Multiple Kernel Clustering with Kernel k-Means Coupled Graph Tensor Learning

Zhenwen Ren[1,2], **Quansen Sun**[1*] **and Dong Wei**[1]

[1] School of Computer Science and Engineering, Nanjing University of Science and Technology, Nanjing, China
[2] School of National Defence Science and Technology, Southwest University of Science and Technology, Mianyang, China
{rzw, sunquansen}@njust.edu.cn, dongweinjust@163.com

Abstract

Kernel k-means (KKM) and spectral clustering (SC) are two basic methods used for multiple kernel clustering (MKC), which have both been widely used to identify clusters that are non-linearly separable. However, both of them have their own shortcomings: 1) the KKM-based methods usually focus on learning a discrete *clustering indicator matrix* via a combined consensus kernel, but cannot exploit the high-order affinities of all pre-defined base kernels; and 2) the SC-based methods require a robust and meaningful *affinity graph* in kernel space as input in order to form clusters with desired clustering structure. In this paper, a novel method, kernel k-means coupled graph tensor (KCGT), is proposed to graciously couple KKM and SC for seizing their merits and evading their demerits simultaneously. In specific, we innovatively develop a new graph learning paradigm by leveraging an explicit theoretical connection between clustering indicator matrix and affinity graph, such that the affinity graph propagated from KKM enjoys the valuable block diagonal and sparse property. Then, by using this graph learning paradigm, base kernels can produce multiple candidate affinity graphs, which are stacked into a low-rank graph tensor for capturing the high-order affinity of all these graphs. After that, by averaging all the frontal slices of the tensor, a high-quality affinity graph is obtained. Extensive experiments have shown the superiority of KCGT compared with the state-of-the-art MKC methods.

Introduction

To overcome the challenging problem of non-linear data clustering, kernel methods have been widely studied. However, for the traditional kernel methods, the most suitable kernel and the associated parameter for a specific dataset are difficult to decide in advance. To alleviate the problem of kernel choice, multiple kernel clustering (MKC) has attracted intense attention in recent years, which only requires users to pre-specify a set of base kernels, and automatically fuse the information of them for clustering purpose. Overall, kernel k-means (KKM) and spectral clustering (SC) are two basic methods used for MKC.

KKM-based MKC methods usually learn a consensus kernel by linearly or non-linearly combining the given base kernels, and simultaneously/subsequently employ the the kernel k-means to output the clustering indicator matrix. Some representative methods, (Huang, Chuang, and Chen 2012; Gönen and Margolin 2014; Du et al. 2015), assume that the optimal kernel is a linear combination of base kernels. In (Huang, Chuang, and Chen 2012), the algorithm alternatively performs kernel k-means and updates the kernel weights. The method proposed in (Gönen and Margolin 2014) combines the base kernels by sample-adaptive weights. To improve the robustness with respect to noises and outliers, (Du et al. 2015) presents a robust kernel k-means by using ℓ_{21}-norm to measure the distances between data points and cluster centers. By relying on neighborhood kernel learning, there are also other methods (Liu et al. 2016, 2017, 2020a), which learn a non-linearly combination of base kernels to enhance the representability of the optimal kernel. Both (Liu et al. 2016) and (Liu et al. 2017) incorporate a matrix-induced regularization to sufficiently consider the correlation among base kernels, with the difference that the former learns a clustering indicator matrix directly, and the latter learns a neighborhood consensus kernel as the input of kernel k-means. Recently, (Liu et al. 2020a) proposes to maximally align the consensus partition with the weighted base partitions to significantly reduce the computational complexity.

SC-based MKC methods consist of first constructing an affinity matrix (graph) based on graphical representations of the relationships among data points in kernel space, and then applying spectral clustering algorithm to obtain the label assignments. It is well know that the performance of such SC-based methods are heavily dependent on the quality of the affinity graph (Huang, Nie, and Huang 2015). Therefore, these methods aim to learn a more high-quality affinity graph by jointing some graph learning paradigms. In the perspective of graph learning, the existing researches on this aspect can roughly be grouped into two categories. The first category learns a consensus graph via kernelized self-expressiveness subspace learning (Kang et al. 2018; Zhou et al. 2019b; Ren and Sun 2020). In (Kang et al. 2018), a consensus affinity graph is directly learned for clustering and semi-supervised classification. (Zhou et al. 2019b) defines a neighbor kernel and linearly combine these base neighbor kernels to learn a consensus affinity matrix via an exact-rank-constrained. The work in (Ren and Sun 2020) considers the global and local structure information of the data simul-

*Corresponding author.

taneously, such that the quality of the affinity graph can be improved. By leveraging the adaptive neighbors graph learning, the other category assigns a probability for each sample as the neighborhood of another sample to construct an affinity graph, such that the local manifold structure of the data can be effectively exploited (Ren et al. 2020a,c). The work in (Ren et al. 2020c) learns multiple candidate affinity graphs from base kernels, and then fuse these graphs to learn a consensus affinity graph via a two-step manner. Similar work has also been done in (Ren et al. 2020a), with the major difference that the candidate graphs and consensus graph are learned in a one-step learning paradigm.

Though the above methods have obtained excellent performance for non-linear data clustering, we find that: 1) due to the limitations of the model, KKM-based methods cannot exploit the high-order affinities of all base kernels, leading to unsatisfying clustering performance; and 2) SC-based methods require a robust and meaningful affinity graph in kernel space; meanwhile, except for self-expressiveness subspace learning and adaptive neighbors graph learning (Ren et al. 2020c), a new graph learning paradigm is urgent needed to enlarge the graph learning method for non-linear data.

Regarding the observations mentioned above, in this paper, we propose a novel MKC method, namely *kernel k-means coupled graph tensor* (KCGT), which seamlessly couples the ideas of both KKM and SC. Specifically, KCGT proposes a graph learning paradigm, *kernel k-means graph* (KKG), to bridge the relationship between the clustering indicator matrix and affinity graph dexterously. Consequently, multiple candidate affinity graphs can be obtained from base kernels via KKG. Afterwards, the high-order affinities of all candidate graphs is captured in a three-order graph tensor by introducing tensor singular value decomposition based tensor nuclear norm, such that an optimal affinity graph can be obtained subsequently. After that, the spectral clustering algorithm is employed to segment the clusters. The contributions of this paper are summarized as follows,

- A new graph learning paradigm, KKG, is proposed to learn a high-quality affinity graph, which bridges the relationship between the clustering indicator matrix of KKM and the affinity graph of SC dexterously, such that the learned candidate affinity graph enjoys the valuable block diagonal and sparse property.
- A novel MKC method, KCGT, is proposed for non-linear data clustering, which deeply exploit the high-order structure information of all candidate graphs produced by KKG. Moreover, an effective solver based on the alternating direction method of multipliers (ADMM) is developed to solve the objective function.
- Compared with state-of-the-art methods, including KKM-based ones and SC-based ones, the superiority of KCGT is demonstrated by conducting extensive experiments.

Notation Summary: We denote bold upper case letters for matrices, *e.g.*, $\mathbf{M}$, bold lower case letters for vectors, *e.g.*, $\mathbf{m}$, lower case letters for entries of vectors or matrices, *e.g.*, m_{ij}, bold calligraphy letters for tensors, *e.g.*, $\boldsymbol{\mathcal{M}}$. Positive semi-definite $\mathbf{M}$ is denoted as $\mathbf{M} \succeq 0$. We denote $\mathbf{M}_1 \preceq \mathbf{M}_2$ or $\mathbf{M}_2 \succeq \mathbf{M}_1$ if $\mathbf{M}_2 - \mathbf{M}_1 \succeq 0$ for symmetric matrices $\mathbf{M}_1$, $\mathbf{M}_2$. For a 3-order tensor $\boldsymbol{\mathcal{M}} \in \mathbb{R}^{n_1 \times n_2 \times n_3}$, it involves the main two definitions, *i.e.*, fiber and slice. The fiber represents that an order is free while the other two orders are fixed, *i.e.*, mode-1 fiber $\boldsymbol{\mathcal{M}}(:, j, k)$, mode-2 fiber $\boldsymbol{\mathcal{M}}(i, :, k)$, and mode-3 fiber $\boldsymbol{\mathcal{M}}(i, j, :)$. The slice indicates that only one order is fixed while the other two orders are free, *i.e.*, horizontal slice $\boldsymbol{\mathcal{M}}(i; :, :)$, lateral slice $\boldsymbol{\mathcal{M}}(:, j, :)$, and frontal slice $\boldsymbol{\mathcal{M}}(:, :, k)$. For simplicity, $\boldsymbol{\mathcal{M}}(:, :, k)$ is simplified as $\mathbf{M}^{(k)}$ or $\boldsymbol{\mathcal{M}}^{(k)}$. And, $\boldsymbol{\mathcal{M}}_f = \texttt{fft}(\boldsymbol{\mathcal{M}}, [\,], 3)$ and $\boldsymbol{\mathcal{M}} = \texttt{ifft}(\boldsymbol{\mathcal{M}}_f, [\,], 3)$ are the fast Fourier transformation (FFT) and inverse FFT along the third direction of tensor $\boldsymbol{\mathcal{M}}$, respectively. $\texttt{bvec}(\boldsymbol{\mathcal{M}}) = [\mathbf{M}^{(1)}; \mathbf{M}^{(2)}; \cdots; \mathbf{M}^{(n_3)}] \in \mathbb{R}^{n_1 n_3 \times n_2}$ and $\texttt{fold}(\texttt{bvec}(\boldsymbol{\mathcal{M}})) = \boldsymbol{\mathcal{M}}$ are defined as the block vectorizing and the inverse operation of $\texttt{bvec}$, respectively. $\texttt{bdiag}(\boldsymbol{\mathcal{M}}) \in \mathbb{R}^{n_1 n_3 \times n_2 n_3}$ and $\texttt{bcirc}(\boldsymbol{\mathcal{M}}) \in \mathbb{R}^{n_1 n_3 \times n_2 n_3}$ denotes the block diagonal matrix and the corresponding block circulant matrix, respectively.

Related Work

Kernel k-Means (KKM) Clustering

Let $\mathbf{X} = \{\mathbf{x}_i\}_{i=1}^n$, $\mathbf{x}_i \in \mathbb{R}^d$ be a data set consisting of n samples with d dimensions from c clusters, and $\phi : \mathbf{x} \in \mathbb{R} \longmapsto \mathbb{H}$ be a feature mapping which maps $\mathbf{x}$ onto a reproducing Hilbert space. The objective of KKM is to minimize the sum-of-squares loss over the clustering indicator matrix $\mathbf{F} \in \{0, 1\}^{n \times c}$, which can be formulated as (Liu et al. 2020b),

$$\min_{\mathbf{F} \in \{0,1\}^{n \times c}} \sum_{i=1}^{n} \sum_{j=1}^{c} f_{ij} \left\| \phi\left(\mathbf{x}_i\right) - \boldsymbol{\mu}_j \right\|_2^2 \text{ s.t.} \sum_{j=1}^{c} f_{ij} = 1, \quad (1)$$

where $n_j = \sum_{i=1}^n f_{ij}$ and $\boldsymbol{\mu}_j = \frac{1}{n_j} \sum_{i=1}^n f_{ij} \phi(\mathbf{x}_i)$ are the size and distribution centroid of the j-th cluster, respectively.

Note here that the variables $\mathbf{F}$ is discrete, which makes (1) very difficult to solve. However, we can approximate (1) through relaxing $\mathbf{F}$ to take arbitrary real values. Specifically, by defining $\mathbf{D} = \mathrm{diag}([1/n_1, \cdots, 1/n_c])$ and $\mathbf{P} = \mathbf{F}\mathbf{D}^{\frac{1}{2}}$, and letting $\mathbf{P}$ take real values, the relaxed version of (1) is

$$\min_{\mathbf{P} \in \mathbb{R}^{n \times c}} \mathrm{Tr}\left(\mathbf{H}\left(\mathbf{I} - \mathbf{P}\mathbf{P}^\intercal\right)\right) \text{ s.t. } \mathbf{P}^\intercal\mathbf{P} = \mathbf{I}. \quad (2)$$

where $\mathbf{H} \in \mathbb{R}^{n \times n}$ is the kernel Gram matrix with entry $h_{ij} = \phi(\mathbf{x}_i)^\intercal \phi(\mathbf{x}_j)$. Accordingly, one can obtain the optimal $\mathbf{P}$ by choosing the c eigenvectors that correspond to the c largest eigenvalues of $\mathbf{H}$.

Preliminaries of Three-Order Tensor

Here, the preliminaries related to three-order tensor are introduced to help understand tensor better (Wu et al. 2020).

Definition 1 (T-Product). The t-product between two tensors with matched dimensions, $\boldsymbol{\mathcal{M}} \in \mathbb{R}^{n_1 \times n_2 \times n_3}$ and $\boldsymbol{\mathcal{N}} \in \mathbb{R}^{n_2 \times n_4 \times n_3}$, is defined as $\boldsymbol{\mathcal{M}} * \boldsymbol{\mathcal{N}} \in \mathbb{R}^{n_1 \times n_4 \times n_3}$, *i.e.*,

$$\boldsymbol{\mathcal{M}} * \boldsymbol{\mathcal{N}} = \texttt{fold}(\texttt{bcirc}(\boldsymbol{\mathcal{M}})\, \texttt{bvec}(\boldsymbol{\mathcal{N}}))\,. \quad (3)$$

Definition 2 (Identity Tensor). The identity tensor $\boldsymbol{\mathcal{I}} \in \mathbb{R}^{n_1 \times n_1 \times n_3}$ satisfies that its first frontal slice is an identity matrix while the others frontal slices are zeros.

Definition 3 (Tensor Transpose). The transpose operator of $\mathcal{M} \in \mathbb{R}^{n_1 \times n_2 \times n_3}$ is denoted as $\mathcal{M}^\intercal \in \mathbb{R}^{n_2 \times n_1 \times n_3}$, which is calculated by transposing all frontal slices of $\mathcal{M}$.

Definition 4 (Orthogonal Tensor). A tensor $\mathcal{F} \in \mathbb{R}^{n_1 \times n_1 \times n_3}$ is orthogonal if

$$\mathcal{F}^\intercal * \mathcal{F} = \mathcal{F} * \mathcal{F}^\intercal = \mathcal{I}\,. \tag{4}$$

Definition 5 (f-Diagonal Tensor). A tensor is called f-diagonal if each of its frontal slices is diagonal matrix.

Definition 6 (Tensor Singular Value Decomposition, t-SVD). The t-SVD of tensor $\mathcal{M} \in \mathbb{R}^{n_1 \times n_2 \times n_3}$ is defined as

$$\mathcal{M} = \mathcal{U} * \mathcal{G} * \mathcal{V}^\intercal\,. \tag{5}$$

where $\mathcal{G} \in \mathbb{R}^{n_1 \times n_2 \times n_3}$ is a f-diagonal tensor, and $\mathcal{U} \in \mathbb{R}^{n_1 \times n_1 \times n_3}$ and $\mathcal{V} \in \mathbb{R}^{n_2 \times n_2 \times n_3}$ are orthogonal tensors.

Definition 7 (t-SVD Based Tensor Nuclear Norm, t-TNN). $\|\mathcal{M}\|_{\circledast}$ is the t-SVD based tensor nuclear norm of $\mathcal{M} \in \mathbb{R}^{n_1 \times n_2 \times n_3}$, which is given by the sum of singular values of all the frontal slices of $\mathcal{M}_f$, *i.e.*,

$$\|\mathcal{M}\|_{\circledast} = \sum_{k=1}^{n_3} \|\mathcal{M}_f^{(k)}\|_* = \sum_{i=1}^{\min(n_1,n_2)} \sum_{k=1}^{n_3} |\mathcal{G}_f^{(k)}(i,i)|. \tag{6}$$

According to the frontal slices of $\{\mathcal{M}_f^{(k)}\}_{k=1}^{n_3}$, we employ t-SVD, $\mathcal{M}_f^{(k)} = \mathcal{U}_f^{(k)} \mathcal{G}_f^{(k)} \mathcal{V}_f^{(k)^\intercal}$, to compute $\{\mathcal{G}_f^{(k)}\}_{k=1}^{n_3}$. It has been demonstrated that the t-TNN can exploit the structural information of a tensor better than the unfolding-based tensor nuclear norm (Zhang et al. 2015).

Proposed Methodology

Kernel k-Means Graph (KKG) Learning Paradigm

The key of graph-based method is to construct an affinity graph (Zhang et al. 2015). As we know, the block diagonal property of affinity graph plays a significant role for graph-based learning task (Lu et al. 2019) and is explicitly pursued in recent SC-based MKC methods (Yang et al. 2019; Kang et al. 2018; Ren et al. 2019, 2020c).

Now, we aim to pursue a high-quality affinity graph based on KKM for clustering purpose. Ideally, since $\mathbf{P} := \{\mathbf{P}|\mathbf{P} \in \mathbb{R}^{n \times c}, \mathbf{P}^\intercal\mathbf{P} = \mathbf{I}\}$ is the clustering indicator matrix produced by KKM, so $\mathbf{PP}^\intercal$ is a strictly block diagonal matrix. That is, we can learn a symmetric affinity graph $\mathbf{S} \in \mathbb{R}^{n \times n}$ that best approximates $\mathbf{PP}^\intercal$, *i.e.*, $\mathbf{S} = \mathbf{PP}^\intercal$. Here, we give a toy example to show the connection between the clustering indicator matrix $\mathbf{P}$ and the affinity graph $\mathbf{S}$ mathematically. As shown in Fig. 1, there are 1, 2, 3 samples in 3 different clusters, respectively. Ideally, the affinity graph $\mathbf{S}$ produced by $\mathbf{PP}^\intercal$ has obvious spare and block diagonal property.

In practice, however, it is difficult to guarantee the learned graph $\mathbf{S}$ can educe an eligible indicator matrix $\mathbf{P}$, under without any constraint. As for this point, we enforce graph $\mathbf{S}$ to have a good sharp by introducing Theorem 1.

Theorem 1 (Boyd, Boyd, and Vandenberghe 2004) *Let* $\mathbb{S}_1 := \{\mathbf{S} = \mathbf{PP}^\intercal|\mathbf{P} \in \mathbb{R}^{n \times c}, \mathbf{P}^\intercal\mathbf{P} = \mathbf{I}\}$ *and* $\mathbb{S}_2 := \{\mathbf{S}|\mathbf{S} = \mathbf{S}^\intercal, Tr(\mathbf{S}) = c, \mathbf{0} \preceq \mathbf{S} \preceq \mathbf{1}\}$*,* $\mathbb{S}_2$ *is the convex hull of* $\mathbb{S}_1$*, and* $\mathbb{S}_1$ *is exactly the set of extreme points of* $\mathbb{S}_2$.

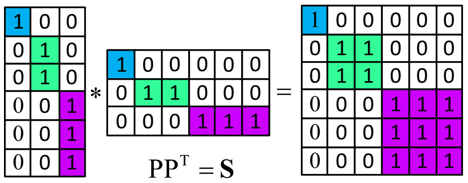


Figure 1: The relation between indicator $\mathbf{P}$ and graph $\mathbf{S}$.

Furthermore, as shown in Fig. 1, it is often more desirable to obtain a sparse affinity graph with high intra-cluster affinity and zero inter-cluster affinity. However, the learned $\mathbf{S}$ is usually not sparse in practice. Therefore, it is essential to impose the spare norm on $\mathbf{S}$ (*i.e.*, $\|\mathbf{S}\|_1$). Hereto, the high-quality affinity graph $\mathbf{S}$ can be obtained by

$$\begin{aligned} &\min_{\mathbf{S}} \ \mathtt{Tr}(-\mathbf{HS}) + \alpha\|\mathbf{S}\|_1 \\ &\quad \text{s.t. } \mathbf{S} \in \mathbb{R}^{n \times n}, \mathtt{Tr}(\mathbf{S}) = c, \mathbf{S}^\intercal = \mathbf{S}, \mathbf{0} \preceq \mathbf{S} \preceq \mathbf{1}\,. \end{aligned} \tag{7}$$

where α is a balancing parameter. We dub this graph learning paradigm as kernel k-means graph (KKG).

In MKL scenarios, since a kernel pool consisting of m base kernels $\{\mathbf{H}^{(k)}\}_{k=1}^m$ is given in advance, we can obtain m candidate affinity kernel graphs $\{\mathbf{S}^{(k)}\}_{k=1}^m$ by leveraging KKG accordingly.

Kernel k-Means Coupled Graph Tensor (KCGT)

With m candidate kernel affinity graphs $\{\mathbf{S}^{(k)}\}_{k=1}^m$ at hand, we strongly desire that an intrinsic graph tensor can be learned to capture the complementary information with high-order affinity by integrating these m graphs. To achieve so, we stack the m kernel graphs into a tensor $\mathcal{S}^* \in \mathbb{R}^{n \times n \times m}$ (*i.e.*, $\mathcal{S}^* = \mathtt{bvfold}([\mathbf{S}^{(1)}; \cdots; \mathbf{S}^{(m)}])$) along the third dimension, and then rotate $\mathcal{S}^*$ to $\mathcal{S} \in \mathbb{R}^{n \times m \times n}$ (*i.e.*, $\mathtt{rotate}(\mathcal{S}^*)$), as illustrated in Fig 2. By the stacking and rotating operators, the graph tensor can better explore the high-order affinities between these candidate graphs. Meanwhile, instead of using $\mathcal{S}^*$, the computational complexity is largely reduced.

Due to the fact that m candidate kernel graphs origin from the same dataset, different $\mathbf{S}^{(k)}$ possesses some consensus structure information; on the other hand, considering the fact that the number of data points is usually much bigger than the number of clusters, thus the learned tensor $\mathcal{S}$ should enjoy the low-rank property (Lu et al. 2019). In this paper, a t-TNN term, $\|\mathcal{S}\|_{\circledast}$, is introduced to regularize $\mathcal{S}$ as a constraint of the intrinsic low-rank graph tensor. Hereto, the final objective function can be written as

$$\begin{aligned} &\min_{\mathbf{S}^{(k)}} \sum_{k=1}^{m} \mathtt{Tr}(-\mathbf{H}^{(k)}\mathbf{S}^{(k)}) + \alpha\|\mathbf{S}^{(k)}\|_1 + \beta\|\mathcal{S}\|_{\circledast} \\ &\quad \text{s.t. } \mathbf{S}^{(k)} \in \mathbb{R}^{n \times n}, \mathtt{Tr}(\mathbf{S}^{(k)}) = c, \\ &\qquad (\mathbf{S}^{(k)})^\intercal = \mathbf{S}^{(k)}, \mathbf{0} \preceq \mathbf{S}^{(k)} \preceq \mathbf{1}, \\ &\qquad \mathcal{S} = \mathtt{rotate}(\mathtt{bvfold}([\mathbf{S}^{(1)}; \cdots; \mathbf{S}^{(m)}]))\,. \end{aligned} \tag{8}$$

where α and β are the balancing parameters. This method is dubbed as kernel k-means coupled graph tensor (KCGT).

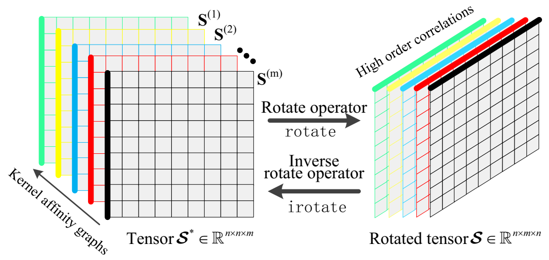


Figure 2: The rotated affinity graph tensor in KCGT. Note here that `rotate` and `irotate` are two shift functions.

Optimization

Solver for Proposed KCGT Method

The final problem (8) is convex and can be effectively solved by the alternating direction method of multipliers (ADMM). According to the principle of ADMM, we first introduce some auxiliary variables, $\{\mathbf{Z}^{(k)}\}_{k=1}^m$ and $\boldsymbol{\mathcal{A}}$, to make (8) separable, and then obtain the following augmented Lagrangian function

$$
\begin{aligned}
&\mathcal{L}(\{\mathbf{S}^{(k)}, \mathbf{Z}^{(k)}\}_{k=1}^m, \boldsymbol{\mathcal{A}}) = \\
&\sum_{k=1}^m \mathtt{Tr}(-\mathbf{H}^{(k)}\mathbf{S}^{(k)}) + \alpha\|\mathbf{Z}^{(k)}\|_1 + \beta\|\boldsymbol{\mathcal{A}}\|_{\circledast} \\
&+ \frac{\mu}{2}\|\mathbf{S}^{(k)} - \mathbf{Z}^{(k)} + \frac{\mathbf{J}^{(k)}}{\mu}\|_F^2 + \frac{\mu}{2}\|\boldsymbol{\mathcal{S}} - \boldsymbol{\mathcal{A}} + \frac{\boldsymbol{\mathcal{Y}}}{\mu}\|_F^2 \\
&\quad \text{s.t. } \mathtt{Tr}(\mathbf{S}^{(k)}) = c, (\mathbf{S}^{(k)})^\intercal = \mathbf{S}^{(k)}, \mathbf{0} \preceq \mathbf{S}^{(k)} \preceq \mathbf{1} \\
&\quad \boldsymbol{\mathcal{S}} = \mathtt{rotate}(\mathtt{bvfold}([\mathbf{S}^{(1)}; \cdots; \mathbf{S}^{(m)}]))\,.
\end{aligned}
\tag{9}
$$

where $\{\mathbf{J}^{(k)}\}_{k=1}^m$ and $\boldsymbol{\mathcal{Y}}$ are the Lagrangian multiplier, and $\mu > 0$ is the penalty parameter. Then, we calculate each variable by fixing the remaining variables respectively.

▶ **Step-1, S-subproblem:** Fixing other variables, we update each candidate affinity graph $\{\mathbf{S}^{(k)}\}_{k=1}^m$ in $\boldsymbol{\mathcal{S}}$ via

$$
\begin{aligned}
&\min_{\mathbf{S}^{(k)}} \mathtt{Tr}(-\mathbf{H}^{(k)}\mathbf{S}^{(k)}) + \frac{\mu}{2}\|\mathbf{S}^{(k)} - \mathbf{Z}^{(k)} + \frac{\mathbf{J}^{(k)}}{\mu}\|_F^2 \\
&\qquad + \frac{\mu}{2}\|\mathbf{S}^{(k)} - \mathbf{A}^{(k)} + \frac{\mathbf{Y}^{(k)}}{\mu}\|_F^2 \\
&\quad \text{s.t. } \mathtt{Tr}(\mathbf{S}^{(k)}) = c, (\mathbf{S}^{(k)})^\intercal = \mathbf{S}^{(k)}, \mathbf{0} \preceq \mathbf{S}^{(k)} \preceq \mathbf{1},
\end{aligned}
\tag{10}
$$

where $\mathbf{A}^{(k)}$ and $\mathbf{Y}^{(k)}$ are the k-th frontal slice of the tensors $\mathtt{rotate}(\boldsymbol{\mathcal{A}})$ and $\mathtt{rotate}(\boldsymbol{\mathcal{Y}})$, respectively. Then, (10) can be rewritten as

$$
\begin{aligned}
&\min_{\mathbf{S}^{(k)}} \frac{1}{2}\|\mathbf{S}^{(k)} - \mathbf{B}^{(k)}\|_F^2 \\
&\quad \text{s.t. } \mathtt{Tr}(\mathbf{S}^{(k)}) = c, (\mathbf{S}^{(k)})^\intercal = \mathbf{S}^{(k)}, \mathbf{0} \preceq \mathbf{S}^{(k)} \preceq \mathbf{1},
\end{aligned}
\tag{11}
$$

where $\mathbf{B}^{(k)} = \frac{\mathbf{H}^{(k)} + \mu(\mathbf{Z}^{(k)} - \frac{\mathbf{J}^{(k)}}{2}) + \mu(\mathbf{A}^{(k)} - \frac{\mathbf{Y}^{(k)}}{2})}{2\mu}$. Such a problem can be solved by Theorem 2.

Theorem 2 *For a symmetric affinity matrix* $\mathbf{S} \in \mathbb{R}^{n\times n}$*, the singular value decomposition of* $\mathbf{S}$ *is denoted as* $\mathbf{B} = \mathbf{U}\mathtt{Diag}(\boldsymbol{\delta})\mathbf{U}^\intercal$. *The following problem*

$$
\min_{\mathbf{S}} \tfrac{1}{2}\|\mathbf{S} - \mathbf{B}\|_F^2 \text{ s.t. } \mathtt{Tr}(\mathbf{S}) = c, \mathbf{S}^\intercal = \mathbf{S}, \mathbf{0} \preceq \mathbf{S} \preceq \mathbf{1} \tag{12}
$$

has the optimal solution given by $\mathbf{S}^* = \mathbf{U}\mathtt{Diag}(\boldsymbol{\rho}^*)\mathbf{U}^\intercal$*, where* $\boldsymbol{\rho}^*$ *is the solution to*

$$
\min_{\boldsymbol{\rho}} \frac{1}{2}\|\boldsymbol{\rho} - \boldsymbol{\delta}\|_2^2, \text{ s.t. } 0 \le \boldsymbol{\rho} \le 1, \boldsymbol{\rho}^\intercal 1 = c. \tag{13}
$$

Refer to supplemental material for the proof.

Finally, problem (13) can be efficiently solved by an iterative algorithm as (Nie et al. 2016).

▶ **Step-2, Z-subproblem:** Fixing other variables, we update each auxiliary variable $\{\mathbf{Z}^{(k)}\}_{k=1}^m$ via

$$
\min_{\mathbf{Z}^{(k)}} \alpha\|\mathbf{Z}^{(k)}\|_1 + \frac{\mu}{2}\|\mathbf{Z}^{(k)} - (\mathbf{S}^{(k)} + \frac{\mathbf{J}^{(k)}}{\mu})\|_F^2. \tag{14}
$$

Such a problem can be solved as (Ren et al. 2020b).

▶ **Step-3, $\boldsymbol{\mathcal{A}}$-subproblem:** Fixing other variables, the optimization problem *w.r.t.* $\boldsymbol{\mathcal{A}}$ can be rewritten as

$$
\min_{\boldsymbol{\mathcal{A}}} \beta\|\boldsymbol{\mathcal{A}}\|_{\circledast} + \frac{\mu}{2}\|\boldsymbol{\mathcal{A}} - (\boldsymbol{\mathcal{S}} + \frac{\boldsymbol{\mathcal{Y}}}{\mu})\|_F^2\,, \tag{15}
$$

which is a t-TNN minimization problem. Let $\boldsymbol{\mathcal{B}} = \boldsymbol{\mathcal{S}} + \frac{\boldsymbol{\mathcal{Y}}}{\mu}$, (15) can be solved by applying the tensor tubal-shrinkage of $\boldsymbol{\mathcal{B}}$, according to the below Theorem 3.

Theorem 3 (Zhou et al. 2019a) *For a scalar* $\rho > 0$ *and two three-order tensors* $\boldsymbol{\mathcal{A}} \in \mathbb{R}^{n_1\times n_2\times n_3}$, $\boldsymbol{\mathcal{B}} \in \mathbb{R}^{n_1\times n_2\times n_3}$*, the global optimal solution of the following problem*

$$
\min_{\boldsymbol{\mathcal{A}}} \rho\|\boldsymbol{\mathcal{A}}\|_{\circledast} + \frac{1}{2}\|\boldsymbol{\mathcal{A}} - \boldsymbol{\mathcal{B}}\|_F^2 \tag{16}
$$

is given by the tensor tubal-shrinkage operator, *i.e.*,

$$
\boldsymbol{\mathcal{A}} = \mathcal{C}_{n_3\rho}(\boldsymbol{\mathcal{B}}) = \boldsymbol{\mathcal{U}} * \mathcal{C}_{n_3\rho}(\boldsymbol{\mathcal{G}}) * \boldsymbol{\mathcal{V}}^\intercal\,, \tag{17}
$$

where $\boldsymbol{\mathcal{B}} = \boldsymbol{\mathcal{U}} * \boldsymbol{\mathcal{G}} * \boldsymbol{\mathcal{V}}^\intercal$ and $\mathcal{C}_{n_3\rho} = \boldsymbol{\mathcal{G}} * \boldsymbol{\mathcal{Q}}$. $\boldsymbol{\mathcal{Q}} \in \mathbb{R}^{n_1\times n_2\times n_3}$ denotes a f-diagonal tensor and each diagonal element of $\boldsymbol{\mathcal{Q}}$ is defined as $\boldsymbol{\mathcal{Q}}_f(i,i,j) = \left(1 - \dfrac{n_3\rho}{\boldsymbol{\mathcal{G}}(i,i,j)}\right)_+$.

▶ **Step-4, ADMM variables:** We update the variables involved ADMM by

$$
\begin{aligned}
&\boldsymbol{\mathcal{Y}} = \boldsymbol{\mathcal{Y}} + \mu(\boldsymbol{\mathcal{A}} - \boldsymbol{\mathcal{S}}) \\
&\mathbf{J}^{(k)} = \mathbf{J}^{(k)} + \mu(\mathbf{S}^{(k)} - \mathbf{Z}^{(k)}) \\
&\mu = \min(\rho\,\mu, \mu_{\max})\ .
\end{aligned}
\tag{18}
$$

where ρ and $\mu_{\max}$ are the scalars involved ADMM.

In each iterator, the algorithm convergence is checked via

$$
\mathit{error} = \mathtt{max}(\|\boldsymbol{\mathcal{A}} - \boldsymbol{\mathcal{S}}\|_\infty, \|\mathbf{S}^{(k)} - \mathbf{Z}^{(k)}\|_\infty) < \epsilon, \tag{19}
$$

where $\epsilon = 10^{-5}$ is a threshold value. After obtaining the graph tensor $\boldsymbol{\mathcal{S}}$ with size $n \times m \times n$, we can `irotate` it to size $n \times n \times m$, *i.e.*, $\boldsymbol{\mathcal{S}}^* = \mathtt{irotate}(\boldsymbol{\mathcal{S}})$. Then, we compute the final comprehensive affinity graph $\bar{\mathbf{S}}$ by averaging all frontal slices of $\boldsymbol{\mathcal{S}}^*$, *i.e.*,

$$
\bar{\mathbf{S}} = \frac{1}{m}\sum_{k=1}^m \boldsymbol{\mathcal{S}}^*(:,:,k)\,. \tag{20}
$$

Subsequently, take $\bar{\mathbf{S}}$ as input, the spectral clustering algorithm is employed to pursue the clustering assignments. The pseudo-code of KCGT is depicted as in Algorithm 1.

Algorithm 1 Algorithm of the proposed KCGT method.

```
Input: $m$ base kernel $\{\mathbf{H}^k\}_{k=1}^m$, and parameters $\alpha$ and $\beta$.
    Initialize: $\{\mathbf{S}^{(k)} = \mathbf{Z}^{(k)}\}_{k=1}^m = \mathbf{I}$, $\boldsymbol{\mathcal{A}} = \boldsymbol{\mathcal{S}}$, $\mu = 10^{-4}$,
    $\rho = 1.2$, and $\mu_{max} = 10^{10}$.
1: while not converge do
2:     Update candidate graphs $\{\mathbf{S}^{(k)}\}_{k=1}^m$ via Eq. (10).
3:     Update auxiliary variables $\{\mathbf{Z}^{(k)}\}_{k=1}^m$ via Eq. (14).
4:     Update graph tensor $\boldsymbol{\mathcal{A}}$ via Eq. (15).
5:     Update ADMM involved variables via (18).
6: end while
7: Perform spectral clustering using affinity matrix $\bar{\mathbf{S}}$.
Output: The clustering results: ACC, NMI, and purity.
```

Computational Complexity

Then, we present the discussion about the computational cost of Algorithm 1. For updating $\{\mathbf{S}^{(k)}\}_{k=1}^m$, the computational complexity of the *step 1* is $\mathcal{O}(mn^3)$. For updating $\mathbf{Z}^{(k)}$, the computational complexity of the *step 2* is $\mathcal{O}(m)$. Meanwhile, the *step 1* and *step 2* can be easily parallelized. For updating $\boldsymbol{\mathcal{A}}$, we need to calculate the FFT and inverse FFT of the tensor $\boldsymbol{\mathcal{S}} \in \mathbb{R}^{n\times m\times n}$ along the third dimension, which takes $\mathcal{O}(mn^2 \log(n))$; moreover, we need to compute the SVD of each frontal slice of the tensor $\boldsymbol{\mathcal{S}}$ in the Fourier domain with complexity $\mathcal{O}(m^2n^2)$. Therefore, the *step 3* takes $\mathcal{O}(mn^2 \log(n) + m^2n^2)$. Note that without `rotate` operator, the complexity is $\mathcal{O}(mn^2 \log(m) + mn^3)$, hence it is necessary to employ `rotate` operator on graph tensor $\boldsymbol{\mathcal{S}}^*$. For the variables involved in ADMM, the computation of *step 4* at each iteration will take $\mathcal{O}(m)$. Theoretically, the computational cost of Algorithm 1 is $\mathcal{O}(t(mn^3 + mn^2 \log(n) + m^2n^2 + m))$, where t denotes the total number of iterations. Till the solver is convergence, the spectral clustering adopted for clustering usually costs $\mathcal{O}(n^3)$. In reality, we have $t \ll n$ and $m \ll n$. Thus the overall cost of Algorithm 1 is $\mathcal{O}(n^3)$. As a matter of fact, the computational complexity of KCGT is the same as that of the existing overwhelming majority MKC methods.

Experiment

Datasets and Settings

Eight public benchmark datasets (*i.e.*, BBCSport2, ProteinFold, Flower17, Caltech101, Mfeat, UCI-Digit, CCV, and Flower102) of various categories are employed to evaluate the performance of the proposed method, and all the kernel matrices of these datasets are pre-computed and publicly available (Liu et al. 2017; Zhou et al. 2019b; Wang et al. 2019). The summaries of these datasets are listed in Table 1. From this table, it can be observed that the number of kernels, clusters, samples and categories of these datasets show considerable variations. Note here that Flower17 is a deep feature dataset produced by deep convolutional neural networks as (Zhou et al. 2019b).

Comparison Methods

We compare the proposed KCGT method with the following state-of-the-art MKC methods, *i.e.*,

Dataset	Category	Samples	Clusters	Kernels	Size
		n	c	m	
BBCSport2	News article	554	5	2	4.40M
ProteinFold	Protein sequence	694	27	12	44.5M
Flower17	Image	1360	17	3	94.3M
Caltech101	Image	1530	102	25	0.32G
Mfeat	Image	2000	10	12	0.37G
UCI-Digit	Image	2000	10	3	83.9M
CCV	Video event	6773	20	6	1.04G
Flower102	Image	8189	102	4	1.98G

Table 1: Summaries of the public benchmark datasets.

- Single kernel methods. They include: average multiple kernel k-means (KKM$_a$) and single best kernel k-means (KKM$_b$).
- KKM-based MKC methods. They include: multiple kernel k-means (MKKM) (Huang, Chuang, and Chen 2012), robust multiple kernel k-means (RMKKM) (Du et al. 2015), localized multiple kernel k-means (LMKKM) (Gönen and Margolin 2014), multiple kernel k-means with matrix-induced regularization (MKKM-MR, MR for short) (Liu et al. 2016), local kernel alignment maximization (LKAM) (Li et al. 2016), multi-view clustering via late fusion alignment maximization (MVC-LFA, LFA for short) (Wang et al. 2019).
- SC-based MKC methods. They include: structure preserving multiple kernel clustering (SPMKC) (Ren and Sun 2020), consensus affinity graph learning (CAGL) (Ren et al. 2020c), and neighbor-kernel subspace segmentation (NKSS) (Zhou et al. 2019b).

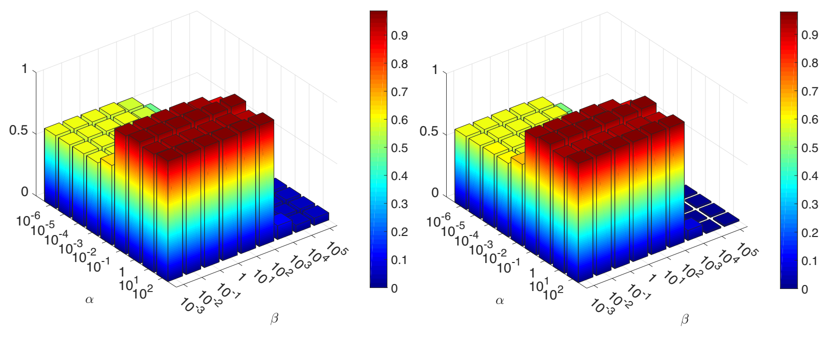
For fair comparison, the parameters of these comparsion methods are carefully tuned by following the recommended experimental settings provided by their respective authors. Moreover, three widely used metrics, clustering accuracy (ACC), normalized mutual information (NMI) and purity, are applied. For these metrics, the larger value indicates the better clustering performance.

Experimental Results

The average clustering results of 20 times independent experiments of all the comparison methods are presented in Table 2. Note here that the standard deviations of most nearly all experiments are less that 1%, so we do not show the standard deviations in Table 2. From the experimental results, we observe that, 1) the proposed KCGT is markedly better than all comparison methods, which obtains the best performance in terms of ACC, NMI and purity on all the datasets; 2) in most cases, the SC-based methods perform better than the KKM-based methods, since KKM-based methods work on the original data in kernel space, but SC-based methods work on the spectral embedding that roughly captures connectivity of data in kernel space; 3) compared with the KKM-based competitors, KCGT can exploit the high-order affinity from base kernels, such that the more important structure information of data is captured for clustering purpose; and 4) compared with the SC-based methods, KCGT has two highlights, one is that the affinity graph produced by

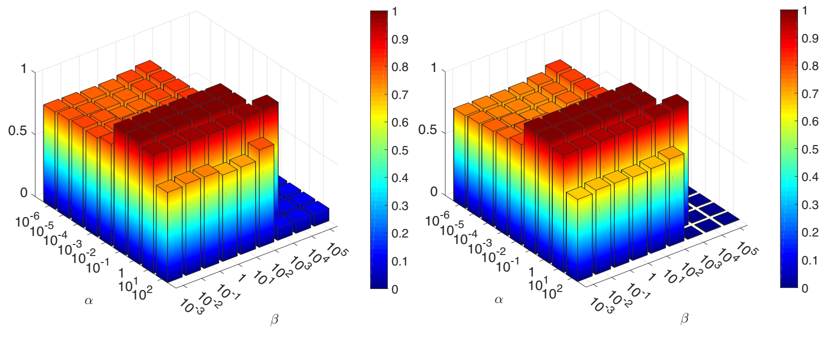
Dataset	Metrics	KKM$_a$	KKM$_b$	MKKM	RMKKM	LMKKM	MR	LKAM	LFA	SPMKC	CAGL	NKSS	KCGT
BBCSport2	ACC	66.18	76.65	66.18	63.79	66.18	66.18	60.48	77.45	79.04	86.03	95.04	**99.82**
	NMI	53.93	58.96	53.93	39.62	54.22	53.93	45.57	55.63	57.84	73.58	84.94	**99.29**
	Purity	77.21	79.41	77.21	67.83	77.39	77.21	70.77	76.27	79.04	86.21	95.04	**99.82**
ProteinFold	ACC	28.10	33.86	27.23	33.29	30.26	36.31	33.00	40.49	33.58	31.70	34.87	**39.63**
	NMI	38.53	42.03	37.16	40.17	40.26	45.89	41.25	48.96	42.21	42.02	46.93	**45.89**
	Purity	36.17	41.21	33.86	37.61	37.18	45.39	37.90	46.85	38.16	38.04	44.38	**41.35**
Flower17	ACC	74.34	70.81	75.51	71.99	74.56	73.90	66.18	61.16	76.53	75.44	78.31	**98.60**
	NMI	73.87	69.14	73.73	71.56	73.97	73.42	71.35	60.79	76.12	75.67	78.75	**97.88**
	Purity	74.56	70.96	75.51	73.68	74.78	74.19	69.19	62.32	77.30	76.86	80.00	**98.60**
Caltech101	ACC	35.29	32.09	34.77	32.03	34.44	37.91	31.90	38.39	39.17	36.72	36.01	**49.22**
	NMI	59.93	58.30	59.64	56.21	58.92	61.47	58.19	61.65	62.68	59.13	60.69	**68.58**
	Purity	37.52	33.92	37.25	33.79	35.88	39.74	34.25	40.28	41.52	37.91	39.54	**52.35**
Mfeat	ACC	71.95	80.20	59.60	65.30	71.35	83.20	80.90	95.15	97.14	91.76	98.00	**99.90**
	NMI	69.86	66.55	55.56	62.67	71.52	78.12	80.50	95.00	95.10	91.12	95.27	**99.73**
	Purity	71.95	80.20	62.55	66.25	72.05	83.20	84.10	95.05	96.74	91.20	98.00	**99.90**
UCI-Digit	ACC	88.75	75.65	47.00	44.00	89.90	90.40	95.50	88.60	95.62	95.45	96.95	**99.95**
	NMI	80.59	68.44	48.16	48.02	82.85	83.22	90.08	88.25	92.11	91.30	92.91	**99.86**
	Purity	88.75	76.30	49.70	47.20	89.90	90.40	95.50	88.90	96.04	95.27	96.95	**99.95**
CCV	ACC	19.71	23.82	19.95	17.57	21.84	24.20	19.46	27.56	30.10	24.42	25.22	**59.78**
	NMI	17.50	19.05	15.41	13.79	18.43	19.84	17.21	20.59	25.76	20.13	22.57	**64.97**
	Purity	24.45	25.29	24.58	21.32	25.79	26.58	24.13	30.71	32.14	26.98	28.41	**66.46**
Flower102	ACC	27.29	33.22	21.96	27.05	26.98	42.24	42.01	42.16	43.29	41.17	42.80	**79.47**
	NMI	46.32	49.08	42.30	46.99	45.92	57.57	58.19	60.48	59.52	58.55	59.69	**94.00**
	Purity	32.28	38.88	27.61	32.13	32.37	48.49	48.36	50.44	51.36	49.10	50.04	**86.64**

Table 2: Clustering results of the comparison methods. The best results are highlighted in bold.



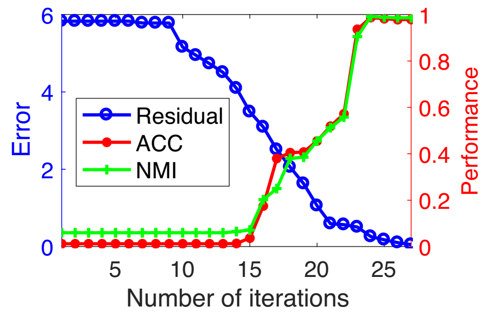
(a) ACC (b) NMI

Figure 3: Clustering performance (in terms of ACC and NMI) *w.r.t.* α and β on the Flower17 dataset.

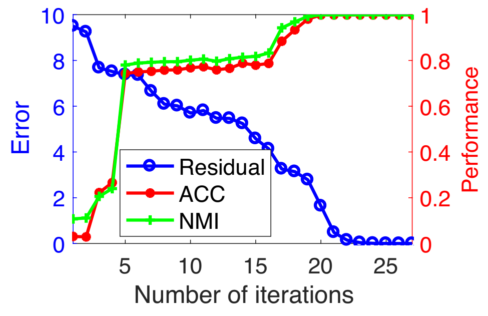


(a) ACC (b) NMI

Figure 4: Clustering performance (in terms of ACC and NMI) *w.r.t.* α and β on the UCI-Digit dataset.



(a) Flower17



(b) UCI-Digit

Figure 5: Convergence and clustering performance versus iteration on the Flower17 and UCI-Digit datasets.

KKG enjoys the valuable block diagonal and sparse property, the other is that KCGT constructs a high-order graph tensor to capture the three-order affinity, rather than the traditional two-order affinity.

Parameter Sensitivity and Convergence

In KCGT, there are two parameters α and β required to be set properly, which are used to control the sparsity of each candidate affinity graph $\mathbf{S}^{(k)}$ (*i.e.*, $\|\mathbf{S}^{(k)}\|_1, 1 \leq k \leq m$) and the effect of the high-order graph tensor $\boldsymbol{\mathcal{S}}$ (*i.e.*, $\|\boldsymbol{\mathcal{S}}\|_{\circledast}$), respectively. By leveraging a grid search technique, we tune α and β from the ranges $[10^{-5}, \cdots, 10^{2}]$ and $[10^{-3}, \cdots, 10^{5}]$ with step size 10, respectively. Take the Flower17 and UCI-Digit datasets for example. As shown in Figs. 3 and 4, the proposed KCGT method works well for a wide range of α and β values, and can be easily tuned. Usually, we need a relatively large α and a relatively small β. The results of other datasets are similar and are omitted due to space limit.

To demonstrate the convergence of Algorithm 1 experimentally, we record the convergence and clustering performance at each iteration on the the Flower17 and UCI-Digit datasets. As illustrated in Fig. 5, we have two observations: 1) the residual curves (*i.e.*, *error w.r.t.* iterator) converge rapidly till to the stable point, some of which meet the stop criterion (*i.e.*, Eq. 19) within 25 epochs; and 2) the clustering performance (*i.e.*, ACC and NMI *w.r.t.* iterator) of the proposed method gradually increases until the residual curves become stable. This proves the fast and stable convergence property of the proposed algorithm.

Conclusion

This paper proposes a new graph learning paradigm (*i.e.*, KKG) and a novel MKC method (*i.e.*, KCGT). Specifically, KKG devotes learning a desired affinity graph in kernel space by coupling the kernel k-means, so as to obtain multiple affinity graphs from base kernels. Meanwhile, KCGT can exploit the high-order information of these candidate graphs to learn a high-quality consensus graph for spectral clustering purpose. Theoretically, KCGT can better address the problem of how to choose the right kernel function and pre-define its parameters optimally for a specific non-linear dataset. Experimentally, KCGT shows clearly superior clustering performance on benchmark datasets, in comparison with state-of-the-art methods.

Acknowledgments

This work was supported by the Sichuan Science and Technology Program (project no. 2021YJ0083), the National Natural Science Foundation of China (project nos. 61673220 and 61906091), the Zhejiang Provincial Natural Science Foundation of China (project no. LGF21F020003), and the Natural Science Foundation of Chongqing (project no. cstc2020jcyjmsxmX0473).

References

Boyd, S.; Boyd, S. P.; and Vandenberghe, L. 2004. *Convex optimization*. Cambridge university press.

Du, L.; Zhou, P.; Shi, L.; Wang, H.; Fan, M.; Wang, W.; and Shen, Y.-D. 2015. Robust multiple kernel k-means using L21-norm. In *Proceedings of the Twenty-Fourth International Joint Conference on Artificial Intelligence*, 3476–3482.

Gönen, M.; and Margolin, A. A. 2014. Localized data fusion for kernel k-means clustering with application to cancer biology. *Advances in Neural Information Processing Systems* 27: 1305–1313.

Huang, H.-C.; Chuang, Y.-Y.; and Chen, C.-S. 2012. Multiple kernel fuzzy clustering. *IEEE Transactions on Fuzzy Systems* 20(1): 120–134.

Huang, J.; Nie, F.; and Huang, H. 2015. A new simplex sparse learning model to measure data similarity for clustering. In *The Twenty-Fourth International Joint Conference on Artificial Intelligence*, 3569–3575.

Kang, Z.; Lu, X.; Yi, J.; and Xu, Z. 2018. Self-weighted multiple kernel learning for graph-based clustering and semi-supervised classification. In *Proceedings of the Twenty-Seventh International Joint Conference on Artificial Intelligence*, 2312–2318.

Li, M.; Liu, X.; Wang, L.; Dou, Y.; Yin, J.; and Zhu, E. 2016. Multiple kernel clustering with local kernel alignment maximization. In *Proceedings of the Twenty-Fifth International Joint Conference on Artificial Intelligence*, 1704–1710.

Liu, J.; Liu, X.; Xiong, J.; Liao, Q.; Zhou, S.; Wang, S.; and Yang, Y. 2020a. Optimal neighborhood multiple kernel clustering with adaptive local kernels. *IEEE Transactions on Knowledge and Data Engineering* .

Liu, X.; Dou, Y.; Yin, J.; Wang, L.; and Zhu, E. 2016. Multiple kernel k-means clustering with matrix-induced regularization. In *Proceedings of the Thirtieth AAAI Conference on Artificial Intelligence*, 1888–1894.

Liu, X.; Zhou, S.; Wang, Y.; Li, M.; Dou, Y.; Zhu, E.; Yin, J.; and Li, H. 2017. Optimal neighborhood kernel clustering with multiple kernels. In *Proceedings of the Thirty-First AAAI Conference on Artificial Intelligence*, 2266–2272.

Liu, X.; Zhu, X.; Li, M.; Wang, L.; Zhu, E.; Liu, T.; Kloft, M.; Shen, D.; Yin, J.; and Gao, W. 2020b. Multiple kernel k-means with incomplete kernels. *IEEE Transactions on Pattern Analysis and Machine Intelligence* 42(5): 1191–1204.

Lu, C.; Feng, J.; Lin, Z.; Mei, T.; and Yan, S. 2019. Subspace clustering by block diagonal representation. *IEEE Transactions on Pattern Analysis and Machine Intelligence* 41(2): 487–501.

Nie, F.; Wang, X.; Jordan, M. I.; and Huang, H. 2016. The constrained laplacian rank algorithm for graph-based clustering. In *Thirtieth AAAI Conference on Artificial Intelligence*, 1969–1976.

Ren, Z.; Li, H.; Yang, C.; and Sun, Q. 2019. Multiple kernel subspace clustering with local structural graph and low-rank consensus kernel learning. *Knowledge-Based Systems* 105040.

Ren, Z.; Mukherjee, M.; Lloret, J.; and Venu, P. 2020a. Multiple kernel driven clustering with locally consistent and selfish graph in industrial IoT. *IEEE Transactions on Industrial Informatics* 17(4): 2956–2963.

Ren, Z.; and Sun, Q. 2020. Simultaneous global and local graph structure preserving for multiple kernel clustering. *IEEE Transactions on Neural Networks and Learning Systems* .

Ren, Z.; Sun, Q.; Wu, B.; Zhang, X.; and Yan, W. 2020b. Learning latent low-rank and sparse embedding for robust image feature extraction. *IEEE Transactions on Image Processing* 29(1): 2094–2107.

Ren, Z.; Yang, S. X.; Sun, Q.; and Wang, T. 2020c. Consensus affinity graph learning for multiple kernel clustering. *IEEE Transactions on Cybernetics* .

Wang, S.; Liu, X.; Zhu, E.; Tang, C.; Liu, J.; Hu, J.; Xia, J.; and Yin, J. 2019. Multi-view clustering via late fusion alignment maximization. In *Twenty-Eighth International Joint Conference on Artificial Intelligence*, 3778–3784.

Wu, J.; Xie, X.; Nie, L.; Lin, Z.; and Zha, H. 2020. Unified graph and low-rank tensor learning for multi-view clustering. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 34, 6388–6395.

Yang, C.; Ren, Z.; Sun, Q.; Wu, M.; Yin, M.; and Sun, Y. 2019. Joint correntropy metric weighting and block diagonal regularizer for robust multiple kernel subspace clustering. *Information Sciences* 500: 48–66.

Zhang, C.; Fu, H.; Liu, S.; Liu, G.; and Cao, X. 2015. Low-rank tensor constrained multiview subspace clustering. In *Proceedings of the IEEE International Conference on Computer Vision*, 1582–1590.

Zhou, P.; Lu, C.; Feng, J.; Lin, Z.; and Yan, S. 2019a. Tensor low-rank representation for data recovery and clustering. *IEEE Transactions on Pattern Analysis and Machine Intelligence* .

Zhou, S.; Liu, X.; Li, M.; Zhu, E.; Liu, L.; Zhang, C.; and Yin, J. 2019b. Multiple kernel clustering with neighbor-kernel subspace segmentation. *IEEE transactions on neural networks and learning systems* 31(4): 1351–1362.