Information Sciences 500 (2019) 48–66

Contents lists available at ScienceDirect

Information Sciences

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

Joint correntropy metric weighting and block diagonal regularizer for robust multiple kernel subspace clustering

Chao Yang[a], Zhenwen Ren[a,b,*], Quansen Sun[b], Mingna Wu[a], Maowei Yin[a], Yuan Sun[a]

[a] School of National Defence Science and Technology, Southwest University of Science and Technology, Mianyang 621010, China
[b] Department of Computer Science, Nanjing University of Science and Technology, Nanjing 210094, China

ARTICLE INFO

Article history:
Received 9 January 2019
Revised 22 May 2019
Accepted 26 May 2019
Available online 27 May 2019

Keywords:
Subspace clustering
Multiple kernel learning
Self-expressiveness
Block diagonal regularizer
Correntropy

ABSTRACT

Nonlinear kernel-based subspace clustering methods that can reveal the multi-cluster nonlinear structure of samples are an emerging research topic. However, the existing kernel subspace clustering methods have the following three flaws: 1) their clustering performance is largely determined by the chosen kernel function; 2) they may lack robustness in the presence of non-Gaussian noise and impulsive noise; and 3) their learned affinity matrix can not hold the desired block diagonal property for clustering purpose, which possibly leads to incorrect clustering when using spectral clustering. In this paper, we propose a Joint Robust Multiple Kernel Subspace Clustering (JMKSC) method for data clustering, which has two primary innovations. First, our multiple kernel weighting strategy introduces the correntropy metric weighting instead of a fixed, or inappropriately assigned weighting, which is more robust to the non-Gaussian noise and contributes to learning the optimal consensus kernel. Second, our method encourages acquiring an affinity matrix with the optimal block diagonal property based on the block diagonal regularizer (BDR) and the self-expressiveness property. Experiments on several different types of datasets confirm that the proposed JMKSC significantly outperforms several state-of-the-art single kernel and multiple kernel subspace clustering methods in terms of accuracy, NMI and purity.

1. Introduction

Subspace clustering plays a crucial role in data mining and machine learning [5,47]. Given that data points are approximately drawn from a union of low-dimensional linear (or affine) subspaces, the aim of subspace clustering is to gather each point into their respective subspaces. Subspace clustering has been widely applied to many practical tasks, such as data classification [48], motion segmentation [21], face clustering [26,35], document clustering [34], heterogeneous data analysis [2], subspace learning [30], hybrid system identification in control [11], and community clustering in social networks [3], to name a few.

During recent years, various methods have been attempted for subspace clustering. According to the mechanisms for representing the subspaces, most existing methods mainly fall into five main categories: matrix factorization methods [1,29],

* Corresponding author.
E-mail address: rzw@njust.edu.cn (Z. Ren).

https://doi.org/10.1016/j.ins.2019.05.063
0020-0255/

Gaussian mixture methods [43], algebraic methods [36], spectral-type methods [6,9,44], and deep learning-based methods [33]. Matrix factorization-based methods reveal the cluster or segmentation information of the input data by applying matrix factorization, which is highly sensitive to noise and outliers. The Gaussian mixture methods assume that all the data points are independent samples drawn from a known mixture of Gaussian distributions; then, the expectation maximization (EM) algorithm is employed for subspace clustering. However, these methods are sensitive to outliers and the initialization state. The algebraic-based methods use mathematical tricks to process data points by, for example, fitting the data points with a quadratic polynomial or a high order polynomial. However, in general, these methods are difficult to solve due to their high cost, especially for high-dimensional data, and they are also sensitive to noise and outliers. Recently, there has been a surge of interest in spectral clustering-based methods due to their simple principles and outstanding performance; in general, these methods consist of two main steps: 1) reveal the local or global information of the input data points to seek a desired block diagonal affinity matrix, wherein each entry measures the similarities between pairs of data points; and 2) apply the spectral clustering algorithm to cluster the input data points into their respective underlying subspaces. In Step 1, there are different ways to construct the desired affinity matrix [10,39], but the self-expressiveness is the most commonly used framework to seek for the affinity matrix to do so. However, most of these methods are linear (or affine) self-expressiveness methods that can handle only linear (or affine) subspaces. Unfortunately, in real-life applications, the data points may not fit exactly into a linear subspace model, i.e., the data points always lie on nonlinear subspaces. Finally, although there are numerous recent studies that focus on clustering based on deep learning [17], these research topics are not the focus of this paper.

To overcome the drawback that linear subspace clustering methods can not handle nonlinear data, kernelization techniques are usually introduced to map the possibly nonlinear raw data to a high-dimensional (or even infinite dimensional) linear feature space by using the "kernel trick" technique, where linear subspace clustering can subsequently be performed. There are several state-of-the-art single kernel subspace clustering methods, such as kernel k-means (KKM) [32], spectral clustering (SC) [27], robust kernel k-means (RKKM) [8], and simplex sparse representation (SSR) [16]. However, it is known that the performance of the single kernel method is largely determined by the choice of kernel function. There are various ways to tackle the practical problem of how to select the most suitable kernel for a particular dataset. Instead of a single fixed kernel, many researchers further extend their models to incorporate multiple kernel learning (MKL) [41] abilities. The MKL model is a flexible learning model that has great potential to integrate complementary information. Recent developments in MKL have shown that the construction of a consensual kernel from a number of basis kernels has wide flexibility, its performance is normally better than that of a single kernel. Accordingly, the key to MKL is designing a kernel weighting strategy to promote the full use of the complementary information, especially for noise case in real-life applications with noise. In fact, weighting strategies have been widely used in machine learning and computer vision, such as [38], which use the Gaussian function to weight neighboring pixels for image segmentation, [42] also uses a simple Gaussian function to boost the discrimination power of different sets for image set classification, and [18] uses a Euclidean distance weighting strategy for multiple kernel learning, etc. For subspace clustering, in the past few years, some MKL-based subspace clustering methods have been proposed, such as multiple kernel k-means (MKKM) [15], robust multiple kernel k-means (RMKKM) [8], affinity aggregation for spectral clustering (AASC) [14], self-weighted multiple kernel learning (SMKL) [18], spectral clustering with multiple kernels (SCMK) [19], sparse kernel learning graph-based clustering (LKGs) [20], and low-rank kernel learning graph-based clustering (LKGr) [20]. Amongst them, MKKM extends k-means into a multiple kernel setting. RMKKM is an extended version of MKKM that simultaneously finds the cluster membership, the optimal combination of multiple kernels, and the best clustering label. Due to the similarity between the affinity matrix and the kernel matrix, AASC replaces the single affinity matrix in spectral clustering with multiple matrices; thus, it can be considered as a multiple kernel clustering version of spectral clustering. SCMK learns an optimal kernel using the same way adopted by RMKKM. In specific, using the fact that the optimal kernel is a linear combination of prespecified kernels. LKGr and LKGs learn a low-rank kernel matrix and a sparse kernel matrix, respectively. The kernel matrix can exploit the similarity characteristics of the kernel matrix and seeks an optimal kernel from a neighborhood of candidate kernels. SMKL is a novel MKL framework for subspace clustering and semisupervised classification that introduces a more flexible kernel learning strategy to enhance the representativeness of the learned optimal consensus kernel and to assigns weights for each base kernel. It is worth noting that SMKL is a recently proposed MKL method that uses a kernel weighting strategy based on the Euclidean distance metric. In summary, these methods have good performance, but they are highly sensitive to impulsive noise and non-Gaussian noise and the learned consensus kernels are inadequate for clustering.

Another key issue that arises for subspace clustering methods based on self-expressiveness frameworks is encouraging the learned affinity matrix to obey a desired block diagonal structure for spectral clustering. In the research literatures [9,10,22,39], various norm regularization terms for the self-expression coefficient matrix have been used to learn a block diagonal solution, such as l_1 norm, l_2 norm, and nuclear norm. The details of these regularization terms will be presented in Section 2.1. Unfortunately, there are two disadvantages to these regularization methods: they can not control the targeted number of blocks, and another is that the learned affinity matrix may not be an optimal block diagonal or each block may not be fully connected due to the noise in the data. To alleviate these disadvantages, the first diagonal regularizer, the block diagonal regularizer (BDR), which encourages an affinity matrix to be a k-block diagonal structure, was proposed in [24]. However, as far as we know, BDR has not been introduced into a multiple kernel subspace clustering scenario.

Therefore, this paper considers these problems: 1) in many real-life applications, data points usually lie on nonlinear subspaces rather than linear ones; 2) an exhaustive search of the most suitable kernel from a prespecified pool of base

kernels is very time consuming and expensive. Moreover, a simple consensual kernel weighting strategy limits the power of multiple kernels; 3) the data in high-dimensional space may be contaminated by non-Gaussian noise and impulsive noise such as large outliers, corruption, and missing data; and 4) the block diagonal property of the affinity matrix is fragile whenever the subspaces are dependent and/or the signal-to-noise ratio (S/N) is small.

To address these problems, this paper first introduces the MKL framework into our method, which can handle nonlinear data and avoids incompatibilities when only one predefined kernel is applied. Second, the key of MKL is to carefully design a kernel weighting strategy to improve the performance of multiple kernel clustering. We propose a correntropy [23] based kernel weighting strategy that is robust to non-Gaussian noise and has not additional parameters. Third, as a method based on spectral clustering that introduces kernelization techniques and a self-expressiveness learning framework, we employ the self-expression coefficient matrix to reproduce the kernel Hilbert feature space as the affinity matrix for spectral clustering. Fourth, to encourage the affinity matrix from kernel self-expression learning framework to obey a desired block diagonal structure favored for clustering purpose, we further consider BDR [24] to improve the quality of the affinity matrix. By comprehensively considering these insights, we propose a robust joint multiple kernel subspace clustering method (JMKSC) method by joining BDR and correntropy-based multiple kernel weighting strategy. Experimental evaluations on 9 datasets demonstrate both the effectiveness and robustness of our proposed JMKSC method. In summary, the main contributions of this work can be summarized as follows:

- To better handle data with nonlinear structure, we propose a robust multiple kernel method, i.e., JMKSC, for subspace clustering. JMKSC automatically learns the most suitable consensus kernel from a pool of predefined candidate kernels to solve the difficult problems of defining a suitable single kernel and tuning the kernel parameters, which results in more reliable clustering results than existing single kernel and multiple kernel methods.
- To excavate the true clustering membership of the input samples for further improving clustering performance, BDR [24] is introduced into our model so that the data points in the resulting reproducing kernel Hilbert feature space is self-expressive and the resulting affinity matrix has an optimal block diagonal structure. Different from previous regularization terms (i.e., l_1 norm, l_2 norm, and nuclear norm), the learned affinity matrix of JMKSC can hold an optimal class-wise block diagonal structure and make dense inner-cluster connections.
- To suppress the non-Gaussian noise and and impulsive noise in data from real-world examples, different form existing MKL methods, we first propose to learn a consensus kernel from 12 base kernels by applying the correntropy metric weighted MKL method. In doing so, the learned weighting coefficients vary in the range [0, 1]; for irrelevant kernels, the coefficients are close to 0, while for significant ones, the coefficients are close to 1. But, the coefficients of other weighted strategy are sensitive to noise, which may lead to partiality towards a certain kernel.

The remainder of this paper is organized as follows. We introduce the basic notations of this paper and briefly review of related works and their applications in Section 2. Then, we describe our method in Section 3 and discuss its optimization algorithm. The experiments and analyses are conducted in Section 4. Section 5 concludes this paper with a discussion of future research.

2. Related works

In this section, we give a brief review of some related works of this paper, including subspace self-expressiveness, kernel trick, block diagonal regularizer, correntropy, and their applications.

2.1. Subspace self-expressiveness

Modern robust subspace clustering algorithms take advantage of the so-called self-expressiveness property of linear subspaces, i.e., each data point can be well represented by a linear combination of other points in the union of subspaces. Recent self-expressiveness-based methods rely on building an affinity matrix such that data points from the same subspace have high affinity values and those from different subspaces have low affinity values. In general, the basic model of the self-expression based graph learning framework is the following:

$$\min_{\boldsymbol{Z}} \frac{1}{2}\|\boldsymbol{X}-\boldsymbol{X}\boldsymbol{Z}\|_F^2+\alpha\mathcal{R}(\boldsymbol{Z}) \quad s.t. \;\; \boldsymbol{Z}\geqslant 0,\; diag(\boldsymbol{Z})=0 \tag{1}$$

where $\mathcal{R}(\boldsymbol{Z})$ is a regularization term about $\boldsymbol{Z}$, $\alpha>0$ defines the trade-off between the loss function term and regularization term, and the self-expression coefficient matrix $\boldsymbol{Z}$ is often assumed to be nonnegative and $\boldsymbol{Z}_{ii}=0$.

The self-expressiveness property, $\boldsymbol{X}-\boldsymbol{X}\boldsymbol{Z}$, has been studied in many methods for subspace clustering and image segmentation. The main difference of these methods depends on the choice of the regularizer term $\mathcal{R}(\boldsymbol{Z})$, such as low-rank ($\|\cdot\|_*$) constraint [22], sparsity ($\|\cdot\|_1$) constraint [9,10], and low-rank plus sparsity constraint [39]. It is proven to be effective in improving the clustering performance by enforcing sparsity and low-rank constraints in some applications. Ideally, the affinity matrix $\boldsymbol{Z}$ obeys the optimal block diagonal property. However, in the noisy cases where data is contaminated, the block diagonal property of $\boldsymbol{Z}$ can not guarantee the correct clustering, even $\boldsymbol{Z}$ no longer maintains the block diagonal property, since each block may not be fully connected or the required independent subspaces assumptions usually may not satisfy. Hence, in this paper, we introduce a new block diagonal regularizer instead of using sparsity and low-rank constraints directly.

2.2. Kernel trick

As we know, linear models in Euclidean space can be limited by their inability to exploit the nonlinear relation between data points. By introducing kernelization techniques, data points can be mapped to higher-dimensional Hilbert feature spaces via a suitable kernel function. Then, the nonlinear pattern analysis in the input data space can be transformed into linear analysis in the feature space. Instead of explicitly computing the coordinates in the high-dimensional feature space, common practice in kernel methods consists of using the "kernel trick" [28].

The detailed description of "kernel trick" is as follows: Suppose that the non-linear feature mapping $\phi(\boldsymbol{X}) : \mathcal{R}^d \to \mathcal{H}$ maps the data points $\boldsymbol{X}$ from the input space $\mathcal{R}$ to a reproducing kernel Hilbert space (RKHS) $\mathcal{H}$. It is not necessary to know the explicit representation of transformation ϕ, we usually need to apply the "kernel trick" to obtain a kernel Gram matrix $\boldsymbol{K} = \phi(\boldsymbol{X})^T\phi(\boldsymbol{X})$. It greatly simplifies the computations in $\mathcal{H}$ when the similarities between data points are pre-computed [45]. In doing so, the kernel similarity between two examples $\boldsymbol{x}_i$ and $\boldsymbol{x}_j$ is defined using the pre-defined kernel function as $k(\boldsymbol{x}_i, \boldsymbol{x}_j)$, hence, we have $\boldsymbol{K}_{ij} =< \phi(\boldsymbol{x}_i), \phi(\boldsymbol{x}_j) >_{\mathcal{H}} = k(\boldsymbol{x}_i, \boldsymbol{x}_j)$. The key restriction is that $< \cdot, \cdot >_{\mathcal{H}}$ must be a proper inner product, as long as $\mathcal{H}$ is an inner product space. The most widely used kernel function is the radial basis function (RBF) kernel, which is defined as $k(\boldsymbol{x}_i, \boldsymbol{x}_j) = exp(-\|\boldsymbol{x}_i - \boldsymbol{x}_j\|^2/(2\sigma^2))$, where σ is a free length scale parameter to control the smoothness degree of the kernel. The dimensionality of the feature space $\mathcal{H}$ given by RBF kernel is infinite. In recent years, kernelization techniques are widely used for subspace clustering, and gained good results [28,41].

2.3. Block diagonal regularizer

Ideally, the desired affinity matrix always obeys the block diagonal property such that a nonzero coefficient is assigned to the points from the same subspace and a zero coefficient to these ones from different subspaces. However, in real-life noisy applications, the block diagonal property of affinity matrix is usually violated due to the data contaminated by polytype noise or corruptions. A recently research [24] reveals that affinity matrix with a block diagonal property would possibly lead to correct subspace clustering. To ensure the k-block diagonal property of the affinity matrix $\boldsymbol{A}$, its corresponding graph Laplacian matrix $\boldsymbol{L}$ should represent k connected components.

Since $\boldsymbol{A} \in \mathbb{R}^{N\times N}$ is an affinity matrix, it is required to be nonnegative and symmetric, i.e., $\boldsymbol{A} \geq 0$ and $\boldsymbol{A} = \boldsymbol{A}^T$. Its corresponding Laplacian matrix $\boldsymbol{L}$ is defined as

$$\boldsymbol{L} = \boldsymbol{D} - \boldsymbol{A} \tag{2}$$

where $\boldsymbol{D} = \boldsymbol{A}\boldsymbol{1}$, i.e., $\boldsymbol{D}_i = \sum_j \boldsymbol{A}_{ij}$. The number of connected components of $\boldsymbol{A}$ is related to the spectral property of $\boldsymbol{L}$. Let $\lambda_i(\boldsymbol{L})$, $s.t. i \in [1, \cdots, N]$ be the ith smallest eigenvalue of $\boldsymbol{L}$ in decreasing order. So, obviously, we have $\boldsymbol{L} \succeq 0$ and thus $\lambda_i(\boldsymbol{L}) \geq 0$ for all i. Then, $\boldsymbol{A}$ has k connected components iff

$$\begin{cases} \lambda_i(\boldsymbol{L}) = 0, & i = N - k + 1, \ldots, N, \\ \lambda_i(\boldsymbol{L}) > 0, & i = 1, \ldots, N - k. \end{cases} \tag{3}$$

Inspired by such a block diagonal property, Lu et al. [24] propose the k-block diagonal regularizer (BDR), which is defined as the sum of the k smallest eigenvalues of $\boldsymbol{L}$, i.e.,

$$\|\boldsymbol{A}\|_{\boxed{\mathrm{k}}} = \sum_{i=N-k+1}^{N} \alpha_i(\boldsymbol{L}) \tag{4}$$

It can be seen that $\|\boldsymbol{A}\|_{\boxed{\mathrm{k}}} = 0$ is equivalent to the fact that the affinity matrix $\boldsymbol{A}$ obeys the optimal k-block diagonal property, and vice versa. The BDR not only can control the number of desired blocks (which is usually set to the true number of classes), but also directly encourages the affinity matrix to be block diagonal, which is an important affecting factor for subspace clustering [24]. A disadvantage is that the BDR is non-convex, but a convex program can be used to approximate it.

2.4. Correntropy

Correntropy was proposed in the field of information theoretic learning; it maximizes rather than maximizes the well-known mean square error (MSE) to improve robustness with respect to non-Gaussian noise [23] and impulsive noise [31]. In [23], given two arbitrary vectors $\boldsymbol{x}$ and $\boldsymbol{y}$, the correntropy of them is given by

$$V(\boldsymbol{x}, \boldsymbol{y}) = E[k_\sigma(\boldsymbol{x}, \boldsymbol{y})], \tag{5}$$

where $E[\cdot]$ denotes the mathematical expectation, and $k_\sigma(\cdot, \cdot)$ is a kernel function that satisfies the Mercer theory [37]. In addition, to the best of our knowledge, correntropy has a clear theoretic foundation, and it is symmetric, positive, and bounded. Based on Eq. (5), a general similarity measurement is extended, and it is suitable for any two discrete vectors; it is called as the Correntropy Induced Metric (CIM) [23], and it is defined by

$$CIM(\boldsymbol{x}, \boldsymbol{y}) = (1 - V(\boldsymbol{x}, \boldsymbol{y}))^{\frac{1}{2}} = (1 - \frac{1}{N}\sum_{i=1}^{N} k_\sigma(x_i, y_i))^{\frac{1}{2}}. \tag{6}$$

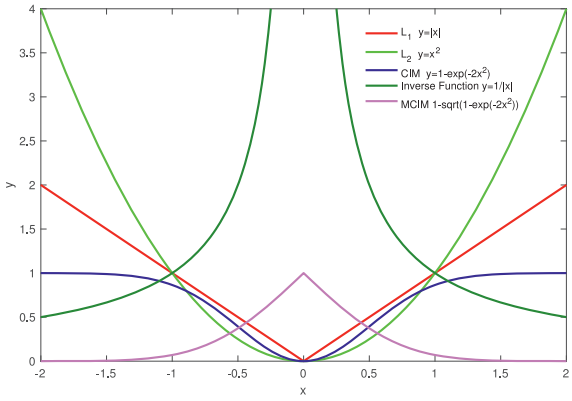


Fig. 1. Comparison of different metric.

Usually, we define the kernel function as a Gaussian kernel $k_\sigma(\boldsymbol{x}_1, \boldsymbol{x}_2) = \exp(-\frac{\|\boldsymbol{x}_1-\boldsymbol{x}_2\|_2^2}{2\sigma^2})$, where parameter σ denotes the kernel size that controls the robustness of the estimator. Another major merit of correntropy is that the kernel size controls its properties. Hence, in this paper, we extend the vector-based correntropy metric CIM for similarity measurements between any kernel matrix pair to weight multiple kernels and provide a strategy for choosing an appropriate kernel size.

For actual learning problems, we often need to solve an optimal minimization problem rather than a maximization problem. Hence, existing methods use CIM instead of correntropy because maximizing correntropy (i.e., the maximum correntropy criterion (MCC) [23]) is equivalent to the minimizing CIM. Fig. 1 shows a comparison of the absolute error (l_1 norm), MSE (l_2 norm) and CIM. The differences between MSE and CIM can be summarized as follows: 1)the MSE is a global metric, but the CIM is a local metric; and 2) MSE increases quadratically for large errors, which is very sensitive to noise, and CIM is close to the absolute error when the errors are relatively small; however, the value of CIM is close to 1 when there are large errors, which has the potential to control noise. Note that large errors are usually caused by polytype noise (such as outliers, badly damaged data, data contaminated by detection errors, and data contaminated by images from other classes), but their negative effects on CIM are limited. Hence, CIM is more robust to non-Gaussian noise and impulsive noise [25] than MSE. The effect of CIM has been widely recognized in the literatures [7,12,13,25,45]. For example, CIM has been used for subspace clustering [13,25], multiview subspace clustering [45], feature learning [12], multi-label learning [7] and face recognition [13].

3. Proposed method

In this section, we present a brand new joint multiple kernel subspace clustering method (JMKSC) method and its optimization strategy for robust subspace clustering.

3.1. Main notations

In the following description, bold uppercase letters denote matrices, bold lowercase letters denote vectors, bold symbols with the subscripts denotes the entry of the matrix or vector, and nonbold symbols denote scalars. Details regarding important notations used throughout the paper are listed in Table 1.

3.2. Joint robust multiple kernel subspace clustering

To handle nonlinear structure of data, we map the nonlinear data into a high-dimensional reproducing kernel Hilbert space (RKHS) where a linear pattern analysis can be performed. Based on the aforementioned self-expressiveness based graph learning framework, we can learn an affinity matrix for spectral clustering. By comprehensively considering both self-expressiveness and kernel mapping, the kernel self-expressiveness optimization problem is formulated as:

$$\min_{\boldsymbol{Z}} \frac{1}{2}\|\phi(\boldsymbol{X}) - \phi(\boldsymbol{X})\boldsymbol{Z}\|_F^2 + \alpha\mathcal{R}(\boldsymbol{Z}) \quad s.t. \quad \boldsymbol{Z} \geqslant 0, \ diag(\boldsymbol{Z}) = 0. \tag{7}$$

where $\mathcal{R}(\boldsymbol{Z})$ is the regularization term on the kernel self-expression coefficient matrix $\boldsymbol{Z}$ and α is a nonnegative trade-off parameter.

Based on the kernel trick, in kernel space, we do not explicitly depend on the unknown kernel mapping function $\phi(\cdot)$, but on the kernel Gram matrix $\boldsymbol{H} = \phi(\boldsymbol{X})^T\phi(\boldsymbol{X})$, where the (i, j)th element of $\boldsymbol{H}$ is $\boldsymbol{H}_{ij} =< \phi(x_i), \phi(x_j) >$. Hence, problem (7) can be formulated as:

$$\|\phi(\boldsymbol{X}) - \phi(\boldsymbol{X})\boldsymbol{Z}\|_F^2 = Trace(\phi(\boldsymbol{X})^T\phi(\boldsymbol{X}) - 2\phi(\boldsymbol{X})^T\phi(\boldsymbol{X})\boldsymbol{Z} + \boldsymbol{Z}^T\phi(\boldsymbol{X})^T\phi(\boldsymbol{X})\boldsymbol{Z}) = Trace(\boldsymbol{H} - 2\boldsymbol{H}\boldsymbol{Z} + \boldsymbol{Z}^T\boldsymbol{H}\boldsymbol{Z}) \tag{8}$$

Table 1
The notations used in this paper.

Notation	Definition
$\boldsymbol{I}$	The identity matrix
$\boldsymbol{1}$	The all-one column vector
N	The number of data points
k	The number of clusters
r	The number of kernels
d	The dimension of data points
$\boldsymbol{X} \in \mathbb{R}^{d\times N}$	The data matrix
$\boldsymbol{Z} \in \mathbb{R}^{N\times N}$	The representation coefficient matrix
$\boldsymbol{A} \in \mathbb{R}^{N\times N}$	The affinity matrix
$\boldsymbol{L} \in \mathbb{R}^{N\times N}$	The laplacian matrix
$\phi(\boldsymbol{X})$	The non-linear feature mapping of data matrix $\boldsymbol{X}$
$\boldsymbol{H}^i \in \mathbb{R}^{N\times N}$	The ith kernel Gram matrix
$\boldsymbol{H} \in \mathbb{R}^{N\times N}$	The consensus kernel matrix
$\|\cdot\|_F$	The frobenius norm
$Trace(\cdot)$	The trace operator of a matrix
$diag(\boldsymbol{A})$	The vector of diagonal elements of matrix $\boldsymbol{A}$
$Diag(\boldsymbol{a})$	The diagonal matrix with its ith element on the diagonal being $\boldsymbol{a}_i$

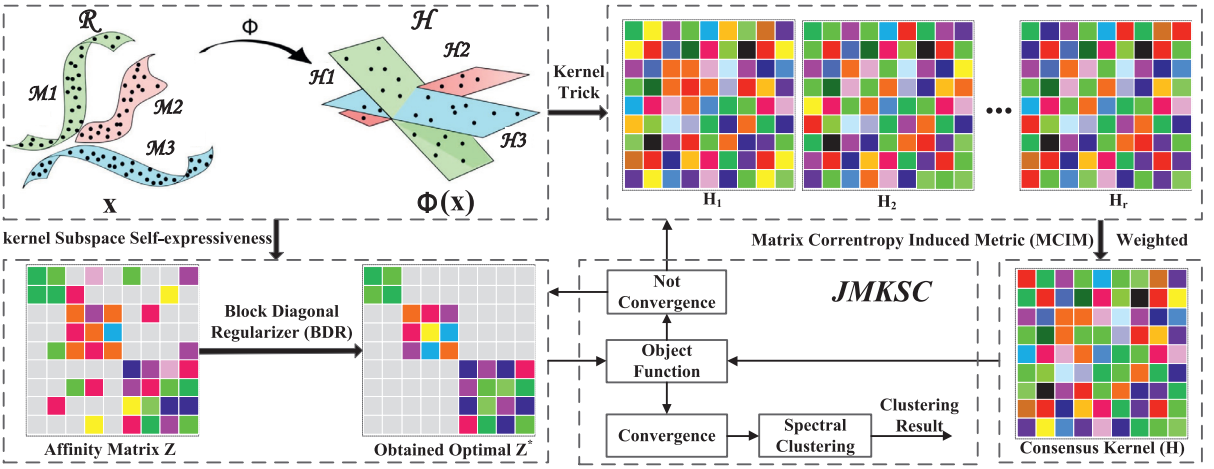


Fig. 2. Conceptual overview of the proposed joint multiple kernel subspace clustering method (JMKSC).

Ideally, we hope to achieve a graph with exactly k connected components if the input data matrix $\boldsymbol{X}$ contains k clusters or classes. In other words, the affinity matrix induced by the kernel self-expression coefficient matrix $\boldsymbol{Z}$ has k block diagonals with proper permutations. Motivated by Lu et al. [24], we introduce the block diagonal regularizer (BDR) to encourage $\boldsymbol{Z}$ to hold the block diagonal property when the underlying subspaces are independent. Furthermore, it should be noted that a main part of this model's input is kernel matrix $\boldsymbol{H}$. It is generally recognized that its performance is largely determined by the choice of kernel, e.g., linear kernel, polynomial kernel, or radial basis function kernel (RBF). Unfortunately, for a particular task based on the kernel method, we do not know which kernel is the most suitable in advance. However, multiple kernel learning (MKL) is a very powerful model for addressing this problem. Hence, in this work, we design a novel correntropy metric induced multiple kernel learning strategy (CMMKL) based on the following three intuitive assumptions: 1) each base kernel $\boldsymbol{H}^i$ is a small perturbation of the desired consensus kernel $\boldsymbol{H}$, i.e., formulated as $\min ||\boldsymbol{H}^i - \boldsymbol{H}||_F^2$; 2) the base kernel $\boldsymbol{H}^i$, which is close to the desired consensus kernel $\boldsymbol{H}$ should be assigned a relatively large weight and vice versa, which will need to be gradually adjusted during the learning process, and 3) the weight of each kernel should be robust to noise to avoid the weight of each kernel being too large or too small when noise is present. Under these assumptions, we directly consider the following multiple kernel learning strategy with a graph BDR form for handling data with noise; we have

$$\min_{\boldsymbol{Z},\boldsymbol{H},\boldsymbol{w}} \frac{1}{2} Trace(\boldsymbol{H} - 2\boldsymbol{H}\boldsymbol{Z} + \boldsymbol{Z}^T\boldsymbol{H}\boldsymbol{Z}) + \alpha \|\boldsymbol{Z}\|_{\boxed{\text{k}}} + \beta \sum_{h=1}^{r} \boldsymbol{w}_i ||\boldsymbol{H}^i - \boldsymbol{H}||_F^2 \quad s.t. \quad \boldsymbol{Z} \geqslant 0, \ \ diag(\boldsymbol{Z}) = 0, \boldsymbol{Z} = \boldsymbol{Z}^T \tag{9}$$

where α and β are the nonnegative trade-off parameters for BDR and CMMKL respectively, and $\boldsymbol{w}_i$ is the weight of the ith predefined base kernel $\boldsymbol{H}^i$. From the above analysis, Fig. 2 presents a conceptual overview of the proposed JMKSC method.

In problem (9), to satisfy the necessary properties of BDR, a simple restriction on the kernel self-expression coefficient matrix $\boldsymbol{Z}$ is that it is symmetric and nonnegative, i.e., $\boldsymbol{Z} \geqslant 0, \ \ diag(\boldsymbol{Z}) = 0, \boldsymbol{Z} = \boldsymbol{Z}^T$. In addition, we note that $\boldsymbol{Z}$ involves two

terms, i.e., $Trace(\cdot)$ and $\|\cdot\|_{\boxed{k}}$. Hence, there still existed some limitations, mainly regarding two aspects of Z: 1) the restrictions on $\boldsymbol{Z}$ will limit its representation capability, and 2) the two terms lead to the difficulty when solving $\boldsymbol{Z}$. To address these limitations, we introduce an auxiliary matrix $\boldsymbol{A}$ and a regularization term $\|\boldsymbol{Z}-\boldsymbol{A}\|_F^2$ to split the variables and to solve subproblems independently, our optimization problem (9) translates to

$$\begin{aligned}\min_{\boldsymbol{Z},\boldsymbol{A},\boldsymbol{H},\boldsymbol{w}} \frac{1}{2}Trace(\boldsymbol{H}-2\boldsymbol{H}\boldsymbol{Z}+\boldsymbol{Z}^T\boldsymbol{H}\boldsymbol{Z})+\alpha\|\boldsymbol{A}\|_{\boxed{k}}+\beta\sum_{h=1}^{r}\boldsymbol{w}_i||\boldsymbol{H}^i-\boldsymbol{H}||_F^2+\frac{\gamma}{2}\|\boldsymbol{Z}-\boldsymbol{A}\|_F^2 \\ s.t.\ \ \boldsymbol{A}\geqslant\boldsymbol{0},\ \ diag(\boldsymbol{A})=\boldsymbol{0},\boldsymbol{A}=\boldsymbol{A}^T\end{aligned} \tag{10}$$

In doing so, the auxiliary matrix $\boldsymbol{A}$ and term $\|\boldsymbol{Z}-\boldsymbol{A}\|_F^2$ make the subproblems with respect to $\boldsymbol{Z}$ and $\boldsymbol{A}$ strongly convex so that the solutions of all involved variables are unique and stable.

3.3. Optimization

In this section, we show an algorithm of efficiently solving problem (10) efficiently. With the alternating minimizing strategy, we separately update the variables of $\boldsymbol{Z}$, $\boldsymbol{A}$, $\boldsymbol{H}$ and $\boldsymbol{w}$ in (10) while fixing the other variables.

For the sake of analysis, we define an object function $\mathcal{L}$ as follows:

$$\mathcal{L}=\frac{1}{2}Trace(\boldsymbol{H}-2\boldsymbol{H}\boldsymbol{Z}+\boldsymbol{Z}^T\boldsymbol{H}\boldsymbol{Z})+\alpha\|\boldsymbol{A}\|_{\boxed{k}}+\beta\sum_{h=1}^{r}\boldsymbol{w}_i||\boldsymbol{H}^i-\boldsymbol{H}||_F^2+\frac{\gamma}{2}\|\boldsymbol{Z}-\boldsymbol{A}\|_F^2 \tag{11}$$

Both convex optimization subproblems and closed-form solutions for each involved variable will be presented.

(1) **Update Z**

Fix $\boldsymbol{A}=\boldsymbol{A}^t,\boldsymbol{H}=\boldsymbol{H}^t,\boldsymbol{w}=\boldsymbol{w}^t$, and the subproblem corresponding to $\boldsymbol{Z}^{t+1}$ is:

$$\min_{\boldsymbol{Z}}\frac{1}{2}Trace(-2\boldsymbol{H}\boldsymbol{Z}+\boldsymbol{Z}^T\boldsymbol{H}\boldsymbol{Z})+\frac{\gamma}{2}\|\boldsymbol{Z}-\boldsymbol{A}\|_F^2 \tag{12}$$

The closed-form solution can be obtained by setting $\frac{\partial\mathcal{L}(\boldsymbol{Z})}{\partial\boldsymbol{Z}}=0$; we have

$$\boldsymbol{Z}^{t+1}=(\gamma\boldsymbol{I}+\frac{1}{2}\boldsymbol{H}+\frac{1}{2}\boldsymbol{H}^T)^{-1}(\boldsymbol{H}^T+\gamma\boldsymbol{A})=(\gamma\boldsymbol{I}+\boldsymbol{H})\backslash(\boldsymbol{H}+\gamma\boldsymbol{A}) \tag{13}$$

(2) **Update A:**

Fix $\boldsymbol{H}=\boldsymbol{H}^t,\boldsymbol{Z}=\boldsymbol{Z}^t,\boldsymbol{w}=\boldsymbol{w}^t$, and the updating step w.r.t $\boldsymbol{A}^{t+1}$ is as follow:

$$\min_{\boldsymbol{A}}\alpha\|\boldsymbol{A}\|_{\boxed{k}}+\frac{\gamma}{2}\|\boldsymbol{Z}-\boldsymbol{A}\|_F^2\ \ s.t.\ \ \boldsymbol{A}\geqslant\boldsymbol{0},\ \ diag(\boldsymbol{A})=\boldsymbol{0},\boldsymbol{A}=\boldsymbol{A}^T \tag{14}$$

Since $\boldsymbol{A}\geq\boldsymbol{0}$ and $\boldsymbol{A}=\boldsymbol{A}^T$, then the Laplacian matrix of $\boldsymbol{A}$ is given by $\boldsymbol{L}=\boldsymbol{D}-\boldsymbol{A}$, where $\boldsymbol{D}$ is the diagonal matrix with diagonal elements $\{\boldsymbol{D}_{ii}=\sum_j^N\boldsymbol{A}_{ij}\}_{i=1}^N$. From the definition of the k-block diagonal regularizer, we have

$$\|\boldsymbol{A}\|_{\boxed{k}}=\sum_{i=N-k+1}^{N}\alpha_i(\boldsymbol{L}) \tag{15}$$

According to the Ky Fan Theorem [4], Eq. (15) can be rewritten as:

$$\sum_{i=N-k+1}^{N}\alpha_i(\boldsymbol{L})=\inf_{\boldsymbol{U}\in\mathcal{R}^{N\times k},\boldsymbol{U}^T\boldsymbol{U}=\boldsymbol{I}}Trace(\boldsymbol{U}\boldsymbol{U}^T\boldsymbol{L})=\min_{\boldsymbol{S}}Trace(\boldsymbol{S}\boldsymbol{L})\ \ s.t.\ \ \boldsymbol{0}\preceq\boldsymbol{S}\preceq\boldsymbol{I},Trace(\boldsymbol{S})=k \tag{16}$$

Consequently, updating $\boldsymbol{A}$ involves solving the following problem:

$$\min_{\boldsymbol{A}}\alpha Trace(\boldsymbol{S}(\boldsymbol{D}-\boldsymbol{A}))+\frac{\gamma}{2}\|\boldsymbol{Z}-\boldsymbol{A}\|_F^2\ \ s.t.\ \ \boldsymbol{A}\geqslant\boldsymbol{0},\ \ diag(\boldsymbol{A})=\boldsymbol{0},\boldsymbol{A}=\boldsymbol{A}^T,\boldsymbol{0}\preceq\boldsymbol{S}\preceq\boldsymbol{I},Trace(\boldsymbol{S})=k \tag{17}$$

Problem (17) involves variables $\boldsymbol{S}$ and $\boldsymbol{A}$, and we need to update these two variables separately. When the result of the t-th iteration is obtained and denoted by $(\boldsymbol{S}^t,\boldsymbol{A}^t)$, the result of the $(t+1)$th iteration $(\boldsymbol{S}^{t+1},\boldsymbol{A}^{t+1})$ can be calculated as follows:

a. Fix $\boldsymbol{A}=\boldsymbol{A}^t$, and update $\boldsymbol{S}^{t+1}$ by

$$\min_{\boldsymbol{A}}\alpha Trace(\boldsymbol{S}(\boldsymbol{D}-\boldsymbol{A}))\ \ s.t.\ \ \boldsymbol{0}\preceq\boldsymbol{S}\preceq\boldsymbol{I},Trace(\boldsymbol{S})=k \tag{18}$$

Problem (18) looks very difficult to solve. Fortunately, by using Eq. (1605) of page 611 in [4], given ordered diagonalization $\boldsymbol{D}-\boldsymbol{A}=\boldsymbol{Q}\Lambda\boldsymbol{Q}^T$, the optimal $\boldsymbol{U}$ for the infimum is $\boldsymbol{U}^*=\boldsymbol{Q}(:,N-k+1:N)\in\mathbb{R}^{N\times k}$. Hence, subproblem (18) has a closed-form solution given by

$$\boldsymbol{S}^{t+1}=\boldsymbol{U}^*\boldsymbol{U}^{*T} \tag{19}$$

b. Fix $\boldsymbol{S} = \boldsymbol{S}^t$, and update $\boldsymbol{A}^{t+1}$ by

$$\boldsymbol{A}^{t+1} = \arg\min_{\boldsymbol{A}} \alpha Trace(\boldsymbol{LS}) + \frac{\gamma}{2}\|\boldsymbol{Z} - \boldsymbol{A}\|_F^2 \quad s.t. \quad \boldsymbol{A} \geqslant \boldsymbol{0}, diag(\boldsymbol{A}) = \boldsymbol{0}, \boldsymbol{A} = \boldsymbol{A}^T \tag{20}$$

Then, problem (20) can be converted into the following formulation:

$$\boldsymbol{A}^{t+1} = \arg\min_{\boldsymbol{A}} \frac{1}{2}\|\boldsymbol{A} - (\boldsymbol{Z} - \frac{\alpha}{\gamma}(diag(\boldsymbol{S})\boldsymbol{1}^T - \boldsymbol{S}))\|_F^2 \quad s.t. \quad \boldsymbol{A} \geqslant \boldsymbol{0}, diag(\boldsymbol{A}) = \boldsymbol{0}, \boldsymbol{A} = \boldsymbol{A}^T \tag{21}$$

We provide the detailed derivation of problem (21) in Appendix A. Then, let $M = \boldsymbol{Z} - \frac{\alpha}{\gamma}(diag(\boldsymbol{S})\boldsymbol{1}^T - \boldsymbol{S})$ and $\hat{\boldsymbol{M}} = \boldsymbol{M} - Diag(diag(\boldsymbol{M}))$; the closed-form solution of $\boldsymbol{A}$ is obtained by

$$\boldsymbol{A}^{t+1} = max\left(0, \frac{\hat{\boldsymbol{M}} + \hat{\boldsymbol{M}}^T}{2}\right). \tag{22}$$

We also provide derivation details of the closed-form solution of $\boldsymbol{A}$ in Appendix B.

(3) **Update $\boldsymbol{H}$:**

Based on the prior information that each input base kernel is a perturbation of the desired consensus kernel $\boldsymbol{H}$, we need to update the consensus kernel so that its position can be gradually adjusted. Thus, we fix $\boldsymbol{A} = \boldsymbol{A}^t, \boldsymbol{Z} = \boldsymbol{Z}^t, \boldsymbol{w} = \boldsymbol{w}^t$ and update $\boldsymbol{H}^{t+1}$ by

$$\min_{\boldsymbol{H}} \frac{1}{2}Trace(\boldsymbol{H} - 2\boldsymbol{HZ} + \boldsymbol{Z}^T\boldsymbol{HZ}) + \beta\sum_{h=1}^{t}\boldsymbol{w}_i||\boldsymbol{H}^i - \boldsymbol{H}||_F^2 \tag{23}$$

The closed-form solution of $\boldsymbol{H}^{t+1}$ can be obtained by setting $\frac{\partial \mathcal{L}(\boldsymbol{H})}{\partial \boldsymbol{H}} = 0$ and solving for $\boldsymbol{H}$; we have

$$\boldsymbol{H}^{t+1} = \frac{2\boldsymbol{Z}^T - \boldsymbol{ZZ}^T - \boldsymbol{I} + 2\beta\sum_{k=1}^{h}\boldsymbol{w}_i\boldsymbol{H}^i}{2\beta\sum_{k=1}^{h}\boldsymbol{w}_i} \tag{24}$$

(4) **Update w:**

As we know, correntropy (see Section 2.4) is robust to the non-Gaussian noise and impulsive noise. In this paper, we extend the concept of CIM for a general similarity measurement between any two square matrices or kernel matrices. The new robust metric is called the Matrix Correntropy Induced Metric (MCIM), and it is defined as Lemma 1.

Lemma 1. *The matrix Correntropy Induced Metric (MCIM) is a metric used to quantize the similarity between two matrices, and it is more robust to the non-Gaussian noise and impulsive noise than $\|\cdot\|_F^2$. It is formally defined as*

$$MCIM(\boldsymbol{U}, \boldsymbol{V}) = \sqrt{(k_\delta(0) - \frac{1}{N^2}\sum_{i=1}^{N}\sum_{j=1}^{N}k_\delta(\boldsymbol{\Delta}_{ij}))} \tag{25}$$

where $\boldsymbol{\Delta} = \boldsymbol{U} - \boldsymbol{V}$ ($\boldsymbol{U} \in \mathbb{R}^{N\times N}$ and $\boldsymbol{V} \in \mathbb{R}^{N\times N}$ are two arbitrary matrices with the same dimensionality), N is both the number of data points and the dimensionality of the given matrices, δ is the kernel size, and $k_\delta(\boldsymbol{\Delta}_{ij}) = exp(-(\boldsymbol{\Delta}_{ij})^2/2\delta^2)$ is the Gaussian kernel with kernel size δ.

In this work, the kernel size δ is calculated as the average distance bias, i.e.,

$$\delta^2 = \frac{1}{2N^2\|\boldsymbol{\Delta}\|_F^2} \tag{26}$$

The base kernel that is close to the desired consensus kernel should be assigned a large weight, and vice versa. Thus, base on our MCIM, we fix $\boldsymbol{A} = \boldsymbol{A}^t, \boldsymbol{Z} = \boldsymbol{Z}^t, \boldsymbol{H} = \boldsymbol{H}^t$, and update $\boldsymbol{w}^{t+1}$ to

$$\boldsymbol{w}_i^{t+1} = 1 - MCIM(\boldsymbol{H}^i, \boldsymbol{H}). \tag{27}$$

Note that $\boldsymbol{w}_i$ takes a value between 0 and 1. $\boldsymbol{w}_i \approx 0$ if the kernel and consensus kernel are very far apart, and $\boldsymbol{w}_i \approx 1$ if the two kernels are very close.

The pseudocode for solving problem (9) and clustering is shown in Algorithm 1, for which these update steps (13), (19), (22), (19), (25), will be repeated until the convergence condition is reached or the maximum number of iterations *maxIter* is exceeded. At each iteration, we check the following convergence criterion for algorithmic stopping:

$$\begin{gathered} diffZ = \|\boldsymbol{Z}^{t+1} - \boldsymbol{Z}^t\|_F, diffA = \|\boldsymbol{A}^{t+1} - \boldsymbol{A}^t\|_F, diffS = \|\boldsymbol{S}^{t+1} - \boldsymbol{S}^t\|_F, diffH = \|\boldsymbol{H}^{t+1} - \boldsymbol{H}^t\|_F, \\ diffL = \mathcal{L}^{t+1} - \mathcal{L}^t, \quad max(diffZ, diffA, diffH, diffL) \leq \varepsilon \end{gathered} \tag{28}$$

where $\mathcal{L}^t$ is the value of our object function of the tth iteration, and ε is the stop threshold.

After we obtain the auxiliary coefficient matrix $\boldsymbol{A}$, we then construct the affinity matrix simply as $\frac{1}{2}(|\boldsymbol{A}| + |\boldsymbol{A}|^T)$, or with an extra normalization step on $\boldsymbol{A}$ as in SSC [10]. Finally, we employ the spectral clustering algorithm [27] to get the clustering results.

Algorithm 1 Joint Robust Multiple Kernel Subspace Clustering (JMKSC).

Input: Data matrix $\boldsymbol{X}$, r kernel matrixs $\{\boldsymbol{H}^i\}_{i=1}^r$, trade-off parameters α, β, γ, number of cluster/class k.

Initialize: $\boldsymbol{S}^1 = \boldsymbol{0}$, $\boldsymbol{A}^1 = \boldsymbol{0}$, $\boldsymbol{Z}^1 = \boldsymbol{0}$, $\boldsymbol{H}^1 = \frac{1}{r}\sum_{i=1}^r \boldsymbol{H}^i$, $\{\boldsymbol{w}_i^1\}_{i=1}^r = \frac{1}{r}$, stop threshold $\varepsilon = 10^{-4}$, $t = 1$, and $maxIter = 100$.

while convergence criterion (28) is not satisfied, and $t < maxIter$ **do**
  Update $\boldsymbol{Z}^{t+1}$ in closed-form by using the solution (13).
  Update $\boldsymbol{S}^{t+1}$ in closed-form by using the solution (19).
  Update $\boldsymbol{A}^{t+1}$ in closed-form by using the solution (22).
  Update $\boldsymbol{H}^{t+1}$ in closed-form by using the solution (19).
  for $i = 1$ to r **do**
    Update $\boldsymbol{w}_i^{t+1}$ in closed-form by using the solution (25).
  end for
  t = t+1;
  Obtain the optimal $\boldsymbol{Z}^*$, $\boldsymbol{S}^*$, $\boldsymbol{A}^*$, $\boldsymbol{H}^*$, and $\boldsymbol{w}^*$.
  Define the affinity matrix $\boldsymbol{A} = \frac{1}{2}(|\boldsymbol{A}| + |\boldsymbol{A}|^T)$.
  Perform spectral clustering using similarity matrix $\boldsymbol{A}$.
end while

Output: ~~The clustering results: ACC, NMI, Purity.

Table 2
Attributes of the 9 widely used datasets.

Dataset	Type	# instances	# features (*row* × *column*)	# classes
Yale	Face image	165	1024 (32 × 32)	15
Jaffe	Face image	213	676 (26 × 26)	10
ORL	Face image	400	1024 (32 × 32)	40
AR (neutral images)	Face image	840	768 (32 × 24)	120
AR (corrupted images)	Face image	720	768 (32 × 24)	120
COIL20	Object image	1440	1024 (32 × 32)	20
BA	Handwritten digits image	1404	320 (20 × 16)	36
tr11	Text	414	6429	9
tr41	Text	878	7454	10
tr45	Text	690	8261	10

4. Experimental verification and analysis

In this section, to demonstrate the clustering performance and robustness of our proposed JMKSC method, we compare the experimental results on several datasets with those from several state-of-the-art methods for subspace clustering.

4.1. Datasets

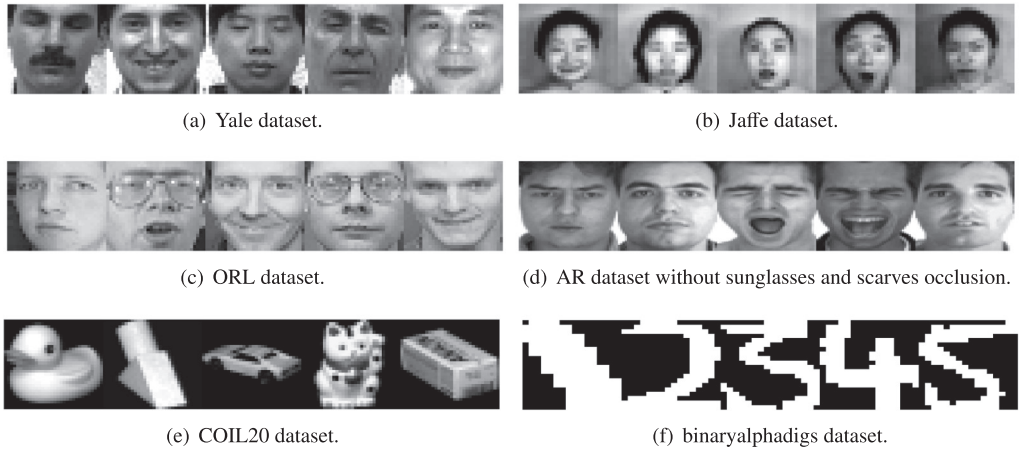
We perform experiments on 9 different public datasets, including 6 image ones (Yale, Jaffe, ORL, AR, COIL20, and Binary Alphadigits) and 3 text corpora ones (tr11, tr41, and tr45). They have been widely used to report the clustering performance and robustness of different methods. The detailed statistics information and data dimensionality of these datasets are summarized in Table 2. The first 6 datasets contain sparse gross corruptions due to the existence of specular reflections, which are non-Lambertian. Non-linearity arises because the face poses and expressions were not exactly the same when images were taken for the same subject/object. We down-sample each image to size *row* × *colume*, and vectorize the image to form an *row***colume*-dimensional column vector. Moreover, it is very necessary to normalize each column vector to an unit norm. The detail of these datasets are given as below.

Yale[1]: The dataset contains 165 grayscale face images from 15 subjects, each subject has 11 images with different facial expression or external environment, including wink, surprised, sleepy, sad, happy, normal, right-light, left-light, center-light, without glasses, with glasses. The face images of different 5 individuals are shown in Fig. 3(a).

JAFFE[2]: The dataset contains 213 face images of 7 facial expressions (1 neutral + 6 basic facial expressions) posed by 10 Japanese female models. Each image has been rated on 6 emotion adjectives by 60 Japanese subjects. Some face images of different individuals are shown in Fig. 3(b).

ORL[3]: The ORL face dataset contains face images from 40 subjects, each has 10 different images. For some subjects, the images were taken at different times, facial expressions, and varying the lighting. Some face images of different subjects are shown in Fig. 3(c).

[1] http://www.cvc.yale.edu/projects/yalefaces/yalefaces.html.
[2] http://www.kasrl.org/jaffe.html.
[3] https://www.cl.cam.ac.uk/research/dtg/attarchive/facedataset.html.



(a) Yale dataset. (b) Jaffe dataset.

(c) ORL dataset. (d) AR dataset without sunglasses and scarves occlusion.

(e) COIL20 dataset. (f) binaryalphadigs dataset.

Fig. 3. Cropped images of the fist five subject/object form six datasets.

AR[4]: AR face dataset contains over 4000 face images corresponding to 126 persons. For each subject, 26 face images are taken in two separate sessions. There are 13 images for each session, in which 3 images with sunglasses, another 3 ones with scarves, and the remaining 7 ones are with different expression and illumination changes. The images with sunglasses and scarves are considered as corrupted images, and the remaining images are considered as neutral images. The images were taken under strictly controlled conditions. The face images of different five individuals are shown in Fig. 3(d).

COIL20[5]: The dataset consists of 20 objects and 72 images for each object. For each object, the images were taken 5 degrees apart as the object is rotating on a turntable. Some kawaii images from five different object classes of COIL20 dataset are shown in Fig. 3(e).

Binary Alphadigits[6]: Binary Alphadigits consists of letters of capita 'A' through 'Z' and digits of '0' through '9'. There are 39 examples for each class. Fig. 3(f) shows some samples.

Text corpora: The text corpora contains TR11, TR41, and TR45, which are derived from TREC-5, TREC-6 and TREC-7 collections[7].

4.2. Baseline algorithm comparison and settings

To demonstrate how clustering performance can be boosted by our method, we compare it with several state-of-the-art single kernel methods and multiple kernel methods.

The single kernel methods include KKM, SC, and RKKM, and their corresponding EW, AVG, and BEST versions, which are summarized as follows:

- **Kernel K-means (KKM)** [32]. The KKM is proposed to cope with the nonlinear structure of many practical datasets, where data points are mapped through a nonlinear transformation into a higher dimensional feature space, in which the data points are linearly separable.
- **Spectral Clustering (SC)** [27]. The SC proposes a particular manner to use the k eigenvectors simultaneously for subspace clustering.
- **Robust Kernel K-means (RKKM)** [8]. The RKKM is an extended version of KKM to deal with data containing noise and outliers, where the squared l_2 norm is replaced by l_{21} norm to restrict the error construction term.
- **KKM-EW, SC-EW, and RKKM-EW**. They are the equally weighted (EW) versions of the four single kernel methods described above (i.e., KKM, SC, and RKKM). For the EW version, the consensus kernel is the combination of multiple input base kernels, and all each base kernels are assigned with equally weights. Since we have 12 base kernels in this paper, the weight of each kernel is 1/12.
- **KKM-AVG, SC-AVG, and RKKM-AVG**. They are the average (AVG) versions of KKM, SC, and RKKM, respectively. For the AVG version, we run each single kernel method on each base kernel separately, and then we compute the average results of the 12 kernels.
- **KKM-BEST, SC-BEST, and RKKM-BEST**. They are the best (BEST) versions of KKM, SC, and RKKM, respectively. The BEST version means that we run each single kernel method on each base kernel separately, and then we select the best one from the results of the 12 kernels.

[4] http://www2.ece.ohio-state.edu/~aleix/ARdataset.html.
[5] http://www.cs.columbia.edu/CAVE/software/softlib/coil-20.php.
[6] https://cs.nyu.edu/~roweis/data/binaryalphadigs.mat.
[7] http://trec.nist.gov.

Table 3
Parameter settings of all experimental methods.

Method	Parameter settings	# Iterations
KKM	$Replicates = \{\mathbf{1}, 5, 10\}$, $clusterMaxIter = 10$	50
SC	τ	100
RKKM	$Replicates = \{\mathbf{1}, 5, 10\}$, $clusterMaxIter = 10$	50
MKKM	$\gamma = \{10^{-1}, 5\times10^{-2}, 10^{-2}, \mathbf{5\times10^{-3}}, 10^{-3}, 5\times10^{-4}, 10^{-4}\}$, $m = 1.08$	50
AASC	$\gamma = 5\times10^{-3}$	20
RMKKM	$\gamma = \{0.1, 0.2, \mathbf{0.3}, 0.4, 0.5, 0.6, 0.7, 0.8, 0.9\}$	50
LKGr	$\alpha = \{10^{-5}, \mathbf{10^{-3}}, 10^{-1}\}$, $\beta = \{\mathbf{1}, 100\}$ and $\gamma = \{10^{-5}, 10^{-3}, \mathbf{10^{-2}}, 10^{-1}, 10^{1}, 10^{3}\}$	50
SCMK	$\alpha = \{10^{-2}, \mathbf{10^{-3}}, 10^{-4}, 10^{-5}\}$, $\beta = \{10^{-3}, \mathbf{10^{-2}}, 10^{-1}\}$, $\gamma = \{10^{-3}, \mathbf{10^{-4}}, 10^{-5}\}$	50
SMKL	$\alpha = \{10^{-3}, 10^{-4}, \mathbf{10^{-5}}, 10^{-6}\}$, $\beta = \{10, \mathbf{15}, 100\}$, $\gamma = \{10^{-1}, 10^{-1}, \mathbf{1}, 10\}$	50
JMKSC	$\alpha = \{10^{-4}, 10^{-3}, 10^{-2}, \mathbf{10^{-1}}, 1, 10\}$, $\beta = \{1, 5, 10, 15, \mathbf{20}, 25, 30\}$, $\gamma = \{10^{-1}, 1, \mathbf{5}, 10, 15, 20, 25, 30\}$	50

* The stop threshold ε for all methods are set to 10^{-5} for all the datasets.

The compared multiple kernel methods include MKKM, AASC, RMKKM, SMKL, SCMK, and LKGr, which are summarized in Section 1.

The parameters from the best clustering results were adopted for the above methods. Regarding the selection process, we either use the parameters suggested in the respective original papers (if the parameters were provided), or we manually tune them and retain those with the best performance. The parameter settings and maximal iterations of all methods are shown in Table 3, in which the recommended parameters are indicated in bold.

4.3. Evaluation metrics

In all experiments, we evaluate the clustering performance of all the methods via 3 popular evaluation metrics, including clustering accuracy (ACC), normalized mutual information (NMI) and Purity [40]. For all these metrics, the higher value indicates the better clustering performance.

Let N be the total number of data points, $\boldsymbol{C} = \{\boldsymbol{C}_1, \cdots, \boldsymbol{C}_t\}$ be the set of clusters reported by a clustering algorithm, $\widetilde{\boldsymbol{C}} = \{\widetilde{\boldsymbol{C}}_1, \cdots, \widetilde{\boldsymbol{C}}_s\}$ be the set of "ground truth" clusters, $|\boldsymbol{C}_i|\,(1 \le i \le t)$ be the number of data points in the ith cluster of the clustering solution, $|\widetilde{\boldsymbol{C}}_j|\ (1 \le j \le s)$ be the number of data points in the jth cluster of the ground truth, and $|\boldsymbol{C}_i \cap \widetilde{\boldsymbol{C}}_j|$ be the number of objects in both the ith cluster of the clustering solution and jth cluster of the ground truth.

ACC discovers the one-to-one relationship between clusters and measures the extent to which each cluster contained data points from the corresponding class. For each data point x_i, let c_i and $\tilde{c}_i$ be the cluster label and ground truth label, respectively. The accuracy is estimated by

$$ACC = \frac{\sum_{i=1}^{N} \delta(\tilde{c}_i, map(c_i))}{N} \tag{29}$$

where $map(c_i)$ is the mapping function that maps each cluster label c_i to the equivalent label from the dataset, and $\delta(x, y)$ is the delta function that equals one if $x = y$ and equals zero otherwise. The best mapping can be found by using the Kuhn–Munkres (KM) algorithm.

NMI is used for determining the quality of clusters, which is estimated by

$$\begin{aligned} NMI(\boldsymbol{C}, \widetilde{\boldsymbol{C}}) &= \frac{MI(\boldsymbol{C}, \widetilde{\boldsymbol{C}})}{\sqrt{H(\boldsymbol{C})H(\widetilde{\boldsymbol{C}})}}, \quad MI(\boldsymbol{C}, \widetilde{\boldsymbol{C}}) = \sum_i \sum_j \frac{|\boldsymbol{C}_i \cap \widetilde{\boldsymbol{C}}_j|}{N} log \frac{N \cdot |\boldsymbol{C}_i \cap \widetilde{\boldsymbol{C}}_j|}{|\boldsymbol{C}_i||\widetilde{\boldsymbol{C}}_j|} \\ H(\boldsymbol{C}) &= \sum_i \frac{|\boldsymbol{C}_i|}{N} log \frac{|\boldsymbol{C}_i|}{N}, \quad H(\widetilde{\boldsymbol{C}}) = \sum_j \frac{|\widetilde{\boldsymbol{C}}_j|}{N} log \frac{|\widetilde{\boldsymbol{C}}_j|}{N}, \end{aligned} \tag{30}$$

where $H(\boldsymbol{C})$ and $H(\widetilde{\boldsymbol{C}})$ are the entropies of $\boldsymbol{C}$ and $\widetilde{\boldsymbol{C}}$, respectively, and $MI(\boldsymbol{C}, \widetilde{\boldsymbol{C}})$ is the mutual information. Note that $MI(\boldsymbol{C}, \widetilde{\boldsymbol{C}})$ takes values between 0 and 1, $NMI = 1$ if the two sets of clusters are identical, and $NMI = 0$ if the two sets are independent.

Purity measures the extent to which each cluster contained data points from primarily one class, which is computed by the weighted sum of individual cluster purity values, i.e.,

$$Purity = \frac{1}{N} \sum_i \max_j |\boldsymbol{C}_i \cap \widetilde{\boldsymbol{C}}_j| \tag{31}$$

4.4. Kernel design

To evaluate the clustering performance of the MKL methods, we built 12 base kernels (i.e., $r = 12$) in this work. The details of these kernels are shown in Table 4. In addition, data normalization was required for some kernels due to their restricted domain; hence, all kernels were rescaled between 0 and 1 by dividing each element by the maximum pairwise squared Euclidean distance (SED).

Table 4
Description of kernel mapping.

Kernel	Expressions	# n	Parameter settings
Gaussian kernel	$k(\boldsymbol{x}_1, \boldsymbol{x}_2) = \exp(-\|\boldsymbol{x}_1 - \boldsymbol{x}_2\|_2^2/(2t\sigma^2))$	7	σ is the maximal distance between $\boldsymbol{x}_1$ and $\boldsymbol{x}_2$, and $t \in \{0.01, 0.05, 0.1, 1, 10, 50, 100\}$
linear kernel	$k(\boldsymbol{x}_1, \boldsymbol{x}_2) = \boldsymbol{x}_1^T\boldsymbol{x}_2$	1	–
polynomial kernel	$\boldsymbol{k}(\boldsymbol{x}_1, \boldsymbol{x}_2) = (\boldsymbol{x}_1^T\boldsymbol{x}_2 + a)^b$	4	$a \in \{0, 1\}$ and $b \in \{2, 4\}$

Table 5
Clustering results measured by ACC/NMI/Purity with standard deviations of the compared single kernel methods.

Data	Metrics	KKM-AVG	KKM-BEST	KKM-EW	SC-AVG	SC-BEST	SC-EW	RKKM-AVG	RKKM-BEST	RKKM-EW
Yale	ACC	0.390(0.031)	0.471(0.040)	0.410(0.028)	0.405(0.026)	0.494(0.042)	0.497(0.023)	0.397(0.034)	0.481(0.041)	0.411(0.028)
	NMI	0.420(0.029)	0.513(0.034)	0.457(0.025)	0.448(0.021)	0.529(0.031)	0.533(0.024)	0.429(0.030)	0.523(0.036)	0.460(0.026)
	Purity	0.411(0.030)	0.492(0.037)	0.435(0.028)	0.431(0.022)	0.516(0.037)	0.515(0.027)	0.417(0.031)	0.498(0.041)	0.436(0.029)
JAFFE	ACC	0.671(0.071)	0.744(0.078)	0.625(0.074)	0.540(0.028)	0.749(0.042)	0.538(0.031)	0.680(0.072)	0.756(0.084)	0.628(0.077)
	NMI	0.715(0.053)	0.801(0.066)	0.696(0.056)	0.594(0.022)	0.821(0.028)	0.591(0.023)	0.740(0.049)	0.835(0.066)	0.702(0.058)
	Purity	0.701(0.061)	0.773(0.069)	0.666(0.061)	0.566(0.025)	0.768(0.037)	0.564(0.028)	0.718(0.055)	0.796(0.069)	0.668(0.063)
ORL	ACC	0.459(0.023)	0.535(0.029)	0.473(0.025)	0.467(0.024)	0.580(0.033)	0.481(0.015)	0.469(0.023)	0.550(0.030)	0.482(0.028)
	NMI	0.634(0.015)	0.734(0.025)	0.676(0.018)	0.667(0.014)	0.752(0.017)	0.694(0.009)	0.639(0.016)	0.742(0.025)	0.685(0.019)
	Purity	0.504(0.022)	0.580(0.027)	0.519(0.023)	0.512(0.021)	0.615(0.029)	0.523(0.015)	0.515(0.022)	0.596(0.027)	0.529(0.025)
AR	ACC	0.309(0.014)	0.330(0.015)	0.319(0.012)	0.222(0.008)	0.288(0.010)	0.212(0.008)	0.312(0.014)	0.334(0.016)	0.318(0.012)
	NMI	0.606(0.007)	0.652(0.008)	0.633(0.008)	0.561(0.007)	0.584(0.010)	0.581(0.006)	0.608(0.008)	0.654(0.008)	0.633(0.008)
	Purity	0.336(0.012)	0.355(0.014)	0.346(0.013)	0.260(0.007)	0.332(0.008)	0.253(0.007)	0.339(0.013)	0.359(0.015)	0.346(0.012)
COIL20	ACC	0.507(0.040)	0.595(0.049)	0.548(0.047)	0.437(0.019)	0.677(0.060)	0.369(0.013)	0.519(0.042)	0.616(0.050)	0.554(0.047)
	NMI	0.636(0.025)	0.741(0.042)	0.707(0.030)	0.543(0.013)	0.810(0.045)	0.465(0.007)	0.637(0.027)	0.746(0.038)	0.710(0.026)
	Purity	0.553(0.032)	0.646(0.039)	0.595(0.041)	0.468(0.017)	0.699(0.055)	0.398(0.011)	0.563(0.035)	0.664(0.040)	0.601(0.039)
BA	ACC	0.337(0.019)	0.412(0.026)	0.364(0.021)	0.263(0.012)	0.311(0.025)	0.290(0.015)	0.344(0.018)	0.422(0.029)	0.370(0.021)
	NMI	0.465(0.011)	0.573(0.017)	0.523(0.013)	0.401(0.011)	0.508(0.022)	0.444(0.007)	0.470(0.011)	0.578(0.018)	0.529(0.015)
	Purity	0.360(0.017)	0.442(0.025)	0.388(0.022)	0.291(0.012)	0.345(0.025)	0.321(0.012)	0.369(0.017)	0.453(0.028)	0.395(0.024)
TR11	ACC	0.447(0.049)	0.519(0.068)	0.438(0.054)	0.433(0.032)	0.510(0.054)	0.465(0.031)	0.450(0.050)	0.530(0.070)	0.438(0.055)
	NMI	0.332(0.037)	0.489(0.056)	0.351(0.035)	0.314(0.018)	0.431(0.029)	0.381(0.020)	0.335(0.041)	0.497(0.062)	0.351(0.036)
	Purity	0.563(0.044)	0.676(0.064)	0.583(0.053)	0.502(0.020)	0.588(0.032)	0.546(0.021)	0.564(0.047)	0.679(0.068)	0.583(0.053)
TR41	ACC	0.463(0.058)	0.556(0.077)	0.476(0.070)	0.448(0.007)	0.635(0.019)	0.472(0.011)	0.468(0.061)	0.568(0.085)	0.478(0.071)
	NMI	0.404(0.050)	0.599(0.073)	0.425(0.062)	0.366(0.007)	0.613(0.019)	0.436(0.013)	0.409(0.052)	0.608(0.070)	0.429(0.061)
	Purity	0.600(0.050)	0.744(0.068)	0.637(0.058)	0.565(0.007)	0.737(0.017)	0.627(0.017)	0.602(0.049)	0.750(0.066)	0.640(0.058)
TR45	ACC	0.456(0.052)	0.588(0.078)	0.451(0.056)	0.460(0.015)	0.574(0.032)	0.533(0.021)	0.457(0.054)	0.581(0.076)	0.453(0.056)
	NMI	0.387(0.046)	0.579(0.062)	0.402(0.057)	0.332(0.011)	0.480(0.018)	0.420(0.022)	0.390(0.045)	0.578(0.060)	0.406(0.056)
	Purity	0.536(0.046)	0.685(0.064)	0.559(0.054)	0.500(0.013)	0.613(0.027)	0.570(0.023)	0.538(0.046)	0.682(0.062)	0.560(0.053)
Avg	ACC	0.449(0.039)	0.528(0.050)	0.456(0.043)	0.408(0.019)	0.535(0.035)	0.429(0.020)	0.455(0.041)	0.538(0.054)	0.459(0.044)
	NMI	0.511(0.030)	0.631(0.043)	0.541(0.034)	0.470(0.014)	0.614(0.024)	0.505(0.015)	0.517(0.031)	0.640(0.042)	0.545(0.034)
	Purity	0.507(0.035)	0.599(0.045)	0.525(0.039)	0.455(0.016)	0.579(0.030)	0.480(0.018)	0.514(0.035)	0.608(0.046)	0.529(0.040)

* For the EW version of single kernel method, the weight of each kernel is set to 1/12 for all the datasets.

4.5. Clustering results and analysis

In this section, multiple kernel clustering experiments are carried out on the 9 datasets mentioned above, i.e., the Yale, Jaffe, ORL, neutral images of AR, BA, COIL20, tr11, tr41, and tr45 datasets. For all the compared clustering methods, we set the number of clusters to the true number of classes k. As suggested in [24], each experiment is independently repeated 20 times, and the average results are reported. It is important to notice that we ran the single kernel methods (i.e., KKM, SC, and RKKM) on each base kernel separately.

The experimental results on all datasets are shown in Table 5 (for single kernel methods) and Table 6 (for multiple kernel methods), which report the average results evaluated by ACC, NMI and purity. For each dataset, there are 3 rows, corresponding to the mean of ACC, NMI and purity of the 20 trials, respectively. It should be noted that the last 3 rows are the mean of ACC, NMI, purity of each method for each separate datasets. According to Tables 5 and 6, it is observed that our JMKSC method basically obtains the highest performance as compare to the other methods in most cases, which reveals the very obvious effectiveness of our method. Specifically, JMKSC achieve improvements of approximately 8.5, 10.8, and 5.7 for ACC, NMI, and purity, respectively, compared with the second-best method, i.e., SCMK. Moreover, JMKSC has an average minimum standard deviation of 1.2, 0.8, and 1.1 for ACC, NMI, purity, respectively.

After an in-depth analysis of the results from Tables 5 and 6, the following observations are noted. (1) In summary, MKL-based methods achieve better results than single kernel methods in all the experiments. It turns out that, MKKM, AASC, RMKKM, SMKL, SCMK, LKGr, and JMKSC perform far better than KKM, SC, and RKKM, which is consistent with our belief that MKL methods can remedy the shortcomings of single kernel methods with poor adaptability to different datasets. The reason is that the MKL methods can exploit complementary information and learn better consensus kernel. (2) The performances of the "-BEST" methods (i.e., KKM-BEST, SC-BEST, RKKM-BEST) are better than those of the "-AVG" methods (i.e., KKM-AVG, SC-AVG, RKKM-AVG), which proves that it is very important to predefine a suitable kernel. In other words,

Table 6
Clustering results measured by ACC/NMI/Purity with standard deviations of the compared multiple kernel methods.

Data	Metrics	MKKM	AASC	RMKKM	SCMK	LKGr	SMKL	JMKSC
Yale	ACC	0.457(0.041)	0.406(0.027)	0.521(0.034)	0.582(0.025)	0.540(0.030)	0.582(0.017)	**0.630(0.006)**
	NMI	0.501(0.036)	0.468(0.028)	0.556(0.025)	0.576(0.012)	0.566(0.025)	0.614(0.015)	**0.631(0.006)**
	Purity	0.475(0.037)	0.423(0.026)	0.536(0.031)	0.610(0.014)	0.554(0.029)	0.667(0.014)	**0.673(0.007)**
JAFFE	ACC	0.746(0.069)	0.304(0.008)	0.871(0.053)	0.869(0.022)	0.861(0.052)	**0.967(0.000)**	**0.967(0.007)**
	NMI	0.798(0.058)	0.272(0.006)	0.893(0.041)	0.868(0.021)	0.869(0.031)	0.951(0.000)	**0.952(0.010)**
	Purity	0.768(0.062)	0.331(0.008)	0.889(0.045)	0.882(0.023)	0.859(0.038)	**0.967(0.000)**	**0.967(0.007)**
ORL	ACC	0.475(0.023)	0.272(0.009)	0.556(0.024)	0.656(0.015)	0.616(0.016)	0.573(0.032)	**0.725(0.014)**
	NMI	0.689(0.016)	0.438(0.007)	0.748(0.018)	0.808(0.008)	0.794(0.008)	0.733(0.027)	**0.852(0.012)**
	Purity	0.514(0.021)	0.316(0.007)	0.602(0.024)	0.699(0.015)	0.658(0.017)	0.648(0.017)	**0.753(0.012)**
AR	ACC	0.286(0.014)	0.332(0.006)	0.344(0.012)	0.544(0.024)	0.314(0.015)	0.263(0.009)	**0.609(0.007)**
	NMI	0.592(0.014)	0.651(0.005)	0.655(0.015)	0.775(0.009)	0.648(0.007)	0.568(0.014)	**0.820(0.002)**
	Purity	0.305(0.012)	0.350(0.006)	0.368(0.010)	0.642(0.014)	0.330(0.014)	0.530(0.014)	**0.656(0.010)**
COIL20	ACC	0.548(0.058)	0.349(0.050)	0.667(0.028)	0.591(0.028)	0.618(0.051)	0.487(0.031)	**0.696(0.016)**
	NMI	0.707(0.033)	0.419(0.027)	0.773(0.017)	0.726(0.011)	0.766(0.023)	0.628(0.018)	**0.818(0.007)**
	Purity	0.590(0.053)	0.391(0.044)	0.699(0.022)	0.635(0.013)	0.650(0.039)	0.683(0.004)	**0.806(0.010)**
BA	ACC	0.405(0.019)	0.271(0.003)	0.434(0.018)	0.384(0.014)	0.444(0.018)	0.246(0.012)	**0.484(0.015)**
	NMI	0.569(0.008)	0.423(0.004)	0.585(0.011)	0.544(0.012)	0.604(0.009)	0.486(0.011)	**0.621(0.007)**
	Purity	0.435(0.014)	0.303(0.004)	0.463(0.015)	0.606(0.009)	0.479(0.017)	**0.623(0.011)**	0.563(0.018)
TR11	ACC	0.501(0.048)	0.472(0.008)	0.577(0.094)	0.549(0.015)	0.607(0.043)	0.729(0.019)	**0.737(0.002)**
	NMI	0.446(0.046)	0.394(0.003)	0.561(0.118)	0.371(0.018)	0.597(0.031)	0.622(0.048)	**0.673(0.002)**
	Purity	0.655(0.044)	0.547(0.000)	0.729(0.096)	0.783(0.011)	0.776(0.030)	**0.879(0.030)**	0.819(0.001)
TR41	ACC	0.561(0.068)	0.459(0.001)	0.627(0.073)	0.650(0.068)	0.595(0.020)	0.671(0.002)	**0.689(0.004)**
	NMI	0.578(0.042)	0.431(0.000)	0.635(0.092)	0.492(0.017)	0.604(0.023)	0.625(0.004)	**0.660(0.003)**
	Purity	0.728(0.042)	0.621(0.001)	0.776(0.065)	0.758(0.034)	0.759(0.031)	0.761(0.003)	**0.799(0.003)**
TR45	ACC	0.585(0.066)	0.526(0.008)	0.640(0.071)	0.634(0.058)	0.663(0.042)	0.671(0.004)	**0.687(0.036)**
	NMI	0.562(0.056)	0.420(0.014)	0.627(0.092)	0.584(0.051)	0.671(0.020)	0.622(0.007)	**0.690(0.022)**
	Purity	0.691(0.058)	0.575(0.011)	0.752(0.074)	0.728(0.048)	0.800(0.026)	0.816(0.004)	**0.822(0.031)**
Avg	ACC	0.507(0.045)	0.377(0.013)	0.582(0.045)	0.607(0.030)	0.584(0.032)	0.577(0.014)	**0.692(0.012)**
	NMI	0.605(0.034)	0.435(0.010)	0.670(0.048)	0.638(0.018)	0.680(0.020)	0.650(0.016)	**0.746(0.008)**
	Purity	0.573(0.038)	0.429(0.012)	0.646(0.042)	0.705(0.020)	0.650(0.027)	0.730(0.011)	**0.762(0.011)**

* The best results are highlighted in bold font.

the performance of a single kernel method is largely dependent on the choice of kernel function. (3) The performances of the "-BEST" methods are better than those of the "-EW" methods (i.e., KKM-EW, SC-EW, RKKM-EW), which proves that, in a multiple kernel setting, designing an appropriate kernel weighting strategy can help boost performance. (4) As seen, KKM-EW, SC-EW and RKKM-EW improve clustering performance considerably more than KKM-AVG, SC-AVG, and RKKM-AVG, respectively, i.e., single kernel methods using an equally weighted kernel (i.e., the "-EW" methods) often give better results than "-AVG" methods, which also proves the importance of kernel weighting strategies. (5) As expected, our JMKSC method can beat SMKL in almost all cases, which demonstrates the effectiveness of our BDR term and the CMMKL weighting strategy for subspace clustering. Using the recently presented SMKL method as a reference, our JMKSC method yields significantly different results. First, the biggest difference is that JMKSC introduces a correntropy-based weighting strategy instead of using a simple Euclidean distance metric. Second, JMKSC encourages the learning of a desired affinity matrix with an optimal k block diagonals structure by introducing BDR. (6) We can infer that the relevant kernels are encouraged by relatively larger weights with an upper bound of 1, while the irrelevant kernels are suppressed by weights closed to 0. (7) Among the MKL methods, our proposed JMKSC method has the smallest standard deviation, which means that JMKSC has good stability.

In summary, experimental results on multiple datasets demonstrate that JMKSC has better clustering performance when compared with some excellent single kernel methods and multiple kernel clustering methods.

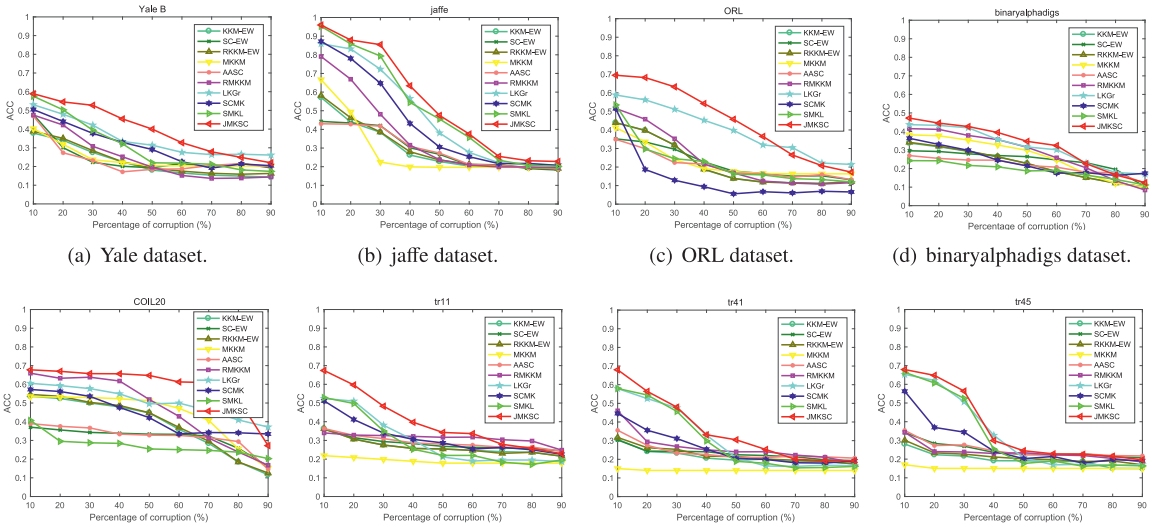
4.6. Robustness experiments

Next, we evaluate the clustering robustness of our method. In robustness experiments, we considered the following two scenarios. (1) Data points are corrupted by different percentages of random pixel corruption. (2) AR face images are corrupted by contiguous occlusion.

For the first scenario, we use the eight datasets mentioned above, i.e., the Yale, Jaffe, ORL, BA, COIL20, tr11, tr41 and tr45 datasets. As in [46] did, a percentage of randomly chosen pixels in all data points were replaced with independently and identically distributed noise, which is uniformly distributed on $[0, x_{\max}]$, where $x_{\max}$ is the largest possible pixel value. The percentage of corrupted pixels in each image varies from 10% to 90%. Fig. 4 shows an image from the Yale dataset with a different percentages of corrupted pixels. To ensure the reliability of the experimental results, we repeated all experiments for 20 times for each method, and the means of ACC and NMI are reported for evaluation, as shown in Figs. 6 and 7, respectively. According to Figs. 6 and 7, for each method, both the ACC and NMI decrease obviously when more pixels of each image are corrupted. More importantly, it can be seen that the proposed JMKSC method outperforms the compared methods with different percentages of corrupted pixels in almost all conditions, mainly because we use the CMMKL weighting

Fig. 4. Corrupted face images from the Yale dataset with the pixel corruption percentage varying from 0% to 90% with a step of 10%, respectively.

Fig. 5. Face images with contiguous occlusion by sunglasses and scarves from the AR dataset.



(a) Yale dataset. (b) jaffe dataset. (c) ORL dataset. (d) binaryalphadigs dataset.

(e) COIL20 dataset. (f) tr11 dataset. (g) tr41 dataset. (h) tr45 dataset.

Fig. 6. Clustering accuracy on eight datasets with different percentages pixel corruption for different algorithms.

strategy to weight multiple kernels so that the effect of the data points with corrupted pixels is limited. Another noteworthy characteristic is that none of the compared methods perform well when the images have a high percentage of pixel corruptions (e.g., $>70\%$), probably due to insufficient discriminative information.

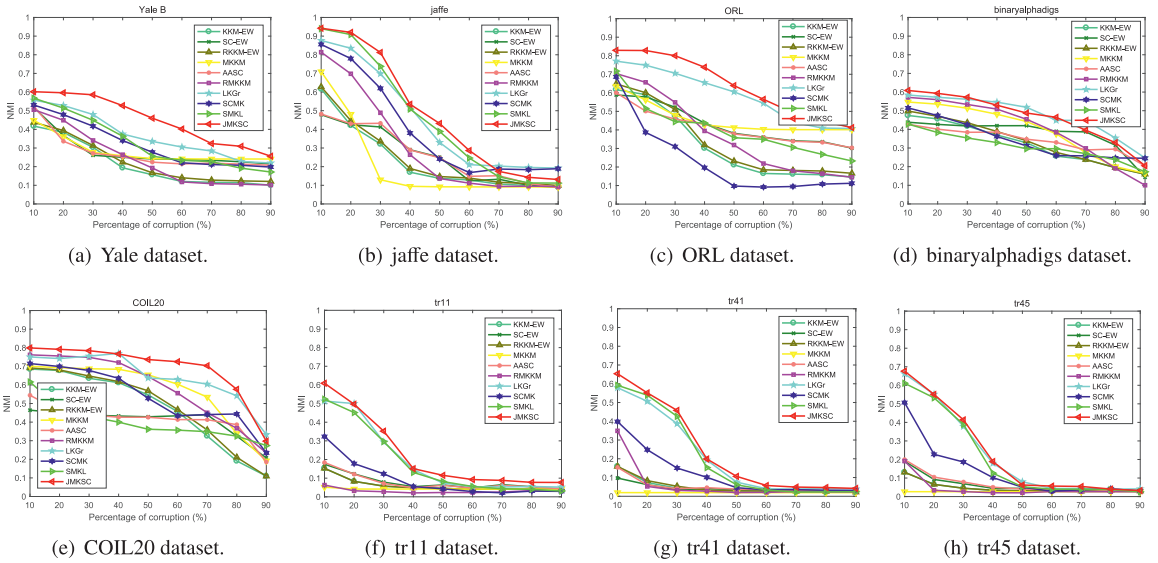
For the second scenario, we evaluated the robustness of JMKSC on the AR dataset with contiguous occlusions by sunglasses and scarves. The statistics of the corrupted AR dataset are summarized in Table 2. See Fig. 5 for sample images with the occlusions. In this scenario, we also use the means of the ACC and NMI indicators to evaluate clustering robustness. The experimental results for all competing methods are shown in Fig. 8, which indicate that the proposed JMKSC method consistently outperforms other competitive methods for this dataset. This result mainly because the face images contaminated by sunglasses and scarves occlusions contain many outliers rows and features, and the influence of noise is suppressed by the CMMKL weighting strategy, hence our JMKSC method is competent for such a challenging task.

From the results of the above two scenarios, our superior performance verifies that, by learning a correntropy metric to suppress the influence of noise, our method can better handle noisy cases better than other methods.

4.7. Timing experiments

First, we present the theoretical analysis of computational complexity for our proposed JMKSC method. The computational complexities of updating $\boldsymbol{Z}$, $\boldsymbol{S}$, $\boldsymbol{A}$, $\boldsymbol{H}$, and $\boldsymbol{w}$ are $O(N^3)$, $O(N^3)$, $O(N)$, $O(N)$, and $O(N)$, respectively. Hence, the total time complexity of JMKSC can be loosely thought of as $O(t(N^3))$, where t is the number of iterations required to reach convergence.

Then, we quantitatively analyze the computational complexity of all compared methods. Our experimental platform is a Mac Mini 2018 with an Intel Core i7 (3.2 GHz) CPU and 16-GB RAM, the operating system is macOS Mojave, and the simulation software is MATLAB 2016b. A comparison of the computational complexity of all methods on the benchmark Yale B dataset is reported in Fig. 9. The results suggest that our method is much faster than other methods, except for MKKM and AASC. However, the ACC and NMI of our proposed JMKSC method far exceeds those of MKKM. Compared to SCMK, it has the second-best recognition rate, but it has less speed. Compared to SCMK, our method is approximately 9 times faster. Hence, our JMKSC method is a fast and efficient MKL clustering method.



(a) Yale dataset. (b) jaffe dataset. (c) ORL dataset. (d) binaryalphadigs dataset.

(e) COIL20 dataset. (f) tr11 dataset. (g) tr41 dataset. (h) tr45 dataset.

Fig. 7. Clustering NMI on eight datasets with different percentages pixel corruption for different algorithms.

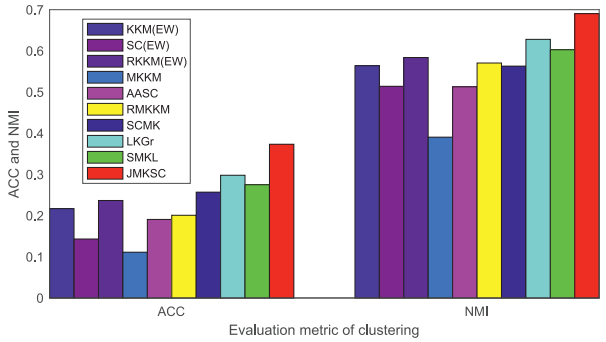


Fig. 8. Clustering accuracy and NMI on the AR dataset with contiguous occlusion by sunglasses and scarves.

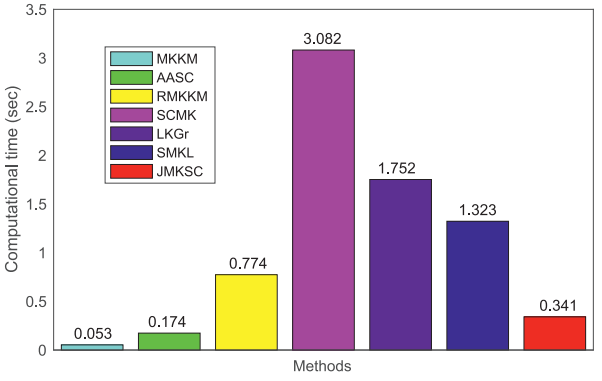
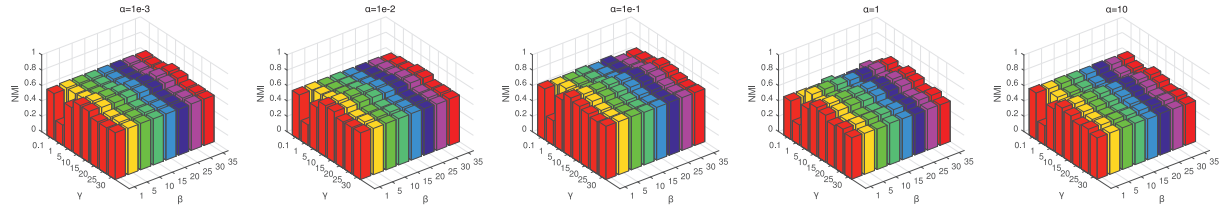


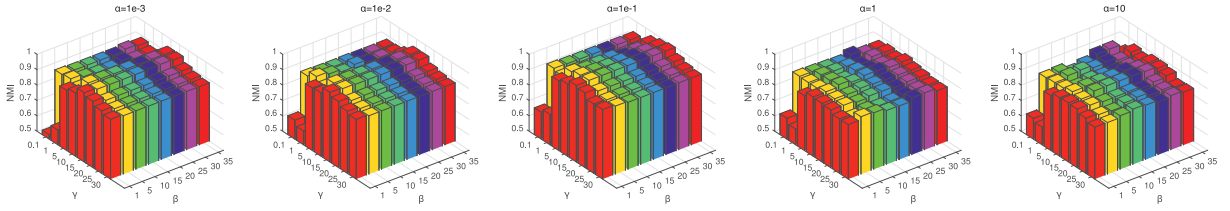
Fig. 9. Computational time (in seconds) of the compared MKL methods on the Yale B dataset.

4.8. Parameter sensitivity and convergence analysis

In the proposed method (JMKSC), three hyperparameters, i.e., α, β and γ, need to be tuned. α controls the significance of the BDR term $\|\cdot\|_{\boxed{k}}$, β is used to balance the term $\sum_{h=1}^{r} \boldsymbol{w}_i ||\boldsymbol{H}^i - \boldsymbol{H}||_F^2$, and γ controls the term $\|\boldsymbol{Z} - \boldsymbol{A}\|_F^2$, which encourages $\boldsymbol{Z}$ to approach $\boldsymbol{A}$. It is obvious that the larger the parameter is, the more important the role the corresponding term plays. Hence, we analyze the sensitivity of the three parameters by using the Yale and Jaffe datasets as examples,

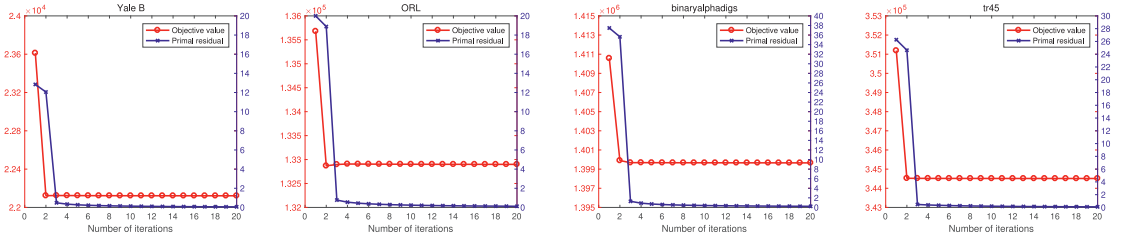


(a) Yale dataset.



(b) Jaffe dataset.

Fig. 10. Clustering NMI metric versus the parameters α, β and γ on the Yale and Jaffe datasets, respectively.



(a) Yale dataset. (b) ORL dataset. (c) binaryalphadigs dataset. (d) TR45 dataset.

Fig. 11. Convergence curves for the objective values and primal residuals on four datasets, (a) Yale, (b) ORL, (c) binaryalphadigs, and (d) tr45.

in terms of NMI. We let α take values of $[10^{-3}, 10^{-2}, 10^{-1}, 1, 10]$, let β take values of 1 to 35 in steps of 5, and let γ take values of [0.1, 1, 5, 10, 15, 20, 25, 30]. Fig. 10 shows the effect of the three parameters evaluated by the NMI metric. It shows that JMKSC performs well for a wide range of β values ($\beta \in [15, 20, 25, 30]$), and γ values ($\gamma \in [5, 10]$) in the experiments on the two datasets. Moreover, $\alpha = 0.1$ generally achieves to good clustering performance. Hence, we fix $\alpha = 0.1$, $\beta = 20$, and $\gamma = 5$ for simplicity in all experiments in this paper.

For convergence analysis, we show the typical objective value curve and the primal residual curve (which is defined as the change of the maximal Frobenius norm between two consecutive iterations, i.e., *max*(*diffZ, diffA, diffH, diffL*)) vs. the number of iterations in Fig. 11 with data sampled from Yale, ORL, BA, and tr45. It can be seen that our method converges fast (within 5 iterations) with the primal residuals quickly reduced close to zero, and then reaches a steady-state with more iterations, indicating that JMKSC converges quickly and efficiently.

5. Conclusions

In this work, we proposed a novel multiple kernel subspace clustering method, i.e., JMKSC, that can efficiently handle nonlinear subspace and suppress impulsive noise and non-Gaussian noise. The objective function of JMKSC elegantly integrates the block diagonal regularizer, the subspace self-expressiveness, the "kernel trick", and the multiple kernel weighting strategy based on the correntropy metric. Extensive clustering experiments have demonstrated that JMKSC significantly outperforms the state-of-the-art single kernel subspace clustering methods with predefined consensual kernel and multiple kernel subspace clustering methods. There are three points remaining for further researching. First, following the idea of kernel weighting strategy, in future work we can research improved multiple kernel clustering methods based on other prior knowledge and explore other robust multiple kernel weighting strategy. Second, we plan to further extend JMKSC to handle a large-scale multi-view subspace clustering problem. Third, in the future, we will try to integrate JMKSC into a deep learning framework to boost the clustering performance.

Declaration of Competing Interest

We confirm that there are no known conflicts of interest associated with this publication and there has been no significant financial support for this work that could have influenced its outcome.

We confirm that the manuscript has been read and approved by all named authors and that there are no other persons who satisfied the criteria for authorship but are not listed. We further confirm that the order of authors listed in the manuscript has been approved by all of us.

We confirm that we have provided a current, correct email address which is accessible by the Corresponding Author and which has been configured to accept email from (rzw@njust.edu.cn)

Acknowledgments

This research was supported by the Department of Science and Technology of Sichuan Province (Grant no. ZYF-2018-106), Sichuan Province Science and Technology Support Program (Grant no. 2018TZDZX0002), State Administration for Science, Technology and Industry for National Defense (Grant nos. JCKY2017209B010, JCKY2018209B001), and the National Natural Science Foundation of China (Grant no. 61673220).

The authors would like to thank Liang Du, who shared their RMKKM code on their github homepage, and to thank Zhao Kang, who shared the SMKL code. The authors would also like to thank the anonymous reviewers who provided substantive suggestions for improving our work.

Appendix A

$$\boldsymbol{A}^* = \arg\min_{\boldsymbol{A}} \alpha Trace(\boldsymbol{LS}) + \frac{\gamma}{2}\|\boldsymbol{Z}-\boldsymbol{A}\|_F^2 \quad s.t. \quad \boldsymbol{A} \geqslant \boldsymbol{0}, diag(\boldsymbol{A}) = \boldsymbol{0}, \boldsymbol{A} = \boldsymbol{A}^T. \tag{32}$$

Sine $\boldsymbol{L} = \boldsymbol{D} - \boldsymbol{A}$ is a Laplacian matrix,[8] we have $\boldsymbol{L}^T = \boldsymbol{L}$ and $\boldsymbol{LS} = \boldsymbol{L}^T\boldsymbol{S}$. Therefore, the problem (32) can be rewritten as:

$$\boldsymbol{A}^* = \arg\min_{\boldsymbol{A}} \alpha < \boldsymbol{D}-\boldsymbol{A}, \boldsymbol{S} > + \frac{\gamma}{2}\|\boldsymbol{Z}-\boldsymbol{A}\|_F^2 \quad s.t. \quad \boldsymbol{A} \geqslant \boldsymbol{0}, diag(\boldsymbol{A}) = \boldsymbol{0}, \boldsymbol{A} = \boldsymbol{A}^T, \tag{33}$$

where $\boldsymbol{D}_{ii} = \sum_j \boldsymbol{A}_{ij}$ is the diagonal degree matrix. Based on $\sum_{i=1}^n \boldsymbol{L}_{ii}\boldsymbol{S}_{ii} = \sum_{i=1}^n \sum_{j=1}^n \boldsymbol{A}_{ij}\boldsymbol{S}_{ii}$, we have $Trace(\boldsymbol{DS}) = Trace(\boldsymbol{A}diag(\boldsymbol{S})\boldsymbol{1}^T)$. Since $\boldsymbol{D}$ and $\boldsymbol{A}$ are symmetric matrixs, we get $Trace(\boldsymbol{D}^T\boldsymbol{S}) = Trace(\boldsymbol{A}^T diag(\boldsymbol{S})\boldsymbol{1}^T)$, i.e., $< \boldsymbol{D}, \boldsymbol{S} > = < \boldsymbol{A}, diag(\boldsymbol{S})\boldsymbol{1}^T >$. The problem (33) can be expanded to

$$\boldsymbol{A}^* = \arg\min_{\boldsymbol{A}} \frac{\alpha}{\gamma}(< \boldsymbol{A}, diag(\boldsymbol{S})\boldsymbol{1}^T > - < \boldsymbol{A}, \boldsymbol{S} >) + \frac{1}{2}\|\boldsymbol{Z}-\boldsymbol{A}\|_F^2 \quad s.t. \quad \boldsymbol{A} \geqslant \boldsymbol{0}, diag(\boldsymbol{A}) = \boldsymbol{0}, \boldsymbol{A} = \boldsymbol{A}^T. \tag{34}$$

In this sub-problem, $\boldsymbol{Z}$ and $\boldsymbol{S}$ are fixed. By introducing two constant $\frac{\alpha}{\gamma} < \boldsymbol{Z}, diag(\boldsymbol{S})\boldsymbol{1}^T >$, $\frac{\alpha}{\gamma} < \boldsymbol{Z}, \boldsymbol{S} >$, The first two items can be changed to four items, i.e.,

$$\frac{\alpha}{\gamma}(< \boldsymbol{A}, diag(\boldsymbol{S})1^T > - < \boldsymbol{A}, \boldsymbol{S} > - < \boldsymbol{Z}, diag(\boldsymbol{S})\boldsymbol{1}^T > + < \boldsymbol{Z}, \boldsymbol{S} >) = \frac{\alpha}{\gamma} < \boldsymbol{A}-\boldsymbol{Z}, (diag(\boldsymbol{S})\boldsymbol{1}^T - \boldsymbol{S}) > \tag{35}$$

Then, the problem (35) can be rewritten as:

$$\arg\min_{\boldsymbol{A}} \frac{\alpha}{\gamma} < \boldsymbol{A}-\boldsymbol{Z}, (diag(\boldsymbol{S})\boldsymbol{1}^T - \boldsymbol{S}) > + \frac{1}{2}\|\boldsymbol{B}-\boldsymbol{Z}\|_F^2 \quad s.t. \quad \boldsymbol{A} \geqslant \boldsymbol{0}, diag(\boldsymbol{A}) = \boldsymbol{0}, \boldsymbol{A} = \boldsymbol{A}^T \tag{36}$$

By adding a constant $\frac{1}{2}\|\frac{\alpha}{\gamma}(diag(\boldsymbol{S})\boldsymbol{1}^T - \boldsymbol{S})\|_F^2$, the problem (36) can also be rewritten as:

$$\begin{aligned} \boldsymbol{A}^* = \arg\min_{\boldsymbol{A}} \frac{1}{2}(\|\boldsymbol{A}-\boldsymbol{Z}\|_F^2 + 2\frac{\alpha}{\gamma} < \boldsymbol{A}-\boldsymbol{Z}, (diag(\boldsymbol{S})\boldsymbol{1}^T - \boldsymbol{S}) > + \|\frac{\alpha}{\gamma}(diag(\boldsymbol{S})\boldsymbol{1}^T - \boldsymbol{S})\|_F^2) \\ s.t. \quad \boldsymbol{A} \geqslant \boldsymbol{0}, diag(\boldsymbol{A}) = \boldsymbol{0}, \boldsymbol{A} = \boldsymbol{A}^T \end{aligned} \tag{37}$$

Then, we have

$$\boldsymbol{A}^* = \arg\min_{\boldsymbol{A}} \frac{1}{2}\|\boldsymbol{A}-\boldsymbol{Z} + \frac{\gamma}{\lambda}(diag(\boldsymbol{S})\boldsymbol{1}^T - \boldsymbol{S})\|_F^2 \quad s.t. \quad \boldsymbol{A} \geqslant \boldsymbol{0}, diag(\boldsymbol{A}) = \boldsymbol{0}, \boldsymbol{A} = \boldsymbol{A}^T \tag{38}$$

The proof is completed.

[8] https://en.wikipedia.org/wiki/Degree_matrix.

Appendix B

The problem (21) can be rewritten as

$$\boldsymbol{A}^* = \arg\min_{\boldsymbol{A}} \frac{1}{2}\|\boldsymbol{A}-\hat{\boldsymbol{M}}\|_F^2 \quad s.t. \quad \boldsymbol{A} \geqslant \boldsymbol{0}, diag(\boldsymbol{A}) = \boldsymbol{0}, \boldsymbol{A} = \boldsymbol{A}^T \tag{39}$$

From the constraint $\boldsymbol{A} = \boldsymbol{A}^T$, we know $\|\boldsymbol{A}-\hat{\boldsymbol{M}}\|_F^2 = \|\boldsymbol{A}-\hat{\boldsymbol{M}}^T\|_F^2$. Thus

$$\frac{1}{2}\|\boldsymbol{A}-\hat{\boldsymbol{M}}\|_F^2 = \frac{1}{4}\|\boldsymbol{A}-\hat{\boldsymbol{M}}\|_F^2 + \frac{1}{4}\|\boldsymbol{A}-\hat{\boldsymbol{M}}^T\|_F^2 \geqslant \frac{1}{2}\|\boldsymbol{A}-(\hat{\boldsymbol{M}}+\hat{\boldsymbol{M}}^T)/2\|_F^2, \tag{40}$$

which has the solution

$$\boldsymbol{A}^* = \max\left(0, \frac{\hat{\boldsymbol{M}}+\hat{\boldsymbol{M}}^T}{2}\right) \tag{41}$$

The proof is completed.

References

[1] J. Bobadilla, R. Bojorque, A.H. Esteban, R. Hurtado, Recommender systems clustering using bayesian non negative matrix factorization, IEEE Access 6 (2018) 3549–3564.
[2] F. Bu, A high-order clustering algorithm based on dropout deep learning for heterogeneous data in cyber-physical-social systems, IEEE Access 6 (2018) 11687–11693.
[3] Z. Bu, J. Cao, H.-J. Li, G. Gao, H. Tao, Gleam: a graph clustering framework based on potential game optimization for large-scale social networks, Knowl. Inf. Syst. 55 (3) (2018) 741–770.
[4] J. Dattorro, Convex optimization & Euclidean distance geometry, Lulu. com, 2010.
[5] Z. Deng, K.-S. Choi, Y. Jiang, J. Wang, S. Wang, A survey on soft subspace clustering, Inf. Sci. 348 (2016) 84–106.
[6] S. Ding, H. Jia, M. Du, Y. Xue, A semi-supervised approximate spectral clustering algorithm based on hmrf model, Inf. Sci. 429 (2018) 215–228.
[7] B. Du, Z. Wang, L. Zhang, L. Zhang, D. Tao, Robust and discriminative labeling for multi-label active learning based on maximum correntropy criterion, IEEE Trans. Image Process. 26 (4) (2017) 1694–1707.
[8] L. Du, P. Zhou, L. Shi, H. Wang, M. Fan, W. Wang, Y.-D. Shen, Robust multiple kernel k-means using l21-norm., in: IJCAI, 2015, pp. 3476–3482.
[9] E. Elhamifar, R. Vidal, Sparse subspace clustering, in: Computer Vision and Pattern Recognition, 2009. CVPR 2009. IEEE Conference on, IEEE, 2009, pp. 2790–2797.
[10] E. Elhamifar, R. Vidal, Sparse subspace clustering: algorithm, theory, and applications, IEEE Trans. Pattern Anal. Mach. Intell. 35 (11) (2013) 2765–2781.
[11] M.M. Ferdaus, S.G. Anavatti, M.A. Garratt, M. Pratama, Fuzzy clustering based nonlinear system identification and controller development of pixhawk based quadcopter, in: Advanced Computational Intelligence (ICACI), 2017 Ninth International Conference on, IEEE, 2017, pp. 223–230.
[12] R. He, T. Tan, L. Wang, W.-S. Zheng, l2,1 regularized correntropy for robust feature selection, in: Computer Vision and Pattern Recognition (CVPR), 2012 IEEE Conference on, IEEE, 2012, pp. 2504–2511.
[13] R. He, W.-S. Zheng, B.-G. Hu, Maximum correntropy criterion for robust face recognition, IEEE Trans. Pattern Anal. Mach. Intell. 33 (8) (2011) 1561–1576.
[14] H.-C. Huang, Y.-Y. Chuang, C.-S. Chen, Affinity aggregation for spectral clustering, in: Computer Vision and Pattern Recognition (CVPR), 2012 IEEE Conference on, IEEE, 2012, pp. 773–780.
[15] H.-C. Huang, Y.-Y. Chuang, C.-S. Chen, Multiple kernel fuzzy clustering, IEEE Trans. Fuzzy Syst. 20 (1) (2012) 120–134.
[16] J. Huang, F. Nie, H. Huang, A new simplex sparse learning model to measure data similarity for clustering., in: IJCAI, 2015, pp. 3569–3575.
[17] P. Ji, T. Zhang, H. Li, M. Salzmann, I. Reid, Deep subspace clustering networks, in: Advances in Neural Information Processing Systems, 2017, pp. 24–33.
[18] Z. Kang, X. Lu, J. Yi, Z. Xu, Self-weighted multiple kernel learning for graph-based clustering and semi-supervised classification, arXiv:1806.07697. (2018a).
[19] Z. Kang, C. Peng, Q. Cheng, Z. Xu, Unified spectral clustering with optimal graph, in: Thirty-Second AAAI Conference on Artificial Intelligence, 2018.
[20] Z. Kang, L. Wen, W. Chen, Z. Xu, Low-rank kernel learning for graph-based clustering, Knowl. Based Syst. 163 (2019) 510–517.
[21] M. Keuper, S. Tang, B. Andres, T. Brox, B. Schiele, Motion segmentation & multiple object tracking by correlation co-clustering, IEEE Trans. Pattern Anal. Mach. Intell. (2018).
[22] G. Liu, Z. Lin, Y. Yu, Robust subspace segmentation by low-rank representation, in: Proceedings of the 27th international conference on machine learning (ICML-10), 2010, pp. 663–670.
[23] W. Liu, P.P. Pokharel, J.C. Príncipe, Correntropy: properties and applications in non-gaussian signal processing, IEEE Trans. Signal Process. 55 (11) (2007) 5286–5298.
[24] C. Lu, J. Feng, Z. Lin, T. Mei, S. Yan, Subspace clustering by block diagonal representation, IEEE Trans. Pattern Anal. Mach. Intell. (2018).
[25] C. Lu, J. Tang, M. Lin, L. Lin, S. Yan, Z. Lin, Correntropy induced l2 graph for robust subspace clustering, in: Proceedings of the IEEE international conference on computer vision, 2013, pp. 1801–1808.
[26] Y. Meng, J. Liang, F. Cao, Y. He, A new distance with derivative information for functional k-means clustering algorithm, Inf. Sci. (2018).
[27] A.Y. Ng, M.I. Jordan, Y. Weiss, On spectral clustering: analysis and an algorithm, in: Advances in neural information processing systems, 2002, pp. 849–856.
[28] V.M. Patel, R. Vidal, Kernel sparse subspace clustering, in: Image Processing (ICIP), 2014 IEEE International Conference on, IEEE, 2014, pp. 2849–2853.
[29] C. Peng, Z. Kang, Y. Hu, J. Cheng, Q. Cheng, Robust graph regularized nonnegative matrix factorization for clustering, ACM Trans. Knowl. Discovery Data (TKDD) 11 (3) (2017) 33.
[30] X. Peng, J. Feng, S. Xiao, W.-Y. Yau, J.T. Zhou, S. Yang, Structured autoencoders for subspace clustering, IEEE Trans. Image Process. (2018).
[31] P.P. Pokharel, W. Liu, J.C. Principe, A low complexity robust detector in impulsive noise, Signal Process. 89 (10) (2009) 1902–1909.
[32] B. Schölkopf, A. Smola, K.-R. Müller, Nonlinear component analysis as a kernel eigenvalue problem, Neural Comput. 10 (5) (1998) 1299–1319.
[33] A. Sekmen, A.B. Koku, M. Parlaktuna, A. Abdul-Malek, N. Vanamala, Unsupervised deep learning for subspace clustering, in: Big Data (Big Data), 2017 IEEE International Conference on, IEEE, 2017, pp. 2089–2094.
[34] E. Sherkat, S. Nourashrafeddin, E.E. Milios, R. Minghim, Interactive document clustering revisited: a visual analytics approach, in: 23rd International Conference on Intelligent User Interfaces, ACM, 2018, pp. 281–292.
[35] X. Shi, Z. Guo, F. Xing, J. Cai, L. Yang, Self-learning for face clustering, Pattern Recognit. 79 (2018) 279–289.
[36] M.C. Tsakiris, R. Vidal, Algebraic clustering of affine subspaces, IEEE Trans. Pattern Anal. Mach. Intell. 40 (2) (2018) 482–489.
[37] V. Vapnik, The Nature of Statistical Learning Theory, Springer Science & Business Media, 2013.
[38] T. Wang, J. Yang, Z. Ji, Q. Sun, Probabilistic diffusion for interactive image segmentation, IEEE Trans. Image Process. 28 (1) (2019) 330–342.
[39] Y.-X. Wang, H. Xu, C. Leng, Provable subspace clustering: when lrr meets ssc, in: Advances in Neural Information Processing Systems, 2013, pp. 64–72.

[40] W. Xu, X. Liu, Y. Gong, Document clustering based on non-negative matrix factorization, in: Proceedings of the 26th annual international ACM SIGIR conference on Research and development in informaion retrieval, ACM, 2003, pp. 267–273.
[41] Z. Xu, R. Jin, I. King, M. Lyu, An extended level method for efficient multiple kernel learning, in: Advances in neural information processing systems, 2009, pp. 1825–1832.
[42] M. Yang, X. Wang, W. Liu, L. Shen, Joint regularized nearest points for image set based face recognition, Image Vis. Comput. 58 (2017) 47–60.
[43] J. Yao, X. Cao, Q. Zhao, D. Meng, Z. Xu, Robust subspace clustering via penalized mixture of gaussians, Neurocomputing 278 (2018) 4–11.
[44] C. You, D. Robinson, R. Vidal, Scalable sparse subspace clustering by orthogonal matching pursuit, in: Proceedings of the IEEE conference on computer vision and pattern recognition, 2016, pp. 3918–3927.
[45] X. Zhang, H. Sun, Z. Liu, Z. Ren, Q. Cui, Y. Li, Robust low-rank kernel multi-view subspace clustering based on the schatten p-norm and correntropy, Inf. Sci. 477 (2019) 430–447.
[46] Y. Zhang, Z. Jiang, L.S. Davis, Learning structured low-rank representations for image classification, in: Proceedings of the IEEE conference on computer vision and pattern recognition, 2013, pp. 676–683.
[47] H. Zhu, R. Vial, S. Lu, X. Peng, H. Fu, Y. Tian, X. Cao, Yotube: searching action proposal via recurrent and static regression networks, IEEE Trans. Image Process. 27 (6) (2018) 2609–2622.
[48] R. Zhu, J.-H. Xue, On the orthogonal distance to class subspaces for high-dimensional data classification, Inf. Sci. 417 (2017) 262–273.