IEEE Access
Multidisciplinary | Rapid Review | Open Access Journal

Received January 24, 2019, accepted March 8, 2019, date of publication March 13, 2019, date of current version April 1, 2019.

Digital Object Identifier 10.1109/ACCESS.2019.2904727

Unsupervised Robust Multiple Kernel Learning via Extracting Local and Global Noises

PENG ZHOU[1], FAN YE[1], AND LIANG DU[2]
[1]School of Computer Science and Technology, Anhui University, Hefei 230601, China
[2]School of Computer and Information Technology, Shanxi University, Taiyuan 030006, China

Corresponding author: Fan Ye (yfan@ahu.edu.cn)

This work was supported in part by the National Natural Science Foundation of China under Grant 61806003 and Grant 61502289, and in part by the Key Natural Science Project of Anhui Provincial Education Department under Grant KJ2018A0010 and Grant KJ2018A0011.

ABSTRACT Kernel-based clustering methods can capture the non-linear structure and identify arbitrarily shaped clusters, so they have been widely used in machine learning tasks. Since the performance of kernel methods critically depends on the choices of kernels, multiple kernel learning methods are proposed to alleviate the effort for kernel designing. The conventional multiple kernel learning methods learn a consensus kernel by linearly combining all candidate kernels, whereas ignoring the influence of the noises. To improve the robustness of multiple kernel learning methods, in this paper, we analyze the local and global noises and design regularized terms to characterize them respectively. Then, we propose a novel local and global de-noising multiple kernel learning method which can explicitly extract the local and global noises to recover the clean kernels for multiple kernel learning. After that, a block coordinate descent algorithm is presented to solve the optimization problem. Finally, the extensive experiments on benchmark data sets will demonstrate the effectiveness of the proposed algorithm.

INDEX TERMS Multiple Kernel learning, robust learning, tensor decomposition.

I. INTRODUCTION

Kernel-based clustering algorithms, such as kernel k-means, have the ability to capture the non-linear inherent structure in many real world data sets and thereby usually achieve better clustering performance than linear partition methods [1]. In real world applications, we can construct various kernels by applying different kernel functions. However, the performance of clustering highly depends on the choice of kernels [2]. Unfortunately, it is still a challenge to determine a suitable one among an extensive range of possible kernels for the given data and task in advance. It is more difficult especially in the unsupervised learning tasks such as clustering, because of the absence of labels. To overcome this difficulty, many unsupervised multiple kernel learning methods have been proposed [2]–[8], which aim to learn a consensus kernel from a user-defined pool of kernels.

Conventional unsupervised multiple kernel learning methods learn a consensus kernel by linearly combining a set of candidate kernels. For example, Kumar *et al.* [9] provided a multi-view spectral clustering method which linearly combined spectral embeddings to get the final clustering. Gönen and Margolin [10] proposed a localized multiple kernel k-means method for cancer biology applications. Liu *et al.* [8] proposed a multiple kernel learning method by learning the optimal neighborhood for clustering.

The above methods concentrate on combining all candidate kernels, whereas ignore the robustness of the methods. However, in real world applications, the kernels are often contaminated by noises. For example, since the original data may contain noises and outliers, the kernel constructed with these data will also be contaminated. In addition, even though the data is clean, improper kernel functions may also introduce noises. To alleviate the effort of noises, recently, some robust multiple kernel learning methods [6], [11], [12] are proposed. These methods focus on the noises caused by the corrupted instances, whereas cannot capture the noises induced by kernel functions well. Note that, some multiple kernel learning methods assign larger weight on the more appropriate kernels can capture the noise induced by kernel functions to some extent. However, in these methods, the weight is imposed on all elements of the kernel matrix, and it is a bit too rough. For example, some inappropriate kernels may have a very low weight in these methods, which means all elements including

The associate editor coordinating the review of this manuscript and approving it for publication was Chao Tong.

the useful parts in the kernel matrix will share the same low weight and will not be helpful to the kernel learning.

To handle noises more comprehensively, in this paper, we propose a novel Local and Global De-noising Multiple Kernel Learning method. We observe that the kernel matrix may contain two kinds of noises: one is caused by contaminated instances, and the other is caused by inappropriate kernel functions. Figure 1 illustrates our observations. Figure 1(a) shows the original data set and Figure 1(b) presents a good Gaussian kernel, where the yellow color means the kernel value is large and the blue color means the kernel value is small. For the first kind of noise, once an instance is contaminated by noises (see $\mathbf{x}_2$ in Figure 1(c)), both the corresponding row and column of the kernel matrix will be also contaminated (see Figure 1(d)). Since this kind of noises only affects a part of kernel matrix (i.e., only one row and one column of the kernel matrix), we call them **local noises**. From Figure 1(c) and Figure 1(d), we see that the local noises have some sparsity. For the second kind of noises, since the inappropriate kernel functions may affect all instances (see Figure 1(e) and Figure 1(f)), i.e., most of the kernel values in Figure 1(e) and Figure 1(f) have changed compared with Figure 1(b) we call them **global noises**. Note that Figure 1(e) and Figure 1(f) shows that the noise kernel matrices calculated by the similar kernel functions (Polynomial kernel function in our examples) will have the similar structure.

In this paper, we take these two kinds of noises into consideration, and explicitly extract them to recover clean kernels for multiple kernel learning. For the two kinds of noises, we first stack noise matrices into tensors and then analyze their structures and design regularized terms to characterize them, respectively. In our method, we integrate the de-noising and multiple kernel fusion into a joint learning framework, for the reason that, on one hand, capturing the noises may improve the kernel learning, and on the other hand, learning a consensus kernel may guide to detect noise more accurately. Since the objective function is a joint learning framework and contains complex regularized terms, the optimization is difficult. To solve it effectively, we first relax the objective function and then present a block coordinate descent scheme to optimize it. At last, we compare our method with the state-of-the-art multiple kernel clustering methods on benchmark data sets, and the experimental results demonstrate the effectiveness of our method.

The remainder of this paper is organized as follows: Section II introduces some related works. Section III presents the notation and preliminaries. Section IV introduce our robust multiple kernel learning methods. Section V shows the experimental results. Section VI concludes this paper.

II. RELATED WORK

Multiple kernel learning has been actively studied [4], [8], [10], [13], [14]. Based on the availability of class labels, multiple kernel learning can be categorized into two classes: supervised algorithms and unsupervised methods.

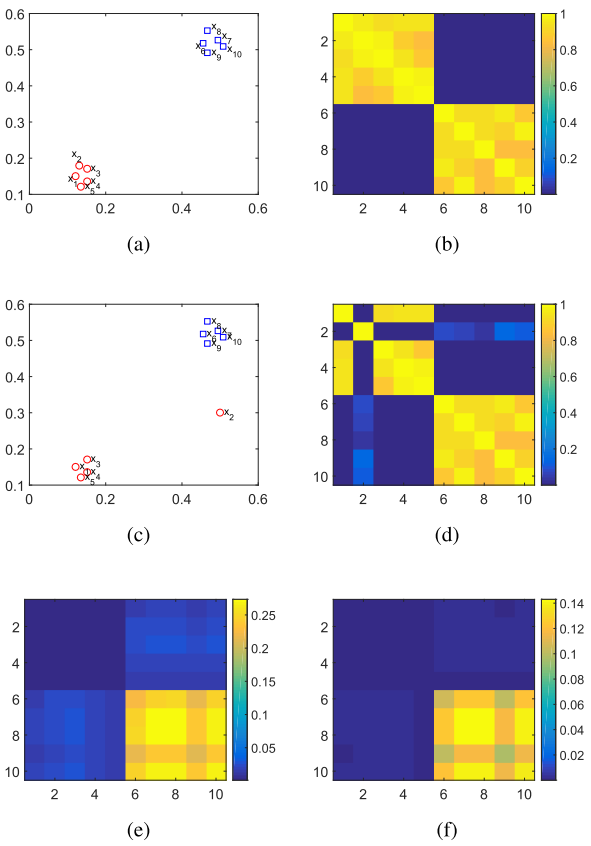


FIGURE 1. (a) The original data set. (b) A good kernel which is a Gaussian kernel $k(\mathbf{x}_i, \mathbf{x}_j) = exp(-\frac{\|\mathbf{x}_i - \mathbf{x}_j\|^2}{2\sigma^2})$ with $\sigma = 0.1$. The yellow color means the kernel value is large and the blue color means the kernel value is small. (c) One instance ($\mathbf{x}_2$) of the data is corrupted with noises. (d) The Gaussian kernel $k(\mathbf{x}_i, \mathbf{x}_j) = exp(-\frac{\|\mathbf{x}_i - \mathbf{x}_j\|^2}{2\sigma^2})$ with $\sigma = 0.1$ of the corrupted data set. (e) An inappropriate Polynomial kernel $k(\mathbf{x}_i, \mathbf{x}_j) = (\mathbf{x}_i^T \mathbf{x}_j)^a$ with $a = 2$ of the original data set. (f) Another inappropriate Polynomial kernel with $a = 3$ of the original data set. (a) Original data. (b) Gaussian kernel of the original data. (c) Corrupted data. (d) The Gaussian kernel of the corrupted data. (e) An polynomial kernel of the original data. (f) Another polynomial kernel of the original data.

In supervised learning, labels of instances are available, which can be helpful to learn a consensus kernel from multiple kernels [13], [15]–[19]. For example, Liu *et al.* [16] integrated radius information into multiple kernel learning to improve the kernel learning performance; Liu *et al.* [19] extended extreme learning machine to multiple kernel learning leading to a multiple kernel extreme learning machine.

Although supervised multiple kernel learning has been extensively studied, unsupervised algorithm is more challenging due to the absence of class labels. In unsupervised learning, some methods extend single kernel based clustering method into multiple kernel setting. For example, Zhao *et al.* [20] proposed a multiple kernel fuzzy clustering method by introducing a matrix-induced regularization; Kang *et al.* [21] provided a self-weighted multiple kernel learning algorithm and applied it to a graph-based clustering method. Since kernel k-means is a popular clustering method, many unsupervised multiple kernel learning methods have

been developed in the framework of it. For example, Tzortzis and Likas [4] developed weighted multiple kernel k-means and multi-view spectral clustering, respectively. Yu *et al.* [2] proposed a multiple kernel k-means where the kernel combinational coefficients are optimized automatically. Huang *et al.* [5] presented a multiple kernel k-means method and extended it to fuzzy k-means. Liu *et al.* [22] proposed a multiple kernel k-means method which focused on integrating incomplete kernels and then they extended it to multi-view clustering [23]. Recently, Liu *et al.* [24] improved the work in [23] for efficiency consideration. Zhu *et al.* [25] proposed a multiple kernel k-means for incomplete kernels which took the local structure among data into consideration.

Due to the connection between kernel k-means and spectral clustering, the latter has also been extended to deal with multiple kernels [1]. Kumar and Daumé [26] presented a co-training approach for multi-view spectral clustering by bootstrapping the clusterings of different views. They [9] also proposed two co-regularization based approaches for multi-view spectral clustering by enforcing the clustering hypotheses on different views to agree with each other. Huang *et al.* [14] aggregated kernels with different weights into a unified one for spectral clustering. By resorting to the spectral method, Gönen and Margolin [10] also proposed to solve a multiple kernel k-means associated with two-layer weights. Similar to [10], Liu *et al.* [7] first presented a multiple kernel k-means method which considered the kernel correlations, and then depending on the results in [1], they transfer the kernel k-means to spectral methods and obtain the clustering results from eigenvalue decomposition. Liu *et al.* [8] proposed a multiple kernel learning method to enhance the representability of the optimal kernel and strengthen the negotiation between kernel learning and clustering.

The above methods do not consider the noises on the kernels. To alleviate the effort of noises, some robust multiple kernel learning methods [6], [11], [12] are proposed. Xia *et al.* [11] proposed a robust multi-view spectral clustering method which learns a consensus matrix from a set of probabilistic transition matrices. Note that, they extracted the sparse noises on the transition matrices instead of the kernel matrix. However, the noises on the kernel matrix are often more sophisticated so that a sparse constraint is too simple to characterize the complex noises. Different from [11], Du *et al.* [12] and Zhou *et al.* [6] tried to characterize the noises on the kernel matrices directly. Du *et al.* [12] present a robust multiple kernel k-means method using $\ell_{2,1}$ loss. Zhou *et al.* [6] extracted the structural noises caused by corrupted instances. Nevertheless, these methods focus on the noises caused by the corrupted instances, whereas cannot capture the noises induced by kernel functions well.

III. NOTATIONS AND PRELIMINARIES

Since this paper will involve some tensor and matrix operations, we first introduce some necessary notions and preliminaries about them in this section.

An $m \times n$ matrix is denoted as $\mathbf{M} \in \mathbb{R}^{m\times n}$, and its (i, j)-th element is denoted as M_{ij}. A tensor of order N is denoted as $\mathcal{T} \in \mathbb{R}^{I_1\times\cdots\times I_N}$. The element of $\mathcal{T}$ is denoted as $T_{i_1\cdots i_N}$. The mode-n vectors of the tensor $\mathcal{T}$ are the I_n dimensional vectors obtained from $\mathcal{T}$ by varying index i_n while keeping the others fixed. The unfolding matrix $\mathbf{T}_{(n)} = \text{unfold}_n(\mathcal{T}) \in \mathbb{R}^{I_n\times(I_1\times\cdots\times I_N)}$ is composed by taking the mode-n vectors of $\mathcal{T}$ as its columns. The unfolding matrices along the n-th mode can be transformed back to the tensor by $\mathcal{T} = \text{fold}_n(\mathbf{T}_{(n)})$.

The mode-n product of a tensor $\mathcal{T} \in \mathbb{R}^{I_1\times\cdots\times I_n\times\cdots\times I_N}$ by a matrix $\mathbf{M} \in \mathbb{R}^{J_n\times I_n}$, denoted by $\mathcal{T} \times_n \mathbf{M}$, is an N-order tensor $\mathcal{C} \in \mathbb{R}^{I_1\times\cdots\times J_n\times\cdots\times I_N}$, with entries: $C_{i_1\cdots i_{n-1}j_n i_{n+1}\cdots i_N} = \sum_{i_n} T_{i_1\cdots i_n\cdots i_N} M_{j_n i_n}$. It is easy to verify that

$$\text{unfold}_n(\mathcal{T} \times_n \mathbf{M}) = \mathbf{M}\mathbf{T}_{(n)}. \tag{1}$$

The Frobenius norm of the matrix $\mathbf{M}$ is denoted by $\|\mathbf{M}\|_F$ and $\|\mathbf{M}\|_F = \left(\sum_{i,j=1}^{m,n} M_{ij}^2\right)^{1/2}$. Similarly, the Frobenius norm of the tensor $\mathcal{T}$ is denoted by $\|\mathcal{T}\|_F = \left(\sum_{i_1,\ldots,i_N=1}^{I_1,\ldots I_N} T_{i_1\ldots i_N}^2\right)^{1/2}$. $\|\mathcal{T}\|_0$ is the ℓ_0 norm of $\mathcal{T}$ which means the number of non-zero elements in $\mathcal{T}$.

Given the tensor $\mathcal{T}$, its Tucker decomposition [27] can be denoted by:

$$\mathcal{T} = \mathcal{S} \times_1 \mathbf{U}_1 \times_2 \cdots \times_N \mathbf{U}_N \tag{2}$$

where $\mathcal{S} \in \mathbb{R}^{r_1\times\cdots\times r_N}$ is called the core tensor, and $\mathbf{U}_i \in \mathbb{R}^{I_i\times r_i}$ is composed by the r_i orthogonal bases along the i-th mode of $\mathcal{T}$.

IV. LOCAL AND GLOBAL DE-NOISING MULTIPLE KERNEL LEARNING

In this section, we present the framework and algorithm of our local and global de-noising multiple kernel learning.

A. FORMULATION

Given a data set with n instances, we construct m kernels $\mathbf{K}^{(1)}, \cdots, \mathbf{K}^{(m)}$ which are $n \times n$ matrices. The task of multiple kernel learning is to learn a consensus kernel matrix $\mathbf{K}^* \in \mathbb{R}^{n\times n}$ from $\mathbf{K}^{(1)}, \cdots, \mathbf{K}^{(m)}$.

As introduced before, the candidate kernels $\mathbf{K}^{(1)}, \cdots, \mathbf{K}^{(m)}$ may be contaminated by two kinds of noises: **local noises**, the noises only appear in a small amount of elements of the kernel matrix and is often induced by outliers or corrupted instances; **global noises**, the noises which appear in most of the elements of the kernel matrix, and is often induced by inappropriate kernels. To obtain the clean consensus kernel, we will handle these two kinds of noises respectively.

For the local noises, we know that, if the j-th instance is corrupted with noises, both the j-th row and the j-th column of the kernel matrix are simultaneously contaminated (See Figure 1(d)). To extract this kind of noises, we use a row-wise sparse matrix $\mathbf{E}^{(i)} \in \mathbb{R}^{n\times n}$ to capture the noises on the rows of the i-th kernel. Naturally, the transpose $\mathbf{E}^{(i)T}$ is a column-wise sparse matrix which captures the noises on the columns. As a result, the symmetric matrix $\mathbf{E}^{(i)} + \mathbf{E}^{(i)T}$ denotes the local noises on the i-th kernel.

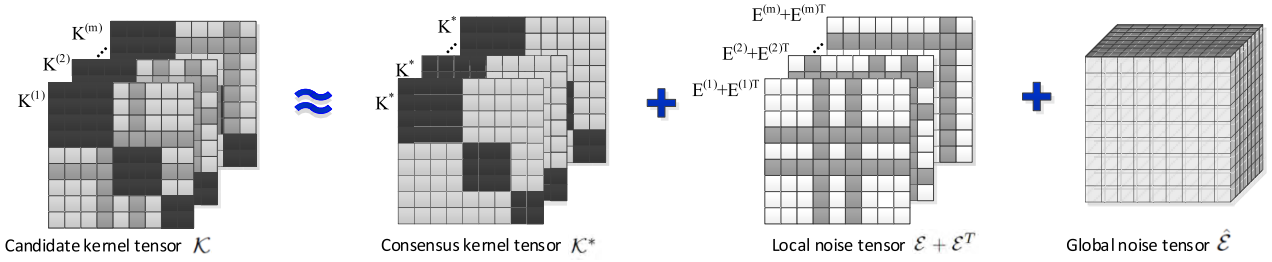


FIGURE 2. The framework of our method.

For the global noises, we also introduce m noise matrices $\hat{\mathbf{E}}^{(i)} \in \mathbb{R}^{n\times n}$ to capture them. Since these noises are caused by kernel functions, the noise matrices of the kernels calculated by the similar kernel functions will have the similar structure (see Figure 1(e) and Figure 1(f)). To characterize the similarity, we stack the m noise matrices $\hat{\mathbf{E}}^{(i)}$ into an $n \times n \times m$ tensor $\hat{\mathcal{E}}$. Since the rank of a tensor reflects the relativity of all the slices in the tensor, we make the noise tensor $\hat{\mathcal{E}}$ low rank to characterize the similarity of the m noise matrices.

Up to now, we introduce some noise matrices and noise tensor to capture the local and global noises. Then we will introduce the formulation of our multiple kernel learning method. Since we use a tensor to capture the global noises, for the simplicity of notation, we also stack the candidate kernels into an $n \times n \times m$ tensor $\mathcal{K}$, where the i-th slice of $\mathcal{K}$ is $\mathbf{K}^{(i)}$. Similarly, we stack $\mathbf{E}^{(i)}$s into a tensor $\mathcal{E}$ and denotes $\mathcal{E}^T$ as the tensor obtained by stacking $\mathbf{E}^{(i)T}$s. At last, we replicate the consensus kernel matrix $\mathbf{K}^*$ m times and also stack them into an $n \times n \times m$ tensor $\mathcal{K}^*$ where each slice of it is $\mathbf{K}^*$. Since $\mathcal{E}+\mathcal{E}^T$ denotes the local noises and $\hat{\mathcal{E}}$ denotes the global noises, the **cleaned** kernel tensor can be characterized by $\mathcal{K}-(\mathcal{E}+\mathcal{E}^T)-\hat{\mathcal{E}}$, i.e., the cleaned kernels are obtained by subtracting the noises form the candidate kernels. To learn the consensus kernel, we wish to minimize the disagreement between the consensus kernel and all the cleaned kernels. So we minimize the following formulation:

$$\begin{aligned}\min_{\mathcal{K}^*,\mathcal{E},\hat{\mathcal{E}}} \ & \left\|\mathcal{K}-(\mathcal{E}+\mathcal{E}^T)-\hat{\mathcal{E}}-\mathcal{K}^*\right\|_F^2+\lambda_1\Omega_1(\mathcal{E}) \\ & +\lambda_2\Omega_2(\hat{\mathcal{E}}), \\ s.t.\ & \mathbf{K}^*=\mathbf{K}^{*T}, \quad \mathbf{K}^*\succeq 0. \end{aligned} \tag{3}$$

where the constraints make sure that the consensus kernel $\mathbf{K}^*$ is a valid kernel matrix which is symmetric and positive semi-definite. The regularized term $\Omega_1(\mathcal{E})$ makes the local noise matrix row sparse and $\Omega_2(\hat{\mathcal{E}})$ makes the global noise tensor low rank. λ_1 and λ_2 are two balancing parameters. Figure 2 illustrates the framework of our method.

Now we introduce the regularized terms. For the $\Omega_1(\cdot)$, since we wish to make $\mathbf{E}^{(i)}$ row sparse, we apply the famous $\ell_{2,1}$-norm on the $\mathbf{E}^{(i)}$, which leads to $\Omega_1(\mathcal{E}) = \sum_{i=1}^m \|\mathbf{E}^{(i)}\|_{2,1}$.

For the low rank regularized term $\Omega_2(\cdot)$, we use the similar regularized term as in [28] and [29]. In more details, given the tensor $\hat{\mathcal{E}} \in \mathbb{R}^{n\times n\times m}$, its Tucker decomposition is written as follows:

$$\hat{\mathcal{E}} = \mathcal{S} \times_1 \mathbf{U}_1 \times_2 \mathbf{U}_2 \times_3 \mathbf{U}_3 \tag{4}$$

where $\mathcal{S} \in \mathbb{R}^{r_1\times r_2\times r_3}$, $\mathbf{U}_1 \in \mathbb{R}^{n\times r_1}$, $\mathbf{U}_2 \in \mathbb{R}^{n\times r_2}$, and $\mathbf{U}_3 \in \mathbb{R}^{m\times r_3}$. r_i $(i = 1, 2, 3)$ is the number of orthogonal bases along the i-th mode of $\hat{\mathcal{E}}$. To make the tensor $\hat{\mathcal{E}}$ low rank, we wish its core tensor as sparse as possible, i.e., we minimize $\|\mathcal{S}\|_0$.

Besides constraining the sparsity of the core tensor, we should also constrain the size of the core tensor. Considering the unfolding matrix $\hat{\mathbf{E}}_{(i)} = \text{unfold}_i(\hat{\mathcal{E}}) \in \mathbb{R}^{I_i\times(I_1\times\cdots\times I_{i-1}\times I_{i+1}\cdots\times I_3)}$, the size of core tensor can be constrained by $\prod_{i=1}^3 \text{rank}\left(\hat{\mathbf{E}}_{(i)}\right)$ according to [28] and [29]. To sum up, the low rank term in our formulation can be defined as follows:

$$\Omega_2(\hat{\mathcal{E}}) = \|\mathcal{S}\|_0 + t\prod_{i=1}^3 \text{rank}\left(\hat{\mathbf{E}}_{(i)}\right) \tag{5}$$

where t is a tradeoff parameter and is set to 10 as suggested in [28].

Taking $\Omega_1(\cdot)$ and $\Omega_2(\cdot)$ into Eq. (3), we get the objective function of our local and global de-noising multiple kernel learning method:

$$\begin{aligned}\min_{\mathcal{K}^*,\mathcal{E},\hat{\mathcal{E}}} \ & \left\|\mathcal{K}-(\mathcal{E}+\mathcal{E}^T)-\hat{\mathcal{E}}-\mathcal{K}^*\right\|_F^2+\lambda_1\sum_{i=1}^m\left\|\mathbf{E}^{(i)}\right\|_{2,1} \\ & +\lambda_2\left(\|\mathcal{S}\|_0+t\prod_{i=1}^3 \text{rank}\left(\hat{\mathbf{E}}_{(i)}\right)\right), \\ s.t.\ & \mathbf{K}^*=\mathbf{K}^{*T}, \quad \mathbf{K}^*\succeq 0. \end{aligned} \tag{6}$$

Note that the key of our formulation is the explicit characterization of the local noises $\mathcal{E}+\mathcal{E}^T$ and the global noises $\hat{\mathcal{E}}$, which makes it more robust. By optimizing Eq. (6), we can de-noise and learn the consensus kernel jointly.

B. OPTIMIZATION

In this subsection, we introduce how to optimize the objective function Eq. (6).

1) RELAXATION

Note that the ℓ_0-norm and rank norm in Eq. (6) are nonconvex and discontinuous, and make it hard to solve. We thus relax them as a log-sum form [28] to simplify the optimization.

Instead of optimizing Eq. (6) directly, we optimize the following relaxation problem:

$$\begin{aligned}\min_{\mathcal{K}^*,\mathcal{E},\hat{\mathcal{E}}} & \left\|\mathcal{K}-(\mathcal{E}+\mathcal{E}^T)-\hat{\mathcal{E}}-\mathcal{K}^*\right\|_F^2+\lambda_1\sum_{i=1}^{m}\left\|\mathbf{E}^{(i)}\right\|_{2,1}\\ &+\lambda_2\left(\mathrm{P}_1(\mathcal{S})+t\prod_{i=1}^{3}\mathrm{P}_2\left(\hat{\mathbf{E}}_{(i)}\right)\right),\\ s.t.\ & \mathbf{K}^*=\mathbf{K}^{*T},\quad \mathbf{K}^*\succeq 0.\end{aligned}\tag{7}$$

where $\mathrm{P}_1(\mathcal{S})=\sum_{i_1,i_2,i_3}\log\left(\frac{|S_{i_1i_2i_3}|+\epsilon}{\epsilon}\right)$ is the approximation of ℓ_0-norm on $\mathcal{S}$ where $S_{i_1i_2i_3}$s are all elements in $\mathcal{S}$ and ϵ is a small constant (we fix it to 2^{-52} as the default value in Matlab). $\mathrm{P}_2\left(\hat{\mathbf{E}}_{(i)}\right)=\sum_j\log\left(\frac{\sigma_j\left(\hat{\mathbf{E}}_{(i)}\right)+\epsilon}{\epsilon}\right)$ where $\sigma_j\left(\hat{\mathbf{E}}_{(i)}\right)$ denotes the j-th singular value of $\hat{\mathbf{E}}_{(i)}$. It is easy to verify that $\mathrm{P}_2(\cdot)$ approximates the rank norm.

Now we present a block coordinate descent scheme to optimize Eq. (7), i.e., we optimize the objective with respect to one variable while fixing the other variables.

2) OPTIMIZE $\hat{\mathcal{E}}$

When $\mathcal{K}^*$ and $\mathcal{E}$ are fixed, taking the Tucker decomposition of $\hat{\mathcal{E}}$ into Eq. (7), Eq. (7) can be rewritten as:

$$\begin{aligned}\min_{\mathcal{S},\mathbf{U}_i} & \|\mathcal{A}-\mathcal{S}\times_1\mathbf{U}_1\times_2\mathbf{U}_2\times_3\mathbf{U}_3\|_F^2\\ &+\lambda_2\left(\mathrm{P}_1(\mathcal{S})+t\prod_{i=1}^{3}\mathrm{P}_2\left(\hat{\mathbf{E}}_{(i)}\right)\right)\\ s.t.\ & \mathbf{U}_i^T\mathbf{U}_i=\mathbf{I},\quad i=1,2,3.\end{aligned}\tag{8}$$

where $\mathcal{A}=\mathcal{K}-(\mathcal{E}+\mathcal{E}^T)-\mathcal{K}^*$.

To solve Eq. (8), we introduce three auxiliary tensors $\mathcal{M}_i$ ($i=1,2,3$) and apply Alternating Direction Method of Multipliers (ADMM) [30] to optimize it.

We first rewrite Eq. (8) as following equivalent formula:

$$\begin{aligned}\min_{\mathcal{S},\mathbf{U}_i,\mathcal{M}_i} & \|\mathcal{A}-\mathcal{S}\times_1\mathbf{U}_1\times_2\mathbf{U}_2\times_3\mathbf{U}_3\|_F^2\\ &+\lambda_2\left(\mathrm{P}_1(\mathcal{S})+t\prod_{i=1}^{3}\mathrm{P}_2\left(\mathbf{M}_{i(i)}\right)\right)\\ s.t.\ & \mathcal{S}\times_1\mathbf{U}_1\times_2\mathbf{U}_2\times_3\mathbf{U}_3=\mathcal{M}_i,\quad \mathbf{U}_i^T\mathbf{U}_i=\mathbf{I}.\end{aligned}\tag{9}$$

where $\mathbf{M}_{i(i)}=\mathrm{unfold}_i(\mathcal{M}_i)$. The augmented Lagrangian function of Eq. (9) is:

$$\begin{aligned}L=&\|\mathcal{A}-\mathcal{S}\times_1\mathbf{U}_1\times_2\mathbf{U}_2\times_3\mathbf{U}_3\|_F^2\\ &+\lambda_2\left(\mathrm{P}_1(\mathcal{S})+t\prod_{i=1}^{3}\mathrm{P}_2\left(\mathbf{M}_{i(i)}\right)\right)\\ &+\sum_{i=1}^{3}\langle\mathcal{S}\times_1\mathbf{U}_1\times_2\mathbf{U}_2\times_3\mathbf{U}_3-\mathcal{M}_i,\mathcal{P}_i\rangle\\ &+\sum_{i=1}^{3}\frac{\mu}{2}\|\mathcal{S}\times_1\mathbf{U}_1\times_2\mathbf{U}_2\times_3\mathbf{U}_3-\mathcal{M}_i\|_F^2.\end{aligned}\tag{10}$$

where $\mathcal{P}_i$s are Lagrange multipliers, μ is a positive scalar, and $\langle\cdot,\cdot\rangle$ denotes the inner production. Then we introduce how to solve $\mathcal{S}$, $\mathbf{U}_i$, $\mathcal{M}_i$ and update $\mathcal{P}_i$ for $i=1,2,3$.

a: SOLVING $\mathcal{S}$

When optimizing $\mathcal{S}$ with other variables fixed, we rewrite Eq. (10) as:

$$\min_{\mathcal{S}}\ \|\mathcal{S}\times_1\mathbf{U}_1\times_2\mathbf{U}_2\times_3\mathbf{U}_3-\mathcal{O}\|_F^2+b\mathrm{P}_1(\mathcal{S}).\tag{11}$$

where $\mathcal{O}=\frac{1}{2+3\mu}\left(2\mathcal{A}+\mu\sum_{i=1}^{3}\mathcal{M}_i-\sum_{i=1}^{3}\mathcal{P}_i\right)$, and $b=\frac{2\lambda_2}{2+3\mu}$. Since for any tensor $\mathcal{D}$ and any orthogonal matrix $\mathbf{U}$ where $\mathbf{U}^T\mathbf{U}=\mathbf{I}$, we have

$$\|\mathcal{D}\times_i\mathbf{U}\|_F^2=\|\mathcal{D}\|_F^2,\tag{12}$$

by mode-i producting $\mathbf{U}_i^T$ on each mode we can rewrite Eq. (11) as:

$$\min_{\mathcal{S}}\ \|\mathcal{S}-\mathcal{Q}\|_F^2+b\mathrm{P}_1(\mathcal{S}).\tag{13}$$

where $\mathcal{Q}=\mathcal{O}\times_1\mathbf{U}_1^T\times_2\mathbf{U}_2^T\times_3\mathbf{U}_3^T$. Taking $\mathrm{P}_1(\cdot)$ into it and setting the derivative of it w.r.t $S_{i_1i_2i_3}$ to zero, we can get the closed-form solution: $\mathcal{S}=\mathrm{D}_{b,\epsilon}(\mathcal{Q})$, where $\mathrm{D}_{b,\epsilon}(\cdot)$ is the thresholding operator defined as:

$$\mathrm{D}_{b,\epsilon}(x)=\begin{cases}\mathrm{sign}(x)\left(\dfrac{c_1+\sqrt{c_2}}{2}\right), & \text{if } c_2>0\\ 0, & \text{if } c_2\le 0\end{cases}\tag{14}$$

where $c_1=|x|-\epsilon$ and $c_2=c_1^2-4(b-\epsilon|x|)$.

b: SOLVING $\mathbf{U}_i$

When optimizing $\mathbf{U}_1$ with other variables fixed, we will optimize the following subproblem:

$$\begin{aligned}\min_{\mathbf{U}_1}\ & \|\mathcal{S}\times_1\mathbf{U}_1\times_2\mathbf{U}_2\times_3\mathbf{U}_3-\mathcal{O}\|_F^2\\ s.t.\ & \mathbf{U}_1^T\mathbf{U}_1=\mathbf{I}\end{aligned}\tag{15}$$

by unfolding the Frobenius norm, we obtain:

$$\begin{aligned}&\min_{\mathbf{U}_1}\ \|\mathcal{S}\times_1\mathbf{U}_1\times_2\mathbf{U}_2\times_3\mathbf{U}_3-\mathcal{O}\|_F^2\\ &=\min_{\mathbf{U}_1}\ \|\mathcal{S}\times_1\mathbf{U}_1\times_2\mathbf{U}_2\times_3\mathbf{U}_3\|_F^2\\ &\quad-2\langle\mathcal{S}\times_1\mathbf{U}_1\times_2\mathbf{U}_2\times_3\mathbf{U}_3,\mathcal{O}\rangle\\ &=\max_{\mathbf{U}_1}\ \langle\mathcal{S}\times_1\mathbf{U}_1\times_2\mathbf{U}_2\times_3\mathbf{U}_3,\mathcal{O}\rangle\end{aligned}\tag{16}$$

The second equality is because that we have

$$\|\mathcal{S}\times_1\mathbf{U}_1\times_2\mathbf{U}_2\times_3\mathbf{U}_3\|_F^2=\|\mathcal{S}\|_F^2\tag{17}$$

when $\mathbf{U}_i$ ($i=1,2,3$) is orthogonal according to Eq. (12).

Due to Eq. (1), we have

$$\begin{aligned}&\mathrm{unfold}_1\left(\mathcal{S}\times_1\mathbf{U}_1\times_2\mathbf{U}_2\times_3\mathbf{U}_3\right)\\ &\qquad=\mathbf{U}_1\mathrm{unfold}_1\left(\mathcal{S}\times_2\mathbf{U}_2\times_3\mathbf{U}_3\right).\end{aligned}\tag{18}$$

Therefore, we can rewrite Eq. (15) as:

$$\begin{aligned}\max_{\mathbf{U}_1}\ & \langle\mathbf{B}_1,\mathbf{U}_1\rangle,\\ s.t.\ & \mathbf{U}_1^T\mathbf{U}_1=\mathbf{I}\end{aligned}\tag{19}$$

where $\mathbf{B}_1 = \mathbf{O}_{(1)}\,(\text{unfold}_1\,(\mathcal{S} \times_2 \mathbf{U}_2 \times_3 \mathbf{U}_3))^T$. Eq. (19) can be solved by the following Theorem.

Theorem 1: Let the Singular Value Decomposition (SVD) of $\mathbf{B}_1$ *be* $\mathbf{B}_1 = \mathbf{L}\mathbf{\Sigma}\mathbf{R}^T$, *then the optima of Eq.* (19) *is* $\mathbf{U}_1 = \mathbf{L}\mathbf{R}^T$.

Proof: According to Von Neumanns trace inequality, we have $tr(\mathbf{B}_1\mathbf{U}_1^T) \leq tr(\mathbf{\Sigma})$. So we get

$$\begin{aligned}\langle \mathbf{B}_1, \mathbf{U}_1 \rangle &= tr(\mathbf{B}_1\mathbf{U}_1^T) = tr(\mathbf{L}\mathbf{\Sigma}\mathbf{R}^T\mathbf{U}_1^T) \\ &= tr(\mathbf{\Sigma}\mathbf{R}^T\mathbf{U}_1^T\mathbf{L}) \leq tr(\mathbf{\Sigma})\end{aligned} \tag{20}$$

where $tr(\cdot)$ is the trace of a matrix. Therefore, $\langle \mathbf{B}_1, \mathbf{U}_1 \rangle$ takes the maximum value when $\mathbf{R}^T\mathbf{U}_1^T\mathbf{L} = \mathbf{I}$, which means $\mathbf{U}_1 = \mathbf{L}\mathbf{R}^T$. □

For efficiency consideration, we use the randomized truncated SVD [31] throughout this paper. The optimization of $\mathbf{U}_2$ and $\mathbf{U}_3$ is similar.

c: SOLVING $\mathcal{M}_i$

When optimizing $\mathcal{M}_1$ with other variable fixed, we need to optimize the following formula:

$$\min_{\mathcal{M}_1} \|\mathcal{M}_1 - \mathcal{F}\|_F^2 + a\text{P}_2(\mathbf{M}_{1(1)}) \tag{21}$$

where $\mathcal{F} = \mathcal{S} \times_1 \mathbf{U}_1 \times_2 \mathbf{U}_2 \times_3 \mathbf{U}_3 + \frac{1}{\mu}\mathcal{P}_1$ and $a = \frac{2\lambda_2 t}{\mu}\text{P}_2(\mathbf{M}_{2(2)})\text{P}_2(\mathbf{M}_{3(3)})$.

Note that Eq. (21) is similar to Eq. (13), we use the similar method to solve it. First, we get $\mathbf{F}_{(1)}$ by unfolding $\mathcal{F}$ by the mode-1, and calculate its SVD decomposition $\mathbf{F}_{(1)} = \mathbf{V}_1\text{diag}(\sigma_1, \cdots, \sigma_n)\mathbf{V}_2^T$. Then we construct the diagonal matrix $\mathbf{\Sigma}_M = \text{diag}(\text{D}_{a,\epsilon}(\sigma_1), \cdots, \text{D}_{a,\epsilon}(\sigma_n))$, where $\text{D}_{a,\epsilon}(\cdot)$ is the thresholding operator defined before. At last, we calculate $\mathcal{M}_1 = \text{fold}_1(\mathbf{V}_1\mathbf{\Sigma}_M\mathbf{V}_2^T)$. The calculation of $\mathcal{M}_2$ and $\mathcal{M}_3$ is similar.

We update the Lagrange multipliers $\mathcal{P}_i$ $(i = 1, 2, 3)$ as follows:

$$\mathcal{P}_i = \mathcal{P}_i + \mu\,(\mathcal{S} \times_1 \mathbf{U}_1 \times_2 \mathbf{U}_2 \times_3 \mathbf{U}_3 - \mathcal{M}_i) \tag{22}$$

and update $\mu = 1.2\mu$.

After using the ADMM algorithm, we obtain $\hat{\mathcal{E}} = \mathcal{S} \times_1 \mathbf{U}_1 \times_2 \mathbf{U}_2 \times_3 \mathbf{U}_3$. Since the kernel matrix is symmetric, we wish the noise matrix on the kernel is also symmetric. So we get $\hat{\mathbf{E}}^{(i)}$s as m slices of $\hat{\mathcal{E}}$, and update $\hat{\mathbf{E}}^{(i)} = \left(\hat{\mathbf{E}}^{(i)} + \hat{\mathbf{E}}^{(i)T}\right)/2$. Finally, we stack them back to $\hat{\mathcal{E}}$.

3) OPTIMIZE $\mathcal{E}$

When $\mathcal{K}^*$ and $\hat{\mathcal{E}}$ are fixed, Eq. (7) can be decoupled into m independent subproblems, where each subproblem involves only one $\mathbf{E}^{(i)}$. Consider the i-th one:

$$\min_{\mathbf{E}^{(i)}} \left\|\mathbf{G} - \mathbf{E}^{(i)} - \mathbf{E}^{(i)T}\right\|_F^2 + \lambda_1 \left\|\mathbf{E}^{(i)}\right\|_{2,1} \tag{23}$$

where $\mathbf{G} = \mathbf{K}^{(i)} - \mathbf{K}^* - \hat{\mathbf{E}}^{(i)}$. It is easy to verify that $\mathbf{G}$ is symmetric.

We introduce a diagonal matrix $\mathbf{H}$, where its j-th diagonal element is $\frac{1}{2\left\|E_j^{(i)}\right\|_2}$, and $\left\|E_j^{(i)}\right\|_2$ is the ℓ_2-norm of the j-th row of $\mathbf{E}^{(i)}$. Then we try to minimize the auxiliary function:

$$\min_{\mathbf{E}^{(i)}} \left\|\mathbf{G} - \mathbf{E}^{(i)} - \mathbf{E}^{(i)T}\right\|_F^2 + \lambda_1 tr\left(\mathbf{E}^{(i)T}\mathbf{H}\mathbf{E}^{(i)}\right). \tag{24}$$

By setting its derivative w.r.t. $\mathbf{E}^{(i)}$ to zero, we obtain

$$(\mathbf{I} + \lambda_1\mathbf{H})\,\mathbf{E}^{(i)} + \mathbf{E}^{(i)T} = \mathbf{G}^T \tag{25}$$

Denote H_{jj} as the (j, j)-th element of $\mathbf{H}$, $E_{jk}^{(i)}$ as the (j, k)-th element in $\mathbf{E}^{(i)}$, and $G_{jk}^{(i)}$ as the (j, k)-th element in $\mathbf{G}$. Considering the (j, k)-th element and (k, j)-th element on both sides of Eq. (25), we obtain

$$\begin{cases}(1 + \lambda_1 H_{jj}^{(i)})E_{jk}^{(i)} + E_{kj}^{(i)} = G_{jk}^{(i)} \\ (1 + \lambda_1 H_{kk}^{(i)})E_{kj}^{(i)} + E_{jk}^{(i)} = G_{kj}^{(i)}\end{cases} \tag{26}$$

Since $\mathbf{G}$ is symmetric, we can obtain the result by solving Eq. (26) as:

$$E_{kj}^{(i)} = \frac{H_{jj}^{(i)}G_{kj}^{(i)}}{\lambda_1 H_{kk}^{(i)}H_{jj}^{(i)} + H_{kk}^{(i)} + H_{jj}^{(i)}}. \tag{27}$$

Theorem 2: Applying Eq. (27) *to update* $\mathcal{E}$ *monotonically decreases the objective function Eq.* (23).

Proof: The detailed proof is similar as that in [32]. □

4) OPTIMIZE $\mathcal{K}^*$

When $\mathcal{E}$ and $\hat{\mathcal{E}}$ are fixed, Eq. (7) can be rewritten as:

$$\begin{aligned}&\min_{\mathcal{K}^*} \left\|\mathcal{J} - \mathcal{K}^*\right\|_F^2 \\ &s.t.\ \mathbf{K}^* = \mathbf{K}^{*T}, \quad \mathbf{K}^* \succeq 0.\end{aligned} \tag{28}$$

where $\mathcal{J} = \mathcal{K} - (\mathcal{E} + \mathcal{E}^T) - \hat{\mathcal{E}}$.

Since $\mathcal{K}^*$ is obtained by stacking m $\mathbf{K}^*$s, we slice $\mathcal{J}$ into m $n \times n$ matrices: $\mathbf{J}^{(1)}, \cdots, \mathbf{J}^{(n)}$, and denote $\mathbf{J} = \frac{1}{m}\sum_{i=1}^m \mathbf{J}^{(i)}$. Then we rewrite Eq. (28) as:

$$\begin{aligned}&\min_{\mathbf{K}^*} \left\|\mathbf{K}^* - \mathbf{J}\right\|_F^2, \\ &s.t.\ \mathbf{K}^* = \mathbf{K}^{*T}, \quad \mathbf{K}^* \succeq 0.\end{aligned} \tag{29}$$

The closed-form solution of Eq. (29) can be obtained by the following Theorem.

Theorem 3: Let the eigenvalue decomposition of $\mathbf{J}$ *is* $\mathbf{J} = \mathbf{W}\mathbf{\Theta}\mathbf{W}^T$ *where* $\mathbf{W}$ *contains the eigenvectors of* $\mathbf{J}$ *and* $\mathbf{\Theta}$ *is a diagonal matrix whose diagonal elements are the eigenvalues of* $\mathbf{J}$. *Then we set the negative eigenvalues to zero and obtain a new diagonal matrix* $\tilde{\mathbf{\Theta}}$. *The closed-form solution of Eq.* (29) *is* $\mathbf{K}^* = \mathbf{W}\tilde{\mathbf{\Theta}}\mathbf{W}^T$.

Proof: Let the SVD decomposition of $\mathbf{K}^*$ is $\mathbf{K}^* = \mathbf{U}_K\mathbf{\Sigma}_K\mathbf{V}_K^T$. Then we have

$$\left\|\mathbf{K}^* - \mathbf{J}\right\|_F^2 = tr(\mathbf{\Sigma}_K^2) - 2tr(\mathbf{K}^*\mathbf{J}) + tr(\mathbf{J}\mathbf{J}^T) \tag{30}$$

According to Von Neumanns trace inequality, we have $tr(\mathbf{K}^*\mathbf{J}) \leq tr(\mathbf{\Sigma}_K\mathbf{\Theta})$. Then

$$\begin{aligned}tr(\mathbf{U}_K\mathbf{\Sigma}_K\mathbf{V}_K\mathbf{J}) &= tr(\mathbf{K}^*\mathbf{J}) \leq tr(\mathbf{\Sigma}_K\mathbf{\Theta}) \\ &= tr(\mathbf{W}\mathbf{\Sigma}_K\mathbf{W}^T\mathbf{W}\mathbf{\Theta}\mathbf{W}^T) = tr(\mathbf{W}\mathbf{\Sigma}_K\mathbf{W}^T\mathbf{J})\end{aligned} \tag{31}$$

which leads to

$$\|\mathbf{K}^* - \mathbf{J}\|_F^2 \geq \|\mathbf{W}\mathbf{\Sigma}_K\mathbf{W}^T - \mathbf{J}\|_F^2 \tag{32}$$

Thus, to minimize Eq. (29), we should set $\mathbf{U}_K = \mathbf{V}_K = \mathbf{W}$. Taking it back to Eq. (29), we obtain:

$$\min_{\mathbf{K}^*} \left\|\mathbf{K}^* - \mathbf{J}\right\|_F^2 = \min_{\mathbf{\Sigma}_K \succeq 0} \|\mathbf{\Sigma}_K - \mathbf{\Theta}\|_F^2 \tag{33}$$

Obviously, the solution of Eq. (33) is that $\mathbf{\Sigma}_K = \tilde{\mathbf{\Theta}}$. It is easy to verify that $\mathbf{K}^* = \mathbf{W}\tilde{\mathbf{\Theta}}\mathbf{W}^T$ satisfies the positive semi-definite and symmetric constraints.

To sum up, the closed-form solution of Eq. (29) is $\mathbf{K}^* = \mathbf{W}\tilde{\mathbf{\Theta}}\mathbf{W}^T$. □

Algorithm 1 summarizes the whole optimization process. After obtaining the consensus kernel matrix $\mathbf{K}^*$, we can run any kernel based clustering methods on it, such as kernel k-means or spectral clustering.

Algorithm 1 Local and Global De-Noising Multiple Kernel Learning

Input: m candidate kernels $\mathbf{K}^{(1)}, \cdots, \mathbf{K}^{(m)}$, parameters λ_1 and λ_2, $\mu^{(0)}$.

Output: Consensus kernel matrix $\mathbf{K}^*$.

1: Stack $\mathbf{K}^{(1)}, \cdots, \mathbf{K}^{(m)}$ into a tensor $\mathcal{K}$.
2: Initialize $\mathbf{K}^* = \frac{1}{m}\sum_{i=1}^m \mathbf{K}^{(i)}$ and replicate it m times to get tensor $\mathcal{K}^*$. Initialize $\hat{\mathcal{E}} = 0$, $\mu = \mu^{(0)}$.
3: **while** not converge **do**
4: Compute $\mathcal{E}$ using Eq. (27).
5: //Using ADMM to update $\hat{\mathcal{E}}$.
6: **while** not converge **do**
7: Update $\mathcal{S}$ by Eq. (14).
8: Update $\mathbf{U}_i$ $(i = 1, 2, 3)$ by optimizing Eq. (19).
9: Update $\mathcal{M}_i$ $(i = 1, 2, 3)$ by optimizing Eq. (21).
10: Update $\mathbf{P}_i$ $(i = 1, 2, 3)$ by Eq. (22).
11: Update μ by $\mu = 1.2\mu$.
12: **end while**
13: Set $\hat{\mathcal{E}} = \mathcal{S} \times_1 \mathbf{U}_1 \times_2 \mathbf{U}_2 \times_3 \mathbf{U}_3$.
14: Update $\hat{\mathbf{E}}^{(i)} = \left(\hat{\mathbf{E}}^{(i)} + \hat{\mathbf{E}}^{(i)T}\right)/2$.
15: Compute $\mathbf{K}^*$ by optimizing Eq. (29).
16: **end while**

C. CONVERGENCE ANALYSIS

As introduced above, updating $\mathcal{E}$ and $\mathcal{K}^*$ can make the objective function decrease monotonically. However, the convergence of the ADMM used to update $\hat{\mathcal{E}}$ is difficult to analyze due to the non-convexity of our model. Fortunately, [29] provides a weak convergence result of the ADMM algorithm as follows:

*Theorem 4 [29]: Let $\mathcal{S}^{(l)}$, $\mathbf{U}_i^{(l)}$, $\mathcal{M}_i^{(l)}$ denote the value of $\mathcal{S}$, $\mathbf{U}_i$, $\mathcal{M}_i$ in the l-th iteration of the inner **while** cycle (Lines 6-12) in Algorithm 1, respectively. Then, we denote $\hat{\mathcal{E}}^{(l)} = \mathcal{S}^{(l)} \times_1 \mathbf{U}_1^{(l)} \times_2 \mathbf{U}_2^{(l)} \times_3 \mathbf{U}_3^{(l)}$. We have:*

$$\left\|\hat{\mathcal{E}}^{(l)} - \mathcal{M}_i^{(l)}\right\|_F = O\left(\mu^{(0)}/1.2^{l/2}\right)$$
$$\left\|\hat{\mathcal{E}}^{(l+1)} - \hat{\mathcal{E}}^{(l)}\right\|_F = O\left(\mu^{(0)}/1.2^{l/2}\right)$$

TABLE 1. Description of the data sets.

	#instances	#features	#classes
BBCSport	737	1000	5
Coil	1440	1024	20
Glass	214	9	6
Hitech	2301	22498	6
Iris	150	4	3
Lung	203	3312	5
Tr31	927	10128	7
UMIST	575	644	20
Webace	2340	1000	20
Texas	814	4029	7
Cornell	827	4134	7
Washington	1166	4165	7
Wisconsin	1210	4189	7
Spambase	4601	57	2

It implies that the difference of $\hat{\mathcal{E}}^{(l)}$ in two adjacent iterations becomes smaller and smaller. In addition, with iterations, $\mathcal{S} \times_1 \mathbf{U}_1 \times_2 \mathbf{U}_2 \times_3 \mathbf{U}_3$ is increasingly closer to $\mathcal{M}_i$.

D. TIME AND SPACE COMPLEXITY

When updating $\mathcal{E}$, we need to compute $n \times n \times m$ values in $\mathcal{E}$ by Eq. (27) which costs $O(n^2m)$ time. When optimizing $\mathbf{K}^*$, we need to compute the SVD of an $n \times n$ matrix $\mathbf{J}$ which costs $O(n^3)$ time.

In the ADMM step, when updating $\mathcal{S}$, the most expensive step is computing $\mathcal{Q} = \mathcal{O} \times_1 \mathbf{U}_1^T \times_2 \mathbf{U}_2^T \times_3 \mathbf{U}_3^T$, whose time complexity is $O(r_1mn^2 + r_1r_2mn + r_1r_2r_3m)$. When optimizing $\mathbf{U}_1$, we first compute $\mathbf{B}_1$ in Eq. (19) in $O(r_1mn^2 + (r_1r_2r_3 + r_1r_3m)n)$ time, and then compute the SVD of $\mathbf{B}_1$ (an $n \times r_1$ matrix) in $O(nr_1^2)$ time. Moreover, we can use randomize truncated SVD to reduce the time complexity to $O(nr_1log(k) + k^2(n + r_1))$ where k is the number of singular values retained to truncate the approximate SVD and satisfies $k \ll r_1$. The complexity of computing $\mathbf{U}_2$ and $\mathbf{U}_3$ is similar. When optimizing $\mathcal{M}_1$, the main cost lies in computing the SVD of $\mathbf{F}_{(1)}$ (an $n \times (mn)$ matrix), which costs $O(n^3m)$ time and if we use randomized truncated SVD, the time complexity is reduced to $O(mn^2log(k) + k^2mn)$. The complexity of computing $\mathcal{M}_2$ and $\mathcal{M}_3$ is similar. In addition, SVD can be deployed on a distributed platform. For example, the MLlib provides a highly scalable and efficient implementation on Spark.[1] Obviously, tensor and matrix multiplications can also easily be implemented on a distributed platform.

Since we need to save and process several $n \times n \times m$ tensors, the space complexity of our method is $O(n^2m)$.

V. EXPERIMENTS

In this section, we evaluate the effectiveness of the proposed method by comparing with several state-of-the-art multiple kernel clustering methods on benchmark data sets.

[1] https://spark.apache.org/docs/latest/mllib-dimensionality-reduction.html

TABLE 2. Clustering results (ACC) with kernel k-means based methods.

Data	KKM-b	KKM-a	KKM-ew	MKKM	RMKKM	RMKC-KKM	Ours-KKM
BBCSport	0.6047	0.4202	0.4824	0.5757	0.5896	0.6450	**0.6706**
Coil	0.5117	0.4609	0.4824	0.5481	0.5226	0.5289	**0.5685**
Glass	0.5332	0.4927	0.5252	0.5112	0.4089	0.5411	**0.6500**
Hitech	0.4415	0.3045	0.3472	0.3371	0.3483	0.4605	**0.5025**
Iris	0.9453	0.8267	0.8993	0.8707	0.8867	0.8347	**0.9800**
Lung	0.6685	0.4357	0.5232	0.5458	0.5414	**0.7182**	0.5517
Tr31	0.4935	0.3407	0.4554	0.4710	0.4631	**0.5358**	0.5191
UMIST	0.3896	0.3012	0.3720	0.3903	0.3694	0.4014	**0.4363**
WEBACE	0.3787	0.2781	0.3175	0.3257	**0.3629**	0.3234	**0.3621**
Texas	0.5151	0.4323	0.3773	0.3937	0.3571	0.5297	**0.5674**
Cornell	0.5114	0.4050	0.3782	0.4277	0.4087	**0.4972**	**0.4948**
Washington	0.5818	0.4358	0.4564	**0.4940**	0.3566	**0.4904**	**0.4937**
Wisconsin	0.5422	0.4327	0.4493	**0.4793**	0.4132	**0.4777**	**0.4795**
Spambase	0.6688	0.6340	0.6664	0.6664	0.6477	0.6359	**0.6690**

TABLE 3. Clustering results (Purity) with kernel k-means based methods.

Data	KKM-b	KKM-a	KKM-ew	MKKM	RMKKM	RMKC-KKM	Ours-KKM
BBCSport	0.6351	0.4949	0.6134	0.3776	0.6251	0.6843	**0.7090**
Coil	0.5469	0.4956	0.5143	0.5739	0.5428	0.5675	**0.6025**
Glass	0.5715	0.5442	0.5659	0.5570	0.4706	0.5944	**0.7523**
Hitech	0.5211	0.3596	0.4345	0.4200	0.4281	0.5494	**0.5673**
Iris	0.9453	0.8360	0.9000	0.8707	0.8873	0.8467	**0.9800**
Lung	0.8054	0.7260	0.7749	**0.7926**	0.7369	0.7458	**0.7990**
Tr31	0.6738	0.5238	0.6676	0.6368	0.6687	0.7070	**0.7152**
UMIST	0.4536	0.3449	0.4283	0.4569	0.4163	**0.4783**	**0.4783**
WEBACE	0.5401	0.3910	0.4491	0.4686	**0.4944**	0.4736	**0.4854**
Texas	0.7601	0.7174	0.6930	0.6961	0.7555	0.7624	**0.8445**
Cornell	0.7586	0.7221	0.7064	0.7047	0.7503	0.7497	**0.8139**
Washington	0.7944	0.7847	0.7798	0.7834	0.7944	0.7917	**0.8277**
Wisconsin	0.7574	0.7474	0.7493	0.7460	0.7679	0.7554	**0.7864**
Spambase	0.6688	0.6346	0.6664	0.6664	0.6477	0.6359	**0.6690**

TABLE 4. Clustering results (ACC) with spectral clustering based methods.

Data	SC-b	SC-a	SC-ew	CoregSC	LMKKM	RMSC	RMKC-SC	ONKC	Ours-SC
BBCSport	0.5411	0.4809	0.5680	0.5866	0.5407	0.5475	0.5582	0.5820	**0.6354**
Coil	0.6272	0.5585	0.6326	0.6251	0.6255	0.5724	0.6172	**0.6419**	**0.6378**
Glass	0.5098	0.4603	0.4579	0.4850	4682	0.5178	0.4963	0.5238	**0.5893**
Hitech	0.4521	0.3418	0.4222	0.4448	0.3847	**0.5740**	0.4994	0.3869	0.5658
Iris	0.9000	0.7883	0.8667	0.9240	0.8800	0.9000	0.7647	0.8933	**0.9553**
Lung	0.6931	0.5401	0.5463	0.5611	0.5355	0.5325	0.5601	0.5251	**0.7020**
Tr31	0.6143	0.4836	0.4374	0.4715	0.4324	0.4625	0.4915	0.5176	**0.5577**
UMIST	0.4605	0.4079	0.4268	0.4228	0.4289	0.3717	0.4110	0.3969	**0.4403**
WEBACE	0.4496	0.3501	0.4265	0.4224	0.4127	0.3124	**0.4362**	0.4100	**0.4398**
Texas	0.6558	0.5065	0.3776	0.4281	0.4742	0.4322	0.4183	0.5257	**0.7349**
Cornell	0.6800	0.4691	0.3862	0.4002	0.4293	0.4056	0.4657	0.4549	**0.7354**
Washington	0.7629	0.5297	0.3846	0.4002	0.3803	0.3912	0.7606	0.6689	**0.7711**
Wisconsin	0.7168	0.4541	0.4236	0.3946	0.4515	0.3732	0.4673	0.4621	**0.4821**
Spambase	0.6200	0.5961	0.6686	0.6569	0.5221	0.6331	0.6133	0.6568	**0.6805**

A. DATA SETS

We use totally 14 data sets to evaluate the effectiveness of our method, including texts, images and so on. Data sets from different areas serve as a good test bed for a comprehensive evaluation. The basic information of these data sets are summarized in Table 1.

TABLE 5. Clustering results (Purity) with spectral clustering based methods.

Data	SC-b	SC-a	SC-ew	CoregSC	LMKKM	RMSC	RMKC-SC	ONKC	Ours-SC
BBCSport	0.5955	0.5137	0.5991	0.6159	0.5845	0.6023	0.5849	0.6155	**0.6925**
Coil	0.6556	0.5927	**0.6546**	0.6504	0.6485	0.6078	0.6422	**0.6583**	**0.6552**
Glass	0.5813	0.5238	0.5589	0.5570	5654	0.5402	0.5846	0.5766	**0.7159**
Hitech	0.5394	0.3827	0.5056	0.5141	0.4939	**0.6620**	0.5446	0.4976	0.6454
Iris	0.9000	0.7918	0.8667	0.9240	0.8800	0.9000	0.7687	0.8933	**0.9553**
Lung	0.7818	0.7352	0.7842	0.7798	0.7808	0.7330	0.7507	0.7606	**0.8842**
Tr31	0.7446	0.6029	0.6004	0.6276	0.6029	0.6385	0.6443	0.7101	**0.7296**
UMIST	0.5191	0.4689	0.4849	0.4819	0.4883	0.4304	0.4666	0.4499	**0.5103**
WEBACE	0.6541	0.4736	0.5889	**0.6031**	**0.6041**	0.4975	0.5900	**0.6022**	**0.6047**
Texas	0.7189	0.6992	0.6921	0.6935	0.7688	0.6892	0.74530	0.7674	**0.8322**
Cornell	0.7561	0.7137	0.7140	0.7314	0.7123	0.7025	0.7541	0.7236	**0.8489**
Washington	0.8089	0.7821	0.7779	0.7312	0.8038	0.7779	0.8111	0.7779	**0.8643**
Wisconsin	0.7478	0.7408	0.7432	0.7417	0.7450	0.7388	0.7480	0.7411	**0.7847**
Spambase	0.6200	0.6086	0.6686	0.6592	0.6060	0.6331	0.6133	0.6568	**0.6805**

B. COMPARED METHODS

To evaluate the quality of the learned consensus kernel, we apply two famous kernel based clustering methods (kernel k-means and spectral clustering) on the learned kernels, and we refer them as Ours-KKM and Ours-SC, respectively. We compare them with the following kernel k-means based and spectral clustering based methods, respectively.

- **Single kernel methods**. We run kernel k-means and spectral clustering on each kernel separately. We report the best and the average results over all kernels, which are referred to as KKM-b, KKM-a, SC-b, and SC-a, respectively. "KKM" refers to kernel k-means, "SC" refers to spectral clustering, "b" refers to the best result, and "a" refers to the average result.
- **Equal weighted methods**. We linearly combine all input kernels into a single kernel with equal weights, and then apply kernel k-means and spectral clustering on it. The results are referred to as KKM-ew and SC-ew.
- **MKKM**. MKKM (Multiple Kernel K-Means) is proposed in [5] which extends kernel k-means in multiple kernel setting.
- **CoregSC**. CoregSC is a co-regularized multi-view spectral clustering proposed by [9].
- **LMKKM**. LMKKM is proposed in [10]. Depending on the results in [1], it transfers the kernel k-means to spectral methods and obtain the clustering results from eigenvalue decomposition. Thus it is a spectral based method in fact.
- **RMSC**. RMSC (Robust Multi-view Spectral Clustering) is proposed in [11]. We first transform the kernels into probabilistic transition matrices following [11], and then apply RMSC to get the final clustering results.
- **RMKKM**. RMKKM (Robust multiple kernel k-means) is proposed in [12], which uses $\ell_{2,1}$ loss to replace the ℓ_2 loss in the multiple kernel k-means.
- **RMKC**. RMKC (Robust Multiple Kernel Clustering) is proposed in [6]. It also learns a consensus kernel matrix. We apply kernel k-means and spectral clustering on it, which are referred to as RMKC-KKM and RMKC-SC, respectively.
- **ONKC** ONKC (Optimal Neighborhood Kernel Clustering) is proposed in [8] which learns the optimal neighborhood for clustering.

C. EXPERIMENTAL SETUP

Following the similar experimental protocol of [6], we apply 8 different kernel functions as candidate kernels. These kernels are 5 Gaussian kernels $K(\mathbf{x}_i, \mathbf{x}_j) = exp(-\|\mathbf{x}_i - \mathbf{x}_j)\|^2 / (2t^2)$ with $t = t_0 \times d_{max}$, where d_{max} is the maximal distance between samples and t_0 varies in the range of $\{0.01, 0.1, 1, 10, 100\}$, 2 polynomial kernels $K(\mathbf{x}_i, \mathbf{x}_j) = (\mathbf{x}_i^T \mathbf{x}_j)^a$ with $a = 2, 4$ and a linear kernel. All kernels are normalized to normalized-cut weighted form as [33] did and rescaled to [0, 1].

The number of clusters is set to the true number of classes for all algorithms and all data sets. We independently repeat the experiments for 10 times and report the average results and t-test results. In our method, we tune λ_1 and λ_2 in $[10^{-2}, 10^2]$ by grid search. For other compared methods, we tune the parameters as suggested in their papers.

Two clustering evaluation metrics are adopted to measure the clustering performance, including clustering Accuracy (ACC) and Purity.

All experiments are conducted using Matlab on a PC computer with Windows 10, 4.2GHz CPU and 32GB memory.

D. EXPERIMENTAL RESULTS

We conduct two groups of experiments: we compare our methods with kernel k-means based methods and spectral clustering methods respectively. Table 2 and Table 3 show the ACC and Purity results of kernel k-means based methods, and Table 4 and Table 5 show the ACC and Purity results of spectral clustering based methods. Bold font indicates that the difference is statically significant (the p-value of t-test is smaller than 0.05). Note that since we aim to compare with other multiple kernel methods, we do not calculate the p-value of KKM-b, KKM-a, SC-b and SC-a.

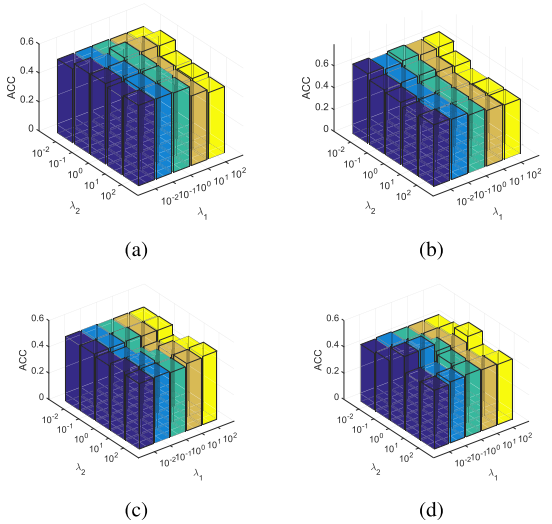


FIGURE 3. ACC of Ours-KKM and Ours-SC w.r.t λ_1, λ_2 on COIL and Tr31 data sets. (a) Kernel k-means results on COIL. (b) Spectral clustering results on COIL. (c) Kernel k-means results on Tr31. (d) Spectral clustering results on Tr31.

From Table 2–5, we find that the performance of kernel k-means and spectral clustering is largely determined by the choice of kernel functions. The performance is significantly deteriorated on inappropriate kernels. Such observations also motivate the development of robust multiple kernel learning.

Our proposed methods outperform other methods significantly on most data sets. Although RMSC also has a mechanism for handling noises, it is developed for transition matrices integration. Different from RMSC, our method is designed for multiple kernel integration. RMSC only extracts the sparse noises in transition matrix. However, kernel noises are more sophisticated. Compared with RMSC, we handle noises more finely, which leads to a better result. RMKKM and RMKC only focus on the local noises, i.e., the noises on the instances, whereas ignore the global noises which is introduced by kernel functions. In our method, we consider both kinds of noises and can recover clean kernels for multiple kernel learning. As a result, our method outperforms them. Moreover, the performance of our method is often close to or even better than the result of the best single kernel. Note that, our method does not need perform exhaustive search on a predefined pool of kernels. Such results well demonstrate the superiority of the proposed method.

E. PARAMETER STUDY

We explore the effect of the parameters on clustering performance. There are two parameters in our method: λ_1 and λ_2. We tune these two parameters from $[10^{-2}, 10^{2}]$. We show the ACC of kernel k-means and spectral clustering on COIL and Tr31 data sets and the results are similar on other data sets. Figure 3 shows the results, from which we see that the performance of our method is stable across a wide range of the parameters.

VI. CONCLUSION

In this paper, we proposed a novel unsupervised robust multiple kernel learning method. By observing that there are two kinds of noises on the kernel matrix, i.e., local noises and global noises, we analyzed the structure of each kind of noises and designed regularized terms to characterize them. Simultaneously, we learned a consensus kernel by minimizing the disagreement over cleaned kernels. Then, a block coordinate descent algorithm was proposed to solve the optimization problem. Experimental results on benchmark data sets demonstrate that our method outperform the state-of-the-art unsupervised multiple kernel learning methods.

Our method involves some complex tensor operation which makes it not suitable for big data. In the future, we will study the scalability issue and use it to handle big data.

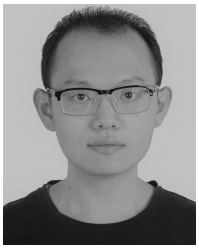
ACKNOWLEDGEMENTS

The authors would like to thank the anonymous reviewers for their helpful comments and suggestions.

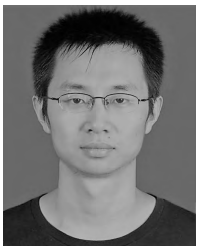
REFERENCES

[1] I. S. Dhillon, Y. Guan, and B. Kulis, "Kernel k-means: Spectral clustering and normalized cuts," in *Proc. 10th ACM SIGKDD Int. Conf. Knowl. Discovery Data Mining*, 2004, pp. 551–556.
[2] S. Yu *et al.*, "Optimized data fusion for kernel k-means clustering," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 34, no. 5, pp. 1031–1039, May 2012.
[3] J. Zhuang, J. Wang, S. C. H. Hoi, and X. Lan, "Unsupervised multiple kernel learning," in *Proc. ACML*, 2011, pp. 129–144.
[4] G. Tzortzis and A. Likas, "Kernel-based weighted multi-view clustering," in *Proc. ICDM*, Dec. 2012, pp. 675–684.
[5] H. C. Huang, Y. Y. Chuang, and C. S. Chen, "Multiple kernel fuzzy clustering," *IEEE Trans. Fuzzy Syst.*, vol. 20, no. 1, pp. 120–134, Feb. 2012.
[6] P. Zhou, L. Du, L. Shi, H. Wang, and Y.-D. Shen, "Recovery of corrupted multiple kernels for clustering," in *Proc. IJCAI*, 2015, pp. 4105–4111.
[7] X. Liu, Y. Dou, J. Yin, L. Wang, and E. Zhu, "Multiple kernel k-means clustering with matrix-induced regularization," in *Proc. AAAI*, 2016, pp. 1888–1894.
[8] X. Liu *et al.*, "Optimal neighborhood kernel clustering with multiple kernels," in *Proc. AAAI*, 2017, pp. 2266–2272.
[9] A. Kumar, P. Rai, and H. Daume, "Co-regularized multi-view spectral clustering," in *Proc. Adv. Neural Inf. Process. Syst.*, 2011, pp. 1413–1421.
[10] M. Gönen and A. A. Margolin, "Localized data fusion for kernel k-means clustering with application to cancer biology," in *Proc. Adv. Neural Inf. Process. Syst.*, 2014, pp. 1305–1313.
[11] R. Xia, Y. Pan, L. Du, and J. Yin, "Robust multi-view spectral clustering via low-rank and sparse decomposition," in *Proc. AAAI*, 2014, pp. 2149–2155.
[12] L. Du *et al.*, "Robust multiple kernel k-means using L21-norm," in *Proc. 24th Int. Joint Conf. Artif. Intell.*, 2015, pp. 1–7.
[13] Z. Xu, R. Jin, I. King, and M. Lyu, "An extended level method for efficient multiple kernel learning," in *Proc. Adv. Neural Inf. Process. Syst.*, 2009, pp. 1825–1832.
[14] H.-C. Huang, Y.-Y. Chuang, and C.-S. Chen, "Affinity aggregation for spectral clustering," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Jun. 2012, pp. 773–780.
[15] Z. Xu, R. Jin, H. Yang, I. King, and M. R. Lyu, "Simple and efficient multiple kernel learning by group lasso," in *Proc. 27th Int. Conf. Mach. Learn. (ICML)*, 2010, pp. 1175–1182.
[16] X. Liu, L. Wang, J. Yin, E. Zhu, and J. Zhang, "An efficient approach to integrating radius information into multiple kernel learning," *IEEE Trans. Cybern.*, vol. 43, no. 2, pp. 557–569, Apr. 2013.
[17] S. C. H. Hoi, R. Jin, P. Zhao, and T. Yang, "Online multiple kernel classification," *Mach. Learn.*, vol. 90, no. 2, pp. 289–316, 2013.
[18] H. Xia, S. C. H. Hoi, R. Jin, and P. Zhao, "Online multiple kernel similarity learning for visual search," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 36, no. 3, pp. 536–549, Mar. 2014.

[19] X. Liu, L. Wang, G.-B. Huang, J. Zhang, and J. Yin, "Multiple kernel extreme learning machine," *Neurocomputing*, vol. 149, pp. 253–264, Feb. 2015.
[20] Y.-P. Zhao, L. Chen, M. Gan, and C. L. P. Chen, "Multiple kernel fuzzy clustering with unsupervised random forests kernel and matrix-induced regularization," *IEEE Access*, vol. 7, pp. 3967–3979, 2018.
[21] Z. Kang, X. Lu, J. Yi, and Z. Xu, "Self-weighted multiple kernel learning for graph-based clustering and semi-supervised classification," in *Proc. IJCAI*, 2018, pp. 2312–2318.
[22] X. Liu *et al.*, "Multiple kernel k-means with incomplete kernels," in *Proc. AAAI*, 2017, pp. 2259–2265.
[23] X. Liu *et al.*, "Late fusion incomplete multi-view clustering," *IEEE Trans. Pattern Anal. Mach. Intell.*, to be published.
[24] X. Liu, X. Zhu, M. Li, C. Tang, E. Zhu, and J. Yin, "Efficient and effective incomplete multi-view clustering," in *Proc. AAAI*, 2019, pp. 1–8.
[25] X. Zhu *et al.*, "Localized incomplete multiple kernel k-means," in *Proc. IJCAI*, Jul. 2018, pp. 3271–3277.
[26] A. Kumar and H. Daumé, "A co-training approach for multi-view spectral clustering," in *Proc. ICML*, 2011, pp. 393–400.
[27] L. R. Tucker, "Some mathematical notes on three-mode factor analysis," *Psychometrika*, vol. 31, no. 3, pp. 279–311, 1966.
[28] Q. Xie *et al.*, "Multispectral images denoising by intrinsic tensor sparsity regularization," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, Jun. 2016, pp. 1692–1700.
[29] Q. Xie, Q. Zhao, D. Meng, and Z. Xu, "Kronecker-basis-representation based tensor sparsity and its applications to tensor recovery," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 40, no. 8, pp. 1888–1892, Aug. 2018.
[30] S. Boyd, N. Parikh, E. Chu, B. Peleato, and J. Eckstein, "Distributed optimization and statistical learning via the alternating direction method of multipliers," *Found. Trends Mach. Learn.*, vol. 3, no. 1, pp. 1–122, Jan. 2011.
[31] N. Halko, P. G. Martinsson, and J. A. Tropp, "Finding structure with randomness: Probabilistic algorithms for constructing approximate matrix decompositions," *SIAM Rev.*, vol. 53, no. 2, pp. 217–288, 2011.
[32] Y. Yang, H. T. Shen, Z. Ma, Z. Huang, and X. Zhou, "L2, 1-norm regularized discriminative feature selection for unsupervised learning," in *Proc. Int. Joint Conf. Artif. Intell. (IJCAI)*, 2011, vol. 22, no. 1, p. 1589.
[33] W. Xu and Y. Gong, "Document clustering by concept factorization," in *Proc. 27th Annu. Int. ACM SIGIR Conf. Res. Develop. Inf. Retr.*, 2004, pp. 202–209.



PENG ZHOU received the B.E. degree in computer science and technology from the University of Science and Technology of China, in 2011, and the Ph.D. degree in computer science from the Institute of Software, University of Chinese Academy of Sciences, in 2017. He is currently a Lecturer with Anhui University. His research interests include machine learning, data mining, and artificial intelligence.



FAN YE received the B.E. degree in information security from the University of Science and Technology of China, in 2005. In 2010, he graduated from the Computer Science Department, School of Computer Science and Technology, University of Science and Technology of China, and the City University of Hong Kong. From 2010 to 2014, he worked at the Research Center for Complex Systems, University of Science and Technology of China, and the Shanghai Institute of Advanced Study, Chinese Academy of Sciences. Since 2014, he has been with the School of Computer Science and Technology, Anhui University. His research interests include machine learning, social network analysis, and recommended systems.

LIANG DU received the B.E. degree in software engineering from Wuhan University, in 2007, and the Ph.D. degree in computer science from the Institute of Software, University of Chinese Academy of Sciences, in 2013. He is currently a Lecturer with Shanxi University. Prior to that, he was a Software Engineer with Alibaba Group, from 2013 and 2014. His research interests include machine learning, data mining, and big data analysis.