Ren, Z.; Mukherjee, M.; Bennis, M.; Lloret, J. (2021). Multikernel Clustering via Non-Negative Matrix Factorization Tailored Graph Tensor Over Distributed Networks. IEEE Journal on Selected Areas in Communications. 39(7):1946-1956. https://doi.org/10.1109/JSAC.2020.3041396

Multi-kernel Clustering via Non-negative Matrix Factorization Tailored Graph Tensor over Distributed Networks

Zhenwen Ren, *Member, IEEE,* Mithun Mukherjee, *Senior Member, IEEE,* Mehdi Bennis, *Senior Member, IEEE*, and Jaime Lloret, *Senior Member, IEEE*

***Abstract*—Next-generation wireless networks are witnessing an increasing number of clustering applications, and produce a large amount of non-linear and unlabeled data. In some degree, single kernel methods face the challenging problem of kernel choice. To overcome this problem for non-linear data clustering, multiple kernel graph-based clustering (MKGC) has attracted intense attention in recent years. However, existing MKGC methods suffer from two common problems: (1) they mainly aim to learn a consensus kernel from multiple candidate kernels, slight affinity graph learning, such that cannot fully exploit the underlying graph structure of non-linear data; (2) they disregard the high-order correlations between all base kernels, which cannot fully capture the consistent and complementary information of all kernels. In this paper, we propose a novel non-negative matrix factorization (NMF) tailored graph tensor MKGC method for non-linear data clustering, namely TMKGC. Specifically, TMKGC integrates NMF and graph learning together in kernel space so as to learn multiple candidate affinity graphs. Afterwards, the high-order structure information of all candidate graphs is captured in a 3-order tensor kernel space by introducing tensor singular value decomposition based tensor nuclear norm, such that an optimal affinity graph can be obtained subsequently. Based on the alternating direction method of multipliers, the effective local and distributed solvers are elaborated to solve the proposed objective function. Extensive experiments have demonstrated the superiority of TMKGC compared to the state-of-the-art MKGC methods.**

***Index Terms*—Multiple kernel clustering, intelligent network, distributed computation, non-negative matrix factorization, tensor learning.**

I. Introduction

DUE to the proliferation of the Internet of Things (IoT) devices, a massive amount of data is being generated at the network edge [1], [2]. At the same time, machine learning over wireless networks [3]–[5] exploits data analytics to improve network optimization. Several distributed machine learning frameworks and algorithms [6], [7] are taking a step forward toward intelligent communication systems. Usually, obtaining the labeled data is not always feasible whereas abundant unlabeled data are available and easier to collect. Clustering is essential to handle unlabeled data and plays a central role in data mining, whose goal is to partition unlabeled data points into clusters [8], [9]. Along with the remarkable growth in data traffic over the past years, *how to effectively handle non-linear data* is a daunting task [10], [11]. Traditional single kernel methods can solve this problem to some extent, nevertheless, they have a weak ability to exploit the underlying relationships of non-linear data, such as those generated by Internet of Things (IoT) sensor and surveillance video data. The main reasons are that (1) each kernel function has a different representation capability; (2) most suitable kernels and the associated parameters for a specific dataset are difficult to select. In this paper, to tackle the challenging problem of non-linear data clustering, we seamlessly integrate graph-based clustering (GBC) [12] and multiple kernel learning (MKL) [8], [13] into a unified objective function. This learning paradigm is dubbed as multiple kernel graph-based clustering (MKGC).

In recent years, GBC methods have been widely investigated due to their effectiveness in capturing the complex graph structure hidden in data. Usually, GBC consists of two continuous steps [12], [14]: (1) constructing an affinity graph based on graphical representations of the relationships among data points, and (2) applying spectral clustering algorithms (*e.g.*, normalized cut and ratio cut) to obtain clustering assignments. Constructing a high-quality affinity graph plays a crucial role in GBC [15]. Overall, the existing GBC methods can be roughly divided into three main categories. The first one uses binary similarity, cosine similarity, or gaussian kernel similarity to construct a predefined similarity graph as affinity graph [16]. The second one is based on the self-expressiveness subspace learning (SESL) [15], which reconstructs every data point by a linear combination of all other data points and produces a coefficient matrix as an affinity graph. The third one is adaptive neighbors graph learning (ANGL) [17], which assigns a probability for each sample as the neighborhood of another sample to construct an affinity graph. Accordingly, homogeneous data points have high affinity values, while heterogeneous data points have low affinity values. On the other hand, MKL aims to learn a consensus kernel from multiple base kernels, which cannot only effectively handle non-linear data but also alleviate the curse of kernel choice [13]. Overall,

Z. Ren is with the Department of National Defence Science and Technology, Southwest University of Science and Technology, Mianyang, China, 621010, and the Department of Computer Science, Nanjing University of Science and Technology, Nanjing, China, 210094 (e-mail: rzw@njust.edu.cn).

M. Mukherjee is with the Department of Electronic and Computer Engineering, Nanjing University of Information Science and Technology, Nanjing 210044, China (e-mail: m.mukherjee@ieee.org).

M. Bennis is with the Centre of Wireless Communication, University of Oulu, Finland (e-mail: mehdi.bennis@oulu.fi).

J. Lloret is with the Instituto de Investigación para la Gestión Integrada de Zonas Costeras (IGIC), Universitat Politecnica de Valencia, 46022 Valencia, Spain and School of Computing and Digital Technologies, Staffordshire University, Stoke, UK (e-mail: jlloret@dcom.upv.es).

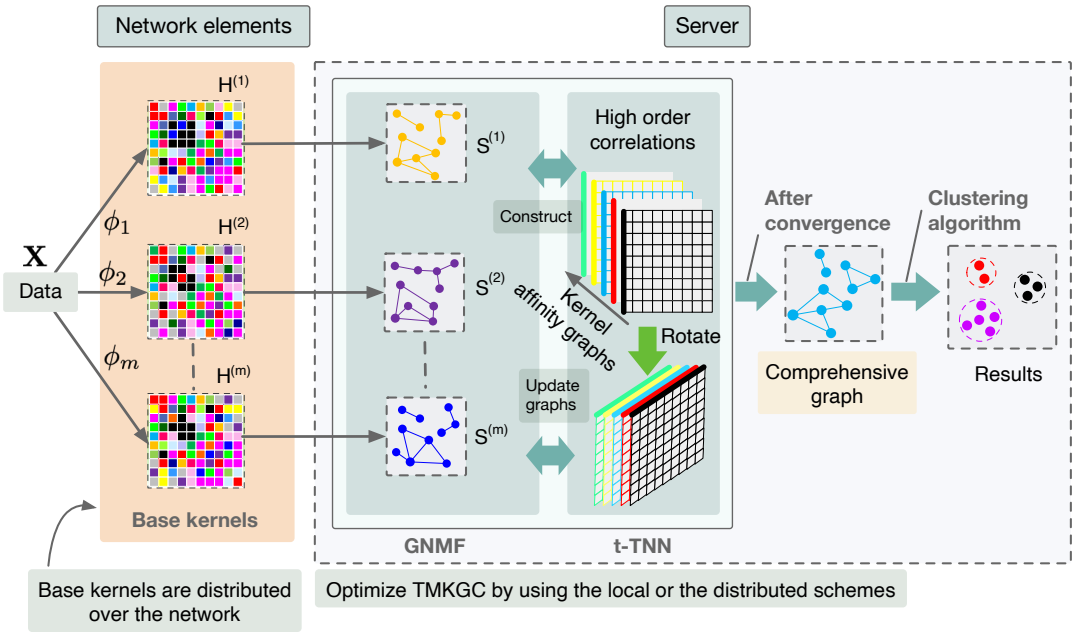


Fig. 1. Framework of the proposed method.

there are three strategies for the comprehensive utilization of the given multiple base kernels, *i.e.*, (1) using linearly kernel combination to generate a consensus kernel [18], [19]; (2) using non-linearly kernel combination to generate a consensus kernel [13], [20]–[22]; (3) changing the goal of consensus kernel learning to consensus graph learning [12], [23]. In practice, MKL has been widely used to handle non-linear data in various applications [8], [22].

Recently, based on GBC and MKC, several state-of-the-art MKGC methods are gradually emerging [13], [19]–[28]. These methods typically work as follows: (1) constructing multiple base kernel Gram matrices by relying on the given multiple base kernels, (2) learning a consensus kernel and an affinity graph, while the focus is on consensus kernel learning, and (3) producing the clustering results on the learned affinity graph. Despite of their success in practice, these existing MKGC methods still suffer from the following drawbacks: (1) they usually pay more attention to consensus kernel rather than affinity graph, this violates the intention of graph-based clustering aiming to learn an optimal affinity graph, and (2) they essentially ignore the high-order correlations (*e.g.*, tensor) underlying the multiple base kernels, so they may not fully explore the consistent and complementary information of the given multiple kernels.

To tackle the aforementioned problems, in this paper, we propose a novel MKGC method, namely *tensor multiple kernel graph-based clustering* (TMKGC), for handling non-linear data clustering. Specifically, by leveraging non-negative matrix factorization (NMF) with multiple kernel Gram matrices, we first learn multiple candidate affinity graphs in kernel space. Moreover, we integrate these candidate graphs into a 3-order graph tensor, and then rotate this tensor for investigating the correlations of these graphs and reducing the computation complexity simultaneously. Finally, the tensor Singular Value Decomposition (t-SVD) based tensor nuclear norm (t-TNN) is employed to exploit the consistency and complementary of candidate kernel graphs in the kernelized tensor space, such that the essential tensor with high-order correlations can be obtained. Figure 1 shows the proposed framework. In summary, this paper has the following contributions:

- Unlike the existing SESL and ANGL-based graph learning paradigm, we propose a novel kernel graph learning paradigm based on NMF (GNMF for short). To the best of our knowledge, it is the first method to learn an affinity graph directly based on NMF in kernel space.
- By folding the candidate affinity graphs produced by GNMF as a graph tensor, the introduced t-TNN can explore the consistency and complementary information of the NMF tailored graph tensor. With the ability to capture the high-order correlations of data in the level of tensor, the multiple base kernels can be sufficiently utilized.
- We integrate the proposed graph learning paradigm and the t-TNN into a unified objective function. Since it is concise and easy to follow, we design local and distributed solvers to solve it efficiently, where the candidate affinity graphs and the graph tensor are mutually reinforced until the optimal tensor is obtained.
- Compared with state-of-the-art MKGC methods, the superiority of TMKGC is demonstrated by conducting extensive experiments.

The remainder of the paper is structured as follows. Section II briefly reviews the NMF and the state-of-the-art MKGC methods. Section III presents the TMKGC method, solver, computational complexity, and convergence. Then, the experimental results are shown in Section IV. Finally, Section V concludes our work.

II. Related Work

In this section, we review NMF and some related MKGC methods for comprehensive understanding of our method.

A. Nonnegative Matrix Factorization (NMF)

Given a nonnegative data matrix $\mathbf{X} = [\mathbf{x}_1, \mathbf{x}_2, \cdots, \mathbf{x}_n] \in \mathbb{R}^{d\times n}$, where $\mathbf{x}_i$, n, and d indicate i-th sample, sample size, and sample dimensionality, respectively. The goal of NMF is to find two nonnegative matrices, $\mathbf{F}_+ \in \mathbb{R}^{d\times r}$ and $\mathbf{P}_+ \in \mathbb{R}^{r\times n}$, $r \ll n$, by solving the following problem

$$\min_{\mathbf{F}_+,\mathbf{P}_+} \|\mathbf{X} - \mathbf{F}_+\mathbf{P}_+\|_F^2 , \tag{1}$$

where $\mathbf{F}_+ \geqslant 0$ and $\mathbf{P}_+ \geqslant 0$ usually demonstrate the basis matrix and representation matrix, respectively. Note that (1) cannot handle nonlinear data directly. To tackle this tricky problem, Concept Factorization (CF) is widely used in many references [29]. However, NMF and CF can usually be applied to linear kernel, but not for MKL; moreover, they cannot learn an affinity graph directly, which hampers their applications on graph-based learning tasks. In this paper, we propose a novel NMF based graph learning paradigm to learn an affinity graph directly.

B. Multiple Kernel Graph-based Clustering (MKGC)

MKGC has been extensively studied during the recent years [13], [19]–[28]. According to the integration strategy of multiple kernels (see Section I), we briefly review some related work.

(1) **Linearly kernel combination:** The linearly kernel combination scheme aims to learn a consensus kernel by linearly combining the base kernels [27]. For example, multiple kernel fuzzy k-means (MKKM) [24] extends the fuzzy k-means algorithm into multiple kernel scenarios. Then, robust multiple kernel k-means (RMKKM) [19] extended from MKKM uncovers the cluster membership, the optimal combination of multiple base kernels, and the best clustering label simultaneously. Inspired by MKKM, multi-view clustering via late fusion alignment maximization (MVCLFA) [25] performs the weighted strategy to learn the consensus partition so that the computational complexity is significantly reduced. Affinity aggregation for spectral clustering (AASC) [26] can be considered as a multiple kernel clustering version of spectral clustering, and then extends spectral clustering to the scenarios with multiple affinities available. Spectral clustering with multiple kernels (SCMK) [27] uses the convex combination of multiple base kernels to learn a better consensus kernel by using the fact that the optimal consensus kernel is a linear combination of base kernels. Overall, this scheme is easily to accomplish; however, the solution set of the learned consensus kernel is limited.

(2) **Non-linearly kernel combination:** The non-linearly kernel combination scheme aims to learn a consensus kernel by assuming that the consensus kernel is the neighbor of each base kernel (*i.e.*, neighborhood kernel learning [13]). For instance, self-weighted multiple kernel learning (SMKL) [13] assumes that the consensus kernel is a neighborhood of multiple base kernels, then proposes a new MKL model to fuse the given multiple kernels. By considering the neighborhood structure among base kernels, neighbor-kernel-based MKL [28] has been proposed. By considering the low-rank property of the samples, low-rank kernel learning graph-based clustering (LKGr) [20] has been proposed to impose a low-rank constraint on kernel matrix. Robust multiple kernel subspace clustering (JMKSC) [22] uses the block diagonal regularizer and the self-expressiveness to learn an affinity matrix with optimal block diagonal property. Local structural graph and low-rank consensus multiple kernel learning (LLMKL) [21] integrates the MKL, the global and local structure, as well as self-expressiveness property in a unified model. Overall, although this scheme has a wide solution set, the consensus kernel cannot capture the structure information of base kernels.

(3) **Kernel graph fusion:** Kernel graph fusion scheme aims to learn a consensus affinity graph by using graph learning paradigms, and then employs spectral clustering to segment clusters [12]. For example, structure preserving multiple kernel clustering (SPMKC) [23] integrates the global and local structure preserving by simultaneously leveraging both SESL and ANGL with a kernel affine weight strategy into a unified MKL optimization model. Consensus affinity graph learning (CAGL) [12] learns a consensus graph directly for MKGC, and has produced promising results. Overall, this scheme can capture the structure information of base kernels, which is very important to unsupervised clustering tasks.

III. Proposed Method

In this section, we elaborate our proposed TMKGC method.

A. Notations

TABLE I
Notations used throughout the paper

Notation	Definition	Notation	Definition
$\mathbf{M}$	A matrix	$\mathbf{m}$	A vector
m	A scalar	$\mathbf{M}^{(k)}$	The k-th matrix of a cell
$\boldsymbol{\mathcal{M}}$	A tensor		
$\mathbf{M}_{ijk}$	The (i,j,k)-th entry	$\boldsymbol{\mathcal{M}}(:,j,k)$	The mode-1 fiber
$\boldsymbol{\mathcal{M}}(i,:,k)$	The mode-2 fiber	$\boldsymbol{\mathcal{M}}(i,j,:)$	The mode-3 fiber
$\boldsymbol{\mathcal{M}}(i,:,:)$	The i-th horizontal slice	$\boldsymbol{\mathcal{M}}(:,j,:)$	The j-th lateral slice
$\boldsymbol{\mathcal{M}}(:,:,k)$	The k-th frontal slice	$\boldsymbol{\mathcal{M}}^{(k)}$	The k-th frontal slice

In this paper, 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.*, $\boldsymbol{\mathcal{M}}$. Table II summarizes the notations used in this paper.

B. Preliminaries

In this section, some operators and definitions related to tensor are introduced to help understand tensor better [30].

First, we give some operators about 3-order tensor. For a 3-order tensor $\boldsymbol{\mathcal{M}} \in \mathbb{R}^{n_1\times n_2\times n_3}$, $\boldsymbol{\mathcal{M}}_f = \texttt{fft}(\boldsymbol{\mathcal{M}}, [\,], 3)$ 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 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, *i.e.*,

$$\texttt{bdiag}(\boldsymbol{\mathcal{M}}) = \begin{bmatrix} \mathbf{M}^{(1)} & & \cdots & \mathbf{0} \\ & \mathbf{M}^{(2)} & & \\ \vdots & & \ddots & \vdots \\ \mathbf{0} & \cdots & & \mathbf{M}^{(n_3)} \end{bmatrix}$$

$$\texttt{bcirc}(\boldsymbol{\mathcal{M}}) = \begin{bmatrix} \mathbf{M}^{(1)} & \mathbf{M}^{(n_3)} & \cdots & \mathbf{M}^{(2)} \\ \mathbf{M}^{(2)} & \mathbf{M}^{(1)} & \cdots & \mathbf{M}^{(3)} \\ \vdots & \ddots & \ddots & \vdots \\ \mathbf{M}^{(n_3)} & \mathbf{M}^{(n_3-1)} & \cdots & \mathbf{M}^{(1)} \end{bmatrix}.$$

Then, we present some some definitions about 3-order tensor as following.

Definition 1 (T-Product). The t-product between two 3-order 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.*,

$$\boldsymbol{\mathcal{M}} * \boldsymbol{\mathcal{N}} = \texttt{fold}(\texttt{bcirc}(\boldsymbol{\mathcal{M}})\, \texttt{bvec}(\boldsymbol{\mathcal{N}}))\,. \tag{2}$$

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 the identity matrix with size $n_1 \times n_1$ while the others frontal slices are zeros.

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

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

$$\boldsymbol{\mathcal{F}}^\intercal * \boldsymbol{\mathcal{F}} = \boldsymbol{\mathcal{F}} * \boldsymbol{\mathcal{F}}^\intercal = \boldsymbol{\mathcal{I}}\,. \tag{3}$$

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 $\boldsymbol{\mathcal{M}} \in \mathbb{R}^{n_1 \times n_2 \times n_3}$ is defined as

$$\boldsymbol{\mathcal{M}} = \boldsymbol{\mathcal{U}} * \boldsymbol{\mathcal{G}} * \boldsymbol{\mathcal{V}}^\intercal\,. \tag{4}$$

where $\boldsymbol{\mathcal{G}} \in \mathbb{R}^{n_1 \times n_2 \times n_3}$ is a f-diagonal tensor, and $\boldsymbol{\mathcal{U}} \in \mathbb{R}^{n_1 \times n_1 \times n_3}$ and $\boldsymbol{\mathcal{V}} \in \mathbb{R}^{n_2 \times n_2 \times n_3}$ are orthogonal tensors. To facilitate understanding, t-SVD operator is shown in Fig. 2.

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

$$\|\boldsymbol{\mathcal{M}}\|_\circledast = \sum_{k=1}^{n_3} \left\|\boldsymbol{\mathcal{M}}_f^{(k)}\right\|_* = \sum_{i=1}^{\min(n_1,n_2)} \sum_{k=1}^{n_3} \left|\boldsymbol{\mathcal{G}}_f^{(k)}(i,i)\right|\,. \tag{5}$$

According to the frontal slices of $\{\boldsymbol{\mathcal{M}}_f^{(k)}\}_{k=1}^{n_3}$, we employ the SVD, $\boldsymbol{\mathcal{M}}_f^{(k)} = \boldsymbol{\mathcal{U}}_f^{(k)} \boldsymbol{\mathcal{G}}_f^{(k)} \boldsymbol{\mathcal{V}}_f^{(k)\intercal}$, to compute the corresponding $\{\boldsymbol{\mathcal{G}}_f^{(k)}\}_{k=1}^{n_3}$. It has been demonstrated that the t-TNN can exploit the structural information of a tensor better than the unfolding-based tensor nuclear norm [31].

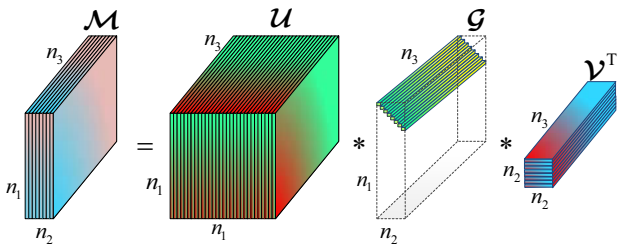


Fig. 2. The t-SVD operator of the tensor $\boldsymbol{\mathcal{M}} \in \mathbb{R}^{n_1 \times n_2 \times n_3}$.

C. Proposed TMKGC Method

In MKL scenarios, give a data matrix $\mathbf{X} = [\mathbf{x}_1, \cdots, \mathbf{x}_n]$ consisting of n samples drawn from c crispy clusters, a kernel pool consisting of m base kernels $\{\mathbf{H}^{(k)}\}_{k=1}^m$ is calculated on $\mathbf{X}$ in advance.

1) Graph learning paradigm based on NMF: As aforementioned in section II-A, NMF and its extended version CF cannot learn a graph directly and cannot be used to MKL scenarios. Without loss of generality, we consider a mapping $\mathbf{x}_i \to \phi(\mathbf{x}_i)$, or $\mathbf{X} \to \phi(\mathbf{X}) = (\phi(\mathbf{x}_1), \cdots, \phi(\mathbf{x}_n))$, where ϕ is a kernel function. By this means, the samples of non-linearly separable in the original space become linearly separable in the reproducing kernel Hilbert space [32].

Supposing the clustering indicator matrix $\mathbf{P} \in \mathbb{R}^{n \times c}$ denotes the posterior cluster probabilities, and the corresponding cluster centroids $\mathbf{F}$ requires each column to be convex combinations of input data. $\mathbf{F}$ can be computed as $\mathbf{f}_i = \phi(\mathbf{X})\mathbf{p}_i / n_i$, *i.e.*, $\mathbf{F} = \phi(\mathbf{X})\mathbf{P}\mathbf{D}_n^{-1}$, where n_i is the number of samples of cluster, i, and $\mathbf{D}_n = \texttt{diag}\,(n_1, \cdots, n_c)$. Therefore, the pair of property: (1) $\mathbf{F}$ encodes centroids, and (2) $\mathbf{P}$ encodes the posterior probabilities, induces a factorization $\phi(\mathbf{X}) \approx \phi(\mathbf{X})\mathbf{P}\mathbf{D}_n^{-1}\mathbf{P}^\intercal$. Absorbing $\mathbf{D}_n^{-1}$ into $\mathbf{P}$ and letting $\mathbf{P}_+ = \texttt{max}(\mathbf{P}\sqrt{\mathbf{D}_n^{-1}}, 0)$, we then have

$$\phi(\mathbf{X}) \approx \phi(\mathbf{X})\mathbf{P}_+\mathbf{P}_+^\intercal\,, \tag{6}$$

where $\mathbf{P}_+$ is the expected clustering indicator matrix.

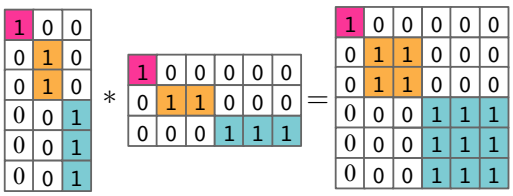
It is easy to see that we can obtain the indicator matrix $\mathbf{P}_+$ via the following optimization problem, *i.e.*,

$$\min_{\mathbf{P}_+} \|\phi(\mathbf{X}) - \phi(\mathbf{X})\mathbf{P}_+\mathbf{P}_+^\intercal\|_F^2 \tag{7}$$

$$= \texttt{Tr}(\phi(\mathbf{X})^\intercal\phi(\mathbf{X}) - 2\mathbf{P}_+^\intercal\phi(\mathbf{X})^\intercal\phi(\mathbf{X})\mathbf{P}_+ \tag{8}$$

$$+ \mathbf{P}_+^\intercal\phi(\mathbf{X})^\intercal\phi(\mathbf{X})\mathbf{P}_+\mathbf{P}_+^\intercal\mathbf{P}_+)\,. \tag{9}$$

Now, we aim to pursue a high-quality affinity graph for clustering purpose. In an ideal case, the block diagonal property of affinity graph plays a significant role for graph-based learning task [15] and is explicitly pursued in recent GMKC methods [12], [13], [21], [22]. Theoretically, since $\mathbf{P}_+$ is the clustering indicator matrix, so $\mathbf{P}_+\mathbf{P}_+^\intercal$ is a strictly block diagonal matrix. Accordingly, we can learn a symmetric affinity graph $\mathbf{S} \in \mathbb{R}^{n \times n}$ that best approximates $\mathbf{P}_+\mathbf{P}_+^\intercal$, *i.e.*, $\mathbf{S} = \mathbf{P}_+\mathbf{P}_+^\intercal$. Here, we give a toy example to show the connections between clustering indicator matrix $\mathbf{P}_+$ and affinity graph $\mathbf{S}$ mathematically. As shown in Fig. 3, there are 1, 2, 3 samples in 3 different clusters, respectively, and



$$\mathbf{P}_+\mathbf{P}_+^\intercal = \mathbf{S}$$

Fig. 3. The relation between indicator $\mathbf{P}_+$ and graph $\mathbf{S}$.

the affinity graph $\mathbf{S}$ produced by $\mathbf{P}_+\mathbf{P}_+^\intercal$ has obvious block diagonal property for clustering purpose.

Furthermore, due to $\mathbf{P}_+^\intercal\mathbf{P}_+ = \mathbf{I}$, we have $\|\mathbf{S}\|_F^2 = \|\mathbf{P}_+\mathbf{P}_+^\intercal\|_F^2 = \mathtt{Tr}(\mathbf{P}_+\mathbf{P}_+^\intercal\mathbf{P}_+\mathbf{P}_+^\intercal) = \mathtt{Tr}(\mathbf{I}) = c$. Hereto, the high-quality affinity graph $\mathbf{S}$ can be obtained according to the kernel trick $\mathbf{H} = \phi(\mathbf{X})^\intercal\phi(\mathbf{X})$, *i.e.*, (9) can be rewritten as

$$\begin{aligned}\min_{\mathbf{S}}\ & \mathtt{Tr}(\mathbf{H} - 2\mathbf{S}^\intercal\mathbf{H} + \mathbf{S}^\intercal\mathbf{H}\mathbf{S}) \\ \text{s.t. }& \|\mathbf{S}\|_F^2 = c, \mathbf{S}^\intercal = \mathbf{S}, \mathbf{S} \geq 0\,.\end{aligned} \tag{10}$$

Based on the aforementioned analysis, we bridge the relations between NMF and affinity graph learning dexterously. We dub this graph learning paradigm as graph NMF (GNMF). In MKL scenarios, since a kernel pool consisting of m base kernels $\{\mathbf{H}^{(k)}\}_{k=1}^m$ in given in advance, we can obtain m candidate affinity kernel graphs $\{\mathbf{S}^{(k)}\}_{k=1}^m$ by leveraging GNMF accordingly.

2) NMF Tailored Graph Tensor: With m candidate kernel affinity graphs $\{\mathbf{S}^{(k)}\}_{k=1}^m$ at hand, we strongly desire that an intrinsic affinity graph can be learned to capture both the consistent and complementary information among these m graphs. To this end, we stack the m kernel graphs into a tensor $\boldsymbol{\mathcal{S}}^* \in \mathbb{R}^{n\times n\times m}$ (*i.e.*, $\boldsymbol{\mathcal{S}}^* = \mathtt{bvfold}([\mathbf{S}^{(1)};\cdots;\mathbf{S}^{(m)}])$), and then rotate $\boldsymbol{\mathcal{S}}^*$ to $\boldsymbol{\mathcal{S}} \in \mathbb{R}^{n\times m\times n}$ (*i.e.*, $\mathtt{rotate}(\boldsymbol{\mathcal{S}}^*)$), as illustrated in Fig 4. By this means, this can better explore the correlations between these candidate graphs in kernelized tensor space. According to (5), t-SVD performs FFT along the third dimension of $\boldsymbol{\mathcal{S}}$, so computing SVD in each frontal slice with the information of these kernel graphs is more meaningful. Meanwhile, instead of using $\boldsymbol{\mathcal{S}}^*$, the computational complexity is largely reduced (see Section III-F).

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 sample is usually much bigger than the number of clusters, thus the learned tensor $\boldsymbol{\mathcal{S}}$ should enjoy the low-rank property. In this paper, a t-TNN term, $\|\boldsymbol{\mathcal{S}}\|_\circledast$, is used to regularize $\boldsymbol{\mathcal{S}}$ as a constraint of the intrinsic low-rank tensor, and then we have

$$\begin{aligned}\min_{\mathbf{S}^{(k)}} \sum_{k=1}^m \mathtt{Tr}\left(\mathbf{H}^{(k)} - 2(\mathbf{S}^{(k)})^\intercal\mathbf{H}^{(k)} + (\mathbf{S}^{(k)})^\intercal\mathbf{H}^{(k)}\mathbf{S}^{(k)}\right) + \beta\|\boldsymbol{\mathcal{S}}\|_\circledast \\ \text{s.t.}\quad \forall k, \|\mathbf{S}^{(k)}\|_F^2 = c, (\mathbf{S}^{(k)})^\intercal = \mathbf{S}^{(k)}, \mathbf{S}^{(k)} \geq 0, \\ \boldsymbol{\mathcal{S}} = \mathtt{rotate}\left(\mathtt{bvfold}([\mathbf{S}^{(1)};\cdots;\mathbf{S}^{(m)}])\right)\,.\end{aligned} \tag{11}$$

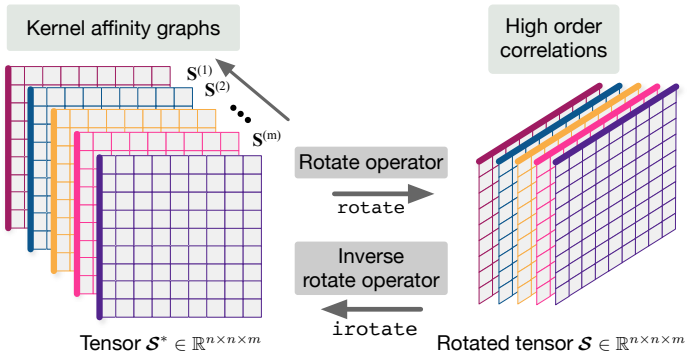


Fig. 4. The rotated affinity graph tensor in our TMKGC. Note that `rotate` and `irotate` are two shift functions.

Since $\|\mathbf{S}^{(k)}\|_F^2 = c$ is hard to solve, we relax this equality constraint according to Lagrangian relaxation, which approximates a difficult problem of constrained optimization by a simpler problem. If α is large enough, we can append $\alpha(\|\mathbf{S}^{(k)}\|_F^2 - c)$ into (11). Therefore, the final objective function can be upgraded to

$$\begin{aligned}\min_{\mathbf{S}^{(k)}} \sum_{k=1}^m \mathtt{Tr}\left(\mathbf{H}^{(k)} - 2(\mathbf{S}^{(k)})^\intercal\mathbf{H}^{(k)} + (\mathbf{S}^{(k)})^\intercal\mathbf{H}^{(k)}\mathbf{S}^{(k)}\right) \\ + \alpha\left\|\mathbf{S}^{(k)}\right\|_F^2 + \beta\|\boldsymbol{\mathcal{S}}\|_\circledast \\ \text{s.t. } \forall k, (\mathbf{S}^{(k)})^\intercal = \mathbf{S}^{(k)}, \mathbf{S}^{(k)} \geq 0, \\ \boldsymbol{\mathcal{S}} = \mathtt{rotate}\left(\mathtt{bvfold}([\mathbf{S}^{(1)};\cdots;\mathbf{S}^{(m)}])\right)\,.\end{aligned} \tag{12}$$

3) Graph Tensor Guided Graph Clustering: 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 $\mathbf{Z}$ by averaging all frontal slices of $\boldsymbol{\mathcal{S}}^*$, *i.e.*,

$$\mathbf{Z} = \frac{1}{m}\sum_{k=1}^m \boldsymbol{\mathcal{S}}^*(:,:,k)\,. \tag{13}$$

Subsequently, take $\mathbf{Z}$ as input, the spectral clustering algorithm is employed to pursue the clustering assignments.

D. Optimization

In this section, we devise a solver based on the alternating direction method of multipliers (ADMM) [33], [34] to iteratively solve the proposed TMKGC method. According to the principle of ADMM, we first introduce an auxiliary tensor variable $\boldsymbol{\mathcal{A}}$ to make problem (12) separable as follows:

$$\begin{aligned}\min_{\mathbf{S}^{(k)},\boldsymbol{\mathcal{A}}} \sum_{k=1}^m \mathtt{Tr}\left(-2(\mathbf{S}^{(k)})^\intercal\mathbf{H}^{(k)} + (\mathbf{S}^{(k)})^\intercal\mathbf{H}^{(k)}\mathbf{S}^{(k)}\right) \\ +\alpha\|\mathbf{S}^{(k)}\|_F^2 + \beta\|\boldsymbol{\mathcal{A}}\|_\circledast \quad \text{s.t. } (\mathbf{S}^{(k)})^\intercal = \mathbf{S}^{(k)}, \mathbf{S}^{(k)} \geq 0, \\ \boldsymbol{\mathcal{A}} = \boldsymbol{\mathcal{S}}, \boldsymbol{\mathcal{S}} = \mathtt{rotate}\left(\mathtt{bvfold}([\mathbf{S}^{(1)};\cdots;\mathbf{S}^{(m)}])\right)\,.\end{aligned} \tag{14}$$

We then form the following augmented Lagrangian function of (14) as follows

$$\begin{aligned}\min_{\mathbf{S}^{(k)},\boldsymbol{\mathcal{A}}} &\sum_{k=1}^{m} \mathtt{Tr}\left(-2(\mathbf{S}^{(k)})^{\intercal}\mathbf{H}^{(k)} + (\mathbf{S}^{(k)})^{\intercal}\mathbf{H}^{(k)}\mathbf{S}^{(k)}\right) \\ &+\alpha\|\mathbf{S}^{(k)}\|_F^2 + \beta\|\boldsymbol{\mathcal{A}}\|_{\circledast} + \frac{\mu}{2}\|\boldsymbol{\mathcal{A}} - \boldsymbol{\mathcal{S}} + \frac{\boldsymbol{\mathcal{Y}}}{\mu}\|_F^2 \qquad (15)\\ \text{s.t.}\quad &(\mathbf{S}^{(k)})^{\intercal} = \mathbf{S}^{(k)}, \mathbf{S}^{(k)} \geq 0, \\ &\boldsymbol{\mathcal{S}} = \mathtt{rotate}\left(\mathtt{bvfold}([\mathbf{S}^{(1)}; \cdots; \mathbf{S}^{(m)}])\right).\end{aligned}$$

where $\boldsymbol{\mathcal{Y}}$ is the Lagrangian multiplier, and μ is the penalty parameter. Then, we calculate each variable by fixing the remaining variables respectively.

► **Step-1, S-subproblem:** By fixing tensor $\boldsymbol{\mathcal{A}}$, we update each candidate affinity graph $\{\mathbf{S}^{(k)}\}_{k=1}^{m}$ via

$$\begin{aligned}\min_{\mathbf{S}^{(k)}\geq 0,(\mathbf{S}^{(k)})^{\intercal}=\mathbf{S}^{(k)}} &\mathtt{Tr}\left(-2\mathbf{H}^{(k)}\mathbf{S}^{(k)} + (\mathbf{S}^{(k)})^{\intercal}\mathbf{H}^{(k)}\mathbf{S}^{(k)}\right) \\ &+\alpha\|\mathbf{S}^{(k)}\|_F^2 + \frac{\mu}{2}\|\mathbf{A}^{(k)} - \mathbf{S}^{(k)} + \frac{\mathbf{Y}^{(k)}}{\mu}\|_F^2 .\end{aligned} \qquad (16)$$

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. Taking the derivative of this equation with respect to $\mathbf{S}^{(k)}$ and setting it to zero, we can learn the closed form solution of $\mathbf{S}^{(k)}$, *i.e.*,

$$\left(\mathbf{S}^{(k)}\right)^{*} = \left(2\mathbf{H}^{(k)} + 2\alpha\mathbf{I} + \mu\mathbf{I}\right)^{-1}\left(2\mathbf{H}^{(k)} + \mu\mathbf{A}^{(k)} + \mathbf{Y}^{(k)}\right), \qquad (17)$$

Then, considering the constraints, $\mathbf{S}^{(k)} \geq 0, (\mathbf{S}^{(k)})^{\intercal} = \mathbf{S}^{(k)}$, we compute the symmetric non-negative affinity graph using $(\mathbf{S}^{(k)})^{*} = 1/2((\mathtt{abs}(\mathbf{S}^{(k)})^{*}) + \mathtt{abs}((\mathbf{S}^{(k)})^{*}))$.

► **Step-2, $\boldsymbol{\mathcal{A}}$-subproblem:** Ignoring the irrelevant items and fixing the $\{\mathbf{S}^{(k)}\}_{k=1}^{m}$, the optimization problem with respect to $\boldsymbol{\mathcal{A}}$ can be rewritten as

$$\min_{\boldsymbol{\mathcal{A}}} \beta\|\boldsymbol{\mathcal{A}}\|_{\circledast} + \frac{\mu}{2}\|\boldsymbol{\mathcal{A}} - \boldsymbol{\mathcal{S}} + \frac{\boldsymbol{\mathcal{Y}}}{\mu}\|_F^2 , \qquad (18)$$

which is a t-TNN minimization problem. Let $\boldsymbol{\mathcal{B}} = \boldsymbol{\mathcal{S}} - \frac{\boldsymbol{\mathcal{Y}}}{\mu}$, (18) can be solved by applying the tensor tubal-shrinkage of $\boldsymbol{\mathcal{B}}$, according to the below theorem.

Theorem 1 *For a scalar $\rho > 0$ and two 3-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.*

$$\min_{\boldsymbol{\mathcal{A}}} \rho\|\boldsymbol{\mathcal{A}}\|_{\circledast} + \frac{1}{2}\|\boldsymbol{\mathcal{A}} - \boldsymbol{\mathcal{B}}\|_F^2 \qquad (19)$$

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

$$\boldsymbol{\mathcal{A}} = \boldsymbol{\mathcal{C}}_{n_3\rho}(\boldsymbol{\mathcal{B}}) = \boldsymbol{\mathcal{U}} * \boldsymbol{\mathcal{C}}_{n_3\rho}(\boldsymbol{\mathcal{G}}) * \boldsymbol{\mathcal{V}}^{\intercal} , \qquad (20)$$

where $\boldsymbol{\mathcal{B}} = \boldsymbol{\mathcal{U}} * \boldsymbol{\mathcal{G}} * \boldsymbol{\mathcal{V}}^{\intercal}$ and $\boldsymbol{\mathcal{C}}_{n_3\rho}(\boldsymbol{\mathcal{G}}) = \boldsymbol{\mathcal{G}} * \boldsymbol{\mathcal{Q}}$. $\boldsymbol{\mathcal{Q}} \in \mathbb{R}^{n_1\times n_2\times n_3}$ denotes a f-diagonal tensor and each diagonal element of $\boldsymbol{\mathcal{Q}}$ is defined as $\boldsymbol{\mathcal{Q}}_f(i,i,j) = \left(1 - \frac{n_3\rho}{\boldsymbol{\mathcal{G}}(i,i,j)}\right)_+$ [35].

► **Step-3, ADMM variables:** We update the variables involved ADMM by

$$\begin{aligned}\boldsymbol{\mathcal{Y}} &= \boldsymbol{\mathcal{Y}} + \mu(\boldsymbol{\mathcal{A}} - \boldsymbol{\mathcal{S}}) \\ \mu &= \min\left(\rho\,\mu, \mu_{\max}\right) .\end{aligned} \qquad (21)$$

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

Algorithm 1 The algorithm of TMKGC

Input: Multiple base kernels $\{\mathbf{H}^{(k)}\}_{k=1}^{m}$, α, and β.
1: Initialize ADMM variables: $\boldsymbol{\mathcal{Y}} = 0$, $\{\boldsymbol{\mathcal{A}}(:,:,k)\}_{k=1}^{m} = \mathbf{I}$, $\{\mathbf{S}^{(k)}\}_{k=1}^{m} = \mathbf{I}$, $\mu = 10^{-4}$, $\rho = 2$, and $\mu_{max} = 10^{10}$
2: **repeat**
3: Update each candidate graph $\{\mathbf{S}^{(k)}\}_{k=1}^{m}$ via (17);
4: Construct $\boldsymbol{\mathcal{S}}$ via `bvfold` and `rotate` on $\{\mathbf{S}^{(k)}\}_{k=1}^{m}$;
5: Update the tensor $\boldsymbol{\mathcal{A}}$ via (18);
6: Update the ADMM variables via (21);
7: **until** The stopping criterion is satisfied;
8: Construct a balanced affinity graph via (13);
9: Use spectral clustering to obtain the final cluster labels.
Output: Clustering results in terms of three metrics.

TABLE II
SUMMARIES OF THE EIGHT USED DATASETS.

Dataset	Clusters (#*c*)	Samples (#*n*)	Features (#*d*)
Yale	15	165	1024
Jaffe	10	213	676
AR	120	840	768
ORL	40	400	1024
COIL20	20	1440	1024
binaryalphadigs	36	1404	320
Deep CIFAR-10	10	1000	1024
Synthetic dataset	2	2000	2
Synthetic dataset	2	2000	2

The stopping criterion of the iterative algorithm is met when the residuals defined below are small enough, *i.e.*,

$$\max\left\{\left|\mathrm{obj}^{t+1} - \mathrm{obj}^{t}\right|, \left\|\boldsymbol{\mathcal{S}}^{t+1} - \boldsymbol{\mathcal{S}}^{t}\right\|_F^2\right\} \leq \epsilon , \qquad (22)$$

where obj^{t} is the objective value of (12) (*i.e.*, $\mathrm{obj} = \sum_{k=1}^{m}\mathtt{Tr}(\mathbf{H}^{(k)} - 2(\mathbf{S}^{(k)})^{\intercal}\mathbf{H}^{(k)} + (\mathbf{S}^{(k)})^{\intercal}\mathbf{H}^{(k)}\mathbf{S}^{(k)}) + \alpha\|\mathbf{S}^{(k)}\|_F^2 + \beta\|\boldsymbol{\mathcal{S}}\|_{\circledast}$) at the t-th iteration, $\boldsymbol{\mathcal{S}}^{t}$ is output tensor $\boldsymbol{\mathcal{S}}$ at the t-th iteration, and ϵ is a pre-defined threshold tolerance to control the convergence sensibility. The optimization procedure of the iterator algorithm is summarized in Algorithm 1, and the demo code will be released on our GitHub homepage (https://github.com/renzhenwen).

E. Distributed Network Optimizing

If the kernel pool contains multiple base kernels, the standalone computer can not execute the proposed method efficiently. Due to the detachable structure of Algorithm 1, it can be solved by a distributed manner. Assume that there are l ($l \leq m$) slave nodes and one master node, and the non-linear data is stored in the master node. We divide these slave nodes into g groups, and each group has a communication node as the virtual master node of this group. All the base kernels are evenly distributed to the slaves nodes first, therefore these nodes can download the data from the master node and construct respective kernel matrices in advance.

When solving Algorithm 1, peers only synchronize with their immediate topological neighbors of the same group in the underlying communication network. To facilitate understanding, refer to Fig. 5. Specifically, for update each candidate kernel graph $\{\mathbf{S}^{(k)}\}_{k=1}^{m}$, it has an obvious parallel structure,

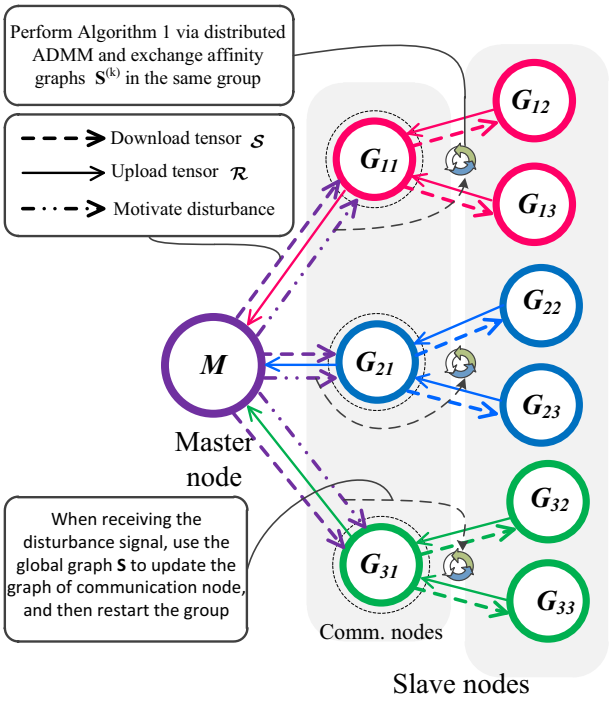


Fig. 5. Illustration of the distributed network optimizing scheme. There are eleven slave nodes and three groups with different colors.

so all the salve nodes in one group can exchange the learned graph with its r neighbors via communication network, where r is the peer neighbor nodes in the same group. For update tensor $\boldsymbol{\mathcal{S}}$, each lateral slice $\{\boldsymbol{\mathcal{S}}_i\}_{i=1}^m$ is a candidate kernel graph, and some slices can be combined into a small-scale tensor $\boldsymbol{\mathcal{R}} \in \mathbb{R}^{n\times r\times n}$. Accordingly, each slave node can solve (18) independently by replacing $\boldsymbol{\mathcal{A}}$, $\boldsymbol{\mathcal{S}}$, and $\boldsymbol{\mathcal{Y}}$ as small-scale tensors. If one group converges, the communication node sends the consensus affinity graph to the master node. When the master node receive g graphs, it will run Algorithm 1, and then sends the learned graph to the communication node as a disturbance. When all the slave nodes and master node converge, the algorithm stops.

F. Computational Complexity and Convergence

Here we give a discussion about the computational cost of Algorithm 1. For updating each candidate affinity graph $\{\mathbf{S}^{(k)}\}_{k=1}^m$, although the matrix inverse operation $(2\mathbf{H}^{(k)} + 2\alpha\mathbf{I} + \mu\mathbf{I})^{-1}$ has high computational complexity, we can pre-calculate it before iteration loops because the matrix inverse operation is independent with $\{\mathbf{S}^{(k)}\}_{k=1}^m$. Therefore, the computational complexity of the *step 1* of Algorithm 1 is $\mathcal{O}(n^2)$, meanwhile, it can be easily parallelized [36]. For updating the tensor $\boldsymbol{\mathcal{A}}$, 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 2* of Algorithm 1 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 these variables involved in ADMM, the computation of *step 3* at each iteration will take $\mathcal{O}(m)$. Theoretically, the computational cost of Algorithm 1 is $\mathcal{O}(t(n^2 + mn^2\log(n) + m^2n^2 + m))$, where t denotes the total number of iterations. In reality, we have $t \ll n$ and a small m. Thus the overall cost for our TMKGC is $\mathcal{O}(tn^2\log(n))$. As a matter of fact, the computational complexity of our method is considerably less than that of the existing overwhelming majority MKGC methods.

For the classical one-block or two-block ADMM, the convergence property has been theoretically proved in [37]. In the proposed algorithm, the updating of $\{\mathbf{S}^{(k)}\}_{k=1}^m$ can be treated as a unified sub-problem during the optimization; moreover, all these involved variables have closed form solutions. Hence, Algorithm 1 will converge to the local optimum. Empirically, we also conduct experiments to prove the convergence property of Algorithm 1 in Section IV-H.

IV. Experiments

In this section, we conduct experiments on seven real-world small-to-medium datasets, one synthetic dataset, and a large-scale dataset to compare the proposed TMKGC method with some state-of-the-art MKGC methods.

A. Datasets and Kernel Setting

We extensively evaluate the clustering performance of the proposed TMKGC on nine widely used datasets: (1) seven small-to-medium datasets [8], [12], [13], [19]–[21], [23], [27], [28], [38], including Yale[1], Jaffe[2], AR[3], ORL[4], binaryalphadigs (BA)[5], COIL20[6], and Deep CIFAR-10[7]; (2) one synthetic dataset; and (3) a large-scale dataset, Flower102[8]. Note here that CIFAR-10 is a deep learning feature dataset, which is built according to [12]. For the synthetic dataset, we simulate two clusters and each cluster contains total 2,000 mobile smart devices in a region. In our experiments, as [12], [13], [19], [23], 12 base kernel functions are adopted for the seven small-to-medium datasets and the synthetic dataset, which consists of a cosine kernel $k_{ij} = (\boldsymbol{x}_i^\intercal\boldsymbol{x}_j)/(\|\boldsymbol{x}_i\|_2^2\cdot\|\boldsymbol{x}_j\|_2^2)$, four polynomial kernels $k_{ij} = (\delta + \boldsymbol{x}_i^\intercal\boldsymbol{x}_j)^p$ where δ varies from $\{0, 1\}$ and p varies from $\{2, 4\}$, and seven radial basis function (RBF) kernels $k_{ij} = \exp(-\|\boldsymbol{x}_i - \boldsymbol{x}_j\|_2^2/(2\pi\tau^2)$, where π varies from the set of $\{0.01, 0.05, 0.1, 1, 10, 50, 100\}$ and τ is the maximum distance between any two samples. For the Flower102 dataset, we construct base kernels by following [25]. In the end, all the kernels are normalized to $[0, 1]$ by $k_{ij} = k_{ij}/\sqrt{k_{ii}k_{jj}}$. Concrete details are summarized in Table II.

[1]http://www.cvc.yale.edu/projects/yalefaces/yalefaces.html
[2]http://www.kasrl.org/jaffe.html
[3]http://www2.ece.ohio-state.edu/~aleix/ARdataset.html
[4]https://www.cl.cam.ac.uk/research/dtg/attarchive/facedataset.html
[5]https://cs.nyu.edu/~roweis/data/binaryalphadigs.mat
[6]http://www.cs.columbia.edu/CAVE/software/softlib/coil-20.php
[7]https://www.cs.toronto.edu/~kriz/cifar.html
[8]https://www.robots.ox.ac.uk/~vgg/data/flowers/102/

TABLE III
PERFORMANCE OF THE COMPARED MKGC METHODS ON EIGHT DATASETS.THE BEST RESULTS ARE HIGHLIGHTED IN BOLDFACE.

Dataset	Metric	MKKM	RMKKM	MVCLFA	AASC	LKGr	SCMK	SMKL	JMKSC	LLMKL	SPMKC-E	SPMKC	CAGL	TMKGC
Year/Ref.	/	12/ [24]	15/ [19]	19/ [25]	12/ [26]	19/ [20]	18/ [27]	18/ [13]	19/ [22]	19/ [21]	20/ [23]	20/ [23]	20/ [12]	/
Yale	ACC	0.457	0.521	0.618	0.406	0.540	0.582	0.585	0.630	0.655	0.658	0.673	0.703	**0.830**
	NMI	0.501	0.556	0.566	0.468	0.609	0.576	0.614	0.631	0.646	0.650	0.660	0.668	**0.866**
	Purity	0.475	0.536	0.624	0.423	0.554	0.610	0.667	0.673	0.683	0.700	0.709	0.709	**0.830**
Jaffe	ACC	0.746	0.871	0.981	0.304	0.861	0.869	0.967	0.967	**1.000**	**1.000**	**1.000**	**1.000**	**1.000**
	NMI	0.798	0.893	0.970	0.272	0.869	0.868	0.951	0.952	**1.000**	**1.000**	**1.000**	**1.000**	**1.000**
	Purity	0.768	0.889	0.981	0.331	0.859	0.882	0.967	0.967	**1.000**	**1.000**	**1.000**	**1.000**	**1.000**
AR	ACC	0.286	0.344	0.667	0.332	0.314	0.544	0.465	0.609	0.853	0.750	0.798	0.891	**0.923**
	NMI	0.592	0.655	0.844	0.651	0.648	0.775	0.681	0.820	0.935	0.889	0.913	0.954	**0.982**
	Purity	0.305	0.368	0.685	0.350	0.330	0.642	0.611	0.656	0.897	0.807	0.858	0.902	**0.979**
ORL	ACC	0.475	0.556	0.692	0.272	0.616	0.656	0.573	0.725	0.800	0.745	0.785	0.863	**0.982**
	NMI	0.689	0.748	0.836	0.438	0.794	0.808	0.733	0.852	0.890	0.855	0.873	0.932	**0.989**
	Purity	0.514	0.602	0.732	0.316	0.658	0.699	0.648	0.753	0.839	0.780	0.803	0.878	**0.982**
COIL	ACC	0.548	0.667	0.664	0.349	0.618	0.591	0.487	0.696	0.636	0.842	0.884	0.860	**0.998**
	NMI	0.707	0.773	0.782	0.419	0.766	0.726	0.628	0.818	0.806	0.909	0.939	0.930	**0.997**
	Purity	0.590	0.699	0.690	0.391	0.650	0.635	0.683	0.806	0.714	0.907	0.944	0.881	**0.998**
BA	ACC	0.405	0.434	0.413	0.271	0.444	0.384	0.246	0.484	0.482	0.508	0.522	0.508	**0.951**
	NMI	0.569	0.585	0.556	0.423	0.604	0.544	0.486	0.621	0.619	0.632	0.649	0.632	**0.971**
	Purity	0.435	0.463	0.438	0.303	0.479	0.606	0.623	0.563	0.593	0.546	0.634	0.546	**0.972**
Deep CIFAR-10	ACC	0.499	0.638	0.698	0.521	0.659	0.727	0.712	0.743	0.760	0.771	0.774	0.788	**0.948**
	NMI	0.467	0.615	0.583	0.498	0.626	0.633	0.611	0.685	0.646	0.673	0.678	0.699	**0.890**
	Purity	0.691	0.654	0.677	0.687	0.734	0.714	0.746	0.760	0.698	0.782	0.796	0.788	**0.948**
Flower102	ACC	0.220	0.271	0.422	0.321	0.415	0.436	0.425	0.440	0.455	0.433	0.441	0.412	**0.795**
	NMI	0.423	0.470	0.605	0.502	0.587	0.610	0.590	0.619	0.632	0.595	0.622	0.586	**0.940**
	Purity	0.276	0.321	0.504	0.411	0.511	0.527	0.522	0.513	0.527	0.514	0.521	0.491	**0.866**

B. Clustering Evaluation Metrics

We report the experimental results employing three most used evaluation metrics, *i.e.*, clustering accuracy (ACC), normalized mutual information (NMI), and Purity [13], [19]. Each metric concentrates on diverse property in terms of clustering results. Additionally, these indicators are positively correlated with the clustering performance, *i.e.*, the higher the value is, the better the clustering performance is.

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, $|\boldsymbol{C}_i|$ $(1 \leq i \leq t)$ be the number of data points in the i-th cluster of the clustering solution, $|\widetilde{\boldsymbol{C}}_j|$ $(1 \leq j \leq s)$ be the number of data points in the j-th cluster of the ground truth, and $|\boldsymbol{C}_i \cap \widetilde{\boldsymbol{C}}_j|$ be the number of objects in both the i-th cluster of the clustering solution and j-th 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

$$ACC = \frac{\sum_{i=1}^{N} \delta(\tilde{c}_i, map(c_i))}{N} \tag{23}$$

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

$$\begin{aligned} NMI(\boldsymbol{C}, \widetilde{\boldsymbol{C}}) &= \frac{MI(\boldsymbol{C}, \widetilde{\boldsymbol{C}})}{\sqrt{H(\boldsymbol{C})H(\widetilde{\boldsymbol{C}})}}, \\ MI(\boldsymbol{C}, \widetilde{\boldsymbol{C}}) &= \sum_i \sum_j \frac{|\boldsymbol{C}_i \cap \widetilde{\boldsymbol{C}}_j|}{N} log \frac{N \cdot |\boldsymbol{C}_i \cap \widetilde{\boldsymbol{C}}_j|}{|\boldsymbol{C}_i||\widetilde{\boldsymbol{C}}_j|} \\ H(\boldsymbol{C}) &= \sum_i \frac{|\boldsymbol{C}_i|}{N} log \frac{|\boldsymbol{C}_i|}{N}, \\ H(\widetilde{\boldsymbol{C}}) &= \sum_j \frac{|\widetilde{\boldsymbol{C}}_j|}{N} log \frac{|\widetilde{\boldsymbol{C}}_j|}{N}, \end{aligned} \tag{24}$$

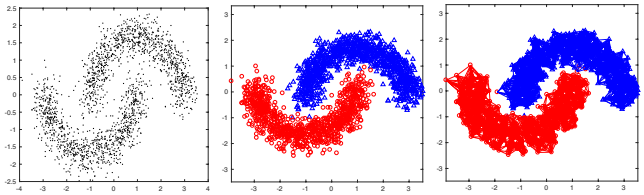
where $H(\boldsymbol{C})$ and $H(\widetilde{\boldsymbol{C}})$ are the entropies of $\boldsymbol{C}$ and $\widetilde{\boldsymbol{C}}$, respectively, and $MI(\boldsymbol{C}, \widetilde{\boldsymbol{C}})$ is the mutual information. Note that $MI(\boldsymbol{C}, \widetilde{\boldsymbol{C}})$ takes values between 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.,

$$Purity = \frac{1}{N} \sum_i \max_j |\boldsymbol{C}_i \cap \widetilde{\boldsymbol{C}}_j| \tag{25}$$

C. Comparison Methods

We thoroughly evaluate proposed TMKGC by comparing with 12 state-of-the-art methods, including MKKM [24], RMKKM [19], AASC [26], LKGr [20], SCMK [27], SMKL [13], MVCLFA [25], JMKSC [22], LLMKL [21], SPMKC-E [23], SPMKC [23], and CAGL [12]. The clustering results are reported in Table III. For the sake of fairness, the



(a) Original data (b) True clusters (c) Predicted clusters

Fig. 6. Simulation of network device clustering. (a) original device distribution, (b) true clusters, and (c) predicted clusters and the constructed graph produced by the proposed TMKGC method.

experimental parameters of these methods are adopted according to the recommended values from their respective papers. To reduce the influence induced by the random initialization in k-means involved in spectral clustering, all experiments on different clustering methods are repeated for 20 times and the average results are reported.

D. Network Device Clustering

In wireless mobile communication networks (WMCN), to expand the communication coverage of heterogeneous or homogeneous WMCN, to avoid frequency interference, and prevent the degradation of network performance, network device clustering is of utmost importance. Here, we consider a toy example to give a visual illustration of the device clustering capability of TMKGC. In this experiment, there are 2,000 devices belonging to two clusters of data distributed in the moon shape. The clustering result is shown in Fig. 6, where the colors of the two clusters are set to be blue and red respectively, and the line width of connecting edges stands for the affinity of two corresponding devices. As shown in Fig. 6, we can easily observe that the proposed TMKGC method can effectively split almost all devices into their respective clusters. Therefore, the experiment can effectively demonstrate the capacity of clustering devices with non-linear distribution. For real-word applications, the network environment, the basic network structure and the QoS objectives (such as transmission reliability, network latency, and energy efficiency) are need to be considered. That is the next phase of our work.

E. Clustering Experimental Results and Discussion

The average clustering results of all comparison methods are reported in Table III. Note here that the standard deviations of almost all experiments are less that 1%, so we don't show the standard deviations in Table III. From these experimental results, we achieve some observations as following:

On all datasets, the proposed TMKGC consistently achieves significant improvements under all these different metrics when comparing with some recent state-of-the-art MKGC methods. Noticeably, on the BA dataset, TMKGC outperforms the second best SPMKC [23] method over 42.9%, 32.2%, and 33.8% in terms of ACC, NMI and purity, respectively. On the deep CIFAR-10 dataset, TMKGC outperforms CAGL [12] method with approximately 16.0%, 19.1%, and 16.0% in terms of ACC, NMI and purity, respectively. The main reason is that our TMKGC can exploit the high-order correlations of the NMF tailored graph tensor. Therefore, these results verify the effectiveness of our TMKGC.

TABLE IV
PERFORMANCE OF THE PROPOSED MKGC METHODS IN STAND-ALONE SCENARIO AND DISTRIBUTED NETWORK SCENARIO.

Method	ACC	NMI	Purity	Running time (in seconds)
Stand-alone TMKGC	0.741	0.897	0.812	3763
Distributed network TMKGC	0.725	0.872	0.786	1542

Moreover, note here that CAGL [12] and the proposed TMKGC are pure graph-based[9] MKGC method, while other graph-based methods are hybrid. From the results, it is obvious that the purebreds are superior to the hybrids. In other words, for a graph-based method, we should pay more attention to graph learning rather than kernel learning.

To sum up, the proposed TMKGC method achieves better performance owing to the following advantages: (1) TMKGC learns an affinity graph for spectral clustering by using a pure graph learning paradigm, rather than the traditional non-linearly or linearly kernel combination; and (2) TMKGC captures the high-order structure information derived from base kernels by using high-order graph tensor learning.

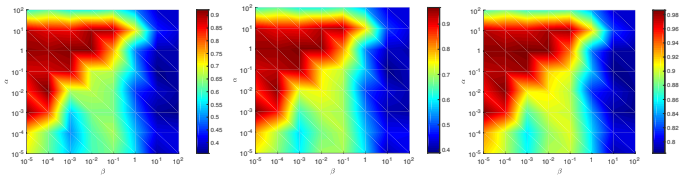
F. Distributed Network Experiment

In this section, we evaluate the proposed TMKGC method in a stand-alone scenario and a distributed scenario. There are four computers in a local area network, including one master node and three slave nodes, where the communication delay and the fail of synchrony is ignored. As shown in Fig. 5, the different color of circle stands for different computer. The experimental results in terms of ACC, NMI, Purity, and running time are shown in Table IV. Then it is easy to see that the distributed network TMKGC has lower time consumption, but has higher clustering performance, by comparing with the stand-alone version. Overall, the proposed TMKGC can be solved via stand-alone manner or distributed manner depending on the balance of time and performance.

G. Parameter Sensitivity

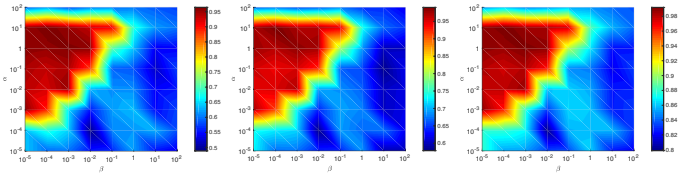
In the proposed method, there are two trade-off parameters α and β required to be set properly, which are used to balance the effects of $\{\|\mathbf{S}^{(k)}\|_F^2\}_{k=1}^m$ and $\|\boldsymbol{\mathcal{S}}\|_{\circledast}$, respectively. By leveraging a grid search technique, we tune both α and β from 10^{-5} to 10^2 with step size 10, respectively. Take the AR, ORL, COIL20 and BA datasets for example, as shown in Figs. 7 to 10, the proposed TMKGC method can achieve the best performance on all evaluated datasets when setting $\alpha \in [10^{-1}, 10^1]$ and $\beta \in [10^{-5}, 10^{-3}]$. These analyses confirm that TMKGC works well for a wide range of α and β values, and can be easily tuned.

[9]For MKGC, the pure graph-based method aims to learn an affinity graph absorbedly, while the hybrid graph-based method aims to learn a graph and a consensus kernel simultaneously.



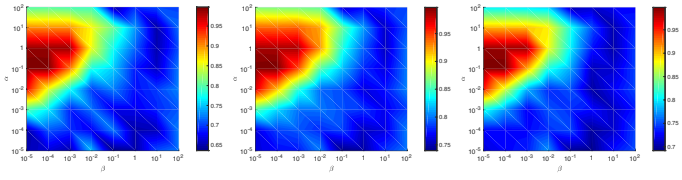
(a) ACC (b) NMI (c) Purity

Fig. 7. Performance with respect to α and β on the AR dataset



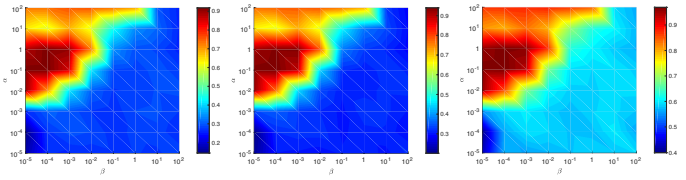
(a) ACC (b) NMI (c) Purity

Fig. 8. Performance with respect to α and β on the ORL dataset



(a) ACC (b) NMI (c) Purity

Fig. 9. Performance with respect to α and β on the COIL20 dataset



(a) ACC (b) NMI (c) Purity

Fig. 10. Performance with respect to α and β on the BA dataset

H. Convergence Analysis

To demonstrate the convergence of Algorithm 1 experimentally, we record the value of the stopping criterion in each iteration on the five datasets (*i.e.*, AR, BA, COIL20, ORL, and Yale), as illustrated in Fig. 11. As we can see, the algorithm converges rapidly till to reach stability, even some of which meet the stopping criterion in just 15 iterators. This proves the fast convergence property of the proposed algorithm. Compared with the existing methods, such as SCMK [27], SMKL [13], MVCLFA [25], JMKSC [22], LLMKL [21], SPMKC [23], and CAGL [12], they usually converge within 10-30 iterations. Although different methods have different stopping criterions, these MKGC methods have comparable iterations to meet their stopping criterions.

V. Conclusion

In this paper, a novel NMF tailored graph tensor method, *TMKGC*, is proposed for non-linear data clustering widely existing in wireless network applications. To effectively handle non-linear data, TMKGC first proposes a NMF-based graph learning paradigm (*i.e.*, GNMF) to learn multiple candidate kernel graphs from the pre-defined kernel pool. Meanwhile, to capture the consistent and complementary information of these graphs for clustering purpose, these graphs are folded and rotated as a graph tensor. Upon that, the t-TNN is employed to explore the high-order correlations of the graph tensor, so as to learn a high-quality affinity graph. Moreover, the local and distributed solvers are also given. Comprehensive experiments on several real-world datasets have demonstrated the remarkable improvements of TMKGC in terms of three widely used clustering metrics.

Acknowledgment

This research was supported by the Sichuan Science and Technology Program (Grant nos. 2019ZDZX0043 and 2020ZDZX0014), the Key Lab of Film and TV Media Technology of Zhejiang Province (Grant no. 2020E10015), the Natural Science Foundation of Chongqing (Grant no. cstc2020jcyj-msxmX0473), the Scientific Research Fund of Sichuan Provincial Education Department (Grant no. 17ZB0441), and the Scientific Research Fund of Southwest University of Science and Technology (Grant no. 17zx7137).

References

[1] S. Yu, X. Chen, L. Yang, D. Wu, M. Bennis, and J. Zhang, "Intelligent edge: Leveraging deep imitation learning for mobile edge computation offloading," *IEEE Wireless Commun.*, vol. 27, no. 1, pp. 92–99, Feb. 2020.

[2] G. Lee, W. Saad, and M. Bennis, "An online framework for ephemeral edge computing in the internet of things," 2020, arXiv:2004.08640.

[3] M. Chen, H. V. Poor, W. Saad, and S. Cui, "Wireless communications for collaborative federated learning in the Internet of things," 2020, arXiv:2006.02499.

[4] I. A. Najm, A. K. Hamoud, J. Lloret, and I. Bosch, "Machine learning prediction approach to enhance congestion control in 5G IoT environment," *Electronics*, vol. 8, no. 6, p. 607, May 2019.

[5] J. Park, S. Samarakoon, M. Bennis, and M. Debbah, "Wireless network intelligence at the edge," *Proc. of the IEEE*, vol. 107, no. 11, pp. 2204–2239, Nov. 2019.

[6] A. Elgabli, J. Park, A. S. Bedi, M. Bennis, and V. Aggarwal, "Communication efficient framework for decentralized machine learning," in *Proc. IEEE 54th Annual Conf. Informat. Sciences and Syst. (CISS)*, Mar. 2020.

[7] A. Elgabli, J. Park, A. S. Bedi, M. Bennis, and V. Aggarwal, "Q-GADMM: Quantized group ADMM for communication efficient decentralized machine learning," in *Proc. IEEE Int. Conf. Acoustics, Speech and Signal Proces. (ICASSP)*, May 2020, pp. 8876–8880.

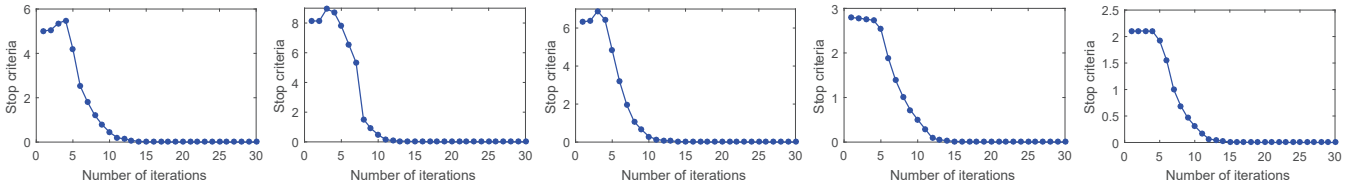
[8] X. Liu, X. Zhu, M. Li, L. Wang, E. Zhu, T. Liu, M. Kloft, D. Shen, J. Yin, and W. Gao, "Multiple kernel k-means with incomplete kernels," *IEEE transactions on pattern analysis and machine intelligence*, vol. 42, no. 5, pp. 1191–1204, 2020.

[9] Z. Kang, X. Lu, Y. Lu, c. Peng, W. Chen, and Z. Xu, "Structure learning with similarity preserving," *Neural Networks*, p. 10.1016/j.neunet.2020.05.030, 2020.

[10] R. Bruschi, F. Davoli, P. Lago, and J. F. Pajo, "A multi-clustering approach to scale distributed tenant networks for mobile edge computing," *IEEE Journal on Selected Areas in Communications*, vol. 37, no. 3, pp. 499–514, 2019.

[11] A. Ali, M. E. Ahmed, F. Ali, N. H. Tran, D. Niyato, and S. Pack, "Nonparametric bayesian channels clustering (nobel) scheme for wireless multimedia cognitive radio networks," *IEEE Journal on Selected Areas in Communications*, vol. 37, no. 10, pp. 2293–2305, 2019.

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



(a) AR (b) BA (c) COIL20 (d) ORL (e) Yale

Fig. 11. Convergence curves of the proposed TMKGC method on five datasets.

[13] Z. Kang, X. Lu, J. Yi, and Z. Xu, "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*, 2018, pp. 2312–2318.

[14] Z. Kang, H. Pan, S. C. Hoi, and Z. Xu, "Robust graph learning from noisy data," *IEEE transactions on cybernetics*, vol. 28, no. 4, pp. 1007–1021, 2020.

[15] C. Lu, J. Feng, Z. Lin, T. Mei, and S. Yan, "Subspace clustering by block diagonal representation," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 41, no. 2, pp. 487–501, 2019.

[16] H. Wang, Y. Yang, B. Liu, and H. Fujita, "A study of graph-based system for multi-view clustering," *Knowledge-Based Systems*, vol. 163, pp. 1009–1019, 2019.

[17] F. Nie, X. Wang, and H. Huang, "Clustering and projected clustering with adaptive neighbors," in *Proceedings of the twentieth ACM SIGKDD international conference on Knowledge discovery and data mining*. ACM, 2014, pp. 977–986.

[18] M. Li, X. Liu, L. Wang, Y. Dou, J. Yin, and E. Zhu, "Multiple kernel clustering with local kernel alignment maximization," in *Proceedings of the Twenty-Fifth International Joint Conference on Artificial Intelligence*. AAAI Press, 2016, pp. 1704–1710.

[19] L. Du, P. Zhou, L. Shi, H. Wang, M. Fan, W. Wang, and Y.-D. Shen, "Robust multiple kernel k-means using l21-norm," in *Proceedings of the Twenty-Fourth International Joint Conference on Artificial Intelligence*, 2015, pp. 3476–3482.

[20] Z. Kang, L. Wen, W. Chen, and Z. Xu, "Low-rank kernel learning for graph-based clustering," *Knowledge-Based Systems*, vol. 163, pp. 510–517, 2019.

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

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

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

[24] H.-C. Huang, Y.-Y. Chuang, and C.-S. Chen, "Multiple kernel fuzzy clustering," *IEEE Transactions on Fuzzy Systems*, vol. 20, no. 1, pp. 120–134, 2012.

[25] S. Wang, X. Liu, E. Zhu, C. Tang, J. Liu, J. Hu, J. Xia, and J. Yin, "Multi-view clustering via late fusion alignment maximization," in *Twenty-Eighth International Joint Conference on Artificial Intelligence*. AAAI Press, 2019, pp. 3778–3784.

[26] H.-C. Huang, Y.-Y. Chuang, and Chen, "Affinity aggregation for spectral clustering," in *IEEE Conference on Computer Vision and Pattern Recognition*. IEEE, 2012, pp. 773–780.

[27] Z. Kang, C. Peng, Q. Cheng, and Z. Xu, "Unified spectral clustering with optimal graph," in *Thirty-Second AAAI Conference on Artificial Intelligence*, 2018, pp. 3366–3373.

[28] S. Zhou, X. Liu, M. Li, E. Zhu, L. Liu, C. Zhang, and J. Yin, "Multiple kernel clustering with neighbor-kernel subspace segmentation," *IEEE Transactions on Neural Networks*, pp. 1–12, 2019.

[29] Z. Zhang, Y. Zhang, S. Li, G. Liu, D. Zeng, S. Yan, and M. Wang, "Flexible auto-weighted local-coordinate concept factorization: A robust framework for unsupervised clustering," *IEEE Transactions on Knowledge and Data Engineering*, 2019.

[30] J. Wu, X. Xie, L. Lin, N. Zhouchen, and H. Zha, "Unified graph and low-rank tensor learning for multi-view clustering," in *Prof. Association for the Advancement of Artificial Intelligence (AAAI)*, 2020.

[31] C. Zhang, H. Fu, S. Liu, G. Liu, and X. Cao, "Low-rank tensor constrained multiview subspace clustering," in *Proceedings of the IEEE international conference on computer vision*, 2015, pp. 1582–1590.

[32] M. Wei, J. Huang, X. Xie, L. Liu, J. Wang, and J. Qin, "Mesh denoising guided by patch normal co-filtering via kernel low-rank recovery," *IEEE Transactions on Visualization and Computer Graphics*, vol. 25, no. 10, pp. 2910–2926, 2019.

[33] Z. Lin, R. Liu, and Z. Su, "Linearized alternating direction method with adaptive penalty for low-rank representation," in *Advances in neural information processing systems*, 2011, pp. 612–620.

[34] K. Huang, N. D. Sidiropoulos, and A. P. Liavas, "A flexible and efficient algorithmic framework for constrained matrix and tensor factorization," *IEEE Transactions on Signal Processing*, vol. 64, no. 19, pp. 5052–5065, 2016.

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

[36] A. Elgabli, J. Park, A. S. Bedi, M. Bennis, and V. Aggarwal, "Q-gadmm: Quantized group admm for communication efficient decentralized machine learning," in *ICASSP 2020-2020 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. IEEE, 2020, pp. 8876–8880.

[37] E. Esser, "Applications of lagrangian-based alternating direction methods and connections to split bregman," *CAM report*, vol. 9, p. 31, 2009.

[38] C. Wang, E. Zhu, X. Liu, L. Gao, J. Yin, and N. Hu, "Multiple kernel clustering with global and local structure alignment," *IEEE Access*, vol. 6, pp. 77 911–77 920, 2018.