Hierarchical Multiple Kernel Clustering

Jiyuan Liu, [1] **Xinwang Liu,** [1*] **Yuexiang Yang,** [1] **Siwei Wang,** [1] **Sihang Zhou** [2]

[1] College of Computer, National University of Defense Technology, Changsha, Hunan, China.
[2] College of Intelligence Science and Technology, National University of Defense Technology, Changsha, Hunan, China.
{liujiyuan13, xinwangliu, yyx, wangsiwei13}@nudt.edu.cn, sihangjoe@gmail.com

Abstract

Current multiple kernel clustering algorithms compute a partition with the consensus kernel or graph learned from the pre-specified ones, while the emerging late fusion methods firstly construct multiple partitions from each kernel separately, and then obtain a consensus one with them. However, both of them directly distill the clustering information from kernels or graphs with size $\mathbb{R}^{n\times n}$ to partition matrices with size $\mathbb{R}^{n\times k}$, where n and k are the number of samples and clusters, respectively. This sudden drop of dimension would result in the loss of advantageous details for clustering. In this paper, we provide a brief insight of the aforementioned issue and propose a hierarchical approach to perform clustering while preserving advantageous details maximumly. Specifically, we gradually group samples into $\{c_t\}_{t=1}^{s}$ clusters, together with generating a sequence of intermediary matrices with size $\mathbb{R}^{n\times c_t}$, in which $n > c_1 > \cdots > c_s > k$. A consensus partition with size $\mathbb{R}^{n\times k}$ is simultaneously learned and conversely guides the construction of intermediary matrices. This cyclic process is modeled into an unified objective and an alternative algorithm is designed to solve it. In addition, the proposed method is validated and compared with other representative multiple kernel clustering algorithms on benchmark datasets, demonstrating state-of-the-art performance by a large margin.

Introduction

Although kernel clustering algorithms, such as kernel k-means, kernel spectral clustering, etc., have already achieved satisfying performance in various applications, how to choose the right kernel mapping and corresponding parameters is still a tricky problem. More lethal is their incapability of dealing with rapidly increasing multi-view data. Multiple kernel clustering (MKC) methods address these problems by generating multiple kernels with a set of parameterized mapping functions corresponding to each data view, and then exploring the complementary information to categorize samples into clusters. According to the time when clustering details are utilized, they can be roughly separated into two categories, i.e. *early-fusion* and *late-fusion* methods.

Early-fusion approaches directly learn a consensus kernel or graph from multiple ones, afterwards generate the final partition [Ren et al. 2020; Wen et al. 2020]. Mostly, the two processes are jointly optimized, as shown in Fig. 1(a) and 1(b). In this group of methods, low rank optimization techniques are widely adopted. For instance, Zhou et al. find rows and columns of kernel will be both contaminated when a sample is corrupted with noise, and employ $\mathcal{L}_{2,1}$ norm regularization to capture such row-wise and column-wise noises [Zhou et al. 2015]. Following multiple kernel learning (MKL) framework [Kloft, Rückert, and Bartlett 2010; Kloft et al. 2011], a large number of algorithms assume that the consensus kernel is a weighted combination of the pre-specified kernels [Huang, Chuang, and Chen 2012; Kloft et al. 2011; Liu et al. 2020a; Peng et al. 2019; Liu et al. 2020c]. Among them, various regularization terms, such as local kernel alignment [Gönen and Margolin 2014], matrix-induced regularization [Liu et al. 2016] and optimal neighborhood kernel [Liu et al. 2017], are proposed to refine the kernel weights. In addition, some multi-view spectral clustering methods firstly construct graphs from kernels, and then adopt graph-based techniques to learn the cluster assignments with the by-product unified graph [Kang et al. 2018; Zhan et al. 2019; Kang et al. 2019; Zhou et al. 2020]. Late-fusion algorithms firstly obtain multiple partitions from each kernel independently with basic kernel clustering algorithms [Liu et al. 2020b]. Then, the final clustering assignments are constructed based on them, as shown in Fig. 1(c). Long et al. introduce the concept of mapping functions to make spaces of each view comparable, hence an optimal clustering assignment can be made from multiple results [Long, Yu, and Zhang 2008]. Wang et al. build the base partitions by performing kernel k-means on pre-specified kernels, then maximize the alignment between consensus partition and the weighted combination of base partitions [Wang et al. 2019].

Although early-fusion and late-fusion MKC methods achieve satisfying performances, both of them try to encode the clustering information from kernels or graphs with size $\mathbb{R}^{n\times n}$ to partition matrices with size $\mathbb{R}^{n\times k}$, where n and k are the number of samples and clusters, respectively. To see this point in depth, we conduct a brief illustration on *CCV* benchmark in Fig. 2. It can be clearly observed that a large volume of details beneficial to clustering would be lost in

[*] Corresponding author

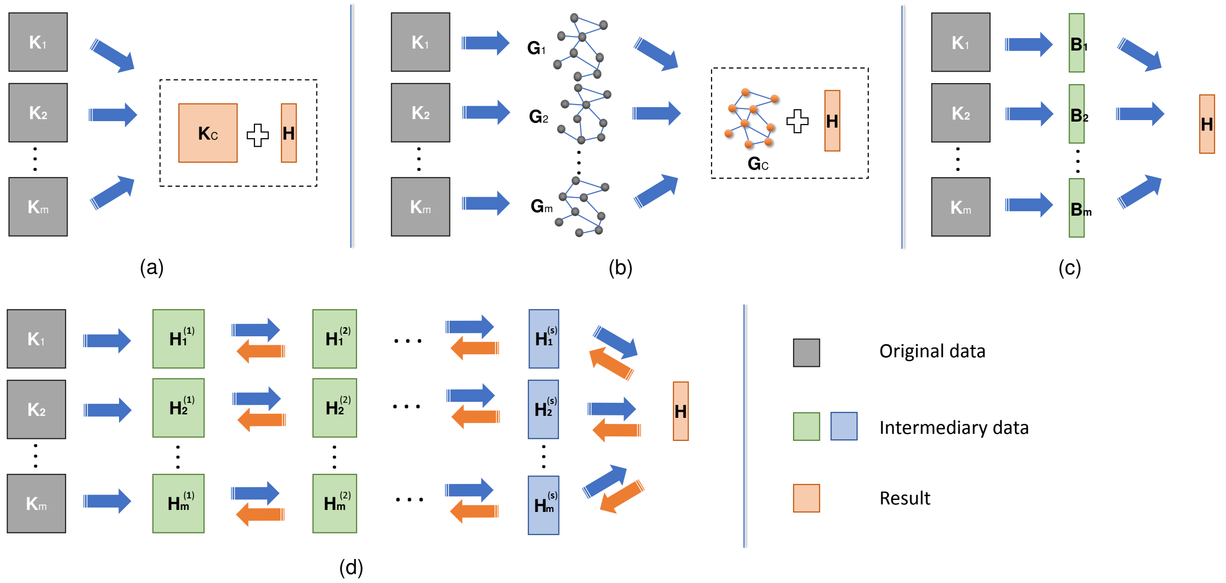


Figure 1: (a) and (b) visualize early-fusion methods with kernels and graphs, while (c) and (d) are the frameworks of late-fusion approaches and the proposed algorithm, respectively. The gray squares or graphs and orange ones indicate the original data and clustering results. Squares of other colors represent the intermediary data. Early-fusion methods obtain the final partition, $\mathbf{H}$, by learning a consensus kernel or graph, $\mathbf{K}_C$ or $\mathbf{G}_C$, from the pre-specified ones, $\{\mathbf{K}_p\}_{p=1}^{m}$ or $\{\mathbf{G}_p\}_{p=1}^{m}$. On the contrary, late-fusion methods construct base partitions, $\{\mathbf{B}_p\}_{p=1}^{m}$, in one-way ticket from each kernel without considering the others, and then integrate them into a final one, $\mathbf{H}$. Different from these two approaches, the proposed algorithm extracts the partition information into smaller matrices, $\{\mathbf{H}_p^{(t)}\}_{p,t=1}^{m,s}$, iteratively, and learn the final partition, $\mathbf{H}$, after one or more steps. In upper cases, two-way arrow represents update of current matrices would affect the previous ones.

this sudden drop of dimension. A brief insight is also provided at the beginning of section 3. In order to balance the partition information distilling and preserving, we propose a *Hierarchical Multiple Kernel Clustering* (HMKC) approach. At the beginning, data samples are categorized into c_1 clusters by constructing an intermediary partition matrix $\mathbf{H}^{(1)} \in \mathbb{R}^{n \times c_1}$ corresponding to each kernel, where $n > c_1 > k$. Based on the obtained $\mathbf{H}^{(1)}$, a smaller intermediary partition matrix $\mathbf{H}^{(2)} \in \mathbb{R}^{n \times c_2}$, in which $n > c_1 > c_2 > k$, are computed to get c_2 clusters. By repeating the aforementioned process s times, beneficial clustering details can be distilled into $\mathbf{H}^{(s)} \in \mathbb{R}^{n \times c_s}$ step by step. With $\{\mathbf{H}_p^{(s)}\}_{p=1}^{m}$ corresponding to m kernels, the consensus partition $\mathbf{H}$ is simultaneously learned and conversely guides the construction of intermediary matrices. We also jointly optimize the intermediary matrices and the consensus partition by modeling them in an unified objective. The above idea is visualized in Fig. 1(d). Furthermore, an alternative algorithm is designed to solve the resultant optimization problem and proven to be convergent. Extensive experiments are conducted on benchmark datasets. It can observed that the proposed algorithm outperforms other representative MKC algorithms.

Related work

Kernel k-means

Given a collection of data observations $\{\mathbf{x}_i\}_{i=1}^{n}$ and a kernel mapping $\phi(\cdot)$, kernel k-means aims to group the samples into k clusters via minimizing the sum-of-square loss, which can be formulated as

$$\min_{\mathbf{A},\mathbf{c}} \sum_{i=1,j=1}^{n,k} A_{ij} \left\| \phi\left(\mathbf{x}_i\right) - \mathbf{c}_j \right\|_2^2 \;\; s.t. \sum_{j=1}^{k} A_{ij} = 1 \tag{1}$$

where $\mathbf{A} \in \{0,1\}^{n \times k}$ is the cluster assignment of each sample and $\mathbf{c}_j$ is the j-th cluster centroid.

In most cases, $\phi(\mathbf{x}_i) \in \mathbb{R}^d$ where $d \gg n$ or infinite. As a result, Eq. (1) cannot be optimized directly. Therefore, we equivalently rewrite its matrix-vector form as

$$\min_{\mathbf{A}} \operatorname{Tr}(\mathbf{K}) - \operatorname{Tr}\left(\mathbf{L}^{\frac{1}{2}}\mathbf{A}^{\top}\mathbf{K}\mathbf{A}\mathbf{L}^{\frac{1}{2}}\right) \tag{2}$$

in which $K_{ij} = \phi(\mathbf{x}_i)^{\top}\phi(\mathbf{x}_j)$, $\mathbf{L} = diag([n_1^{-1}, n_2^{-1}, \cdots, n_k^{-1}])$ with $n_j = \sum_{i=1}^{n} A_{ij}$. The discrete $\mathbf{A}$ makes the upper equation hard to solve, and a common trick is to relax it to take arbitrary values. By defining $\mathbf{H} = \mathbf{A}\mathbf{L}^{\frac{1}{2}}$, the above problem can be transformed to

$$\begin{aligned} \min_{\mathbf{H}} \;\; & \operatorname{Tr}\left[\mathbf{K}\left(\mathbf{I}_n - \mathbf{H}\mathbf{H}^{\top}\right)\right] \\ \text{s.t.} \;\; & \mathbf{H}^{\top}\mathbf{H} = \mathbf{I}_k, \mathbf{H} \in \mathbb{R}^{n \times k}, \end{aligned} \tag{3}$$

which can be simplified as follows:

$$\begin{aligned} \max_{\mathbf{H}} \;\; & \operatorname{Tr}\left(\mathbf{K}\mathbf{H}\mathbf{H}^{\top}\right) \\ \text{s.t.} \;\; & \mathbf{H}^{\top}\mathbf{H} = \mathbf{I}_k, \mathbf{H} \in \mathbb{R}^{n \times k}, \end{aligned} \tag{4}$$

With the obtained partition matrix $\mathbf{H}$, standard k-means clustering is applied to compute the final clustering assignments.

Multiple kernel k-means

Following the MKL framework, multiple kernel k-means (MKKM) assumes the consensus kernel $\mathbf{K}_{\boldsymbol{\beta}}$ as a weighted combination of the pre-specified ones [Huang, Chuang, and Chen 2012], and minimize

$$\begin{aligned} \min_{\mathbf{H},\boldsymbol{\beta}} \ & \operatorname{Tr}\left[\mathbf{K}_{\boldsymbol{\beta}}\left(\mathbf{I}_n - \mathbf{H}\mathbf{H}^\top\right)\right] \\ \text{s.t. } & \boldsymbol{\beta}^\top \mathbf{1} = 1, \mathbf{H}^\top\mathbf{H} = \mathbf{I}_k, \mathbf{H} \in \mathbb{R}^{n\times k} \end{aligned} \tag{5}$$

in which $\mathbf{K}_{\boldsymbol{\beta}} = \sum_{p=1}^{m} \beta_p^2 \mathbf{K}_p$.

The proposed algorithm

In the following, we provide a brief insight on the clustering detail loss of current MKC algorithms when constructing the consensus partition. The results are obtained on *CCV* dataset which is collected from 20 clusters and thoroughly described in Table 1.

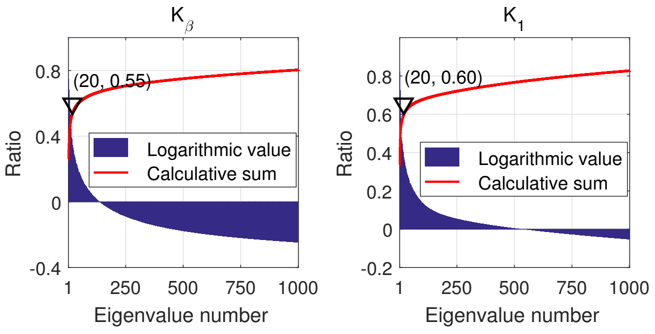


Figure 2: Eigenvalue distributions of the first kernel $\mathbf{K}_1$ and consensus kernel $\mathbf{K}_{\boldsymbol{\beta}}$ obtained by MKKM on *CCV*. The eigenvalues are sorted from large to small, and only the first 1000 out of 6773 ones are plotted. The bar plot shows the times of each logarithmic eigenvalue to the first one. Meanwhile, the curve presents the calculative sum of the sorted eigenvalues.

Early-fusion MKC algorithms try to construct the consensus kernel or graph by exploring the complementary information among the pre-specified ones. For the learned consensus kernel, most of them simultaneously perform kernel k-means to obtain the partition matrix with size $\mathbb{R}^{n\times k}$. Meanwhile, the graph methods produce the partition matrix of the same size by employing spectral clustering (normalized cut), which has been proven to be a special case of weighted kernel k-means [Dhillon, Guan, and Kulis 2004]. So, we only analyze the properties of kernel k-means here due to the space limit. Specifically, we take MKKM as an instance and present the eigenvalue distribution of the consensus kernel $\mathbf{K}_{\boldsymbol{\beta}}$ on the left of Fig. 2. As claimed in section 4.2 of [Schölkopf, Smola, and Müller 1998], the eigenvector corresponding to a larger eigenvalue carries more discriminative information. By selecting the eigenvectors corresponding to top-k eigenvalues, kernel k-means preserves most of the discriminative details in a kernel. In an ideal kernel for clustering, eigenvalues except the top-k are supposed to be 0 [Kalman 1996; Zhang 2015]. However, it can be observed that there are around 100 more eigenvalues larger than 1, apart from the biggest 20 ones. If taking the eigenvalue to roughly measure the volume of discriminative information in a corresponding eigenvector, we can see that the resulting partition matrix only keeps 55% kernel details.

Most early-fusion MKC approaches adopt kernel k-means to compute multiple partitions separately [Liu et al. 2019; Wang et al. 2019]. Therefore, we visualize the eigenvalue distribution of the first kernel matrix on the right of Fig. 2. Similar results are observed that there are almost 500 eigenvalues larger than 1, while the top-20 eigenvectors keep about 60% kernel discriminative details. As a result, the multiple independent partition matrices are of low quality, acting as the bottleneck of performance improvement. We can summarize two observations as follows:

1. The discriminative details advantageous for clustering in a kernel with size $\mathbb{R}^{n\times n}$, whether consensus or original one, can be encoded in a proportion of eigenvectors by classical kernel methods, such as kernel k-means and spectral clustering.
2. Directly adopting a partition matrix with size $\mathbb{R}^{n\times k}$ would lose a large volume of the kernel discriminative information.

To balance the partition information distilling and preserving, we firstly group data samples into c_1 clusters by employing intermediary matrices with size $\mathbb{R}^{n\times c_1}$, where $n > c_1 > k$, to extract the clustering details from kernels. In specific, p-th intermediary matrix $\mathbf{H}_p^{(1)}$ can be constructed by performing kernel k-means on $\mathbf{K}_p$ as

$$\begin{aligned} \max_{\mathbf{H}_p^{(1)}} \ & \operatorname{Tr}\left(\mathbf{K}_p \mathbf{H}_p^{(1)} \mathbf{H}_p^{(1)\top}\right) \\ s.t. \ & \mathbf{H}_p^{(1)\top}\mathbf{H}_p^{(1)} = \mathbf{I}_{c_1},\ \mathbf{H}_p^{(1)} \in \mathbb{R}^{n\times c_1}. \end{aligned} \tag{6}$$

Furthermore, we find c_1 might take values which are greatly larger than k, such k^2 or even k^3, especially when data samples are generated from a small number of distributions. This abrupt drop of dimension would cause the loss of important partition information. Therefore, we construct a new kernel $\mathbf{H}_p^{(1)}\mathbf{H}_p^{(1)\top}$, afterwards group samples into c_2 clusters via kernel k-means and compute the intermediary matrix $\mathbf{H}_p^{(2)} \in \mathbb{R}^{n\times c_2}$. By repeating this process s times, data samples are gradually grouped into $\{c_t\}_{t=1}^{s}$ clusters, together with generating a sequence of intermediary matrices $\mathbf{H}_p^{(t)} \in \mathbb{R}^{n\times c_t}$, where $n > c_1 > \cdots > c_s > k$. Without loss of generality, we formulate the relation between $\mathbf{H}_p^{(t-1)}$ and $\mathbf{H}_p^{(t)}$ as

$$\begin{aligned} \max_{\mathbf{H}_p^{(t)}} \ & \operatorname{Tr}\left(\mathbf{H}_p^{(t-1)}\mathbf{H}_p^{(t-1)\top}\mathbf{H}_p^{(t)}\mathbf{H}_p^{(t)\top}\right) \\ s.t. \ & \mathbf{H}_p^{(t)\top}\mathbf{H}_p^{(t)} = \mathbf{I}_{c_t},\ \mathbf{H}_p^{(t)} \in \mathbb{R}^{n\times c_t}. \end{aligned} \tag{7}$$

With the learned matrix $\{\mathbf{H}_p^{(s)}\}_{p=1}^{m}$ for m pre-specified kernels, we also perform kernel k-means on each one to obtain a consensus partition and employ an vector $\boldsymbol{\beta} \in \mathbb{R}^m$ to adjust their weights, which can be formulated as

$$
\begin{aligned}
&\max_{\mathbf{H},\boldsymbol{\beta}} \ \sum_{p=1}^{m} \beta_p \operatorname{Tr}\left(\mathbf{H}_p^{(s)}\mathbf{H}_p^{(s)\top}\mathbf{H}\mathbf{H}^\top\right) \\
&\text{s.t. } \boldsymbol{\beta}^\top\boldsymbol{\beta} = 1, \mathbf{H}^\top\mathbf{H} = \mathbf{I}_k, \mathbf{H} \in \mathbb{R}^{n\times k},
\end{aligned}
\tag{8}
$$

Utilizing Eq. (6), (7) and (8) into an unified objective, we can obtain the proposed final objective as

$$
\begin{aligned}
\max_{\mathbf{H},\{\mathbf{H}_p\}_{p=1}^m,\boldsymbol{\beta},\boldsymbol{\gamma}} \ & \sum_{p=1}^{m} \gamma_p^{(1)}\operatorname{Tr}\left(\mathbf{K}_p\mathbf{H}_p^{(1)}\mathbf{H}_p^{(1)\top}\right) \\
& + \sum_{t=2}^{s}\sum_{p=1}^{m} \gamma_p^{(t)}\operatorname{Tr}\left(\mathbf{H}_p^{(t-1)}\mathbf{H}_p^{(t-1)\top}\mathbf{H}_p^{(t)}\mathbf{H}_p^{(t)\top}\right) \\
& + \sum_{p=1}^{m} \beta_p\operatorname{Tr}\left(\mathbf{H}_p^{(s)}\mathbf{H}_p^{(s)\top}\mathbf{H}\mathbf{H}\right) \\
s.t. \ \mathbf{H}^\top\mathbf{H} = & \mathbf{I}_k, \ \mathbf{H} \in \mathbb{R}^{n\times k}, \ \mathbf{H}_p^{(t)\top}\mathbf{H}_p^{(t)} = \mathbf{I}_{c_t}, \\
\mathbf{H}_p^{(t)} \in \ & \mathbb{R}^{n\times c_t}, \ n > c_1 > \cdots > c_s > k, \ \boldsymbol{\beta}^\top\boldsymbol{\beta} = 1, \\
\beta_p \geq 0, \ & \boldsymbol{\beta} \in \mathbb{R}^m, \ \boldsymbol{\gamma}^{(t)\top}\boldsymbol{\gamma}^{(t)} = 1, \ \gamma_p^{(t)} \geq 0, \ \boldsymbol{\gamma}^{(t)} \in \mathbb{R}^m,
\end{aligned}
\tag{9}
$$

in which $\boldsymbol{\beta}$ and $\boldsymbol{\gamma}^{(t)}$ are the weights corresponding to each kernel. It is worth to note that no hyper-parameters are employed, for we treat all partition information distilling processes equally. Overall, the partition information are firstly distilled from kernel matrices, $\mathbf{K}_p$, to $\mathbf{H}_p^{(1)}$, then to $\mathbf{H}_p^{(2)}$, ..., to $\mathbf{H}_p^{(s)}$, finally fused in $\mathbf{H}$. With the dimensions reducing in steps, the relative important similarity information advantageous for clustering is gradually gathered and used to update the consensus partition matrix, $\mathbf{H}$. Utilizing all steps into an unified objective, the update of $\mathbf{H}$ will guide the information distilling processes, achieving an ideal performance at last.

Optimization

Optimization strategy

The alternative optimization strategy is adopted to optimize the resultant objective. In specific, every unknown variable is solved and ensured to be globally optimal with the others fixed in each step, as shown in Eq. (10) - (18).

i) **Optimizing** $\{\mathbf{H}_p^{(1)}\}_{p=1}^m$. Given $\mathbf{K}_p$, $\mathbf{H}_p^{(2)}$, $\gamma_p^{(1)}$ and $\gamma_p^{(2)}$, the optimization reduces to

$$
\begin{aligned}
&\max_{\mathbf{H}_p^{(1)}} \ \operatorname{Tr}\left[\mathbf{H}_p^{(1)}\mathbf{H}_p^{(1)\top}\left(\gamma_p^{(1)}\mathbf{K}_p + \gamma_p^{(2)}\mathbf{H}_p^{(2)}\mathbf{H}_p^{(2)\top}\right)\right] \\
&s.t. \ \mathbf{H}_p^{(1)\top}\mathbf{H}_p^{(1)} = \mathbf{I}_{c_1}, \ \mathbf{H}_p^{(1)} \in \mathbb{R}^{n\times c_1}.
\end{aligned}
\tag{10}
$$

ii) **Optimizing** $\{\mathbf{H}_p^{(t)}\}_{t=2}^{s-1}$. Given $\mathbf{H}_p^{(t-1)}$, $\mathbf{H}_p^{(t+1)}$, $\gamma_p^{(t)}$ and $\gamma_p^{(t+1)}$, Eq. (9) w.r.t $\mathbf{H}_p^{(t)}$ can be solved as

$$
\begin{aligned}
\max_{\mathbf{H}_p^{(t)}} \ & \operatorname{Tr}\left[\mathbf{H}_p^{(t)}\mathbf{H}_p^{(t)\top}\left(\gamma_p^{(t)}\mathbf{H}_p^{(t-1)}\mathbf{H}_p^{(t-1)\top}\right.\right. \\
& \left.\left. + \gamma_p^{(t+1)}\mathbf{H}_p^{(t+1)}\mathbf{H}_p^{(t+1)\top}\right)\right] \\
s.t. \ & \mathbf{H}_p^{(t)\top}\mathbf{H}_p^{(t)} = \mathbf{I}_{c_t}, \ \mathbf{H}_p^{(t)} \in \mathbb{R}^{n\times c_t}.
\end{aligned}
\tag{11}
$$

iii) **Optimizing** $\mathbf{H}_p^{(s)}$. With fixed $\mathbf{H}_p^{(s-1)}$, $\mathbf{H}$, $\gamma_p^{(s)}$ and β_p, the optimization can be optimized as

$$
\begin{aligned}
&\max_{\mathbf{H}_p^{(s)}} \ \operatorname{Tr}\left[\mathbf{H}_p^{(s)}\mathbf{H}_p^{(s)\top}\left(\gamma_p^{(s)}\mathbf{H}_p^{(s-1)}\mathbf{H}_p^{(s-1)\top} + \beta_p\mathbf{H}\mathbf{H}^\top\right)\right] \\
&s.t. \ \mathbf{H}_p^{(s)\top}\mathbf{H}_p^{(s)} = \mathbf{I}_{c_s}, \ \mathbf{H}_p^{(s)} \in \mathbb{R}^{n\times c_s}.
\end{aligned}
\tag{12}
$$

iv) **Optimizing** $\mathbf{H}$. Given $\{\mathbf{H}_p^{(s)}\}_{p=1}^m$ and $\boldsymbol{\beta}$, $\mathbf{H}$ can be optimized via

$$
\begin{aligned}
&\max_{\mathbf{H}} \ \operatorname{Tr}\left[\mathbf{H}\mathbf{H}^\top\sum_{p=1}^{m}\left(\beta_p\mathbf{H}_p^{(s)}\mathbf{H}_p^{(s)\top}\right)\right] \\
&s.t. \ \mathbf{H}^\top\mathbf{H} = \mathbf{I}_k, \ \mathbf{H} \in \mathbb{R}^{n\times k}.
\end{aligned}
\tag{13}
$$

v) **Optimizing** $\boldsymbol{\gamma}^{(t)}$. $\boldsymbol{\gamma}^{(t)}$ are the weights across m views at t distilling stage. In the first stage, $\mathbf{K}_p$ and $\mathbf{H}_p^{(1)}$ are known. In $t = 2\cdots s$ stage, $\mathbf{H}_p^{(t-1)}$ and $\mathbf{H}_p^{(t+1)}$ are known. The optimization problem w.r.t. $\boldsymbol{\gamma}^{(t)}$ can be solved as

$$
\begin{aligned}
&\max_{\boldsymbol{\gamma}^{(t)}} \ \boldsymbol{\nu}^{(t)\top}\boldsymbol{\gamma}^{(t)} \\
&s.t. \ \nu_p^{(1)} = \operatorname{Tr}\left(\mathbf{K}_p\mathbf{H}_p^{(1)}\mathbf{H}_p^{(1)\top}\right) \\
&\qquad \nu_p^{(t)} = \operatorname{Tr}\left(\mathbf{H}_p^{(t-1)}\mathbf{H}_p^{(t-1)\top}\mathbf{H}_p^{(t)}\mathbf{H}_p^{(t)\top}\right) \\
&\qquad \boldsymbol{\gamma}^{(t)\top}\boldsymbol{\gamma}^{(t)} = 1, \gamma_p^{(t)} \geq 0, \boldsymbol{\gamma}^{(t)} \in \mathbb{R}^m.
\end{aligned}
\tag{14}
$$

vi) **Optimizing** $\boldsymbol{\beta}$. Given $\mathbf{H}$ and $\{\mathbf{H}_p^{(s)}\}_{p=1}^m$, $\boldsymbol{\beta}$ is able to be optimized as

$$
\begin{aligned}
&\max_{\boldsymbol{\beta}} \ \boldsymbol{\nu}^\top\boldsymbol{\beta} \\
&s.t. \ \nu_p = \operatorname{Tr}\left(\mathbf{H}\mathbf{H}^\top\mathbf{H}_p^{(s)}\mathbf{H}_p^{(s)\top}\right) \\
&\qquad \boldsymbol{\beta}^\top\boldsymbol{\beta} = 1, \beta_p \geq 0, \boldsymbol{\beta} \in \mathbb{R}^m.
\end{aligned}
\tag{15}
$$

The four optimization problems, from i) to iv), belong to one type and can be generalized as

$$
\begin{aligned}
&\max_{\mathbf{U}} \ \operatorname{Tr}\left(\mathbf{U}\mathbf{U}^\top\mathbf{B}\right) \\
&s.t. \ \mathbf{U}^\top\mathbf{U} = \mathbf{I}_c, \ \mathbf{U} \in \mathbb{R}^{n\times c},
\end{aligned}
\tag{16}
$$

which can be easily solved via singular value decomposition (SVD) on $\mathbf{B}$ [Wang et al. 2019]. With $\mathbf{B} = \mathbf{S}_c\boldsymbol{\Sigma}_c\mathbf{V}_c^\top$, where $\mathbf{S}_c \in \mathbb{R}^{n\times c}$, $\boldsymbol{\Sigma}_c \in \mathbb{R}^{c\times c}$ and $\mathbf{V}_c \in \mathbb{R}^{n\times c}$, the solution can be computed as $\mathbf{U} = \mathbf{S}_c\mathbf{V}_c^\top$. Meanwhile, the last two equations are equivalent to

$$
\max_{\boldsymbol{\alpha}} \ \boldsymbol{\nu}^\top\boldsymbol{\alpha} \ \ s.t. \ \boldsymbol{\alpha}^\top\boldsymbol{\alpha} = 1, \alpha_p \geq 0, \boldsymbol{\alpha} \in \mathbb{R}^m \tag{17}
$$

whose closed-form solution is

$$
\alpha_p = \nu_p / \left(\sum_{p=1}^{m}\nu_p^2\right)^{\frac{1}{2}}. \tag{18}
$$

A trick can be adopted in the above optimization process with defining two procedures, i.e. forwarding and back propagation. Forwarding indicates updating the variables in such

Algorithm 1: Hierarchical multiple kernel clustering

```
Input: kernel matrices $\{\mathbf{K}_p\}_{p=1}^{m}$, clustering number $k$
       and layer sizes $\{c_t\}_{t=1}^{s}$
Initialize: $\{\mathbf{H}_p^{(t)}\}_{p,t=1}^{m,s}$, $\mathbf{H}$, $\{\boldsymbol{\gamma}^{(t)}\}_{t=1}^{s}$ and $\boldsymbol{\beta}$
while not convergence do
    update $\{\mathbf{H}_p^{(t)}\}_{p,t=1}^{m,s}$ by solving Eq. (10) - (12)
    update $\mathbf{H}$ by solving Eq. (13)
    update $\{\boldsymbol{\gamma}^{(t)}\}_{t=1}^{s}$ by solving Eq. (14)
    update $\boldsymbol{\beta}$ by solving Eq. (15)
end
Output: partition matrix $\mathbf{H}$.
```

order: $\mathbf{H}_p^{(1)} \rightarrow \mathbf{H}_p^{(2)} \rightarrow \cdots \rightarrow \mathbf{H}_p^{(s)} \rightarrow \mathbf{H}$, i.e. blue arrows in Fig. 1(d), while back propagation refers to updating them in the inverse order, i.e. orange arrows in Fig. 1(d). We perform forwarding and back propagation alternately, which updates the variables more efficiently than only applying forwarding in whole optimization. At last, the optimization procedure is outlined in Algorithm 1.

Convergence and complexity analysis

For ease of expression, we formulate the objective in Eq. (9) as

$$\max_{\{\mathbf{H}_p^{(t)}\}_{t=1}^{s},\mathbf{H},\{\boldsymbol{\gamma}^{(t)}\}_{t=1}^{s},\boldsymbol{\beta}} \mathcal{J}(\{\mathbf{H}_p^{(t)}\}_{t=1}^{s},\mathbf{H},\{\boldsymbol{\gamma}^{(t)}\}_{t=1}^{s},\boldsymbol{\beta}) \tag{19}$$

It is obvious that

$$\begin{aligned}&\mathcal{J}(\{\mathbf{H}_p^{(t)}\}_{t=1}^{s\ (r)},\mathbf{H}^{(r)},\{\boldsymbol{\gamma}^{(t)}\}_{t=1}^{s\ (r)},\boldsymbol{\beta}^{(r)})\\ \leq&\mathcal{J}(\{\mathbf{H}_p^{(t)}\}_{t=1}^{s\ (r+1)},\mathbf{H}^{(r)},\{\boldsymbol{\gamma}^{(t)}\}_{t=1}^{s\ (r)},\boldsymbol{\beta}^{(r)}),\end{aligned} \tag{20}$$

where superscript r indicates the optimization at round r. Eq. (20) holds, because an optimal solution can be obtained when fixing the other variables. The similar inequality holds at each step, leading to

$$\begin{aligned}&\mathcal{J}(\{\mathbf{H}_p^{(t)}\}_{t=1}^{s\ (r)},\mathbf{H}^{(r)},\{\boldsymbol{\gamma}^{(t)}\}_{t=1}^{s\ (r)},\boldsymbol{\beta}^{(r)})\\ \leq&\mathcal{J}(\{\mathbf{H}_p^{(t)}\}_{t=1}^{s\ (r+1)},\mathbf{H}^{(r+1)},\{\boldsymbol{\gamma}^{(t)}\}_{t=1}^{s\ (r+1)},\boldsymbol{\beta}^{(r+1)}).\end{aligned} \tag{21}$$

Therefore, the objective monotonically increases at each round. Meanwhile, it is obvious that the whole optimization objective is upper-bounded. As a result, the alternative algorithm is guaranteed to converge to a local maximum.

For Eq. (10)-(13), SVD technique is adopted on matrices of size $\mathbb{R}^{n\times n}$, requiring $\mathcal{O}(n^3)$ complexity. Eq. (14) and Eq. (15) are calculated with multiplication of matrices and require $\mathcal{O}(n^2)$ and $\mathcal{O}(n)$ complexities, respectively. Overall, the computational complexity is $\mathcal{O}(n^3)$. By the way, all comparative methods in experiments are of $\mathcal{O}(n^3)$ complexity, guaranteeing the fairness of comparison.

Experiment

Experiment settings

Datasets. We evaluate the proposed algorithm on eight widely used MKC benchmark datasets, including *AR10P*[1], *BBCSport*[2], *CCV*[3], *Flower17*[4], *Flower102*[5], *Ionosphere*[6], *Heart*[7] and *Plant*[8]. Their details are shown in Table 1.

Table 1: Specifications of the used datasets.

Dataset	Samples	Number of Kernels	Clusters
AR10P	130	6	10
BBCSport	544	6	5
CCV	6773	3	20
Flower17	1360	7	17
Flower102	8189	4	102
Heart	270	13	2
Ionosphere	351	33	2
Plant	940	69	4

Comparative methods. Thirteen algorithms are chosen from the MKC literatures for comparison. Specifically, **A-MKKM** (*baseline*) obtains the consensus kernel by linearly combining all base kernels with the same weights and then performs kernel k-means on it. **SB-KKM** (*baseline*) performs kernel k-means on all kernels and selects the best one. We also select seven classical algorithms, including **CSRC** [Kumar, Rai, and III 2011], **MKKM** [Huang, Chuang, and Chen 2012], **RMSC** [Xia et al. 2014], **RMKC** [Zhou et al. 2015], **RMKKM** [Du et al. 2015], **MKKM-MR** [Liu et al. 2016] and **ONKC** [Liu et al. 2017]. Additionally, we choose three recently proposed methods in high-quality conferences and journals, i.e. **LFAM** [Wang et al. 2019], **SPC** [Kang et al. 2019], **MVCC** [Zhan et al. 2019] and **SPMKC** [Ren and Sun 2020]. Their source codes are publicly available and we directly use them without revision in comparison. Meanwhile, we open the source code of **HMKC** on *Github*[9].

Parameter settings. Only the sizes of the intermediary matrices, $\{\mathbf{H}_p^{(t)}\}_{t=1}^{s}$ are supposed to be given manually. In the following, we consider two experimental settings. The first is called HMKC-1 with employing 1-layer of intermediary matrices, $\{\mathbf{H}_p^{(1)}\}_{p=1}^{m} \in \mathbb{R}^{n\times c}$, where c is searched from $[2k, 3k, \cdots, 20k]$. While the second one is named HMKC-2 with employing 2-layer of intermediary matrices, $\{\mathbf{H}_p^{(1)}\}_{p=1}^{m} \in \mathbb{R}^{n\times c_1}$ and $\{\mathbf{H}_p^{(2)}\}_{p=1}^{m} \in \mathbb{R}^{n\times c_2}$, in which $c_1 \geq c_2$ and c_1, c_2 are grid searched from $[2k, 3k, \cdots, 20k]$. In specific, the search region is restricted to $[2k, 3k, \cdots, 10k]$ for *AR10P* since it has relatively small number of samples in each cluster. Note that the aforementioned k refers to the number of clusters. For the other comparative methods, we perform the grid search on the parameters recommended in corresponding papers and report the best results.

[1] featureselection.asu.edu/old/datasets.php

[2] mlg.ucd.ie/datasets/segment.html

[3] www.ee.columbia.edu/ln/dvmm/CCV/

[4] www.robots.ox.ac.uk/~vgg/data/flowers/17/

[5] www.robots.ox.ac.uk/~vgg/data/flowers/102/

[6] archive.ics.uci.edu/ml/datasets/Ionosphere

[7] archive.ics.uci.edu/ml/datasets/Statlog+(Heart)

[8] www.imageclef.org/2013/plant

[9] https://github.com/liujiyuan13/HMKC-code_release

Table 2: The performance comparison of MSC algorithms in recent literature.

	Algorithm	AR10P	BBCSport	CCV	Flower17	Flower102	Heart	Ionosphere	Plant
ACC	A-MKKM	38.46	65.99	19.74	51.03	27.29	82.22	61.25	61.70
	SB-KKM	49.23	76.65	20.08	42.06	33.13	76.30	70.09	51.60
	CSRC	36.92	67.65	23.40	51.69	35.19	80.37	75.78	54.57
	MKKM	37.69	66.36	18.01	45.37	21.96	53.33	63.82	56.38
	RMSC	34.62	86.03	16.29	52.57	32.97	83.33	84.62	55.53
	RMKC	43.85	66.36	19.74	52.35	33.55	82.22	66.10	61.70
	RMKKM	32.31	53.13	17.11	53.16	29.61	76.30	65.81	55.32
	MKKM-MR	43.08	66.36	22.47	60.00	40.27	83.33	61.25	62.87
	ONKC	48.46	67.65	23.12	59.85	41.26	83.70	63.82	*64.89*
	LFAM	46.92	72.61	26.66	59.63	44.61	82.22	68.09	62.77
	SPC	38.46	83.09	-	57.50	-	75.93	71.23	60.43
	MVCC	43.85	74.26	22.10	51.47	37.23	82.96	55.56	55.64
	SPMKC	54.62	40.81	13.54	35.81	-	57.04	72.93	53.94
	HMKC-1	*56.15*	*89.52*	*36.57*	*66.91*	*46.84*	**86.67**	*86.32*	64.68
	HMKC-2	**60.00**	**90.99**	**37.37**	**71.18**	**50.32**	*86.30*	**86.89**	**67.02**
NMI	A-MKKM	34.57	53.92	17.16	50.19	46.32	32.40	3.29	26.82
	SB-KKM	51.44	59.39	17.73	45.14	48.99	20.49	10.35	17.18
	CSRC	36.90	55.41	20.96	52.63	53.73	28.53	15.99	21.82
	MKKM	38.92	54.67	15.52	45.35	42.30	0.02	5.12	20.02
	RMSC	29.70	*73.89*	13.79	56.35	53.36	34.51	36.52	23.83
	RMKC	44.38	54.30	17.16	50.42	49.74	32.40	7.02	26.82
	RMKKM	27.96	28.48	12.54	53.31	48.55	20.49	9.33	19.71
	MKKM-MR	42.62	54.67	18.62	57.11	57.38	35.22	3.29	28.29
	ONKC	51.86	54.74	19.19	56.85	57.39	36.22	31.13	31.13
	LFAM	47.11	54.99	19.85	57.83	57.58	32.40	1.15	27.64
	SPC	39.02	65.49	-	61.32	-	19.67	0.35	29.65
	MVCC	38.56	51.52	15.97	56.26	52.35	34.54	0.63	20.85
	SPMKC	*58.39*	7.31	8.11	40.78	-	10.95	1.64	19.28
	HMKC-1	54.46	72.73	*31.64*	*63.37*	*60.40*	*42.98*	*39.11*	*34.11*
	HMKC-2	**60.32**	**76.72**	**32.71**	**65.45**	**62.47**	**43.40**	**40.55**	**36.43**

Experiment results

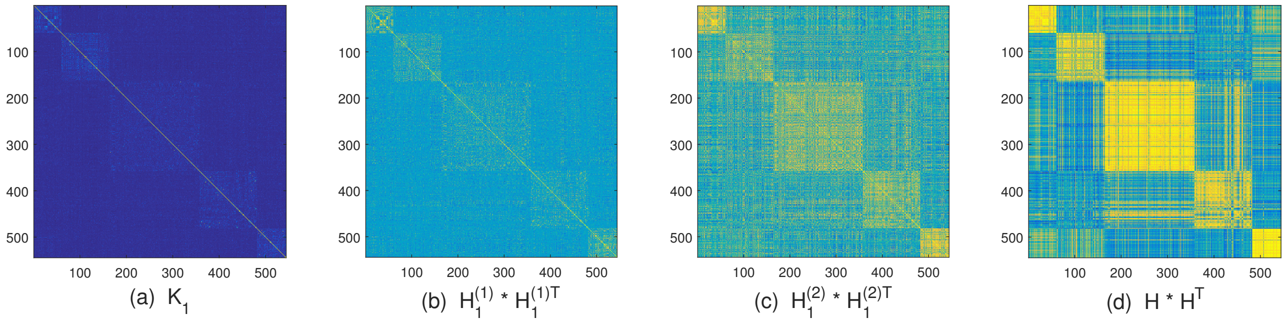
In order to show the effectiveness and the superiority over algorithms in MKC literatures, we compare the proposed method with the other thirteen ones on eight widely used benchmark datasets. Correspondingly, two common metrics, including accuracy (ACC) and normalized mutual information (NMI), are adopted to measure the performances. The results are shown in Table 2, where '-' indicates the results of corresponding algorithms are unavailable due to the long execution time. We have the following observations:

Effectiveness. Some MKC algorithms perform worse than the baselines, A-MKKM and SB-KKM. For instance, MKKM has 45.37% accuracy on *Flower17* which is 5.66% lower than A-MKKM. However, both the proposed HMKC-1 and HMKC-2 outperform the baselines consistently across all datasets. This illustrates the proposed algorithm is effective in exploring the complementary information of multiple kernels to boost clustering performance.

Superiority. Although, comparing with the best of the second best results, HMKC-1 shows little performance drops, such as 0.21% accuracy on *Plant*, 3.93% NMI on *AR10P* and 1.16% NMI on *BBCSport*, it shows obvious promotion on the other datasets. Moreover, HMKC-2 consistently and largely outperforms the other algorithms across all datasets. Especially, the proposed algorithm exhibits excellent performances on *CCV*, *Flower17* and *Flower102*, where around **5/10% increases** are obtained. In addition, SPC is of high complexity, and its results on big datasets, such as *CCV* and *Flower102*, are unavailable in limited execution time. This indicates the superiority of our methods again. Overall, the proposed method outperforms the recent algorithms by a large margin, achieving the state-of-the-art performance.

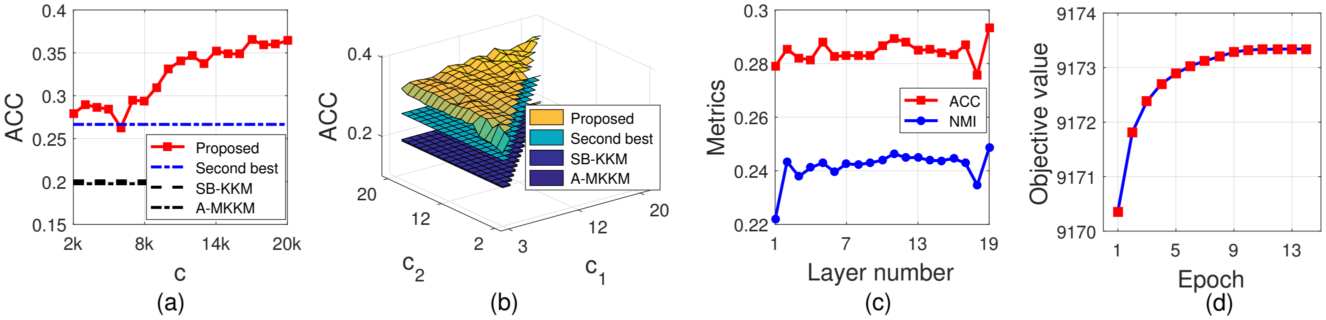
Validation on intermediary matrix. CRSC[Kumar, Rai, and III 2011] also use base partitions of each view to construct the final partition with updating them concurrently. But it fails to relax the base partitions into $\mathbb{R}^{n\times c}$ where $c > k$. HMKC-1 exceeds it by 19.23% on *AR10P*, 21.87% on *BBCSport*, 13.17% on *CCV*, 15.22% on *Flower17*, 11.65% on *Flower102*, 6.30% on *Heart*, 10.54% on *Ionosphere* and 10.11% on *Plant* respect to accuracy. Meanwhile, the first kernel matrix $\mathbf{K}_1$ of *CCV*, the corresponding intermediary $\mathbf{H}_1$, $\mathbf{H}_2$ and the final partition $\mathbf{H}$ in the proposed HMKC-2 are visualized in Fig. 3. It can be seen that a more and more clear clustering structure is presented along with the clustering process. The two observations validate that adopting the intermediary matrices of decreasing sizes is able to keep more advantageous details for clustering.

Multiple layers. Comparing the results in Table 2, it can be observed that HMKC-2 has consistent performance im-



(a) K_1 (b) $H_1^{(1)} * H_1^{(1)T}$ (c) $H_1^{(2)} * H_1^{(2)T}$ (d) $H * H^T$

Figure 3: Visualization of kernel matrix, the proposed intermediary matrices and final partition of in HMKC-2 model.



(a) (b) (c) (d)

Figure 4: (a). ACC variation of the proposed HMKC-1 model at $2k \leq c \leq 20k$; (b). ACC variation of the proposed *2-layer* model at $2k \leq c \leq 20k$; (c). The objective value by epoch at $c_1 = 20k$ and $c_2 = 2k$; (d). The number of epochs at different c_1 and c_2. Note that, k represents the number of clusters and legend "Second best" refers to **the best result of the second best algorithm** in recent literature.

provements over all datasets and metrics. Nevertheless, we construct a HMKC-1 model with $c = 2k$. By adding a layer intermediary matrices one k bigger each step, 19 models are built finally. These models are tested on *CCV*, and the results are presented in Fig. 4(c). The performance shows a relatively large increase, around 2% in NMI at first. Although a sudden drop is observed at 18 layers, it rises up steadily when increasing the number of layers. In all, the two above observations illustrate that adopting multiple layers of intermediary matrices in the proposed algorithm will increase the performance.

Parameter study and convergence

In order to investigate the parameter stability of the proposed algorithm, we perform grid search on *CCV*. Specifically, we vary parameter c of HMKC-1 model from $2k$ to $20k$. Corresponding accuracies are shown in Fig. 4(a). It can be observed that the proposed algorithm consistently outperforms the baselines, including A-MKKM and SB-KKM, and the best result of the second best algorithm in a wide parameter range. Also, the performance on *CCV* steadily increases while enlarging the size of intermediary matrices, verifying the effectiveness of the proposed intermediary matrix again. Fig. 4(b) presents the performance variation of HMKC-2 model when parameter c_1 and c_2 vary from $3k$ to $20k$ and $2k$ to $20k$, respectively. We can find the proposed algorithm exhibits large performance promotions over the baselines and the second best algorithm across all parameter ranges. Similar to HMKC-1 model, the accuracy also rises up with employing larger c_1 and c_2. This well verifies that better performances can be obtained by designing larger intermediary matrices to keep more advantageous partition details in clustering. Fig. 4(c) shows the performances when increasing the layer number of intermediary matrices. Although some improvements are observed on bigger layer numbers, we recommend to use the HMKC-2 model in clustering tasks for its aforementioned stability and the relatively low complexity compared with models with 3-19 layers.

Fig. 4(d) presents the variation of objective value on *CCV* at $c_1 = 20k$ and $c_2 = 2k$. It can be seen that the objective monotonically increases, and quickly converges with 14 iterations. In fact, the algorithm converges in less than 15 iterations across all datasets in most cases, making it practical in real-word applications.

Conclusion

Both early-fusion and late-fusion MKC methods suffer from information loss when encoding the clustering details from kernels or graphs with size $\mathbb{R}^{n\times n}$ to partition matrix with size $\mathbb{R}^{n\times k}$. To address this issue, We propose a hierarchical algorithm. The proposed method is tested on benchmark datasets and outperforms the comparative algorithms, showing state-of-the-art performance by a large margin. We will explore the relationship between eigenvalues of kernel matrix with sizes of the intermediary matrices.

Acknowledgment

We would like to thank the support from National Key R & D Program of China (project no. 2020AAA0107100), Education Ministry-China Mobile Research Funding (project no. MCM20170404) and Natural Science Foundation of China (project no. 62006237).

References

Dhillon, I. S.; Guan, Y.; and Kulis, B. 2004. Kernel k-means: spectral clustering and normalized cuts. In Kim, W.; Kohavi, R.; Gehrke, J.; and DuMouchel, W., eds., *Proceedings of the Tenth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, Seattle, Washington, USA, August 22-25, 2004*, 551–556. ACM.

Du, L.; Zhou, P.; Shi, L.; Wang, H.; Fan, M.; Wang, W.; and Shen, Y. 2015. Robust Multiple Kernel K-means Using L21-Norm. In *Proceedings of the Twenty-Fourth International Joint Conference on Artificial Intelligence, IJCAI 2015, Buenos Aires, Argentina, July 25-31, 2015*, 3476–3482.

Gönen, M.; and Margolin, A. A. 2014. Localized Data Fusion for Kernel k-Means Clustering with Application to Cancer Biology. In *Advances in Neural Information Processing Systems 27: Annual Conference on Neural Information Processing Systems 2014, December 8-13 2014, Montreal, Quebec, Canada*, 1305–1313.

Huang, H.; Chuang, Y.; and Chen, C. 2012. Multiple Kernel Fuzzy Clustering. *IEEE Trans. Fuzzy Systems* 20(1): 120–134.

Kalman, D. 1996. A Singularly Valuable Decomposition: The SVD of a Matrix. *College Mathematics Journal* 27(1): 2–23.

Kang, Z.; Peng, C.; Cheng, Q.; and Xu, Z. 2018. Unified Spectral Clustering With Optimal Graph. In *Proceedings of the Thirty-Second AAAI Conference on Artificial Intelligence, (AAAI-18), New Orleans, Louisiana, USA, February 2-7, 2018*, 3366–3373.

Kang, Z.; Xu, H.; Wang, B.; Zhu, H.; and Xu, Z. 2019. Clustering with Similarity Preserving. *Neurocomputing* 365: 211 – 218.

Kloft, M.; Brefeld, U.; Sonnenburg, S.; and Zien, A. 2011. l_p-Norm Multiple Kernel Learning. *J. Mach. Learn. Res.* 12: 953–997.

Kloft, M.; Rückert, U.; and Bartlett, P. L. 2010. A Unifying View of Multiple Kernel Learning. In Balcázar, J. L.; Bonchi, F.; Gionis, A.; and Sebag, M., eds., *Machine Learning and Knowledge Discovery in Databases, European Conference, ECML PKDD 2010, Barcelona, Spain, September 20-24*, volume 6322, 66–81.

Kumar, A.; Rai, P.; and III, H. D. 2011. Co-regularized Multi-view Spectral Clustering. In *Advances in Neural Information Processing Systems 24: 25th Annual Conference on Neural Information Processing Systems 2011. Proceedings of a meeting held 12-14 December 2011, Granada, Spain*, 1413–1421.

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 (TKDE)* .

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, February 12-17, 2016, Phoenix, Arizona, USA*, 1888–1894.

Liu, X.; Li, M.; Tang, C.; Xia, J.; Xiong, J.; Liu, L.; Kloft, M.; and Zhu, E. 2020b. Efficient and Effective Regularized Incomplete Multi-view Clustering. *IEEE Transactions on Pattern Analysis and Machine Intelligence* 1–1. doi: 10.1109/TPAMI.2020.2974828.

Liu, X.; Zhou, S.; Wang, Y.; Li, M.; Dou, Y.; Zhu, E.; and Yin, J. 2017. Optimal Neighborhood Kernel Clustering with Multiple Kernels. In *Proceedings of the Thirty-First AAAI Conference on Artificial Intelligence, February 4-9, 2017, San Francisco, California, USA*, 2266–2272.

Liu, X.; Zhu, E.; Liu, J.; Hospedales, T. M.; Wang, Y.; and Wang, M. 2020c. SimpleMKKM: Simple Multiple Kernel K-means. *ArXiv* abs/2005.04975. URL https://arxiv.org/abs/2005.04975.

Liu, X.; Zhu, X.; Li, M.; Wang, L.; Tang, C.; Yin, J.; Shen, D.; Wang, H.; and Gao, W. 2019. Late Fusion Incomplete Multi-View Clustering. *IEEE Trans. Pattern Anal. Mach. Intell.* 41(10): 2410–2423.

Long, B.; Yu, P. S.; and Zhang, Z. M. 2008. A General Model for Multiple View Unsupervised Learning. In *Proceedings of the SIAM International Conference on Data Mining, SDM 2008, April 24-26, 2008, Atlanta, Georgia, USA*, 822–833.

Peng, X.; Huang, Z.; Lv, J.; Zhu, H.; and Zhpu, J. T. 2019. COMIC: Multi-view Clustering Without Parameter Selection. In Chaudhuri, K.; and Salakhutdinov, R., eds., *Proceedings of the 36th International Conference on Machine Learning, 9-15 June 2019, Long Beach, California, USA*, volume 97, 5092–5101.

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* 1–13.

Ren, Z.; Yang, S. X.; Sun, Q.; and Wang, T. 2020. Consensus Affinity Graph Learning for Multiple Kernel Clustering. *IEEE Transactions on Systems, Man, and Cybernetics* 1–12.

Schölkopf, B.; Smola, A. J.; and Müller, K. 1998. Nonlinear Component Analysis as a Kernel Eigenvalue Problem. *Neural Computation* 10(5): 1299–1319.

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 *Proceedings of the Twenty-Eighth International Joint Conference on Artificial Intelligence, IJCAI 2019, Macao, China, August 10-16, 2019*, 3778–3784.

Wen, J.; Zhang, Z.; Zhang, Z.; Fei, L.; and Wang, M. 2020. Generalized Incomplete Multiview Clustering With Flexible Locality Structure Diffusion. *IEEE Transactions on Systems, Man, and Cybernetics* 1–14.

Xia, R.; Pan, Y.; Du, L.; and Yin, J. 2014. Robust Multi-View Spectral Clustering via Low-Rank and Sparse Decomposition. In *Proceedings of the Twenty-Eighth AAAI Conference on Artificial Intelligence, July 27 -31, 2014, Québec City, Québec, Canada*, 2149–2155.

Zhan, K.; Nie, F.; Wang, J.; and Yang, Y. 2019. Multiview Consensus Graph Clustering. *IEEE Trans. Image Processing* 28(3): 1261–1270.

Zhang, Z. 2015. The Singular Value Decomposition, Applications and Beyond. *CoRR* abs/1510.08532.

Zhou, P.; Du, L.; Shi, L.; Wang, H.; and Shen, Y. 2015. Recovery of Corrupted Multiple Kernels for Clustering. In *Proceedings of the Twenty-Fourth International Joint Conference on Artificial Intelligence, IJCAI 2015, Buenos Aires, Argentina, July 25-31, 2015*, 4105–4111.

Zhou, S.; Liu, X.; Liu, J.; Guo, X.; Zhao, Y.; Zhu, E.; Zhai, Y.; Yin, J.; and Gao, W. 2020. Multi-View Spectral Clustering with Optimal Neighborhood Laplacian Matrix. In *The Thirty-Fourth AAAI Conference on Artificial Intelligence, AAAI 2020, New York, NY, USA, February 7-12, 2020*, 6965–6972. AAAI Press.