Research on Matrix Factorization Algorithm based on Multiple Kernel Learning in Collaborative Filtering

Zhaoyong Liu, Yiting Zhang

Sichuan Vocational College of Chemical Technology Luzhou City, Sichuan Province 646099, China.

Abstract. The matrix factorization (MF) algorithm plays an important role in collaborative filtering (CF). The traditional MF algorithms usually assume that the correlated data is distributed on a linear hyperplane, which is not always the case. Therefore, it is a hot topic to combine the kernel algorithm with the matrix factorization algorithm for collaborative filtering. Aiming at the shortcomings of the existing CF algorithm, an single matrix factorization method based on dictionary is firstly proposed for collaborative filtering. On this basis, considering that data can be efficiently embedded into high dimensional space, only one kernel function can not achieve the limitation of optimality energy, a matrix factorization algorithm based on multi kernel learning (MKLMF) is proposed for collaborative filtering. The MKLMF algorithm assumes that there are a set of p positive defined base kernels, and then the two potential factor matrices are embedded into the high dimensional Hilbert space by the best linear group of the learned p kernels. Finally, the product of the two potential factor matrices is used to realize the nonlinear reconstruction of the scoring matrix in the original space. A full simulation experiment is carried out with real data sets. The simulation results show that the proposed algorithm can accurately grasp the nonlinear correlation between data, and the prediction accuracy is better than the latest CF algorithm.

Keywords: Matrix factorization; Collaborative filtering; Kernel algorithm; potential factor matrices; prediction accuracy.

1. Introduction

With the rapid growth of online available data, how to obtain useful information from massive data becomes an urgent problem. Collaborative filtering technology ([1]), as one of the various solutions, has attracted much attention in recent years. This technology is one of the most mature recommendation algorithms used in today's recommendation systems [2]. It uses the preferences of groups with similar interests and common experiences to recommend information that users are interested in. Individuals respond to information (such as ratings) to a considerable extent through a cooperative mechanism and record it for filtering purposes. And help others to filter information. Compared with other recommendation technologies, collaborative filtering does not need any information about the content of the project or users, mainly by scoring observations to analyze users'preferences. In addition, it does not require any domain knowledge, so it is suitable for large-scale data sets and different system types.

Collaborative filtering algorithm can be divided into two types of [3]: neighbor based algorithm and model-based algorithm. Among them, the neighbor-based algorithm estimates the score of the target user or item as follows: First, the vector cosine similarity, conditional probability similarity and other indicators [4] are used to determine the score of a group of adjacent users or adjacent items; then, these known scores are fused to generate the prediction score. This kind of algorithm is relatively easy to implement, but it is vulnerable to the impact of data sparsity, sometimes it is difficult to find a very similar set of neighbors. In addition, this kind of algorithm needs to search the whole data space to determine similar users or similar items. When the number of users or items is large, the cost of scoring prediction process is very high, and the scalability of the algorithm is poor [5].

Model-based algorithms learn from known training data to acquire robust models that can recognize complex patterns. Therefore, the performance of these algorithms is better than that of neighborhood-based algorithms in the face of sparse problems. At present, the typical algorithms are Matrix factorization (MF) [6], AVG [7], IVC-COS [8], IVC-PERSON [8] and SVD [9]. Matrix factorization method has many advantages in solving matrix completion problem, and it is a hot

research topic at present. Matrix factorization method will also be used to study collaborative filtering problem.

Nowadays, the research focus of matrix factorization in collaborative filtering mainly focuses on how to obtain low-rank primitive eigenvalue matrix by singular value decomposition. However, for many data sets in real life, it is impossible to obtain complete matrices effectively by linear low rank factorization, so the traditional method is to embed data into high-dimensional feature space to solve this problem. However, the prediction accuracy of collaborative filtering based on thecombination of accounting and matrix factorization still needs to be improved [12]. The specific reasons are as follows: 1) At present, the matrix factorization method based on kernel function [13,14] often needs metadata, social graph, text review and other auxiliary information. However, recommendation systems in real life sometimes fail to provide the above information. 2) In the traditional matrix factorization method, the latent factors of users and items are often explicitly defined as two low-rank matrices. But for the factorization problem based on kernel function, the matrix of potential factors only exists in implicitly defined Hilbert feature space. Therefore, it is difficult to develop an accounting method suitable for matrix completion problems.

In order to solve these problems, this paper proposes an MKLMF algorithm which combines matrix factorization with multi-kernel learning to solve the cooperative filtering problem. Unlike previous studies, MKLMF algorithm accurately grasps the non-linear correlation in the scoring data, and does not need any auxiliary information, effectively improving the prediction accuracy of collaborative filtering. The rest of the article is structured as follows. Section 2 gives the basic concepts and problem descriptions; Section 3 gives the MKLMF algorithm proposed in this paper; Section 4 discusses the experimental results; Section 5 summarizes the full text.

2. Problem Description

2.1 Mark Method

For ease of description, the meaning of the relevant symbols used in the paper is first presented, as shown in Table 1.

Known m users and n projects, the observed score is three tuple (u,i,r_{ui}) ,It indicates that user u assigns the score r_{ui} to the project i .user $u \in \{1,\cdots,m\}$,project $i \in \{1,\cdots,n\}$, score $r_{ui} \in \Re$ .Assuming that each user can only score once for each item, all scores can be expressed as an $m \times n$ matrix $\mathbf{R}$,The first ui element is r_{ui} . If user p fails to score for project q , the corresponding element r_{pq} in $\mathbf{R}$ is unknown or unobserved. The observed elements are represented as set $\Omega = \{(u,i): r_{ui} - \text{Be observed}\}$, $|\Omega|$ indicates that the number of observations is often far less than $m \times n$.

Let Ω_u indicate the index of the observation score $\mathbf{r}_u$ of line u , Ω^i represents the index of the i column observation score $\mathbf{r}_{:,i}$. If the score of r_u is not observed except first and third values, then there are: $\Omega_u = (1,3)^{\mathrm{T}}$ and $r_{\Omega_u} = (r_{u1}, r_{u3})^{\mathrm{T}}$.

For any given matrix $\mathbf{A}$, $\mathbf{A}_{:,\Omega_u}$ is represented as a submatrix consisting of a matrix $\mathbf{A}$ first column and a third column. Similarly, if the score of r_u is not observed except second and fourth values, Then we have: $\Omega^i = (2,4)^{\mathrm{T}}$ and $\mathbf{r}_{:,\Omega^i} = (r_{2i}, r_{4i})^{\mathrm{T}}$ .Let $\mathbf{A}_{:,\Omega^i}$ represent the submatrix of the matrix $\mathbf{A}$ second column and the fourth column. The goal of collaborative filtering is to estimate the unobserved score $\{r_{ui} \mid (u,i) \notin \Omega\}$ based on the observed score.

Table 1. meaning of related symbols

Symbol	Meaning
$\mathbf{R}$	Scoring matrix with only partial observations
$\hat{\mathbf{R}}$	A dense scoring matrix deduced from
$\mathbf{U}$	User latent factor matrix
$\mathbf{V}$	Project latent factor matrix
$\Omega = \{(u,i)\}$	Index set for observation score, $(u,i) \in \Omega$, r_{ui} Representing the observed values in $\mathbf{R}$
Ω_u	Index of observation score for line u in $\mathbf{R}$
Ω^i	Index of u column observation score in $\mathbf{R}$
$\mathbf{A}_{:,\Omega_u}$ or $\mathbf{A}_{:,\Omega^i}$	The submatrix of matrix $\mathbf{A}$ composed of columns indexed by Ω_u or Ω^i
$\{\mathbf{d}_1,\cdots,\mathbf{d}_k\}$	Collection of dictionary vectors
$\phi(\cdot)$	Implicit mapping function of partial kernel
$\Phi=(\phi(\mathbf{d}_1),\cdots,\phi(\mathbf{d}_k))$	A matrix composed of dictionary in Hilbert feature space
$\mathbf{a}_u$ and $\mathbf{b}_i$	User's Dictionary weight vector; dictionary weight vector of item
$\mathbf{A}$ and $\mathbf{B}$	The user's Dictionary weight matrix; the dictionary weight matrix of the item.
$\{\kappa_1,\cdots,\kappa_p\}$	Set of bases kernel functions
$\{\mathbf{K}_1,\cdots,\mathbf{K}_p\}$	Set of base kernel matrices

2.2 Factorization of Matrix

Matrix factorization, as a main method to solve matrix completion problem, can be applied to collaborative filtering of online recommendation system. The problem of collaborative filtering based on matrix factorization can be expressed as follows: the scoring matrix of some elements is known, each row of the matrix represents the user of the recommendation system, and each list represents the items of the system. Our goal is to predict the unknown element (i, j) by calculating the intrinsic eigenvector of row I (user) and the inner product of column J, As shown in Figure 1.

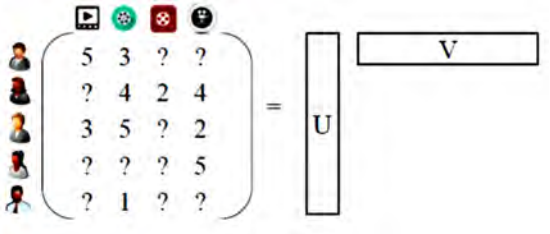


Fig 1. predict unknown score through matrix factorization.

In order to ensure the accuracy of collaborative filtering, the main idea of this method is to approximate the observation matrix $\mathbf{R}$ to the product of two low rank matrices: $\mathbf{R} \approx \mathbf{U}^{\mathrm{T}}\mathbf{V}$ .among, $\mathbf{U}$ represents $k \times m$ matrix, $\mathbf{V}$ represents $k \times n$ matrix. Parameter k is used to control the rank of factorization and to represent the number of original features of each user and item.In most cases, there are $k << \min(m,n)$ .Once the low rank decomposition matrix $\mathbf{U}$ and $\mathbf{V}$ are obtained, the estimated score of user u for item i can be calculated as follows:

$$\hat{r}_{ui} = \sum_{j=1}^{k} u_{ju} v_{ji} = \mathbf{u}_u^{\mathrm{T}} \mathbf{v}_i \tag{1}$$

Among them, u_{ju} represents the j row u column element of matrix $\mathbf{U}$; v_{ji} represents the j row u column element of matrix $\mathbf{V}$; $\mathbf{u}_u$ represents the u column of matrix $\mathbf{U}$; and $\mathbf{v}_i$ represents the i column of matrix $\mathbf{V}$. By solving the following problems, we can determine the low rank parameter matrices $\mathbf{U}$ and $\mathbf{V}$:

$$\min_{\mathbf{U},\mathbf{V}} \text{imize} \left\| P_{\Omega}\left(\mathbf{R} - \mathbf{U}^{\mathrm{T}}\mathbf{V}\right)\right\|_F^2 + \lambda\left(\|\mathbf{U}\|_F^2 + \|\mathbf{V}\|_F^2\right) \tag{2}$$

Projection $P_{\Omega}(\mathbf{X})$ denotes the matrix formed when the observed elements in $\mathbf{X}$ are preserved and the unobserved elements are all set to zero. $\|\cdot\|_F^2$ Representing Frobenius 2 - norm; λ a normalization term that can avoid overfitting. These problems are non-convex and can be solved by gradient descent algorithm and alternative Least Square (ALS) [15]:Assuming that $\mathbf{U}$ is fixed, $\mathbf{V}$ is obtained by solving equation (2).At this point, the problem can be decomposed into N separate ridge regression problems [16]. For the j column of $\mathbf{V}$, the ridge regression problem can be expressed as follows:

$$\min_{\mathbf{v}_j} \text{imize} \left\| \mathbf{r}_{:,\Omega^j} - \mathbf{U}_{:,\Omega^j}^{\mathrm{T}} \mathbf{v}_j \right\|_F^2 + \lambda \left\| \mathbf{v}_j \right\|_2 \tag{3}$$

$\mathbf{r}_{:\Omega^j}$ represents the j column vector obtained by deleting the unobserved elements in matrix $\mathbf{R}$, and $\mathbf{U}_{:,\Omega^j}$ obtained by deleting the corresponding columns in $\mathbf{U}$, as defined in Section 2.1.The closed form solution of the ridge regression problem can be expressed as:

$$\hat{\mathbf{v}}_j = \left(\mathbf{U}_{:,\Omega^j}\mathbf{U}_{:,\Omega^j}^{\mathrm{T}} + \lambda \mathbf{I}\right)^{-1} \mathbf{U}_{:,\Omega^j}\mathbf{r}_{:,\Omega^j} \tag{4}$$

For n single ridge regression problem, a solution $\hat{\mathbf{v}} \in \Re^k$ can be obtained for each problem, and a $k \times n$ matrix $\hat{\mathbf{V}}$ can be obtained by combining n $\hat{\mathbf{v}}$ .Similarly, by fixing $\mathbf{V}$, we can get the solution of $\hat{\mathbf{U}}$ by solving m separate ridge regression problems. Repeat this step until convergence, and finally get the solution of $\hat{\mathbf{U}}$ and $\hat{\mathbf{V}}$.

3. Multi Core Collaborative Filtering

However, the matrix factorization method assumes that the data of matrix $\mathbf{R}$ are distributed on a linear hyperplane, but this assumption is not always true in reality. If the data of matrix $\mathbf{R}$ is distributed on a nonlinear hyperplane, the kernel-based matrix factorization method [17,18] works, as shown in Figure 2. Firstly, two potential factor matrices $\mathbf{U}$ and $\mathbf{V}$ are embedded into a high-dimensional Hilbert space by a linear combination of kernel functions. Then, the nonlinear reconstruction of the score matrix in the original space is realized through the product of two latent factor matrices.

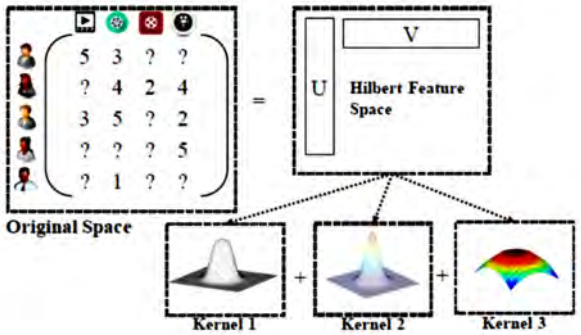


Fig 2. unknown score prediction based on matrix factorization of kernel function

The main function of kernel function is to embed data into high dimensional (sometimes infinite dimensional) feature space H, and the embedding process is indirectly defined by the kernel. Suppose the kernel feature space mapping is expressed as $\phi: \chi \to H$, where χ represents the original space and H represents the Hilbert feature space. The embedding of known data vectors $\mathrm{x} \in \chi$ and x in H can be expressed as $\phi(\mathrm{x})$. Although $\phi(\mathrm{x})$ is not clear enough at this time, the inner product of data points in the feature space is clear, and $\phi(\mathrm{x})^{\mathrm{T}} \phi(\mathrm{x}') = \kappa(x, x') \in \Re$, where κ represents the kernel function of the corresponding kernel. The commonly used kernel function is the Gauss kernel function (or RBF kernel function):

$$\kappa(\mathrm{x}, \mathrm{x}') = \exp\left(-\frac{\left\|\mathrm{x} - \mathrm{x}'\right\|^2}{2\sigma^2}\right) \tag{5}$$

Among them, σ^2 is called bandwidth parameter. Unlike kernel functions, data embedding is different and data association is different. In this paper, two matrix factorization methods based on kernel function are proposed for collaborative filtering problems. Details are given in the following section.

3.1 Dictionary based Single Matrix Factorization (KMF)

Suppose we have k dictionary vector $\{\mathbf{d}_1, \cdots, \mathbf{d}_k\}$, among $\mathbf{d} \in \Re^d$. At the same time, we assume that the eigenvector $\phi(\mathbf{u}_u)$ of $\mathbf{u}_u$ can be expressed as a linear representation of the dictionary vector in the kernel space.

$$\phi(\mathbf{u}_u) = \sum_{j=1}^{k} a_{uj} \phi(\mathbf{d}_j) = \mathbf{\Phi} \mathbf{a}_u \tag{6}$$

$a_{uj} \in \Re$ denotes the weight of each dictionary vector; $\phi(\mathbf{d}_i)$ denotes the eigenvectors in the Hilbert feature space, $\mathbf{a}_u = (a_{u1}, \cdots, a_{uk})^{\mathrm{T}}$ and $\mathbf{\Phi} = (\phi(\mathbf{d}_1), \cdots, \phi(\mathbf{d}_k))$. Similarly, suppose that the eigenvector of $\mathbf{v}_i$ can be expressed as:

$$\phi(\mathbf{v}_i) = \sum_{j=1}^{k} b_{ij} \phi(\mathbf{d}_j) = \mathbf{\Phi} \mathbf{b}_i \tag{7}$$

Among them, $b_i j$ represents the weight of each dictionary vector and $\mathbf{b}_i = (b_{i1}, \cdots, b_{ik})^{\mathrm{T}}$. Therefore, for each user $u \in \{1, \cdots, m\}$, we have weight vector $\mathbf{a}_u$. For each item $i \in \{1, \cdots, n\}$, we have weight vectors $\mathbf{b}_i$.

Similar to Eq. (3), by fixing all $\phi(\mathbf{u}_u)$ (equivalent to fixed weight matrix $\mathbf{A} = (\mathbf{a}_1, \cdots, \mathbf{a}_m)$), we want to solve all $\phi(\mathbf{v}_i)$, that is, the weight matrix $\mathbf{B} = (\mathbf{b}_1, \cdots, \mathbf{b}_n)$.

$$\min\limits_{\phi(\mathbf{v}_i)\in H} \text{imize} \sum_u \left(r_{ui} - \phi(\mathbf{u}_u)^{\mathrm{T}} \phi(\mathbf{v}_i)\right)^2 + \lambda \phi(\mathbf{v}_i)^{\mathrm{T}} \phi(\mathbf{v}_i) \tag{8}$$

Obviously, we have:

$$\phi(\mathbf{u}_u)^{\mathrm{T}} \phi(\mathbf{v}_i) = \mathbf{a}_u^{\mathrm{T}} \mathbf{\Phi}^{\mathrm{T}} \mathbf{\Phi} \mathbf{b}_i = \mathbf{a}_u^{\mathrm{T}} \mathbf{K} \mathbf{b}_i \tag{9}$$

$$\phi(\mathbf{v}_i)^{\mathrm{T}} \phi(\mathbf{v}_i) = \mathbf{b}_i^{\mathrm{T}} \mathbf{\Phi}^{\mathrm{T}} \mathbf{\Phi} \mathbf{b}_i = \mathbf{b}_i^{\mathrm{T}} \mathbf{K} \mathbf{b}_i \tag{10}$$

Among them, $\mathbf{K} = \mathbf{\Phi}^{\mathrm{T}} \mathbf{\Phi}$ represents the Gram matrix (or kernel matrix) of the dictionary vector set $\{d_1, ..., d_k\}$. Thus, formula (8) can be rewritten as:

$$\min\limits_{\mathbf{b}_i \in \Re^k} \text{imize} \sum_u \left(r_{ui} - \mathbf{a}_u^{\mathrm{T}} \mathbf{K} \mathbf{b}_i\right)^2 + \lambda \mathbf{b}_i^{\mathrm{T}} \mathbf{K} \mathbf{b}_i \tag{11}$$

The upper form is equivalent to:

$$\min\limits_{\mathbf{b}_i \in \Re^k} \text{imize} \left\| \mathbf{r}_{:,\Omega^i} - \mathbf{A}_{:,\Omega^i}^{\mathrm{T}} \mathbf{K} \mathbf{b}_i \right\|_F^2 + \lambda \mathbf{b}_i^{\mathrm{T}} \mathbf{K} \mathbf{b}_i \tag{12}$$

The upper form is similar to the ridge regression of kernel function, and its closed form solution is:

$$\hat{\mathbf{b}}_i = \left(\mathbf{K}^{\mathrm{T}} \mathbf{A}_{:,\Omega^i} \mathbf{A}_{:,\Omega^i}^{\mathrm{T}} \mathbf{K} + \lambda \mathbf{K}\right)^{\dagger} \mathbf{K}^{\mathrm{T}} \mathbf{A}_{:,\Omega^i} \mathbf{r}_{:,\Omega^i} \tag{13}$$

By putting n $\hat{\mathbf{b}}$ together separately, we can get an estimate matrix $k \times n$ of $\hat{\mathbf{B}} = \left(\hat{\mathbf{b}}_1, \cdots, \hat{\mathbf{b}}_n\right)$. Similarly, if $\mathbf{B}$ is fixed and $\hat{\mathbf{A}}$ solver is solved by image type (12), the solution of m can be obtained. At this point, the closed form solution of each $\hat{\mathbf{a}}_u$ is as follows:

$$\hat{\mathbf{a}}_u = \left(\mathbf{K}^{\mathrm{T}} \mathbf{B}_{:,\Omega_u} \mathbf{B}_{:,\Omega_u}^{\mathrm{T}} \mathbf{K} + \lambda \mathbf{K}\right)^{\dagger} \mathbf{K}^{\mathrm{T}} \mathbf{B}_{:,\Omega_u} \mathbf{r}_{\Omega_u} \tag{14}$$

3.2 Matrix Factorization Based on Multicore Learning

A kernel function can represent a similarity, but when data is efficiently embedded into high-dimensional space, only one kernel function can not achieve optimal performance [19]. For this reason, many algorithms have been proposed to study the linear combination of multiple kernels, that is, multiple kernel learning (MKL) [20] is applied to matrix factorization problem based on kernel function. The basic idea of MKL is to merge multiple cores instead of using only one kernel.Suppose there is a set of p $\left\{\mathbf{K}_1, \cdots, \mathbf{K}_p\right\}$ with clear definition,Our goal is to learn the kernel-based prediction model by finding the best linear combination of p kernels (i.e. weighted combination $\boldsymbol{\mu} = \left(\mu_1, \cdots, \mu_p\right)^{\mathrm{T}}$). Learning problems can be expressed as follows:

$$\min\limits_{\mathbf{a}_u, \mathbf{b}_i \in \Re^k} \text{imize} \sum_{(u,i)\in\Omega} (r_{ui} - \underbrace{\sum_{j=1}^{p} \mu_j \mathbf{a}_u^{\mathrm{T}} \mathbf{K}_j \mathbf{b}_i}_{\mathbf{v}_{ui}^{\mathrm{T}} \boldsymbol{\mu}})^2 + \lambda (\underbrace{\mathbf{b}_i^{\mathrm{T}} \sum_{j=1}^{p} \mu_j \mathbf{K}_j \mathbf{b}_i + \mathbf{a}_u^{\mathrm{T}} \sum_{j=1}^{p} \mu_j \mathbf{K}_j \mathbf{a}_u}_{\gamma_{ui}^{\mathrm{T}} \boldsymbol{\mu}}) \tag{15}$$

Among them, $\boldsymbol{\mu} \in \Re_+^p$ and $\boldsymbol{\mu}^{\mathrm{T}}\mathbf{l}_p = 1$. In order to facilitate computation, vector is introduced in this paper. $\gamma_{ui} = \left(\mathbf{a}_u^{\mathrm{T}}\mathbf{K}_1\mathbf{a}_u + \mathbf{b}_i^{\mathrm{T}}\mathbf{K}_1\mathbf{b}_i, \cdots, \mathbf{a}_u^{\mathrm{T}}\mathbf{K}_p\mathbf{a}_u + \mathbf{b}_i^{\mathrm{T}}\mathbf{K}_p\mathbf{b}_i\right)^{\mathrm{T}}$, $\upsilon_{ui} = \left(\mathbf{a}_u^{\mathrm{T}}\mathbf{K}_1\mathbf{b}_i, \cdots, \mathbf{a}_u^{\mathrm{T}}\mathbf{K}_p\mathbf{b}_i\right)^{\mathrm{T}}$.Then, the optimization problem in formula (15) can be re represented as:

$$\begin{aligned} &\text{minimize}: \mu^{\mathrm{T}} Y \mu + Z\mu \\ &\text{constraint}: \quad \mu \geq 0; 1_{\mathrm{P}}^{\mathrm{T}}\mu = 1 \end{aligned} \tag{16}$$

Among them, $\mathbf{Y} = \sum_{(u,i)\in\Omega} \boldsymbol{\upsilon}_{ui}\boldsymbol{\upsilon}_{ui}^{\mathrm{T}}$, $\mathbf{Z} = \sum_{(u,k)\in\Omega}(\lambda - 2r_{ui})\gamma_{ui}$.When all $\boldsymbol{\upsilon}_{ui}$ and γ_{ui} are determined ($\mathbf{A}$ and $\mathbf{B}$ are determined), the optimization problem described in equation (16) can be regarded as a quadratic programming problem [21], which can be solved efficiently by Cvxopt or Cvx package [22].

Algorithm 1: matrix factorization (MKLMF) based on multicore
Input:, , , k , d , $\{\kappa_1, \cdots, \kappa_p\}$, $\mathbf{R}$, Ω , λ , $\mathrm{iter}_{\max}$
1:Set up $\mathbf{A} \in \Re^{k\times m}$, $\mathbf{B} \in \Re^{k\times n}$, $\boldsymbol{\mu} \in \Re^p$, $D = \{\mathbf{d}_i \in \Re^d : 1 \leq i \leq k\}$, $\{\mathbf{K}_i \in \Re^{k\times k} : 1 \leq i \leq p\}$, $\mathbf{K} \in \Re^{k\times k}$;
2:Initialization $\boldsymbol{\mu} = \left(\frac{1}{p}, \cdots, \frac{1}{p}\right)^{\mathrm{T}}$, $\mathbf{A}$, $\mathbf{B}$;
3:for $i \leftarrow 1, p$ do
4: $\mathbf{K}_i \leftarrow \left(\kappa_i\left(\mathbf{d}_h, \mathbf{d}_j\right)\right)_{1\leq h, j\leq k}$
5:end for
6: $iter \leftarrow 0$
7:repeat
8: $\mathbf{K} \leftarrow \sum_{i=1}^p \mu_i \mathbf{K}_i$
9: Update $\mathbf{B}$ by using (13);
10: Update $\mathbf{A}$ by using (14);
11: Update $\boldsymbol{\mu}$ by using (16);
12:until $iter = iter_{\max}$ Or converge;
13:Return $\mathbf{A}$, $\mathbf{B}$, $\boldsymbol{\mu}$, $\mathbf{K}$.

Combining Sections 3.1 and 3.2, this paper proposes a multi-core matrix factorization method for cooperative filtering problems, as shown in algorithm 1.At the beginning, we have known rank k, dictionary dimension d, and p basis kernel function $\{\kappa_1, \cdots, \kappa_p\}$,The algorithm initializes k dictionary vector $D = (\mathbf{d}_1, \cdots, \mathbf{d}_k)$ initializes, and then calculates the p basis kernel matrix $\mathbf{K}_i = \left(\kappa_i\left(\mathbf{d}_h, \mathbf{d}_j\right)\right)_{1\leq h, j\leq k}$. The algorithm initializes the kernel weight vector to $\boldsymbol{\mu}^0 = \left(\frac{1}{p}, \cdots, \frac{1}{p}\right)^{\mathrm{T}}$ and generates low rank matrix $\mathbf{A}^0$ and $\mathbf{B}^0$ randomly. So at the beginning, the synthetic kernel matrix $\mathbf{K}^0 = \sum_{1\leq i\leq p} \mu_i^0 \mathbf{K}_i$. After initialization, we first fix $\mathbf{A}^0$ and $\boldsymbol{\mu}^0$, determine $\mathbf{B}^1$ by solving m separate optimization problems in Eq. (12), and obtain each solution directly by calculating the closed-form expression in Eq. (13).Similarly, $\mathbf{B}^1$ and $\boldsymbol{\mu}^0$ can be fixed to get $\mathbf{A}^1$. Finally, fixed $\mathbf{A}^1$ and $\mathbf{B}^1$, using the previously solved convex optimization series algorithm (16), and then get $\boldsymbol{\mu}^1$ .Repeat the above steps similar to ALS until the algorithm converges or reaches the maximum number of iterations

specified before.The optimal solution obtained by the above iteration steps is expressed as $\overset{*}{\mathbf{A}}, \overset{*}{\mathbf{B}}$ and $\overset{*}{\mu}$, and the corresponding synthetic kernel matrix is expressed as $\overset{*}{\mathbf{K}} = \sum_{i=1}^{p} \overset{*}{\mu_i} \mathbf{K}_i$ .Therefore, for each test 3 tuple (u, i, r_{ui}), the predictive value of the algorithm in this paper is $\hat{r}_{ui} = \mathbf{a}_u^{\mathrm{T}} \overset{*}{\mathbf{K}} \mathbf{b}_i$, which is also the element of the U th row I column of the recovery matrix $\hat{\mathbf{R}} = \overset{*\mathrm{T}}{\mathbf{A}} \overset{*}{\mathbf{K}} \overset{*}{\mathbf{B}}$. The scoring estimation method is shown in algorithm 2.

Algorithm 2: scoring estimation
input: $\hat{\Omega}$, $\overset{*}{\mathbf{K}}$, $\overset{*}{\mathbf{A}}$, $\overset{*}{\mathbf{B}}$ 1:distribution P ; 2:for $u, i \in \hat{\Omega}$ do 3: $\hat{r}_{ui} \leftarrow \overset{*\mathrm{T}}{\mathbf{a}_u} \overset{*}{\mathbf{K}} \overset{*}{\mathbf{b}_i}$; 4: take $\hat{r}_{ui}$ Join P ; 5:end for 6:Return P .

4. Simulation Experiment

In order to evaluate the collaborative filtering performance of the proposed algorithm, a comprehensive simulation experiment was carried out using several real data sets. This section first introduces data sets and experiment settings, and then gives experimental results and results analysis.

4.1 Data Set

This algorithm uses the following six real data sets of the recommendation system for performance evaluation: MovieLens, Jester, Flixster, Dating Agency, Yahoo Music and ASSISTTments. The details are shown in Table 2. Among them, each data set has different scoring ranges. For example, the Jester data set scored between - 10 and 10 consecutive points, while Yahoo Music scored between 1 and 100. For each data set, we select the top 1000 items with the highest rating frequency and randomly select 1000 users to generate 1000×1000 (Jester data set is 1000×100 matrix). This test is divided into two experimental scenarios: the first scenario, each user randomly selected a score as a test, the rest of the score for training. Each dataset is randomly selected 5 times. In the second scenario, each user randomly selected 3 scores as the test, and the rest scores were used for training. Each dataset is also randomly selected 5 times independently. The first scene is represented as Leave 1, the second scene is represented as Leave 2, and the training matrix of Scene 2 is more sparse than that of scene 1.

Table 2. basic data sets

data set	Number of users	Number of projects	density(%)
MovieLens	6,040	3,900	6.3
Jester	73,421	100	55.8
Flixster	147,612	48,784	0.11
Dating Agency	135,359	168,791	0.76
Yahoo Music	1,948,882	98,211	0.006
ASSISTments	46,627	179,084	0.073

4.2 Evaluation Index

In order to verify the effectiveness of the proposed KMF algorithm and MKLMF algorithm, the performance of the proposed algorithm is compared with several typical collaborative filtering algorithms, such as AVG [7], IVC-COS [8], IVC-PERSON [8], SVD [9], and MF [6]. Among them, the KMF algorithm can be regarded as a single core version of MKLMF, using only 1 kernel functions at a time. In order to ensure the fairness of comparison, the maximum iteration times of all algorithms are set to 20 times. The performance of the collaborative filtering algorithm is evaluated by the accuracy of the grading prediction. The commonly used score prediction accuracy index is the root mean square error (RMSE):

$$\mathrm{RMSE} = \sqrt{\frac{\sum_{(u,i)\in\Omega}\left(r_{ui} - \hat{r}_{ui}\right)^2}{|\Omega|}} \tag{17}$$

4.3 Result Analysis

In the simulation test, we first use a set of three simple kernels: 1) linear kernel function (LINEAR) $\kappa_1(\mathrm{x},\mathrm{x}') = \mathrm{x}^{\mathrm{T}}\mathrm{x}$; 2) degree polynomial kernel function (POLY) $\kappa_2(\mathrm{x},\mathrm{x}') = \left(1+\mathrm{x}^{\mathrm{T}}\mathrm{x}\right)^2$; 3) Gaussian kernel (RBF kernel) function $\kappa_3(\mathrm{x},\mathrm{x}') = \exp\left(-0.5(\mathrm{x}-\mathrm{x}')^{\mathrm{T}}(\mathrm{x}-\mathrm{x}')/\sigma\right)$ and $\sigma = 0.5$ .Using the optimization problem of the MKLMF algorithm (15) in this paper, we study the linear combination of 3 base kernels. At the same time, we use 3 base cores to test the KMF algorithm proposed in this paper. Tables 3 and 4 show the experimental results of various algorithms on six real data sets. The following conclusions can be drawn: 1) In general, the RMSE values of the proposed MKLMF algorithm on each data set are better than those of other algorithms. For ASSISTments datasets, AVG performs best in RMSE in both scenarios because ASSISTments datasets are scored binary. However, the performance of MKLMF algorithm in Leave 1 scenario is still better than that of all other algorithms. The performance of KMF algorithm based on RBF kernel in Leave 2 scenario is better than that of all other algorithms. This shows that the kernel used in matrix factorization is helpful for the model to grasp the non-linear relationship between the data and to improve the overall accuracy. 2) In some scenarios, the performance of KMF is not as good as MF, but the performance of MKLMF algorithm which combines the three kernels used by KMF is better than MF and other algorithms. This shows that MKLMF has a better data embedding effect when learning the weights of multiple kernels using scoring data. 3) the performance of MKLMF under Leave 2 is slightly worse than that of Leave 1, while KMF is not affected. This shows that compared with solving kernel ridge regression problem, multi-core learning algorithm is more sensitive to sparse problems.

Table 3. the running results of each algorithm in the Leave1 scenario (RMSE (rank))

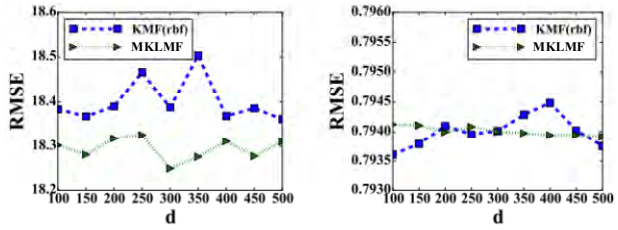
algorithm	data set												Average
	Flixster		Yahoo Music		MovieLens		Jester		Dating		ASSISTments		Rank
MKLMF	0.8270	(1)	18.3036	(1)	0.8223	(1)	4.0398	(1)	1.6697	(1)	0.7920	(2)	1.1667
KMF(LINEAR)	0.8290	(4)	18.3734	(3)	0.8241	(3.5)	4.0471	(2.5)	1.6804	(4)	0.7933	(4)	3.5
KMF(POLY)	0.8289	(3)	18.3766	(4)	0.8241	(3.5)	4.0472	(4)	1.6797	(3)	0.7934	(5)	3.75
KMF(RBF)	0.8292	(5)	18.3883	(5)	0.8246	(5)	4.0471	(2.5)	1.6701	(2)	0.7927	(3)	3.75
MF	0.8286	(2)	18.3214	(2)	0.8235	(2)	4.0549	(5)	1.6849	(5)	0.8001	(6)	3.6667
SVD	0.9441	(9)	26.2255	(9)	0.9406	(7)	4.2794	(6)	1.7920	(8)	0.8919	(9)	8
IVC-COS	0.9223	(7)	22.9477	(7)	1.0016	(8)	4.6137	(8)	2.7052	(7)	0.8106	(7)	7.333
IVC-PEARSON	0.9226	(8)	22.9486	(8)	1.0020	(9)	4.6142	(9)	2.8209	(9)	0.8446	(8)	8.5
AVG	0.9006	(6)	22.4159	(6)	0.8887	(6)	4.3867	(7)	1.7349	(6)	0.7658	(1)	5.333

Table 4. the running results of each algorithm in the Leave2 scenario (RMSE (rank))

algorithm	data set												Average
	Flixster		Yahoo Music		MovieLens		Jester		Dating		ASSISTments		Rank
MKLMF	0.8153	(1)	18.5034	(1)	0.8168	(1)	4.0809	(1)	1.6675	(5)	0.7917	(4.5)	2.25
KMF(LINEAR)	0.8163	(3.5)	18.5426	(3)	0.8178	(2.5)	4.0826	(3)	1.6561	(3)	0.7917	(3)	3
KMF(POLY)	0.8163	(3.5)	18.5500	(4)	0.8179	(4)	4.0824	(2)	1.6562	(4)	0.7919	(4.5)	3.6667
KMF(RBF)	0.8157	(5)	18.5521	(5)	0.8186	(5)	4.0829	(4)	1.6488	(1)	0.7903	(2)	3.6667
MF	0.8155	(2)	18.5375	(2)	0.8178	(2.5)	4.0865	(5)	1.6663	(2)	0.7955	(6)	3.25
SVD	0.9335	(9)	24.3480	(9)	0.9346	(7)	4.2601	(6)	1.8226	(7)	0.8852	(9)	7.8333
IVC-COS	0.9050	(7)	23.0763	(7)	1.0034	(8)	4.6282	(8)	2.6964	(8)	0.8069	(7)	7.5
IVC-PEARSON	0.9051	(8)	23.0767	(8)	1.0047	(9)	4.6284	(9)	2.8212	(9)	0.8464	(8)	8.5
AVG	0.8868	(6)	22.6101	(6)	0.8865	(6)	4.3656	(7)	1.7561	(6)	0.7632	(1)	5.3333

4.4 Parameter Analysis

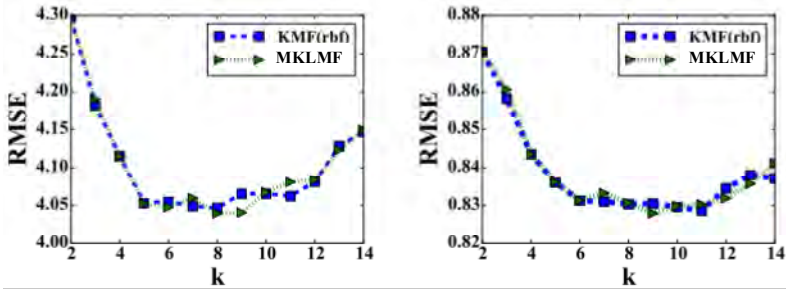
Figure 3 (a) and 3 (b) analyze the sensitivity of the algorithm to the parameter d (the dimension of the dictionary vector). As can be seen from Figure 3, the RMSE values of the KMF algorithm and the MKLMF algorithm are generally at a relatively stable level as the d value increases. For the Yahoo Music data set, the optimal value of parameter d is about 300; for the ASSISTments data set, the optimal value of parameter d is about 100. The two algorithm is similar to other data sets, and is not discussed here in detail due to the limited space. Intuitively, the bigger the parameter d, the more information the dictionary can describe, and the performance of the algorithm depends on the parameter d. However, the dictionary vectors in KMF algorithm and MKLMF algorithm are finally embedded into the Hilbert feature space whose dimension is higher than d, which makes the algorithm less sensitive to parameters.



(a)Yahoo Music data set (b)ASSISTments data set

Fig 3. parameter d (Dictionary vector dimension) compares the algorithm performance with different values.

In addition, we also analyze the influence of parameter k on the performance of our algorithm, where k denotes the rank of two low rank characteristic matrices. It is well known that parameter k should be adjusted by cross-validation or multi-round test data and training data to optimize the matrix factorization method [11]. Figures 4 (a) and 4 (b) show the performance of KMF and MKLMF algorithms on Jester and Flixster datasets with different k values. The optimal values of rank under two data sets are $k=9$. Thereafter, when k increased, KMF and MKLMF appeared to be over fitting. KMF algorithm and MKLMF algorithm have similar performance in other data sets, that is, they have the same k value selection rule as the non-kernel matrix factorization method. In addition, we find that the optimal value of k in ASSISTments data set is about 120, which is not consistent with the principle of $k \ll \min(m,n)$. We speculate that this is because the ASSISTments dataset may contain some special attributes that the low rank matrix factorization method cannot effectively handle.



(a)Jester data set (b)Flixster data set

Fig 4. Algorithm performance comparison of parameters k (rank of characteristic matrix) with different values

5. Concluding Remarks

Two new algorithms of KMF and MKLMF are proposed for collaborative filtering. Both algorithms can simultaneously utilize the nonlinear underlying association between rows (users) and columns (items) of the scoring matrix. Specifically, KMF introduces the accounting method into the matrix factorization problem, embeds the low rank characteristic matrix into the higher dimensional space, and uses the score data of the original space to learn the nonlinear association. MKLMF algorithm further expands KMF, uses the observation data in the score matrix to learn the weight of each kernel function, and then synthesizes multiple kernel functions for collaborative filtering. The simulation results show that the prediction accuracy of this algorithm is better than other current algorithms. In the next step, we will analyze the problem that the discrete score can not reasonably express the user's views and the sparsity of the traditional collaborative filtering algorithm, and study a collaborative filtering algorithm based on trapezoidal fuzzy number.

References

[1]. Silva E Q D, Camilo-Junior C G, Pascoal L M L, et al. An evolutionary approach for combining results of recommender systems techniques based on collaborative filtering [J]. Expert Systems with Applications, 2016, 53(11):204-218.

[2]. Boutet A, Frey D, Guerraoui R, et al. Privacy-preserving distributed collaborative filtering [J]. Computing, 2016, 98(8):827-846.

[3]. Wu Yitao, Zhang Xingming, Wang Xingmao, et al. Collaborative filtering algorithm based on user fuzzy similarity [J]. Journal of Communications, 2016, 37 (1): 198-206.

[4]. Wang Cheng, Zhu Zhigang, Zhang Yuxia, et al. Recommendation Efficiency and Personalized Improvement of User-Based Collaborative Filtering Algorithm [J]. Minicomputer System, 2016, 37 (3): 428-432.

[5]. Li B, Zhu X, Li R, et al. Rating Knowledge Sharing in Cross-Domain Collaborative Filtering [J]. IEEE Transactions on Cybernetics, 2017, 45(5):1068-1082.

[6]. Guan X, Li C T, Guan Y. Matrix Factorization with Rating Completion: An Enhanced SVD Model for Collaborative Filtering Recommender Systems [J]. IEEE Access, 2017, 5(99):27668-27678.

[7]. Bakir C. Collaborative Filtering with Temporal Dynamics with Using Singular Value Decomposition[J]. Tehnicki Vjesnik, 2018, 25(1):130-135.

[8]. Wei S, Ye N, Zhang S, et al. Item-Based Collaborative Filtering Recommendation Algorithm Combining Item Category with Interestingness Measure[C]// International Conference on Computer Science & Service System. IEEE Press, 2012:2038-2041.

[9]. Guan X, Li C T, Guan Y. Matrix Factorization with Rating Completion: An Enhanced SVD Model for Collaborative Filtering Recommender Systems[J]. IEEE Access, 2017, 5(99):27668-27678.

[10]. Ma Jianwei, Chen Honghui, STEPHAN, et al. Service Recommendation Methods Based on Mixed Recommendation and Hidden Markov Model [J]. Journal of Central South University (Natural Science Edition), 2016, 47 (1): 82-90.

[11]. Yang X, Liang C, Zhao M, et al. Collaborative Filtering-Based Recommendation of Online Social Voting [J]. IEEE Transactions on Computational Social Systems, 2017, 4(1):1-13.

[12]. Zhao F, Yan F, Jin H, et al. Personalized Mobile Searching Approach Based on Combining Content-Based Filtering and Collaborative Filtering[J]. IEEE Systems Journal, 2017, 11(1):324-332.

[13]. Gönen M, Kaski S. Kernelized Bayesian Matrix Factorization [J]. IEEE Transactions on Pattern Analysis & Machine Intelligence, 2014, 36(10):2047-2060.

[14]. Tan F, Li L, Zhang Z, et al. A multi-attribute probabilistic matrix factorization model for personalized recommendation [J]. Pattern Analysis & Applications, 2016, 19(3):857-866.

[15]. Tichavský P, Phan A H, Cichocki A. Partitioned Alternating Least Squares Technique for Canonical Polyadic Tensor Decomposition [J]. IEEE Signal Processing Letters, 2016, 23(7):993-997.

[16]. Zhang L, Suganthan P N. Benchmarking Ensemble Classifiers with Novel Co-Trained Kernel Ridge Regression and Random Vector Functional Link Ensembles [Research Frontier] [J]. IEEE Computational Intelligence Magazine, 2017, 12(4):61-72.

[17]. Lu X, Wang Y, Zhou X, et al. Erratum to: A Method for Metric Learning with Multiple-Kernel Embedding [J]. Neural Processing Letters, 2016, 43(3):905-921.

[18]. Ghoshdastidar D, Adsul A P, Dukkipati A. Learning with Jensen-Tsallis Kernels [J]. IEEE Transactions on Neural Networks & Learning Systems, 2016, 27(10):2108-2119.

[19]. Gu Y, Wang Q, Wang H, et al. Multiple Kernel Learning via Low-Rank Nonnegative Matrix Factorization for Classification of Hyperspectral Imagery[J]. IEEE Journal of Selected Topics in Applied Earth Observations & Remote Sensing, 2017, 8(6):2739-2751.

[20]. Gu Y, Liu T, Jia X, et al. Nonlinear Multiple Kernel Learning with Multiple-Structure-Element Extended Morphological Profiles for Hyperspectral Image Classification [J]. IEEE Transactions on Geoscience & Remote Sensing, 2016, 54(6):3235-3247.

[21]. Zhang L W, Ge Y E, Lu Y. An alternating direction method for solving a class of inverse semi-definite quadratic programming problems[J]. Journal of Industrial & Management Optimization, 2017, 12(1):317-336.