User-Specific Feature-based Similarity Models for Top-n Recommendation of New Items

ASMAA ELBADRAWY, University of Minnesota
GEORGE KARYPIS, University of Minnesota

Recommending new items for suitable users is an important yet challenging problem due to the lack of preference history for the new items. Non-collaborative user modeling techniques that rely on the item features can be used to recommend new items. However, they only use the past preferences of each user to provide recommendations for that user. They do not utilize information from the past preferences of other users which can potentially be ignoring useful information. More recent factor models transfer knowledge across users using their preference information in order to provide more accurate recommendations. These methods learn a low rank approximation for the preference matrix which can lead to loss of information. Moreover, they might not be able to learn useful patterns given very sparse datasets. In this work we present UFSM, a method for top-n recommendation of new items given binary user preferences. UFSM learns **U**ser-specific **F**eature-based item-**S**imilarity **M**odels and its strength lies in combining two points: (i) exploiting preference information across all users to learn multiple global item similarity functions, and (ii) learning user-specific weights that determine the contribution of each global similarity function in generating recommendations for each user. UFSM can be considered as a sparse high-dimensional factor model where the previous preferences of each user are incorporated within his latent representation. This way UFSM combines the merits of item similarity models that capture local relations among items and factor models that learn global preference patterns. A comprehensive set of experiments was conduced to compare UFSM against state-of-the-art collaborative factor models and non-collaborative user modeling techniques. Results show that UFSM outperforms other techniques in terms of recommendation quality. UFSM manages to yield better recommendations even with very sparse datasets. Results also show that UFSM can efficiently handle high-dimensional as well as low-dimensional item feature spaces.

Categories and Subject Descriptors: H.3.3 [**Information Search and Retrieval**]: Information filtering; I.2.6 [**Learning**]: Parameter learning

General Terms: Algorithms, Experimentation, Measurement, Performance

Additional Key Words and Phrases: Recommender systems, Cold start, Top-n, Item content, Item features, Item similarity

ACM Reference Format:
Asmaa Elbadrawy, and George Karypis, 2013. Feature-based Similarity Models for Top-n Recommendation of New Items *ACM Trans. Intelligent Sys. and Tech.* 9, 4, Article 39 (March 2013), 21 pages.
DOI: http://dx.doi.org/10.1145/0000000.0000000

1. INTRODUCTION

Top-n Recommender systems are used in many online applications in order to identify items of interest for the different users. These systems rely on historical information as to which users liked which items in order to identify the sets of items to be recommended. However, in most applications of recommender systems, new items are con-

Author's addresses: A. Elbadrawy, Computer Science Department, University of Minnesota; G.Karypis, Computer Science Department, University of Minnesota

DOI: http://dx.doi.org/10.1145/0000000.0000000

tinuously added. For example, new products are introduced, new books and articles are written, and news stories break. Being able to provide high quality recommendations for these new items will lead to purchasing a new product, reading a recently published scientific article, or watching a recently released movie. For new items, recommender systems that rely on the items' historical information cannot be used, as simply such information is not available. A special type of recommender systems, referred to as *cold-start recommender systems*, are used to recommend such new items.

Methods for cold-start recommender systems require that the items being recommended are represented via a set of features that capture their intrinsic characteristics. For example, a movie can have features about its genre, actors and director; a book can have a textual description for the book's content; and a scientific article can have an abstract that describes its content. In the cold-start setting, these item features are used to compensate for the lack of preference history for the new items. Non-collaborative personalized user modeling techniques determine if a new item will be liked by a user based on how similar it is to the items that the user previously liked [Billsus and Pazzani 1999]. These techniques provide recommendations to each user relying only on the preference history of that user and they do not utilize the past preferences of other users which can potentially be ignoring useful information. State-of-the-art methods that are based on latent factors learn recommendation models by transferring knowledge across users using their preferences [Gantner et al. 2010; Agarwal and Chen 2009; Rendle 2012; Park and Chu 2009]. However, the performance of latent factor methods is compromised when the datasets are very sparse and each item has a limited number of preferences [Ahmed et al. 2013]. At the same time, latent factor methods provide low-rank approximation to the preference matrix, which can potentially lead to loss of information.

In this work we present UFSM: A **U**ser-specific **F**eature-based item-**S**imilarity **M**odel. UFSM learns multiple global similarity functions for estimating the similarity between items based on their feature representations. These functions are informed by the historical preferences of all users, which gives UFSM an advantage over non-collaborative user modeling techniques. UFSM also learns user-specific memberships that specifies for each user the weight by which each global similarity function contributes to the new items' recommendation scores as estimated for that user. UFSM can be viewed as a sparse high-dimensional latent factor model. It is a special type of the recently developed regression-based latent factor models [Agarwal and Chen 2009] and the attribute to feature mapping models [Gantner et al. 2010] with the difference that each user has his own set of mapping functions. These user-specific mapping functions depend on the features of the items that the user has provided preference for, and a common set of feature weights that are shared among all users. This allows the models learned by UFSM to be highly personalized and also leverage information from the other users. The parameters of UFSM are estimated by optimizing either a squared error or a Bayesian personalized ranking (BPR) loss function.

We evaluated the performance of UFSM on a variety of real datasets and compared it against the state-of-the-art latent factor techniques and non-collaborative user modeling techniques. Our results show that UFSM optimized using the BPR loss function outperformed the other methods in terms of recommendation quality. These quality improvements hold for low- and high-dimensional item feature spaces and for datasets with different levels of sparsity.

The rest of the paper is organized as follows. Section 2 defines the notation that will be used throughout the paper. Section 3 discusses the related work. Section 4 describes UFSM and the optimization methods that are used to estimate the model's parameters. Section 5 shows how UFSM can be represented using a factor model and describes why UFSM can better model user preferences. Section 6 includes the evalu-

ation methodology and Section 7 includes the results. Finally Section 8 concludes the work presented in the paper.

2. NOTATIONS AND DEFINITIONS

Throughout the paper, all vectors are row vectors and are represented by bold lower case letters (e.g., $\boldsymbol{f}_i$). Upper case letters are used for representing matrices (e.g., R, P, Q). The i^{th} row of a matrix Q is represented as $\boldsymbol{q}_i$.

The historical preference information is represented by a matrix R. We will refer to R as the "preference matrix". Each row in R corresponds to a user and each column corresponds to an item. The entries of R are binary, reflecting user preferences on items. The preference given by user u for item i is represented by entry $r_{u,i}$ in R. The symbol $\tilde{r}_{u,i}$ represents the score predicted by the model for the actual preference $r_{u,i}$.

Sets are represented with calligraphic letters. The set of users $\mathcal{U}$ has size n_U, and the set of items $\mathcal{I}$ has a size n_I. $\mathcal{R}_u^+$ represents the set of items that user u liked (i.e., $\forall\, i \in \mathcal{R}_u^+, r_{u,i} = 1$). $\mathcal{R}_u^-$ represents the set of items that user u did not like or did not provide feedback for (i.e., $\forall\, i \in \mathcal{R}_u^-, r_{u,i} = 0$).

Each item has a feature vector that represents some intrinsic characteristics of that item. The feature vectors of all items are represented as the matrix F whose rows $\boldsymbol{f}_i$ correspond to the item feature vectors. The total number of item features is referred to as n_F.

When describing latent-factor based models, we assume that the factorization of the matrix R is expressed as $R = PQ^t$, where the P and Q matrices hold the user and item latent factors, respectively. Row $\boldsymbol{p}_u$ in P holds the latent representation of user u, and the length of $\boldsymbol{p}_u$ (that is, the number of latent factors) is h. Similar interpretation is given to each row $\boldsymbol{q}_i$ in $\mathbf{Q}$ with respect to item i.

In the cold-start scenario, we are given a set of new items and their corresponding item features. For each user in the system we need to predict his preference scores for all the new items. The preference scores are predicted using some model and for each user, the new items are sorted using their predicted scores in non increasing order. The n items at the top of the sorted list are recommended to the user. This process is repeated for each user in the system.

3. RELATED WORK

The literature is rich with different classes of methods for solving the cold-start recommendation problem. We focus on two main categories: non-collaborative user modeling techniques that generate personalized recommendations relying only on each user's prior history, and collaborative latent factor techniques that exploit preference information across the different users.

One of the earliest approaches for identifying which of the new items may be relevant to a user is the user-modeling approach developed by Billsus and Pazzani [Billsus and Pazzani 1999]. In this approach, the set of items that a user liked/disliked in the past were used as the training set to learn a model for that user in order to classify new items. The items were represented by some features (e.g., words in the case of articles) and the learning algorithms used these features to build the user models. Billsus and Pazzani experimented with two different algorithms: k-nearest neighbor and naive Bayes. Thought this approach was primarily designed to assign a new item into the "relevant" or "irrelevant" class, it can be easily generalized to compute a relevance score to each item, which can then be used to rank the new items in order to return the n most relevant items for each user.

In subsequent years, other researchers have investigated the use of more advanced user modeling techniques. The work done in [Rodríguez et al. 2001] built personalized user models in the context of classifying news feeds. This work modeled short-term

user needs using the text-based features of the items recently viewed by the user, and modeled long-term user needs using news topics/categories. The work done by [Banos et al. 2006] built more accurate content-based user models for classifying news articles by exploiting topic taxonomies and topic synonyms.

In recent years, latent factor models have emerged as a popular technique for developing collaborative filtering based recommender systems. In the case of the cold start problem, these techniques incorporated the item features in the factorization process. The most general of these techniques was the regression-based latent-factor models (RLFM) [Agarwal and Chen 2009] that can work in different scenarios including item cold-start. RLFM generalized the typical latent factor representation of the preference matrix by adding another step in which the user/item features were transformed to the latent space using linear regression. RLFM also included feature based user/item bias terms. Applying RLFM to the case in which we have item features, no user features and no preference-specific features, the preference $r_{u,i}$ is predicted as

$$\tilde{r}_{u,i} = \alpha + \boldsymbol{b}\boldsymbol{f}_i^t + \boldsymbol{p}_u A^t \boldsymbol{f}_i^t, \tag{1}$$

where α is a scalar bias term, $\boldsymbol{b}$ is a $1 \times n_F$ regression coefficient vector, $\boldsymbol{p}_u$ is u's latent factor representation, $\boldsymbol{f}_i$ is the feature vector of item i and A is a matrix of regression coefficients. The dimensions of A is the number of item features times the number of latent factors. The term $\boldsymbol{b}\boldsymbol{f}_i^t$ represents a global item bias. Item i's representation in the latent space is given by $A^t\boldsymbol{f}_i^t$, which is a transformation of i's feature vector $\boldsymbol{f}_i$ into the latent space. The term $\boldsymbol{p}_u A^t \boldsymbol{f}_i^t$ is the main term that estimates the user preference as the dot-product between u and i's representations in the latent space. An extension of the RLFM model was proposed in [Zhang et al. 2011] where more flexible regression models were applied. However, the additional improvements were limited.

A method to learn attribute to feature mapping (AFM) was proposed in [Gantner et al. 2010]. Item cold start was handled by first learning a factorization of the preference matrix into user and item latent factors $R = PQ^t$. Next, a mapping function was learned to transform the feature representation of items into their latent space representation. In matrix notation, this is expressed as $R = PQ^t = PAF^t$, where each row in A corresponds to a latent factor and each column corresponds to an item feature. Accordingly, the preference $r_{u,i}$ for user u and item i is now predicted as

$$\tilde{r}_{u,i} = \boldsymbol{p}_u A \boldsymbol{f}_i^t. \tag{2}$$

Bayesian Personalized Ranking (BPR) Matrix Factorization [Rendle et al. 2009] was used to learn $R = PQ^t$. Also a BPR loss function was used in learning the mapping function A. Results obtained on the MovieLens dataset showed that AFM works well with low-dimensional item feature spaces, whereas its predictive performance can degrade when high-dimensional feature spaces are used.

Comparing RLFM with AFM, the differences are: (i) RLFM has feature-based bias terms, (ii) RLFM learns all the model parameters simultaneously while AFM learns the user factors first then it learns the feature-to-latent space mapping function in a separate step, and (iii) AFM optimizes a BPR loss function with stochastic gradient descent to learn model parameters whereas RLFM uses a stochastic process model with maximum likelihood estimation to learn model parameters.

The CTR method proposed in [Wang and Blei 2011] had a special case that works with new items. As mentioned in [Wang and Blei 2011], this method would be equivalent to RLFM [Agarwal and Chen 2009] using item features and no user features. Other techniques for addressing the item cold-start problem include probabilistic modeling techniques [Zhang and Koren 2007; Zhang and Zhang 2010; Wang and Blei 2011],

and techniques that utilize item features to densify the preference matrix in order to improve the recommendation quality [Katz et al. 2011; Hu and Pu 2011].

4. UFSM: USER-SPECIFIC FEATURE-BASED SIMILARITY MODELS

The algorithms for solving the item cold-start problem that we developed are inspired by the user modeling technique of [Billsus and Pazzani 1999] that relied on k nearest neighbor classification (Section 3). In that scheme, when the training data for each user contains only the relevant items and the value of k is sufficiently large to include all of the user's relevant items, the relevance/preference score of a new item i for user u is given by

$$\tilde{r}_{u,i} = \sum_{j \in \mathcal{R}_u^+} \text{sim}(i,j) = \sum_{j \in \mathcal{R}_u^+} \boldsymbol{f}_i \boldsymbol{f}_j^t = \boldsymbol{f}_i \sum_{j \in \mathcal{R}_u^+} \boldsymbol{f}_j^t, \tag{3}$$

where $\text{sim}(i,j)$ is the cosine similarity between items i and j given that the feature vectors of both items are of unit length[1]. This model suffers from three major drawbacks. First, the similarity function is predefined and it does not utilize the historical preferences in order to estimate a similarity function that better predicts the observed preferences. Second, all users use the same similarity function, which fails to account for the fact that different users may like items for different reasons that are better modeled via different similarity functions. Third, the preference score that is computed for a new item with respect to user u relies entirely on the set of items previously liked by u (i.e., $\sum_{j \in \mathcal{R}_u^+} \boldsymbol{f}_j^t$), and as such it does not use information from any other users. However, under the assumption that there may be sets of users with similar likes/dislikes, being able to incorporate information from different users can potentially lead to better results.

UFSM is designed to overcome all of these drawbacks. First, each user has his own similarity function, which leads to a higher degree of personalization. Second, these user-specific similarity functions are derived as a linear combination of a fixed number of user-independent similarity functions, referred to as *global similarity* functions. These global similarity functions are the same for all the users but they are combined in a way that is specific for each user, resulting in distinct user-specific similarity functions. Third, the global similarity functions and their user-specific combination weights are estimated by taking into account the historical preferences of all users, allowing them to leverage information across the entire dataset.

In UFSM, the preference score of a new item i for user u is given by

$$\tilde{r}_{u,i} = \sum_{j \in \mathcal{R}_u^+} \text{sim}_u(i,j),$$

where $\text{sim}_u(i,j)$ is the user-specific similarity function given by

$$\text{sim}_u(i,j) = \sum_{d=1}^{l} m_{u,d}\ \text{gsim}_d(i,j),$$

where $\text{gsim}_d(.)$ is the d^{th} global similarity function, l is the number of global similarity functions, and $m_{u,d}$ is a scalar that determines how much the d^{th} global similarity function contributes to u's similarity function. We will refer by $\boldsymbol{m}_u$ to u's *membership* vector.

[1]Also note that an expression similar to Equation 3 can be derived from the item-based collaborative filtering method [Sarwar et al. 2001] where the similarity between two items does not depend on the set of users that co-preferred items i and j, but their feature vectors $\boldsymbol{f}_i$ and $\boldsymbol{f}_j$.

The similarity between two items i and j under the d^{th} global similarity function $\text{gsim}_d(.)$ is estimated as

$$\text{gsim}_d(i,j) = \boldsymbol{w}_d(\boldsymbol{f}_i \odot \boldsymbol{f}_j)^t,$$

where $\odot$ is the element-wise Hadamard product operator, $\boldsymbol{f}_i$ and $\boldsymbol{f}_j$ are the feature vectors of items i and j, respectively, and $\boldsymbol{w}_d$ is a vector of length n_F with each entry $w_{d,c}$ holding the weight of feature c under the global similarity function $\text{gsim}_d(.)$. This weight reflects the contribution of feature c in the item-item similarity estimated under $\text{gsim}_d(.)$. Note that $\boldsymbol{w}_d$ is nothing more than a linear model on the feature vector resulting by the Hadamard product.

Putting it all together, the estimated preference score $\tilde{r}_{u,i}$ of user u for item i is given by

$$\begin{aligned}
\tilde{r}_{u,i} &= \sum_{j \in \mathcal{R}_u^+} \text{sim}_u(i,j) \\
&= \sum_{j \in \mathcal{R}_u^+} \sum_{d=1}^{l} m_{u,d}\ \text{gsim}_d(i,j) \\
&= \sum_{d=1}^{l} m_{u,d} \left(\sum_{j \in \mathcal{R}_u^+} \boldsymbol{w}_d(\boldsymbol{f}_i \odot \boldsymbol{f}_j)^t \right) \\
&= \sum_{d=1}^{l} m_{u,d} \sum_{c=1}^{n_F} f_{i,c}(w_{d,c} \sum_{j \in \mathcal{R}_u^+} f_{j,c}).
\end{aligned} \tag{4}$$

The idea of having multiple weighted similarity measures was proposed in [Yih 2009] in the context of relevant advertisement retrieval. In this work, the document-document similarity was estimated as a weighted sum over multiple predefined similarity measures. These similarity measures compute the cosine similarity between the two documents using different feature representations (e.g., term presence and term frequency). The weight by which each similarity measure contributes to the final similarity value was learned using labeled data. This is different from UFSM because the similarity measures are predefined and the contribution weights are global for all the documents.

4.1. Estimation of UFSM

UFSM learns a model $\Theta = [M, \boldsymbol{w}_1, \ldots, \boldsymbol{w}_l]$, where $\boldsymbol{w}_1, \ldots, \boldsymbol{w}_l$ are the parameters of the global similarity functions and M is a $n_U \times l$ matrix of user memberships. The inputs to the learning process are: (i) the preference matrix R, (ii) the item-feature matrix F, and (iii) the number of global similarity functions l that we want to learn.

We developed two types of UFSM models that are based on two different loss functions: UFSM$_{rmse}$ and UFSM$_{bpr}$. UFSM$_{rmse}$ uses the squared error loss function

$$\mathcal{L}_{rmse}(\Theta) = \sum_{u \in U} \sum_{i \in I} (r_{u,i} - \tilde{r}_{u,i}(\Theta))^2, \tag{5}$$

and UFSM$_{bpr}$ uses Bayesian personalized ranking loss function [Rendle et al. 2009]

$$\mathcal{L}_{bpr}(\Theta) = -\sum_{u \in U} \sum_{\substack{i \in \mathcal{R}_u^+, \\ j \in \mathcal{R}_u^-}} \ln \sigma(\tilde{r}_{u,i}(\Theta) - \tilde{r}_{u,j}(\Theta)). \tag{6}$$

In both equations, $r_{u,i}$ and $\tilde{r}_{u,i}$ are the observed and estimated values of user u's preference for item i whereas in Equation 6, $\sigma(x)$ is the sigmoid function. The RMSE loss function tries to learn a model that estimates the like and dislike preferences as precisely 1 and 0, respectively. The BPR loss function, on the other hand, tries to learn item preference scores such that the items that a user liked have higher preference scores than the ones she did not like, regardless of the actual item preference scores. Since the predicted preference scores are used to rank the items in order to select the highest scoring n items, the BPR loss functions better models the problem requirements and in general lead to better performance [Gantner et al. 2010; Rendle et al. 2009].

The value of $\tilde{r}_{u,i}$ is estimated as

$$\tilde{r}_{u,i} = \sum_{\substack{j \in \mathcal{R}_u^+, \\ j \neq i}} \text{sim}_u(i,j), \tag{7}$$

which is identical to Equation 4 except that item i is excluded from the summation. This is done to ensure that the variable being estimated (the dependent variable) is not used during the estimation as an independent variable as well. We refer to this as the *Estimation Constraint*.

4.2. Optimization of UFSM

The model parameters $\Theta = [M, \boldsymbol{w}_1, \ldots, \boldsymbol{w}_l]$ are estimated via an optimization process of the form:

$$\begin{aligned} \underset{\Theta=[M,\boldsymbol{w}_1,\ldots,\boldsymbol{w}_l]}{\text{minimize}} \quad & \mathcal{L}(\Theta) + Reg(\Theta), s.t. \\ & \boldsymbol{w}_d \boldsymbol{w}_d^t = 1, \forall\, d = 1, \ldots, l, \\ & \boldsymbol{w}_d \boldsymbol{w}_{d'}^t = 0, \forall\, d \neq d', \end{aligned} \tag{8}$$

where $\mathcal{L}(\Theta)$ represents the loss function and $Reg(\Theta)$ represents a regularization term that controls the model's complexity. The two constraints enforce orthogonality between the weight vectors $\boldsymbol{w}_d$'s associated with the global similarity functions. This is done in order to ensure that the learned models are different[2]. These constraints are softly applied by incorporating them into the regularization term $Reg(\Theta)$ using a quadratic penalty function. Accordingly, the regularization function $Reg(\Theta)$ becomes:

$$Reg(\Theta) = \lambda ||M||_F^2 + \mu_1 \sum_{d=1}^{l} (\boldsymbol{w}_d \boldsymbol{w}_d^t - 1)^2 + \mu_2 \sum_{d=1}^{l} \sum_{d'=d+1}^{l} (\boldsymbol{w}_d \boldsymbol{w}_{d'}^t)^2. \tag{9}$$

The term $\lambda ||M||_F^2$ controls the complexity of the user memberships matrix M and thus is used to avoid overfitting. The term $(\mu_1 \sum_{d=1}^{l} (\boldsymbol{w}_d \boldsymbol{w}_d^t - 1)^2)$ is used to make the vectors $\boldsymbol{w}_d$ be of unit length. The term $(\mu_2 \sum_{d=1}^{l} \sum_{d'=d+1}^{l} \boldsymbol{w}_d \boldsymbol{w}_{d'}^t)$ enforces orthogonality among the $\boldsymbol{w}_d$ vectors.

For both UFSM$_{rmse}$ and UFSM$_{bpr}$, stochastic gradient descent (SGD) [Bottou 1998] was used to learn the model parameters Θ. Choosing SGD was due to its efficiency.

[2]Our initial experiments showed that without these orthogonality constraints the resulting $\boldsymbol{w}_d$ vectors of the global similarity models $\text{gsim}_d(.)$ were often nearly parallel and as such redundant.

For UFSM$_{rmse}$, the update steps for M, $\boldsymbol{w}_1, \ldots, \boldsymbol{w}_l$ based on a preference $r_{u,i}$ become

$$w_{d,c} = w_{d,c} - \alpha_1 \Big(\frac{\partial}{\partial w_{d,c}} (r_{u,i} - \tilde{r}_{u,i})^2 + \mu_1 \frac{\partial}{\partial w_{d,c}} (\boldsymbol{w}_d \boldsymbol{w}_d^t - 1)^2 + \mu_2 \frac{\partial}{\partial w_{d,c}} \sum_{d' \neq d} (\boldsymbol{w}_d \boldsymbol{w}_{d'}^t)^2 \Big), \tag{10}$$

and

$$m_{u,d} = m_{u,d} - \alpha_2 \Big(\frac{\partial}{\partial m_{u,d}} (r_{u,i} - \tilde{r}_{u,i})^2 + \lambda \, m_{u,d} \Big). \tag{11}$$

The gradients in Equations 10 and 11 are estimated as:

$$\frac{\partial}{\partial w_{d,c}} (r_{u,i} - \tilde{r}_{u,i})^2 = 2 m_{u,d} (\tilde{r}_{u,i} - r_{u,i}) \sum_{\substack{q \in \mathcal{R}_u^+, \\ q \neq i}} f_{i,c} f_{q,c}, \tag{12a}$$

$$\frac{\partial}{\partial m_{u,d}} (r_{u,i} - \tilde{r}_{u,i})^2 = 2 (\tilde{r}_{u,i} - r_{u,i}) \sum_{\substack{q \in \mathcal{R}_u^+, \\ q \neq i}} \sum_{c=1}^{n_F} f_{i,c} f_{q,c} w_{d,c}, \tag{12b}$$

$$\frac{\partial}{\partial w_{d,c}} \mu_1 (\boldsymbol{w}_d \boldsymbol{w}_d^t - 1)^2 = 4 \mu_2 w_{d,c} (\boldsymbol{w}_d \boldsymbol{w}_d^t - 1)), \tag{12c}$$

$$\frac{\partial}{\partial w_{d,c}} \mu_2 \sum_{d' \neq d} (\boldsymbol{w}_d \boldsymbol{w}_{d'}^t)^2 = 2 \mu_1 \sum_{d' \neq d} w_{d',c} (\boldsymbol{w}_d \boldsymbol{w}_{d'}^t). \tag{12d}$$

For UFSM$_{bpr}$, the update steps for $M, \boldsymbol{w}_1, \ldots, \boldsymbol{w}_l$ based on a triplet (u, i, j) and its corresponding estimated relative rank $\tilde{r}_{u,ij}$ become

$$w_{d,c} = w_{d,c} + \alpha_1 \Big(\frac{e^{-\tilde{r}_{u,ij}}}{1 + e^{-\tilde{r}_{u,ij}}} \times \frac{\partial}{\partial w_{d,c}} \tilde{r}_{u,ij} + \mu_1 \frac{\partial}{\partial w_{d,c}} (\boldsymbol{w}_d \boldsymbol{w}_d^t - 1)^2 + \mu_2 \frac{\partial}{\partial w_{d,c}} \sum_{d' \neq d} (\boldsymbol{w}_d \boldsymbol{w}_{d'}^t)^2 \Big), \tag{13}$$

and

$$m_{u,d} = m_{u,d} + \alpha_2 \Big(\frac{e^{-\tilde{r}_{u,ij}}}{1 + e^{-\tilde{r}_{u,ij}}} \times \frac{\partial}{\partial m_{u,d}} \tilde{r}_{u,ij} - \lambda \, m_{u,d} \Big). \tag{14}$$

The relative rank $\tilde{r}_{u,ij}(\Theta)$ is estimated using Equation 4 as

$$\tilde{r}_{u,ij} = \sum_{d=1}^{l} m_{u,d} \left(\sum_{\substack{q \in \mathcal{R}_u^+, \\ q \neq i}} \sum_{c=1}^{n_F} (f_{i,c} - f_{j,c}) f_{q,c} w_{d,c} \right). \tag{15}$$

The gradients $\frac{\partial}{\partial w_{d,c}}\tilde{r}_{u,ij}$ and $\frac{\partial}{\partial m_{u,d}}\tilde{r}_{u,ij}$ are estimated using Equation 15 as

$$\frac{\partial}{\partial w_{d,c}}\tilde{r}_{u,ij} = m_{u,d} \sum_{\substack{q \in \mathcal{R}_u^+, \\ q \neq i}} (f_{i,c} - f_{j,c}) f_{q,c}, \tag{16a}$$

$$\frac{\partial}{\partial m_{u,d}}\tilde{r}_{u,ij} = \sum_{\substack{q \in \mathcal{R}_u^+, \\ q \neq i}} \sum_{c=1}^{n_F} (f_{i,c} - f_{j,c}) f_{q,c} w_{d,c}. \tag{16b}$$

The gradients $\frac{\partial}{\partial w_{d,c}}(\boldsymbol{w}_d \boldsymbol{w}_d^t - 1)^2$ and $\frac{\partial}{\partial w_{d,c}} \sum_{d' \neq d} (\boldsymbol{w}_d \boldsymbol{w}_{d'}^t)^2$ in Equation 13 are estimated according to Equations 12c and 12d, respectively.

The high-level structures of the algorithms used to estimate the UFSM$_{rmse}$ and the UFSM$_{bpr}$ models are shown in Algorithms 1 and 2, respectively. In estimating the UFSM$_{rmse}$ model, we sample a user u and an item i such that u liked i, update the model parameters accordingly, then we sample for the same user an item j that u did not like and update the model parameters accordingly. In estimating the UFSM$_{bpr}$ model, we sample a user u and a pair of items (i, j) such that u liked i and did not like j and the model parameters are updated based on this triplet (u, i, j). Note that in both algorithms, the optimization process continues until convergence is detected. This part is described in details in the Section 6.4 as it is related to the evaluation metrics that are used to measure the performance of the different methods.

The factorization machines library [Rendle 2012] can be used to learn a model that is similar to UFSM as discussed in Section 5.2.

4.3. Time analysis for learning the UFSM models

The time taken to estimate the parameters of UFSM depends on the number of sampled instances per learning iteration n_R, the average number of preferences per user n_R^u, and the average number of features per item n_F^i. With each learning iteration, we sample a number of training instances/pairs n_R that is equal to the total number of preferences. With each sample, the number of computations made are of order $O(n_R^u \times n_F^i)$. Therefore, each learning iteration takes time of order $O(n_R \times n_R^u \times n_F^i)$.

5. UFSM VS. LATENT FACTOR METHODS

We show in this section how UFSM relates to the different latent factor methods and what are the advantages that UFSM possess over these methods.

5.1. Representing UFSM as an AFM Model

In this section we show that the UFSM model can be considered as a sparse high-dimensional latent factor model that is expressed as a special case of the AFM model described in Section 3.

Given a UFSM model with l global similarity functions and parameters $\Theta = [M, \boldsymbol{w}_1, \ldots, \boldsymbol{w}_l]$, it is possible to represent it as an AFM model ($R = PAF^t$) by constructing the P and A matrices as follows.

Let the number of latent factors for AFM be $h = l \times n_U$. Let P be an $n_U \times h$ matrix of user latent factors and for each user u, the latent space representation $\boldsymbol{p}_u$ in P is set as follows:

$p_{u,0}, p_{u,1}, \ldots, p_{u,(l(u-1)-1)} = 0,$
$p_{u,(l(u-1))}, \ldots, p_{u,(lu)} = m_{u,0}, \ldots, m_{u,l},$
$p_{u,(lu+1)}, \ldots, p_{u,h} = 0.$

Let A be an $h \times n_F$ matrix such that the rows of A are divided into n_U chunks of size

Algorithm 1 UFSM$_{rmse}$-Learn

procedure UFSM$_{rmse}$_LEARN
  $\lambda \leftarrow$ M regularization weight
  $\mu_1, \mu_2 \leftarrow$ W regularization weights
  $\alpha_1, \alpha_2 \leftarrow$ W and M learning rates
  Initialize $\Theta = [M, \boldsymbol{w}_1, \ldots, \boldsymbol{w}_l]$ randomly

  while not converged **do**
    for each user u **do**
      sample an item i s.t. $i \in \mathcal{R}_u^+$

      estimate $\frac{\partial}{\partial w_{d,c}}(r_{u,i} - \tilde{r}_{u,i})^2$ using (12a)
      estimate $\frac{\partial}{\partial m_{u,d}}(r_{u,i} - \tilde{r}_{u,i})^2$ using (12b)
      estimate $\frac{\partial}{\partial w_{d,c}}(\boldsymbol{w}_d \boldsymbol{w}_d^t - 1)^2$ using (12c)
      estimate $\frac{\partial}{\partial w_{d,c}} \sum_{d' \neq d}(\boldsymbol{w}_d \boldsymbol{w}_{d'}^t)^2$ using (12d)

      estimate $\tilde{r}_{u,i}$ using (7)
      update $m_{u,d}\ \forall\, d$ using (11)
      update $w_{d,c}\ \forall\, d,\ \forall c\ s.t.\ f_{i,c}\ \neq 0$

      sample an item j s.t. $j \in \mathcal{R}_u^-$ and repeat steps 11 to 18

    end for
  end while

  return $\Theta = [M, \boldsymbol{w}_1, \ldots, \boldsymbol{w}_l]$
end procedure

l each. The u^{th} chunk in A corresponds to user u in R. The entry $a_{lu+d,c}$ in chunk u is set to $w_{d,c} \sum_{j \in \mathcal{R}_u^+} f_{j,c}$. Using this construction, the preference score $\tilde{r}_{u,i}$ under AFM (Equation 2) is

$$\tilde{r}_{u,i} = \boldsymbol{p}_u A \boldsymbol{f}_i^t = \sum_{d=1}^{n_U \times l} p_{u,d} \sum_{c=1}^{n_F} f_{i,c}\, a_{lu+d,c} = \sum_{d=1}^{l} m_{u,d} \sum_{c=1}^{n_F} f_{i,c} (w_{d,c} \sum_{j \in \mathcal{R}_u^+} f_{j,c}), \tag{17}$$

which is identical to Equation 4 of UFSM. This shows that UFSM can be represented using AFM such that each user has his own set of mapping functions that are derived from his own item preferences. Accordingly, representing UFSM using AFM requires a latent space with a large number of dimensions. In particular, the l global similarity functions in UFSM are represented using $(l \times n_U)$ latent factors in AFM. Looking at the nature of these latent factors, we see that each user u has his own set of l latent factors that are derived from the features of the items that u previously liked. This direct incorporation of user preferences reduces the loss of information due to low rank approximation, which occurs with latent factor models.

Although a UFSM model can be represented using AFM, the reverse is not true. That is, there is no UFSM equivalent for every AFM model. Moreover, AFM cannot be used to find the model that corresponds to UFSM without the introduction of additional constraints to enforce the special structure required by UFSM.

Algorithm 2 UFSM$_{bpr}$-Learn

procedure UFSM$_{bpr}$_LEARN
  $\lambda \leftarrow$ M regularization weight
  $\mu_1, \mu_2 \leftarrow$ W regularization weights
  $\alpha_1, \alpha_2 \leftarrow$ W and M learning rates
  Initialize $\Theta = [M, \boldsymbol{w}_1, \ldots, \boldsymbol{w}_l]$ randomly

  while not converged **do**
    for each user u **do**
      sample a pair (i, j) s.t. $i \in \mathcal{R}_u^+$, $j \in \mathcal{R}_u^-$

      estimate $\frac{\partial}{\partial w_{d,c}} \tilde{r}_{u,ij}$ using (16a)
      estimate $\frac{\partial}{\partial m_{u,d}} \tilde{r}_{u,ij}$ using (16b)
      estimate $\frac{\partial}{\partial w_{d,c}} (\boldsymbol{w}_d \boldsymbol{w}_d^t - 1)^2$ using (12c)
      estimate $\frac{\partial}{\partial w_{d,c}} \sum_{d' \neq d} (\boldsymbol{w}_d \boldsymbol{w}_{d'}^t)^2$ using (12d)

      estimate $\tilde{r}_{u,ij}$ using (15)
      update $m_{u,d}\ \forall\, d$ using (14)
      update $w_{d,c}\ \forall\, d,\ \forall\, c\ s.t.\ f_{i,c}\ or\ f_{j,c} \neq 0$ using (13)
    end for
  end while

  return $\Theta = [M, \boldsymbol{w}_1, \ldots, \boldsymbol{w}_l]$
end procedure

5.2. Using the Factorization Machine Model to Learn UFSM

In this section we show how the factorization machine framework (FM) proposed in [Rendle 2012] and implemented within the LibFM library can be used to generate a model that is similar to UFSM. FM incorporates features in learning factorization models while accounting for dependencies among the different features. It can also learn feature-based bias terms. In FM, each training instance is represented by $(r_{u,i}, \boldsymbol{x}_{ui})$, where $r_{u,i}$ is an entry in the preference matrix R and $\boldsymbol{x}_{ui}$ is a vector of input features for $r_{u,i}$. Ignoring the bias terms for simplicity, the recommendation score under FM is estimated as

$$\tilde{r}_{u,i} = \sum_{c=1}^{k} \sum_{c'=c+1}^{k} x_{ui,c}\, x_{ui,c'}\, \boldsymbol{v}_c \boldsymbol{v}_{c'}^t, \tag{18}$$

where k is the length of the feature vectors, $x_{ui,c}$ is the value of feature c in feature vector $\boldsymbol{x}_{ui}$, and $\boldsymbol{v}_c$ is the vector of latent factors representing feature c.

FM can be used to generate a model similar to UFSM as follows. For each training instance $(r_{u,i}, \boldsymbol{x}_{ui})$, the vector $\boldsymbol{x}_{ui}$ has length $n_U + n_F$ (i.e., $k = n_U + n_F$). The first n_U features of $\boldsymbol{x}_{ui}$ are binary with only $x_{ui,u} = 1$, and the other features set to 0 (that is $x_{ui,c} = 0,\ \ 1 \leq c \leq n_U,\ \ c \neq u$). The remaining n_F features are set as $x_{ui,n_U+c} = f_{i,c} \sum_{j \in \mathcal{R}_u^+} f_{j,c}$, $1 \leq c \leq n_F$.
Then according to Equation 18, the recommendation score $\tilde{r}_{u,i}$ is estimated as

$$\tilde{r}_{u,i} = \sum_{c=n_U+1}^{n_U+n_F} x_{ui,u}\, x_{ui,c}\, \boldsymbol{v}_u \boldsymbol{v}_c^t \ + \sum_{c=n_U+1}^{n_U+n_F} \sum_{c'=c+1}^{n_U+n_F} x_{ui,c}\, x_{ui,c'}\, \boldsymbol{v}_c \boldsymbol{v}_{c'}^t. \tag{19}$$

The first term results from having the first n_U features set to all 0's except for entry $x_{ui,u}$ which was set to 1. The second term shows interactions between the rest of the features. Substituting the feature values described above into Equation 19 and rearranging the sums, we get

$$\begin{aligned}\tilde{r}_{u,i} = &\sum_{d=1}^{l} v_{u,d} \sum_{c=1}^{n_F} f_{i,c} \left(v_{(c+n_U),d} \sum_{j \in \mathcal{R}_u^+} f_{j,c} \right) \\ &+ \sum_{c=1}^{n_F} \sum_{c'=c+1}^{n_F} f_{i,c} f_{i,c'} \left(\sum_{j \in \mathcal{R}_u^+} f_{j,c} \right) \left(\sum_{j \in \mathcal{R}_u^+} f_{j,c'} \right) \sum_{d=1}^{l} v_{(c+n_U),d} v_{(c'+n_U),d}.\end{aligned} \tag{20}$$

The first term in Equation 20 is identical to Equation 4 of UFSM with $v_{u,d}$ corresponding to $m_{u,d}$ and $v_{(c+n_U),d}$ corresponding to $w_{d,c}$. The second term in Equation 20 accounts for dependencies among the different item features. In the UFSM setting, this term accounts for dependencies among the feature pairs that appear in the history of different users. That is, it accounts for dependencies among the feature pairs that the users find interesting. It is clear that FM subsumes UFSM since the feature vectors can be constructed in any arbitrary way in order to fulfill any model-specific assumptions.

6. EXPERIMENTAL EVALUATION

6.1. Datasets

We used five datasets (CiteULike, Amazon Books, Book Crossing, MovieLens-IMDB and MovieLens-HetRec) to evaluate the performance of UFSM.

CiteULike (CUL)[3] is an online service that allows researchers to add scientific articles to their libraries. For each user, the articles that were added in his library are considered as preferred articles (i.e., their preference values in matrix R are set to 1and there are no explicit 0 preferences). We have collected the articles' titles and abstracts and used them as the item's content. Amazon Books (ABB) is a dataset collected from the best selling books at Amazon and their ratings. The ratings were binarized by setting all ratings above 3 to 1 and all ratings below or equal to 3 to 0. Each book had one or two paragraphs of textual description that was used as the item's content. Book Crossing (BX) is a dataset extracted from the Book Crossing dataset [Ziegler et al. 2005] such that each user/book provided/received at least four ratings. We have collected the book descriptions form Amazon using the ISBN and used them as the items' features. The rating scale is 1-10. The ratings were binarized by setting all ratings above 6 to 1 and all ratings below or equal to 6 to 0. MovieLens-IMDB (ML-IMDB) is a dataset extracted from the IMDB and the MovieLens-1M datasets[4] by mapping the MovieLens and IMDB movie IDs and collecting the movies that have plots and keywords. The ratings were binarized the same way as we did with the ABB dataset. The movies plots and keywords were used as the item's content. MovieLens-HetRec (ML-HR(genre)) is the dataset described in [Cantador et al. 2011]. The ratings were binarized the same way as we did with the ABB dataset. The movie genres were used as the item's content.

For the CUL, BX, ABB and ML-IMDB datasets, the words that appear in the item descriptions were collected, stop words were removed and the remaining words were stemmed to generate the terms that were used as the item features. All words that

[3]http://www.citeulike.org/

[4]http://www.movielens.org, http://www.imdb.com

appear in less than 20 items and all words that appear in more than 20% of the items were omitted. The remaining words were represented with TF-IDF scores. The item feature matrix F was normalized row-wise in all the datasets except for the ML-HR(genre), which has 20 features corresponding to movie genres.

Various statistics about these datasets are shown in Table I. Most of these datasets contain items that have high-dimensional feature spaces. The only exception is ML-HR(genre) whose items are represented in a 20-dimensional feature space. Also comparing the densities of the different datasets we can see that the two MovieLens datasets have significantly higher density than that of the other three datasets.

Table I. Statistics for the 5 datasets used for testing

Dataset	# users	# items	# features	# preferences	# prefs/user	# prefs/item	density
CUL	3,272	21,508	6,359	180,622	55.2	8.4	0.13%
ABB	13,097	11,077	5,766	175,612	13.4	15.9	0.12%
BX	17,219	36,546	8,946	574,127	33.3	15.7	0.09%
ML-IMDB	2,113	8,645	8,744	739,973	350.2	85.6	4.05%
ML-HR(genre)	2,113	10,109	20	855,598	404.9	84.6	4.01%

6.2. Comparison with Other Methods

We compare UFSM against non-collaborative user modeling techniques and collaborative latent factor based techniques.

(1) **Non-Collaborative User Modeling Techniques**

The two techniques described here are inspired by the ideas in [Billsus and Pazzani 1999].

— **Simple Cosine-Similarity (CoSim):** A personalized user modeling technique that was described at the beginning of Section 4. The preference score $r_{u,i}$ of user u for item i is estimated using Equation 3. For each user u, we estimate u's preference scores over all the test items, then the estimated preference scores are sorted and the top-n items are selected. The evaluation metrics are computed over this list of recommended items.

— **Personalized Feature Weighting (PFW):** A non-collaborative technique that learns user models independently. A feature weighting vector $\boldsymbol{w}_u$ of length n_F is estimated for each user u to reflect the importance of the different item features for each user. The preference score $r_{u,i}$ of user u for item i is estimated as $\tilde{r}_{u,i} = \sum_{j \in \mathcal{R}_u^+} \boldsymbol{w}_u(\boldsymbol{f}_i^t \odot \boldsymbol{f}_j^t)$. The problem is formulated as a personalized ranking problem and a BPR loss function is used to learn each of the user models $\boldsymbol{w}_u,\ \forall u \in \mathcal{U}$. Using a BPR-loss achieved better results compared to using an RMSE-loss function. The optimization problem used to estimate each $\boldsymbol{w}_u$ takes the form

$$\underset{\boldsymbol{w}_u}{\text{minimize}} \quad - \sum_{\substack{i \in \mathcal{R}_u^+, \\ j \in \mathcal{R}_u^-}} \ln \sigma(\tilde{r}_{u,i}(\boldsymbol{w}_u) - \tilde{r}_{u,j}(\boldsymbol{w}_u)) + \mu ||\boldsymbol{w}_u||_2^2,$$

where the second term is used for regularization. The number of sampled pairs that were used to learn a user model w_u were eight times the size of $\mathcal{R}_u$.

(2) **Collaborative Latent Factor-based Techniques**

— **RLFMI:** The Regression-based Latent Factor Modeling technique that was described in Section 3. We experimented with both the original RLFM technique described in [Agarwal and Chen 2009] and the version that is implemented in LibFM that accounts for inter-feature interactions (referred to as RLFMI). RLFMI consistently gave slightly better results over all the datasets and for this reason we only reported the results for RLFMI. We used the Factorization Machine library LibFM [Rendle 2012] with SGD learning to obtain the RLFMI results.
— **AFM$_{bpr}$:** The attribute to feature mapping technique [Gantner et al. 2010] that was described in Section 3, which we implemented ourselves. The subscript bpr describes the BPR loss function that was used to learn the model parameters.
— **AFMI$_{rmse}$:** The AFM technique as implemented in LibFM. AFMI$_{rmse}$ accounts for inter-feature interactions among the item features. The reported results were obtained using LibFM with SGD optimizer and RMSE loss function.

6.3. Evaluation Methodology and Metrics

We evaluated the performance of the different methods using the following procedure. For each dataset we split its corresponding user-item preference matrix R into two matrices R_{train} and R_{test}. The R_{test} matrix contains a randomly selected 10% of the columns of R and the R_{train} matrix contains the remaining 90% of the columns. Since each column corresponds to an item, the set of items in R_{test} are disjoint from those in R_{train}. The information in R_{train} was used to train each of the models, which were then used to predict for each user the preference scores for the items in R_{test}. These scores were then used to sort the items in descending order and the first n items were returned as the top-n recommendations for each user. The evaluation metrics (described next) are computed using the top-n recommendations that are made for each user. Note that after making the train-test split, some users end up having no items in the test set. For the CUL, BX and ABB datasets, 13%, 33%, and 40% of the users had no items in the test set, respectively. For this reason we only evaluated the performance of the different methods on the users that had at least one item in their test set. For the two MovieLens datasets, all users had items in the test set. This split-train-evaluate procedure is repeated three times for each dataset, and the obtained values for the evaluation metrics are averaged over the three runs and reported in the results.

We used two metrics to assess the performance of the various methods. The first is the Recall at n metric (Rec@n). Given the list of the top-n recommended items for user u, Rec@n measures how many of the items liked by u appeared in that list. Rec@n is computed for each user $u \in \mathcal{U}$ and then averaged over all users. The second metric is Discounted Cumulative Gain at n (DCG@n). Given the list of the top-n recommended items for user u, the DCG@n of user u is defined as

$$DCG@n = \text{imp}_1 + \sum_{p=2}^{n} \frac{\text{imp}_p}{\log_2(p)},$$

where the importance score imp_p of the item with rank p in the top-n list is

$$\text{imp}_p = \begin{cases} 1/n, & \text{if item at rank } p \in R^+_{u,test} \\ 0, & \text{if item at rank } p \notin R^+_{u,test}. \end{cases}$$

Just like Rec@n, the DCG@n is computed for each user and then averaged over all the users. The main difference between Rec@n and DCG@n is that DCG@n is sensitive to the rank of the items in the top-n list. Note that we also computed the Precision at n

metric (Prec@n); however, the ranking of the methods under Prec@n was the same as their ranking under Rec@n and for this reason we only report the Rec@n values.

6.4. Model Training and Selection

In estimating UFSM's parameters, the train set R_{train} was furthur divided into training and validation sets with weights 90% and 10%, respectively. The model was estimated using the training set and the validation set was used to detect convergence as follows. After each major SGD iteration of Algorithms 1 and 2, the Rec@n was estimated over the validation set and the model was saved if it achieved a better Rec@n value than that obtained in any of the earlier iterations. The learning process ends when no improvement in Rec@n is observed for ten major iterations. At the end of the learning process the model that has achieved the best Rec@n on the validation set was returned.

For estimating UFSM$_{bpr}$ and UFSM$_{rmse}$, each major SGD iteration involves drawing a number of samples that is equal to the number of entries in R. According to Algorithms 1 and 2, each sample involves a user, an item that he liked and/or an item that he did not like. The CUL dataset contained only preferences about articles that each user has liked. In this case the disliked articles are sampled from among the unknown entries in R (i.e., the articles that the user did not add to his library). All the other datasets contain both like and dislike user preferences. For these datasets, the dislike preferences are sampled from among both the real dislikes and the unknown entries. We tried regularization parameters in the range $[1e-5, 1]$ and a number of global similarity functions $l = 1, 2, \ldots, 7$ for estimating UFSM$_{bpr}$ and UFSM$_{rmse}$.

For estimating AFMI$_{rmse}$ and RLFMI, LibFM was provided with the train and validation sets and the model that achieved the best performance on the validation set was returned. The training set must contain both 0's and 1's. Since the CUL dataset does not contain 0's, we sampled 0's from the unknown values. We performed a series of experiments in which we sampled a number of 0's equal to $1\times$, $5\times$, $10\times$ and $15\times$ the number of 1's in R. The overall performance of these models was very similar. For this reason we only reported results for the $1\times$ sampling rate. For estimating AFMI$_{rmse}$, AFM$_{bpr}$ and RLFMI, we tried regularization parameters in the range $[1e-5, 1]$, and a number of latent factors $h = 10, 20, 30, \ldots, 150$. For the ML-HR(genre) dataset that has only 20 item features, we tried a number of latent factors $h = 1, 2, 3, \ldots, 20$.

For estimating the PFW models, for each user u we sampled triplets (u, i, j), where u liked item i and did not like item j, and updated u's model accordingly. For each user u we sampled a number of triplets that is $60\times$ the number of non-zero entries in the training set of u. The model that achieved the best recall on u's validation set was returned.

7. RESULTS AND DISCUSSION

We structure the presentation of the results into two parts. The first (Section 7.1) compares the performance of UFSM with the other methods described in Section 6.2. The second (Sections 7.2 – 7.4) shows the effect of the different model parameters and dataset characteristics on the recommendation quality of UFSM.

7.1. Comparison with other methods

Table II shows the performance achieved by the various methods across the different datasets. These results correspond to the best set of results that we obtained for the different values for the various parameters associated with each method (e.g., number of latent factors, number of global similarity functions, and regularization parameters).

The results show that UFSM$_{bpr}$ achieves the best overall performance across the different datasets, both in terms of Rec@n and DCG@n. For the two MovieLens datasets, its performance difference over the second best performing scheme is considerable

Table II. Performance of UFSM and Other Techniques on the Different Datasets

Method	CUL			BX			ABB		
	Params	Rec@10	DCG@10	Params	Rec@10	DCG@10	Params	Rec@10	DCG@10
CoSim	-	0.336	0.076	-	0.176	0.070	-	0.283	0.101
PFW	μ_1=5e-1	0.341	0.077	μ_1=3e-1	0.181	0.073	μ_1=5e-1	0.287	0.107
RLFMI	h=30, λ=1e-3	0.174	0.041	h=40, λ=1e-4	0.036	0.003	h=40, λ=1e-3	0.025	0.008
AFMI$_{rmse}$	h=30, λ=1e-3	0.170	0.043	h=30, λ=1e-2	0.033	0.003	h=30, λ=1e-3	0.022	0.007
AFM$_{bpr}$	h=30, λ=1e-1	0.298	0.069	h=30, λ=1e-2	0.101	0.031	h=30, λ=5e-2	0.153	0.066
UFSM$_{rmse}$	l=1, μ_1=1e-1	0.334	0.077	l=1, μ_1=22e-1	0.170	0.070	l=1, μ_1=1e-1	0.276	0.098
UFSM$_{bpr}$	l=6, μ_1=1e-4, μ_2=5e-5, λ=1e-3	<u>0.370</u>	<u>0.086</u>	l=3, μ_1=3e-5, μ_2=5e-5, λ=1e-3	<u>0.199</u>	<u>0.076</u>	l=1, μ_1=5e-3	<u>0.295</u>	<u>0.112</u>

Method	ML-IMDB			ML-HR(genre)		
	Params	Rec@10	DCG@10	Params	Rec@10	DCG@10
CoSim	-	0.105	0.080	-	0.065	0.039
PFW	μ_1=9e-1	0.148	0.082	μ_1=5e-1	0.079	0.040
RLFMI	h=50, λ=1e-3	0.071	0.058	h=12, λ=1e-2	0.052	0.042
AFMI$_{rmse}$	h=40, λ=1e-3	0.065	0.041	h=12, λ=1e-2	0.041	0.018
AFM$_{bpr}$	h=30, λ=2e-2	0.147	0.093	h=12, λ=1e-2	0.090	0.045
UFSM$_{rmse}$	l=1, μ_1=5e-2	0.109	0.081	l=2, μ_1=5e-2, μ_2=5e-5, λ=5e-2	0.043	0.012
UFSM$_{bpr}$	l=1, μ_1=5e-3	<u>0.179</u>	<u>0.150</u>	l=3, μ_1=5e-3, μ_2=1e-4, λ=3e-3	<u>0.128</u>	<u>0.092</u>

The "Params" column shows the main parameters for each method. For RLFMI, AFMI$_{rmse}$ and AFM, h is the number of latent factors and λ is the regularizer used to learn the model parameters. For UFSM$_{rmse}$ and UFSM$_{bpr}$, l is the number of similarity functions, and λ, μ_1 and μ_2 are the regularization parameters described in Section 4.2 for the UFSM method. The "Rec@10" and "DCG@10" columns show the values obtained for these evaluation metrics. The entries that are underlined represent the best performance obtained for each dataset.

(21.8% and 42.2%), whereas for the CUL, BX and ABB datasets, the gains are 8.5%, 9.7% and 2.8%, respectively.

The performance of UFSM$_{rmse}$ is considerably worse than that of UFSM$_{bpr}$. This is consistent with previous research that showed that ranking-based loss functions are better suited for ranking binary user preferences than the RMSE loss function [Gantner et al. 2010]. This is also evident by the relative performance advantage of AFM$_{bpr}$ over AFMI$_{rmse}$. We expect that the relative performance of UFSM$_{rmse}$ will improve if the training procedure is provided with user ratings instead of binary user preferences since the RMSE loss function is suitable for estimating actual user ratings.

Comparing the performance of the latent factor based methods RLFMI, AFMI$_{rmse}$, and AFM$_{bpr}$ among themselves, we can see that AFM$_{bpr}$ does considerably better, substantially outperforming the other two across all datasets. We believe this is due to the BPR loss function as the nearly identical AFMI$_{rmse}$ method that uses an RMSE loss function does considerably worse. Also the results show that among RLFMI and AFMI$_{rmse}$, both of which used an RMSE loss function, the former does somewhat better especially for the denser ML-IMDB and ML-HR(genre) datasets.

Among the two non-collaborative based schemes, PFW does considerably better than CoSim across all the datasets. This is not surprising as PFW is inherently more powerful since it uses a ranking loss function to estimate a model from the data. Nevertheless, the gains that it achieves over CoSim are rather small for the very sparse datasets

(CUL, BX and ABB), and for the ML-HR(genre) dataset whose items are described in a low dimensional space.

One surprising aspect of the results is that the non-collaborative based methods (CoSim and PFW) did quite well relative to the latent factor models. For nearly all the datasets both schemes outperformed the latent factor models. The exceptions were the two MovieLens datasets where AFM$_{bpr}$ performed better or quite comparable to PFW. As mentioned in [Billsus and Pazzani 1999], such personalized neighborhood methods can provide recommendations even with a single preference per user, unlike other more sophisticated learning techniques which usually require a large number of training examples in order to be able to detect a meaningful pattern. Moreover, as mentioned in [Koren 2008], neighborhood methods can efficiently detect localized relationships as they only rely on a few yet significant item-item relations. Latent factor methods, on one hand, are more suited for detecting overall preference patterns while, on the other hand, they are not as efficient as neighborhood methods in detecting associations among related items. This can explain why CoSim and PFW performed better than the latent factor methods on the three sparser datasets.

7.2. Effect of the number of global similarity functions

Table III shows the performance achieved by UFSM$_{bpr}$ for different number of global similarity functions. These results show that for all datasets with high-dimensional feature spaces (CUL, BX, ABB and ML-IMDB) the performance achieved for the different number of global similarity functions does not change significantly. In fact for all of these datasets, the performance achieved using a single global similarity function is either the best (ABB and ML-IMDB) or very close to the best (CUL and BX). On the other had, for the ML-HR(genre) dataset, significant gains can be achieved by using more than one global similarity function. However, even for that dataset, there is little change in performance past three global similarity functions.

We believe that the reason why UFSM performs well with relatively small number of global similarity functions can be attributed to the fact that its underlying model can compactly capture the preferences of the users, even when these preferences are quite diverse. Recall from section 5 that the UFSM model can be represented as a sparse latent factor model in which the number of factors is $l \times n_U$. Even thought some of these factors are probably similar (e.g., for the users that have very similar membership functions), the resulting effective dimensionality can still model diverse user preferences.

Another factor is the dimensionality and sparsity of the item-feature matrix F. To show this, consider the case in which we have a sparse high-dimensional F that has a large number of features describing the items; as it is the case with all the datasets except ML-HR(genre). In this case, each feature potentially appears with a small number of items. For the CUL, BX, ML-IMDB and ABB datasets, each feature appears on average with less that 1% of the items. Accordingly, the subset of features appearing in the preference history of one user can be highly disjoint from the subset of features appearing in the history of other users. In this case, given a global similarity function gsim and its feature weight vector $\boldsymbol{w}$, only the weights corresponding to the features that appear in a user's history can influence the estimation of the preference scores for that user, and the other feature weights have no effect since these features do not appear in the user's history. Therefore, each global similarity function can model different preferences for different users and this can result in good performance using a small number of global similarity functions. On the other hand, in the case of a low-dimensional item-feature matrix F with limited number of features (i.e., F is a long thin matrix), a single feature can appear in many items as it is the case with the ML-HR(genre) dataset where each feature appears on average with more than 10%

Table III. Time for learning UFSM$_{bpr}$

# Global Similarity Functions	Rec@10				
	CUL	BX	ABB	ML-IMDB	ML-HR(genre)
1	0.3672	0.1983	0.2951	0.1792	0.1080
2	0.3665	0.1979	0.2948	0.1786	0.1205
3	0.3585	0.1986	0.2944	0.1786	0.1285
4	0.3669	0.1975	0.2895	0.1779	0.1280
5	0.3690	0.1976	0.2897	0.1781	0.1279
6	0.3704	0.1974	0.2904	0.1776	0.1283
7	0.3694	0.1971	0.2894	0.1772	0.1281

Table IV. Effect of the *Estimation Constraint* on the performance of UFSM$_{bpr}$ on the different datasets.

Method	Rec@10				
	CUL	BX	ABB	ML-IMDB	ML-HR(genre)
UFSM$_{bpr}$ with constraint	0.3672	0.1983	0.2951	0.1793	0.1284
UFSM$_{bpr}$ without constraint	0.3583	0.1913	0.2872	0.1785	0.1278

For the CUL, BX, ABB and ML-IMDB datasets, the results were generated using $l = 1$, whereas for the ML-HR(genre) dataset, the results were generated using $l = 3$.

of the items. In this case each feature can appear in the history of many users and the subset of features appearing in the preference history of one user can be highly overlapping with the subset of features appearing in the history of other users. In this case, a larger number of global similarity functions might be needed in order to define the key features that identify the preferences for each of the different users.

7.3. Effect of the *Estimation Constraint*

Recall from Section 4.1 that when we estimate the user preferences during training we do not include i in estimating its own preference score $\tilde{r}_{u,i}$. To show the effect of this constraint on the performance of UFSM, we estimated UFSM$_{bpr}$ with and without using it. Table IV shows the Rec@10 that we obtained for these two scenarios. These results show that excluding the known preference during the estimating process leads to higher Rec@10. These results are consistent with those reported in [Kabbur et al. 2013] in the context of estimating a factored representation of an item-item model.

7.4. Effect of Changing the Number of Recommended Items

Figure 1 shows the Rec@n for all the methods for values of n = 5, 10, 15, 20, and 25 on the ABB and the ML-HR(genre) datasets. UFSM$_{bpr}$ outperforms all the other methods for all values of n. For the ABB dataset, the rankings of the different methods do not change as n changes. For the ML-HR(genre) dataset, the different methods roughly maintain their rankings as n changes with one exception being the RLFMI and AFMI$_{rmse}$ methods whose Rec@n values are so close to one another.

7.5. Timing Performance of UFSM

Table V shows the training times for the different datasets for estimating UFSM$_{bpr}$. Times for estimating UFSM$_{rmse}$ are similar. As mentioned in Section 4.3, time to estimate UFSM depends on the total number of samples per learning iteration n_R, the

Fig. 1. Performance of the different methods with changing value of n for the ABB (left) and the ML-HR(genre) (right) datasets.

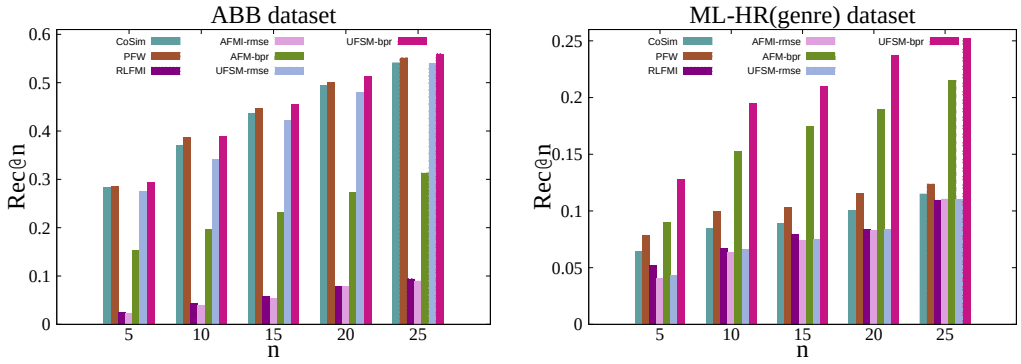

Table V. Time for estimating UFSM$_{bpr}$ in minutes.

CUL	BX	ABB	ML-IMDB	ML-HR(genre)
44.74	62.79	20.23	566.31	15.66

average number of preferences per user n_R^u, and the average number of features per item n_F^i. The ML-IMDB dataset takes the longest training time since all of the three factors have high values in this dataset. The ML-HR(genre) dataset takes the shortest training time since it has a relatively very small n_F^i.

8. CONCLUSION AND FUTURE WORK

In this paper we presented UFSM, a method for top-n item cold start recommendation. UFSM learns multiple feature-based global item similarity functions with user-specific combination weights. Each global similarity function has feature weights that determine the contribution of each feature to the similarity score under this global similarity function. UFSM can learn multiple global similarity functions as needed to account for the preferences of the different users. The user-specific combination weights allow UFSM to estimate similarity functions that are personalized for each user. In addition, we showed that UFSM can be mathematically represented as a sparse factor model in a latent space whose dimensionality is proportional to the number of users. The latent factors of each user directly incorporate the user's past preferences in order to account for the similarity of new items with the previously preferred items.

While in this work the global similarity functions of UFSM were represented using cosine similarity, other forms of similarity measures can be used in order to adjust UFSM to the designated application. Likewise, while the recommendation scores for each user are estimated using a set of user-specific features (i.e., the features of the items that the user has previously liked), it is possible to generate other forms of user-specific features in ways that can suite different applications. For example, if temporal information is available, one can construct user-specific features that make more use of the more recent preferences and discard the older ones.

We compared UFSM against non-collaborative user modeling techniques and state-of-the art latent factor-based techniques. Our results showed that UFSM outperforms all other methods in terms of recommendation quality on datasets with low- and high-

dimensional item feature spaces. Finally our results showed that using the BPR loss function in estimating the model parameters gives better recommendations compared to using a squared error loss function.

In the future, we plan to investigate the performance of UFSM on actual ratings instead of binary preferences. We expect that the performance of UFSM$_{rmse}$ would improve since the squared error loss function can better handle actual user ratings than binary user preferences.

ACKNOWLEDGMENTS

This work was supported in part by NSF (IOS-0820730, IIS-0905220, OCI-1048018, CNS-1162405, and IIS-1247632) 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

Deepak Agarwal and Bee-Chung Chen. 2009. Regression-based latent factor models. In *Proceedings of the 15th ACM SIGKDD international conference on Knowledge discovery and data mining (KDD '09)*. ACM, 19–28.

Amr Ahmed, Bhargav Kanagal, Sandeep Pandey, Vanja Josifovski, Lluis Garcia Pueyo, and Jeffrey Yuan. 2013. Latent factor models with additive and hierarchically-smoothed user preferences.. In *WSDM*. ACM, 385–394.

E. Banos, I. Katakis, N. Bassiliades, G. Tsoumakas, and I. Vlahavas. 2006. PersoNews: a personalized news reader enhanced by machine learning and semantic filtering. In *Proceedings of the 2006 Confederated international conference on On the Move to Meaningful Internet Systems: CoopIS, DOA, GADA, and ODBASE - Volume Part I (ODBASE'06/OTM'06)*. Springer-Verlag, Berlin, Heidelberg, 975–982. DOI:http://dx.doi.org/10.1007/11914853_62

Daniel Billsus and Michael J. Pazzani. 1999. A hybrid user model for news story classification. In *Proceedings of the seventh international conference on User modeling*. 99–108.

Léon Bottou. 1998. Online Algorithms and Stochastic Approximations. In *Online Learning and Neural Networks*, David Saad (Ed.). Cambridge University Press, Cambridge, UK. http://leon.bottou.org/papers/bottou-98x

Iván Cantador, Peter Brusilovsky, and Tsvi Kuflik. 2011. 2nd Workshop on Information Heterogeneity and Fusion in Recommender Systems (HetRec 2011). In *Proceedings of the 5th ACM conference on Recommender systems (RecSys 2011)*.

Zeno Gantner, Lucas Drumond, Christoph Freudenthaler, Steffen Rendle, and Schmidt-Thie Lars. 2010. Learning Attribute-to-Feature Mappings for Cold-Start Recommendations. In *Proceedings of the 2010 IEEE International Conference on Data Mining (ICDM '10)*. IEEE Computer Society, Washington, DC, USA, 176–185.

Rong Hu and Pearl Pu. 2011. Enhancing collaborative filtering systems with personality information. In *Proceedings of the fifth ACM conference on Recommender systems (RecSys '11)*. ACM, New York, NY, USA, 197–204. DOI:http://dx.doi.org/10.1145/2043932.2043969

Santosh Kabbur, Xia Ning, and George Karypis. 2013. 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 (KDD '13)*. ACM, 659–667.

Gilad Katz, Nir Ofek, Bracha Shapira, Lior Rokach, and Guy Shani. 2011. Using Wikipedia to boost collaborative filtering techniques. In *Proceedings of the fifth ACM conference on Recommender systems (RecSys '11)*. ACM, New York, NY, USA, 285–288. DOI:http://dx.doi.org/10.1145/2043932.2043984

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 (KDD '08)*. ACM, New York, NY, USA, 426–434. DOI:http://dx.doi.org/10.1145/1401890.1401944

Seung-Taek Park and Wei Chu. 2009. Pairwise preference regression for cold-start recommendation. In *Proceedings of the third ACM conference on Recommender systems (RecSys '09)*. ACM, New York, NY, USA, 21–28. DOI:http://dx.doi.org/10.1145/1639714.1639720

Steffen Rendle. 2012. Factorization Machines with libFM. *ACM Trans. Intell. Syst. Technol.* 3, 3, Article 57 (May 2012), 22 pages.

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 (UAI '09)*. AUAI Press, Arlington, Virginia, United States, 452–461. http://dl.acm.org/citation.cfm?id=1795114.1795167

Manuel de Buenaga Rodríguez, Manuel J. Maña López, Alberto Díaz Esteban, and Pablo Gervás Gómez-Navarro. 2001. A User Model Based on Content Analysis for the Intelligent Personalization of a News Service. In *Proceedings of the 8th International Conference on User Modeling 2001 (UM '01)*. Springer-Verlag, London, UK, UK, 216–218. http://dl.acm.org/citation.cfm?id=647664.733412

Badrul Sarwar, George Karypis, Joseph Konstan, and John Riedl. 2001. Item-based Collaborative Filtering Recommendation Algorithms. In *Proceedings of the 10th International Conference on World Wide Web (WWW '01)*. ACM, New York, NY, USA, 11.

Chong Wang and David M. Blei. 2011. Collaborative topic modeling for recommending scientific articles. In *Proceedings of the 17th ACM SIGKDD international conference on Knowledge discovery and data mining (KDD '11)*. ACM, New York, NY, USA, 448–456. DOI:http://dx.doi.org/10.1145/2020408.2020480

Wen-tau Yih. 2009. Learning Term-weighting Functions for Similarity Measures. In *Proceedings of the 2009 Conference on Empirical Methods in Natural Language Processing: Volume 2 - Volume 2 (EMNLP '09)*.

Liang Zhang, Deepak Agarwal, and Bee-Chung Chen. 2011. Generalizing matrix factorization through flexible regression priors. In *Proceedings of the fifth ACM conference on Recommender systems (RecSys '11)*. ACM, New York, NY, USA, 13–20. DOI:http://dx.doi.org/10.1145/2043932.2043940

Lanbo Zhang and Yi Zhang. 2010. Discriminative factored prior models for personalized content-based recommendation. In *Proceedings of the 19th ACM international conference on Information and knowledge management (CIKM '10)*. ACM, New York, NY, USA, 1569–1572. DOI:http://dx.doi.org/10.1145/1871437.1871674

Yi Zhang and Jonathan Koren. 2007. Efficient bayesian hierarchical user modeling for recommendation system. In *Proceedings of the 30th annual international ACM SIGIR conference on Research and development in information retrieval (SIGIR '07)*. 47–54.

Cai-Nicolas Ziegler, Sean M. McNee, Joseph A. Konstan, and Georg Lausen. 2005. Improving Recommendation Lists Through Topic Diversification. In *Proceedings of the 14th International Conference on World Wide Web (WWW '05)*. ACM, New York, NY, USA, 22–32.