Feature-based factorized Bilinear Similarity Model for Cold-Start Top-n Item Recommendation*

Mohit Sharma †, Jiayu Zhou ‡, Junling Hu‡, George Karypis†

Abstract

Recommending new items to existing users has remained a challenging problem due to absence of user's past preferences for these items. The user personalized non-collaborative methods based on item features can be used to address this item *cold-start* problem. These methods rely on similarities between the target item and user's previous preferred items. While computing similarities based on item features, these methods overlook the interactions among the features of the items and consider them independently. Modeling interactions among features can be helpful as some features, when considered together, provide a stronger signal on the relevance of an item when compared to case where features are considered independently. To address this important issue, in this work we introduce the Feature-based factorized Bilinear Similarity Model (FBSM), which learns factorized bilinear similarity model for Top-n recommendation of new items, given the information about items preferred by users in past as well as the features of these items. We carry out extensive empirical evaluations on benchmark datasets, and we find that the proposed FBSM approach improves upon traditional non-collaborative methods in terms of recommendation performance. Moreover, the proposed approach also learns insightful interactions among item features from data, which lead to deep understanding on how these interactions contribute to personalized recommendation.

1 Introduction

Top-n recommender systems are used to identify from a large pool of items those n items that are the most relevant to a user and have become an essential personalization and information filtering technology. They rely on the historical preferences that were either explicitly or implicitly provided for the items and typically employ various machine learning methods to build *content-agnostic* predictive models from these preferences. However, when new items are introduced into the system, these approaches cannot be used to compute personalized recommendations, because there are no prior preferences associated with those items. As a result, the methods used to recommend new items, referred to as (item) *cold-start* recommender systems, in addition to the historical information, take into account the characteristics of the items being recommended; that is, they are *content aware*. The items' characteristics are typically captured by a set of domain-specific features. For example, a movie may have features like genre, actors, and plot keywords; a book typically has features like content description and author information. These item features are intrinsic to the item and as such they do not depend on historical preferences.

Over the years, a number of approaches have been developed towards solving the item cold-start problem [7, 1, 17] that exploit the features of the new items and the features of the items on which a user has previously expressed his interest. A recently introduced approach, which was shown to outperform other approaches is the User-specific Feature-based Similarity Models (UFSM) [6]. In this approach, a linear similarity function is estimated for each user that depends entirely on features of the items previously liked by the user, which is then used to compute a score indicating how relevant a new item will be for that user. In order to leverage information across users (i.e., the transfer learning component that is a key component of collaborative filtering), each user specific similarity function is computed as a linear combination of a small number of *global* linear similarity functions that are shared across users. Moreover, due to the way that it computes the preference scores, it can achieve a high-degree of personalization while using only a very small number of global linear similarity functions.

In this work we extend UFSM in order to account for interactions between the different item features. We believe that such interactions are important and quite common. For example, in an e-commerce website, the items that users tend to buy are often designed to go well with previously purchased items (e.g., a pair of shoes that goes well with a dress). The set of features

*This work was supported in part by NSF (IIS-0905220, OCI-1048018, CNS-1162405, IIS-1247632, IIP-1414153, IIS-1447788), Army Research Office (W911NF-14-1-0316), Samsung Research America, and the Digital Technology Center at the University of Minnesota.

†University of Minnesota.

‡Samsung Research America

describing items of different type will be different (e.g., shoe material and fabric color) and as such a linear model can not learn from the data that for example a user prefers to wear leather shoes with black clothes. Being able to model such dependencies can lead to item cold-start recommendation algorithms that achieve better performance.

Towards this goal, we present a method called **F**eature-based factorized **B**ilinear **S**imilarity **M**odel (FBSM) that uses bilinear model to capture pairwise dependencies between the features. Like UFSM, FBSM learns a similarity function for estimating the similarity between items based on their features. However, unlike UFSM's linear global similarity function, FBSM's similarity function is bilinear. A challenge associated with such bilinear models is that the number of parameters that needs to be estimated becomes quadratic on the dimensionality of the item's feature space, which is problematic given the very sparse training data. FBSM overcomes this challenge by assuming that the pairwise relations can be modeled as a combination of a linear component and a low rank component. The linear component allows it to capture the direct relations between the features whereas the low rank component allows it to capture the pairwise relations. The parameters of these models are estimated using stochastic gradient descent and a ranking loss function based on Bayesian Personalized Ranking (BPR) that optimizes the area under the receiver operating characteristic curve.

We performed extensive empirical studies to evaluate the performance of the proposed FBSM on a variety benchmark datasets and compared it against state-of-the-art models for cold-start recommendation, including latent factor methods and non-collaborative user-personalized models. In our results FBSM optimized using BPR loss function outperformed other methods in terms of recommendation quality.

2 Notations and Definitions

Throughout the paper, all vectors are column vectors and are represented by bold lowercase letters (e.g., $\boldsymbol{f}_i$). Matrices are represented by upper case letters (e.g., R, P, Q).

The historical preference information is represented by a preference matrix R. 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 $n_{\mathcal{U}}$, and the set of items $\mathcal{I}$ has a size $n_{\mathcal{I}}$. $\mathcal{R}_u^+$ 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 intrinsic characteristics of that item. The feature vectors of all items are represented as the matrix F whose columns $\boldsymbol{f}_i$ correspond to the item feature vectors. The total number of item features is n_F.

The objective of the Top-n recommendation problem is to identify among the items that the user has not previously seen, the n items that he/she will like.

3 Related Work

The prior work to address the cold-start item recommendation can be divided into *non-collaborative* user personalized models and *collaborative* models. The non-collaborative models generate recommendations using only the user's past interaction history and the collaborative models combine information from the preferences of different users.

Billsus and Pazzani [3] developed one of the first user-modeling approaches to identify relevant new items. In this approach they used the users' past preferences to build user-specific models to classify new items as either "relevant" or "irrelevant". The user models were built using item features e.g., lexical word features for articles. Personalized user models [12] were also used to classify news feeds by modeling short-term user needs using text-based features of items that were recently viewed by user and long-term needs were modeled using news topics/categories. Banos [2] used topic taxonomies and synonyms to build high-accuracy content-based user models.

Recently collaborative filtering techniques using latent factor models have been used to address cold start item recommendation problems. These techniques incorporate item features in their factorization techniques. Regression-based latent factor models (RLFM) [1] is a general technique that can also work in item cold-start scenarios. RLFM learns a latent factor representation of the preference matrix in which item features are transformed into a low dimensional space using regression. This mapping can be used to obtain a low dimensional representation of the cold-start items. User's preference on a new item is estimated by a dot product of corresponding low dimensional representations. The RLFM model was further improved by applying more flexible regression models [17]. AFM [7] learns item attributes to latent feature mapping by learning a factorization of the preference matrix into user and item latent factors $R = PQ^T$. A mapping function is then learned to trans-

form item attributes to a latent feature representation i.e., $R = PQ^T = PAF^T$ where F represents items' attributes and A transforms the items' attributes to their latent feature representation.

User-specific Feature-based Similarity Models (UFSM) [6] learns a personalized user model by using historical preferences from all users across the dataset. In this model for each user an item similarity function is learned, which is a linear combination of user-independent similarity functions known as *global similarity functions*. Along with these global similarity functions, for each user a personalized linear combination of these global similarity functions is learned. It is shown to outperform both RLFM and AFM methods in cold-start Top-n item recommendations.

Predictive bilinear regression models [5] belong to the feature-based machine learning approach to handle the cold-start scenario for both users and items. Bilinear models can be derived from Tucker family [15]. They have been applied to separate "style" and "content" in images[14], to match search queries and documents [16], to perform semi-infinite stream analysis [13], and etc. Bilinear regression models try to exploit the correlation between user and item features by capturing the effect of pairwise associations between them. Let $\boldsymbol{x}_i$ denotes features for user i and $\boldsymbol{x}_j$ denotes features for item j, and a parametric bilinear indicator of the interaction between them is given by $s_{ij} = \boldsymbol{x}_i^T W \boldsymbol{x}_j$ where W denotes the matrix that describes a linear projection from the user feature space onto the item feature space. The method was developed for recommending cold-start items in the real time scenario, where the item space is small but dynamic with temporal characteristics. In another work [9], authors proposed to use a pairwise loss function in the regression framework to learn the matrix W, which can be applied to scenario where the item space is static but large, and we need a ranked list of items.

4 Feature-based Similarity Model

In this section we firstly introduce the feature-based linear model, analyzing the drawbacks of the model, and finally elaborate the technical details of our bilinear similarity model.

4.1 Linear Similarity Models. In UFSM [6] the preference score for 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.

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 a linear model on the feature vector resulting by the Hadamard product.

In author's results [6] for datasets with large number of features only one global similarity function was sufficient to outperform AFM and RLFM method for Top-n item cold-start recommendations. In case of only one global similarity function the user-specific similarity function is reduced to single global similarity function. Estimated preference score for new item i for user u is given by

$$\tilde{r}_{u,i} = \sum_{j \in \mathcal{R}_u^+} \text{sim}_u(i,j) = \sum_{j \in \mathcal{R}_u^+} \boldsymbol{w}_d(\boldsymbol{f}_i \odot \boldsymbol{f}_j)^T,$$

where $\boldsymbol{w}_d$ is the parameter vector, which can be estimated from training data using different loss functions.

4.2 Factorized Bilinear Similarity Models. An advantage that the linear similarity method(UFSM) has, over state of art methods such as RLFM and AFM, is that it uses information from all users across dataset to estimate the parameter vector $\boldsymbol{w}_d$. As in the principle of collaborative filtering, there exists users who have similar/dissimilar tastes and thus being able to use information from other users can improve recommendation for a user. However, we notice that a major drawback of this model is that it fails to discover pattern affinities between item features. Capturing these correlations among features sometimes can lead to significant improvements in estimating preference scores.

We thus propose FBSM to overcome this drawback: It uses bilinear formulation to capture correlation

among item features. Similar to UFSM, it considers information from all the users in dataset to learn these bilinear weights. In FBSM, the preference score for a new item i for user u is given by

$$\tilde{r}_{u,i} = \sum_{j \in \mathcal{R}_u^+} \text{sim}(i,j), \tag{4.1}$$

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

$$\text{sim}(i,j) = \boldsymbol{f_i}^T W \boldsymbol{f_j}$$

where W is the weight matrix which captures correlation among item features. Diagonal of matrix W determines how well a feature of item i say k^{th} feature of i i.e., f_{ik} interacts with corresponding feature of item j i.e., f_{jk} while off-diagonal elements of W gives the contribution to similarity by interaction of item feature with other features of item j i.e., contribution of interaction between f_{ik} and f_{jl} where $l \neq k$. Cosine similarity can be reduced to our formulation where W is a diagonal matrix with all the elements as ones.

A key challenge in estimating the bilinear model matrix W is that the number of parameters that needs to be estimated is quadratic in the number of features used to describe the items. For low-dimensional datasets, this is not a major limitation; however, for high-dimensional datasets, sometimes sufficient training data is not present for reliable estimation. This can become computationally infeasible, and moreover, lead to poor generalization performance by overfitting the training data. In order to overcome this drawback, we need to limit the degree of freedom of the solution W, and we propose to represent W as sum of diagonal weights and low-rank approximation of the off-diagonal weights:

$$W = D + V^T V \tag{4.2}$$

where D is a diagonal matrix of dimension equal to number of features whose diagonal is denoted using a vector $\boldsymbol{d}$, and $V \in \mathbf{R}^{h \times n_F}$ is a matrix of rank h. The columns of V represent latent space of features i.e., $\boldsymbol{v}_p$ represent latent factor of feature p. Using the low-rank approximation, the parameter matrix W of the similarity function is thus given by:

$$\begin{aligned} \text{sim}(i,j) &= \boldsymbol{f}_i^T W \boldsymbol{f}_j = \boldsymbol{f}_i^T (D + V^T V) \boldsymbol{f}_j \\ &= \boldsymbol{d}(\boldsymbol{f}_i \odot \boldsymbol{f}_j)^T + \sum_{k=1}^{n_F} \sum_{p=1}^{n_F} f_{ik} f_{jp} v_k^T v_p \end{aligned} \tag{4.3}$$

The second part of equation 4.3 captures the effect of off-diagonal elements of W by inner product of latent factor of features. Since we are now estimating only diagonal weights and low-rank approximation of off-diagonal weights, the computation reduces significantly compared to when we were trying to estimate the complete matrix W. This also gives us a flexible model where we can regularize diagonal weights and feature latent factors separately.

The bilinear model may look similar to the formulation described in [5], and however the two are very different in nature: in [5] the bilinear model is used to capture correlation among user and item features, on contrary the FBSM is trying to find correlation within features of items itself. The advantage of modeling interactions among item features is especially attractive when there is no explicit user features available. Note that it is not hard to encode the user features in the proposed bilinear model such that the similarity function is parameterized by user features, and we leave a detailed study to an extension of this paper.

4.3 Parameter Estimation of FBSM. FBSM is parameterized by $\Theta = [D, V]$, where D, V are the parameters of the similarity function. The inputs to the learning process are: (i) the preference matrix R, (ii) the item-feature matrix F, and (iii) the dimension of latent factor of features. There are many loss functions we can choose to estimate Θ, among which the Bayesian Personalized Ranking (BPR) loss function [11] is designed especially for ranking problems. In the Top-n recommender systems, the predicted preference scores are used to rank the items in order to select the highest scoring n items, and thus the BPR loss function can better model the problem than other loss functions such as least squares loss and in general lead to better empirical performance [7, 11]. As such, in this paper, we propose to use the BPR loss function, and in this section we show how the loss function can be used to estimate the parameters Θ. Note that other loss functions such as least squared loss can be applied similarity.

We denote the problem of solving FBSM using BPR as FBSM_{bpr}, and the loss function is given by

$$\mathcal{L}_{bpr}(\Theta) \equiv - \sum_{u \in U} \sum_{\substack{i \in \mathcal{R}_u^+, \\ j \in \mathcal{R}_u^-}} \ln \sigma(\tilde{r}_{u,i}(\Theta) - \tilde{r}_{u,j}(\Theta)), \tag{4.4}$$

where, $\tilde{r}_{u,i}$ is the predicted value of the user u's preference for the item i and σ is the sigmoid function. The BPR loss function tries to learn item preference scores such that the items that a user likes have higher preference scores than the ones he/she does not like, regardless of the actual item preference scores. The prediction value $\tilde{r}_{u,i}$ is given by:

$$\tilde{r}_{u,i} = \sum_{j \in \mathcal{R}_u^+ \setminus i} \text{sim}_u(i,j), \tag{4.5}$$

which is identical to Equation 4.1 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* [8].

To this end, the model parameters $\Theta = [D, V]$ are estimated via an optimization process of the form:

$$\min_{\Theta=[D,V]} \mathcal{L}_{bpr}(\Theta) + \lambda\|V\|_F^2 + \beta\|D\|_F^2, \tag{4.6}$$

where we penalize the frobenius norm of the model parameters in order to control the model complexity and improve its generalizability.

To optimized Eq. (4.6) we proposed to use stochastic gradient descent(SGD) [4], in order to handle large-scale datasets. The update steps for D, V are based on triplets (u, i, j) sampled from training data. For each triplet, we need to compute the corresponding estimated relative rank $\tilde{r}_{u,ij} = \tilde{r}_{u,i} - \tilde{r}_{u,j}$. Let

$$\tau_{u,ij} = \text{sigmoid}(-\tilde{r}_{u,ij}) = \frac{e^{-\tilde{r}_{u,ij}}}{1 + e^{-\tilde{r}_{u,ij}}},$$

the updates are then given by:

$$D = D + \alpha_1\Big(\tau_{u,ij}\nabla_D\tilde{r}_{u,ij} - 2\beta D\Big), and \tag{4.7}$$

$$\boldsymbol{v}_p = \boldsymbol{v}_p + \alpha_2\Big(\tau_{u,ij}\nabla_{\boldsymbol{v}_p}\tilde{r}_{u,ij} - 2\lambda\boldsymbol{v}_p\Big). \tag{4.8}$$

4.4 Performance optimizations In our approach, the direct computation of gradients is time-consuming and is prohibitive when we have high-dimensional item features. For example, the relative rank $\tilde{r}_{u,ij}$ given by

$$\begin{aligned}\tilde{r}_{u,ij} = &\left(\boldsymbol{f}_i^T(D + V^TV)\Big(\Big(\sum_{q\in\mathcal{R}_u^+\setminus i}\boldsymbol{f}_q\Big) - \boldsymbol{f}_i\Big)\right) - \\ &\left(\boldsymbol{f}_j^T(D + V^TV)\Big(\sum_{q\in\mathcal{R}_u^+}\boldsymbol{f}_q\Big)\right),\end{aligned} \tag{4.9}$$

has complexity of $O(|\mathcal{R}_u^+|n_Fh)$, where h is the dimensionality of latent factors, n_F is the number of features.

To efficiently compute these, let

$$\boldsymbol{f_u} = \sum_{q\in\mathcal{R}_u^+}\boldsymbol{f}_q,$$

which can be precomputed once for all users.

Then, Equation 4.9 becomes

$$\begin{aligned}\tilde{r}_{u,ij} &= \Big(\boldsymbol{f}_i^T(D + V^TV)(\boldsymbol{f}_u - \boldsymbol{f}_i)\Big) - \\ &\quad \Big(\boldsymbol{f}_j^T(D + V^TV)\boldsymbol{f}_u\Big) \\ &= \Big((\boldsymbol{f}_i - \boldsymbol{f}_j)^TD\boldsymbol{f}_u - \boldsymbol{f}_i^TD\boldsymbol{f}_i\Big) + \\ &\quad \Big((\boldsymbol{f}_i - \boldsymbol{f}_j)^T(V^TV)\boldsymbol{f}_u - \boldsymbol{f}_i^TV^TV\boldsymbol{f}_i\Big) \\ &= \Big(\boldsymbol{\delta}_{ij}^TD\boldsymbol{f}_u - \boldsymbol{f}_i^TD\boldsymbol{f}_i\Big) + \\ &\quad \Big(\boldsymbol{\delta}_{ij}^T(V^TV)\boldsymbol{f}_u - \boldsymbol{f}_i^TV^TV\boldsymbol{f}_i\Big) \\ &= \Big(\boldsymbol{\delta}_{ij}^TD\boldsymbol{f}_u - \boldsymbol{f}_i^TD\boldsymbol{f}_i\Big) + \\ &\quad \Big((V\boldsymbol{\delta}_{ij})^T(V\boldsymbol{f}_u) - (V\boldsymbol{f}_i)^T(V\boldsymbol{f}_i)\Big),\end{aligned}$$

where $\boldsymbol{\delta}_{ij} = \boldsymbol{f}_i - \boldsymbol{f}_j$.

The complexity of computing the relative rank then becomes $O(n_Fh)$, which is lower than complexity of Equation 4.9.

The gradient of the diagonal component is given by

$$\frac{\partial\tilde{r}_{u,ij}}{\partial D} = \Big(\boldsymbol{\delta}_{ij}\otimes\boldsymbol{f}_u - \boldsymbol{f}_i\otimes\boldsymbol{f}_i\Big). \tag{4.10}$$

where $\otimes$ represents elementwise scalar product. The complexity of Equation 4.10 is given by $O(n_F)$.

The gradient of the low rank component is given by

$$\frac{\partial\tilde{r}_{u,ij}}{\partial\boldsymbol{v}_p} = \boldsymbol{\delta}_{ij,p}(Vf_u) + \boldsymbol{f}_{up}(V\boldsymbol{\delta}_{ij}) - 2\boldsymbol{f}_{ip}(Vf_i), \tag{4.11}$$

whose complexity is $O(n_Fh)$.

Hence, the complexity of gradient computation for all the parameters is given by $O(n_Fh + n_F) \approx O(n_Fh)$. We were able to obtain both the estimated relative rank and all the gradients in $O(n_Fh)$, which is linear with respect to feature dimensionality as well as the size of latent factors and the number of global similarity functions. This allows the FBSM$_{bpr}$ to process large-scale datasets.

We note that the proposed FBSM$_{bpr}$ method is closely related to the factorization machine (FM) [10], in that both are exploring the interactions among the features. However, there is one key difference between these two: while the FM is heavily dependent on the quality of the user features, the proposed method does not depend on such user features.

5 Experimental Evaluation

In this section we perform experiments to demonstrate the effectiveness of the proposed algorithm.

Algorithm 1 FBSM$_{bpr}$-Learn

procedure FBSM $_{bpr}$-LEARN
  $\lambda \leftarrow$ V regularization weight
  $\beta \leftarrow$ D regularization weight
  $\alpha_1, \alpha_2 \leftarrow$ D and V learning rates
  Initialize $\Theta = [D, V]$ randomly

  while not converged **do**
    for each user u **do**
      sample a pair (i, j) s.t. $i \in \mathcal{R}_u^+$, $j \in \mathcal{R}_u^-$
      compute $\tilde{r}_{u,ij} = \tilde{r}_{u,i} - \tilde{r}_{u,j}$
      compute $\nabla_D \tilde{r}_{u,ij}$
      compute $\nabla_{\boldsymbol{v}_p} \tilde{r}_{u,ij}$
      update D using (4.7)
      update $\boldsymbol{v}_p \forall p$ using (4.8)
    end for
  end while

  return $\Theta = [D, V]$
end procedure

5.1 Datasets We used four datasets (Amazon Books, MovieLens-IMDB, CiteULike, Book Crossing) to evaluate the performance of FBSM.

Amazon Books(AMAZON) is a dataset collected from amazon about best-selling books and and their ratings. The ratings are binarized by treating all ratings greater than equal to 3 as 1 and ratings below 3 as 0. Each is accompanied with a description which was used as item's content.

CiteULike(CUL)[1] aids researchers by allowing them to add scientific articles to their libraries. For users of the CUL, the articles present in their library are considered as preferred articles i.e. 1 in a preference matrix while rest are considered as implicit 0 preferences.

MovieLens-IMDB (ML-IMDB) is a dataset extracted from the IMDB and the MovieLens-1M datasets[2] by mapping the MovieLens and IMDB movie IDs and collecting the movies that have plots and keywords. The ratings were binarized similar to AMAZON by treating all ratings greater than 5 as 1 and below or equal to 5 as 0. The movies plots and keywords were used as the item's content. Book Crossing (BX) dataset is extracted from Book Crossing data [18] such that user has given at least four ratings and each book has received the same amount of ratings. Description of these books were collected from Amazon using ISBN and were used as item features.

For the AMAZON, CUL, BX 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 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.

Various statistics about these datasets are shown in Table 5.1. Most of these datasets contain items that have high-dimensional feature spaces. Also comparing the densities of the datasets we can see that the MovieLens dataset have significantly higher density than other dataset.

5.2 Comparison methods We compared FBSM against non-collaborative personalized user modeling methods and collaborative methods.

1. **Non-Collaborative Personalized User Modeling Methods** Following method is quite similar to method described in [3]
   - **Cosine-Similarity (CoSim):** This is a personalized user-modeling method. The preference score of user u on target item i is estimated using equation 4.5 by using *cosine similarity* between item features.
2. **Collaborative Methods**
   - **User-specific Feature-based Similarity Models (UFSM):** As mentioned before, this method [6] learns personalized user model by using all past preferences from users across the dataset. It outperformed other state of the art collaborative latent factor based methods e.g., RLFM[1], AFM[7] by significant margin.
   - **RLFMI**: We used the Regression-based Latent Factor Modeling(RLFM) technique implemented in factorization machine library LibFM[10] that accounts for inter-feature interactions. We used LibFM with SGD learning to obtain RLFMI results.

5.3 Evaluation Methodology and Metrics We evaluated performance of methods using the following procedure. For each dataset we split the corresponding user-item preference matrix R into three matrices R_{train}, R_{val} and R_{test}. R_{train} contains a randomly selected 60% of the columns (items) of R, and the remaining columns were divided equally among R_{val} and R_{test}. Since items in R_{test} and R_{val} are not present in R_{train}, this allows us to evaluate the methods for item cold-start problems as users in R_{train} do not have any preferences

[1] http://citeulike.org/
[2] http://www.movielens.org, http://www.imdb.com

Table 1: Statistics for the 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%
BX	17,219	36,546	8,946	574,127	33.3	15.7	0.09%
AMAZON	13,097	11,077	5,766	175,612	13.4	15.9	0.12%
ML-IMDB	2,113	8,645	8,744	739,973	350.2	85.6	4.05%

Table 2: Performance of FBSM and Other Techniques on the Different Datasets

Method	CUL			BX		
	Params	Rec@10	DCG@10	Params	Rec@10	DCG@10
CoSim	-	0.1791	0.0684	-	0.0681	0.0119
RLFMI	h=75	0.0874	0.0424	h=75	0.0111	0.003
UFSM$_{bpr}$	l=1, μ_1=0.25	0.2017	0.0791	l=1, μ_1=0.1	0.0774	0.0148
FBSM$_{bpr}$	λ=0.25, β=10, h=5	<u>0.2026</u>	<u>0.0792</u>	λ=1, β=100, h=1	<u>0.0776</u>	<u>0.0148</u>
Method	**ML-IMDB**			**AMAZON**		
	Params	Rec@10	DCG@10	Params	Rec@10	DCG@10
CoSim	-	0.0525	0.1282	-	0.1205	0.0228
RLFMI	h = 15	0.0155	0.0455	h = 30	0.0394	0.0076
UFSM$_{bpr}$	l=1, μ_1=0.005	0.0937	0.216	l=1, μ_1=0.25,	0.1376	0.0282
FBSM$_{bpr}$	λ=0.01, β=0.1, h=5	<u>0.0964</u>	<u>0.227</u>	λ=0.1, β=1, h=1	<u>0.1392</u>	<u>0.0284</u>

The "Params" column shows the main parameters for each method. For UFSM $_{bpr}$, l is the number of similarity functions, and λ, μ_1 is the regularization parameter. For FBSM, λ and β are regularization parameters and h is dimension of feature latent factors. 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.

for items in R_{test} or R_{val}. The models are learned using R_{train} and the best model is selected based on its performance on the validation set R_{val}. The selected model is then used to estimate the preferences over all items in R_{test}. For each user the items are sorted in decreasing order and the first n items are returned as the Top-n recommendations for each user. The evaluation metrics as described later are computed using these Top-n recommendation for each user.

After creating the train, validation and test split, there might be some users who do not have any items in validation or test split. In that case we evaluate performance on the splits for only those user who have at least one item in corresponding test split. This split-train-evaluate procedure is repeated three times for each dataset and evaluation metric scores are averaged over three runs before being reported in results.

We used two metrics to assess the performance of the various methods: Recall at n (Rec@n) and Discounted Cumulative Gain at n (DCG@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, whereas the DCG@n measures how high the relevant items were placed in the list. The Rec@n is

defined as

$$REC@n = \frac{|\{\text{Items liked by user}\} \cap \{\text{Top-n items}\}|}{|\text{Top-n items}|}$$

The DCG@n 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

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

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. Both the Rec@n and the DCG@n are computed for each user and then averaged over all the users.

5.4 Model Training FBSM's model parameters are estimated using training set R_{train} and validation set R_{val}. After each major SGD iteration of Algorithm 1 we compute the Rec@n on validation set and save the current model if current Rec@n is better than those computed in previous iterations. The learning process ends when the optimization objective converges or no further improvement in validation recall is observed for 10 major SGD iterations. At the end of learning process we return the model that achieved the best Rec@n on the validation set.

To estimate the model parameters of FBSM_{bpr}, we draw samples equal to the number of preferences in R for each major SGD iteration. Each sample consists of a user, an item preferred by user and an item not preferred by user. If a dataset does not contain items not preferred by user then we sample from items for which his preference is not known.

For estimating RLFMI model parameters, LibFM was given the training and validation sets and the model that performed best on the validation set was used for evaluation on test sets. For RLFMI, the training set must contain both 0's and 1's. Since the CUL dataset does not contain both 0's and 1's, we sampled 0's equal to number of 1's in R from the unknown values.

6 Results and Discussion

6.1 Comparison with previous methods We compared the performance of FBSM with other methods described in Section on 5.2. Results are shown in Table 2 for different datasets. We tried different values for various parameters e.g., latent factors and regularization parameters associated with methods and report the best results found across datasets.

These results illustrate that FBSM_{bpr} by modeling the cross feature interactions among items can improve upon the UFSM method [6] which has been shown to outperform the existing state of the art methods like RLFM[1] and AFM[7]. Similar to the UFSM method, FBSM_{bpr} method has outperformed latent-factor based RLFMI method. An example of cross-feature interactions found by FBSM is interaction among terms *tragic, blockbuster,* and *famous.*

6.2 Performance investigation at user level We further looked at some of our datasets ($ML-IMDB$ and $AMAZON$) and divided the users based on the performance achieved by FBSM in comparison with UFSM i.e., users for which FBSM performed better, similar and worse than UFSM. These finding are presented in Table 3. For ML-IMDB dataset there is an increase of 22% in number of users for whom recommendation is better on using FBSM method, while for $AMAZON$ dataset the number of users that benefited from FBSM is not significant. On comparing the two datasets in Table 3, $ML-IMDB$ has much more preferences per item or existing items have been rated by more users compared to $AMAZON$. Hence our proposed method FBSM takes the advantage of availability of more data while UFSM fails to do so.

7 Conclusion

We presented here FBSM for Top-n recommendation in item cold-start scenario. It tries to learn a similarity function between items, represented by their features, by using all the information available across users and also tries to capture interaction between features by using a bilinear model. Computation complexity of bilinear model estimation is significantly reduced by modeling the similarity as sum of diagonal component and off-diagonal component. Off-diagonal component are further estimated as dot product of latent spaces of features.

In future, we want to investigate the effect of non-negativity constraint on model parameters and effectiveness of the method on actual rating prediction instead of Top-n recommendation.

References

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

[2] E. Banos, I. Katakis, N. Bassiliades, G. Tsoumakas, and I. Vlahavas. Personews: a personalized news

Table 3: User level investigation for ML-IMDB and AMAZON

Dataset	FBSM against UFSM	users	items	average user preferences	average item preferences
ML-IMDB	BETTER	887	4770	224	42
	SAME	802	4371	119	22
	WORSE	424	3928	137	15
AMAZON	BETTER	325	4170	23	2
	SAME	12458	6638	7	13
	WORSE	314	4294	24	2

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, pages 975–982, Berlin, Heidelberg, 2006. Springer-Verlag.

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

[4] Léon Bottou. Online algorithms and stochastic approximations. In David Saad, editor, *Online Learning and Neural Networks*. Cambridge University Press, Cambridge, UK, 1998.

[5] Wei Chu and Seung-Taek Park. Personalized recommendation on dynamic content using predictive bilinear models. In *Proceedings of the 18th International Conference on World Wide Web*, WWW '09, pages 691–700, New York, NY, USA, 2009. ACM.

[6] Asmaa Elbadrawy and George Karypis. Feature-based similarity models for top-n recommendation of new items. Technical Report 14-016, Department of Computer Science, University of Minnesota, Minneapolis, Minnesota, June 2013.

[7] Zeno Gantner, Lucas Drumond, Christoph Freudenthaler, Steffen Rendle, and Schmidt-Thie Lars. Learning attribute-to-feature mappings for cold-start recommendations. In *Proceedings of the 2010 IEEE International Conference on Data Mining*, ICDM '10, pages 176–185, Washington, DC, USA, 2010. IEEE Computer Society.

[8] Santosh Kabbur, Xia Ning, and George Karypis. Fism: Factored item similarity models for top-n recommender systems. In *Proceedings of the 19th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, KDD '13, pages 659–667. ACM, 2013.

[9] Seung-Taek Park and Wei Chu. Pairwise preference regression for cold-start recommendation. In *Proceedings of the Third ACM Conference on Recommender Systems*, RecSys '09, pages 21–28, New York, NY, USA, 2009. ACM.

[10] Steffen Rendle. Factorization machines with libFM. *ACM Trans. Intell. Syst. Technol.*, 3(3):57:1–57:22, May 2012.

[11] Steffen Rendle, Christoph Freudenthaler, Zeno Gantner, and Lars Schmidt-Thieme. Bpr: Bayesian personalized ranking from implicit feedback. In *Proceedings of the Twenty-Fifth Conference on Uncertainty in Artificial Intelligence*, UAI '09, pages 452–461, Arlington, Virginia, United States, 2009. AUAI Press.

[12] Manuel de Buenaga Rodríguez, Manuel J. Maña López, Alberto Díaz Esteban, and Pablo Gervás Gómez-Navarro. 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, pages 216–218, London, UK, UK, 2001. Springer-Verlag.

[13] Jimeng Sun, Dacheng Tao, and Christos Faloutsos. Beyond streams and graphs: Dynamic tensor analysis. In *In KDD*, pages 374–383, 2006.

[14] Joshua B. Tenenbaum and William T. Freeman. Separating style and content with bilinear models. *Neural Comput.*, 12(6):1247–1283, June 2000.

[15] LedyardR Tucker. Some mathematical notes on three-mode factor analysis. *Psychometrika*, 31(3):279–311, 1966.

[16] Wei Wu, Zhengdong Lu, and Hang Li. Learning bilinear model for matching queries and documents. *J. Mach. Learn. Res.*, 14(1):2519–2548, January 2013.

[17] Liang Zhang, Deepak Agarwal, and Bee-Chung Chen. Generalizing matrix factorization through flexible regression priors. In *Proceedings of the fifth ACM conference on Recommender systems*, RecSys '11, pages 13–20, New York, NY, USA, 2011. ACM.

[18] Cai-Nicolas Ziegler, Sean M. McNee, Joseph A. Konstan, and Georg Lausen. Improving recommendation lists through topic diversification. In *Proceedings of the 14th International Conference on World Wide Web*, WWW '05, pages 22–32, New York, NY, USA, 2005. ACM.