Synergies that Matter: Efficient Interaction Selection via Sparse Factorization Machine

Jianpeng Xu, Kaixiang Lin, Pang-Ning Tan, Jiayu Zhou

Department of Computer Science and Engineering, Michigan State University, East Lansing, MI
Email: {xujianpe, linkaixi, ptan, jiayuz}@msu.edu

Abstract

Collaborative filtering has been widely used in modern recommender systems to provide accurate recommendations by leveraging historical interactions between users and items. The presence of cold-start items and users has imposed a huge challenge to recommender systems based on collaborative filtering, because of the unavailability of such interaction information. The factorization machine is a powerful tool designed to tackle the cold-start problems by learning a bilinear ranking model that utilizes content information about users and items, exploiting the interactions with such content information. While a factorization machine makes use of all possible interactions between all content features to make recommendations, many of the features and their interactions are not predictive of recommendations, and incorporating them in the model will deteriorate the generalization performance of the recommender systems. In this paper, we propose an efficient Sparse Factorization Machine (SFM), that simultaneously identifies relevant user and item content features, models interactions between these relevant features, and learns a bilinear model using only these synergistic interactions. We have carried out extensive empirical studies on both synthetic and real-world datasets, and compared our method to other state-of-the-art baselines, including Factorization Machine. Experimental results show that SFM can greatly outperform other baselines.

1 Introduction

The explosion of information has created a huge demand on developing recommender systems to help people navigate through the ever increasing amount of information and scope out only a small and relevant subset. The job of the recommender systems is to recommend a few *items* (e.g., books sold by Amazon) to human *users* from all the items stored in the database. Among many types of recommender systems, the ones based on collaborative filtering are shown to be especially effective, thanks to its capability to make personalized recommendation by exploiting the historical interactions between users and items to be recommended. The key idea behind this powerful technique is very intuitive: recommend similar items to users with similar preferences. Over the past decade, the collaborative filtering based recommender systems have been intensively studied and widely implemented in various applications, including recommendations for movies [23], music [14], news [5], books [30], research articles [26], dating [6], and TV shows [7].

Since the collaborative filtering models are heavily dependent on the historical interactions to model preferences and make recommendations (a.k.a content-agnostic recommender system [25]), they are susceptible to the so-called cold-start problem [9]. A *cold-start item* is referred as an item that is just added to the database and there is little historical information on how users interact with it. For example, a book that is just published and posted on the Amazon website is a cold-start item. Similarly, a *cold-start user* is referred as a user that has no historical interactions with any items in the system, for example, a newly registered user who has neither made any purchase on Amazon, nor left any ratings and reviews. Due to the lack of historical information of the cold-start items and users, the recommender systems which rely on historical information cannot be directly applied to get personalized recommendations.

In order to tackle the cold-start problem, side information (also known as content information) about users and items is typically taken into account to bridge them to the recommender systems. In the movie recommendation, examples of such content information include title, actors, directors of the movies, keywords for movies and so on. The content information about users include genders, ages, incomes, addresses for the users and etc. From the user/item content information we are able to extract corresponding numerical user/item features to

represent such information. Gartner et al. [12] and Forbes and Zhu [11] have proposed to integrate the content information to handle cold-start issues. [12] learns feature mappings from both user and item features to latent factors, and use the mapped factors to recover the rating matrix. [11] proposed to learn the mapping and factor simultaneously, but only used item features.

The factorization machine [20] is among the recommender systems that consider synergistic interactions between the features. It has attracted a lot of attention in the community because it provides a principled framework to combine the collaborative filtering and the power of feature engineering from user/item content information. The factorization machine can be considered as a bilinear regression model that generates a ranking score for each user-item pair based on the interactions between the user content features and item content features. The algorithm for factorization machine can be very efficient even for a huge feature space, since it enforces a low-rank bilinear parameter matrix by factorizing it into two small matrix factors. However, we note that even though the computation bottleneck has been properly addressed, the factorization machine is in nature learning a *dense* bilinear matrix, and thus the ranking score is given by a linear combination of ALL interactions between EVERY pair of features. For example, the number of content features (including both user and item) in a typical TV show recommender system can easily go over 10,000 (e.g., natural language processing features such as TF-IDF extracted from synopsis of a TV show), where the final ranking score is given by a combination of 49,995,000 interactions. Since there may only be a small subset of features that are relevant to the recommendation preferences, those interactions involving irrelevant features can only be considered as noisy interactions. As such, the factorization machine ranking model is extremely hard to interpret, and can lead to suboptimal recommendation performance.

To address the aforementioned problem, we propose a novel recommendation model called Sparse Factorization Machine (SFM), which simultaneously identifies predictive content features and corresponding relevant interactions among these features, and builds a bilinear recommendation model using only these relevant interactions. Thus the resulting recommendation model is then generating ranking scores using interactions from only relevant content features. Specifically, we firstly reformulate the factorization machine formulation and then leverage the $\ell_{2,1}$-norm to induce group sparsity on the latent factors of factorization machine model. We propose to use the efficient proximal alternating linearized minimization framework to solve the optimization problem. We evaluate the proposed SFM on benchmark recommendation datasets, including Amazon book, MovieLens, and Book Cross. The results from SFM have demonstrated promising recommendation performance of the proposed approach.

The rest of paper is organized as follows. In section 2 we introduce the related works towards to the state-of-the-art of learning to rank algorithms. In section 3, we formulate the SFM framework formally and provide an efficient optimization method for the solution. In section 4, we perform extensive experiments on real world datasets. Finally, section 5 concludes the paper.

2 Related Works

Collaborative filtering recommender systems assume that the users with similar opinions in historical records tend to share similar opinions in the future[22, 8]. Matrix factorization techniques[28, 15, 29, 27] has been widely used in collaborative filtering, due to its advantageousness over k-nearest neighbor approaches[1]. By learning the latent factors of users and items, the rating matrix is factorized into two low-rank matrices, each of which represents a latent factor for user or item. Even though matrix factorization can achieve good recommendation results, it lacks of the ability for handling cold-start problem, due to the fact that the matrix factorization works as matrix completion.

In order to handle cold-start recommendation problem, content information has been incorporated into the matrix factorization methods[2, 3, 24, 12, 11, 18]. For example, Agarwal et al. proposed a graphical model by introducing the latent factors from both users and items[2][3]. However, solving the graphical model by Monte-Carlo EM algorithm is not efficient when the data is large. Similarly, Shan et al. proposed a generalized probabilistic matrix factorization by incorporating topic models over the side information[24]. However, the algorithm only considered the side information from the item. Gantner et al. proposed a two-step procedure, where the rating matrix is factorized into two factors, and then a mapping is learned from the features of users and items to the factors[12]. Forbes et al. improved Gantner's work by integrating the matrix factorization step and mapping step into one optimization problem[11]. However, this work only considered the mapping from item features to the factor related to items, which indicates that it only handles the cold-start on items.

Factorization machines[19, 21] (FM) has been proposed as a general framework of content-aware collaborative filtering method. It considers a bilinear modeling in the formulation to take into account the pairwise interactions between features, which are usually designed via feature engineering in order to get good per-

formance. Hong et al. [13] proposed a co-factorization machine (CoFM) to model user interests and discover the topics from content of tweets simultaneously. Two FMs are built for each problem and correlated with each other by sharing some common features or latent factors. Loni et al. [16] applied FMs to cross-domain collaborative filtering. This method exploits knowledge from auxiliary domains to improve recommendation on a target domain, by concatenating the features from auxiliary domains and target domain. Although these methods are potentially applicable to cold-start problem, they are all based on FMs. Note that FMs need to formulate all pair-wise feature interactions, which increases the model complexity and hence will possibly subject to overfitting. Cheng et al. [10] proposed a new framework named Gradient Boosting Factorization Machines (GBFM) for context-aware recommendation system. It reduces the model complexity by selecting feature interactions using a greedy algorithm based on gradient boosting. However, it might not be optimal to select interactions using heuristics and the number of feature interactions could still be large. To tackle this challenge, we propose to select feature interactions by selecting useful features directly from the learning models where all interactions from a useless feature will be removed.

3 Sparse Factorization Machine

In this section, we will introduce our sparse factorization machine recommender system.

3.1 Problem Formulation Consider a dataset $\mathcal{D} = \{(x_p^U, x_q^I, r_{p,q})\}$, where $x_p^U \in \mathcal{R}^{d_U}$ represents the feature of user p, $x_q^I \in \mathcal{R}^{d_I}$ represents the feature of item q, and $r_{p,q} \in \mathcal{R}$ is the rating of user p on item q. The goal is to learn a model to predict the ranking scores $R = \{r_{p,q}\} \in \mathcal{R}^{n_U \times n_I}$ given $X_U = \{x_p^U\} \in \mathcal{R}^{n_U \times d_U}$ and $X_I = \{x_q^I\} \in \mathcal{R}^{n_I \times d_I}$. Let Θ be the parameters of the ranking function, and the ranking score of item q for user p is given by $\hat{r}_{p,q} = f(x_p^U, x_q^I; \Theta)$. In order to simplify our representation, we denote $x = [x^U, x^I] \in \mathcal{R}^d$, where $d \equiv d_U + d_I$, and denote $\hat{y}_i = f(x_i; \Theta)$. We can collectively denote $X = \{x\}$. Note that the simple index i implies a pair of indices: a user index p and an item index q.

We learn the parameters Θ using a ranking loss function $\mathcal{L}$ and the learning problem becomes:

$$\min_{\Theta} \mathcal{L}(\Theta; \mathcal{D}) \equiv \mathcal{L}(\Theta) = \sum_{i \in |\mathcal{D}|} \ell(f(x_i; \Theta), r_i) \tag{3.1}$$

where we have dropped the dependency on the data $\mathcal{D}$ for simplicity of the notations. Our recommendation problem therefore falls into the supervised learning paradigm. The key to this problem is to design the ranking function $f(x; \Theta)$. A good ranking function should have a relative low complexity that can be computed efficiently, and meanwhile should have the capability to identify key features contributing to the ranking score, as well as learn complex interactions among features.

3.2 Reformulating Factorization Machine The focus of factorization machine is to model the pairwise interactions between the features. To see the importance of the interaction, one can easily examine the fact that a linear model is not able to make personalized recommendations. Let $w \in \mathcal{R}^d$ be the coefficient of a vector, then we have:

$$w^T x = (w^I)^T x^I + (w^U)^T x^U,$$

where w^I and w^U are corresponding item and user part of the coefficient respectively. Therefore the ranking of different items will be exactly the same for all users. In fact, the two linear terms are modeling item bias and user bias according to their content features. With proper normalization, the two terms will have minimal effects and we thus don't discuss the linear terms in the rest of the paper. We note that all the algorithms follows if such linear terms are added.

In order to model the interactions between the content features, a bilinear model can be used in the modeling process. Given the feature X, the loss of the model can be given by:

$$\mathcal{L}_{\mathrm{BM}}(B) = \sum_i (r_i - x_i^T B x_i)^2, \tag{3.2}$$

where $B \in \mathcal{R}^{d \times d}$ is the matrix modeling the interactions.

An obvious disadvantage of this approach is that when feature size is large, the storage of B and associated computations can be prohibitive. To reduce the complexity of the model, a low-rank structure can be imposed on the matrix $B = UU^T$, where $U \in \mathbf{R}^{d \times r}$, and r is the latent dimension (rank of the bilinear model). The problem now becomes:

$$\mathcal{L}_{\mathrm{FM}}(U) = \sum_i (r_i - x_i^T U U^T x_i)^2 \tag{3.3}$$

and this becomes the factorization machine model [19].

Let $U = [P; Q]$, where $P \in \mathcal{R}^{d_U \times r}$ and $Q \in \mathcal{R}^{d_I \times r}$.

We can expand the model following the representation:

$$\begin{aligned} x^T UU^T x = & \begin{bmatrix} x^U \\ x^I \end{bmatrix}^T \begin{bmatrix} P \\ Q \end{bmatrix} \begin{bmatrix} P \\ Q \end{bmatrix}^T \begin{bmatrix} x^U \\ x^I \end{bmatrix} \\ = & x^U PP^T (x^U)^T + x^I QQ^T (x^I)^T \\ & + 2x^U PQ^T (x^I)^T \end{aligned}$$

where the first two terms are interactions within user features and item features, and may not be predictive because of the same reason as the linear terms. We can thus further reduce the model complexity, and drop the interaction terms within the user features and within item features. Rewriting Eq.(3.3) by replacing the feature x by $[x^U; x^I]$ and removing the within feature interactions, we get:

$$\begin{aligned} (3.4) \qquad & \mathcal{L}_{\mathrm{FM}}(P, Q) \\ = & \sum_{p,q} \left(r_{pq} - \begin{bmatrix} x_p^U \\ x_q^I \end{bmatrix}^T \begin{bmatrix} P \\ 0 \end{bmatrix} \begin{bmatrix} 0 \\ Q \end{bmatrix}^T \begin{bmatrix} x_p^U \\ x_q^I \end{bmatrix} \right)^2 \\ = & \|R - X_U PQ^T X_I^T\|_F^2 \end{aligned}$$

Thus, the learning problem becomes Eq.(3.5) if Frobenious norm is used.

$$(3.5) \qquad \min_{P,Q} \mathcal{L}_{FM}(P, Q) + \frac{1}{2}\lambda_R \|[P, Q]\|_F^2$$

where $\|[P, Q]\|_F^2$ is the regularization term and λ_R is the regularization coefficient.

3.3 Sparse Factorization Machine The model learned by solving Eq.(3.5) computes ranking scores by incorporating all user and item features, and potentially all their interactions. One problem is that when the feature space (the number of features d) is large, the model becomes very complex and also hard to interpret, even P and Q can be limited to low-rank matrices. Since not all features in the models are relevant, and not all the interactions between features are useful, identifying relevant features and thus the interactions only involving relevant features are very important for reducing model complexity and improving the generalization performance. One way to perform the interaction selection is to do embedded feature selection during the learning phase, which identifies important features that are relevant to the model. If a feature does not play a roll in the predictive modeling, all the interactions between this feature and other features will be deactivated. Note that enforcing element-wise sparsity by adding ℓ_1-norm term to P and Q does not help. We thus propose to enforce group sparsity to remove the effect of a feature of user or a feature of item. The objective function becomes:

$$\min_{P,Q} \mathcal{L}(P, Q) + \lambda_1 \|P\|_{1,2} + \lambda_2 \|Q\|_{1,2}$$

where $\|P\|_{1,2} = \sum_{i=1}^{d} \|P_i\|_2$ is the group Lasso, and P_i is the i-th row of the matrix P.

3.4 Optimization Since the variables of the optimization problem is naturally divided into two blocks, i.e., P block and Q block. And we can follow the standard block coordinate descent algorithm with proximal to solve the problem:

Solving P, fixing Q: The objective function for solving P is given by

$$(3.6) \qquad \min_P \|R - X_U PQ^T X_I^T\|_F^2 + \lambda_1 \|P\|_{1,2}$$

We employ proximal gradient descent method to solve the optimization problem. The proximal descent procedure for updating P is:

$$P^{(t+1)} = \mathbf{Prox}^f_{\tau_{(t)}} \left(P^{(t)} - \frac{1}{\tau_{(t)}} \nabla_P \mathcal{L}(P^{(t)}, Q^{(t)}) \right)$$

where $\frac{1}{\tau_{(t)}}$ is the step size, f is the non-smooth function of P: $f(P) = \lambda_1 \|P\|_{1,2}$ and $\nabla_P \mathcal{L}(P, Q) = 2X_U^T (X_U P^T Q X_I^T - R^T) X_I Q^T$. The proximal mapping for group lasso is defined as:

$$(3.7) \qquad \mathbf{Prox}_f(P_i) = \frac{P_i}{\|P_i\|} (\|P_i\| - \lambda_1)_+$$

where the thresholding function is given by $(x)_+ = \max(x, 0)$. To obtain the step size $\tau_{(t)}$, we need to employ line search strategies.

Solving Q, fixing P: Similarly, we can solve the following objective function to get Q.

$$(3.8) \qquad \min_Q \|R - X_U PQ^T X_I^T\|_F^2 + \lambda_2 \|Q\|_{1,2}$$

The proximal descent procedure for updating Q is:

$$Q^{(t+1)} = \mathbf{Prox}^g_{\phi_{(t)}} \left(Q^{(t)} - \frac{1}{\phi_{(t)}} \nabla_Q \mathcal{L}(P^{(t+1)}, Q^{(t)}) \right)$$

where $\frac{1}{\phi_{(t)}}$ is the step size, g is the non-smooth function of Q: $g(Q) = \lambda_2 \|Q\|_{1,2}$ and $\nabla_Q \mathcal{L}(P, Q) = 2X_I^T (X_I Q^T P X_U^T - R^T) X_U P^T$. And $\phi_{(t)}$ is again to be determined by a proper line search procedure.

Proximal alternating linearized minimization Since computation of each block involves expensive line search processes and the algorithm may not scale to large datasets. More recently, the proximal alternating linearized minimization (PALM) has provided an attracting framework to compute alternating optimization [4], given that the Lipschitz constant can be computed analytically for each block. The Lipschitz constants $L_P(Q^{(t)})$ and $L_Q(P^{(t+1)})$ can be analytically

computed and given in Theorem 1 and 2 and we can thus avoid expensive line search. The proof of Theorem 1 and 2 are straightforward so we omit the proof in this paper.

THEOREM 1. *The partial gradient $\nabla_P H(P,Q)$ is Lipschitz continuous with the constant*

$$L_P(Q^{(t)}) \quad = \quad \|X_U^T X_U\| \|Q^{(t)} X_I^T X_I {Q^{(t)}}^T\|$$

The analytical form of Lipschitz constant $L_Q(P^{(t+1)})$ is given in Theorem 2.

THEOREM 2. *The partial gradient $\nabla_Q H(P,Q)$ is Lipschitz continuous with the constant*

$$L_Q(P^{(t)}) \quad = \quad \|X_I^T X_I\| \|P^{(t)} X_U^T X_U (P^{(t)})^T\|$$

Since there is no line search process involved in the algorithm, the algorithm can be very efficient and scale to large-scale datasets. We summarized the proximal alternating linearized minimization algorithm for Sparse Matrix Factorization in Algorithm 1.

Algorithm 1 Proximal alternating linearized minimization algorithm for Sparse Matrix Factorization

INPUT: $\gamma_1 > 1$, $\gamma_2 > 1$, X_U, X_I, R
OUTPUT: Interaction parameter: P^*, Q^*
Set $t = 1$
while true **do**
  set $\tau_{(t)} = \gamma_1 \|X_U^T X_U\| \|Q^{(t)} X_I^T X_I {Q^{(t)}}^T\|$, compute:

$$P^{(t+1)} = \quad \mathbf{Prox}^f_{\tau_{(t)}} \left(P^{(t)} - \frac{1}{\tau_{(t)}} \nabla_P \mathcal{L}(P^{(t)}, Q^{(t)}) \right)$$

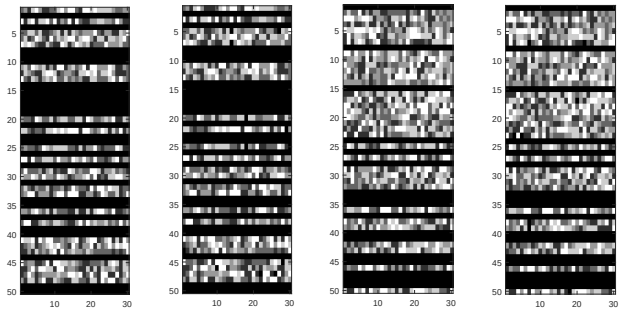
  set $\phi_{(t)} = \gamma_2 \|X_I^T X_I\| \|P^{(t)} X_U^T X_U (P^{(t)})^T\|$, compute:

$$Q^{(t+1)} = \quad \mathbf{Prox}^g_{\phi_{(t)}} \left(Q^{(t)} - \frac{1}{\phi_{(t)}} \nabla_Q \mathcal{L}(P^{(t+1)}, Q^{(t)}) \right)$$

  if meet convergence criteria **then**
    Set $P^* = P^{(t+1)}$, $Q^* = Q^{(t+1)}$
    break
  end if
  Set $t = t + 1$
end while

4 Experiments

In this section, we carry out empirical studies to evaluate the proposed SFM algorithm on the capability of recovering relevant features as well as recommendation performance using real world datasets.



(a) P (b) $\hat{P}$ (c) Q (d) $\hat{Q}$

Figure 1: Feature space recovery on synthetic dataset.

4.1 Recovery of Relevant Features In this subsection, we examine how well the proposed SFM can recover features that are relevant to ranking scores. The sparsity inducing ℓ_1/ℓ_p-norm is used to recover relevant groups of variables from the data. And support recovery property describes the condition under which the relevant variable groups can be recovered. Even though the support recover property of the group sparsity based on ℓ_1/ℓ_p-norm regularization has been studied before [17], in our paper the SFM model solves a non-convex formulation involving two multiplied components regularized by the group sparsity inducing norm. This formulation is very different from traditional group lasso settings. Therefore it is interesting to study if the proposed model can indeed recover the relevant features, given that the underlying data is generated by the interactions of the these relevant features.

To study the support recovery of the proposed SFM, we use a synthetic dataset, in which the ratings are generated using only a subset of original user/item feature spaces. We firstly generate one random matrix X_U for user features and another for item features X_I. We then generate the interaction matrix by multiplying two random matrices PQ^T, in which both P and Q have randomly sparse rows (to simulate irrelevant features). The rating matrix is then given by $R = X_U P Q^T X_I^T + E$, E is a noise matrix and elements of E conforms a normal distribution of $\mathcal{N}(0,1)$. The SFM is then applied to learn the optimal $\hat{P}, \hat{Q}$ from the given rating matrix R and user/item feature matrices. We has shown both the generated matrices and learned matrices in Figure 1. We observe that the proposed SFM can accurately identify irrelevant features from relevant features and thus the interactions captured by $\hat{P}\hat{Q}^T$ only include the interactions between those relevant features. The exceptional results attract us to perform theoretical analysis on the properties of the proposed SFM formulation in our future work.

Table 1: Detail statistics of the datasets used to evaluate the proposed SFM model.

Dataset	# users	# items	# features	# rating	density
AMAZON	13097	11077	5766	175612	0.12%
BX	17219	36545	8946	197520	0.03%
ML	2113	10197	20	855598	4.0%

Table 2: Top-N recall comparison of competing methods on the recommendation task with cold-start items

Cold-start items		Rec@5	Rec@10	Rec@15	Rec@20
Amazon	LAFM-R	0.0134±4.58e-6	0.0115±3.09e-6	0.0106±1.46e-6	0.0096±1.22e-6
	LAFM-L	0.0096±1.21e-6	0.0081±8.47e-7	0.0072±9.84e-7	0.0068±8.64e-7
	CBMF	0.0200±4.15e-7	0.0174±7.33e-7	0.0161±9.46e-7	**0.0149±1.23e-6**
	SFM	**0.0203±3.66e-7**	**0.0176±8.58e-7**	**0.0162±1.02e-6**	0.0148±1.14e-6
BX	LAFM-R	0.0049±2.75e-6	0.0046±7.32e-7	0.0043±7.75e-7	0.0040±4.26e-7
	LAFM-L	0.0030±1.56e-8	0.0031±5.17e-7	0.0032±6.67e-7	0.0034±6.82e-7
	CBMF	0.0055±4.50e-7	0.0054±7.13e-7	0.0050±8.72e-7	0.0046±1.22e-6
	SFM	**0.0065±1.53e-7**	**0.0055±6.48e-7**	**0.0051±8.12e-7**	**0.0047±1.03e-6**
ML	LAFM-R	0.0351±5.72e-4	**0.0730±2.10e-3**	**0.0873±1.40e-3**	**0.0709±8.57e-4**
	LAFM-L	0.0316±5.16e-4	0.0392±4.56e-4	0.0516±6.49e-4	0.0476±1.74e-4
	CBMF	**0.0528±1.70e-3**	0.0480±1.10e-3	0.0472±1.00e-3	0.0434±5.11e-4
	SFM	**0.0528±1.70e-3**	0.0487±1.20e-3	0.0472±1.00e-3	0.0434±5.11e-4

Table 3: Top-N recall comparison of competing methods on the recommendation task of cold-start users

Cold-start users		Rec@5	Rec@10	Rec@15	Rec@20
Amazon	LAFM-R	0.0214±5.34e-7	0.0180±7.07e-7	0.0166±3.48e-7	0.0155±5.99e-7
	LAFM-L	0.0229±2.35e-6	0.0200±2.47e-6	0.0181±6.10e-7	0.0165±4.24e-7
	CBMF	0.0302±1.13e-5	0.0272±1.09e-6	0.0247±1.82e-6	0.0235±7.00e-7
	SFM	**0.0313±4.33e-6**	**0.0281±3.24e-6**	**0.0256±1.91e-6**	**0.0238±4.81e-7**
BX	LAFM-R	0.0375±9.15e-9	0.0251±5.36e-7	0.0195±9.91e-7	0.0167±1.34e-6
	LAFM-L	0.0385±1.15e-6	0.0253±2.17e-7	0.0199±3.32e-7	0.0166±3.21e-7
	CBMF	0.0469±2.86e-7	0.0318±2.76e-8	0.0254±5.89e-7	0.0216±5.65e-7
	SFM	**0.0476±2.55e-6**	**0.0321±5.04e-6**	**0.0255±1.94e-6**	**0.0217±1.95e-6**
ML	LAFM-R	0.0461±8.66e-5	0.0592±7.48e-5	**0.0903±4.33e-4**	**0.0946±1.50e-3**
	LAFM-L	0.0414±1.85e-4	0.0490±5.67e-5	0.0692±9.14e-4	0.0773±2.30e-3
	CBMF	**0.0572±4.44e-4**	0.0790±1.30e-3	0.0827±1.60e-3	0.0763±1.70e-3
	SFM	**0.0572±4.44e-4**	**0.0795±1.30e-3**	0.0827±1.60e-3	0.0763±1.70e-3

4.2 Recommendation Performance In this section, we will study the recommendation performance and compare the proposed SFM with representative methods that exploit content features. In Section 4.2.1 we describe the datasets that will be used in our empirical studies, then we introduce the baselines methods used for comparison in Section 4.2.2, and finally we present the comparison results in Section 4.2.4.

4.2.1 Dataset Description A set of three popular benchmark datasets are used to evaluate the performance of SFM, including Amazon Books, Book Crossing and MovieLens.

Amazon Book (AMAZON) is a dataset which collects the best-selling books and their ratings from Amazon. The ratings are ranging from 1 to 5. The rating is 0 if it is missing or no user rated that book. The item feature is collected from the description of the book. Book Crossing (BX) is extracted from the Book Crossing dataset [31]. The features of these books are extracted from Amazon. The rating for BX is from 1 to 9. MovieLens (ML) is extracted from the MovieLens-1M datasets [1]. Similar to AMAZON dataset, the ratings are ranging from 1 to 5. The features for the items are generated from the genres of the movies. In order to

[1] http://www.movielens.org

Table 4: Top-N recall comparison of competing methods on the recommendation task with both cold-start users and items scenario

Cold-start users and items		Rec@5	Rec@10	Rec@15	Rec@20
Amazon	LAFM-R	0.0140±3.00e-7	0.0122±5.15e-7	0.0113±2.19e-7	0.0105±1.07e-6
	LAFM-L	0.0097±1.09e-7	0.0084±4.13e-7	0.0074±6.16e-7	0.0067±3.95e-7
	CBMF	0.0207±1.28e-6	0.0190±4.03e-6	0.0175±7.55e-7	0.0161±3.53e-7
	SFM	**0.0211±3.25e-6**	**0.0189±3.63e-6**	**0.0172±5.88e-7**	**0.0161±5.38e-7**
BX	LAFM-R	0.0055±6.50e-7	0.0051±1.24e-8	0.0043±9.62e-8	0.0041±2.64e-7
	LAFM-L	0.0037±4.88e-7	0.0036±1.83e-7	0.0035±4.93e-7	0.0034±7.89e-7
	CBMF	0.0069±1.35e-6	0.0058±4.48e-7	**0.0052±7.51e-7**	0.0046±2.87e-7
	SFM	**0.0072±1.07e-6**	**0.0059±3.60e-7**	**0.0052±7.31e-7**	**0.0047±3.32e-7**
ML	LAFM-R	0.0373±9.94e-4	**0.0728±2.20e-3**	**0.0893±1.70e-3**	**0.0720±9.86e-4**
	LAFM-L	0.0316±7.69e-4	0.0390±7.67e-4	0.0511±1.11e-3	0.0474±4.28e-4
	CBMF	0.0515±1.70e-3	0.0469±9.14e-4	0.0471±1.11e-3	0.0429±6.36e-4
	SFM	**0.0521±1.70e-3**	0.0477±9.66e-5	0.0471±1.11e-3	0.0429±6.44e-4

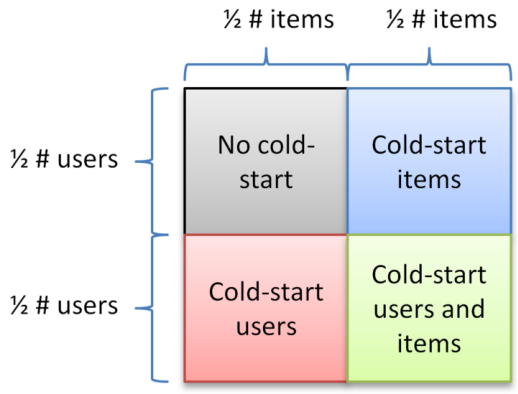


Figure 2: Illustration of the partitioning of the dataset

generate item features from item descriptions such as in AMAZON, we have adopted standard natural language processing procedures. We firstly removed stop words, words appearing in less than 20 items, and the words appearing in more than 20% of the items. We then calculate the TF-IDF values for the remaining words as the feature of the items.

The user features are not available in all these datasets. We thus follow the conventional approach to generate the user features by aggregating the item features from items that are rated by the user. The aggregated user feature vector describes the user's preference via the item feature space. In typical commercial recommender system, the number of ratings in the whole dataset is very limited, and however the datasets we have in hand is considerably dense. We would like to evaluate the methods in a more realistic recommender system setting, and we thus proceed to sparsify the dataset to reduce the number of ratings by subsampling the ratings from the original rating set. In the experiments, the subsampling rate is 50%. The details of the 4 processed datasets are given in Table 1.

4.2.2 Competing Approaches The competing methods include the representative methods that exploit content information in different ways, and are given below:

- Learning Attribute-to-Feature Mappings with Ridge regression (LAFM-R) [12]: This method combines the matrix factorization and content-based method to consider both the historical records and the cold-start items. First it will factorize the rating matrix into a user factor U and an item factor V. Then the method learns a mapping function from the user features X_U to U and a mapping function from the item features X_I to V. For cold-start users with feature vector x_u or items with feature vector x_i, first the mapping functions are applied on x_u or x_i to get the factor U_u or V_i, then the ratings can be calculated from the learned user or item factors by recovering the partial matrix for the cold-start users or items. In this baseline, the mapping function is using a Ridge regression.

- Learning Attribute-to-Feature Mappings with Lasso regression (LAFM-L): Unlike LAFM-R, in this baseline, we use Lasso regression to learn the mapping from features to the factorized factors.

- Content-boosted Matrix Factorization (CBMF) [11]: This method is similar to the above two baselines with two differences: one is that it only learns a mapping from the features of items to one of the factorization terms; another is that it learns the factorization and the mapping function simultaneously. To be specific, the ranking matrix R is assumed to be $R = UV^T$ where $V = XW$ and X is the feature matrix for items. In our case, we need to modify the assumption so that we can adopt the user features into the formulation. To do so, we

assume the $U = X_U W_U$ and $V = X_I W_I$, where $W_U \in \mathcal{R}^{d_U \times k}$ and $W_I \in \mathcal{R}^{d_I \times k}$. We add Frobenius norm of W_U and W_I as the regularizer to avoid overfitting. The objective function is given by

$$\min_{W_U, W_I} \|R - X_U W_U W_I^T X_I^T\|_F^2 + \lambda(\|W_U\|_F^2 + \|W_I\|_F^2) \tag{4.9}$$

Note that this baseline is equivalent to an improved version of factorization machine (FM) [20], as shown in the previous sections.

4.2.3 Recommendation Tasks and Evaluation

We evaluate the competing algorithms on three different recommendation tasks: 1) recommendation involving only cold-start items, 2) recommendation involving only cold-start user, and 3) recommendation involving both cold-start users and cold-start items. We thus partition our rating matrix into four parts, where one of the partitions works as training data, one of the partitions includes testing data for cold start items, one of the partitions includes testing data for cold start users, and one of the partitions include testing data for cold start with new users and new items. Figure 2 illustrates the idea of such partitioning. We note that the partition is performed in a random way. The selection of parameters of the models used an independent validation dataset.

For each task, we learn models on the training data, and the testing is done by firstly ranking the items to be recommended according to the scores produced by the models. We then use the top-N recommendation evaluation metric to compute the recall at n ($Rec@n$), defined as follows:

$$Rec@n = \frac{\#\text{items liked by user in top-n items}}{n}$$

The metric measures on average the percentage of items preferred by a user in the top n item set from the recommended list provided by the recommender system. We report the results for $Rec@5$, $Rec@10$, $Rec@15$ and $Rec@20$ in this paper.

4.2.4 Experimental Results

We perform the methods on three different recommendation tasks and the results are shown in Table 2, 3 and 4, respectively. Overall, the proposed SFM is always among the best performance in all three task compared with other competing methods. In the Amazon and BX datasets, the SFM can always outperform other competing methods. However, we notice that in the ML dataset, the proposed SFM cannot outperform baselines when N is larger than 5. This may relate to the limited features space of the user/item content features. Indeed feature selection is mainly design to handle large scale feature space, and are not ideal for cases where only a small feature space present. Sharing the similar spirit, the proposed SFM is designed to perform feature selection and thus interaction selection, and may not perform well with small feature space.

5 Conclusions

The factorization machine is a powerful tool designed to tackle the cold-start problems, exploiting the interactions with such content information. A factorization machine makes use of all possible pair-wise interactions within the content information to make recommendations, many of which are not interactions between relevant features that are predictive of recommendations. In this paper, we propose an efficient Sparse Factorization Machine (SFM), that simultaneously identifies predictive user features and item features, models synergistic interactions between these relevant features, and learns a bilinear model using only these interactions. We have carried out extensive empirical studies on both synthetic and real-world datasets, and compared our method to other state-of-the-art baselines, including Factorization Machine. Experimental results show that SFM can greatly outperform other baselines.

6 Acknowledgments

This research is partially supported by Office of Naval Research through grant N00014-14-1-0631.

References

[1] G. Adomavicius and A. Tuzhilin. Toward the next generation of recommender systems: A survey of the state-of-the-art and possible extensions. *IEEE Trans. on Knowl. and Data Eng.*, 17(6):734–749, June 2005.

[2] D. Agarwal and B.-C. 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, 2009.

[3] D. Agarwal and B.-C. Chen. flda: Matrix factorization through latent dirichlet allocation. In *Proceedings of the Third ACM International Conference on Web Search and Data Mining*, WSDM '10, pages 91–100, 2010.

[4] J. Bolte, S. Sabach, and M. Teboulle. Proximal alternating linearized minimization for nonconvex and nonsmooth problems. *Mathematical Programming*, 146(1-2):459–494, 2014.

[5] H. Borges and A. Lorena. A survey on recommender systems for news data. In E. Szczerbicki and N. Nguyen, editors, *Smart Information and Knowledge Management*, volume 260 of *Studies in Computational Intelligence*, pages 129–151. Springer Berlin Heidelberg, 2010.

[6] L. Brozovsky and V. Petricek. Recommender system for online dating service. In *Proceedings of Conference Znalosti 2007*, Ostrava, 2007. VSB.

[7] N. Chang, M. Irvan, and T. Terano. A {TV} program recommender framework. *Procedia Computer Science*, 22(0):561 – 570, 2013. 17th International Conference in Knowledge Based and Intelligent Information and Engineering Systems - {KES2013}.

[8] S. Chang, G. Qi, C. C. Aggarwal, J. Zhou, M. Wang, and T. S. Huang. Factorized similarity learning in networks. In *2014 IEEE International Conference on Data Mining, ICDM 2014*, pages 60–69, 2014.

[9] S. Chang, J. Zhou, P. Chubak, J. Hu, and T. S. Huang. A space alignment method for cold-start TV show recommendations. In *Proceedings of the Twenty-Fourth International Joint Conference on Artificial Intelligence, IJCAI 2015*, pages 3373–3379, 2015.

[10] C. Cheng, F. Xia, T. Zhang, I. King, and M. R. Lyu. Gradient boosting factorization machines. In *Proceedings of the 8th ACM Conference on Recommender systems*, pages 265–272. ACM, 2014.

[11] P. Forbes and M. Zhu. Content-boosted matrix factorization for recommender systems: experiments with recipe recommendation. In *Proceedings of the fifth ACM conference on Recommender systems*, pages 261–264. ACM, 2011.

[12] Z. Gantner, L. Drumond, C. Freudenthaler, S. Rendle, and L. Schmidt-Thieme. Learning attribute-to-feature mappings for cold-start recommendations. In *Data Mining (ICDM), 2010 IEEE 10th International Conference on*, pages 176–185. IEEE, 2010.

[13] L. Hong, A. S. Doumith, and B. D. Davison. Co-factorization machines: modeling user interests and predicting individual decisions in twitter. In *Proceedings of the sixth ACM international conference on Web search and data mining*, pages 557–566. ACM, 2013.

[14] P. Knees and M. Schedl. A survey of music similarity and recommendation from music context data. *ACM Trans. Multimedia Comput. Commun. Appl.*, 10(1):2:1–2:21, Dec. 2013.

[15] Y. Koren. Factor in the neighbors: Scalable and accurate collaborative filtering. *ACM Trans. Knowl. Discov. Data*, 4(1), Jan. 2010.

[16] B. Loni, Y. Shi, M. Larson, and A. Hanjalic. Cross-domain collaborative filtering with factorization machines. In *Advances in Information Retrieval*, pages 656–661. Springer, 2014.

[17] G. Obozinski, M. J. Wainwright, and M. I. Jordan. Support union recovery in high-dimensional multivariate regression. *The Annals of Statistics*, pages 1–47, 2011.

[18] A. M. Rashid, G. Karypis, and J. Riedl. Learning preferences of new users in recommender systems: An information theoretic approach. *SIGKDD Explor. Newsl.*, 10(2):90–100, Dec. 2008.

[19] S. Rendle. Factorization machines. In *Data Mining (ICDM), 2010 IEEE 10th International Conference on*, pages 995–1000. IEEE, 2010.

[20] S. Rendle. Factorization machines with libfm. *ACM Transactions on Intelligent Systems and Technology (TIST)*, 3(3):57, 2012.

[21] S. Rendle, Z. Gantner, C. Freudenthaler, and L. Schmidt-Thieme. Fast context-aware recommendations with factorization machines. In *Proceedings of the 34th international ACM SIGIR conference on Research and development in Information Retrieval*, pages 635–644. ACM, 2011.

[22] P. Resnick, N. Iacovou, M. Suchak, P. Bergstrom, and J. Riedl. Grouplens: An open architecture for collaborative filtering of netnews. Working paper series, MIT Center for Coordination Science, 1994.

[23] J. B. Schafer, D. Frankowski, J. Herlocker, and S. Sen. The adaptive web. chapter Collaborative Filtering Recommender Systems, pages 291–324. Springer-Verlag, Berlin, Heidelberg, 2007.

[24] H. Shan and A. Banerjee. Generalized probabilistic matrix factorizations for collaborative filtering. In *Data Mining (ICDM), 2010 IEEE 10th International Conference on*, pages 1025–1030, 2010.

[25] M. Sharma, J. Zhou, J. Hu, and G. Karypis. Feature-based factorized bilinear similarity model for cold-start top-n item recommendation. In *Proceedings of the 15th SIAM International Conference on Data Mining*, SDM '15, pages 190–198, 2015.

[26] K. Sugiyama and M.-Y. Kan. Exploiting potential citation papers in scholarly paper recommendation. In *Proceedings of the 13th ACM/IEEE-CS Joint Conference on Digital Libraries*, JCDL '13, pages 153–162, New York, NY, USA, 2013. ACM.

[27] H.-F. Yu, C.-J. Hsieh, S. Si, and I. S. Dhillon. Parallel matrix factorization for recommender systems. *Knowl. Inf. Syst.*, 41(3):793–819, Dec. 2014.

[28] S. Zhang, W. Wang, J. Ford, and F. Makedon. Learning from incomplete ratings using non-negative matrix factorization. In *In Proc. of the 6th SIAM Conference on Data Mining (SDM*, pages 549–553, 1996.

[29] Y. Zhang, M. Zhang, Y. Liu, S. Ma, and S. Feng. Localized matrix factorization for recommendation based on matrix block diagonal forms. In *Proceedings of the 22Nd International Conference on World Wide Web*, WWW '13, pages 1511–1520, 2013.

[30] C.-N. Ziegler, S. M. McNee, J. A. Konstan, and G. 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.

[31] C.-N. Ziegler, S. M. McNee, J. A. Konstan, and G. Lausen. Improving recommendation lists through topic diversification. In *Proceedings of the 14th international conference on World Wide Web*, pages 22–32. ACM, 2005.