Expert Systems With Applications 168 (2021) 114436

Contents lists available at ScienceDirect

Expert Systems With Applications

journal homepage: www.elsevier.com/locate/eswa

Kernel meets recommender systems: A multi-kernel interpolation for matrix completion

Zhaoliang Chen [a,b], Wei Zhao [c], Shiping Wang [a,b,*]

[a] College of Mathematics and Computer Science, Fuzhou University, Fuzhou 350116, China
[b] Fujian Provincial Key Laboratory of Network Computing and Intelligent Information Processing, Fuzhou University, Fuzhou 350116, China
[c] Department of Architecture and Civil Engineering, City University of Hong Kong, Hong Kong SAR, China

ARTICLE INFO

Keywords:
Recommender systems
Kernel learning
Multi-kernel learning
Matrix completion
Matrix interpolation

ABSTRACT

A primary research direction for recommender systems is matrix completion, which attempts to recover the missing values in a user–item rating matrix. There are numerous approaches for rating tasks, which are mainly classified into latent factor models and neighborhood-based models. Most neighborhood-based models seek similar neighbors by computing similarities in the original data space for final predictions. In this paper, we propose a new neighborhood-based interpolation model with a kernelized matrix completion framework, with the impact weights provided by neighbors computed in a new Hilbert space containing more features. In our model, the kernel function is combined with a similarity measurement to achieve better approximation for unknown ratings. Furthermore, we extend our model with a non-linear multi-kernel framework which learns weights automatically to improve the model. Finally, we conduct extensive experiments on several real-world datasets. The outcomes show that the proposed methods work effectively and improve the performance of the rating prediction task compared to both the traditional and state-of-the-art approaches.

1. Introduction

Recommender systems are widely employed in various spheres and have become a popular research topic in decades (Hwang et al., 2016; Qian et al., 2019; Wang, Zhou and Lu, 2019). For instance, the famous media-service provider Netflix held the Netflix Prize competition to explore algorithms to predict user ratings of movies. This task is also considered as matrix completion issue that retrieves missing ratings in a rating matrix. Table 1 provides a simple example of the user–item rating matrix waiting to be completed. Most of the ratings in this table are missing. In real-world datasets, the rating matrices are even more sparse, which leads to the cold-start problem in recommender systems. The primary target of matrix completion is to retrieve the missing ratings in the user–item rating matrix. A classical solution for matrix completion is nonnegative matrix factorization (NMF) (Lee & Seung, 1999), which factorizes the incomplete matrix into two low-rank matrices. A variety of methods have been proposed for matrix completion. Kang et al. (2016) completed the rating matrix based on low-rank assumption, which adopted a nonconvex rank relaxation to achieve a better rank approximation. Xue et al. (2017) leveraged two parallel deep neural networks to factorize a user–item interaction matrix and predict the unknown ratings. Inspired by word embedding models, Liang et al. (2016) jointly factorized the user–item interaction matrix and the item–item co-occurrence matrix with shared item latent factors.

Collaborative filtering (CF) has been widely investigated by many researchers and is a mature application that has been utilized extensively in industry (Chen et al., 2017, 2019; Wang, Zhou, Chen et al., 2019). The algorithm attempts to determine the hidden relationships between users and items in a data-driven method and recommends similar items to users with the same interests. There are two primary types of CF, latent factor models (LFMs) and neighborhood-based models (NBMs). LFMs discover the latent features of users or items and project them into feature vectors that are generally of low-rank. Matrix factorization (MF) is a typical method of LFMs, which factorizes the raw rating matrix into two low-rank matrices known as the user latent matrix and the item latent matrix. The unknown ratings are predicted by the dot product of the corresponding latent vectors. A large number of MF-based models have been proposed in decades. For example, Koren (2008) applied a singular value decomposition (SVD) based model named SVD++ that considered the influence of the neighborhood. Ning and Karypis (2011) presented a sparse linear model (SLIM) that explored an item–item similarity matrix by factorizing the original user–item interaction matrix. Wang et al. (2018) employed

[*] Corresponding author at: College of Mathematics and Computer Science, Fuzhou University, Fuzhou 350116, China.
E-mail addresses: chenzl23@outlook.com (Z. Chen), wzhao22@cityu.edu.hk (W. Zhao), shipingwang@fzu.edu.cn (S. Wang).

https://doi.org/10.1016/j.eswa.2020.114436
Received 14 December 2019; Received in revised form 9 August 2020; Accepted 1 December 2020
Available online 3 December 2020

Table 1
Simple example of a rating matrix for completion.

	I_1	I_2	I_3	I_4	I_5
U_1			3	2	
U_2	1	4		3	3
U_3		1			
U_4		5		2	

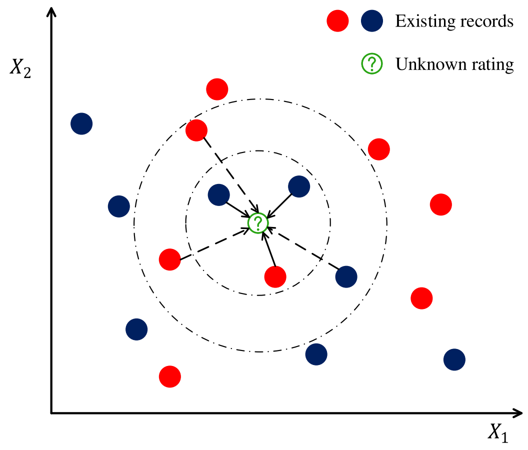

Fig. 1. Basic concept of NBMs. Different data points contribute together to the prediction point based on weights. The points inside the circle are neighbors of the predicting point, whose influence is computed by the similarity measures. Closer points have greater impact on prediction.

a confidence-aware MF framework to optimize both the precision of rating estimation and prediction confidence.

Different from LFMs, NBMs aim to explore similar users or items by computing the similarities among them and make estimations by considering the influence or contribution of each neighbor. A classic algorithm of NBMs is the k-nearest neighbors (KNN) approach (Sarwar et al., 2001). The item-based KNN models calculate the similarities among items and then sort them for top-k recommendations. For example, Park et al. (2015) proposed a KNN-based CF model named reversed CF, which utilized a KNN graph to locate the KNN of the rated items. As for rating tasks, models commonly compute the unknown ratings with a weighted average of other existing ratings. Different neighbors contribute to the final estimations of target ratings based on the similarities between them. Fig. 1 illustrates this schema. By measuring the similarities or distance between the predicting point and existing points, we can select the most relative points to estimate the value of unknown point. Accordingly, a smaller interval between data points should lead to stronger influence, which means that similar points work better on the recovery of unknown data.

Kernel learning is a technique that applies kernel functions to map the raw data into a high-dimensional space without computing the corresponding projection functions. It is best known in support vector machines (SVMs), which make raw linearly inseparable data separable in a high-dimensional space. Among various kernel functions, radial basis function (RBF) kernels such as the Gaussian kernel are the most widely used, which are often leveraged to train RBF networks. RBF kernels have been extensively applied in recommender systems and improve the performance of MF-based approaches (Liu et al., 2016; Pal & Jenamani, 2018; Zhou et al., 2012). Because RBF kernels are able to calculate the similarities among samples, and the performance of NBMs is closely related to the similarity metric, we also consider it as a powerful technique for improving NBMs. Actually, RBF kernels have been applied in many fields like feature selection (Kuo et al., 2013), clustering (Cruz et al., 2016) and image processing (Romani et al., 2019) due to the ability to measure similarities. Nevertheless, to our knowledge, limited studies have been devoted to the application of RBF kernels in NBMs for recommender system databases.

In this paper, we propose a new kernel-based matrix completion (KMC) framework for recommender systems, which aims to solve the rating tasks with NBMs for a user–item interaction matrix. The model applies RBF kernels that are reformulated by similarity measures and provides estimation for a user on a specific item. Inspired by the interpolation condition, the proposed KMC is a closed-form solution calculated by kernel matrices. This speeds up the rating predictions for a specific user or item. Moreover, we improve this model with a multi-kernel framework for KMC (M-KMC) to merge different features in different latent spaces generated by diverse kernels. Different from extensively used linear combination of kernels, M-KMC applied a non-linear auto-weighted strategy to merge different kernels. In summary, our contributions are as follows:

1. We propose a kernelized model with a closed-form solution for matrix completion, which applies the interpolation method for rating prediction.
2. In our proposed model, the similarity metric is combined with the Gaussian kernel to compute the weights of neighbors, which generates a more precise approximation for unknown ratings.
3. M-KMC is presented with the multi-kernel framework, which adaptively adjusts the weights of the multiple kernel functions and improves the performance of KMC.
4. We conduct rich experiments on KMC and M-KMC and discuss the effect of different parameters. Our model achieves the performance that is competitive with or superior to the traditional and state-of-the-art models.

2. Related work

2.1. Neighborhood-based models

NBMs are commonly used techniques in recommender systems. These models compute the similarities or correlations among different users or items, based on rating records or extracted latent features. A common metric is known as the cosine similarity, which calculates the cosine value between two vectors. For a user-based similarity measure, assume that $I_{uv} = \{1, \dots, n\}$ is the item set that both user u and user v have co-rated, then vector $Y_u = \{y_{u1}, \dots, y_{un}\}$ and vector $Y_v = \{y_{v1}, \dots, y_{vn}\}$ are the rating vectors of user u and v. The cosine similarity $cos(u, v)$ is computed by

$$cos(u,v) = \frac{Y_u \cdot Y_v}{\|Y_u\| \, \|Y_v\|} = \frac{\sum_{i\in I_{uv}} y_{ui}y_{vi}}{\sqrt{\sum_{i\in I_{uv}} y_{ui}^2}\sqrt{\sum_{i\in I_{uv}} y_{vi}^2}}. \tag{1}$$

The scale of the cosine similarity is $[0, 1]$, and a higher cosine value corresponds to a higher correlation. However, for the reason that different users may have their own rating scales, such computations may be inaccurate. To handle this problem, a solution is considering the deviation from the average value, which is known as decentralization. Then Eq. (1) becomes

$$pearson(u,v) = \frac{\sum_{i\in I_{uv}} (y_{ui} - \bar{y}_u)(y_{vi} - \bar{y}_v)}{\sqrt{\sum_{i\in I_{uv}} (y_{ui} - \bar{y}_u)^2}\sqrt{\sum_{i\in I_{uv}} (y_{vi} - \bar{y}_v)^2}}, \tag{2}$$

where $\bar{y}_u$ and $\bar{y}_v$ are the averages of user ratings on co-rated item set I_{uv}. Eq. (1) is also called Pearson correlation coefficient. This decentralization process also considers the various rating scales of users, because different users may rate items with the same interests differently. As a result, the distance between individuals with the same interests but rating differently is closer, so that a more precise correlation is measured.

Similar to the cosine similarity metric, the mean square difference (MSD) considers the square differences between user u and user v with the co-rated item set I_{uv}, which is defined by

$$MSD(u,v) = \frac{1}{|I_{uv}|} \cdot \sum_{i \in I_{uv}} (y_{ui} - y_{vi})^2. \tag{3}$$

The similarity measures for items are analogous to those for users. Contrary to cosine similarity, a higher MSD value corresponds to a lower similarity. Thus there is a variation version of MSD to compute similarity s_{ij} so that it is analogous to other similarity metrics, as Eq. (4) shows:

$$s_{uv} = \frac{1}{MSD(u,v)+1}. \tag{4}$$

In the following sections, we adopt MSD to measure the similarities among different users and items.

2.2. Kernel learning

Kernel learning is a useful technique in machine learning because it can project non-linear divided data in low-dimensional space to high-dimensional space where they can be linearly separated. It is extensively applied in SVMs to transform the non-linear problem into a linear one. Assume that H is a Hilbert space defined on $\Omega \in \mathbb{R}^d$ with inner product operation $\langle \cdot, \cdot \rangle_H$, we have equivalent computation

$$K(x,x') = \langle \varphi(x), \varphi(x') \rangle_H \tag{5}$$

for a given kernel function $K(\cdot,\cdot)$, where operation $\varphi(\cdot) \in H$ is the projection to Hilbert space for a single variable. The formula demonstrates that the product operation in Hilbert space is equivalent to the function $K(x,x')$ in low-dimensional space without defining the mapping operation $\varphi(\cdot)$. There are many kernel functions, including linear, polynomial and RBF kernels. Among these, RBF kernels are extensively used because of their capacity to map the original data in space with an infinite number of dimensions. The Gaussian kernel is a representative RBF kernel, formulated as

$$K(x,x') = e^{-\frac{\|x-x'\|^2}{2\sigma^2}}, \tag{6}$$

where symbol σ is a bandwidth parameter that controls the spread of the Gaussian kernel. A large value of σ indicates that the kernel has a wide spread, whereas a small σ causes a narrow spread of the kernel function. Term $\|x - x'\|$ is the Euclidean distance between the two variables. If set $c = \frac{1}{\sqrt{2}\sigma}$, Eq. (6) is simplified to

$$K(x,x') = K(\|x-x'\|) = e^{-(c\|x-x'\|)^2}. \tag{7}$$

Because the value of the Gaussian kernel function ranges from zero to one (when $x = x'$), and decreases as the distance between x and x' increases, it is regarded as a similarity metric.

Different kernels generate different feature spaces and, even the same kernel with different parameters forms multiple spaces. This inspires us to apply the multi-kernel learning (MKL) framework in our model to merge different feature spaces. The fundamental linear combination of q different kernels is

$$K'(x,x') = \sum_{t=1}^{q} \alpha_t K_t(x,x'), \quad s.t. \sum_{t=1}^{q} \alpha_t = 1, \alpha_t \geqslant 0, \tag{8}$$

where $K_t(\cdot,\cdot)$ is the tth kernel function and α_t is the weight of the tth kernel.

Many studies have solved matrix completion problems with a kernelized framework in recent years. Lee et al. (2013) applied smooth kernel functions as a non-negative symmetric unimodal function in their local low-rank matrix approximation (LLORMA) model for recommender systems. Liu et al. (2016) proposed kernelized matrix factorization for recommender systems, and applied multi-kernel framework which considered linear combination of different kernel spaces to improve the performance. Li et al. (2019) presented a deep matrix completion (DMC) with adversarial kernel embedding to conduct a matrix completion in Hilbert space for recommender systems. Chen and Li (2019) investigated memory-efficient kernel PCA to solve low-rank approximation and clustering problems. The kernel trick is also applied to solve high-rank matrix completion problems in terms of various applications including subspace clustering (Fan & Chow, 2018; Fan et al., 2020). However, most of these methods directly applied original kernel functions in their works. In this paper, we consider reformulating the widely used RBF kernel function with similarity metric, to learn a better representation for recommender system in Hilbert space. Moreover, instead of the commonly applied linear combination of kernel functions, a non-linear MKL framework is proposed which learns weights automatically.

3. Our proposed models

Before the description of our proposed methods, we first provide explanations for primary mathematical notations used in this section. The set $\mathbb{R}^{m\times n}$ is the space of $m\times n$ dimensional real matrix. Assume there are m users and n items in the dataset, all observed user–item rating pairs are stored in set $\Omega = \{(u,i)|y_{ui}\ is\ observed\}$, and set $\bar{\Omega}$ represents the pairs where ratings are missing. Then

$$Y_{ui} = \begin{cases} y_{ui} & (u,i) \in \Omega \\ null & (u,i) \in \bar{\Omega}. \end{cases} \tag{9}$$

denotes the user–item interaction matrix. The similarity between two data points is measured by s. Functions $p(x_k)$ and $f(x_k)$ denote the interpolation value and real value of point x_k, respectively. As we have discussed before, $K(x,x')$ represents the kernel function. Therefore $\mathrm{K} = [K(x_i,x_j)]_{i,j=1}^{L} \in \mathbb{R}^{L\times L}$ denotes the kernel matrix where $\mathrm{K}_{ij} = K(x_i,x_j)$. The neighborhood weight vector for recovering x_k is denoted by $\xi(x_k)$, and the weights for multi-kernel learning are measured by $\{\alpha_t\}_{t=1}^{q}$.

3.1. Similarity metric and neighbor selection

In the field of image interpolation and image super-resolution, we usually sample the pixel from the pixels surrounding it. However, for the recommender systems, there are no spatial relationships for the rating matrix. Therefore, we have to select several neighbors for a certain data point $x = (u,i) \in \Omega$ by computing the similarities between them. First, we define the similarity $s_{xx'}$ between $x = (u,i)$ and $x' = (v,j)$ with the squared Euclidean distance:

$$s_{xx'} = \sqrt{s_{uv}^2 + s_{ij}^2}. \tag{10}$$

Here we adopt the MSD defined in Eq. (4) to compute s_{uv} and s_{ij}. Because similarity metrics are inaccurate when co-rated sets are small, we include a penalty term δ to reduce the magnitude of these similarity weights. The adjusted similarity metric is

$$s'_{uv} = \frac{min\{|I_{uv}|,\delta\}}{\delta} s_{uv}, \tag{11}$$

where the influence of δ decreases as the size of the set I_{uv} increases, and disappears when the size of the set I is large enough ($|I_{uv}| \geqslant \delta$). Only when there are enough users who have both rated two items do the punishment disappears. Finally, we obtain the similarity of neighbor point x' by considering the punishment in Eq. (11), as defined by $s_{xx'} = \sqrt{s'^2_{uv} + s'^2_{ij}}$.

To select suitable neighbors for interpolations, we sorted the known data points in descending order from the candidate points. To decrease the number of alternative points and the cost of computations, we preferentially consider the points in the same row or column as being the estimation point, that is, only the ratings rated by the same users or on the same items are preferentially considered and added to the candidate points. If there are not enough neighbors, we randomly sample the data points and place them in candidate points to guarantee the ratings are predicted by enough neighbors.

3.2. Local kernel-based approximation

To apply the kernel trick in the matrix approximation, we project the original data into the Hilbert space and compute the weighted contributions of neighbors. This concept has been proposed in many other fields. For example, in network science, Martinčić-Ipšić et al. (2017) employed weighted similarity metrics for link prediction on Twitter. Yuan et al. (2019) also presented a graph kernel based link prediction method by measuring user similarity. Assume that there is a set of neighbors $\{x_t, f(x_t)\}_{t=1}^{L}$, we apply a data-dependent linear function to represent each data x_k, as the following equation shows:

$$\begin{aligned} p(x_k) &= \sum_{t=1}^{L} \lambda_t K(x_k, x_t) \\ &= [K(x_k, x_1) \cdots K(x_k, x_L)][\lambda_1 \cdots \lambda_L]^T, \end{aligned} \tag{12}$$

where λ_t is the weight of the tth neighbor. Because there are no spatial relations inside the rating matrix, the RBF kernel $K(x, x')$ is computed with similarities defined in Eq. (10), instead of with the Euclidean distance. Then the Gaussian kernel function applied in our framework is defined by

$$K(x, x') = K(s_{xx'}) = e^{-(cs_{xx'})^2}. \tag{13}$$

According to the interpolation condition, for all known data points, the value of interpolation and the real value should be equal, that is,

$$p(x_k) = f(x_k), x_k \in \Omega. \tag{14}$$

Therefore, considering both Eqs. (12) and (14), we have

$$f(x_k) = [K(x_k, x_1) \cdots K(x_k, x_L)][\lambda_1 \cdots \lambda_L]^T. \tag{15}$$

For all L known neighbors $x_1, \ldots, x_L$, then we compute the weight vector $\lambda = [\lambda_1 \cdots \lambda_L]^T$ with closed-form solution, as shown below:

$$\begin{aligned} \lambda &= \begin{bmatrix} K(x_1, x_1) & \cdots & K(x_1, x_L) \\ \vdots & \ddots & \vdots \\ K(x_L, x_1) & \cdots & K(x_L, x_L) \end{bmatrix}^{-1} \begin{bmatrix} f(x_1) \\ \vdots \\ f(x_L) \end{bmatrix} \\ &= [K(x_i, x_j)]_{i,j=1}^{L}{}^{-1} F(x_k), \end{aligned} \tag{16}$$

where $F(x_k) = [f(x_1), \ldots, f(x_L)]^T$ is the vector of the groundtruth of L neighbors. After obtaining λ, the unknown ratings are computed by substituting Eq. (16) into Eq. (12), as written in Eq. (17):

$$p(x_k) = [K(x_k, x_i)]_{i=1}^{L} [K(x_i, x_j)]_{i,j=1}^{L}{}^{-1} F(x_k), \tag{17}$$

where term $[K(x_k, x_i)]_{i=1}^{L} [K(x_i, x_j)]_{i,j=1}^{L}{}^{-1}$ is the final interpolation weight vector for x_k. Let $\xi(x_k) = [K(x_k, x_i)]_{i=1}^{L} [K(x_i, x_j)]_{i,j=1}^{L}{}^{-1}$ denote the function that calculates the final contribution of L neighbors on unknown point x_k, the localized estimation for the unknown rating is computed by

$$p(x_k) = \xi(x_k) F(x_k). \tag{18}$$

The outputs of the model are only computed by the nearest neighbors. Algorithm 1 describes the steps for rating interpolations on an unrated data point x_k. Similarity matrices $S_U \in \mathbb{R}^{m \times m}$ and $S_I \in \mathbb{R}^{n \times n}$ are calculated in advance for subsequent computations including neighbor selection and kernel matrix generation procedures.

Fig. 2 briefly illustrates the process of our KMC model. The proposed KMC first searches for neighbors according to the pre-calculated similarity matrices, and then computes the vector $\xi(x_k)$ and $F(x_k)$ for specific x_k with selected neighbors. Finally the recovery of the unknown data point is measured by Eq. (18). The steps of the algorithm correspond to closed-form solution, without training any extra parameters. The primary time consuming components of the algorithm are the steps to compute the similarity matrices, finding neighbors and calculating kernel matrices. For similarity computations, the time consumption is related to the number of users and items. With m users and n items, the time complexity is $O(max(m^2 n, n^2 m))$, and the memory complexity is $O(max(m^2, n^2))$. Because the number of neighbors is usually small, localized computation for L neighbors is effective, and the time complexity for kernel generation is $O(L^2)$. As the computation for $\xi(x_k)$ contains inverse computation, the model only works when the kernel matrix $[K(x_i, x_j)]_{i,j=1}^{L}$ is invertible. Later we will discuss the influence of this problem through experiments.

Algorithm 1 Interpolation Algorithm for KMC

```
Input:
    $Y \in \mathbb{R}^{m \times n}$: original rating matrix,
    $L$: number of neighbors,
    $x_k = (u, i)$: predicting data point.
Output:
    $p(x_k)$: prediction for data point $x_k = (u, i)$.
1: Generate similarity matrices $S_U$ and $S_I$.
2: Find out $L$ neighbors of data point $x_k$.
3: Generate vector of neighborhood real ratings $F(x_k)$.
4: Compute kernel matrices $[K(x_k, x_i)]_{i=1}^{L}$ and $[K(x_i, x_j)]_{i,j=1}^{L}$.
5: Compute $\xi(x_k) = [K(x_k, x_i)]_{i=1}^{L} [K(x_i, x_j)]_{i,j=1}^{L}{}^{-1}$.
6: Compute $p(x_k)$ using Eq. (18).
7: return $p(x_k)$.
```

3.3. Multi-kernel framework

In this subsection, we improve our framework by computing $\xi(x)$ with multiple kernels, which is named M-KMC. Fig. 3 illustrates the structure of M-KMC. It can be seen from the figure that M-KMC requires the same neighbor selection procedures as that of KMC. However, it adds a new step to calculate more kernel matrices with multiple kernel functions, and then merges them with a non-linear combination. Associated with combined kernel spaces, the new weight vector $\xi(x_k)$ can be computed. Finally, the ratings are predicted by the dot product of neighborhood weight vector $\xi(x_k)$ and known rating vector $F(x_k)$ as in KMC.

The linear multi-kernel framework defined in Eq. (8) considers the weighted average of different kernels. However, this linear combination may result in the phenomenon that only one kernel is selected when one $\alpha_t = 1$ and $\alpha_t = 0$ otherwise. In this paper, we use a trick to improve this framework based on the methodology described by Wang et al. (2007). The M-KMC introduces a new parameter r and let $r \geqslant 1$, as shown below:

$$\begin{aligned} K'(x, x') &= \sum_{t=1}^{q} \alpha_t^r K_t(x, x') \\ s.t. & \sum_{t=1}^{q} \alpha_t = 1, \alpha_t \geqslant 0, \end{aligned} \tag{19}$$

where $\sum_{t=1}^{q} \alpha_t^r$ reaches its minimum when $\alpha_t = 1/q$ with the constraint $\sum_{t=1}^{q} \alpha_t = 1$. This potentially makes α_t similar to each other so that each kernel contributes to the final model instead of only one working. When $r = 1$, the formula is equivalent to the linear combination of multiple kernel functions. To simplify the constraints in Eq. (19), we let $\alpha_t = \frac{|w_t|}{\sum_{i=1}^{q} |w_i|}$ to ensure that the sum of α_t equals 1, where w_i is the weight of the i_{th} kernel. With aforementioned analysis, we compute $\xi(x)$ by

$$\xi(x_k) = \sum_{t=1}^{q} \alpha_t^r [K_t(x_k, x_i)]_{i=1}^{L} \left(\sum_{t=1}^{q} \alpha_t^r [K_t(x_i, x_j)]_{i,j=1}^{L}\right)^{-1}. \tag{20}$$

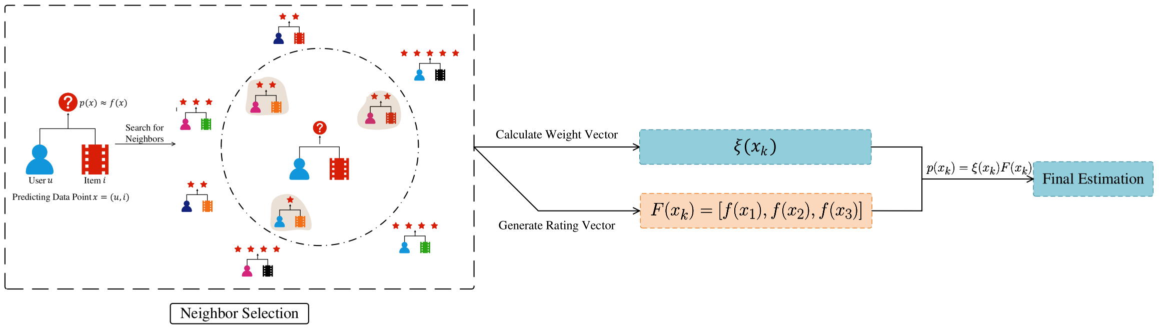

Fig. 2. Structure of the proposed KMC. Different neighbors of rating records contribute to the estimation in the Hilbert space mapped by kernel function $K(x, x')$.

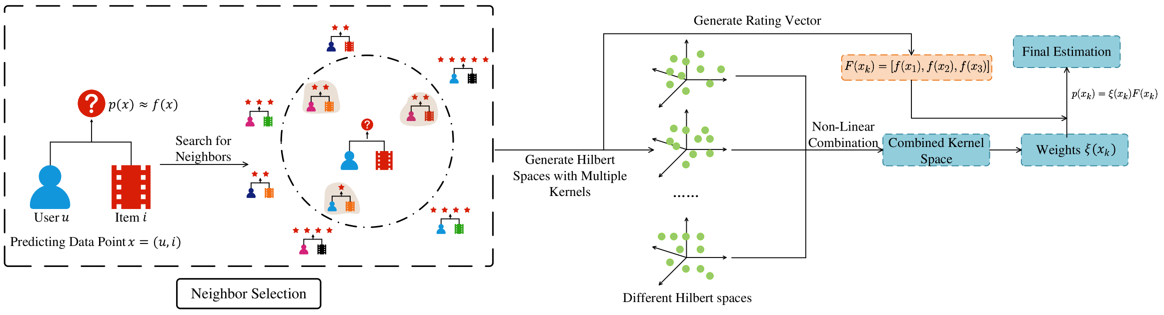

Fig. 3. The framework of proposed M-KMC. All q kernel functions firstly project the original data of neighbors into Hilbert spaces, and then the model combines these spaces to compute the final estimation.

Then the final estimations are obtained by Eq. (18). The value of α_t is learned by optimizing the loss function $\mathcal{L}$ defined as

$$\mathcal{L} = \sum_{k=1}^{|\Omega|} \|f(x_k) - \xi(x_k)F(x_k)\|_2^2 - \gamma(\sum_{t=1}^{q} \alpha_t), \tag{21}$$

where γ is a coefficient that controls the degree of regularization. In our experiments, we adopt stochastic gradient descent (SGD) to minimize the loss function and update α_t.

Algorithm 2 shows the details of the training procedure for M-KMC. Similar to KMC, M-KMC undertakes processes of similarity matrix computation, neighbor selection and kernel generation. However, it calculates multiple kernel matrices with different kernels, and attempts to learn the weights $\{\alpha_t\}_{t=1}^{q}$ during the training iterations. The time and memory complexity for similarity computation is the same as that for KMC. Because we need to generate q different spaces, the time and memory complexity for kernel generation is $O(qL^2)$.

Algorithm 2 Training Algorithm for M-KMC

```
Input:
    $Y \in \mathbb{R}^{m \times n}$: original rating matrix,
    $maxIter$: training iterations,
    $L, q, r$: number of neighbors, number of kernel space number and parameter for multi-kernel framework.
Output:
    $\alpha_1, \cdots, \alpha_q$: weights for multiple kernels.
 1: Initialize weights $\alpha_1, \cdots, \alpha_q$ with $\frac{1}{q}$.
 2: Generate similarity matrices $S_U$ and $S_I$.
 3: for all $x_k \in \bar{\Omega}$ do
 4:     Find out $L$ neighbors of data point $x_k$.
 5:     Generate vector of neighborhood real ratings $F(x_k)$.
 6:     for $t = 1 \to q$ do
 7:         Compute kernel matrices $[K_t(x_k, x_i)]_{i=1}^{L}$ and $[K_t(x_i, x_j)]_{i,j=1}^{L}$.
 8:     end for
 9: end for
10: $iter \leftarrow 0$.
11: while $iter < maxIter$ or loss decreases do
12:     for all $x_k \in \bar{\Omega}$ do
13:         Compute $\xi(x_k)$ using Eq. (20).
14:         Compute $p(x_k)$ using Eq. (18).
15:         Update $\alpha_t$ using SGD method.
16:     end for
17:     $iter \leftarrow iter + 1$.
18: end while
19: return $\alpha_1, \cdots, \alpha_q$.
```

4. Experiments and analysis

In this section, we conduct several experiments for our proposed KMC and M-KMC on real-world datasets from different recommendation environments. The performances of different parameter settings are compared to analyze the parameter sensitivity. Finally, we compare our proposed models with both the traditional and state-of-the-art methods via the same metrics to prove the feasibility of our models.

4.1. Descriptions of datasets

In our experiments, we adopt seven different real-world datasets that are widely used in recommender systems for various fields, including movie, book, music and joke recommendations. The details of these datasets are outlined below.

Filmtrust[1] is a small recommender dataset for movie recommendations that was crawled from the Filmtrust website in 2011. The dataset includes 35,497 rating records over 1,508 users and 2,071 items.

Epinions[2] is a dataset for product recommendations with a 5-star rating system. In our experiments, we extracted all ratings from the first 4,000 users and 12,000 items.

[1] https://www.librec.net/datasets.html.

[2] http://www.trustlet.org/wiki/Epinions_dataset.

Table 2
Statistics of the real-world datasets in our experiments.

	Filmtrust	Epinions	Amusic	Jester	Ciao	BookCrossing	ML-1M
Type of items	Movies	Products	Music	Jokes	Videos	Books	Movies
Number of users	1,508	4,000	5,249	15,000	17,615	20,000	6,040
Number of items	2,071	12,000	4,874	150	16,121	15,000	3,952
Number of ratings	35,497	81,513	53,316	390,772	72,665	44,648	1,000,201
Rating scale	[0.5, 4.0]	[1.0, 5.0]	[1.0, 5.0]	[−10.0, 10.0]	[1.0, 5.0]	[1.0, 10.0]	[1.0, 5.0]
Data density	0.01137	0.00170	0.00208	0.17368	0.00025	0.00002	0.04190

Amazon music (Amusic)[3] is a music recommendation dataset collected from the Amazon website, which is also a database with 5-star rating records. We use first 5,294 users and 4,874 items for our experiments. All users have rated at least 30 items and all items are rated by at least 60 users.

Jester[4] is a benchmark dataset for joke recommender systems which contains substantial ratings provided by users. The rating scale ranges from −10.0 to 10.0.

Ciao[1] dataset was crawled from the ratings of DVDs on the Ciao website in 2013, which is composed of 72,665 ratings from 17,615 users on 16,121 items.

BookCrossing[5] is a book recommendation dataset collected from the BookCrossing community, which contains both explicit and implicit feedback on books. We selected the first 5,000 users and 6,000 books for our prediction.

MovieLens 1M (ML-1M)[6] is a famous benchmark dataset for ratings on movies, which includes approximately 6,000 users and 4,000 movies.

Table 2 shows more details of the real-world datasets that are used in our experiments. Most datasets are sparse and over 99% of the ratings are unknown, while Jester is a dense dataset compared to the others, with over 17% of ratings are available.

4.2. Performance evaluation

In this paper, we adopt the mean absolute error (MAE) and root mean squared error (RMSE) to evaluate the performance of our models, which are both widely applied in matrix completion. Assume that there are t real ratings $y_1, \dots, y_t$ and t prediction ratings $\hat{y}_1, \dots, \hat{y}_t$, the MAE and RMSE values are computed by $MAE = \frac{1}{t}\sum_{i=1}^{t} \|y_i - \hat{y}_i\|$ and $RMSE = \sqrt{\frac{1}{t}\sum_{i=1}^{t}(y_i - \hat{y}_i)}$, respectively. In our experiments, all datasets are divided into an 80% training set and 20% testing set. We evaluate the accuracy of estimations on testing set.

4.3. Experiments and analysis

Analysis of singular matrices

First, as there is a matrix inverse procedure in our framework, we test the situation of singular matrices. If the kernel matrix is not invertible, the framework will not work when computing $[K(x_i, x_j)]_{i,j=1}^{N}{}^{-1}$. We experiment our model on all seven tested datasets with different neighbor number settings, and find that the phenomenon of singularity only appears in Filmtrust and Ciao. Fig. 4 shows the percentage of singular matrices for these two datasets.

In the experiments, we use the Gaussian kernel function with parameter $c = 0.01$. From Fig. 4 we discover that as the number of neighbors increases, the appearance of singular matrices decreases overall. The lowest percentage of singular matrices for Ciao is less than 1%, and the percentage for Filmtrust decreases to zero when the neighbor number reaches 20. Considering that the other five datasets do not generate singular matrix, we believe that the occurrence of singular matrices has limited negative impact on our model, especially when there are enough neighbors. In the following experiments, to avoid the influence of few singular matrices, we predict the rating by computing the average value of the specific item rating vector when the kernel matrix is singular.

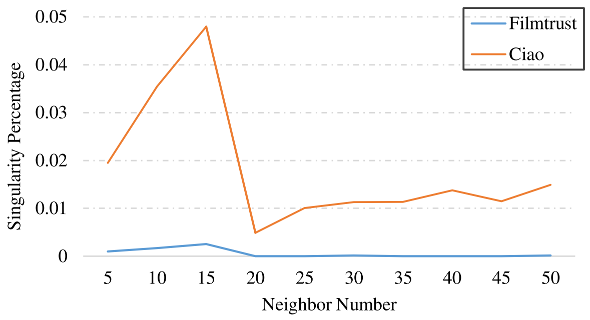

Fig. 4. Percentage of singular matrices for Filmtrust and Ciao.

Neighbor influence

Next, we conduct experiments to test our models with different numbers of neighbors. Fig. 5 shows the influence of the neighbor number on all tested datasets. The experimental results manifest that the MAE and RMSE values first decrease with an increase in neighbor numbers, and then increase or stay stable as neighbor numbers increase. It is noticed that our previous analysis also shows that the lack of neighbors may lead to singular matrix issue, which sometimes results in the failure of the proposed methods. For most datasets, the model with 15 or 20 neighbors achieves the best performance. However, for the Jester and Amusic datasets, the MAE and RMSE values do not reach the lowest point simultaneously, with RMSE usually requiring more neighbors for the best performance. The results of these experiments indicate that moderate neighbor number significantly contributes to the final estimation, whereas too many neighbors may lead to worse performance. This phenomenon might be because too many irrelevant ratings will have a negative effect on the final estimation.

In our experiments, when there are not enough candidate neighbors, we sample other points randomly to ensure that the ratings are estimated by enough reference points. Table 3 provides statistics on the random sampling of neighbors. Due to the sparsity in the BookCrossing dataset, the frequency of random sampling in BookCrossing is higher than that in Epinions. Both datasets achieve their best performance with 20 neighbors. Thus a proper setting of neighbor numbers maintains a balance between the quality of neighbors and the number of sample points. For relatively dense datasets such as Jester and ML-1M, the random sampling rate is almost 0, with more neighbors not necessarily leading to better performance in these datasets. It is shown that excessive neighbors have a negative influence on the final estimation, regardless of whether the randomly selected neighbors are included in the candidate points.

Finally we conduct experiments to explore the computational time for interpolating per 1,000 predicting ratings with different neighbor numbers, as shown in Table 4. The experiments are run on computer

[3] http://jmcauley.ucsd.edu/data/amazon/.
[4] https://goldberg.berkeley.edu/jester-data/.
[5] http://www2.informatik.uni-freiburg.de/∼cziegler/BX/.
[6] https://grouplens.org/datasets/movielens/.

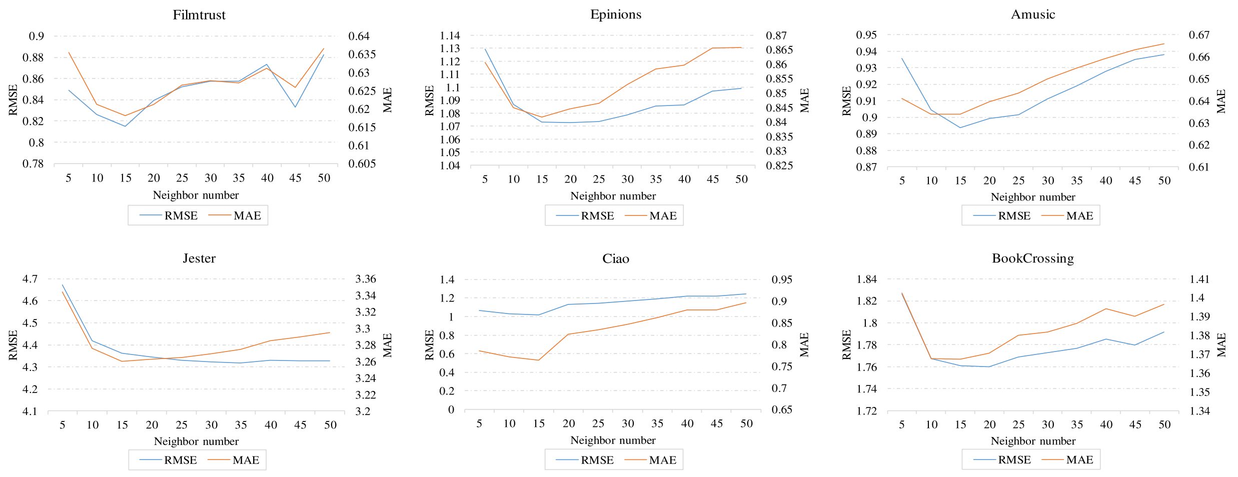

Fig. 5. Performance comparisons of neighbor number on all tested datasets except ML-1M. The primary axis records RMSE and the secondary axis records MAE.

Table 3
Statistics of the random sampling on Epinions and Bookcrossing.

	Neighbors required	Random sampling rate	Average sampling times	RMSE
Epinions	5	0.10%	1.99	1.1292
	10	0.42%	3.90	1.0869
	15	0.98%	5.49	1.0729
	20	1.70%	7.32	1.0725
	25	2.49%	9.34	1.0735
	30	3.37%	11.33	1.0785
BookCrossing	5	3.01%	2.93	1.8262
	10	5.79%	5.62	1.7673
	15	7.64%	8.80	1.7609
	20	9.22%	11.96	1.7600
	25	10.67%	15.08	1.7689
	30	11.72%	18.57	1.7727

Table 4
Average runtime of KMC with different neighbor numbers for interpolating per 1,000 predicting ratings.

Neighbor number	5	10	15	20	25	30	35	40
Runtime (s)	15.12	18.42	21.34	24.34	31.43	39.32	45.66	53.18

with an I5-7200U CPU and 8G RAM. We repeat the experiments 10 times and record the average runtime. The experiment results show that the runtime increases when we select more neighbors for interpolation. Associated with aforementioned analysis, it can be observed from the table that 15 or 20 neighbors are good enough in terms of both time cost and accuracy.

Multi-kernel framework

As to M-KMC, we adopt Gaussian kernels with different c to map the original data into different spaces. Based on experimental experience, we set parameter c to $\{0.01, 0.03, \ldots, 0.39\}$ with a step of 0.02, therefore, there are totally 20 different kernel spaces waiting for merging. The weight of each kernel space α_t is computed by the optimization problem defined in the preceding section. These weights control the contribution of each kernel, so that better results are obtained by the proper non-linear combination of different kernels.

Epinions and BookCrossing are used as examples here of the analysis. Table 5 shows the comparisons for KMC and M-KMC in Epinions and BookCrossing datasets. Different values of parameter r defined in Eq. (19) are compared, as well as the simple average combination for different kernel functions. When $r = 1$, the experimental results are equivalent to the linear model defined in Eq. (8). The experiments demonstrate that M-KMC has greater improvement than KMC. The

Table 5
Performance comparisons for KMC and M-KMC.

	Epinions		BookCrossing	
	MAE	RMSE	MAE	RMSE
KMC	0.8583	1.0757	1.3734	1.7658
M-KMC (average weighted)	0.8580	1.0755	1.3674	1.7609
M-KMC (r = 1)	0.8336	1.0626	1.3624	1.7578
M-KMC (r = 4)	0.8297	1.0617	1.3621	1.7576
M-KMC (r = 16)	**0.8294**	**1.0616**	**1.3618**	**1.7574**

models obtain better performance overall when r increases, however, a r value that is too large does not significantly improve performance. The outcomes also indicate that calculating the average of multiple kernels directly brings limited improvement compared to the single-kernel models, which shows that the training for weights $\{\alpha_t\}_{t=1}^{q}$ is indispensable.

Figs. 6 and 7 show the weights of different kernel spaces in Epinions and BookCrossing. When the kernels are combined linearly ($r = 1$), some kernel spaces account for a large proportion of the final space. For Epinions dataset, the kernel function with $c = 0.01$ has a distinct contribution in final estimations. Simultaneously, kernel functions where parameter c ranges from 0.01 to 0.09 play an important role in generating final kernel matrices in BookCrossing. This phenomenon indicates that these kernels are adaptive in these datasets, which is why we select $c = 0.01$ for our experiments on KMC. However, this is contrary to the original intention of bringing in the MKL framework. When we introduce into parameter $r > 1$, the deviations of weights between each kernel become smaller as r increases, therefore, each kernel works together instead of only one or several kernels working. Table 5 indicates that such a trick helps improve performance as analyzed before.

4.4. Performance comparisons

In this subsection we compare our models with both the traditional and state-of-the-art approaches, with the compared algorithms included as follows: NMF (Lee & Seung, 1999), Item-KNN (Sarwar et al., 2001), LLORMA (Lee et al., 2013), AutoRec (Sedhain et al., 2015), IIR-G (Sun et al., 2015), KMF, MKMF (Liu et al., 2016), RRN (Wu et al., 2017) and DR (Kuchaiev & Ginsburg, 2018). Notice that LLORMA, KMF and MKMF are also methods applying kernel techniques.

The optimal experimental settings for the compared algorithms are determined by our experiments or suggested by the original studies. The parameter settings for these compared algorithms are outlined as follows: (1) NMF: latent factor number $f = 100$; (2) Item-KNN: set

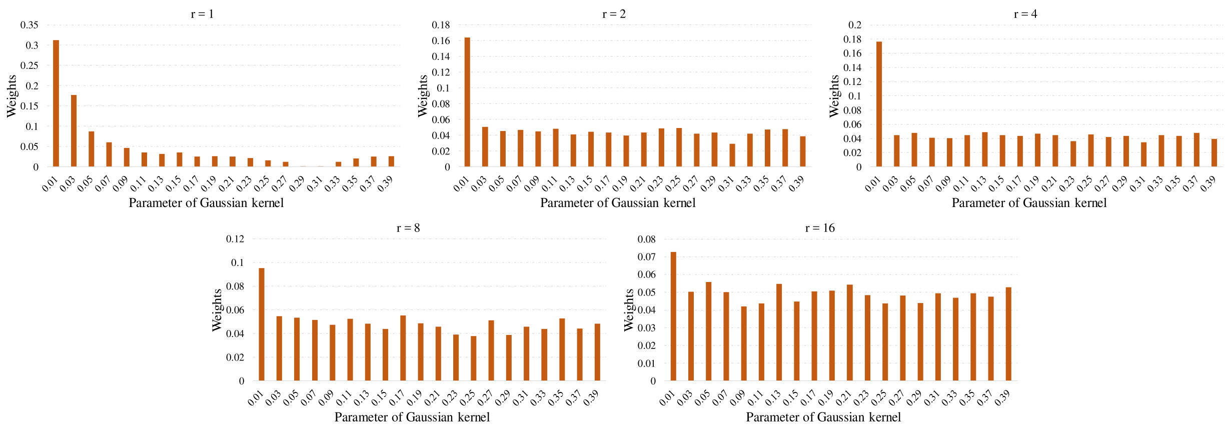

Fig. 6. The weights for each kernel space on Epinions dataset.

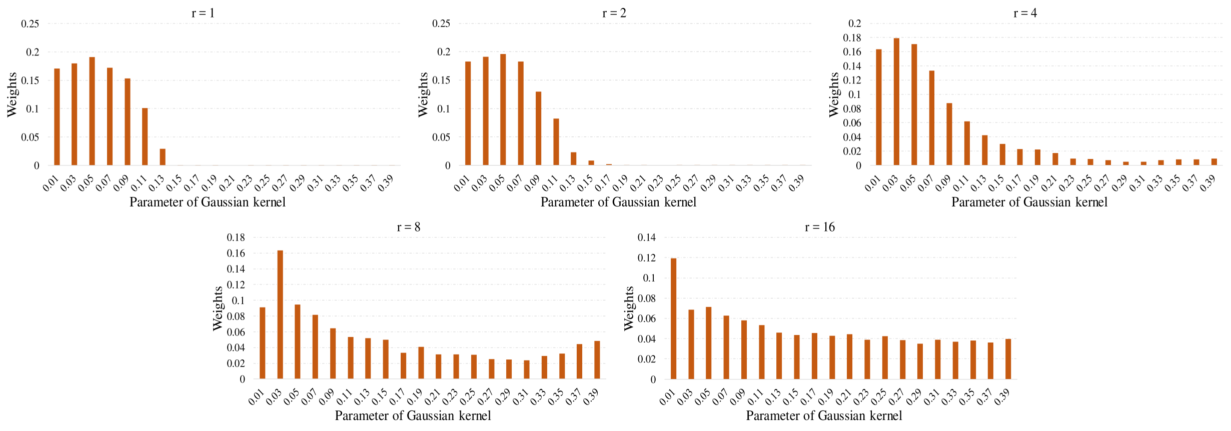

Fig. 7. The weights for each kernel space on BookCrossing dataset.

max number of neighbors as 40; (3) LLORMA: global factor number $f_{global} = 10$; (4) AutoRec: the dimension of hidden layer is set to 200, the activation functions of hidden layer and output layer are sigmoid function and identity function, respectively. (5) IRR-G: $f = 10$, the importance of regularization by implicit item relationships is set as $\alpha = 0.1$; (6) KMF and MKMF: apply Gaussian kernels with σ ranging in $\{0.3, 0.6, 0.9\}$; (7) RRN: set a single-layer LSTM with 40-dimensional input embeddings, 40-dimensional hidden layer and 20-dimensional dynamic states; (8) DR: the dimensions of the 6-layer network are set at $\{64, 64, 128, 64, 64\}$ for Filmtrust dataset, while the other datasets are set at $\{512, 512, 1024, 512, 512\}$. The drop probability is set $p = 0.8$. For all learning algorithms we set the learning rate $lr = 0.01$ except that for the DR that is set $lr = 0.0005$, and all regularization coefficients are set at 0.01.

As to our proposed KMC and M-KMC, we set the number of neighbors $L = 15$ for Filmtrust, Amusic, Ciao, ML-1M and BookCrossing datasets, while $L = 20$ for Epinions dataset and $L = 35$ for Jester dataset. Parameter δ for generating similarity matrix is tuned in $\{5, 10, 15, 20, 25, 30, 35, 40\}$ for best performance. We select parameter $c = 0.01$ for KMC. The values of r for M-KMC range in $\{2, 4, 6, 8\}$. Especially, we tune the learning rate as $lr = 0.0001$ for M-KMC.

Tables 6 and 7 compare our models with other approaches. Five-fold cross-validation is conducted and we record the average performance and standard deviations. Comparing the two proposed models, M-KMC performs better than KMC for all datasets. The comparative experiments prove that our framework works effectively, and achieves a performance that is competitive or superior to the other approaches. Our approach performs better in the RMSE metric for six of the seven tested datasets, and obtains the best performance in the MAE metric for five of the seven datasets. Both KMC and M-KMC perform better on most sparse datasets with over 99% missing ratings. Even for extremely sparse datasets such as BookCrossing where over 99.9% ratings are unknown, our models are competitive or even outperform on both MAE and RMSE.

5. Conclusion

In this paper, we proposed a kernel-based framework KMC for neighborhood-based recommender systems, which aimed to retrieve missing ratings in the user–item interaction matrix. The model projected the original data into a new Hilbert space with RBF kernels and realized the local matrix approximation. Associated with the interpolation condition, the weights of different neighbors were computed by kernel matrices, so that the final estimations were conducted by the dot product of weight vectors and rating vectors of neighbors. This improved the performance of traditional NBMs because of better representation of Hilbert space. Besides, we reformulated the widely used Gaussian kernels with variation of MSD metric, which learned a more effective low-dimensional representation for databases of recommender systems. Furthermore, we presented the M-KMC that utilized a multi-kernel framework to improve the performance of KMC. Different from widely used linear combination of multiple kernels, a non-linear strategy was adopted with the auto-weighted method. Substantial experiments were conducted to analyze different parameters and proved the effectiveness of our framework. The experiments demonstrated that our models worked better than other related works on most datasets.

There remain several directions of extensions or improvements for our models. More methods for kernel combination or feature co-learning might be discovered from recent studies on multiview learning. Additionally, a more effective neighbor selection or similarity

Table 6
Performance comparisons for our models on all tested datasets with MAE (mean ± std). The last line of the table calculates the improvement (%) of our model compared to other approaches that perform the best.

	Filmtrust	Epinions	Amusic	Jester	Ciao	BookCrossing	ML-1M
NMF	0.6530 ± 0.6%	0.8852 ± 0.8%	0.7028 ± 0.7%	3.4370 ± 0.7%	1.1532 ± 0.7%	2.3276 ± 1.6%	0.7270 ± 0.1%
Item-KNN	0.6544 ± 0.2%	0.9900 ± 0.3%	0.7381 ± 0.2%	4.4486 ± 1.4%	0.8077 ± 0.5%	2.8816 ± 0.8%	0.7273 ± 0.2%
LLORMA	0.6396 ± 0.8%	0.8761 ± 0.6%	0.7789 ± 1.8%	**3.2421 ± 1.2%**	1.1937 ± 6.3%	2.2159 ± 1.2%	0.6762 ± 0.1%
AutoRec	0.6381 ± 1.1%	0.8944 ± 2.4%	0.7978 ± 1.6%	3.5221 ± 2.5%	1.2461 ± 2.2%	2.5887 ± 1.9%	0.6821 ± 0.7%
IRR-G	0.6183 ± 0.8%	0.8325 ± 0.8%	0.6608 ± 0.8%	3.9750 ± 1.3%	0.7627 ± 1.4%	1.3704 ± 1.5%	**0.6751 ± 0.7%**
KMF	0.6209 ± 0.9%	0.8535 ± 0.8%	0.7321 ± 1.0%	3.4218 ± 1.2%	0.8122 ± 1.5%	1.4016 ± 1.2%	0.6799 ± 0.8%
MKMF	0.6196 ± 0.8%	0.8520 ± 0.7%	0.7204 ± 0.9%	3.4012 ± 1.3%	0.8017 ± 1.7%	1.3977 ± 1.0%	0.6782 ± 0.5%
RRN	0.7123 ± 0.7%	0.9155 ± 1.9%	0.7484 ± 1.2%	4.2082 ± 1.1%	0.8171 ± 1.6%	1.3949 ± 1.1%	0.8703 ± 0.9%
DR	0.6990 ± 0.6%	0.8271 ± 0.9%	0.7389 ± 0.8%	3.4807 ± 0.7%	0.8423 ± 1.5%	1.4832 ± 0.7%	0.7764 ± 0.7%
KMC	0.6181 ± 0.3%	0.8583 ± 0.5%	0.6477 ± 0.6%	3.2743 ± 0.8%	0.7336 ± 1.0%	1.3734 ± 0.6%	0.7079 ± 0.6%
M-KMC	**0.6137 ± 0.1%**	**0.8264 ± 0.4%**	**0.6351 ± 0.4%**	3.2700 ± 0.9%	**0.7288 ± 0.7%**	**1.3618 ± 0.7%**	0.7077 ± 0.2%
Improve	0.7%	0.1%	3.9%	–	4.4%	0.6%	–

Table 7
Performance comparisons for our models on all tested datasets with RMSE (mean ± std). The last line of the table calculates the improvement (%) of our model compared to other approaches that perform the best.

	Filmtrust	Epinions	Amusic	Jester	Ciao	BookCrossing	ML-1M
NMF	0.8620 ± 0.9%	1.1480 ± 0.8%	1.0721 ± 1.2%	4.5241 ± 0.9%	1.6684 ± 1.0%	3.3416 ± 1.6%	0.9214 ± 0.2%
Item-KNN	0.8639 ± 0.3%	1.3225 ± 0.4%	1.0132 ± 0.2%	5.3306 ± 1.2%	1.0690 ± 0.6%	3.2692 ± 0.5%	0.9234 ± 0.3%
LLORMA	0.8601 ± 1.1%	1.1823 ± 0.9%	1.1644 ± 3.2%	4.3793 ± 1.1%	1.6653 ± 7.9%	3.1287 ± 1.5%	0.8644 ± 0.2%
AutoRec	0.8404 ± 1.1%	1.1533 ± 0.7%	1.1437 ± 1.6%	4.6457 ± 1.7%	1.7103 ± 0.4%	3.5435 ± 1.5%	0.8703 ± 1.1%
IRR-G	0.8155 ± 1.0%	1.0959 ± 0.6%	0.9141 ± 1.4%	4.9157 ± 0.9%	0.9901 ± 0.6%	1.7772 ± 1.0%	**0.8551 ± 0.8%**
KMF	0.8206 ± 1.2%	1.1256 ± 0.7%	0.9653 ± 1.1%	4.5231 ± 0.6%	1.0621 ± 0.7%	1.8011 ± 0.8%	0.8624 ± 0.5%
MKMF	0.8198 ± 0.9%	1.1248 ± 0.7%	0.9622 ± 1.2%	4.5091 ± 0.8%	1.0554 ± 0.5%	1.7928 ± 0.7%	0.8609 ± 0.4%
RRN	0.9049 ± 1.5%	1.1898 ± 0.9%	0.9770 ± 0.7%	5.1191 ± 1.1%	1.0248 ± 1.4%	1.7896 ± 0.9%	1.1560 ± 0.7%
DR	0.8915 ± 1.6%	1.0998 ± 0.8%	0.9978 ± 0.9%	4.5579 ± 0.8%	1.1261 ± 1.6%	1.9117 ± 1.8%	0.9972 ± 0.7%
KMC	0.8148 ± 0.8%	1.0757 ± 0.5%	0.8968 ± 0.9%	4.3170 ± 1.9%	0.9927 ± 0.6%	1.7658 ± 0.5%	0.9025 ± 0.5%
M-KMC	**0.8118 ± 0.9%**	**1.0617 ± 0.6%**	**0.8926 ± 0.6%**	**4.3079 ± 1.5%**	**0.9846 ± 0.9%**	**1.7574 ± 0.7%**	0.9022 ± 0.3%
Improve	0.5%	3.1%	2.4%	1.6%	0.6%	1.1%	–

computation method would lead to a significant improvement for our models. We may further explore more powerful similarity metrics because they are essential to our methods. We will explore more efficient algorithms for recommender systems with kernel learning in the future.

CRediT authorship contribution statement

Zhaoliang Chen: Conceptualization, Formal analysis, Methodology, Writing - original draft. **Wei Zhao:** Conceptualization, Formal analysis, Methodology, Writing - revision. **Shiping Wang:** Funding acquisition, Writing - review & editing.

Declaration of competing interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

Acknowledgments

This work was partially supported by the National Natural Science Foundation of China (Nos. U1705262 and 61672159), the Technology Innovation Platform Project of Fujian Province, China under Grant (Nos. 2014H2005 and 2009J1007), the Fujian Collaborative Innovation Center for Big Data Application in Governments, the Fujian Engineering Research Center of Big Data Analysis and Processing.

References

Chen, J., & Li, X. (2019). Model-free nonconvex matrix completion: Local minima analysis and applications in memory-efficient kernel PCA. *Journal of Machine Learning Research*, *20*, 142:1–142:39.

Chen, C., Li, D., Lv, Q., Yan, J., Shang, L., & Chu, S. M. (2017). GLOMA: Embedding global information in local matrix approximation models for collaborative filtering. In *Proceedings of the 31st AAAI conference on artificial intelligence* (pp. 1295–1301).

Chen, J., Lian, D., & Zheng, K. (2019). Improving one-class collaborative filtering via ranking-based implicit regularizer. In *Proceedings of the 33rd AAAI conference on artificial intelligence* (pp. 37–44).

Cruz, D. P. F., Maia, R. D., da Silva, L. A., & de Castro, L. N. (2016). Beerbf: a bee-inspired data clustering approach to design rbf neural network classifiers. *Neurocomputing*, *172*, 427–437.

Fan, J., & Chow, T. W. S. (2018). Non-linear matrix completion. *Pattern Recognition*, *77*, 378–394.

Fan, J., Zhang, Y., & Udell, M. (2020). Polynomial matrix completion for missing data imputation and transductive learning. In *The 24th AAAI conference on artificial intelligence* (pp. 3842–3849).

Hwang, W., Lee, H., Kim, S., Won, Y., & Lee, M. (2016). Efficient recommendation methods using category experts for a large dataset. *Information Fusion*, *28*, 75–82.

Kang, Z., Peng, C., & Cheng, Q. (2016). Top-n recommender system via matrix completion. In *Proceedings of the 30th AAAI conference on artificial intelligence* (pp. 179–185).

Koren, Y. (2008). Factorization meets the neighborhood: a multifaceted collaborative filtering model. In *Proceedings of the 14th ACM SIGKDD international conference on knowledge discovery and data mining* (pp. 426–434).

Kuchaiev, O., & Ginsburg, B. (2018). Training deep autoencoders for recommender systems. In *Proceedings of the 6th international conference on learning representations*.

Kuo, B.-C., Ho, H.-H., Li, C.-H., Hung, C.-C., & Taur, J.-S. (2013). A kernel-based feature selection method for svm with rbf kernel for hyperspectral image classification. *IEEE Journal of Selected Topics in Applied Earth Observations and Remote Sensing*, *7*(1), 317–326.

Lee, J., Kim, S., Lebanon, G., & Singer, Y. (2013). Local low-rank matrix approximation. In *Proceedings of the 30th international conference on machine learning* (pp. 82–90).

Lee, D. D., & Seung, H. S. (1999). Learning the parts of objects by non-negative matrix factorization. *Nature*, *401*(6755), 788.

Li, H., Pan, S. J., Wan, R., & Kot, A. C. (2019). Heterogeneous transfer learning via deep matrix completion with adversarial kernel embedding. In *Proceedings of the 33rd AAAI conference on artificial intelligence* (pp. 8602–8609).

Liang, D., Altosaar, J., Charlin, L., & Blei, D. M. (2016). Factorization meets the item embedding: regularizing matrix factorization with item co-occurrence. In *Proceedings of the 10th ACM conference on recommender systems* (pp. 59–66).

Liu, X., Aggarwal, C. C., Li, Y., Kong, X., Sun, X., & Sathe, S. (2016). Kernelized matrix factorization for collaborative filtering. In *Proceedings of the 2016 SIAM international conference on data mining* (pp. 378–386).

Martinčić-Ipšić, S., Moibob, E., & Perc, M. (2017). Link prediction on Twitter. *PLOS ONE*, *12*(7), 1–21.

Ning, X., & Karypis, G. (2011). SLIM: Sparse linear methods for top-n recommender systems. In *Proceedings of the 11th IEEE international conference on data mining* (pp. 497–506).

Pal, B., & Jenamani, M. (2018). Kernelized probabilistic matrix factorization for collaborative filtering: exploiting projected user and item graph. In *Proceedings of the 12th ACM conference on recommender systems* (pp. 437–440).

Park, Y., Park, S., Jung, W., & Lee, S.-g. (2015). Reversed cf: A fast collaborative filtering algorithm using a k-nearest neighbor graph. *Expert Systems with Applications*, *42*(8), 4022–4028.

Qian, Y., Zhang, Y., Ma, X., Yu, H., & Peng, L. (2019). EARS: emotion-aware recommender system based on hybrid information fusion. *Information Fusion*, *46*, 141–146.

Romani, L., Rossini, M., & Schenone, D. (2019). Edge detection methods based on RBF interpolation. *Journal of Computational and Applied Mathematics*, *349*, 532–547.

Sarwar, B. M., Karypis, G., Konstan, J. A., & Riedl, J. (2001). Item-based collaborative filtering recommendation algorithms. In *Proceedings of the 10th international conference on world wide web* (pp. 285–295).

Sedhain, S., Menon, A. K., Sanner, S., & Xie, L. (2015). Autorec: Autoencoders meet collaborative filtering. In *Proceedings of the 24th international conference on world wide web* (pp. 111–112).

Sun, Z., Guo, G., & Zhang, J. (2015). Exploiting implicit item relationships for recommender systems. In *Proceedings of international conference on user modeling, adaptation, and personalization* (pp. 252–264).

Wang, M., Hua, X.-S., Yuan, X., Song, Y., & Dai, L.-R. (2007). Optimizing multi-graph learning: towards a unified video annotation scheme. In *Proceedings of the 15th ACM international conference on multimedia* (pp. 862–871).

Wang, C., Liu, Q., Wu, R., Chen, E., Liu, C., Huang, X., & Huang, Z. (2018). Confidence-aware matrix factorization for recommender systems. In *Proceedings of the 32nd AAAI conference on artificial intelligence* (pp. 434–442).

Wang, W., Zhang, G., & Lu, J. (2019). Hierarchy visualization for group recommender systems. *IEEE Transactions on Systems, Man, and Cybernetics: Systems*, *49*(6), 1152–1163.

Wang, C., Zhou, T., Chen, C., Hu, T., & Chen, G. (2019). CAMO: A collaborative ranking method for content based recommendation. In *Proceedings of the 33rd AAAI conference on artificial intelligence* (pp. 5224–5231).

Wu, C.-Y., Ahmed, A., Beutel, A., Smola, A. J., & Jing, H. (2017). Recurrent recommender networks. In *Proceedings of the 10th ACM international conference on web search and data mining* (pp. 495–503).

Xue, H., Dai, X., Zhang, J., Huang, S., & Chen, J. (2017). Deep matrix factorization models for recommender systems. In *Proceedings of the 26th international joint conference on artificial intelligence* (pp. 3203–3209).

Yuan, W., He, K., Guan, D., Zhou, L., & Li, C. (2019). Graph kernel based link prediction for signed social networks. *Information Fusion*, *46*, 1–10.

Zhou, T., Shan, H., Banerjee, A., & Sapiro, G. (2012). Kernelized probabilistic matrix factorization: Exploiting graphs and side information. In *Proceedings of the 12th SIAM international conference on data mining* (pp. 403–414).