FISM: Factored Item Similarity Models for Top-N Recommender Systems

Santosh Kabbur
Computer Science & Engineering
University of Minnesota, Minneapolis, MN, USA
skabbur@cs.umn.edu

Xia Ning
NEC Laboratories America
Princeton, NJ, USA
xning@nec-labs.com

George Karypis
Computer Science & Engineering
University of Minnesota, Minneapolis, MN, USA
karypis@cs.umn.edu

ABSTRACT

The effectiveness of existing top-N recommendation methods decreases as the sparsity of the datasets increases. To alleviate this problem, we present an item-based method for generating top-N recommendations that learns the item-item similarity matrix as the product of two low dimensional latent factor matrices. These matrices are learned using a structural equation modeling approach, wherein the value being estimated is not used for its own estimation. A comprehensive set of experiments on multiple datasets at three different sparsity levels indicate that the proposed methods can handle sparse datasets effectively and outperforms other state-of-the-art top-N recommendation methods. The experimental results also show that the relative performance gains compared to competing methods increase as the data gets sparser.

Categories and Subject Descriptors

H.2.8 [**Database Applications**]: Data Mining

Keywords

recommender systems; topn; sparse data; item similarity

1. INTRODUCTION

Top-N recommender systems have been widely used in E-commerce applications to recommend ranked lists of items so as to help the users in identifying the items that best fit their personal tastes. Over the years, many algorithms have been developed to address the top-N recommender problem [12]. These algorithms make use of the user feedback (purchase, rating or review) to compute the recommendations. Typically these algorithms represent the feedback information as a user-purchase matrix and act on it. The existing methods can be broadly classified into two classes: collaborative filtering (CF) based methods and content based methods. User/Item co-rating information is utilized in collaborative methods to build models. One class of CF methods, referred to as nearest-neighborhood-based methods, compute the similarities between the users/items using the co-rating information and new items are recommended based on these similarity values. Another class of CF methods, referred to as model-based methods, employ a machine learning algorithm to build a model (in terms of similarities or latent factors), which is then used to perform the recommendation task. The state-of-the-art methods for rating prediction and top-N recommendation problem learn the relationship between items in the form of an item similarity matrix [8, 5, 11, 7]. In content based methods [6, 9], the features associated with users/items are used to build models.

Recently, a novel top-N recommendation method has been developed, called SLIM [7], which improves upon the traditional item-based nearest neighbor collaborative filtering approaches by learning directly from the data, a sparse matrix of aggregation coefficients that are analogous to the traditional item-item similarities. SLIM has been shown to achieve good performance on a wide variety of datasets and to outperform other state-of-the-art approaches. However, an inherent limitation of SLIM is that it can only model relations between items that have been co-purchased/co-rated by at least some users. As a result, it cannot capture transitive relations between items that are essential for good performance of item-based approaches in sparse datasets.

In this paper we propose a method, called FISM, which learns the item-item similarity matrix as a product of two low-dimensional latent factor matrices. This factored representation of the item-item similarity matrix allows FISM to capture and model relations between items even on very sparse datasets. Our experimental evaluation on multiple datasets and at different sparsity levels confirms that and shows that FISM performs better than SLIM and other state-of-the-art methods. Moreover, the relative performance gains increase with the sparsity of the datasets.

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

(i) extends the factored item-based methods to the top-N problem, which allow them to effectively handle sparse datasets;

(ii) estimates the factored item-based top-N models using a structural equation modeling approach;

(iii) estimates the factored item-based top-N models using both squared error and a ranking loss; and

KDD'13, August 11–14, 2013, Chicago, Illinois, USA.

(iv) investigates the impact of various parameters as they relate to biases, neighborhood agreement, and model's induced sparsity.

The rest of the paper is organized as follows. Section 2 introduces the notations used in the paper. In Section 3 the relevant existing methods are presented. Section 4 motivates the need for a better model and contrasts the proposed approach against existing schemes. In Section 5, the details of FISM models are presented. Section 6 provides the evaluation methodology and the data set characteristics. In Section 7 the results of the experimental evaluation are provided. Finally, Section 8 provides some concluding remarks.

2. 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 value is denoted by having a $\tilde{}$(tilde) over it (e.g., $\tilde{r}$) and an estimated value is denoted by having a $\hat{}$ (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 $n \times m$. 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 feedback information for user u on item i. $\mathbf{R}$ is a binary matrix. 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.

3. REVIEW OF RELEVANT RESEARCH

The methods developed in this work are motivated by two classes of methods that were recently developed for top-N recommendation and rating prediction.

The first method, SLIM, proposed by Ning et. al. [7], predicts the recommendation scores of a user u for all items as

$$\tilde{\mathbf{r}}_u = \mathbf{r}_u \mathbf{S}, \tag{1}$$

where $\mathbf{r}_u$ is the rating vector of u on all items and $\mathbf{S}$ is a $m \times m$ sparse matrix of aggregation coefficients.

Matrix $\mathbf{S}$ can be considered as an item-item similarity matrix, and as such the recommendation strategy employed by SLIM is similar in nature to that of the traditional item-based nearest-neighbor top-N recommendation approaches [3]. However, unlike these methods, SLIM directly estimates the similarity values from the data using a simultaneous regression approach, which is similar to structural equation modeling with no exogenous variables [10]. Specifically, SLIM estimates the sparse matrix $\mathbf{S}$ as the minimizer for the following regularized optimization problem:

$$\begin{aligned} \underset{\mathbf{S}}{\text{minimize}} \;\; & \frac{1}{2}\|\mathbf{R} - \mathbf{RS}\|_F^2 + \frac{\beta}{2}\|\mathbf{S}\|_F^2 + \lambda\|\mathbf{S}\|_1 \\ \text{subject to} \;\; & \mathbf{S} \geq 0\,, \;\; \text{diag}(\mathbf{S}) = 0, \end{aligned} \tag{2}$$

where $\|\mathbf{S}\|_F$ is the matrix Frobenius norm of $\mathbf{S}$ and $\|\mathbf{S}\|_1$ is the entry-wise ℓ_1-norm of $\mathbf{S}$. In Equation 2, $\mathbf{RS}$ is the estimated matrix of recommendation scores (i.e., $\tilde{\mathbf{R}}$). The constraint $\text{diag}(\mathbf{S}) = 0$ conforming to the structural equation modeling is also applied to ensure that r_{ui} is not used to compute r_{ui}. The non-negativity constraint is applied on $\mathbf{S}$ so that the learned $\mathbf{S}$ corresponds to positive aggregations over items. In order to learn a sparse $\mathbf{S}$, SLIM introduces the ℓ_1-norm of $\mathbf{S}$ as a regularizer in Equation 2 [13]. The matrix $\mathbf{S}$ learned by SLIM is referred to as SLIM's aggregation coefficient matrix. Extensive experiments in [7] have shown that SLIM outperforms the rest of the state-of-the-art top-N recommendation methods.

The second method is called NSVD and was developed by Paterek in [8]. This is a factored item-item collaborative filtering method developed for the rating prediction problem. In this method, an item-item similarity was learned as a product of two low-rank matrices, $\mathbf{P}$ and $\mathbf{Q}$, where $\mathbf{P} \in \mathbb{R}^{m \times k}$, $\mathbf{Q} \in \mathbb{R}^{m \times k}$, and $k \ll m$. This approach extends the traditional item-based neighborhood methods by learning the similarity between items as a product of their corresponding latent factors. Given two items i and j, the similarity $sim(i, j)$ between them is computed as the dot product between the corresponding factors from $\mathbf{P}$ and $\mathbf{Q}$ i.e., $sim(i, j) = \mathbf{p}_i \cdot \mathbf{q}_j^\mathsf{T}$. The rating for a given user u on item i is both predicted and estimated as

$$\hat{r}_{ui} = \tilde{r}_{ui} = b_u + b_i + \sum_{j \in \mathcal{R}_u^+} \mathbf{p}_j \mathbf{q}_i^\mathsf{T}, \tag{3}$$

where b_u and b_i are the user and item biases and $\mathcal{R}_u^+$ is the set of items rated by u. The parameters of this model are estimated as the minimizer to the following optimization problem:

$$\underset{\mathbf{P},\mathbf{Q}}{\text{minimize}} \;\; \frac{1}{2}\sum_{u \in \mathcal{C}} \sum_{i \in \mathcal{R}_u^+} \|r_{ui} - \hat{r}_{ui}\|_F^2 + \frac{\beta}{2}(\|\mathbf{P}\|_F^2 + \|\mathbf{Q}\|_F^2), \tag{4}$$

where $\hat{r}_{ui}$ is the estimated value for user u and item i (as in Equation 3).

In another method based on NSVD, Koren proposed a hybrid approach called SVD++ [5]. This method merged the idea of latent factor models and traditional neighborhood based models to learn similarities between users or items. Both these models (i.e., NSVD and SVD++) were evaluated by computing the root mean square error (RMSE) on the test ratings in the Netflix competition data set. Hence the goal of these models was to minimize the RMSE and only the non-zero entries of the rating matrix were used in training.

4. MOTIVATION

In real world scenarios, users typically provide feedback (purchase, rating or review) to only a handful of items out of possibly thousands or millions of items. This results in the user-item rating matrix becoming very sparse. Methods like SLIM (as well as traditional methods like ItemKNN [3]), which rely on learning similarities between items, fail to capture the dependencies between items that have not been co-rated by at least one user. It can be shown that the minimizer in Equation 2 will have $s_{ij} = 0$, if i and j have not been co-rated by at least one user. But two such items can be similar to each other by virtue of another item which is

similar to both of them (transitive relation). Methods based on matrix factorization, alleviate this problem by projecting the data onto a low dimensional space, thereby implicitly learning better relationships between the users and items (including items which are not co-rated). However, such methods are consistently out-performed by SLIM [7].

To overcome this problem, our proposed item-oriented FISM method uses a factored item similarity model similar in spirit to that used by NSVD and SVD++. Learning the similarity matrix by projecting the values in a latent space of much smaller dimensionality, implicitly helps to learn transitive relations between items. Hence, this model is expected to perform better even on sparse data, as it can learn relationships between items which are not co-rated.

Comparing FISM with NSVD, besides the fact that these two methods are designed to solve different problems (top-N vs rating prediction), their key difference lies in how the factored matrices are estimated. FISM employs a regression approach based on structural equation modeling in which, unlike NSVD (and SVD++), the known rating information for a particular user-item pair (r_{ui}) is not used when the rating for that item is being estimated. This impacts how the diagonal entries of the item-item similarity matrix corresponding to $\mathbf{S} = \mathbf{P}\mathbf{Q}^{\mathsf{T}}$ influence the estimation of the recommendation score. Diagonal entries in the item similarities matrix correspond to including an item's own value while computing the prediction for that item. NSVD does not exclude the diagonal entries while estimating the ratings during learning and prediction phases, while FISM explicitly excludes the diagonal entries while estimating. This shortcoming of NSVD impacts the quality of the estimated factors when the number of factors becomes large. In this case it can lead to rather trivial estimates, in which an item ends up recommending itself. This is illustrated in our experimental results (Section 7), which show that for a small number of factors, the two estimation approaches produce similar results, whereas as the number of factors increases moderately, FISM's estimation approach consistently and significantly outperforms the approach used by NSVD.

5. FISM- FACTORED ITEM SIMILARITY METHODS

In FISM, the recommendation score for a user u on an unrated item i (denoted by $\tilde{r}_{ui}$) is calculated as an aggregation of the items that have been rated by u with the corresponding product of $\mathbf{p}_j$ latent vectors from $\mathbf{P}$ and the $\mathbf{q}_i$ latent vector from $\mathbf{Q}$. That is,

$$\tilde{r}_{ui} = b_u + b_i + (n_u^+)^{-\alpha} \sum_{j \in \mathcal{R}_u^+} \mathbf{p}_j \mathbf{q}_i^{\mathsf{T}}, \tag{5}$$

where $\mathcal{R}_u^+$ is the set of items rated by user u, $\mathbf{p}_j$ and $\mathbf{q}_i$ are the learned item latent factors, n_u^+ is the number of items rated by u, and α is a user specified parameter between 0 and 1.

The term $(n_u^+)^{-\alpha}$ in Equation 5 is used to control the degree of agreement between the items rated by the user with respect to their similarity to the item whose rating is being estimated (i.e., item i). To better understand this, consider the case in which $\alpha = 1$. In this case (excluding the bias), the predicted rating is the average similarities between the items rated by the user (i.e., $\mathcal{R}_u^+$) and item i. Item i will get a high rating if nearly all of the items in $\mathcal{R}_u^+$ are similar to i. On the other hand, if $\alpha = 0$, then the predicted rating is the aggregate similarity between i and the items in $\mathcal{R}_u^+$. Thus, i can be rated high, even if only one (or few) of the items in $\mathcal{R}_u^+$ are similar to i. These two settings represent different extremes and we believe that in most cases the right choice will be somewhere in between. That is, the item for which the rating is being predicted needs to be similar to a substantial number of items to get a high rating. To capture this difference, we have introduced the parameter α, to control the number of neighborhood items that need to be similar for an item to get the high rating. The value of α is expected to be dependent on the characteristics of the dataset and its best performing value is determined empirically.

We developed two different types of FISM models that use different loss functions and associated optimization methods, which are described in the next two sections.

5.1 FISMrmse

In FISMrmse, we compute the loss using the squared error loss function, given by

$$\mathcal{L}(\cdot) = \sum_{i \in \mathcal{D}} \sum_{u \in \mathcal{C}} (r_{ui} - \hat{r}_{ui})^2, \tag{6}$$

where r_{ui} is the ground truth value and $\hat{r}_{ui}$ is the estimated value. The estimated value $\hat{r}_{ui}$, for a given user u and item i is computed as

$$\hat{r}_{ui} = b_u + b_i + (n_u^+ - 1)^{-\alpha} \sum_{j \in \mathcal{R}_u^+ \setminus \{i\}} \mathbf{p}_j \mathbf{q}_i^{\mathsf{T}}, \tag{7}$$

where $\mathcal{R}_u^+ \setminus \{i\}$ is the set of items rated by user u, excluding the current item i, whose value is being estimated. This exclusion is done to conform to regression models based on structural equation modeling. This is also one of the important differences between FISM and other factored item similarities model (like NSVD and SVD++) as discussed in Section 4.

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

$$\underset{\mathbf{P},\mathbf{Q}}{\text{minimize}} \ \frac{1}{2} \sum_{u,i \in R} \|r_{ui} - \hat{r}_{ui}\|_F^2 + \frac{\beta}{2}(\|\mathbf{P}\|_F^2 + \|\mathbf{Q}\|_F^2) + \frac{\lambda}{2}\|\mathbf{b_u}\|_2^2 + \frac{\gamma}{2}\|\mathbf{b_i}\|_2^2, \tag{8}$$

where the vectors $\mathbf{b_u}$ and $\mathbf{b_i}$ correspond to the vector of user and item biases, respectively. The regularization terms are used to prevent overfitting and β, λ and γ are the regularization weights for latent factor matrices, user bias vector and item bias vector respectively.

Following the common practices for top-N recommendation [2, 7], note that the loss function in Equation 6 is computed over all entries of $\mathbf{R}$ (i.e., both rated and unrated). This is in contrast with rating prediction methods, which compute the loss over only the rated items. However, in order to reduce the computational requirements for optimization, the zero entries are sampled and used along with all the non-zero values of $\mathbf{R}$. During each iteration of learning, $\rho \cdot nnz(R)$ zeros are sampled and used for optimization. Here ρ is a constant and $nnz(R)$ is the number of non-zero entries in $\mathbf{R}$. Our experimental results indicate that a small value of ρ (in the range $3 - 15$) is sufficient to produce the

best model. This sampling strategy makes FISMrmse computationally efficient.

The optimization problem of Equation 8 is solved using a Stochastic Gradient Descent (SGD) algorithm [1]. Algorithm 1 provides the detailed procedure and gradient update rules. $\mathbf{P}$ and $\mathbf{Q}$ are initialized with small random values as the initial estimate (line 6). In each iteration of SGD (Lines 8 – 26), based on the sampling factor (ρ), a different set of zeros are sampled and used for training along with the non-zero entries of $\mathbf{R}$. This process is repeated until the error on the validation set does not decrease further or the number of iterations has reached a predefined threshold.

Algorithm 1 FISMrmse:Learn.

1: **procedure** FISMrmse_LEARN
2: $\eta \leftarrow$ learning rate
3: $\beta \leftarrow \ell_F$ regularization weight
4: $\rho \leftarrow$ sample factor
5: $iter \leftarrow 0$
6: Init $\mathbf{P}$ and $\mathbf{Q}$ 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*($\mathbf{R}$, ρ)
10: $\mathcal{R}' \leftarrow$ *RandomShuffle*($\mathcal{R}'$)
11:
12: **for all** $r_{ui} \in \mathcal{R}'$ **do**
13: $\mathbf{x} \leftarrow (n_u^+ - 1)^{-\alpha} \sum_{j \in \mathcal{R}_u^+ \setminus \{i\}} \mathbf{p}_j$
14:
15: $\tilde{r}_{ui} \leftarrow b_u + b_i + \mathbf{q}_i^\mathsf{T}\mathbf{x}$
16: $e_{ui} \leftarrow r_{ui} - \tilde{r}_{ui}$
17: $b_u \leftarrow b_u + \eta \cdot (e_{ui} - \lambda \cdot b_u)$
18: $b_i \leftarrow b_i + \eta \cdot (e_{ui} - \gamma \cdot b_i)$
19: $\mathbf{q}_i \leftarrow \mathbf{q}_i + \eta \cdot (e_{ui} \cdot \mathbf{x} - \beta \cdot \mathbf{q}_i)$
20:
21: **for all** $j \in \mathcal{R}_u^+ \setminus \{i\}$ **do**
22: $\mathbf{p}_j \leftarrow \mathbf{p}_j + \eta \cdot (e_{ui} \cdot (n_u^+ - 1)^{-\alpha} \cdot \mathbf{q}_i - \beta \cdot \mathbf{p}_j)$
23: **end for**
24: **end for**
25: $iter \leftarrow iter + 1$
26: **end while**
27:
28: **return** $\mathbf{P}, \mathbf{Q}$
29: **end procedure**

5.2 FISMauc

As a second loss function, we consider a ranking error based loss function. This is motivated by the fact that the Top-N recommendation problem deals with ranking the items in the right order, unlike the rating prediction problem where minimizing the RMSE is the goal. We used a ranking loss function based on Bayesian Personalized Ranking (BPR) [11], which optimizes the area under the curve (AUC). Given user's rated items in $\mathcal{R}_u^+$ and unrated items in $\mathcal{R}_u^-$, the overall ranking loss is given by

$$\mathcal{L}(\cdot) = \sum_{u \in \mathcal{C}} \sum_{i \in \mathcal{R}_u^+, j \in \mathcal{R}_u^-} ((r_{ui} - r_{uj}) - (\hat{r}_{ui} - \hat{r}_{uj}))^2, \quad (9)$$

where the estimates $\hat{r}_{ui}$ and $\hat{r}_{uj}$ are computed as in Equation 7. As we can see in Equation 9, the error is computed as the relative difference between the actual non-zero and zero entries and the difference between their corresponding estimated values. Thus, this loss function focuses not on estimating the right value, but on the ordering of the zero and non-zero values.

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

$$\underset{\mathbf{P},\mathbf{Q}}{\text{minimize}} \ \frac{1}{2} \sum_{u \in \mathcal{C}} \sum_{i \in \mathcal{R}_u^+, j \in \mathcal{R}_u^-} \|(r_{ui} - r_{uj}) - (\hat{r}_{ui} - \hat{r}_{uj})\|_F^2 + \frac{\beta}{2}(\|\mathbf{P}\|_F^2 + \|\mathbf{Q}\|_F^2) + \frac{\gamma}{2}(\|\mathbf{b_i}\|_2^2), \quad (10)$$

where the terms mean the same as in Equation 8. Note that there are no user bias terms (i.e., b_u), since the terms cancel out when taking the difference of the ratings. For each user, FISMauc computes loss over all possible pairs of entries in $\mathcal{R}_u^+$ and $\mathcal{R}_u^-$. Similar to FISMrmse, to reduce the computational requirements, zero entries for each user are sampled from $\mathcal{R}_u^-$ based on sample factor (ρ).

The optimization problem in Equation 10 is solved using a Stochastic Gradient Descent (SGD) based algorithm. Algorithm 2 provides the detailed procedure.

5.3 Scalability

The scalability of these methods consists of two aspects. First, the training phase needs to be scalable, so that these methods can be used with larger datasets. Second, the time taken to compute the recommendations needs to be reduced and ideally made independent of the total number of recommendable items. Regarding the first aspect, the training for both FISMrmse and FISMauc is done using SGD algorithm. The gradient computations and updates of SGD can be parallelized and hence these algorithms can be easily applied to larger datasets. In [4], a distributed SGD is proposed. A similar algorithm with modifications can be used to scale the FISM methods to larger datasets. The main difference is in computing the rows of $\mathbf{P}$ that can be updated independently in parallel. There are also software packages like Spark[1] which can be used to implement SGD based algorithms on a large cluster of processing nodes.

For computing the recommendations efficiently during run time, methods like SLIM enforce sparsity constraint on $\mathbf{S}$ while learning and utilizes this sparsity structure to reduce the number of computations during run time. However, in FISM, the factored matrices learned are usually dense and as such, the predicted vector $\tilde{\mathbf{r}}_u$ will be dense (because $\mathbf{P}\mathbf{Q}^\mathsf{T}$ is dense). Sparsity in $\tilde{\mathbf{r}}_u$ can be introduced by computing $\mathbf{S} = \mathbf{P}\mathbf{Q}^\mathsf{T}$ and then setting the smaller values to zero. One systematic way of doing this is to selectively retain only those non-zero entries which contribute the most to the length of the item similarities vector represented by the column in $\mathbf{S}$ to which the entry belongs. The impact of this sparsification is further explored in the experimental results.

6. EXPERIMENTAL EVALUATION

6.1 Data Sets

We evaluated the performance of FISM on three different real datasets, namely ML100K, Netflix and Yahoo Music.

[1]http://spark-project.org/

Algorithm 2 FISMauc:Learn.

procedure FISMauc_LEARN
  $\eta \leftarrow$ learning rate
  $\beta \leftarrow \ell_F$ regularization weight
  $\rho \leftarrow$ number of sampled zeros
  $iter \leftarrow 0$
  Init $\mathbf{P}$ and $\mathbf{Q}$ with random values in (-0.001, 0.001)

  while $iter < maxIter$ or error on validation set decreases **do**
    for all $u \in \mathcal{C}$ **do**
      for all $i \in \mathcal{R}_u^+$ **do**
        $\mathbf{x} \leftarrow 0$
        $\mathbf{t} \leftarrow (n_u^+ - 1)^{-\alpha} \sum_{j \in \mathcal{R}_u^+ \setminus \{i\}} \mathbf{p}_j$
        $\mathcal{Z} \leftarrow$*SampleZeros(ρ)*

        for all $j \in \mathcal{Z}$ **do**
          $\tilde{r}_{ui} \leftarrow b_i + \mathbf{t} \cdot \mathbf{q}_i^\top$
          $\tilde{r}_{uj} \leftarrow b_j + \mathbf{t} \cdot \mathbf{q}_j^\top$
          $r_{uj} \leftarrow 0$
          $e \leftarrow (r_{ui} - r_{uj}) - (\tilde{r}_{ui} - \tilde{r}_{uj})$
          $b_i \leftarrow b_i + \eta \cdot (e - \gamma \cdot b_i)$
          $b_j \leftarrow b_j - \eta \cdot (e - \gamma \cdot b_j)$
          $\mathbf{q}_i \leftarrow \mathbf{q}_i + \eta \cdot (e \cdot \mathbf{t} - \beta \cdot \mathbf{q}_i)$
          $\mathbf{q}_j \leftarrow \mathbf{q}_j - \eta \cdot (e \cdot \mathbf{t} - \beta \cdot \mathbf{q}_j)$
          $\mathbf{x} \leftarrow \mathbf{x} + e \cdot (\mathbf{q}_i - \mathbf{q}_j)$
        end for
      end for

      for all $j \in \mathcal{R}_u^+ \setminus \{i\}$ **do**
        $\mathbf{p}_j \leftarrow \mathbf{p}_j + \eta \cdot (\frac{1}{\rho} \cdot (n_u^+ - 1)^{-\alpha} \cdot \mathbf{x} - \beta \cdot \mathbf{p}_j)$
      end for
    end for

    $iter \leftarrow iter + 1$
  end while

  return $\mathbf{P}, \mathbf{Q}$
end procedure

ML100K is the subset of data obtained from the MovieLens[2] research project, Netflix is a subset of data extracted from Netflix Prize dataset[3] and finally Yahoo Music is the subset of data obtained from Yahoo! Research Alliance Webscope program[4]. For each of the three datasets, we created three different versions at different sparsity levels. This was done to specifically evaluate the performance of FISM on sparse datasets. For each dataset, we started by randomly choosing a subset of users and items from the main dataset. These datasets are represented with a *'-1'* suffix. Keeping the same set of users and items, the first sparser version of the datasets with the *'-2'* suffix are created by randomly removing entries from the first datasets' user-item matrices. The second sparser version of the datasets with the *'-3'* suffix are similarly created by randomly removing entries from second datasets' user-item matrices. Note that all these datasets have rating values and we converted them into implicit feedback by setting the positive entries to 1. The characteristics of all the datasets is summarized in Table 1.

Table 1: Datasets.

Dataset	#Users	#Items	#Ratings	Rsize	Csize	Density
ML100K-1	943	1,178	59,763	63.99	50.73	5.43%
ML100K-2	943	1,178	39,763	42.57	33.75	3.61%
ML100K-3	943	1,178	19,763	21.16	16.78	1.80%
Netflix-1	6,079	5,641	429,339	70.63	76.11	1.25%
Netflix-2	6,079	5,641	221,304	36.40	39.23	0.65%
Netflix-3	6,079	5,641	110,000	18.10	19.50	0.32%
Yahoo-1	7,558	3,951	282,075	37.32	71.39	0.94%
Yahoo-2	7,558	3,951	149,050	19.72	37.72	0.50%
Yahoo-3	7,558	3,951	75,000	9.92	18.98	0.25%

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 × #Items)).

6.2 Evaluation Methodology

To evaluate the performance of the proposed model 5-fold Leave-One-Out-Cross-Validation (LOOCV) is employed. For each fold, dataset is split into training and test set by randomly selecting one item for each user and placing it in the test set. The rest of the data is used as the training set. Such a training set is used to build the model and the trained model is then 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. For all the results presented in this paper, N is equal to 10.

The recommendation quality is measured using Hit Rate (HR) and Average Reciprocal Hit Rank (ARHR) [3]. HR is defined as

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

where #users is the total number of test users and #hits is the number of users for which the model was successfully able to recall the test item in the size-N recommendation list. 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.

[2]http://www.grouplens.org/node/12
[3]http://www.netflixprize.com/
[4]http://research.yahoo.com/academic_relations

6.3 Comparison Algorithms

We compare the performance of FISM against that achieved by ItemKNN (cos) [3], ItemKNN (cprob) [3], ItemKNN (log)[5], PureSVD [2], BPRkNN, BPRMF [11] and SLIM [7]. We also compare the performance of a set of FISM models in which the estimated value of $\hat{r}_{ui}$ is given by Equation 5 ($\tilde{r}_{ui}$) which also includes item i. This is done to access the performance improvements by the new estimation approach. 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.

7. RESULTS

The experimental evaluation consists of two parts. First, we study the effect of various model parameters of FISM on the recommendation performance. Specifically, we look at how bias, neighborhood agreement, induced sparsity, estimation approach and non-negativity affect the top-N performance. Due to the lack of space, we present these studies only on the ML100K-3 (represented as ML100K), Yahoo-2 (represented as Yahoo) and Netflix-3 (represented as Netflix) datasets. However the same results and conclusions carry over to the rest of the datasets as well. These datasets are chosen to represent the datasets from different sources and at different sparsity levels. Unless specified all results in the first set of experiments are based on FISMrmse.

Second, we present the comparison results with other competing methods (Section 6.3) on all the datasets. We also compare the performance of FISM for different values of N (as in top-N) and finally we compare the performance of FISM with respect to data sparsity.

7.1 Effect of Bias

In FISM's model, the user and item biases are learned as part of the model. In this study we compare the influence of user and item biases on the overall performance of the model. We compare the following four different schemes, $NoBias$ - where no user or item bias is learned as part of the model, $UserBias$ - only the user bias is learned, $ItemBias$ - only the item is learned and $User\&ItemBias$ - where both user and item biases are learned. The results are presented in Table 2. The results indicate that the biases affect the overall performance, with item bias leading to the greatest gains in performance.

7.2 Effect of Neighborhood Agreement

In this study, we compare the effect of neighborhood agreement on the recommendation performance. Keeping the rest of the parameters constant, we did a full parameter study for different values of the normalization constant (α). The results for both FISMrmse and FISMauc methods are presented in Figure 1. From the results, we can see that the best performance is obtained when the value of α is in the range 0.4 to 0.5, indicating that, on average, for the items to be rated high and hence recommended, a substantial number of neighborhood items need to have a high similarity value. Another interesting result to note is that the performance of the FISMauc is more stable than the corresponding FISMrmse's performance for the different values of α. This can be attributed to the fact that FISMauc minimizes the ranking loss, where the user related biases (number of items rated) is to a large extent nullified.

[5]Part of Mahout library (http://mahout.apache.org/)

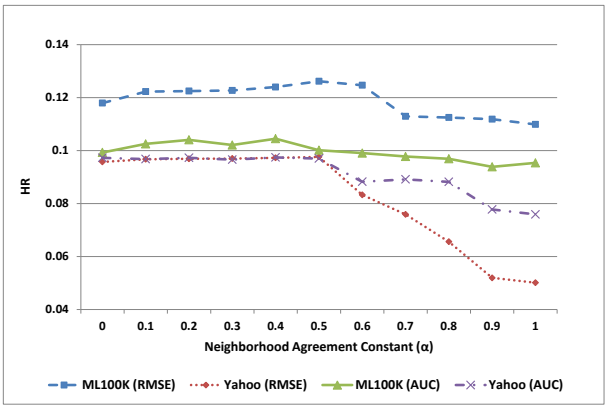


Figure 1: Effect of neighborhood agreement on performance.

7.3 Performance of Induced Sparsity on S

Figure 2 shows FISM's performance on the sparsified $\mathbf{S}$ matrix. The x-axis represents the density of $\mathbf{S}$ after sparsifying the matrix as explained in Section 5.3. We can see that there is only a minimal reduction in the recommendation performance up to a density in the range 0.1 to 0.15. At the same time, the average time required to compute the recommendations for each user reduces drastically. This gain in recommendation efficiency comes at a very small cost in terms of recommendation performance, which may justify it's use in applications in which high-throughput recommendation rates are required.

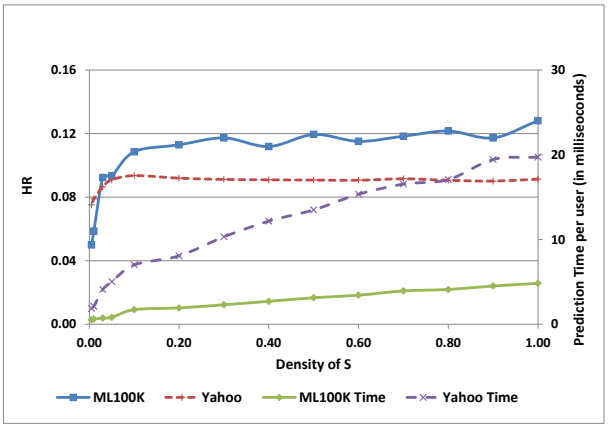


Figure 2: Performance of induced Sparsity on $\mathbf{S}$.

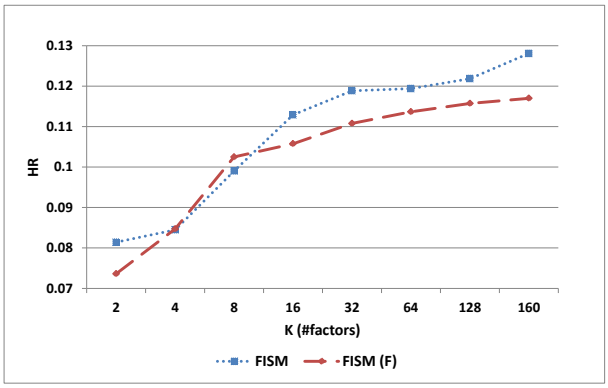
7.4 Effect of Estimation Approach

To study the effect of FISM's estimation approach, which excludes the item's own rating during estimation, we compare the performance of FISM with an approach which is the same as FISM except that it includes the rating's own value during estimation. We call this method FISM(F), where F corresponds to similar approaches used in factorization (F) based schemes for rating prediction (NSVD and SVD++).

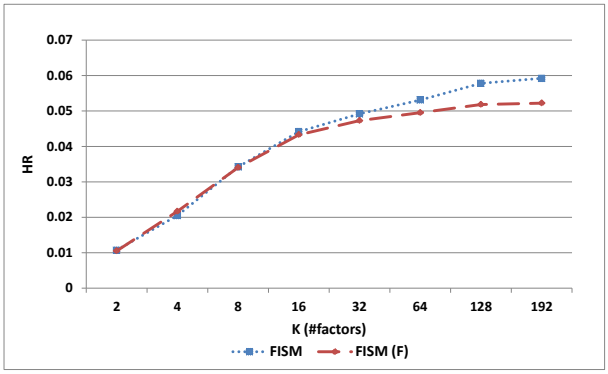
Keeping the rest of the parameters constant, the number of latent factors k is varied and the performance of FISM and

Table 2: Performance of different bias schemes.

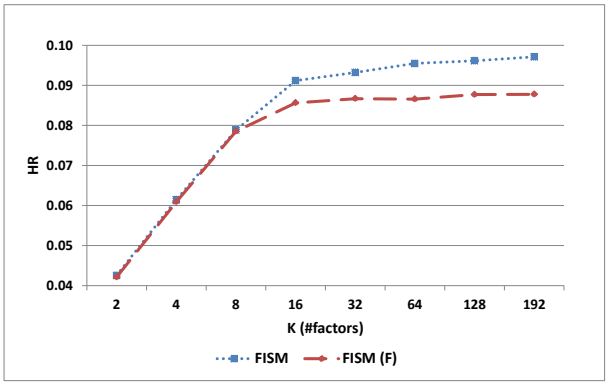
Scheme	ML100K				Yahoo			
	Beta	Lambda	Gamma	HR	Beta	Lambda	Gamma	HR
NoBias	8e-4	-	-	0.1281	2e-5	-	-	0.0974
UserBias	6e-4	0.1	-	0.1336	4e-5	0.1	-	0.1012
ItemBias	2e-4	-	0.01	0.1401	4e-5	-	1e-4	0.1007
User&ItemBias	6e-4	0.1	1e-4	0.1090	4e-5	0.1	1e-4	0.0977



(a) ML100K dataset.



(b) Netflix dataset.



(c) Yahoo dataset.

Figure 3: Effect of estimation approach on performance.

FISM(F) is compared. Figure 3 shows the results for different datasets. We can see that, for smaller values of k, the performance of both the schemes is very similar. However, when the value of k is increased, FISM starts to perform better than FISM(F) and the gap between the performance of the methods increases as the value of k increases. This confirms the fact that the estimation approach used by FISM is superior to that used by approaches like NSVD and SVD++ and helps to avoid trivial solutions when the number of factors becomes large.

7.5 Effect of Non-Negativity

SLIM enforces non-negativity constraint to ensure that the learned item similarities correspond to positive relationships. SLIM has also shown that the adding such a constraint helps to improve the recommendation performance. In FISM, no such explicit constraint is enforced. We implemented the FISMrmse and FISMauc algorithms with non-negativity constraints and, to our surprise there was no improvement in the performance. In fact, the performance dropped considerably (HR of 0.0933 for ML100K and 0.0848 for Yahoo, compared to 0.1281 and 0.0974 without the constraints).

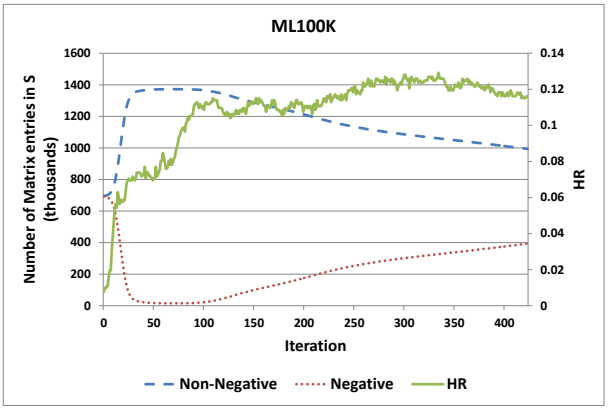


Figure 4: Non-negative and negative entries in $\mathbf{S}$.

To gain some insights on this issue, we observed the properties of $\mathbf{S} = \mathbf{P}\mathbf{Q}^T$ during the learning process. In particular, we observed the number of negative and non-negative entries in $\mathbf{S}$ during each iteration of the learning process. The observations are plotted in Figure 4. We can see that initially the number of negative and non-negative entries is similar, but as the model starts to learn, the number of negative entries decreases drastically. The best performance is obtained when the number of negative entries is significantly smaller compared to the number of non-negative entries (of the order 1:3). This shows that, even though the non-negativity

constraint is not explicitly enforced in FISM, the model still learns the majority of the similarity values as non-negative entries.

7.6 Comparison With Other Approaches

Table 3 shows the overall performance of FISM in comparison to other state-of-the-art algorithms for the top-N recommendation task. For each of the methods, the following parameter space was explored and the best performing model in that parameter space is reported. For all the ItemKNN based methods and PureSVD, parameter k was selected from the range 2 to 800. For ItemKNN (cprob), the α parameter was selected from the range 0 to 1 with a step size of 0.05. For BPRkNN, the learning rate and λ were selected from the range 10^{-5} to 1.0 with a multiplicative increment of 10. For BPRMF, the number of latent factors was selected from the range 2 to 800 and the learning rate from the range 10^{-5} to 1.0 with a multiplicative increment of 10. For FISM, both β and λ were selected from the range 10^{-5} to 30.

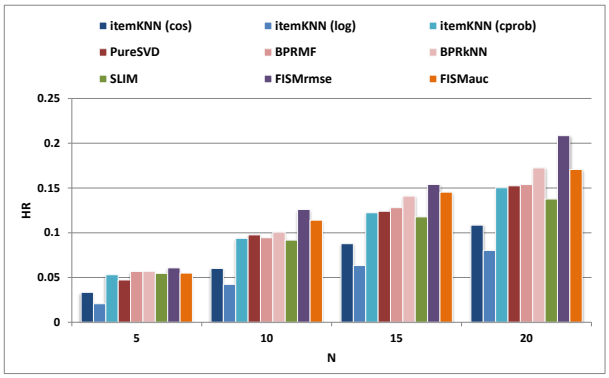
The results in Table 3 show that FISM performs better than all the other methods across all the datasets. For many of these datasets, the improvements achieved by FISM against the next best performing schemes are quite substantial. In terms of the two loss functions, quite surprisingly, the RMSE loss (FISMrmse) achieved better performance than the AUC loss (FISMauc). This is contrary to the results reported by other studies and we are currently investigating it.

Note that for all the results presented so far, the number of top-N items chosen is 10 (i.e., $N = 10$). Figure 5 shows the performance achieved by the various schemes for different values of N. These results are fairly consistent with those presented in Table 3, with FISM performing the best.

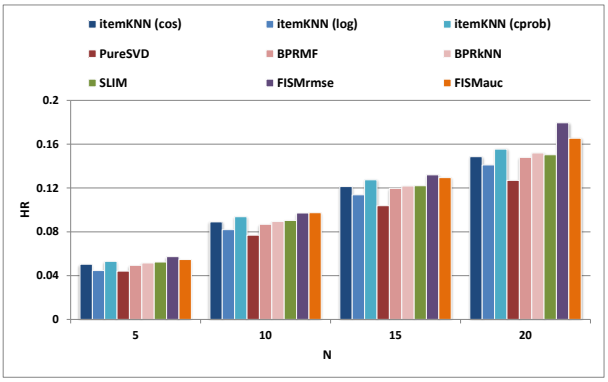
To better illustrate the gains achieved by FISM over the other competing approaches as the sparsity of the datasets increases, Figure 6 shows the percentage improvement achieved by FISM against the next best performing scheme for each dataset across the three sparsity levels. These results show that, as the datasets become sparser, the relative performance of FISM (in terms of HR) increases and, on the sparsest datasets, outperforms the next best scheme by at least 24%.

8. CONCLUSION

In this paper, we presented a factored item similarity based method (FISM) for the top-N recommendation problem. FISM learns the item similarities as the product of two matrices, allowing it to generate high quality recommendations even on sparse datasets. The factored representation is estimated using a structural equation modeling approach, which leads to better estimators as the number of factors increases. We conducted a comprehensive set of experiments on multiple datasets at different sparsity levels and compared FISM's performance against that of other state-of-the-art top-N recommendation algorithms. The results showed that FISM outperforms the rest of the methods and the performance gaps increases as the datasets become sparser. For faster recommendation, we showed that sparsity can be induced in the resulting item similarity matrix with minimal reduction in the recommendation quality.



(a) ML100K dataset.



(b) Yahoo dataset.

Figure 5: Performance for different values of N.

9. REFERENCES

[1] L. Bottou. Online algorithms and stochastic approximations. In D. Saad, editor, *Online Learning and Neural Networks*. Cambridge University Press, Cambridge, UK, 1998. revised, oct 2012.

[2] 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, 2010.

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

[4] 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*, pages 69–77. ACM, 2011.

[5] 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*, pages 426–434. ACM, 2008.

[6] 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*, pages 195–204. ACM, 2000.

[7] X. Ning and G. Karypis. Slim: Sparse linear methods for top-n recommender systems. In *Data Mining*

Table 3: Comparison of performance of *top-N* recommendation algorithms with FISM.

Method	ML100K-1					ML100K-2					ML100K-3				
	Params			HR	ARHR	Params			HR	ARHR	Params			HR	ARHR
ItemKNN (cos)	100	-	-	0.1604	0.0578	100	-	-	0.1214	0.0393	100	-	-	0.0602	0.0193
ItemKNN (log)	100	-	-	0.1047	0.0336	100	-	-	0.0809	0.0250	100	-	-	0.0424	0.0116
ItemKNN (cprob)	500	0.6	-	0.1711	0.0581	500	0.3	-	0.1308	0.0440	400	0.1	-	0.0938	0.0293
PureSVD	10	-	-	0.1700	0.0594	10	-	-	0.1362	0.0438	5	-	-	0.0438	0.0316
BPRkNN	1e-4	0.01	-	0.1621	0.0564	1e-5	0.01	-	0.1272	0.0447	1e-5	14	-	0.1006	0.0319
BPRMF	400	0.1	-	0.1610	0.0512	700	0.1	-	0.1224	0.0407	700	0.25	-	0.0943	0.0305
SLIM	0.1	20	-	0.1782	0.0620	0.01	18	-	0.1283	0.0448	1e-4	14	-	0.0919	0.0303
FISMrmse	96	2e-5	0.001	0.1908	0.0641	64	8e-4	0.01	0.1482	0.0462	96	8e-4	0.001	0.1260	0.0384
FISMauc	64	0.001	1e-4	0.1518	0.0504	144	2e-5	5e-5	0.1304	0.0424	144	8e-5	1e-5	0.1140	0.0340

Method	Netflix-1					Netflix-2					Netflix-3				
	Params			HR	ARHR	Params			HR	ARHR	Params			HR	ARHR
ItemKNN (cos)	100	-	-	0.1516	0.0689	100	-	-	0.0849	0.0316	100	-	-	0.0374	0.0123
ItemKNN (log)	100	-	-	0.0630	0.0240	100	-	-	0.0838	0.0303	100	-	-	0.0188	0.0062
ItemKNN (cprob)	20	0.5	-	0.1555	0.0678	500	0.5	-	0.0879	0.0326	200	0.1	-	0.0461	0.0162
PureSVD	600	-	-	0.1783	0.0865	400	-	-	0.0807	0.0297	400	-	-	0.0382	0.0131
BPRkNN	1e-3	1e-4	-	0.1678	0.0781	1e-4	1	-	0.0889	0.0329	0.01	1e-3	-	0.0439	0.0148
BPRMF	800	0.1	-	0.1638	0.0719	700	0.1	-	0.0862	0.0318	5	0.01	-	0.0454	0.0153
SLIM	1e-3	8	-	0.2025	0.1008	0.1	8	-	0.0947	0.0374	1e-4	12	-	0.0422	0.0149
FISMrmse	192	2e-5	0.001	0.2118	0.1107	192	6e-5	0.001	0.1041	0.0386	128	6e-5	0.001	0.0578	0.0185
FISMauc	192	1e-5	1e-4	0.2095	0.1016	240	2e-5	1e-4	0.0979	0.0341	160	4e-4	5e-4	0.0548	0.0177

Method	Yahoo-1					Yahoo-2					Yahoo-3				
	Params			HR	ARHR	Params			HR	ARHR	Params			HR	ARHR
ItemKNN (cos)	100	-	-	0.1344	0.0502	100	-	-	0.0890	0.0295	100	-	-	0.0366	0.0116
ItemKNN (log)	100	-	-	0.1046	0.0358	100	-	-	0.0820	0.0261	100	-	-	0.0489	0.0153
ItemKNN (cprob)	500	0.6	-	0.1387	0.0510	200	0.4	-	0.0908	0.0313	20	0.1	-	0.0571	0.0187
PureSVD	50	-	-	0.1229	0.0459	20	-	-	0.0769	0.0257	20	-	-	0.0494	0.0154
BPRkNN	1e-3	1e-4	-	0.1432	0.0528	1e-3	1e-4	-	0.0894	0.0304	0.1	0.01	-	0.0549	0.0183
BPRMF	700	0.1	-	0.1337	0.0473	700	0.1	-	0.0869	0.0288	10	0.01	-	0.0530	0.0169
SLIM	0.1	12	-	0.1454	0.0542	1e-3	12	-	0.0904	0.0304	0.1	2	-	0.0491	0.0159
FISMrmse	192	1e-4	0.001	0.1522	0.0542	192	2e-5	5e-4	0.0971	0.0371	160	0.002	0.001	0.0740	0.0230
FISMauc	144	8e-5	1e-4	0.1426	0.0488	160	2e-5	5e-4	0.0974	0.0315	176	2e-4	0.001	0.0722	0.0228

Columns corresponding to "params" indicate the model parameters for the corresponding method. For ItemKNN (cos) and ItemKNN (log) methods, the parameter is the number of neighbors. For ItemKNN (cprob), the parameters are the number of neighbors and α. For PureSVD method, the parameter is the number of singular values used. For BPRkNN method, the parameters are the learning rate and λ. For BPRMF method, the parameters are the number of latent factors and the learning rate. For SLIM, the parameters are β and λ and for FISM the parameters are number of latent factors (k), regularization weight (β) and learning rate (η). The columns corresponding to HR and ARHR represent the hit rate and average reciprocal hit-rank metrics, respectively. Underlined numbers represent the best performing model measured in terms of HR for each dataset.

(ICDM), 2011 IEEE 11th International Conference on, pages 497–506. IEEE, 2011.

[8] A. Paterek. Improving regularized singular value decomposition for collaborative filtering. In *Proceedings of KDD Cup and Workshop*, volume 2007, pages 5–8, 2007.

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

[10] J. Pearl. *Causality: models, reasoning and inference*, volume 29. Cambridge Univ Press, 2000.

[11] 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.

[12] 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*, 1, 2011.

[13] R. Tibshirani. Regression shrinkage and selection via the lasso. *Journal of the Royal Statistical Society. Series B (Methodological)*, pages 267–288, 1996.

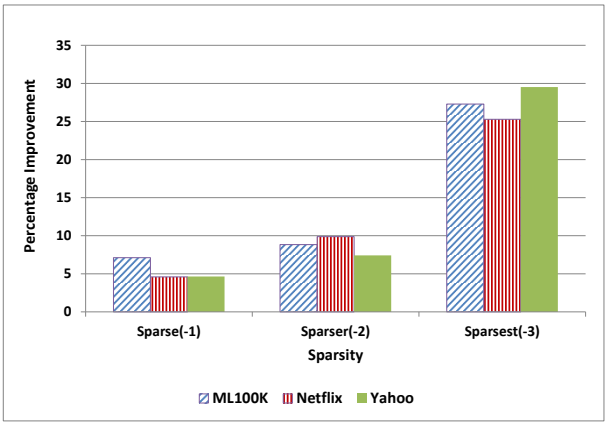


Figure 6: Effect of sparsity on performance for various datasets.