Enabling Kernel-based Attribute-aware Matrix Factorization for Rating Prediction

Jia-Dong Zhang, Chi-Yin Chow, *Member, IEEE*, and Jin Xu, *Member, IEEE*

Abstract—In recommender systems, one key task is to predict the personalized rating of a user to a new item and then return the new items having the top predicted ratings to the user. Recommender systems usually apply collaborative filtering techniques (e.g., matrix factorization) over a sparse user-item rating matrix to make rating prediction. However, the collaborative filtering techniques are severely affected by *the data sparsity of the underlying user-item rating matrix* and often confront *the cold-start problems for new items and users*. Since the attributes of items and social links between users become increasingly accessible in the Internet, this paper exploits the rich attributes of items and social links of users to alleviate the rating sparsity effect and tackle the cold-start problems. Specifically, we first propose a **K**ernel-based **A**ttribute-aware **M**atrix **F**actorization model called **KAMF** to integrate the attribute information of items into matrix factorization. KAMF can discover the nonlinear interactions among attributes, users and items, which mitigate the rating sparsity effect and deal with the cold-start problem for new items by nature. Further, we extend KAMF to address the cold-start problem for new users by utilizing the social links between users. Finally, we conduct a comprehensive performance evaluation for KAMF using two large-scale real-world data sets recently released in Yelp and MovieLens. Experimental results show that KAMF achieves significantly superior performance against other state-of-the-art rating prediction techniques.

Index Terms—Rating prediction, matrix factorization, attribute-aware, kernel trick, incremental learning.

1 Introduction

Nowadays vast amounts of information are created and accessible in the Internet including portal websites (e.g., Yahoo and Sina), e-commerce websites (e.g., Amazon and Alibaba), social media (e.g., Facebook and Twitter), and location-based social networks (LBSNs) (e.g., Foursquare and Yelp). The presence of too much information causes information overload, i.e., the difficulty for a person to find interesting information. To overcome this difficulty, recommender systems have been well studied by both the academia and industry in the past decades. Specifically, a variety of recommender systems have been developed to match users (e.g., consumers) with their preferred items (e.g., products or services). Recommender systems have been widely used in the portal websites for recommending news articles, e-commerce websites for recommending products (e.g., books, cell phones, and laptops), social media for recommending services (e.g., songs, movies, and friends), and LBSNs for recommending venues (e.g., restaurants, stores, and museums).

In recommender systems, given a sparse user-item rating matrix with a large amount of unknown ratings as depicted in Fig. 1, the essential task is to predict the rating of a user to a new item that has not been rated by the user, called the **personalized item rating prediction**, and then return the new items with the top ratings for the user. Most recommender systems apply matrix factorization methods [1], [2] over a sparse user-item rating matrix, because they perform efficiently in dealing with large data sets. However, the matrix factorization methods may suffer from the data sparsity problem due to the large amount of unknown ratings in the sparse user-item rating matrix, because in practice users only rated a very small proportion of items. In particular, they may also encounter the cold-start problems, namely, *recommending new items to users* or *recommending items to new users*, in which new items or users are only attached with few ratings.

		Items						
		v_1	v_2	v_3	$\cdots$	v_{N-2}	v_{N-1}	v_N
Users	u_1	5	?	?	$\cdots$	?	1	?
	u_2	?	3	?	$\cdots$	2	?	?
	$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\ddots$	$\vdots$	$\vdots$	$\vdots$
	u_{M-1}	?	3	?	$\cdots$	?	2	?
	u_M	1	?	?	$\cdots$	?	?	?

Fig. 1. A sparse user-item rating matrix, where each "?" denotes an unknown rating and the goal of rating prediction is to uncover these unknown ratings.

Fortunately, it is increasingly accessible to the attributes of items (note that the attributes of users are often unavailable due to the privacy issue) and social links between users. For example, in LBSNs like Yelp, a venue is described by a large number of attributes, e.g., categories (restaurants, stores, and museums), geographical locations (latitude and longitude coordinates), noise level, and ambience; and users can build social links with their friends to share the experiences of visiting interesting venues. The rich attributes of items and social links of users bring us new opportunities to alleviate the data sparsity effect of user-item rating matrices, especially to tackle the cold-start problems for new items and users.

One may consider integrating the rich attribute information with a sparse user-item rating matrix by extending matrix factorization methods into factorization machines [3], [4], [5], [6], in

- *J.-D. Zhang and C.-Y. Chow are with Department of Computer Science, City University of Hong Kong, Hong Kong. E-mail: jzhang26-c@my.cityu.edu.hk, chiychow@cityu.edu.hk.*
- *J. Xu is with Department of Management Information Systems and Electronic Commerce, School of Economics & Management, Southwest Jiaotong University, Chengdu, 610031, China. E-mail: xj_james@163.com.*

which besides *users* and *items*, each *continuous attribute* or *possible value of a discrete attribute* is associated with a latent factor vector and the interaction of every pair of latent factor vectors is utilized for rating prediction. Nonetheless, **the factorization machines contain three major limitations**: (1) **Unscalable to high dimension of attributes.** They were initially developed for incorporating at most *tens of context attributes* and are not scalable to *thousands of item attributes* because they enumerate the interaction of every pair of attributes. (2) **Two-attribute interaction.** In the factorization machines, an interaction is limited to two attributes, i.e., a pair of attributes, but in reality more than two attributes may interact with each other. (3) **Linear interaction between attributes.** The interaction between every pair of attributes is modeled as the linear combination of their latent factor vectors, but it is unrealistic to assume that the interaction between any two attributes is always linear.

To address the above-mentioned three limitations, in this study we propose a kernel-based regression model to seamlessly integrate the attribute information of items into matrix factorization for the personalized item rating prediction, called **Kernel-based Attribute-aware Matrix Factorization**, or **KAMF** for short. (1) The proposed regression model adaptively learns the most potential interactions among two or more attributes instead of enumerating the interaction of every pair of attributes. Therefore, our regression model can be scalable to high-dimension of attributes and uncover the interaction involving more than two attributes. (2) The proposed regression model leverages the kernel trick to turn a linear model into a nonlinear model. The kernel trick enables our learning algorithm to learn a nonlinear model without requiring the explicit feature (attribute) mapping. Note that the kernel trick is also used by the well-known support vector machines [7].

The main contributions of this study are listed below:

- We propose a kernel-based attribute-aware matrix factorization model, called KAMF for personalized item rating prediction. KAMF combines the attribute information of items with a sparse user-item rating matrix, which greatly mitigates the data sparsity effect of the user-item rating matrix. KAMF can uncover the potential nonlinear interactions among users, items and attributes, and thus considerably uplift the predictive ability on the ratings of users to items. We also develop an incremental learning algorithm to estimate the model parameters of KAMF. (Section 3)
- KAMF can intrinsically deal with the cold-start problem for new items due to the use of their attributes. Further, we devise a sampling-based approach to extend KAMF in order to tackle the cold-start problem for new users through taking full advantage of the social links between users. (Section 4)
- Extensive experiments are conducted to evaluate the performance of KAMF using two large-scale real-world data sets that were released in Yelp and MovieLens recently. Experimental results show that KAMF significantly outperforms other state-of-the-art rating prediction techniques. (Section 5)

The rest of this paper is organized as follows. The related work is highlighted in Section 2. We present the kernel-based attribute-aware matrix factorization in Section 3, followed by its extension for the cold-start problems in Section 4. The experimental evaluations and results are reported in Section 5. Finally, Section 6 concludes this paper.

2 Related Work

This section briefly reviews the recent advances on rating prediction in recommender systems. The recommender systems apply knowledge discovery techniques to predict the ratings of users on products or services so as to make personalized recommendations for the users. The techniques on rating prediction can be generally classified into three main categories: *collaborative filtering*, *content-based*, and *context-aware*.

Collaborative filtering techniques. Among the three categories, collaborative filtering (CF) is the most popular technique used for rating prediction, because it only requires a user-item rating matrix as the input and the implicit or explicit ratings are widely available in recommender systems. The CF techniques can be divided into *memory-based CF* and *model-based CF*. The memory-based methods include user-based CF [8] and item-based CF [9]. The user-based CF predicts the rating of a user on an item based on the ratings of similar users giving to the item, while the item-based CF estimates the rating of a user on an item based on users' ratings of similar items. Here the key task is to compute the similarity of all pairs of users or items using the user-item rating matrix with some selected similarity measurement such as cosine similarity or Pearson correlation. Recently, the similarity between users can be derived by utilizing their social links in the user-based CF, also known as friend-based CF [10].

As an alternative approach of CF, latent factor models aim to uncover the latent factor vectors of users and items to explain observed ratings and predict unobserved ratings. Some of the most successful realizations of latent factor models are based on matrix factorization [2]. For example, the probabilistic matrix factorization minimizes the regularized squared error on the set of known ratings [11], [12], [13], and the nonnegative matrix factorization imposes the constraint of nonnegativity on the latent factors of users and items [14], [15], [16]. Further, plenty of studies [17], [18], [19], [20], [21], [22], [23] augment probabilistic matrix factorization with a social regularization term using social links between users [24], while the work [1] develops an extended singular value decomposition (SVD++) to complement the latent factor vectors of users using their implicit feedback, e.g., the purchase history or browsing history of users. The latent factors of matrix factorization models on massive data can be learned through distributed stochastic gradient descent or multiplicative update [25].

Content-based techniques. A simple way using attribute information is based on the content-based techniques. They utilize the attributes or features of items rated by a user to profile the user using classification models, e.g., user-specific feature-based similarity models [26] and support vector machines [20]. The rating of a user to an item can be predicted in terms of the similarity between the user profile (i.e., her classification model) and the item contents (i.e., attributes or features). Clearly, the content-based techniques suffer the limitation of making inaccurate prediction to users with very few ratings, because they only use the past preference of the user to make rating predictions for that user and ignore the past preferences of other users [26]. As a result, a certain user's ratings may not be sufficient to build a reliable classification model. This is one of the reasons that the content-based techniques have received much less attention than CF techniques from both the academia and industry. The content-based techniques are usually applied in social media recommendations, in which the item attributes can be extracted from item contents,

web pages, news articles, or user reviews [27], [28].

Context-aware techniques. The context-aware techniques integrate the CF techniques with the context information, including static context (e.g, attributes or features of users and items) and dynamic context attached to rating events (e.g., time, location, and texts). Some work [29], [30] simply employs the contextual information for pre-processing or post-processing, in which the context is used to guide the selection of training data or recommendation results. Most studies [31], [32], [33], [34], [35], [36] incorporate context variables into latent factor models. For instance, the literatures [31], [32] combine matrix factorization with texts to derive the latent word topics of texts, users and venues for predicting review helpfulness or recommending venues. Further, the work [37] extends matrix factorization to incorporate the contexts of venues such as neighbors, categories, popularity and geographical locations. This method shows good performance on rating prediction for venues (i.e., items) in location-based social networks (LBSNs), but it is tailored to predict the ratings of venues. More sophisticatedly, the generic factorization machines [3], [4], [5], [6] attach a latent vector to each user, item and context, and then model the interaction between each pair of latent factor vectors of users, items and contexts for rating prediction. However, they usually suffer from three major limitations including inability of scaling up to high dimension of attributes, two-attribute interaction, and linear interaction between attributes, as discussed in Section 1.

Our previous works. To predict the overall ratings of users on venues (i.e., items) in LBSNs, we have developed techniques that utilize the static attributes of venues (e.g., geographical locations and categories [38], [39], [40], [41]) and dynamic attributes/contexts attached to rating events (e.g., time and current location [42], [43]). In these studies [38], [39], [40], [41], [42], [43], we have two important findings: (1) In reality, the preferences of users on items are highly correlated to the attributes of items or dynamic attributes. (2) It is a good idea to model the nonlinear interactions of attributes and combine them into matrix factorization to uplift the predictive ability of rating prediction models. To this end, this study proposes a kernel-based regression model to seamlessly integrate the attribute information of items into matrix factorization for personalized item rating prediction. The proposed model adaptively learns the most potential nonlinear interactions of multiple attributes and greatly alleviates the data sparsity effect of the user-item rating matrix. We can distinguish this paper from our previous works in terms of the following two facets: (1) The proposed model in this paper is generic and can exploit all attributes of items. In contrast, the methods in our previous works are dedicated to utilizing some specific attributes, e.g., the geographical locations of venues. As a result, they are not suitable for recommending the items without such attributes. (2) These methods employ each attribute individually and cannot deal with the interaction between attributes which is the main focus of this paper.

3 Kernel-based Attribute-aware Matrix Factorization

In this section, we define the problem in Section 3.1 and introduce the probabilistic matrix factorization in Section 3.2. Then, we describe our attribute-aware matrix factorization, kernel-based nonlinear model, and incremental learning algorithm in Sections 3.3, 3.4 and 3.5, respectively.

TABLE 1
Key Notations in the Paper

Sym.	Meaning
	Constants
u_i	Some user
v_j	Some item
M	Number of users
N	Number of items
L	Dimension of a latent factor vector for a user or item
D	Dimension of a attribute vector for an item
N_{u_i}	Number of items rated by user u_i
n_{u_i}	Number of training samples of user u_i
m_{u_i}	Number of items rated by users u_i and sampled from her friends ($m_{u_i} \geq N_{u_i}$)
λ	Regularization constant
η	Learning rate constant
β	Decay rate constant
	Input data
$\mathbf{R}$	Sparse user-item rating matrix, i.e., $\mathbf{R} \in \mathbb{R}^{M \times N}$
r_{ij}	Known rating of user u_i on item v_j
$\hat{r}_{ij}$	Predicted rating of user u_i on item v_j
e_{ij}	Error, i.e., $e_{ij} = r_{ij} - \hat{r}_{ij}$
$\mathbf{X}$	Item attribute matrix, i.e., $\mathbf{X} \in \mathbb{R}^{D \times N}$
$\mathbf{x}_j$	The jth column of $\mathbf{X}$ or attribute vector of item v_j
	Model parameters
$\mathbf{U}$	User latent factor matrix, i.e., $\mathbf{U} \in \mathbb{R}^{L \times M}$
$\mathbf{u}_i$	The ith column of $\mathbf{U}$ or latent factor vector of user u_i
$\mathbf{V}$	Item latent factor matrix, i.e., $\mathbf{V} \in \mathbb{R}^{L \times N}$
$\mathbf{v}_j$	The jth column of $\mathbf{V}$ or latent factor vector of item v_j
$\mathbf{W}$	**Implicit coefficient** matrix, i.e., $\mathbf{W} \in \mathbb{R}^{D \times M}$
$\mathbf{w}_i$	The ith column of $\mathbf{W}$ or coefficient vector for user u_i
$\mathbf{b}$	Bias vector, i.e., $\mathbf{b} \in \mathbb{R}^M$
b_i	The ith element of $\mathbf{b}$ or bias for user u_i
α_{ij}	**Explicit coefficient** associated user u_i and item v_j
	Functions
$\phi(\mathbf{x})$	Nonlinear mapping function on the attribute vector $\mathbf{x}$
$\mathcal{F}$	Optimizing objective function
$\mathcal{K}$	Kernel function
$\mathcal{I}$	Indicator function

3.1 Problem Statement

This section defines the data and problem of this work. For the sake of clarity, TABLE 1 lists the key notations used in this paper.

Definition 1 (**Sparse user-item rating matrix).** Let $\mathbf{R} \in \mathbb{R}^{M \times N}$ be a sparse user-item rating matrix, in which M and N are the total number of users and items in a recommender system, respectively. An entry r_{ij} denotes the rating of user u_i to item v_j and most ratings are unknown because a user only interacts with a small proportion of items in the recommender system. Note that only the non-zero elements of $\mathbf{R}$ are considered.

Definition 2 (**Item attribute matrix).** Let $\mathbf{X} \in \mathbb{R}^{D \times N}$ be an item attribute matrix, in which a column $\mathbf{x}_j$ denotes the attribute vector of item v_j, and D denotes the dimension of attributes. Note that an attribute may be continuous, discrete, or nominal.

Definition 3 (**Research problem).** Given sparse user-item rating matrix $\mathbf{R}$ and item attribute matrix $\mathbf{X}$, the goal is to predict the rating of a user to any new items, i.e., estimating the unknown ratings of the sparse user-item rating matrix $\mathbf{R}$.

3.2 Matrix Factorization

Matrix factorization is one of the most popular methods on rating prediction for recommender systems. The basic idea is to find two low-rank factor matrices $\mathbf{U} \in \mathbb{R}^{L \times M}$ and $\mathbf{V} \in \mathbb{R}^{L \times N}$ ($L \ll M, N$) such that $\mathbf{R} \approx \mathbf{U}^{\mathrm{T}}\mathbf{V}$; each column vectors $\mathbf{u}_i$ and $\mathbf{v}_j$ in $\mathbf{U}$ and $\mathbf{V}$ represent the user-specific and item-specific latent factor vectors, respectively. The famous probabilistic matrix

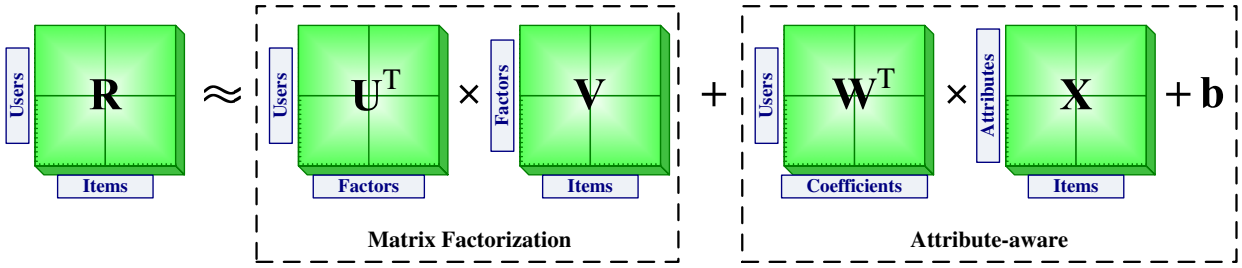


Fig. 2. Attribute-aware matrix factorization

factorization [12] obtains the matrices $\mathbf{U}$ and $\mathbf{V}$ by minimizing the sum-of-squared-errors objective function with quadratic regularization terms on the set of **known ratings (i.e., non-zero elements)** r_{ij} in the sparse user-item rating matrix $\mathbf{R}$, given by

$$\min_{\mathbf{U},\mathbf{V}} \frac{1}{2} \sum_{r_{ij} \in \mathbf{R}} \left(r_{ij} - \mathbf{u}_i^{\mathrm{T}} \mathbf{v}_j\right)^2 + \frac{\lambda_1}{2} \sum_{i=1}^{M} \|\mathbf{u}_i\|^2 + \frac{\lambda_2}{2} \sum_{j=1}^{N} \|\mathbf{v}_j\|^2, \tag{1}$$

where $||\cdot||$ denotes the Euclidean norm, λ_1 and λ_2 are regularization parameters to avoid overfitting.

3.3 Attribute Awareness

Matrix factorization methods severely depend on the sparse user-item rating matrix with a large amount of unknown ratings and usually suffer from the data sparsity problem. To alleviate the data sparsity effect of the user-item rating matrix, this study takes full advantage of the rich attribute information of items. In reality, the preferences of users on items are highly correlated to the attributes of items, as shown in our recent papers [38], [39], [40], [41], [42], [43]. For instance, in location-based social networks like Yelp, on the category (i.e., one attribute) of venues (i.e., items), some users would like to check in restaurants for food while others prefer to visit tourist attractions for fun; on the geographical location, indoorsy persons like visiting venues around their living areas (home and office) while outdoorsy persons prefer traveling around the world to explore new venues; some people care about the noise level of venues while others are insensitive to noise. Therefore, the attribute-aware preferences of users on items complement the sparse ratings and bring benefit for rating prediction. In this work, we focus on using the attributes of items because they are more widely available than the attributes of users, but our proposed model can utilize the attributes of users in the same way.

Attribute-aware matrix factorization. To this end, we exploit a regression model to derive the preferences of users to the attributes of items. In the regression model, the attribute-aware preference of each user u_i is represented as an attribute-aware coefficient or parameter vector $\mathbf{w}_i$ and bias b_i; then the attribute-aware preference of user u_i to item v_j is estimated by linearly combining the attribute vector $\mathbf{x}_j$ of item v_j with the coefficient vector $\mathbf{w}_i$ and bias b_i, i.e., $\mathbf{w}_i^{\mathrm{T}} \mathbf{x}_j + b_i$ (note that in Section 3.4 we will turn the linear model into a nonlinear model based on the kernel trick). Thus, we devise an attribute-aware matrix factorization to predict the rating $\hat{r}_{ij}$ of user u_i to item v_j based on *latent factor vectors* and *attribute-aware coefficient vectors*, as depicted in Fig. 2, given by

$$\hat{r}_{ij} = \mathbf{u}_i^{\mathrm{T}} \mathbf{v}_j + \mathbf{w}_i^{\mathrm{T}} \mathbf{x}_j + b_i. \tag{2}$$

In Fig. 2 or Equation (2), there is no need to put an extra parameter to balance the weight between matrix factorization and attribute-aware regression, because the parameter will be absorbed into $\mathbf{u}_i, \mathbf{v}_j, \mathbf{w}_i$, and b_i. The devised model in Equation (2) can intrinsically mitigate the data sparsity effect of new items rated by few users. For example, given a new item v_j that gets few ratings from users, the rating $\hat{r}_{ij}$ for new user u_i can be predicted through combining $\mathbf{w}_i$ and $\mathbf{x}_j$, in which $\mathbf{w}_i$ is learned from other "less-sparse" items with the attribute values similar to $\mathbf{x}_j$. However, this model cannot deal with the data sparsity problem caused by the new users who have rated few items. In Section 4, this model will be extended to address this problem through resorting to other "less-sparse" users in terms of the social links between users.

Parameter learning. Both matrix factorization and attribute-aware regression are tightly interdependent on each other. They are required to collaborate for predicting the ratings of users on items and need to be learned from data at the same time. In order to obtain the model parameters $\{\mathbf{u}_i, \mathbf{v}_j, \mathbf{w}_i, b_i\}$ in Equation (2) for all users u_i and items v_j, we can minimize the regularized objective function on the set of **known ratings (i.e., non-zero elements)** r_{ij} of the sparse user-item rating matrix $\mathbf{R}$ as follows:

$$\min_{\mathbf{U},\mathbf{V},\mathbf{W},\mathbf{b}} \mathcal{F} = \frac{1}{2} \sum_{r_{ij} \in \mathbf{R}} \left(r_{ij} - \mathbf{u}_i^{\mathrm{T}} \mathbf{v}_j - \mathbf{w}_i^{\mathrm{T}} \mathbf{x}_j - b_i\right)^2 + \frac{\lambda_1}{2} \sum_{i=1}^{M} \|\mathbf{u}_i\|^2 + \frac{\lambda_2}{2} \sum_{j=1}^{N} \|\mathbf{v}_j\|^2 + \frac{\lambda_3}{2} \sum_{i=1}^{M} \|\mathbf{w}_i\|^2. \tag{3}$$

In this paper, we apply stochastic gradient descent [44] to solve the optimization function in Equation (3), following previous studies [1], [37]. Specifically, we randomly loop through all known ratings in the sparse user-item rating matrix $\mathbf{R}$. For each known rating (i.e., non-zero element) r_{ij} of $\mathbf{R}$, let e_{ij} be the error associated with the prediction $\hat{r}_{ij}$, i.e.,

$$e_{ij} = r_{ij} - \hat{r}_{ij}. \tag{4}$$

The gradient of the optimization function $\mathcal{F}$ respecting the parameters $\{\mathbf{u}_i, \mathbf{v}_j, \mathbf{w}_i, b_i\}$ is calculated by

$$\nabla_{\mathbf{u}_i} \mathcal{F} = -e_{ij} \cdot \mathbf{v}_j + \lambda_1 \cdot \mathbf{u}_i, \tag{5}$$

$$\nabla_{\mathbf{v}_j} \mathcal{F} = -e_{ij} \cdot \mathbf{u}_i + \lambda_2 \cdot \mathbf{v}_j, \tag{6}$$

$$\nabla_{\mathbf{w}_i} \mathcal{F} = -e_{ij} \cdot \mathbf{x}_j + \lambda_3 \cdot \mathbf{w}_i, \text{ and} \tag{7}$$

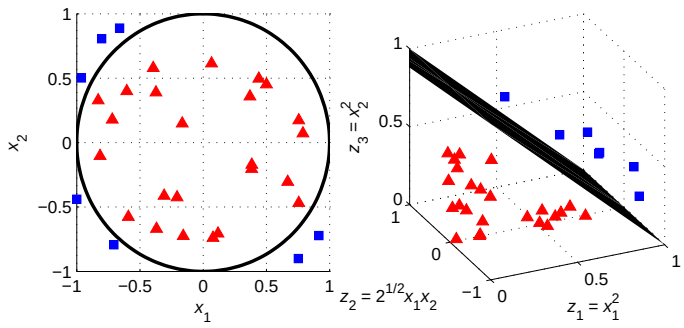
$$\nabla_{b_i} \mathcal{F} = -e_{ij}. \tag{8}$$

Then the parameters are updated by moving in the opposite direction of the gradient of the optimization function regarding the parameters, yielding:

$$\mathbf{u}_i \leftarrow \mathbf{u}_i + \eta_1 (e_{ij} \cdot \mathbf{v}_j - \lambda_1 \cdot \mathbf{u}_i), \tag{9}$$

$$\mathbf{v}_j \leftarrow \mathbf{v}_j + \eta_2 (e_{ij} \cdot \mathbf{u}_i - \lambda_2 \cdot \mathbf{v}_j), \tag{10}$$

$$\mathbf{w}_i \leftarrow \mathbf{w}_i + \eta_3 (e_{ij} \cdot \mathbf{x}_j - \lambda_3 \cdot \mathbf{w}_i), \text{ and} \tag{11}$$



(a) Original feature space (b) Mapped feature space

Fig. 3. An example of nonlinear mapping of feature (attribute) space $\phi(\mathbf{x}) = (x_1^2, 2^{1/2}x_1x_2, x_2^2)$

$$b_i \leftarrow b_i + \eta_3 \cdot e_{ij}, \tag{12}$$

where η_1, η_2, and η_3 are the positive learning rates (i.e., step sizes).

3.4 Kernel-based Nonlinear Model

In Section 3.3, we have developed a linear-regression enhanced matrix factorization to integrate the attribute information of items with a user-item rating matrix for rating prediction. To overcome the limitations of factorization machines (i.e., the *inability of scaling up to high dimension of attributes*, *two-attribute interaction* and *linear interaction between attributes*) [3], [4], [5], [6], [37], this section turns the linear model into a nonlinear model based on the kernel trick and hence uplifts the predictive ability on ratings.

Nonlinear mapping of feature or attribute space. To extend a linear model into a nonlinear one, we nonlinearly map the original feature (attribute) space into a high-dimensional feature space. In other words, we turn to find *a linear function in the high-dimensional feature space* which equals *a nonlinear function in the original feature space*. For example, in Fig. 3(a), the data points marked as triangles and squares can be completely separated by the nonlinear function $x_1^2 + x_2^2 = 1$ in the original two-dimensional feature space $\mathbf{x} = (x_1, x_2)$. After transforming the two-dimensional feature space into the three-dimensional feature space through the nonlinear mapping $\phi(\mathbf{x}) = \phi(x_1, x_2) = (x_1^2, 2^{1/2}x_1x_2, x_2^2) = (z_1, z_2, z_3)$, we can find a linear function $z_1 + z_3 = 1$ that completely separates the two types of data points in the three-dimensional feature space $\mathbf{z} = (z_1, z_2, z_3)$, as depicted in Fig. 3(b).

Formally, let $\phi(\mathbf{x})$ be the nonlinear mapping on the attribute vector $\mathbf{x}$ of an item. Accordingly, the nonlinear model for rating prediction in Equation (2) is rewritten as

$$\hat{r}_{ij} = \mathbf{u}_i^{\mathrm{T}}\mathbf{v}_j + \mathbf{w}_i^{\mathrm{T}}\phi(\mathbf{x}_j) + b_i, \tag{13}$$

and the optimization function in Equation (3) is rewritten as

$$\min_{\mathbf{U},\mathbf{V},\mathbf{W},\mathbf{b}} \mathcal{F} = \frac{1}{2}\sum_{r_{ij}\in\mathbf{R}} \left(r_{ij} - \mathbf{u}_i^{\mathrm{T}}\mathbf{v}_j - \mathbf{w}_i^{\mathrm{T}}\phi(\mathbf{x}_j) - b_i\right)^2 + \frac{\lambda_1}{2}\sum_{i=1}^{M}\|\mathbf{u}_i\|^2 + \frac{\lambda_2}{2}\sum_{j=1}^{N}\|\mathbf{v}_j\|^2 + \frac{\lambda_3}{2}\sum_{i=1}^{M}\|\mathbf{w}_i\|^2. \tag{14}$$

The optimization function in Equation (14) is solved based on stochastic gradient descent as well. The formulations for updating the parameters $\mathbf{u}_i$, $\mathbf{v}_j$ and b_i are the same as in Equations (9), (10) and (12). The Equation (11) for updating the parameter $\mathbf{w}_i$ is rewritten as

$$\mathbf{w}_i \leftarrow \mathbf{w}_i + \eta_3(e_{ij} \cdot \phi(\mathbf{x}_j) - \lambda_3 \cdot \mathbf{w}_i). \tag{15}$$

Motivation and challenge on the kernel trick. The parameter $\mathbf{w}_i$ cannot be updated by Equation (15) directly, because in practice the nonlinear mapping $\phi(\mathbf{x})$ is not specified explicitly due to two reasons: (1) It is hard to search for an appropriate nonlinear mapping from unlimited candidates. (2) Usually, the dimension of the mapped feature space is very high or even unlimited, which makes the calculation prohibitively expensive on the mapping $\phi(\mathbf{x})$. Thus, it is required to leverage the kernel trick to avoid the explicit feature mapping $\phi(\mathbf{x})$. Specifically, the kernel trick replaces the dot product $\phi^{\mathrm{T}}(\mathbf{x})\phi(\mathbf{x}')$ in the high-dimensional feature space (i.e., the mapped feature space $\phi(\mathbf{x})$) with the kernel function $\mathcal{K}(\mathbf{x}, \mathbf{x}')$ in the original feature space (i.e., $\mathbf{x}$) [7]. It is challenging to exploit the kernel trick for learning the parameter $\mathbf{w}_i$ by Equation (15), since Equation (15) does not involve the dot product in the mapped feature space. Thus, we have to fill the gap from the update of Equation (15) to the kernel trick.

Maintaining the coefficient of the nonlinear mapping $\phi(\mathbf{x})$. To this end, we contrive a method to maintain the coefficient of the mapping $\phi(\mathbf{x})$ to discover its relation with respect to the parameter $\mathbf{w}_i$. At first, we reformat Equation (15) a little into

$$\mathbf{w}_i \leftarrow (1 - \lambda_3 \cdot \eta_3)\mathbf{w}_i + \eta_3 \cdot e_{ij} \cdot \phi(\mathbf{x}_j). \tag{16}$$

Based on the iterative Equation (16), **we have two important observations**: (1) The initial coefficient of $\phi(\mathbf{x}_j)$ is $\eta_3 e_{ij}$ when $\mathbf{x}_j$ is selected for training. (2) Then the coefficient will be multiplied by $1 - \lambda_3\eta_3$ at each subsequent update.

Without loss of generality, we focus on the update of the parameter $\mathbf{w}_i$ for user u_i. Let $\mathbf{x}_{j_k}$ be the kth selected training sample for updating the parameter $\mathbf{w}_i$ in Equation (16). The coefficients of $\phi(\mathbf{x}_j)$ on training samples $\mathbf{x}_{j_1}, \mathbf{x}_{j_2}, \cdots, \mathbf{x}_{j_n}$ at each update round are depicted in TABLE 2. Therefore, at the nth update round (the last row of TABLE 2), the parameter $\mathbf{w}_i$ is identical with the weighed sum of $\phi(\mathbf{x}_j)$ on training samples $\mathbf{x}_{j_1}$, $\mathbf{x}_{j_2}, \cdots, \mathbf{x}_{j_n}$, given by

$$\begin{aligned}\mathbf{w}_i &= \sum_{k=1}^{n} \eta_3 e_{ij_k}(1 - \lambda_3\eta_3)^{n-k}\phi(\mathbf{x}_{j_k}) \\ &= \eta_3(1 - \lambda_3\eta_3)^n \sum_{k=1}^{n}(1 - \lambda_3\eta_3)^{-k} e_{ij_k}\phi(\mathbf{x}_{j_k}),\end{aligned} \tag{17}$$

in which n **is a user-specified count, i.e., the total number of training samples currently selected for user u_i, also denoted as n_{u_i}.**

The kernel trick. Whence, the term $\mathbf{w}_i^{\mathrm{T}}\phi(\mathbf{x}_j)$ in Equations(13) and (14) is replaced by

$$\begin{aligned}\mathbf{w}_i^{\mathrm{T}}\phi(\mathbf{x}_j) &= \eta_3(1 - \lambda_3\eta_3)^n \sum_{k=1}^{n}(1 - \lambda_3\eta_3)^{-k} e_{ij_k}\phi^{\mathrm{T}}(\mathbf{x}_{j_k})\phi(\mathbf{x}_j) \\ &= \eta_3(1 - \lambda_3\eta_3)^n \sum_{k=1}^{n}(1 - \lambda_3\eta_3)^{-k} e_{ij_k}\mathcal{K}(\mathbf{x}_{j_k}, \mathbf{x}_j),\end{aligned} \tag{18}$$

where the dot product $\phi^{\mathrm{T}}(\mathbf{x}_{j_k})\phi(\mathbf{x}_j)$ in the high-dimensional space is avoided via the kernel function $\mathcal{K}(\mathbf{x}_{j_k}, \mathbf{x}_j)$. For example, on the nonlinear mapping $\phi(\mathbf{x}) = (x_1^2, 2^{1/2}x_1x_2, x_2^2)$ in Fig. 3, we have $\mathcal{K}(\mathbf{x}_{j_k}, \mathbf{x}_j) = \phi^{\mathrm{T}}(\mathbf{x}_{j_k})\phi(\mathbf{x}_j) = (\mathbf{x}_{j_k}^{\mathrm{T}}\mathbf{x}_j)^2$. It

TABLE 2
Coefficients of $\phi(\mathbf{x}_j)$ on training samples $\mathbf{x}_{j_1}, \mathbf{x}_{j_2}, \cdots, \mathbf{x}_{j_n}$ at each update round

Round	$\phi(\mathbf{x}_{j_1})$	$\phi(\mathbf{x}_{j_2})$	$\cdots$	$\phi(\mathbf{x}_{j_k})$	$\cdots$	$\phi(\mathbf{x}_{j_n})$
1st	$\eta_3 e_{ij_1}$					
2nd	$\eta_3 e_{ij_1}(1-\lambda_3\eta_3)$	$\eta_3 e_{ij_2}$				
$\vdots$	$\vdots$	$\vdots$	$\ddots$			
kth	$\eta_3 e_{ij_1}(1-\lambda_3\eta_3)^{k-1}$	$\eta_3 e_{ij_2}(1-\lambda_3\eta_3)^{k-2}$	$\cdots$	$\eta_3 e_{ij_k}$		
$\vdots$	$\vdots$	$\vdots$	$\ddots$	$\vdots$	$\ddots$	
nth	$\eta_3 e_{ij_1}(1-\lambda_3\eta_3)^{n-1}$	$\eta_3 e_{ij_2}(1-\lambda_3\eta_3)^{n-2}$	$\cdots$	$\eta_3 e_{ij_k}(1-\lambda_3\eta_3)^{n-k}$	$\cdots$	$\eta_3 e_{ij_n}$

is important to note that: (1) The kernel trick greatly reduces the computational cost through replacing the high-dimensional dot product with the kernel function that calculates the dot-product in the original feature space, i.e., $\mathbf{x}_{j_k}^{\mathrm{T}}\mathbf{x}_j$. (2) Choosing a kernel is much easier than searching for a mapping, since there are some common kernels [7]:

- Normalized polynomial function:

$$\mathcal{K}(\mathbf{x}_{j_k}, \mathbf{x}_j) = \left(\frac{\mathbf{x}_{j_k}^{\mathrm{T}}\mathbf{x}_j}{\|\mathbf{x}_{j_k}\| \cdot \|\mathbf{x}_j\|} + 1\right)^d, d \in \mathbb{N}. \quad (19)$$

  When the polynomial degree $d = 1$, the proposed model is degraded into the linear model, as presented in Section 3.3.
- Radial basis function:

$$\mathcal{K}(\mathbf{x}_{j_k}, \mathbf{x}_j) = \exp\left(-\gamma \left\|\mathbf{x}_{j_k} - \mathbf{x}_j\right\|^2\right), \gamma > 0. \quad (20)$$

- Hyperbolic tangent (Sigmoid) function:

$$\mathcal{K}(\mathbf{x}_{j_k}, \mathbf{x}_j) = \tanh\left(s\mathbf{x}_{j_k}^{\mathrm{T}}\mathbf{x}_j + c\right), s > 0, c \leq 0, \quad (21)$$

  where s denotes the slope and c is the intercept constant.

In sum, the model parameters $\{\mathbf{u}_i, \mathbf{v}_j, \mathbf{w}_i, b_i\}$ can be learned through Equations (9), (10), (12) and (17). It is worth emphasizing that in Equation (17) we just update the coefficient of the mapping $\phi(\mathbf{x}_{j_k})$. In other words, we maintain the relation between the parameter $\mathbf{w}_i$ and the mapping $\phi(\mathbf{x}_{j_k})$. Hence, the parameter $\mathbf{w}_i$ can be used for rating prediction in Equation (13) based on the kernel trick in Equation (18). Therefore, the proposed kernel-based attribute-aware matrix factorization (KAMF) always operates in the original feature (attribute) space (i.e., $\mathbf{x}$) and avoids the explicit feature mapping (i.e., $\phi(\mathbf{x})$) that is needed to turn a linear model into a nonlinear model.

3.5 Incremental Learning Algorithm

As discussed in Section 3.4, because the nonlinear mapping $\phi(\mathbf{x}_{j_k})$ in Equation (17) is never specified explicitly, the parameter $\mathbf{w}_i$ cannot be calculated directly. As a result, it requires a lot of computational cost to maintain the relation between the parameter $\mathbf{w}_i$ and the mapping $\phi(\mathbf{x}_{j_k})$, due to repeated calculation of the coefficient of each training sample $\mathbf{x}_{j_k}$ at each update round.

Coefficient grouping and updating. To this end, we develop an approach that *only stores* and *incrementally updates* the coefficient of $\phi(\mathbf{x}_{j_k})$. Specifically, we group the coefficients for each training sample $\mathbf{x}_{j'}$ of item $v_{j'}$, $(j' = 1, 2, \cdots N)$ by summing them together for user u_i as $a_{ij'}$, given by

$$\alpha_{ij'} = \sum_{k=1}^{n} \mathcal{I}(j_k = j')(1-\lambda_3\eta_3)^{-k} e_{ij_k}, \quad (22)$$

where $\mathcal{I}(j_k = j')$ is the indicator function which takes a value of one if the argument $j_k = j'$ is true and zero otherwise. **Note**

Algorithm 1 Incremental learning process for KAMF

Input: Sparse rating matrix $\mathbf{R}$ and attribute matrix $\mathbf{X}$
Output: $\{\mathbf{u}_i, \mathbf{v}_j, b_i, \alpha_{ij}, n_{u_i}\}$
1: **// Phase 1: The initializing phase**
2: Randomize parameters $\{\mathbf{u}_i, \mathbf{v}_j, b_i\}$ from (0,1) for each user u_i and item v_j
3: $\alpha_{ij} \leftarrow 0$ for each user u_i and item v_j
4: $n_{u_i} \leftarrow 0$ for the count of training samples of user u_i
5: **// Phase 2: The incremental learning phase**
6: **for** each iteration until convergence **do**
7: **for** each randomly selected known rating $r_{ij} \in \mathbf{R}$ **do**
8: SGD(u_i, v_j, r_{ij}) in Algorithm 2
9: **end for**
10: **end for**

Algorithm 2 Stochastic Gradient Descent: SGD(u_i, v_j, r_{ij})

Input: User u_i, item v_j, and rating r_{ij}
Output: $\{\mathbf{u}_i, \mathbf{v}_j, b_i, \alpha_{ij}, n_{u_i}\}$
1: Predict rating $\hat{r}_{ij}$ based on Equation (24)
2: Compute error e_{ij} based on Equation (4)
3: Update parameter $\mathbf{u}_i$ based on Equation (9)
4: Update parameter $\mathbf{v}_j$ based on Equation (10)
5: Update parameter b_i based on Equation (12)
6: $n_{u_i} \leftarrow n_{u_i} + 1$
7: $\alpha_{ij} \leftarrow \alpha_{ij} + (1-\lambda_3\eta_3)^{-n_{u_i}} e_{ij}$ based on Equation (22)

two important points: (1) Each $\alpha_{ij'}$ represents the coefficient of item $v_{j'}$ on user u_i. (2) $\alpha_{ij'}$ can be incrementally modified as the number n of training samples increases. Accordingly, Equation (17) is reformatted into

$$\mathbf{w}_i = \eta_3(1-\lambda_3\eta_3)^n \sum_{j'=1}^{N} \alpha_{ij'}\phi(\mathbf{x}_{j'}), \quad (23)$$

which represents the relation between *explicit coefficient* $\alpha_{ij'}$ and *implicit coefficient* $\mathbf{w}_i$. Eventually, the rating prediction in Equation (13) is reformatted into

$$\begin{aligned}\hat{r}_{ij} &= \mathbf{u}_i^{\mathrm{T}}\mathbf{v}_j + \eta_3(1-\lambda_3\eta_3)^n \sum_{j'=1}^{N} \alpha_{ij'}\phi^{\mathrm{T}}(\mathbf{x}_{j'})\phi(\mathbf{x}_j) + b_i \\ &= \mathbf{u}_i^{\mathrm{T}}\mathbf{v}_j + \eta_3(1-\lambda_3\eta_3)^n \sum_{j'=1}^{N} \alpha_{ij'}\mathcal{K}(\mathbf{x}_{j'}, \mathbf{x}_j) + b_i, \quad (24)\end{aligned}$$

where n is a user-specified count, i.e., n_{u_i} or the total number of training samples currently selected for user u_i which is generally proportional to the number of iterations on a training data set in Algorithm 1.

Complexity analysis. Algorithm 1 summarizes the incremental learning process. At each update round (Line 8 in Algorithm 1), the most time-consuming calculation is the estimation of the rating (Line 1 in Algorithm 2), which linearly increases with the number N_{u_i} of items rated by user u_i based on Equation (24), because at most N_{u_i} of α_{ij} take non-zero values. In addition, the dot product in Equation (24) costs $O(L)$ work, in which L is the dimension of latent factor vectors, and updating $\mathbf{u}_i$ and $\mathbf{v}_j$ also costs $O(L)$ work (Lines 3 and 4 in Algorithm 2). Moreover, the update will

be executed with n_{u_i} times, i.e., the number of training samples selected for user u_i. Hence, by summing over all update rounds for all users, the total time complexity is $O(\sum_{i=1}^{M} n_{u_i}(N_{u_i}+L))$, which shows the linear relation with respect to N_{u_i}, L and n_{u_i}. In terms of memory cost, Algorithm 1 needs to store two low-rank factor matrices $\mathbf{U} \in \mathbb{R}^{L\times M}$ and $\mathbf{V} \in \mathbb{R}^{L\times N}$, α_{ij} and $\mathcal{K}(\mathbf{x}_{j'}, \mathbf{x}_j)$ with the very low sparsity as the user-item rating matrix $\mathbf{R}$, and $\mathbf{b} \in \mathbb{R}^M$. Thus, the total space complexity is $O(ML+NL+|\mathbf{R}|)$, in which $|\mathbf{R}|$ denotes the number of training ratings.

4 Cold-Start Problems

Recommender systems usually encounter the cold-start problems: how to recommend *new items* to users, known as *the item cold-start problem*, and how to recommend items to *new users*, known as *the user cold-start problem*. The traditional collaborative filtering techniques cannot deal with the two types of cold-start problems, because new items have received few ratings from users while new users have rated few items. In contrast, **our proposed kernel-based attribute-aware matrix factorization model (KAMF) can overcome the item cold-start problem by nature, since KAMF derives the attribute-aware preferences of users on new items to predict the ratings of the users on the new items.** Therefore, we concentrate on addressing the user cold-start problem by extending the proposed model.

4.1 Main Ideas

Utilizing social strength of users. For a cold-start user who has never rated any items or rated only a few items, the extended model resorts to the ratings of other users, i.e., the friends of the cold-start user. The main reason is intuitive: in reality friends show more common interests on items than strangers. Fortunately, the social relationships between users are widely accessible in the Internet (note that the attributes of users are often unavailable due to the privacy issue). For example, in social networks like Facebook, Foursquare and Yelp, users establish social links with others in order to exchange their ideas and share their experiences of reading books, watching movies, or checking in venues. Moreover, the social networks often provide APIs for third parties to crawl the social links. Each social link only consists of two users and indicates they are friends online and have common interests. The social strength of two users can be measured by the social link between them and the number of their common friends. Formally, let $F(u_i)$ be the set of friends of user u_i. The social strength $soc(u_i, u_{i'})$ between users u_i and $u_{i'}$ is defined by

$$soc(u_i, u_{i'}) = \begin{cases} \frac{1}{2} + \frac{F(u_i)\cap F(u_{i'})}{2F(u_i)\cup F(u_{i'})}, & u_i, u_{i'} \text{ have social link;} \\ 0, & \text{otherwise.} \end{cases} \tag{25}$$

Note that this simple definition is effective in our experiments, and other methods using social links can also be applied here, which is not the focus of this paper due to space limitations.

The sampling-based approach. The basic idea of the extended model is to learn the model parameters (i.e., $\mathbf{u}_i$, $\mathbf{w}_i$ and b_i) for the user cold-start users by sampling the ratings of their friends. Moreover, the rating sampling probability is clearly related to the social strength among users, because a user will be affected by their friends to different degree. The reason is twofold for devising the sampling-based approach rather than using social regularization [17], [18], [19], [20], [21], [22], [30] or random walk [45]: (1) The sampled ratings can be seamlessly integrated with KAMF proposed in Section 3, since KAMF iteratively updates its models parameters in terms of each rating, as shown in Algorithms 1 and 2. (2) The attribute information of items associated with the sampled ratings can be also exploited to adjust the model parameters. However, the social regularization and random walk utilize social relationships and ratings but ignore the attribute information of items.

Decay of rating sampling probability. Further, the rating sampling probability should decay with the increase of the number of current ratings of a user, including *her own ratings* and *the ratings sampled from her friends*. This decay property is very important due to two reasons: (1) It avoids to set up a threshold on the number of ratings to define the cold-start users. That is, the decay property allows all users to be processed in a universal way. (2) It achieves an adaptive balance on the effect of *a user's own ratings* and *her friends' ratings*. If a user rates a large number of items, we can derive her preference using her own sufficient ratings. Consistently, based on the decay property, the rating sampling probability for the user will be low and thus her friends' ratings will play little effect on the user. On the contrary, for a cold-start user with few ratings, the rating sampling probability for the user cold-start user will be high and her friends' ratings will greatly benefit the cold-start user. In other words, the social links of the cold-start users are more useful than those of the non-cold-start users. The decay property ensures that the social links of users are adaptively chosen for rating sampling, namely, the social links of the cold-start users are employed with higher probability for rating prediction.

Integration of social strength and decay rate. In sum, the probability on sampling a rating for a user from her friend depends on both *the social strength between the user and her friend* and *the number of current ratings of the user*. Formally, let two users u_i and $u_{i'}$ be friends with the social strength $soc(u_i, u_{i'})$ in Equation (25), $m_{u_{i'}}$ be the number of current ratings of user $u_{i'}$, and r_{ij} be a rating of user u_i on item v_j. The probability on sampling rating r_{ij} for user $u_{i'}$ is defined by

$$\Pr(r_{ij}|u_{i'}) = soc(u_i, u_{i'}) \cdot \exp(-\beta \cdot m_{u_{i'}}), \beta > 0, \tag{26}$$

where β controls the decay rate. The larger β is, the faster is the decay rate. The rating sampling probability exponentially decays as the number of current ratings increases, because the importance of the ratings of friends greatly decreases when a user has enough ratings that are sufficient to learn her preference on items. The decay rate β can be determined by grid searches: a coarse grid/step search is first used to identify a good region/range and then a finer grid search on that region is conducted to find the best β.

4.2 Algorithm

Extension of KAMF for the user cold-start problems. Algorithm 3 extends Algorithm 1 to address the cold-start problem. Here we focus on analyzing the differences of Algorithm 3 from Algorithm 1, i.e., the extra steps in Algorithm 3 for solving the cold-start problem. In the initializing phase, Algorithm 3 prepares the set $V(u_i)$ of items rated by user u_i (Line 5) and the set $F(u_i)$ of friends of user u_i (Line 6), which will be used for the user cold-start problem later. In the incremental learning phase, for each rating r_{ij} of user u_i on item v_j, Algorithm 3 updates the model parameters of user u_i and item v_j as usual (Line 11). Then, for each friend $u_{i'} \in F(u_i)$ of user u_i, Algorithm 3 computes the social strength $soc(u_i, u_{i'})$ (Line 14), number of current ratings

Algorithm 3 Extension of KAMF for cold-start problems

Input: Sparse rating matrix $\mathbf{R}$ and attribute matrix $\mathbf{X}$
Output: $\{\mathbf{u}_i, \mathbf{v}_j, b_i, \alpha_{ij}, n_{u_i}\}$
1: **// Phase 1: The initializing phase**
2: Randomize parameters $\{\mathbf{u}_i, \mathbf{v}_j, b_i\}$ from (0,1) for each user u_i and item v_j
3: $\alpha_{ij} \leftarrow 0$ for each user u_i and item v_j
4: $n_{u_i} \leftarrow 0$ for the count of training samples of user u_i
5: $V(u_i) \leftarrow$ the set of items rated by user u_i
6: $F(u_i) \leftarrow$ the set of friends of user u_i
7: **// Phase 2: The incremental learning phase**
8: **for** each iteration until convergence **do**
9: **for** each randomly selected known rating $r_{ij} \in \mathbf{R}$ **do**
10: **// Phase 2.1 runs Algorithm 2, as in Algorithm 1**
11: SGD(u_i, v_j, r_{ij}) in Algorithm 2
12: **// Phase 2.2 for the user cold-start problem**
13: **for** each friend $u_{i'} \in F(u_i)$ **do**
14: Compute social strength $soc(u_i, u_{i'})$ based on Equation (25)
15: Compute the number of current ratings (more precisely, rated different items): $m_{u_{i'}} \leftarrow |V(u_{i'})|$
16: Compute sampling probability $\Pr(r_{ij}|u_{i'})$ based on Equation (26)
17: Generate random number $p \in (0, 1)$
18: **if** $p < \Pr(r_{ij}|u_{i'})$ **then**
19: SGD($u_{i'}, v_j, r_{ij}$) in Algorithm 2
20: $V(u_{i'}) \leftarrow V(u_{i'}) \cup \{v_j\}$
21: **end if**
22: **end for**
23: **end for**
24: **end for**

$m_{u_{i'}}$ (Line 15) and sampling probability $\Pr(r_{ij}|u_{i'})$ (Line 16). If a generated random number is less than the sampling probability (Lines 17 and 18), the rating r_{ij} is sampled to update the model parameters of friend $u_{i'}$ (Line 19) and the item v_j is added into the set $V(u_{i'})$ (Line 20).

Complexity analysis. In comparison to Algorithm 1, Algorithm 3 executes the extra update for solving the user cold-start problem (Line 19); thus, the number n_{u_i} of updates for user u_i becomes larger. Moreover, the set $V(u_i)$ of items that are used for training the parameters of user u_i includes not only *the items rated by user* u_i (Line 5) but also *the items sampled from her friends* (Line 20). Hence, the cost for the rating prediction in Equation (24) linearly increases with respect to $m_{u_i} = |V(u_i)|$, rather than the number N_{u_i} of items rated by user u_i herself in Algorithm 1. Accordingly, the total time complexity is $O(\sum_{i=1}^{M} n_{u_i}(m_{u_i} + L))$. It is important to note that: only for the user cold-start users, n_{u_i} of Algorithm 3 is possibly significantly larger than n_{u_i} of Algorithm 1, and m_{u_i} of Algorithm 3 is larger than N_{u_i} of Algorithm 1, because the rating sampling probability is considerably low or negligible for the non-cold-start users. On the memory cost, opposed to Algorithm 1, Algorithm 3 just needs some extra memory to store the cold-start user $u_{i'}$'s $\alpha_{i'j}$ corresponding to the sampled item v_j rated by her friend u_i. Similarly, the total space complexity is $O(ML + NL + |\mathbf{R}'|)$, in which $\mathbf{R}'$ is the original sparse user-item rating matrix $\mathbf{R}$ with the sampled ratings for the user cold-start users.

5 Experiments

In this section, we conduct extensive experiments to evaluate the performance of KAMF, compared to state-of-the-art rating prediction techniques. We present experimental settings in Section 5.1 and analyze experimental results in Section 5.2.

5.1 Experimental Settings

Data sets. We use two publicly available large-scale real data sets: Yelp [46] and MovieLens [47] that were released in 2015. In the yelp data set, most users had rated less than 20 venues, as depicted in Fig. 4. In the MovieLens data set, all users had rated at least 20 movies. TABLE 3 shows the statistics of the two data sets. Note that the Yelp data set includes a large number of attributes, but not all attributes are adaptable for all venues; TABLE 4 lists the top-5 popular attributes of venues. The MovieLens data set provides 19 attributes of movies on genres (e.g., Action, Animation, Comedy, and Crime). However, the MovieLens data set does not contain the social links between users and thus are not applicable for evaluating the social strength based methods.

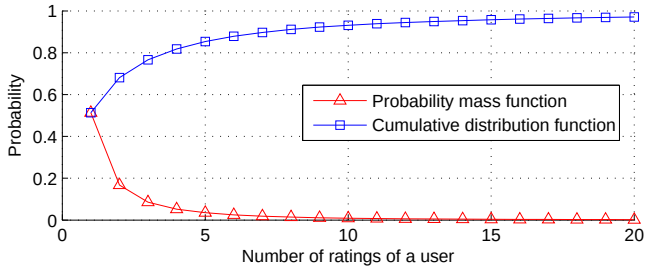


Fig. 4. Distribution of the number of items rated by a user in Yelp

TABLE 3
Statistics of the two data sets

	Yelp	MovieLens
Number (M) of users	366,000	138,000
Number (N) of items (venues)	61,000	27,000
Number (D) of distinct attributes	1,285	19
Number of known ratings	1,600,000	20,000,000
Number of social links	2,900,000	—
Sparsity of user-item rating matrix	7.17×10^{-5}	5.38×10^{-3}

TABLE 4
Top-5 popular attributes of venues in Yelp

Attribute Name	Type of Value
Category	Nominal
Geo-Coordinates	Real
Accepts Credit Cards	Boolean
Price Range	Real
Noise Level	Nominal

In the preprocessing, it is required to assign a real-value to a nominal attribute, e.g., the category of venues. We apply a simple but effective method that defines a binary variable for each possible nominal value of the nominal attribute. For example, supposing the category of venues takes three possible nominal values: Restaurant, Store, and Museum; if a venue belongs to the Restaurant category, the corresponding binary variable takes one and zero otherwise. We evaluate all techniques on rating prediction using five-fold cross-validation. That is, the whole set of ratings are divided into five groups, and each group is used to test the rating prediction models that are learned from the other four groups.

Evaluated techniques. The basic KAMF (i.e., Algorithm 1) is compared with the state-of-the-art rating prediction competitors (without social information between users) including:

- UFSM: This method is the User-specific Feature-based Similarity Model [26] that utilizes the attribute information of items to train a regression model for rating prediction.
- SVM: This method is the Support Vector Machine that utilizes the attribute information of items to train a kernel-based regression model for rating prediction [20].
- ICF: This method applies the Item-based Collaborative Fil-

tering (CF), in which the unknown ratings are predicted by considering the ratings given for similar items [9].
- UCF: This method employs the User-based CF, in which the unknown ratings are predicted by considering the ratings given by similar users [8].
- NMF: This method utilizes the Nonnegative Matrix Factorization that imposes the constraint of nonnegativity on the latent factors of users and items [16].
- PMF: This method exploits the famous Probabilistic Matrix Factorization [12], as presented in Section 3.2.
- SVD: This method considers the Singular Value Decomposition for rating prediction [1].
- RLFM: This method uses the Regression-based Latent Factor Model to map contexts (attributes) into latent factors [33].
- CFM: This method uses the Context-aware Factorization Machine [5], in which each context (attribute) is also associated with a latent factor vector.
- NCPD: This method tailors matrix factorization to incorporate the influence of Neighborhood, Category, Popularity and Geographical distance of venues for rating prediction [37].

On the cold-start problem, the extension of KAMF, denoted by KAMF+ (i.e., Algorithm 3) is compared with the state-of-the-art rating prediction competitors (with social information between users) including:
- UCF+: This method extends the User-based CF by utilizing social links to derive similarities [10].
- PMF+: This method improves the Probabilistic Matrix Factorization by augmenting a social regularization term [17].
- TRMF+: This method combines Topic Regression and Matrix Factorization with a social regularization term, in which contexts are used to pre-process (i.e., sub-group) the user-item rating matrix before training [30].
- CFM+: This method enhances the Context-aware Factorization Machine [5] by adding a social regularization term.
- NCPD+: This method extends NCPD [37] by augmenting a social regularization term.
- KAMF*: It extends KAMF by adding a social regularization term instead of using the sampling-based approach.

Performance metrics. We adopt two popular performance metrics for rating prediction, namely, Mean Absolute Error (MAE) and Root Mean Square Error (RMSE), formally defined as:

$$\text{MAE} = \frac{1}{|T|}\sum\nolimits_{(i,j)\in T} |r_{ij} - \hat{r}_{ij}| \text{ and}$$

$$\text{RMSE} = \sqrt{\frac{1}{|T|}\sum\nolimits_{(i,j)\in T} (r_{ij} - \hat{r}_{ij})^2},$$

where T is the set of user-item rating pairs (i, j) in the testing data set, r_{ij} is the known rating, and $\hat{r}_{ij}$ is the predicted rating. Clearly, the lower MAE or RMSE indicates better prediction accuracy.

Parameter settings. In terms of the popular grid search method for hyperparameter optimization, we empirically set the hyperparameters including the regularization parameters $\lambda_1 = \lambda_2 = \lambda_3 = 0.05$, the learning rates $\eta_1 = \eta_2 = \eta_3 = 0.005$, the decay rate $\beta = 0.05$, and the dimension of latent factor vectors $L = 50$. The kernel is set to the normalized polynomial function with $d = 4$ in Equation (19). The number of iterations in Algorithms 1 and 3 is set to 50 (each iteration passes on the entire data set), at which the stochastic gradient descent is convergent. Due to better performance in our experiments, we apply the popular cosine metric for both ICF and UCF. For the social strength based methods, we adopt the social regularization term that uses the average latent factors of a user's friends as the user's prior, as presented in the work [30].

TABLE 5
Average personalized item rating prediction error for all non-cold-start users (the lower the better)

		Yelp		MovieLens	
	Method	MAE	RMSE	MAE	RMSE
Content-based	UFSM	1.4349	2.0141	1.3346	1.8904
	SVM	1.3580	1.9886	1.3266	1.8635
Collaborative filtering	ICF	1.1268	1.5333	1.0023	1.3892
	UCF	1.1249	1.5295	1.1024	1.4213
	NMF	1.1289	1.5486	1.0932	1.3998
	PMF	1.1150	1.5188	0.9874	1.3035
	SVD	1.1008	1.4632	1.0034	1.4124
Attribute-aware	RLFM	1.0823	1.4189	0.9626	1.2869
	CFM	1.0609	1.3887	0.9468	1.2224
	NCPD	1.0381	1.3191	0.9589	1.2810
	KAMF	**0.9388**	**1.1942**	**0.8421**	**1.0215**
Social strength based	UCF+	1.1167	1.5032		
	PMF+	1.1200	1.5249		
	TRMF+	1.0812	1.4162		
	CFM+	1.0566	1.3323	N/A	
	NCPD+	1.0202	1.2886		
	KAMF*	0.9608	1.2279		
	KAMF+	**0.9189**	**1.1704**		

TABLE 6
Average personalized item rating prediction error for all cold-start users (the lower the better)

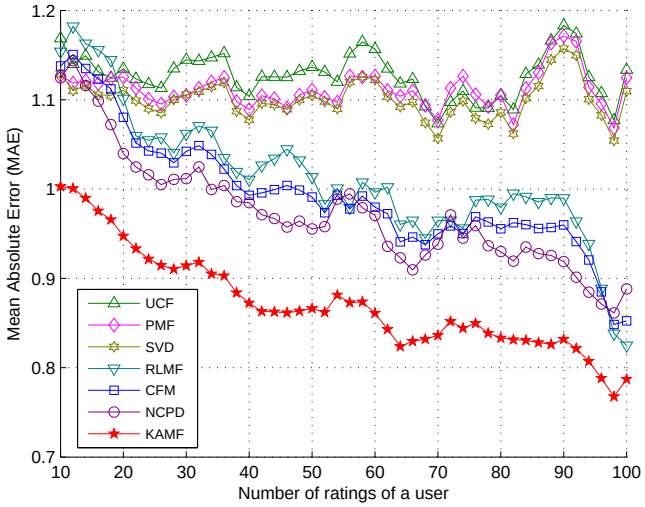
		Yelp		MovieLens	
	Method	MAE	RMSE	MAE	RMSE
Content-based	UFSM	1.8964	2.5012	1.6234	2.2653
	SVM	1.7692	2.4563	1.6215	2.3480
Collaborative filtering	ICF	1.4589	1.8864	1.3123	1.7896
	UCF	1.4535	1.8689	1.4208	1.8092
	NMF	1.4680	1.8965	1.3876	1.8001
	PMF	1.4441	1.8494	1.2876	1.6964
	SVD	1.4319	1.8356	1.3245	1.7998
Attribute-aware	RLFM	1.4099	1.8108	1.2632	1.6668
	CFM	1.3943	1.8120	1.1986	1.4876
	NCPD	1.3676	1.7954	1.2459	1.5623
	KAMF	**1.2225**	**1.5909**	**1.0869**	**1.4680**
Social strength based	UCF+	1.2529	1.6760		
	PMF+	1.2419	1.6485		
	TRMF+	1.2399	1.6376		
	CFM+	1.2388	1.6103	N/A	
	NCPD+	1.2133	1.5966		
	KAMF*	1.1780	1.5289		
	KAMF+	**1.1252**	**1.4624**		

5.2 Experimental Results

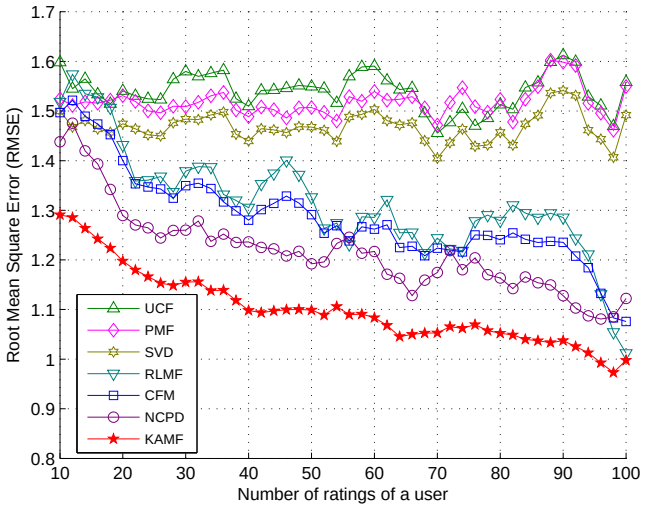
We compare the performance of KAMF against the state-of-the-art rating prediction methods (Section 5.2.1), study the extension of KAMF for the cold-start users (Section 5.2.2), investigate the impact of attributes, kernel functions, and decay rates in Sections 5.2.3, 5.2.4, and 5.2.5, respectively, and then evaluate the complexity of KAMF in Section 5.2.6.

5.2.1 Comparison of Methods

TABLE 5 compares the rating prediction accuracy of the evaluated methods on the non-cold-start users who have rated more than ten

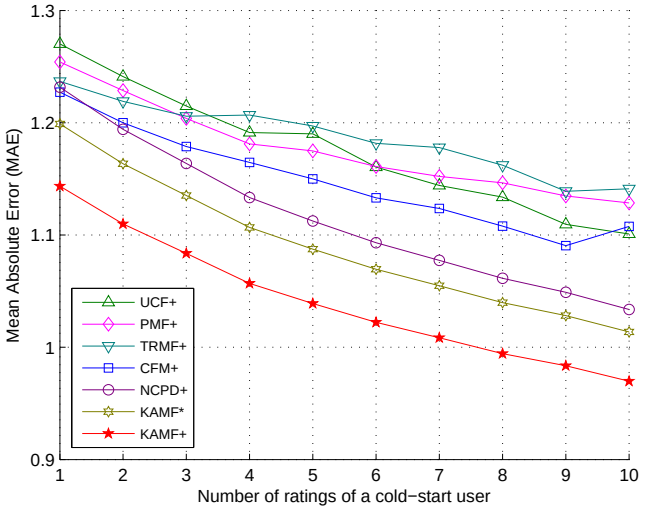


(a) MAE on users with the same number of ratings

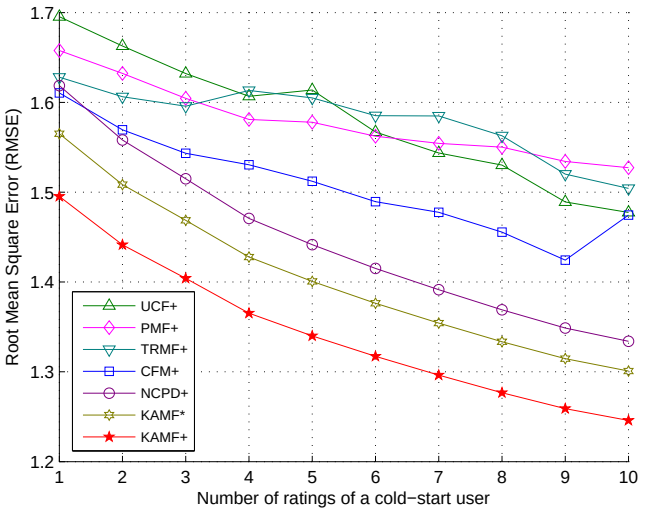


(b) RMSE on users with the same number of ratings

Fig. 5. Effect of each individual subset of non-cold-start users with a certain number of ratings in the training set of Yelp on prediction error



(a) MAE on users with the same number of ratings



(b) RMSE on users with the same number of ratings

Fig. 6. Effect of each individual subset of cold-start users with a certain number of ratings in the training set of Yelp on prediction error

items in the training set. We have five key observations: (1) The content-based methods ignore the interactions of different users on the same item and build a classification model for each user by using her own ratings. However, users usually rate a few items that are not sufficient to train a reliable classification model for the user. As a result, they report the highest rating prediction error. Hereafter, we focus on CF techniques that consider the interactions of users on items. (2) The CF methods are competitive to each other and show a significant improvement over the content-based methods in terms of MAE and RMSE. For example, SVD using $\mathbf{u}_i^{\mathrm{T}}\mathbf{v}_j$ outperforms SVM using $\mathbf{w}_i^{\mathrm{T}}\phi(\mathbf{x}_j) + b_i$ in Equation (13), which verifies the importance of the user interaction part $\mathbf{u}_i^{\mathrm{T}}\mathbf{v}_j$. However, these methods still generate relatively high error due to data sparsity. (3) The attribute-aware methods decrease the prediction error further through using the extra information, i.e., the attributes of items which can mitigate the data sparsity effect of the user-item rating matrix. For example, given new items with few ratings from users, KAMF can leverage the attribute information to derive the user preferences on the new items, which complements the insufficient ratings of new items in PMF. (4) All evaluated methods report lower prediction error on the MovieLens data set than the Yelp data set, because MovieLens has a much denser user-item rating matrix. However, the experimental results of these methods on both data sets are similar. Hereafter, we focus on analyzing the performance on the Yelp data set which includes the social links between users for the social strength based methods. (5) The social strength of users shows limited benefit for the non-cold-start users (PMF+ is even a little worse than PMF), as their own ratings are more valuable to derive their own preferences on items.

Fig. 5 depicts the prediction error of the attribute-aware methods and three CF methods for each individual subset of users with a certain number of ratings in the training set. Note that most methods have been extended to incorporate the social strength of users to address the cold-start problem, which will be presented in Section 5.2.2. Based on Fig. 5, we can conclude three findings: (1) The CF methods (UCF, PMF and SVD) generate the steadily fluctuant rating prediction error with the increase of the number of ratings of a user in the training set. The reason is that their prediction error is prone to being convergent when the number of training ratings for a user is larger than a certain threshold, e.g., ten in the current case. (2) RLMF, CFM and NCPD show the decreasing error by using the attributes of more items. Moreover, NCPD is better than RLMF and CFM, since NCPD has been

customized to predict the user ratings on venues by employing the attributes of venues such as neighbors, categories, popularity and geographical locations, but both RLMF and CFM are a generic method for rating prediction on any kind of items. (3) The prediction error of KAFM generally declines as the number of ratings of a user gets larger. Note that the little fluctuation of prediction error is caused by the common random noise in statistics, because in reality most users rate only a few items, e.g., rating less than 20 venues in the Yelp data set, and thus there may be not sufficient users with more than 20 ratings to obtain the steady performance. However, the macro-trend is more important than the micro-fluctuation. Most importantly, in comparison to other attribute-aware competitors, KAFM achieves significantly lower prediction error, because it is able to model the nonlinear interaction among multiple attributes based on the kernel-based regression, which overcomes existing methods' limitations: *two-attribute interaction* and/or *linear interaction between attributes*.

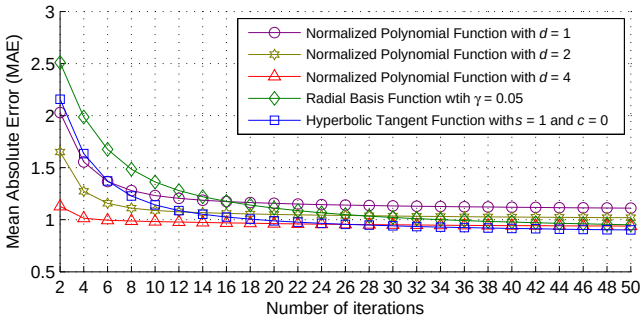
5.2.2 Study on Cold-Start Users

TABLE 6 contrasts the rating prediction accuracy of the evaluated methods on the cold-start users who have rated less than ten items in the training set. The techniques without the social strength of users record the high prediction error. Here we focus on the methods that utilize the social strength. We have four findings: (1) TRMF+ shows the higher prediction error than the other attribute-aware methods (e.g., CFM+), because it uses contexts or attributes to sub-group the user-item rating matrix before applying matrix factorization and cannot infer the attribute-aware preferences of users on items. (2) All the social strength based methods greatly reduce the prediction error, compared to their counterparts without the social strength of users. This validates that the social strength of users is very beneficial for the cold-start users. Our explanation is that: friends generally show more common interests on items than strangers; the social links of the cold-start users can significantly complement their insufficient ratings. (3) According to the results of KAMF+ and KAMF*, the sampling-based approach is a better way to utilize the social information of users than the social regularization method. This is because the former utilizes both *the social links of a cold-start user* and *the attributes of the sampled items* to learn her latent factor vector $\mathbf{u}_i$ and coefficient vector $\mathbf{w}_i$, while the latter only uses the social links to regularize the latent factor vector $\mathbf{u}_i$. (4) Our KAMF+ reaches the lowest prediction error and lowers the error about 10% against the best competitor NCPD+. The reason is twofold: (i) The underlying KAMF is superior than other methods, as depicted in Fig. 5. (ii) KAMF+ applies the sampling-based approach instead of using the social regularization method.

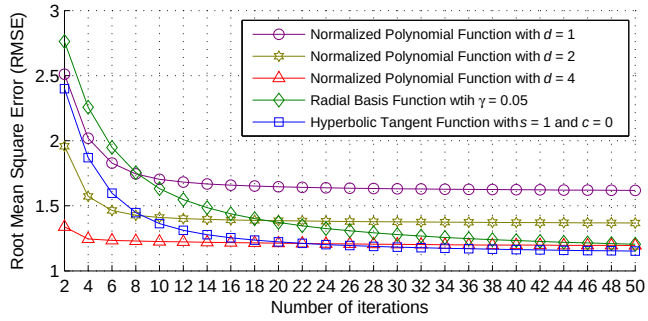
Fig. 6 depicts the prediction error of the social strength based methods for each individual subset of cold-start users with a certain number of ratings in the training set. We also have two observations: (1) As expected, the prediction error generally decreases as the number of ratings gets larger, because the prediction models can infer the preferences of users on items more accurately based on more training ratings. (2) Importantly, our KAMF+ brings the largest benefit for the cold-start users by lowering the rating prediction error significantly compared to other competitors, which validates the superiority of the kernel-based attribute-aware matrix factorization with social strength enhancement.

TABLE 7
Rating prediction error of KAMF with top-5 popular attributes in Yelp

Method	MAE	RMSE
PMF (no any attributes)	1.1150	1.5188
KAMF (Category)	1.0668	1.3956
KAMF (Geo-Coordinates)	1.0642	1.3813
KAMF (Accepts Credit Cards)	1.1029	1.5124
KAMF (Price Range)	1.0876	1.4271
KAMF (Noise Level)	1.0946	1.4873
KAMF (all attributes)	0.9388	1.1942



(a) MAE

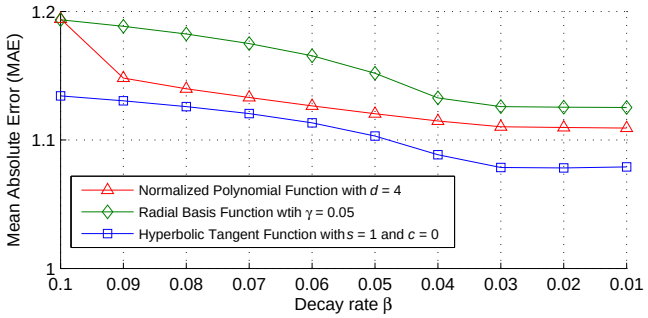


(b) RMSE

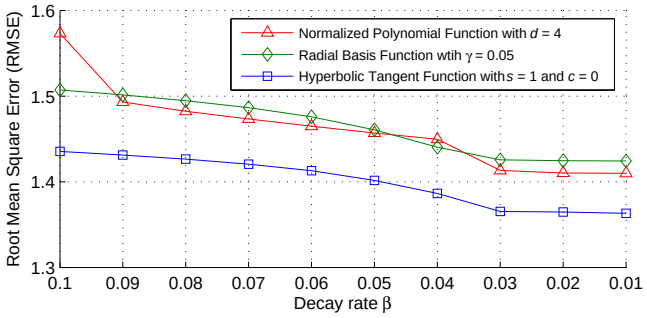
Fig. 7. Rating prediction error of KAMF (Algorithm 1) with three different kernel functions for the Yelp data set

5.2.3 Impact of Attributes

This section investigates the effect of different attributes on the performance of rating prediction. We focus on the top-5 popular attributes (TABLE 4) of the Yelp data set due to their generality. TABLE 7 depicts the performance improvement of the top-5 popular attributes against the baseline PMF that does not use any attributes. From TABLE 7, all attributes lower the rating prediction error to some degree opposed to PMF without any attributes. Specifically, the attribute of "Category" or "Geo-coordinates" helps reduce the prediction error a lot, because people are sensitive to the two attributes of items (i.e., venues) and show different preferences on them. For example, a foodie often visits restaurants to taste a variety of food while a travelling enthusiast usually visits different kinds of tourism attractions, and indoorsy persons like visiting venues around their living areas while outdoorsy persons prefer exploring new interesting places by traveling around the world. On the contrary, the attribute of "Accepts Credit Cards" improves the rating prediction performance a little, in that people are much less sensitive to this attribute. Most importantly, our KAMF uplifts the performance significantly by integrating all attributes to learn the most potential interactions among attributes adaptively.

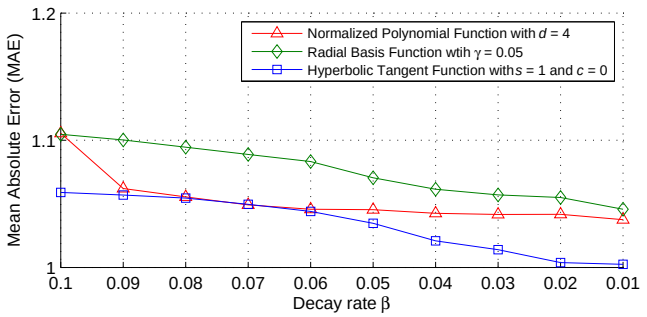


(a) MAE on cold-start users

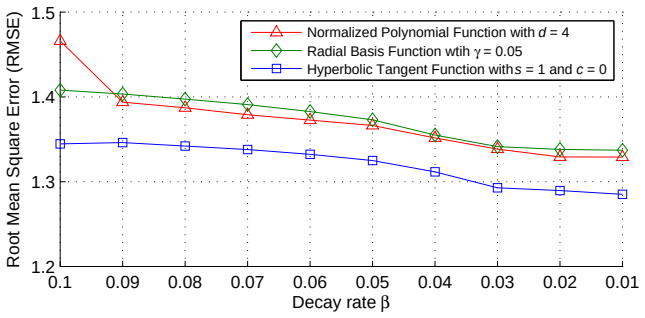


(b) RMSE on cold-start users

Fig. 8. Rating prediction error of KAMF+ (Algorithm 3) with different decay rates on cold-start users for the Yelp data set



(a) MAE on all users



(b) RMSE on all users

Fig. 9. Rating prediction error of KAMF+ (Algorithm 3) with different decay rates on all users for the Yelp data set

5.2.4 *Impact of Kernel Functions*

Heretofore, our KAMF or KAMF+ always applies the normalized polynomial kernel function with $d = 4$ in Equation (19). Here, we investigate the rating prediction error of KAMF and KAMF+ with the three kernel functions in Equations (19), (20) and (21). Due to similar results, Fig. 7 only depicts the performance of KAMF respecting the number of iterations. We conclude three important observations: (1) As an example, we study the normalized polynomial function with different values on the hyperparameter d. When adopting $d = 1$, KAMF is degraded to the linear regression without any kernel. Subsequently, KAMF reports the high prediction error. When changing to another value, e.g., $d = 2$ or $d = 4$, KAMF generates the much lower error. (2) The radial basis function ($\gamma = 0.05$) records the higher error than the normalized polynomial function ($d = 4$), but the hyperbolic tangent function ($s = 1$ and $c = 0$) is superior to the normalized polynomial function, in which the hyperparameters of the kernel functions are optimal based on cross-validation. Accordingly, the experimental results of KAMF in Sections 5.2.1 and 5.2.2 can be improved via the hyperbolic tangent kernel function. Thus, we believe the predictive ability of KAMF on ratings can be uplifted through choosing more appropriate kernel functions. (3) As the number of iterations (or the number n_{u_i} of training samples of user u_i) increases, KAMF quickly converges to the lowest prediction error regardless of the kernel functions. In particular, with the normalized polynomial function, KAMF is converged after ten iterations, so KAMF can save time by setting a small iteration number.

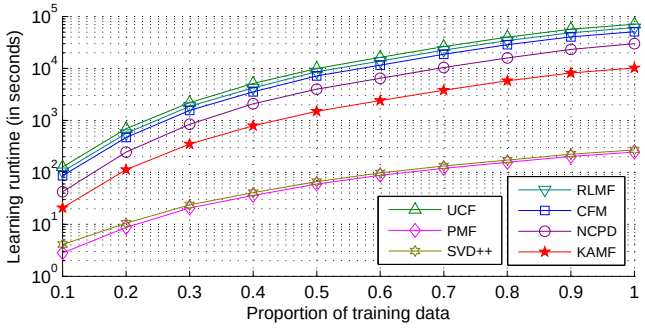
5.2.5 *Impact of Decay Rates*

Fig. 8 depicts the impact of the decay rate β in Equation (26) on the performance of KAMF+ for the cold-start users. At first, the performance of the three kernel functions in Fig. 8 is consistent with that in Fig. 7. Further, when the decay rate β becomes smaller, the prediction error of KAMF+ gradually declines and then remains stable (Fig. 8). Our explanation is that: (1) A smaller decay rate indicates a larger probability of sampling ratings for the cold-start users from their friends according to Equation (26) and these sampled ratings benefit for deducing the preferences of the cold-start users on items, so the prediction error for them decreases. (2) The cold-start users can sample a large number of ratings from their friends with a small decay rate and these ratings may cause some redundance, so the prediction error approaches to the stable state. In addition, the rating sampling probability exponentially decays as the number of current ratings increases; it will become more and more difficult for a cold-start user to acquire the ratings from her friends as time goes on.

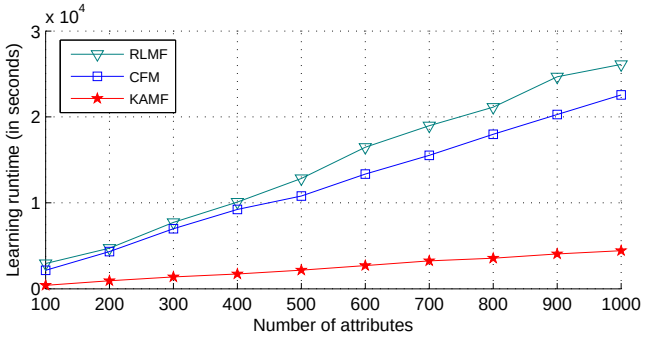
Fig. 9 further shows the overall performance for all users on the decay rate β, in which the performance trend is very similar with that for the cold-start users in Fig. 8, but the rating prediction error is lower. The reason is twofold: (1) The decay rate β has much less effect on the non-cold-start users than the cold-start users, because the non-cold-start users have a larger number of ratings that significantly reduces the rating sampling probabilities due to the exponential decay property of Equation (26). Thus, the performance for the cold-start users dominates the overall trend for all users. (2) In terms of TABLES 5 and 6, the prediction error for the non-cold-start users is lower than that for the cold-start user, so the performance on all users is better than that on the cold-start users. Based on these results, our proposed sampling-based approach allows all users to be processed in a universal way that avoids to set up a threshold on the number of ratings to define the cold-start users and achieves an adaptive balance on the effect of *a user's own ratings* and *her friends' ratings*.

5.2.6 *Computational Complexity*

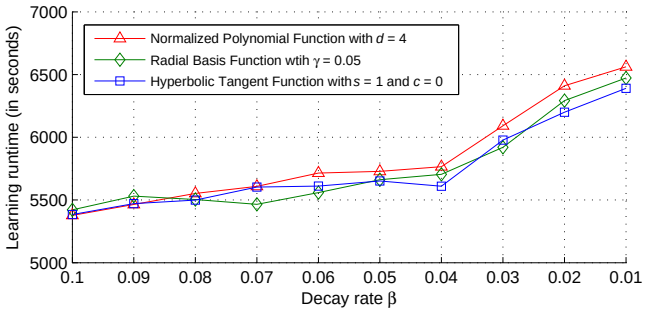
The evaluated methods were run on a machine with 3.4 GHz Intel Core i7 Processor and 16 GB RAM. Fig. 10(a) demonstrates the learning runtime of KAMF against the competitors with respect



(a) Learning runtime on proportions of training data



(b) Learning runtime on numbers of attributes of items



(c) Learning runtime on decay rates

Fig. 10. Computational complexity for the Yelp data set

to different proportions of training data (or the number n_{u_i} of training samples of user u_i). As expected, all evaluated methods require more learning runtime as the proportion of training data becomes larger. UCF learns the prediction model the most slowly, because it has the highest time complexity of $O(M^2N)$, in which M and N are the large numbers of users and items, respectively, as shown in TABLE 3. In contrast, PMF and SVD are the most time-saving method due to the lowest time complexity of $O(|\mathbf{R}|L)$ per iteration, in which $|\mathbf{R}|$ denotes the number of training ratings and L is the dimension of latent factor vectors. However, the high efficiency of PMF and SVD is at the expense of their high prediction error, since they do not take into account other information, e.g., the attributes of items. RLMF and CFM have the time complexity of $O(|\mathbf{R}|LD)$ per iteration, in which D is the dimension of attributes, i.e., $D = 1,285$ (TABLE 3) in our experiments. Thus, RLMF and CFM approximately cost one thousand times more work than PMF. NPCD with the complexity of $O(|\mathbf{R}|LC)$ per iteration also needs high cost to learn the prediction model, in which C is a constant, i.e., the number of neighbors that are considered for an item. Our KAMF consumes more time than the non-attribute-aware PMF and SVD, but it is faster than the attribute-aware RLMF, CFM and NPCD. These empirical results match to the theoretical complexity analysis in Section 3.5.

Fig. 10(b) compares the learning runtime of the attribute-aware RLMF, CFM and KAMF. Note that NPCD just applies several attributes, e.g., categories and geographical locations of items. The learning runtime of RLMF and CFM dramatically increases when they consider more attributes of items, since their time complexity $O(|\mathbf{R}|LD)$ is linear to the number D of attributes with the large slope of $|\mathbf{R}|L$. Hence, they are not scalable to high dimension of attributes. In contrast, the learning runtime of KAMF slightly inclines as the number of attributes gets larger. Actually, KAMF pre-calculates the kernel value $\mathcal{K}(\mathbf{x}_{j'}, \mathbf{x}_j)$ only once for each pair of items $v_{j'}$ and v_j rated by the same user, which costs the linear time to the number of attributes with a small slope. Given the stored kernel values, the learning algorithm for KAMF is independent of the number of attributes, as discussed in Section 3.5. Therefore, KAMF scales well into a large number of attributes.

Fig. 10(c) depicts the effect of the decay rate β in Equation (26) on the learning runtime of KAMF+ with three different kernel functions. As the decay rate decreases from 0.1 to 0.04, the learning runtime of KAMF+ gradually rises, and then it steeply climbs when the decay rate is smaller than 0.04. The reason is that: the rating sampling probability gets larger with the decrease of the decay rate, especially when the decay rate is smaller than 0.04. Subsequently, a large amount of ratings are sampled for the cold-start users, which causes much time to update the model parameters of the cold-start users. In practice, the decay rate should not be set to a very small value which greatly increases the learning runtime but brings very limited benefit on the predictive error, as shown in Fig. 8.

6 Conclusion and Future Work

In this paper, we have proposed a kernel-based attribute-aware matrix factorization model KAMF to integrate the attribute information of items into matrix factorization. KAMF exploits the rich attributes of items to uncover the nonlinear interactions among attributes, users and items, which alleviate the data sparsity effect of the user-item rating matrix on rating prediction. We also have devised an incremental learning algorithm to estimate the model parameters of KAMF. KAMF has been further extended to address the user cold-start problem by utilizing the social links between users, although KAMF can deal with the item cold-start problem by nature. Finally, the experimental results on the two large-scale real-world data sets from Yelp and MovieLens show that KAMF is significantly superior to the current state-of-the-art rating prediction techniques. As a part of our future study, we plan to design more appropriate kernels for our kernel-based attribute-aware matrix factorization model and investigate more sophisticated methods for measuring the social strength between users in order to improve the performance of KAMF.

Acknowledgements

This research was partially supported by the City University of Hong Kong under Grant 7004217 and the National Natural Science Foundation of China under Grant 71472158 and 71490725.

REFERENCES

[1] Y. Koren, "Factorization meets the neighborhood: a multifaceted collaborative filtering model," in *ACM KDD*, 2008.

[2] Y. Koren, R. Bell, and C. Volinsky, "Matrix factorization techniques for recommender systems," *Computer*, vol. 42, no. 8, pp. 30–37, 2009.

[3] C. Cheng, F. Xia, T. Zhang, I. King, and M. R. Lyu, "Gradient boosting factorization machines," in *ACM RecSys*, 2014.

[4] T. V. Nguyen, A. Karatzoglou, and L. Baltrunas, "Gaussian process factorization machines for context-aware recommendations," in *ACM SIGIR*, 2014.

[5] S. Rendle, "Factorization machines with libFM," *ACM TIST*, vol. 3, no. 3, pp. 57:1–57:22, 2012.

[6] S. Rendle, Z. Gantner, C. Freudenthaler, and L. Schmidt-Thieme, "Fast context-aware recommendations with factorization machines," in *ACM SIGIR*, 2011.

[7] M. Mohri, A. Rostamizadeh, and A. Talwalkar, *Foundations of Machine Learning*. The MIT Press, 2012.

[8] J. L. Herlocker, J. A. Konstan, A. Borchers, and J. Riedl, "An algorithmic framework for performing collaborative filtering," in *ACM SIGIR*, 1999.

[9] B. Sarwar, G. Kapypis, J. Konstan, and J. Riedl, "Item-based collaborative filtering recommendation algorithms," in *WWW*, 2001.

[10] M. Ye, P. Yin, W.-C. Lee, and D.-L. Lee, "Exploiting geographical influence for collaborative point-of-interest recommendation," in *ACM SIGIR*, 2011.

[11] B. Liu, H. Xiong, S. Papadimitriou, Y. Fu, and Z. Yao, "A general geographical probabilistic factor model for point of interest recommendation," *IEEE TKDE*, vol. 27, no. 5, pp. 1167–1179, 2015.

[12] R. Salakhutdinov and A. Mnih, "Probabilistic matrix factorization," in *NIPS*, 2008.

[13] H. Yang, G. Ling, Y. Su, R. Lyu, and I. King, "Boosting response aware model-based collaborative filtering," *IEEE TKDE*, vol. 27, no. 8, pp. 2064–2077, 2015.

[14] D. D. Lee and H. S. Seung, "Learning the parts of objects by non-negative matrix factorization," *Nature*, vol. 401, no. 6755, pp. 788–791, 1999.

[15] B. Liu, Y. Fu, Z. Yao, and H. Xiong, "Learning geographical preferences for point-of-interest recommendation," in *ACM KDD*, 2013.

[16] C. Liu, H.-C. Yang, J. Fan, L.-W. He, and Y.-M. Wang, "Distributed nonnegative matrix factorization for web-scale dyadic data analysis on mapreduce," in *WWW*, 2010.

[17] M. Jiang, P. Cui, F. Wang, W. Zhu, and S. Yang, "Scalable recommendation with social contextual information," *IEEE TKDE*, vol. 26, no. 11, pp. 2789–2802, 2014.

[18] H. Ma, "An experimental study on implicit social recommendation," in *ACM SIGIR*, 2013.

[19] H. Ma, D. Zhou, C. Liu, M. R. Lyu, and I. King, "Recommender systems with social regularization," in *ACM WSDM*, 2011.

[20] J. Noel, S. Sanner, K.-N. Tran, P. Christen, L. Xie, E. V. Bonilla, E. Abbasnejad, and N. Della Penna, "New objective functions for social collaborative filtering," in *WWW*, 2012.

[21] Y. Shen and R. Jin, "Learning personal + social latent factor model for social recommendation," in *ACM KDD*, 2012.

[22] S.-H. Yang, B. Long, A. Smola, N. Sadagopan, Z. Zheng, and H. Zha, "Like like alike: Joint friendship and interest propagation in social networks," in *WWW*, 2011.

[23] D. Yang, D. Zhang, Z. Yu, and Z. Wang, "A sentiment-enhanced personalized location recommendation system," in *ACM HT*, 2013.

[24] J. Tang, X. Hu, and H. Liu, "Social recommendation: a review," *Social Network Analysis and Mining*, vol. 3, no. 4, pp. 1113–1133, 2013.

[25] R. Gemulla, E. Nijkamp, P. J. Haas, and Y. Sismanis, "Large-scale matrix factorization with distributed stochastic gradient descent," in *ACM KDD*, 2011.

[26] A. Elbadrawy and G. Karypis, "User-specific feature-based similarity models for top-n recommendation of new items," *ACM TIST*, vol. 6, no. 3, pp. 33:1–33:20, 2015.

[27] D. Tang, B. Qin, T. Liu, and Y. Yang, "User modeling with neural network for review rating prediction," in *IJCAI*, 2015.

[28] X. Wang and Y. Wang, "Improving content-based and hybrid music recommendation using deep learning," in *ACM MM*, 2014.

[29] L. Baltrunas and F. Ricci, "Context-based splitting of item ratings in collaborative filtering," in *ACM RecSys*, 2009.

[30] C. Chen, X. Zheng, Y. Wang, F. Hong, and Z. Lin, "Context-aware collaborative topic regression with social matrix factorization for recommender systems," in *AAAI*, 2014.

[31] J. Tang, H. Gao, X. Hu, and H. Liu, "Context-aware review helpfulness rating prediction," in *ACM RecSys*, 2013.

[32] H. Gao, J. Tang, X. Hu, and H. Liu, "Content-aware point of interest recommendation on location-based social networks," in *AAAI*, 2015.

[33] D. Agarwal and B.-C. Chen, "Regression-based latent factor models," in *ACM KDD*, 2009.

[34] Z. Gantner, L. Drumond, C. Freudenthaler, S. Rendle, and L. Schmidt-Thieme, "Learning attribute-to-feature mappings for cold-start recommendations," in *IEEE ICDM*, 2010.

[35] X. Liu and K. Aberer, "SoCo: A social network aided context-aware recommender system," in *WWW*, 2013.

[36] Y. Shi, A. Karatzoglou, L. Baltrunas, M. Larson, and A. Hanjalic, "CARS^2: Learning context-aware representations for context-aware recommendations," in *ACM CIKM*, 2014.

[37] L. Hu, A. Sun, and Y. Liu, "Your neighbors affect your ratings: On geographical neighborhood influence to rating prediction," in *ACM SIGIR*, 2014.

[38] J.-D. Zhang and C.-Y. Chow, "iGSLR: Personalized geo-social location recommendation - a kernel density estimation approach," in *ACM SIGSPATIAL*, 2013.

[39] ——, "CoRe: Exploiting the personalized influence of two-dimensional geographic coordinates for location recommendations," *Information Sciences*, vol. 293, pp. 163–181, 2015.

[40] ——, "GeoSoCa: Exploiting geographical, social and categorical correlations for point-of-interest recommendations," in *ACM SIGIR*, 2015.

[41] J.-D. Zhang, C.-Y. Chow, and Y. Li, "iGeoRec: A personalized and efficient geographical location recommendation framework," *IEEE TSC*, vol. 8, no. 5, pp. 701–714, 2015.

[42] J.-D. Zhang and C.-Y. Chow, "Spatiotemporal sequential influence modeling for location recommendations: A gravity-based approach," *ACM TIST*, vol. 7, no. 1, pp. 11:1–11:25, 2015.

[43] ——, "TICRec: A probabilistic framework to utilize temporal influence correlations for time-aware location recommendations," *IEEE TSC, accepted to appear*, 2015.

[44] C. M. Bishop, *Pattern Recognition and Machine Learning*. Springer, 2006.

[45] M. Jiang, P. Cui, X. Chen, F. Wang, W. Zhu, and Y. Shiqiang, "Social recommendation with cross-domain transferable knowledge," *IEEE TKDE*, vol. 27, no. 11, pp. 3084–3097, 2015.

[46] Yelp, "Challenge Data Set," http://www.yelp.com/dataset_challenge, 2015.

[47] F. M. Harper and J. A. Konstan, "The movielens datasets: History and context," *ACM TiiS*, vol. 5, no. 4, pp. 19:1–19:19, 2015.

Jia-Dong Zhang received the M.Sc. degree from Yunnan University, China, in 2009, and the Ph.D. degree from City University of Hong Kong in 2015. He is currently a postdoctoral fellow in Department of Computer Science, City University of Hong Kong. His research work has been published in premier conferences (e.g., *ACM SIGIR*, *CIKM* and *SIGSPATIAL*), transactions (e.g., *ACM TIST*, *IEEE TKDE*, *TDSC TSC* and *TITS*), and journals (e.g., *Pattern Recognition* and *Information Sciences*). His research interests include data mining, location-based services and location privacy.

Chi-Yin Chow received the M.S. and Ph.D. degrees from the University of Minnesota-Twin Cities, USA in 2008 and 2010, respectively. He is currently an assistant professor in Department of Computer Science, City University of Hong Kong. His research interests include big data analytics, data management, GIS, mobile computing, location-based services, and data privacy. He is the co-founder and co-chair of the ACM SIGSPATIAL MobiGIS 2012 to 2016, and the editor of the ACM SIGSPATIAL Newsletter. He received the VLDB "10-year award" in 2016, and the best paper awards in ICA3PP 2015 and IEEE MDM 2009.

Jin Xu received his Ph.D and M.Phil degree in management science and engineering from Southwest Jiaotong University, Chengdu, China. He is currently an Associate Professor with the Department of Management Information Systems & Electronic Commerce, School of Economics and Management, Southwest Jiaotong University. His research work has been published in journals such as IEEE TITS, Transportation Research Part C: Emerging Technologies, Expert Systems with Applications. His research interests include intelligent transportation systems, project management systems, big data and knowledge discovery.