specific preference component corresponding to the interest t and $\mathbf{y}_i$ is the item latent vector in the interest-specific preference component. We can see that, for a given item i, NLMFi has two independent item factors ($\mathbf{q}_i$ and $\mathbf{y}_i$), each one corresponding to the global preference and interest-specific preference components.

The recommendation score $\hat{r}_{ui}$ for a given user u and item i is computed as,

$$\hat{r}_{ui} = \mathbf{p}_u\mathbf{q}_i^\mathsf{T} + \max_{t=1,\ldots,T} \mathbf{w}_{ut}\mathbf{y}_i^\mathsf{T} \tag{6}$$

where $\mathbf{p}_u$ and $\mathbf{q}_i$ are the user and item latent vectors in the global preference component respectively. Thus, NLMFi is an additive model which independently learns two non-overlapping models corresponding to global preference and interest-specific component and computes their sum as the final prediction score.

Note that, the number of latent factors for the global preference component (i.e., $\mathbf{p}_u\mathbf{q}_i^\mathsf{T}$) and the interest-specific component (i.e., $\mathbf{w}_{ut}\mathbf{y}_i^\mathsf{T}$) need not be the same. Thus, this model has the flexibility of having different number of latent factors for the two components. We use k to represent the number of latent factors for the global preference component (i.e., $\mathbf{p}_u, \mathbf{q}_i \in \mathbb{R}^{1\times k}$) and we use l to represent the number of latent factors for the interest-specific component (i.e., $\mathbf{w}_{ut}, \mathbf{y}_i \in \mathbb{R}^{1\times l}$).

In NLMFi, the martices $\mathbf{P}$, $\mathbf{Q}$, $\mathbf{W}$ and $\mathbf{Y}$ are learned by minimizing the following regularized optimization problem:

$$\underset{\mathbf{P},\mathbf{Q},\mathbf{W},\mathbf{Y}}{\text{minimize}}\ \frac{1}{2}\sum_{u,i\in R}\|r_{ui} - \hat{r}_{ui}\|_F^2 + \frac{\lambda}{2}(\|\mathbf{P}\|_F^2 + \|\mathbf{Q}\|_F^2 + \|\mathbf{W}\|_F^2 + \|\mathbf{Y}\|_F^2), \tag{7}$$

where λ is the l_2-regularization constant for latent factor matrices. l_2 regularization is used to prevent overfitting.

The optimization problem in Equation 7 is solved using a Stochastic Gradient Descent (SGD) algorithm (10). Algorithm 1 provides the detailed procedure and the gradient update rules for the learning algorithm. Initially the matrices $\mathbf{P}$, $\mathbf{Q}$ and $\mathbf{Y}$, and tensor $\mathbf{W}$ are initialized with small random values as the initial estimate. Then, in each iteration the parameter values are updated based on the gradients computed w.r.t. the parameter being updated. This process is repeated until the error on validation set does not decrease further or the number of iterations has reached a predefined threshold.

Note that the gradient updates for model paramters are computed for both rated and non-rated entries of $\mathbf{R}$. This is in accordance with common practice followed for top-N recommendation problem (3; 11; 12). This is in contrast with rating prediction problem, where only the rated items are typically used for computing gradient updates. In order to reduce the computational complexity of the learning process, the zero entries corresponding to non-rated items are sampled and used along with all the non-zero entries (corresponding to rated items) of $\mathbf{R}$. Given a sampling constant ρ and $nnz(\mathbf{R})$, the number of non-zeros in $\mathbf{R}$, $\rho \cdot nnz(\mathbf{R})$ zeros are sampled and used for optimization in each iteration of the learning

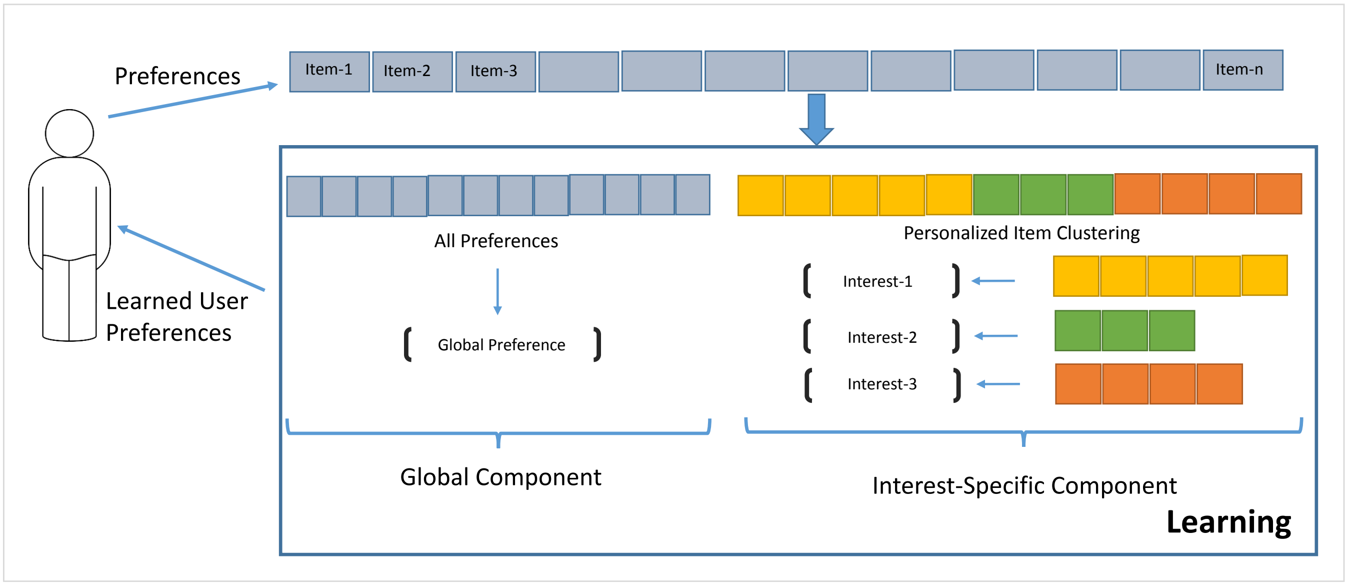

Fig. 1: NLMF Method Overview.

algorithm. Our experimental results indicate that a small value of ρ (in the range $3-5$ is sufficient to produce the best model. This sampling strategy makes NLMF methods computationally efficient and scalable.

Algorithm 1 NLMFi:Learn.

```
1:  procedure NLMFI_LEARN
2:      $\eta \leftarrow$ learning rate
3:      $\lambda \leftarrow \ell_F$ regularization weight
4:      $\rho \leftarrow$ sample factor
5:      $iter \leftarrow 0$
6:      Init $\mathbf{P}$, $\mathbf{Q}$, $\mathbf{W}$ and $\mathbf{Y}$ with random values in (-0.001, 0.001)
7:
8:      while $iter < maxIter$ or error on validation set decreases do
9:          $\mathcal{R}' \leftarrow \mathbf{R} \cup$ SampleZeros($\mathbf{R}$, $\rho$)
10:         $\mathcal{R}' \leftarrow$ RandomShuffle($\mathcal{R}'$)
11:
12:         for all $r_{ui} \in \mathcal{R}'$ do
13:             $\hat{r}_{ui} \leftarrow \mathbf{p}_u \mathbf{q}_i^{\mathsf{T}} + \max_{t=1,\ldots,T} \mathbf{w}_{ut} \mathbf{y}_i^{\mathsf{T}}$
14:
15:             $t^* \leftarrow$ interest corresponding to max score
16:             $e_{ui} \leftarrow r_{ui} - \hat{r}_{ui}$
17:             $\mathbf{p}_u \leftarrow \mathbf{p}_u + \eta \cdot (e_{ui} \cdot \mathbf{q}_i - \lambda \cdot \mathbf{p}_u)$
18:             $\mathbf{q}_i \leftarrow \mathbf{q}_i + \eta \cdot (e_{ui} \cdot \mathbf{p}_u - \lambda \cdot \mathbf{q}_i)$
19:             $\mathbf{w}_{ut^*} \leftarrow \mathbf{w}_{ut^*} + \eta \cdot (e_{ui} \cdot \mathbf{y}_i - \lambda \cdot \mathbf{w}_{ut^*})$
20:             $\mathbf{y}_i \leftarrow \mathbf{y}_i + \eta \cdot (e_{ui} \cdot \mathbf{w}_{ut} - \lambda \cdot \mathbf{y}_i)$
21:         end for
22:
23:         $iter \leftarrow iter + 1$
24:     end while
25:
26:     return $\mathbf{P}, \mathbf{Q}, \mathbf{W}, \mathbf{Y}$
27: end procedure
```

B. NLMFs - Shared Item Factors

In NLMFs, the interest-specific component $f(u,i,t)$ is given by,

$$f(u,i,t) = \mathbf{w}_{ut}\mathbf{q}_i^{\mathsf{T}}, \tag{8}$$

where $\mathbf{w}_{ut}$ is the user latent vector for u in the interest-specific component corresponding to the interest t and $\mathbf{q}_i$ is the shared item latent vector between the global preference and interest-specific components. By using the shared item latent vectors, this model has the ability to transfer the learning between the global preference and interest-specific components. Contrast this with NLMFi model, which has independent item factors ($\mathbf{q}_i$ and $\mathbf{y}_i$) for global preference and interest-specific components.

The recommendation score $\hat{r}_{ui}$ for a given user u and item i is computed as,

$$\tilde{r}_{ui} = \mathbf{p}_u\mathbf{q}_i^{\mathsf{T}} + \max_{t=1,\ldots,T} \mathbf{w}_{ut}\mathbf{q}_i^{\mathsf{T}} \tag{9}$$

where $\mathbf{p}_u$ is the user latent vector for u in the global preference component.

In NLMFs, the martices $\mathbf{P}$, $\mathbf{Q}$ and $\mathbf{W}$ are learned by minimizing the following regularized optimization problem:

$$\underset{\mathbf{P},\mathbf{Q},\mathbf{W}}{\text{minimize}} \ \frac{1}{2}\sum_{u,i\in R} \|r_{ui} - \hat{r}_{ui}\|_F^2 + \frac{\lambda}{2}(\|\mathbf{P}\|_F^2 + \|\mathbf{Q}\|_F^2 + \|\mathbf{W}\|_F^2), \tag{10}$$

where the common terms mean the same as in Equation 7.

Similar to NLMFi, the optimization problem in Equation 10 is solved using a SGD based algorithm. The details of the procedure is presented in Algorithm 2. The learning algorithm and details are similar to Algorithm 1, except the gradient update rules.

Algorithm 2 NLMFs:Learn.

```
1: procedure NLMFs _LEARN
2:     $\eta \leftarrow$ learning rate
3:     $\lambda \leftarrow \ell_F$ regularization weight
4:     $\rho \leftarrow$ sample factor
5:     $iter \leftarrow 0$
6:     Init $\mathbf{P}$, $\mathbf{Q}$ and $\mathbf{W}$ with random values in (-0.001, 0.001)
7:
8:     while $iter < maxIter$ or error on validation set decreases do
9:         $\mathcal{R}' \leftarrow \mathbf{R} \cup$ SampleZeros($\mathbf{R}$, $\rho$)
10:        $\mathcal{R}' \leftarrow$ RandomShuffle($\mathcal{R}'$)
11:
12:        for all $r_{ui} \in \mathcal{R}'$ do
13:            $\hat{r}_{ui} \leftarrow \mathbf{p}_u \mathbf{q}_i^\mathsf{T} + \max_{t=1,\ldots,T} \mathbf{w}_{ut} \mathbf{q}_i^\mathsf{T}$
14:
15:            $t^* \leftarrow$ interest corresponding to max score
16:            $e_{ui} \leftarrow r_{ui} - \hat{r}_{ui}$
17:            $\mathbf{p}_u \leftarrow \mathbf{p}_u + \eta \cdot (e_{ui} \cdot \mathbf{q}_i - \lambda \cdot \mathbf{p}_u)$
18:            $\mathbf{q}_i \leftarrow \mathbf{q}_i + \eta \cdot (e_{ui} \cdot (\mathbf{p}_u + \mathbf{w}_{ut^*}) - \lambda \cdot \mathbf{q}_i)$
19:            $\mathbf{w}_{ut^*} \leftarrow \mathbf{w}_{ut^*} + \eta \cdot (e_{ui} \cdot \mathbf{q}_i - \lambda \cdot \mathbf{w}_{ut^*})$
20:        end for
21:
22:        $iter \leftarrow iter + 1$
23:    end while
24:
25:    return $\mathbf{P}, \mathbf{Q}, \mathbf{W}$
26: end procedure
```

C. Scalability

The optimization algorithm used in the training phase of NLMFs and NLMFi is based on SGD algorithm. The gradient computations and updates for SGD can be parallelized. Hence, these algorithms can be efficiently applied to larger datasets. In (13), a distributed SGD is proposed. A similar algorithm with modifications can be used to scale the NLMF methods to larger datasets. Software packages like Spark[1] can be used to execute SGD based algorithms on a large cluster of processing nodes.

VI. Experimental Evaluation

A. Data Sets

We evaluated the performance of NLMF methods on two different real datasets, namely Netflix and Flixster. Netflix is a subset of data extracted from Netflix Prize Dataset[2] and Flixster is a subset of data extracted from publicly available data set collected from Flixster[3]. For both the datasets, we removed the top 5% of the frequently rated items. All the ratings were binarized, i.e., converted to 1. The characterisitcs of all the datasets is summarized in Table II.

[1]http://spark.apache.org/

[2]http://www.netflixprize.com/

[3]http://www.flixster.com/

TABLE II: Datasets.

Dataset	#Users	#Items	#Ratings	Rsize	Csize	Density
Netflix	5,403	2,933	2,197,096	406.64	749.09	13.86%
Flixster	4,627	3,295	1,184,817	256.06	359.58	7.77%

The "#Users", "#Items" and "#Ratings" columns are the number of users, items and ratings respectively, in each of the datasets. The "Rsize" and "Csize" columns are the average number of ratings for each user and for each item (i.e., row and column density of the user-item matrix), respectively, in each of the datasets. The "Density" column is the density of each dataset (i.e., density = #Ratings/(#Users $\times$ #Items)).

B. Evaluation Methodology

To evaluate the performance of the proposed model, we employ a 5-fold Leave-One-Out-Cross-Validation (LOOCV) method similar to the one employed in (11; 12). Training and test set is created by randomly selecting one item per user from the dataset and placing it in the test set. The rest of the data is used as the training set. This process is repeated to create five different folds. Training set is used to build the model and the trained model is used to generate a ranked list of size-N items for each user. The model is then evaluated by comparing the ranked list of recommended items with the item in the test set. N is equal to 10, for all the results presented in this paper.

The recommendation quality is measured using Hit Rate (HR) and Average Reciprocal Hit Rank (ARHR) (14). HR is defined as,

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

where $\#hits$ is the number of users for which the model was successfully able to recall the test item in the size-N recommendation list and $\#users$ is the total number of test users. The ARHR is defined as,

$$ARHR = \frac{1}{\#users} \sum_{i=1}^{\#hits} \frac{1}{pos_i},$$

where pos_i is the position of the test item in the ranked recommendation list for the i^{th} hit. ARHR represents the weighted version of HR, as it measures the inverse of the position of the recommended item in the ranked list.

We chose HR and ARHR as evaluation metrics since they directly measure the performance of the model on the ground truth data i.e., what users have already provided feedback for.

C. Comparison Algorithms

We compare the performance of NLMF against that achieved by UserKNN (14), PureSVD (3), BPRMF (15), SLIM (11) and MaxMF (7). This set of methods constitute the current state-of-the-art for top-N recommendation task. Hence they form a good set of methods to compare and evaluate our proposed approach against.

VII. Results

The experimental evaluation consists of three parts. First, we assess the effect of various model parameters of NLMF on the recommendation performance. These include how the number of latent factors and the number of interests affect the top-N performance. Second, we present the top-N performance comparison with the MaxMF method, which is also a non-linear method based on modeling user with multiple interests. Due to lack of space, we present these studies only for the Netflix dataset. However the same trend in results and conclusions carry over to the Flixster dataset as well. In the third part of the results, we present the comparison with other competing state-of-the-art methods (Section VI-C).

A. *Effect of Number of Latent Factors*

Figure 2 shows the effect of varying the number of latent factors (k) on the performance of the NLMFs model. For this experiment, the number of interests was set to 2 (i.e., $T = 2$). We can see that the hit-rate gradually increases with the increasing number of latent factors and reaches the peak when $k = 192$ and then it starts to decline. The possible reason for this is that, the model starts to overfit the training data due to large number of latent factors.

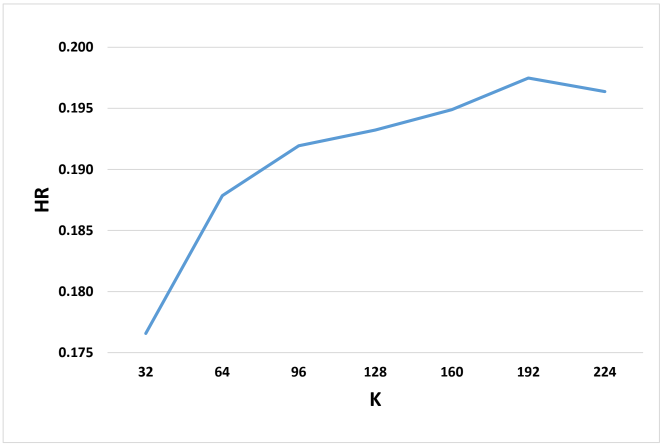

Fig. 2: NLMFs - Effect of Number of Latent Factors.

For NLMFi, Figure 3 shows the effect of varying number of latent factors. Since NLMFi model can have different number of latent factors for global preference (k) and interest-specific preference (l) components, the figure shows multiple line graphs, each corresponding to a different value of k. For a given value of k, l is varied and the performance is presented in this figure. Similar to NLMFs, the performance for NLMFi initially increases with l and then either plateaus out or starts declining due to overfitting. For this experiment, the number of interests was set to 2 (i.e., $T = 2$)

B. *Effect of Number of Interests*

In this study, we compare the effect of number of interests (T) on the recommendation performance. Figure 4 shows the results for both NLMFs and NLMFi models for two different values of k. For NLMFi model, we have set l to be same as k. We can see that in both models, the performance initially increases with increasing T and reaches a peak value when $T = 3$ or $T = 4$. This indicates that, modeling the users with three or four distinct interests provides the best recommendation performance. Further increasing the value of T, the performance starts to decrease. This is possibly due to the fact that, the support for each interest in terms of number of items decreases, thus leading to learning less meaningful user preferences for different interests.

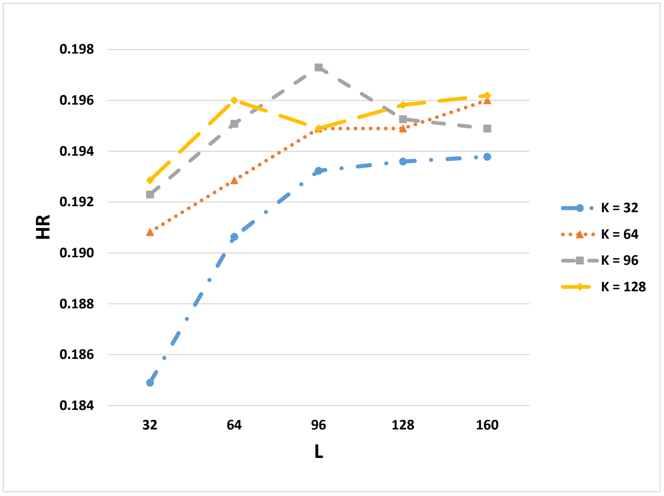

Fig. 3: NLMFi - Effect of Number of Latent Factors.

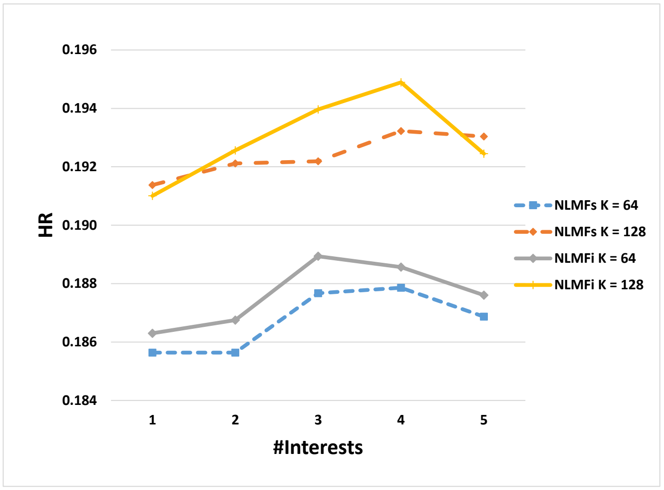

Fig. 4: Effect of Number of Interests.

C. *Comparision with MaxMF*

As discussed in Section III, MaxMF is an existing method which also employs a non-linear method based on max function to learn multiple interest preferences for users. In this study, we compare the performance of NLMF methods with MaxMF for different number of latent factors (k) and interests (T). The results of this study are presented in Figure 5. We can see that both NLMF methods outperform MaxMF for different values of k, with NLMFi performing better than NLMFs. It is interesting to note that, for some values of k, one interest model of MaxMF performs better than the two interests one. Whereas, for NLMF methods, two interests model performs better than one interest model for all values of k. This is possibly due to the reason that, MaxMF learns only the interest-specific user preference, which can potentially lead to decrease in the support for each interest in terms of number of items. On the other hand, NLMF methods balance the interest-specific preference with the global preference by learning a combined model with both the global preference and interest-specific components.

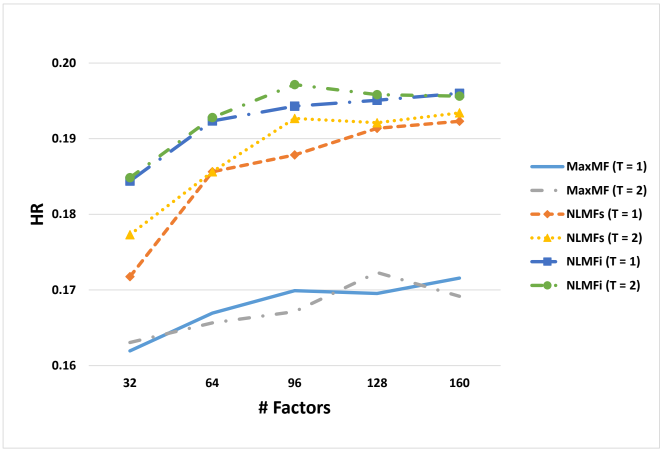

Fig. 5: Comparison with MaxMF.

D. *Comparision with Other Approaches*

Table III shows the overall recommendation performance of NLMF methods in terms of HR and ARHR in comparison to other state-of-the-art methods (Section VI-C). For all the results presented, the number of top-N items chosen is 10 (i.e., $N = 10$). Following parameter space was explored for each of the methods and the best performing model in that parameter space in terms of HR is reported. For UserKNN, PureSVD and BPRMF, parameter k was selected from the range 2 to 800. Learning rate for BPRMF was selected from the range 10^{-5} to 1.0, with a multiplicative increment of 10. For SLIM, the regularization constants were selected from the range 10^{-5} to 20. For MaxMF and NLMF methods the regularization constants were selected from the range 10^{-5} to 5 and learning rate was selected from the range 10^{-5} to 1.0.

The results in Table III show that NLMF methods perform better than the rest of the competing methods for all the datasets. The performance gains of NLMF methods compared to the next best performing baseline method are of the order of 6% and 10% for Netflix and Flixster respectively. Note that, contrary to the results presented in (7), the MaxMF model does not outperform PureSVD for the datasets considered in this study. In terms of the two proposed NLMF methods, independent item factors model (NLMFi) achieved better performance than shared item factors model (NLMFs). The reason for this could be that, NLMFi has the ability to learn the global preference and interest-specific components independently, as the items factors are not overlapping, thereby resulting in learning better respresentation of users and items. This allows the model to strike a better balance between the two components compared to NLMFs, which shares the item factors during the learning process.

VIII. Conclusion

In this paper we presented a non-linear matrix factorization based method (NLMF) for the top-N recommendation task. NLMF models the users preference using a richer representation using a nonlinear model for predicting the recommendation score to perform top-N recommendation task. The recommendation score is computed as a sum of the scores from the components representing the global preference and interest-specific user preference. For modeling the interest-specific component, we presented two different approaches. First approach learns the item factors independently in the global preference and interest-specific components, whereas the second approach shares the item factors between the global preference and interest-specific components. The results showed that the proposed method outperforms rest of the state-of-the-art methods in terms of top-N recommendation performance. As future work, we plan to evaluate this method on multiple datasets at different sparsity levels to measure how NLMF methods perform relative to other methods when the training data gets sparser. We also plan to extend this work for rating prediction task.

Acknowledgements

This work was supported in part by NSF (IIS-0905220, OCI-1048018, CNS-1162405, 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.

References

[1] 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*, vol. 17, no. 6, pp. 734–749, 2005.

[2] F. Ricci, L. Rokach, B. Shapira, and P. Kantor, “Recommender systems handbook,” *Recommender Systems Handbook:, ISBN 978-0-387-85819-7. Springer Science+ Business Media, LLC, 2011*, vol. 1, 2011.

[3] 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*, 2010, pp. 39–46.

[4] Y. Koren, “Factorization meets the neighborhood: a multifaceted collaborative filtering model,” in *Proceeding of the 14th ACM SIGKDD international conference on Knowledge discovery and data mining*. ACM, 2008, pp. 426–434.

[5] R. J. Mooney and L. Roy, “Content-based book recommending using learning for text categorization,” in *Proceedings of the fifth ACM conference on Digital libraries*. ACM, 2000, pp. 195–204.

TABLE III: Comparison of performance of top-N recommendation algorithms with NLMF

Method	Netflix						Flixster					
	Params				HR	ARHR	Params				HR	ARHR
UserKNN	100	-	-	-	0.1412	0.0515	100	-	-	-	0.1013	0.0295
PureSVD	50	-	-	-	0.1821	0.0807	100	-	-	-	0.1273	0.0494
BPRMF	400	0.01	-	-	0.1890	0.0813	200	0.01	-	-	0.1165	0.0437
SLIM	0.001	0.1	-	-	0.1888	0.0872	0.01	1.0	-	-	0.1303	0.0502
MaxMF	192	2	0.0005	0.0005	0.1743	0.0704	160	2	0.0001	0.0005	0.1345	0.0493
NLMFs	192	2	0.01	0.0005	0.1975	0.0870	256	2	0.01	0.005	0.1401	0.0532
NLMFi	256/160	2	0.008	0.001	<u>0.1999</u>	0.0835	288/192	2	0.01	0.001	<u>0.1441</u>	0.0546

Columns corresponding to "params" indicate the model parameters for the corresponding method. For UserKNN method, the parameter is the number of neighbors. For PureSVD method, the parameter is the number of latent factors. For BPRkNN method, the parameters are the number of latent factors used and the learning rate. For SLIM method, the parameters correspond to the ℓ_2 and ℓ_1 regularization constants. For MaxMF and NLMFs methods, the parameters correspond to the number of latent factors, number of interests, regularization constant and learning rate. For NLMFi method, the parameters correspond to number of latent factors for global preference and interest-specific preference components, number of interests, regularization constant and learning rate. The columns corresponding to HR and ARHR represent the hit rate and average reciprocal hit rank metrics. <u>Underlined</u> numbers represent the best performing model measured in terms of HR for each dataset.

[6] M. Pazzani and D. Billsus, "Content-based recommendation systems," *The adaptive web*, pp. 325–341, 2007.

[7] J. Weston, R. J. Weiss, and H. Yee, "Nonlinear latent factorization by embedding multiple user interests," in *Proceedings of the 7th ACM conference on Recommender systems*. ACM, 2013, pp. 65–68.

[8] J. A. Konstan, B. N. Miller, D. Maltz, J. L. Herlocker, L. R. Gordon, and J. Riedl, "Grouplens: applying collaborative filtering to usenet news," *Communications of the ACM*, vol. 40, no. 3, pp. 77–87, 1997.

[9] U. Shardanand and P. Maes, "Social information filtering: algorithms for automating word of mouth," in *Proceedings of the SIGCHI conference on Human factors in computing systems*. ACM Press/Addison-Wesley Publishing Co., 1995, pp. 210–217.

[10] L. Bottou, "Online algorithms and stochastic approximations," in *Online Learning and Neural Networks*, D. Saad, Ed. Cambridge, UK: Cambridge University Press, 1998, revised, oct 2012. [Online]. Available: http://leon.bottou.org/papers/bottou-98x

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

[12] S. Kabbur, X. Ning, and G. Karypis, "Fism: factored item similarity models for top-n recommender systems," in *Proceedings of the 19th ACM SIGKDD international conference on Knowledge discovery and data mining*. ACM, 2013, pp. 659–667.

[13] R. Gemulla, E. Nijkamp, P. J. Haas, and Y. Sismanis, "Large-scale matrix factorization with distributed stochastic gradient descent," in *Proceedings of the 17th ACM SIGKDD international conference on Knowledge discovery and data mining*. ACM, 2011, pp. 69–77.

[14] M. Deshpande and G. Karypis, "Item-based top-n recommendation algorithms," *ACM Transactions on Information Systems (TOIS)*, vol. 22, no. 1, pp. 143–177, 2004.

[15] S. Rendle, C. Freudenthaler, Z. Gantner, and L. Schmidt-thieme, "Ls: Bpr: Bayesian personalized ranking from implicit feedback," in *In: Proceedings of the 25th Conference on Uncertainty in Artificial Intelligence (UAI*, 2009.