Information Fusion 100 (2023) 101902

Contents lists available at ScienceDirect

Information Fusion

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

Simple multiple kernel k-means with kernel weight regularization ☆

Miaomiao Li [a,1], Yi Zhang [b,1], Suyuan Liu [b], Zhe Liu [c,d,*], Xinzhong Zhu [e,**]

[a] College of Electronic Information and Electrical Engineering, Changsha University, Changsha, 410073, Hunan, China
[b] School of Computer, National University of Defense Technology, Changsha, 410073, Hunan, China
[c] Zhejiang Lab, Hangzhou, 311121, Zhejiang, China
[d] College of Computer Science and Technology, Nanjing University of Aeronautics and Astronautics, Nanjing, 210016, Jiangsu, China
[e] College of Mathematics, Physics and Information Engineering, Zhejiang Normal University, Jinhua, 321004, Zhejiang, China

ARTICLE INFO

Keywords:
Multiple kernel clustering
Multi-view clustering
Kernel k-means

ABSTRACT

Multiple kernel clustering (MKC) aims to determine the optimal kernel from several pre-computed basic kernels. Most of MKC algorithms follow a common assumption that the optimal kernel is linearly combined by basic kernels. Based on a min–max framework, a newly proposed MKC method termed simple multiple kernel k-means can acquire a high-quality unified kernel. Although it has achieved promising clustering performance, we have observed that it cannot benefit from any regularization or prior knowledge, thus the learned kernel weight coefficients may be seriously sparse or over-selected. To tackle this issue, we have developed a new algorithm termed simple multiple kernel k-means with kernel weight regularization (SMKKM-KWR), where we introduce average coefficients to avoid too sparse kernel weights. Specially, we add the average kernel coefficients as a regularization term to prevent the learned weight coefficients being far from the average values. After that, an efficient optimization strategy is proposed to solve the new resultant problem. By this way, the unified partition can learn clustering structure from fusion information of all the kernel views, with the goal to reach better clustering performance. Extensive experiments on nine benchmark datasets and four large-scale datasets have demonstrated the effectiveness and efficiency of the proposed algorithm.

1. Introduction

Non-linearly separable data pose a challenge for clustering techniques [1–5]. Kernel K-means is a popular method that can handle such data, but it depends on the choice of the kernel function and related parameters, such as the bandwidth of Gaussian kernels [6–11]. A solution to this problem is multiple kernel clustering (MKC), which learns an optimal kernel from a set of pre-computed basic kernels with different characteristics [12–18]. MKC generally follows the principle of multiple kernel learning and assumes that the optimal kernel is a linear combination of the base kernels.

A common approach in MKC is to jointly optimize the kernel weights and the clustering assignment in one unified framework. Du et al. [19] aim to enhance the robustness of clustering by using a different error measure than the squared error, such as the $\ell_{2,1}$-norm based error. Liu et al. [20] seek to increase the variety of chosen kernels by adding a regularization term that favors kernels with more distinctive information. Li et al. [21] exploit the sample-specific information to localize the base kernels and then obtains the partition matrix by locally maximizing the kernel alignment. Liu et al. [22] learn a neighbor kernel from the combination of basic kernels and use it for clustering. The work in [23] tries integrating different fusion methods corresponding to different strategic stages. The work in [24] tries dealing with the scalable kernel k-means problem with randomized sketching.

Simple multiple kernel k-means (SMKKM) is a novel method that uses a minimization–maximization learning criterion [25]. It aims to optimize the kernel alignment by jointly minimizing the kernel weights and maximizing the partition matrix. This objective is then converted to a minimization problem that can be solved by a reduced gradient descent method. The authors provide theoretical insights into the convexity and differentiability of the objective function, which guarantees the global optimality of the solution. SMKKM outperforms most MKC algorithms that rely on alternating optimization and often converge to local optima. It achieves state-of-the-art clustering performance on

☆ This work is supported by the National Key R&D Program of China 2020AAA0107703, and the Research Initiation Project of Zhejiang Lab 2022PD0AC02.
* Corresponding author at: Zhejiang Lab, Hangzhou, 311121, Zhejiang, China.
** Corresponding author.
E-mail addresses: miaomiaolinudt@gmail.com (M. Li), zhangy@nudt.edu.cn (Y. Zhang), suyuanliu@nudt.edu.cn (S. Liu), zhe.liu@zhejianglab.edu.cn, zhe.liu@nuaa.edu.cn (Z. Liu), zxz@zjnu.edu.cn (X. Zhu).
[1] First authors with equal contribution.

https://doi.org/10.1016/j.inffus.2023.101902
Received 6 April 2023; Received in revised form 21 May 2023; Accepted 27 June 2023
Available online 1 July 2023

several benchmark datasets. A localized variant of SMKKM is also developed to better exploit the local structure of the data. In common with other MKKM approaches, SMKKM and its variants imposes a sum-of −1 constraint on the kernel combination coefficients to avoid trivial solutions. However, the existing objective formula still inevitably yields sparse weight coefficients as the optimization ends, which means that some kernels are much less useful in multi-view learning than the other. It is not reasonable to judge the importance of a view only by its corresponding loss term. It is expected that the fusion yields a common representation that makes full and fair use of the information from all kernels.

To solve the above problem, a simple way is to replace the existing adaptive weighting strategy by directly using average kernels. Nevertheless, considering the qualitative differences between the kernels, it is necessary to measure the merits of the kernels by different weights and to balance their impact on the common kernel. In this paper, we propose Simple Multiple Kernel K-Means with Kernel Weight Regularization (SMKKM-KWR) to regulate the sparsity of the adaptive weights so that each kernel can participate effectively in the learning process. Specifically, a kernel weight regularization is introduced in the original SMKKM method, which align adaptive coefficient to a uniform vector. The alignment module helps more kernel to contribute the combined kernel. With the corresponding parameter on regularization, we can easily balance the average kernel and the loss-adaptive kernel. To solve the resulting optimization problem, we adopt a singular value decomposition based method to alternately update the partition matrix, and a reduced gradient descent method to update the kernel weights. We prove that our algorithm converges to a global optimum under mild conditions. Extensive experiments compared our algorithm with nine state-of-the-art MKC algorithms are conducted on nine benchmark datasets and four large-scale datasets. The experiments have demonstrated that our proposed SMKKM-KWR reaches superior clustering performance and robustness.

Our contributions can be summarized as:

- A novel kernel weight alignment module is proposed to encourage more kernels participating in the combined kernel learning. The sparsity of kernel weight can be controlled with the inductive parameter, which helps balance between the adaptive kernel and the average kernel.
- To solve the resulting optimization problem, we have designed an efficient and effective algorithm with the reduced gradient descent method. Moreover, we theoretically prove that the proposed algorithm is able to converge to the global optimal results under mild conditions.
- Carefully designed experiments conducted on nine benchmark datasets and four large-scale datasets compared with a set of MKKM algorithms demonstrate the effectiveness of SMKKM-KWR.

2. Related work

This section provides a brief overview of multiple kernel k-means and simple multiple kernel k-means, both of which are closely connected to our method. In the manuscript, n, m, k denotes the number of samples, views, and clusters, respectively.

2.1. Multiple kernel k-means clustering

For the given data matrix $\mathbf{X} \in \mathbb{R}^{n\times d}$, x_i is the ith row, corresponding to the ith sample point with dimension of d. The goal of k-means is to divide the n sample points into k clusters based on the distance relationship. Although k-means works well in dividing linearly divisible data, it is difficult to handle complex and high-dimensional data. To solve the above problem, mapping these linearly indistinguishable data sets into the Hilbert space $\mathcal{H}$ with mapping function ϕ is an effective approach. Note that the dimensionality of the data after mapping can be very large and may even reach infinite dimensions. Kernel k-means therefore employs the kernel trick $K_{ij} = \phi(x_i)^\top \phi(x_j)$ to compute the kernel matrix $\mathbf{K}$, addressing the computational risk associated with potentially high-dimensional mappings. Based on the predefined kernel matrix $\mathbf{K}$, the objective function of Kernel k-means can be written as follows:

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

where $\mathbf{H} \in \mathbb{R}^{n\times k}$ is the clustering partition matrix, $\mathbf{I}_n \in \mathbb{R}^{n\times n}$ and $\mathbf{I}_k \in \mathbb{R}^{k\times k}$ are both identity matrices.

Extended to multiple views, $\mathbf{K}_p$ represents the predefined kernel matrix on the pth view. Multi-kernel k-means aims to learn the exact partition of data points based on the optimal combined kernel matrix. A simple and intuitive strategy is to average the predefined kernels on each view by adding them together to obtain the combined kernel. This strategy of having all kernels participate uniformly in the learning of the clustering matrix ignores differences in the quality of the different kernels. The strategy of allowing all kernels to participate uniformly in the learning of the clustering partition matrix ignores differences in the quality of the different kernels. To measure the quality of individual kernels, existing MKKM algorithms typically weight the kernels to obtain the final combinatorial kernel $\mathbf{K}_{\boldsymbol{\gamma}}$. specifically, $\boldsymbol{\gamma} = \{\gamma_1, \ldots, \gamma_m\}$ represents the weights of the different kernels, and $\mathbf{K}_{\boldsymbol{\gamma}} = \sum_{p=1}^{m} \gamma^2 \mathbf{K}_p$ consists of a linear combination of all base kernels $\{\mathbf{K}_1, \ldots, \mathbf{K}_m\}$. Furthermore, the pre-specified weight coefficient $\boldsymbol{\gamma}$ is not flexible enough. MKKM unifies the learning of $\boldsymbol{\gamma}$ with the optimization of the clustering partition matrix in a unified framework as:

$$\begin{aligned} &\min_{\mathbf{H},\boldsymbol{\gamma}} \ \mathrm{Tr}\left(\mathbf{K}_{\boldsymbol{\gamma}}(\mathbf{I}_n - \mathbf{H}\mathbf{H}^\top)\right) \\ &s.t. \ \ \mathbf{H} \in \mathbb{R}^{n\times k}, \ \mathbf{H}^\top\mathbf{H} = \mathbf{I}_k, \\ &\quad \boldsymbol{\gamma}^\top \mathbf{1}_m = 1, \ \gamma_q \geq 0, \end{aligned} \tag{2}$$

The constraint of $\boldsymbol{\gamma}^\top \mathbf{1}_m = 1$ avoids trivial solutions and ensures that all kernels can contribute to the learning of the cluster partition matrix.

Optimizing the two variables together in Eq. (2) is a hard problem, which is not jointly convex. Therefore, an alternative coordinate descent algorithm is often employed to solve the above problems. Specifically, when optimizing $\mathbf{H}$ with fixed $\boldsymbol{\gamma}$, the original optimization problem can be transformed into a common kernel k-means problem, which can be easily solved through corresponding solutions. When optimizing $\boldsymbol{\gamma}$ with fixed $\mathbf{H}$, it is easy to obtain an analytical solution to update $\boldsymbol{\gamma}$. After alternately optimizing the two variables until the objective convergence, the classical k-means are performed on the assignment matrix $\mathbf{H}$ to obtain the final clustering result.

2.2. Simple multiple kernel k-means clustering

As described in the previous section, the existing MKKM methods solve the joint variable optimization problem by the block coordinate descent algorithm. The algorithm that optimizing one variable with others fixed leads to the objective function value falling into a local optimum. Liu et al. [20] solves the above problem by adding regularization terms. However, the additional introduced hyperparameters are difficult to determine and need to be searched in a predefined range. In construct, SMKKM proposes a new solution. Specifically, SMKKM uses the min–max framework to replace the original MKKM framework and proposes an efficient optimization algorithm to obtain a stable global optimal solution. The objective function of SMKKM is as follows:

$$\begin{aligned} &\min_{\boldsymbol{\gamma}} \max_{\mathbf{H}} \ \mathrm{Tr}(\mathbf{K}_{\boldsymbol{\gamma}}\mathbf{H}\mathbf{H}^\top), \\ &s.t. \ \ \mathbf{H} \in \mathbb{R}^{n\times k}, \ \mathbf{H}^\top\mathbf{H} = \mathbf{I}_k, \ \boldsymbol{\gamma}^\top \mathbf{1}_m = 1, \ \gamma_q \geq 0, \end{aligned} \tag{3}$$

By requiring the kernel matrices with lower alignment values to acquire greater combination weights, SMKKM can significantly enhance the diversity of the multiple kernel information in Eq. (3). SMKKM

exhibits the most advanced performance in a range of applications in addition to the theoretical results. However, the kernel weight coefficients of SMKKM are still inevitably too sparse, resulting in some kernels playing a much smaller role in multi-kernel combinations than others.

3. Proposed algorithm

The aforementioned SMKKM is empirically verified to be effective, and has promising performance in many applications. However, Upon observation, it can be noted that the algorithm does not facilitate the inclusion of prior knowledge, which could be critical to improve clustering performance in practical applications.

To take full advantage of prior knowledge, we propose a novel SMKKM-KWR which takes the average as a regularization term. As a result, the objective optimization of our proposed algorithm is as follows,

$$\min_{\boldsymbol{\gamma}\in\Delta}\max_{\mathbf{H}\in\Gamma}\ \mathrm{Tr}(\mathbf{K}_{\boldsymbol{\gamma}}\mathbf{H}\mathbf{H}^{\top})+\lambda\|\boldsymbol{\gamma}-\boldsymbol{\gamma}_0\|_2^2, \tag{4}$$

where $\Gamma=\{\mathbf{H}\in\mathbb{R}^{n\times k}|\mathbf{H}^{\top}\mathbf{H}=\mathbf{I}_k\}$ and $\Delta=\{\boldsymbol{\gamma}\in\mathbb{R}^m|\boldsymbol{\gamma}^{\top}\mathbf{1}_m=1,\ \boldsymbol{\gamma}\geq 0\}$.

According to the analysis in [25], the optimization in Eq. (4) cannot be directly solved by coordinate descent optimization, which is widely adopted in MKKM. This is because the entire objective function cannot be proven to decrease monotonically in such optimization mode. Instead, we have developed an efficient gradient descent based method to optimize it. To do so, we firstly transform the Eq. (4) into the following equivalence,

$$\min_{\boldsymbol{\gamma}\in\Delta}\ \mathcal{G}(\boldsymbol{\gamma}), \tag{5}$$

with

$$\mathcal{G}(\boldsymbol{\gamma})=\left\{\max_{\mathbf{H}\in\Gamma}\ \mathrm{Tr}\left(\mathbf{K}_{\boldsymbol{\gamma}}\mathbf{H}\mathbf{H}^{\top}\right)+\lambda\|\boldsymbol{\gamma}-\boldsymbol{\gamma}_0\|_2^2\right\}. \tag{6}$$

By this way, the min–max problem in Eq. (4) is equivalently transformed into a minimization one. The objective function of this optimization is a value function that depends on the optimization of $\mathbf{H}$.

Significantly, it can be proven that $\mathcal{G}(\boldsymbol{\gamma})$ in Eq. (5) is differentiable, which will be demonstrated below.

Theorem 1. *$\mathcal{G}(\boldsymbol{\gamma})$ in Eq. (5) is differentiable.*

Proof. For any given $\boldsymbol{\gamma}\in\Delta$, the optimization of $\mathcal{G}(\boldsymbol{\gamma})$ w.r.t. $\boldsymbol{\gamma}$ in Eq. (6) reduces to $\max_{\mathbf{H}\in\Gamma}\ \mathrm{Tr}\left(\mathbf{K}_{\boldsymbol{\gamma}}\mathbf{H}\mathbf{H}^{\top}\right)$, which has the global optimal solutions though it is not convex. As a result, $\mathcal{G}(\boldsymbol{\gamma})$ in Eq. (5) is differentiable according to Theorem 4.1 in [26]. Further, the pth component of its gradient is calculated as $\frac{\partial\mathcal{G}(\boldsymbol{\gamma})}{\partial\gamma_p}=2\gamma_p\mathrm{Tr}\left(\mathbf{H}^{*\top}\tilde{\mathbf{K}}_p\mathbf{H}^{*}\right)+2\lambda(\gamma_p-\gamma_{0p})$, where $\mathbf{H}^{*}$ is the global optimal solution of Eq. (6) with a given $\boldsymbol{\gamma}$, and γ_{0p} is the p-component of $\boldsymbol{\gamma}_0$. □

Based on Theorem 1, we then show how to compute the reduced gradient of $\mathcal{G}(\boldsymbol{\gamma})$ w.r.t. $\boldsymbol{\gamma}$, and adopt the gradient descent method to optimize Eq. (5).

3.1. Calculating the reduced gradient and the whole optimization

We solve the resultant optimization problem with the reduced gradient descent method in Eq. (5). Firstly, we can calculate the gradient of $\mathcal{G}(\boldsymbol{\gamma})$. Then, we update $\boldsymbol{\gamma}$ with a descent direction while keeping the constraints on $\boldsymbol{\gamma}$.

To achieve this goal, we start by computing the reduced gradient using the method in [27] to handle the equality constraint. Make γ_u a non-zero component of $\boldsymbol{\gamma}$ and $\bigtriangledown\mathcal{G}(\boldsymbol{\gamma})$, which represents the reduced gradient of $\mathcal{G}(\boldsymbol{\gamma})$. The qth $(1\leq q\leq m)$ element of $\bigtriangledown\mathcal{G}(\boldsymbol{\gamma})$ is

$$\left[\bigtriangledown\mathcal{G}(\boldsymbol{\gamma})\right]_q=\frac{\partial\mathcal{G}(\boldsymbol{\gamma})}{\partial\gamma_q}-\frac{\partial\mathcal{G}(\boldsymbol{\gamma})}{\partial\gamma_u}\quad\forall\ q\neq u, \tag{7}$$

Table 1
Specification of used nine used datasets.

Dataset	Number of		
	Samples	Kernels	Clusters
Sonar	207	10	2
MSRA	210	6	7
Wdbc	569	10	2
Pima	768	8	2
Flower17	1360	7	17
Caltech101	1530	25	102
Mfeature	2000	3	10
Segment	2310	12	7
Cora	2708	2	7

and

$$\left[\bigtriangledown\mathcal{G}(\boldsymbol{\gamma})\right]_u=\sum_{q=1,q\neq u}^{m}\left(\frac{\partial\mathcal{G}(\boldsymbol{\gamma})}{\partial\gamma_u}-\frac{\partial\mathcal{G}(\boldsymbol{\gamma})}{\partial\gamma_q}\right). \tag{8}$$

We select u as the index of the maximum component of the vector $\boldsymbol{\gamma}$ because it improves numerical stability by following the advice in [27]. After that, we consider the descent direction which guarantees the positivity constraints on $\boldsymbol{\gamma}$. Because our goal is to minimize $\mathcal{G}(\boldsymbol{\gamma})$, we know that $-\bigtriangledown\mathcal{G}(\boldsymbol{\gamma})$ is a descent direction. However, we cannot directly adopt it as the descent direction, because the positivity constraints would be violated in the case that if there is a component q such that $\gamma_q=0$ and $\left[\bigtriangledown\mathcal{G}(\boldsymbol{\gamma})\right]_q>0$. In this condition, we should set the descent speed for the component as 0. Together considering all the aforementioned condition, we can update $\boldsymbol{\gamma}$ by using the following descent direction

$$d_q=\begin{cases}0 & \text{if } \gamma_q=0 \text{ and } \left[\bigtriangledown\mathcal{G}(\boldsymbol{\gamma})\right]_q>0\\ -\left[\bigtriangledown\mathcal{G}(\boldsymbol{\gamma})\right]_q & \text{if } \gamma_q>0 \text{ and } q\neq u\\ -\left[\bigtriangledown\mathcal{G}(\boldsymbol{\gamma})\right]_u & \text{if } q=u.\end{cases} \tag{9}$$

After a descent direction $\mathbf{d}=[d_1,\ldots,d_m]^{\top}$ is calculated by Eq. (9), we can compute $\boldsymbol{\gamma}$ by the updating scheme $\boldsymbol{\gamma}\leftarrow\boldsymbol{\gamma}+\alpha\mathbf{d}$, where α is termed the optimal step size. Usually, a line search approach, such as Armijo's rule, can be utilized to determine it. We outline the entire algorithm procedure which optimizes Eq. (4) in Algorithm 1.

Algorithm 1 SMKKM-KWR

Input: $\{\mathbf{K}_q\}_{q=1}^m$, $\boldsymbol{\gamma}_0$, k, $t=1$.
Initialize $\boldsymbol{\gamma}^{(1)}=\mathbf{1}/m$, flag $=True$.
while flag **do**
  compute $\mathbf{H}^{*}$ by solving KKM problem with $\mathbf{K}_{\boldsymbol{\gamma}^{(t)}}=\sum_{q=1}^{m}\left(\gamma_q^{(t)}\right)^2\mathbf{K}_q$.
  compute $\frac{\partial\mathcal{G}(\boldsymbol{\gamma})}{\partial\gamma_q}$ $(q=1,\cdots,m)$ and the descent direction $\mathbf{d}^{(t)}$ in Eq. (9).
  update $\boldsymbol{\gamma}^{(t+1)}\leftarrow\boldsymbol{\gamma}^{(t)}+\alpha\mathbf{d}^{(t)}$.
  if $\max|\boldsymbol{\gamma}^{(t+1)}-\boldsymbol{\gamma}^{(t)}|\leq e^{-4}$ **then**
    flag=Flase.
  end if
  $t\leftarrow t+1$.
end while

3.2. The global convergence

To analyze the convergence of the proposed algorithm, we need to prove the convexity of the objective in Eq. (5).

Theorem 2. *$\mathcal{G}(\boldsymbol{\gamma})$ in Eq. (5) is convex.*

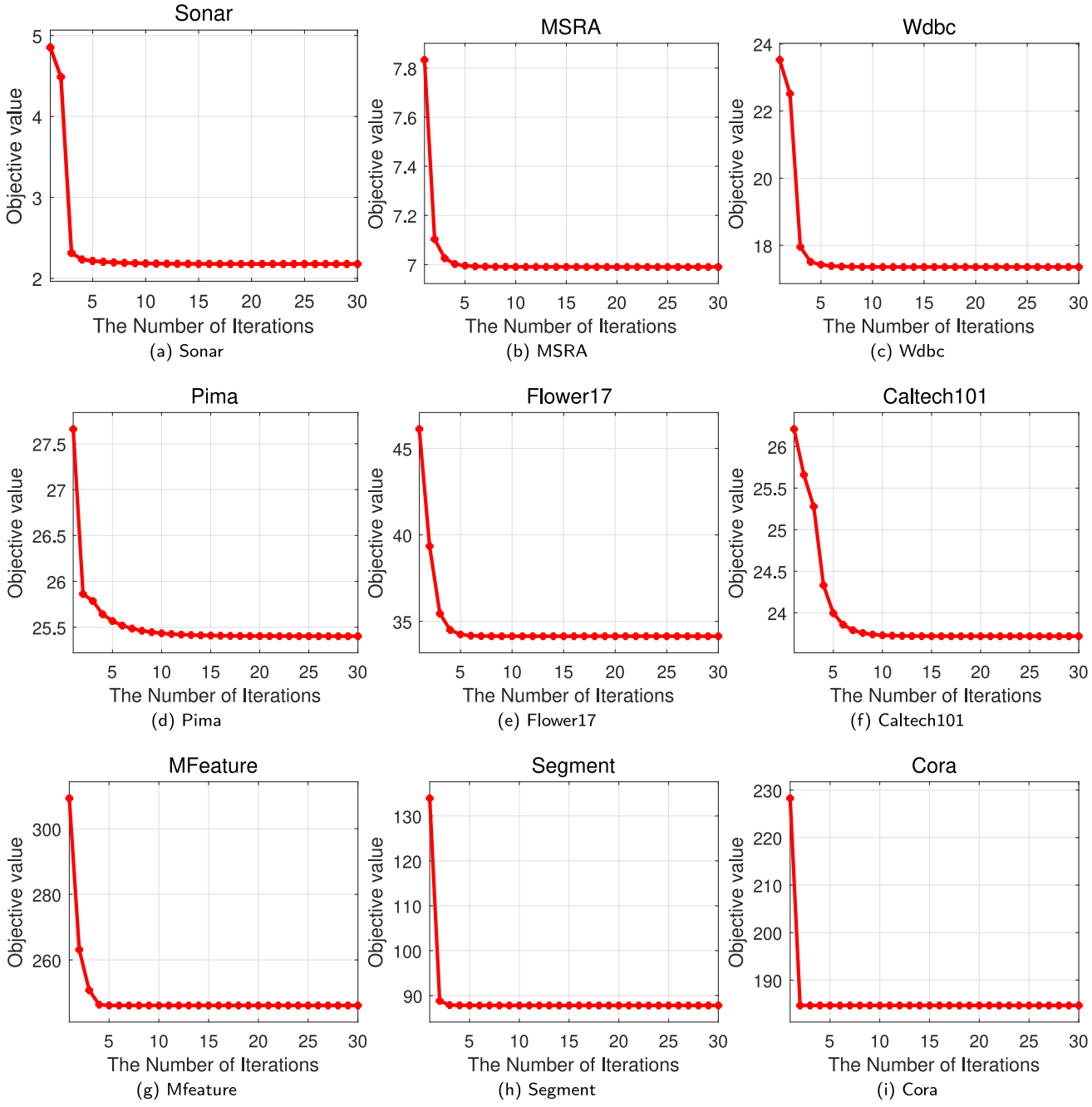

Fig. 1. The objective value learned by proposed algorithm across iterations on nine datasets.

Proof. Let $\mathcal{F}_1(\boldsymbol{\gamma}) = \left\{\max_{\mathbf{H}\in\Gamma} \operatorname{Tr}\left(\mathbf{K}_{\boldsymbol{\gamma}}\mathbf{H}\mathbf{H}^{\top}\right)\right\}$ and $\mathcal{F}_2(\boldsymbol{\gamma}) = \lambda\|\boldsymbol{\gamma} - \boldsymbol{\gamma}_0\|_2^2$. $\mathcal{F}_1(\boldsymbol{\gamma})$ has been proven to be convex. Furthermore, $\mathcal{F}_2(\boldsymbol{\gamma})$ is a quadratic term w.r.t. γ, obviously convex. Moreover, the feasible solution set of $\boldsymbol{\gamma}$ is a simplex which is a convex set. So that, $\mathcal{G}(\boldsymbol{\gamma}) = \mathcal{F}_1(\boldsymbol{\gamma}) + \mathcal{F}_2(\boldsymbol{\gamma})$ is convex. The proof is complete. □

The reduced gradient descent algorithm makes $\mathcal{G}(\boldsymbol{\gamma})$ decrease monotonically. Moreover, $\mathcal{G}(\boldsymbol{\gamma})$ in Eq. (5) is convex. Thus, the objective can be guaranteed to converge to a minimum. The convergence of our proposed algorithm is experimentally verified by the results in Fig. 1.

3.3. Discussion

We end up this section by discussing the computational complexity of the proposed SMKKM-KWR. From Algorithm 1, at each iteration, our proposed algorithm needs to perform a kernel k-means algorithm with complexity of $\mathcal{O}(n^2t)$, calculate the reduced gradient with complexity of $\mathcal{O}(mn^3)$, and search the optimal step size with complexity of $\mathcal{O}(mn_0)$, where n_0 is the maximal number of operations required to search the optimal step size. As seen, our proposed algorithm does not significantly increase the computational complexity of existing MKKM and SMKKM, which is also verified in Fig. 5.

Table 2
Clustering performance comparison on the nine benchmark datasets.

Dataset	Avg-KKM	MKKM	LMKKM	ONKC	MKKM-MiR	LKAM	LF-MVC	MKKM-MM	SMKKM	Proposed
ACC										
Sonar	57.1 ± 1.0	57.2 ± 0.7	57.2 ± 0.9	61.8 ± 0.0	57.0 ± 0.0	57.0 ± 0.0	56.0 ± 0.1	57.1 ± 1.0	63.8 ± 0.0	**64.7 ± 0.0**
MSRA	83.3 ± 0.8	81.3 ± 3.1	81.9 ± 0.7	85.4 ± 0.4	88.1 ± 0.1	89.1 ± 0.2	87.8 ± 0.4	83.3 ± 0.8	86.5 ± 0.2	**91.2 ± 0.4**
Wdbc	91.0 ± 0.0	91.0 ± 0.0	91.0 ± 0.0	81.4 ± 0.0	81.5 ± 0.0	79.4 ± 0.0	91.0 ± 0.0	91.0 ± 0.0	90.5 ± 0.0	**91.7 ± 0.0**
Pima	50.7 ± 0.0	51.3 ± 0.0	51.6 ± 0.7	65.8 ± 0.0	65.9 ± 0.0	63.8 ± 0.0	63.4 ± 0.0	50.7 ± 0.0	54.3 ± 6.3	**67.4 ± 0.7**
Flower17	50.8 ± 1.5	44.9 ± 2.4	37.5 ± 1.6	54.2 ± 2.2	58.5 ± 1.5	50.0 ± 0.8	**61.0 ± 0.7**	50.8 ± 1.5	59.5 ± 1.3	59.9 ± 1.6
Caltech101	34.2 ± 1.0	32.8 ± 0.9	27.9 ± 0.8	34.0 ± 0.9	34.8 ± 1.0	32.3 ± 1.0	34.4 ± 1.3	34.2 ± 1.0	35.8 ± 0.7	**36.6 ± 1.2**
MFeature	77.8 ± 1.5	63.2 ± 1.6	64.1 ± 1.5	80.4 ± 1.1	80.0 ± 1.4	90.7 ± 0.0	82.6 ± 0.0	77.8 ± 1.5	94.3 ± 2.4	**95.4 ± 0.0**
Segment	66.0 ± 0.0	65.9 ± 0.0	43.4 ± 4.3	60.9 ± 0.2	60.8 ± 0.1	54.0 ± 0.9	66.0 ± 0.0	66.0 ± 0.0	60.3 ± 0.8	**68.9 ± 0.1**
Cora	30.7 ± 0.8	25.3 ± 0.4	22.5 ± 0.2	40.8 ± 0.3	35.7 ± 0.1	34.5 ± 0.1	41.0 ± 1.1	30.7 ± 0.8	35.7 ± 0.1	**41.2 ± 0.0**
NMI										
Sonar	1.6 ± 0.3	1.6 ± 0.3	1.6 ± 0.3	3.8 ± 0.0	1.2 ± 0.0	1.2 ± 0.0	1.1 ± 0.1	1.6 ± 0.3	5.3 ± 0.0	**6.1 ± 0.0**
MSRA	74.0 ± 1.0	73.2 ± 1.7	75.0 ± 1.4	74.9 ± 0.7	77.6 ± 0.3	79.8 ± 0.2	79.4 ± 0.8	74.0 ± 1.0	75.2 ± 0.5	**82.7 ± 0.9**
Wdbc	55.2 ± 0.0	55.0 ± 0.0	55.0 ± 0.0	37.0 ± 0.0	36.3 ± 0.0	34.2 ± 0.0	55.3 ± 0.0	55.2 ± 0.0	54.3 ± 0.0	**59.5 ± 0.0**
Pima	0.0 ± 0.0	0.1 ± 0.0	0.2 ± 0.1	7.8 ± 0.0	7.8 ± 0.1	7.5 ± 0.0	6.4 ± 0.0	0.0 ± 0.0	1.5 ± 4.3	**9.3 ± 0.3**
Flower17	49.7 ± 1.0	44.9 ± 1.5	38.8 ± 1.1	52.6 ± 1.2	56.4 ± 0.9	49.8 ± 0.6	**58.9 ± 0.4**	49.7 ± 1.0	57.8 ± 0.9	58.1 ± 0.9
Caltech101	59.3 ± 0.6	58.6 ± 0.5	55.3 ± 0.5	59.3 ± 0.5	59.7 ± 0.5	58.5 ± 0.6	59.5 ± 0.6	59.3 ± 0.6	60.4 ± 0.5	**60.8 ± 0.7**
MFeature	73.4 ± 1.0	63.5 ± 1.1	65.1 ± 0.6	72.1 ± 0.9	71.9 ± 1.2	82.3 ± 0.1	78.2 ± 0.0	73.4 ± 1.0	89.1 ± 2.2	**90.5 ± 0.0**
Segment	57.1 ± 0.0	57.4 ± 0.1	43.7 ± 1.5	60.5 ± 0.8	54.7 ± 0.0	42.8 ± 0.7	56.9 ± 0.0	57.1 ± 0.0	60.4 ± 0.5	**61.5 ± 0.3**
Cora	15.7 ± 1.4	9.5 ± 0.2	6.7 ± 0.3	23.1 ± 0.3	18.9 ± 0.2	16.1 ± 0.1	22.4 ± 0.5	15.7 ± 1.4	18.8 ± 0.2	**23.6 ± 0.1**
Purity										
Sonar	57.1 ± 1.0	57.2 ± 0.7	57.2 ± 0.9	61.8 ± 0.0	57.0 ± 0.0	57.0 ± 0.0	56.0 ± 0.1	57.1 ± 1.0	63.8 ± 0.0	**64.7 ± 0.0**
MSRA	83.3 ± 0.8	81.5 ± 2.7	81.9 ± 0.7	85.4 ± 0.4	88.1 ± 0.1	89.1 ± 0.2	87.8 ± 0.4	83.3 ± 0.8	86.5 ± 0.2	**91.2 ± 0.4**
Wdbc	91.0 ± 0.0	91.0 ± 0.0	91.0 ± 0.0	81.4 ± 0.0	81.5 ± 0.0	79.4 ± 0.0	91.0 ± 0.0	91.0 ± 0.0	90.5 ± 0.0	**91.7 ± 0.0**
Pima	65.1 ± 0.0	65.1 ± 0.0	65.1 ± 0.0	65.8 ± 0.0	65.9 ± 0.0	65.1 ± 0.0	65.1 ± 0.0	65.1 ± 0.0	65.9 ± 2.3	**67.4 ± 0.7**
Flower17	51.9 ± 1.5	46.2 ± 2.0	39.2 ± 1.3	55.4 ± 2.2	59.7 ± 1.6	51.4 ± 0.7	**62.4 ± 0.7**	51.9 ± 1.5	60.9 ± 1.2	61.2 ± 1.5
Caltech101	36.2 ± 1.0	34.9 ± 0.9	29.6 ± 0.8	36.2 ± 0.9	36.8 ± 0.8	34.3 ± 0.9	36.7 ± 1.3	36.2 ± 1.0	38.0 ± 0.7	**38.6 ± 1.2**
MFeature	78.2 ± 0.7	63.9 ± 0.9	65.1 ± 1.0	80.4 ± 1.1	80.0 ± 1.4	90.7 ± 0.0	82.6 ± 0.0	78.2 ± 0.7	94.3 ± 2.4	**95.4 ± 0.0**
Segment	67.0 ± 0.0	66.9 ± 0.0	50.3 ± 2.1	63.5 ± 0.5	61.6 ± 0.1	56.7 ± 0.2	67.0 ± 0.0	67.0 ± 0.0	64.8 ± 0.8	**69.3 ± 0.1**
Cora	41.5 ± 1.3	36.1 ± 1.0	35.0 ± 0.2	48.6 ± 0.3	47.0 ± 0.1	43.3 ± 0.1	47.2 ± 0.5	41.5 ± 1.3	47.0 ± 0.1	**49.3 ± 0.1**
Rand index										
Sonar	1.6 ± 0.5	1.6 ± 0.4	1.6 ± 0.4	5.1 ± 0.0	1.5 ± 0.0	1.5 ± 0.0	1.0 ± 0.1	1.6 ± 0.5	7.1 ± 0.0	**8.2 ± 0.0**
MSRA	68.1 ± 1.0	66.2 ± 3.1	68.0 ± 1.1	69.8 ± 0.7	74.5 ± 0.1	76.7 ± 0.4	74.5 ± 0.8	68.1 ± 1.0	71.2 ± 0.5	**80.3 ± 0.8**
Wdbc	67.2 ± 0.0	67.2 ± 0.0	67.2 ± 0.0	39.3 ± 0.0	39.7 ± 0.0	34.5 ± 0.0	67.2 ± 0.0	67.2 ± 0.0	65.5 ± 0.0	**69.6 ± 0.0**
Pima	−0.1 ± 0.0	−0.1 ± 0.0	−0.1 ± 0.2	9.8 ± 0.1	10.0 ± 0.1	7.5 ± 0.0	7.1 ± 0.0	−0.1 ± 0.0	2.1 ± 6.2	**12.1 ± 1.0**
Flower17	32.2 ± 1.3	27.2 ± 1.8	20.6 ± 1.1	35.2 ± 1.5	39.9 ± 1.3	31.6 ± 0.8	**44.1 ± 0.4**	32.2 ± 1.3	41.5 ± 1.5	41.9 ± 1.5
Caltech101	18.4 ± 0.9	17.3 ± 0.7	13.4 ± 0.8	18.3 ± 0.8	18.8 ± 0.8	16.8 ± 0.9	18.8 ± 1.0	18.4 ± 0.9	19.8 ± 0.7	**20.7 ± 1.1**
MFeature	64.0 ± 1.8	49.6 ± 1.3	50.6 ± 0.9	64.7 ± 1.5	64.6 ± 2.0	80.5 ± 0.0	72.1 ± 0.0	64.0 ± 1.8	88.3 ± 3.6	**90.1 ± 0.0**
Segment	48.1 ± 0.0	48.4 ± 0.0	26.4 ± 3.1	48.4 ± 1.2	46.1 ± 0.0	35.0 ± 0.5	47.9 ± 0.0	48.1 ± 0.0	52.7 ± 0.5	**52.8 ± 0.2**
Cora	6.5 ± 0.6	3.6 ± 0.3	1.7 ± 0.1	15.6 ± 0.4	11.4 ± 0.1	11.1 ± 0.1	14.5 ± 0.4	6.5 ± 0.6	11.4 ± 0.1	**16.7 ± 0.1**

4. Experiment

4.1. Experimental setting

In our experiments, we utilized nine benchmark datasets, including *Sonar*,[2] *MSRA*[1], *Wdbc*[1], *Pima*[1], *Flower17*,[3] *Caltech101*,[4] *Mfeature*[1], *Segment*,[5] *Cora*,[6] to evaluate the clustering performance of our proposed SMKKM-KWR. The detailed information regarding these benchmarks is summarized in Table 1.

It is assumed that the number of clusters k is given for all datasets. To evaluate the algorithm clustering performance, we utilized four publicly-used clustering metrics, namely accuracy (ACC), normalized mutual information (NMI), purity (Pur) and rand index (RI).

We compared our proposed SMKKM-KWR with ten baseline and state-of-the-art methods, including:

- **Average kernel k-means (Avg-KKM)**. The kernel k-means algorithm is performed to obtain the clustering results after generating the average kernel from all views.
- **Multiple kernel k-means (MKKM)** [28]. The consensus kernel is obtained by linearly combining the base kernels and jointly optimizing the weights and clustering assignment partition.
- **Localized multiple kernel k-means (LMKKM)** [13]. It utilized the adaptive weights and local information on the sample to produce the weighted kernel.
- **Optimal neighborhood kernel clustering (ONKC)** [22]. It learns an optimal kernel among the neighbors of the linear combination of the basic kernels.
- **Multiple kernel k-means with matrix-induced regularization (MKKM-MiR)** [20]. Through the guidance of matrix-induced regularization, the learned combination weights are more excellent, and redundancy is reduced.
- **Multiple kernel clustering with local alignment maximization (LKAM)** [21]. The alignment of the similarity of a sample's k-nearest neighbors with the ideal similarity matrix is what the algorithm tried to learn.
- **Multi-view clustering via late fusion alignment maximization (LFMVC)** [29]. The algorithm initially computes the base kernels of all the views, which are subsequently merged into a single consensus division.
- **Robust multiple kernel k-means with min–max optimization (MKKM-MM)** [30]. To enhance the robustness, it proposes to reduce perturbation with min–max framework.

[2] http://archive.ics.uci.edu/ml/datasets/.
[3] www.robots.ox.ac.uk/~vgg/data/flowers/17/.
[4] http://www.vision.caltech.edu/ImageDatasets/Caltech101/.
[5] https://bmi.inf.ethz.ch/supplements/protsubloc/.
[6] http://mlg.ucd.ie/aggregation/.

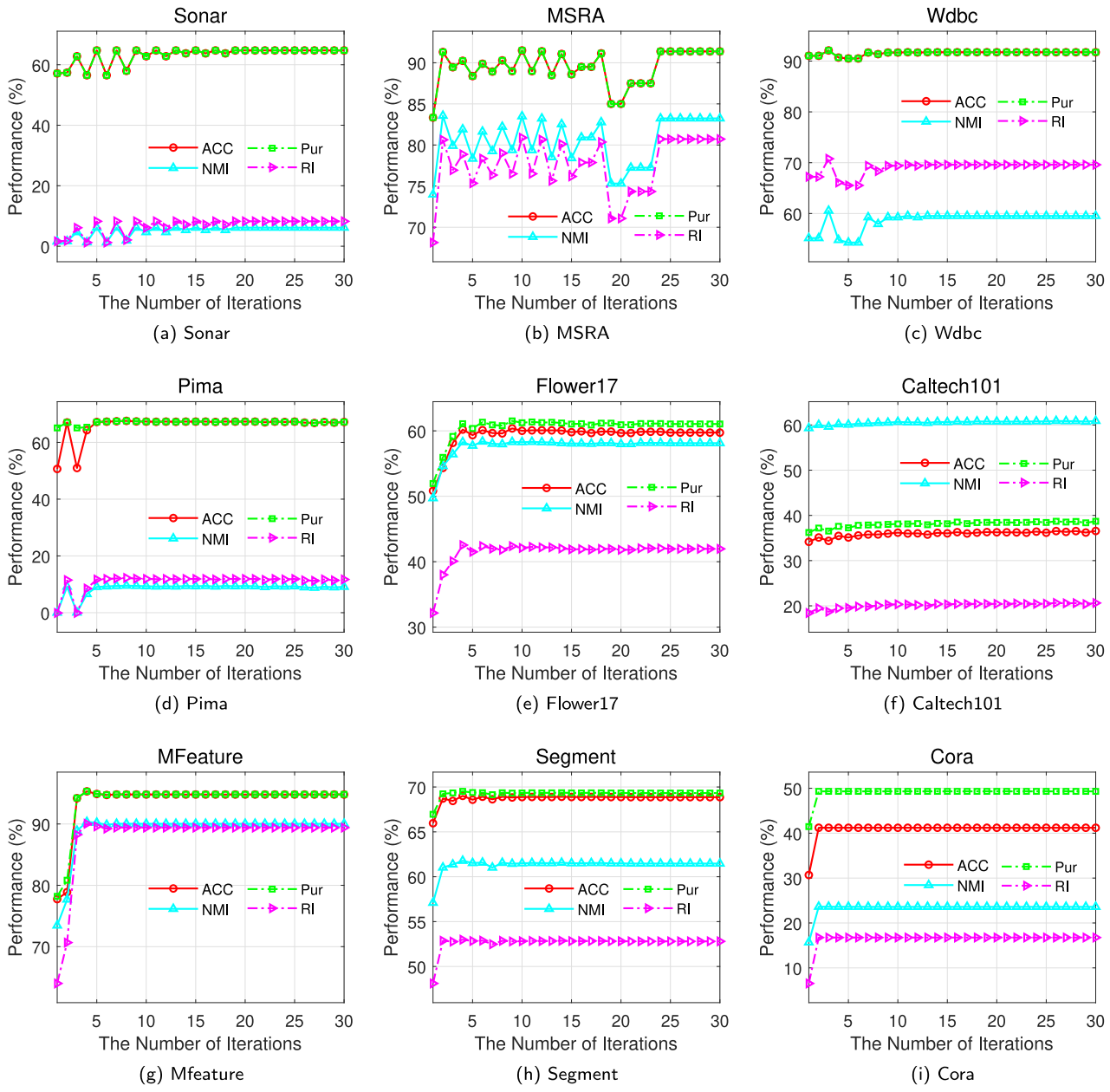

Fig. 2. The clustering performance evolution learned by proposed algorithm across iterations on nine datasets.

- **Simple multiple kernel k-means (SMKKM)** [25]. This work introduces the min–max optimization framework, and adopts the reduced gradient descent algorithm to address the MKKM problem with new formulation.

To ensure consistency, we downloaded publicly available code from the source website and set parameters based on the corresponding literature recommendations for each of the listed algorithms. In our proposed method, we tuned the hyper-parameter λ within the range of $\left[2^{-10}, 2^{-9}, \ldots, 2^{10}\right]$ to optimize the results. To mitigate the potential impact of random initialization of k-means, we conducted each experiment 50 times and reported the average result. Our experiments are all conducted the environment of MATLAB R2020b on a PC with i9-10900X CPU and 64 GB RAM.

4.2. Experimental result

4.2.1. Overall performance

The results of the compared methods on the 9 datasets are presented in Table 2, including ACC, NMI, Pur, and RI values. The best results are highlighted in bold, while the second-best results are underlined. According to these results, we can have the following observations.

- The recent introduction of SMKKM [25] has demonstrated substantial improvements in performance metrics such as ACC, NMI, Pur, and RI compared to many previous methods. However, the results of SMKKM were unsatisfactory in the Pima and Segment datasets due to its inability to integrate prior knowledge. In certain extreme scenarios, the learning outcome may deviate

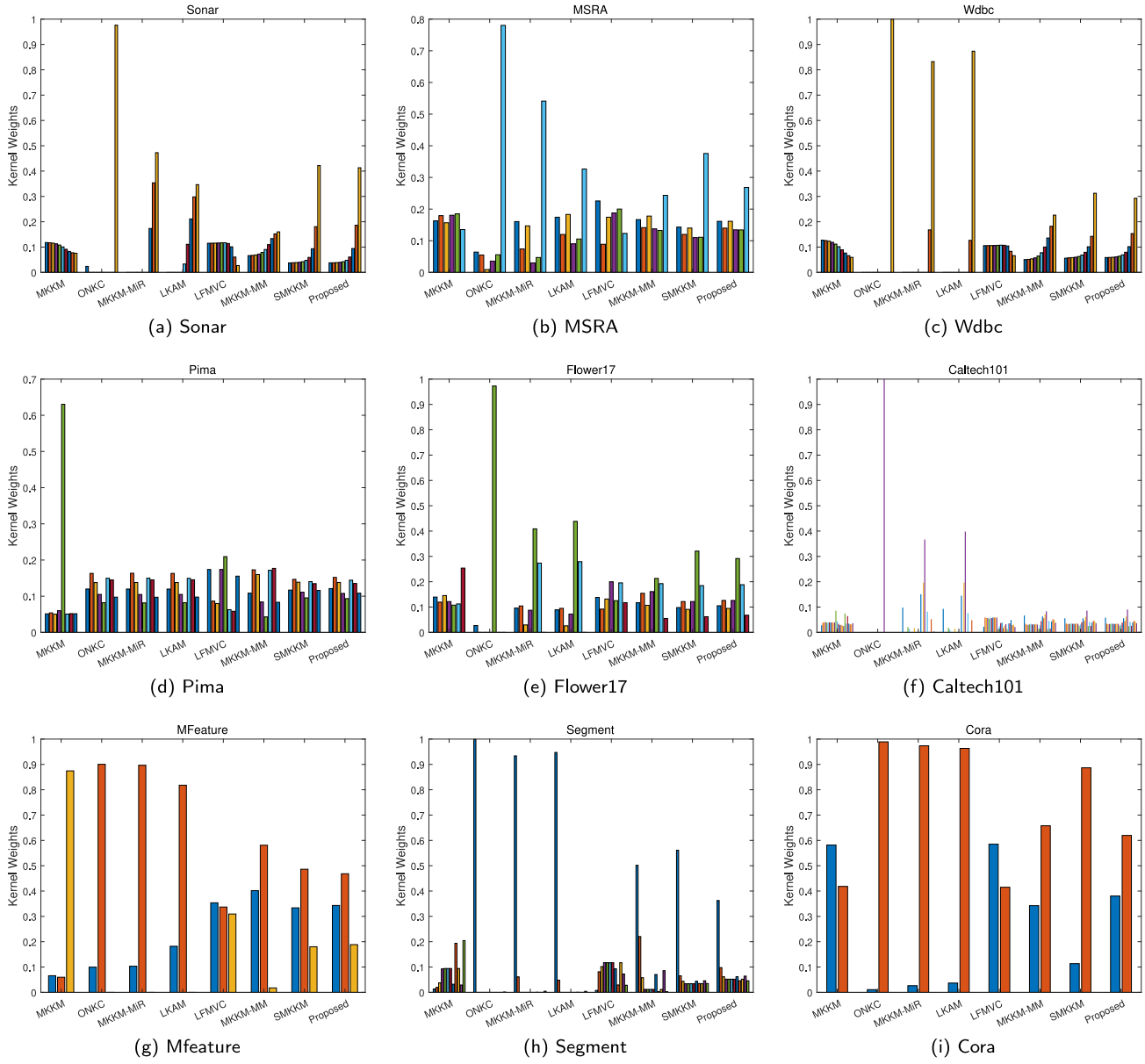

Fig. 3. Comparison of the weight coefficients obtained by the algorithm on nine datasets.

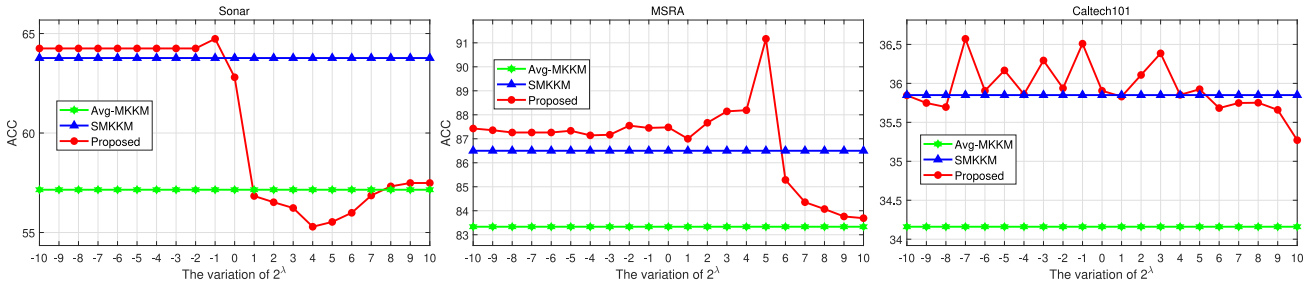

Fig. 4. The sensitivity analysis of λ on Sonar, MSRA and Caltech101 datasets.

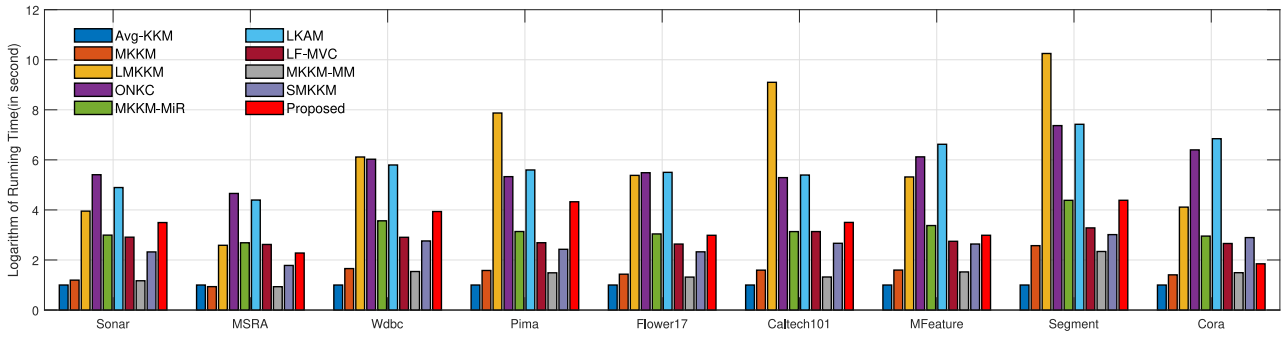

Fig. 5. Time cost of the tested algorithms on nine datasets.

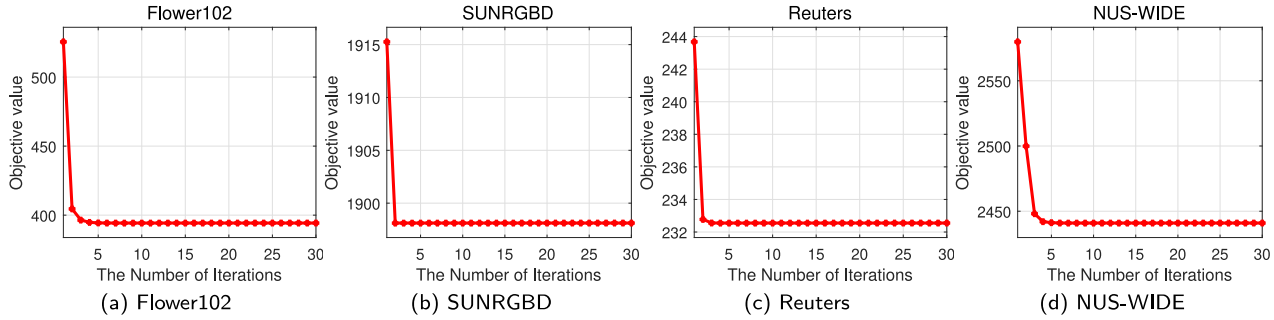

Fig. 6. The objective value with iterations learned by proposed algorithm on four large-scale datasets.

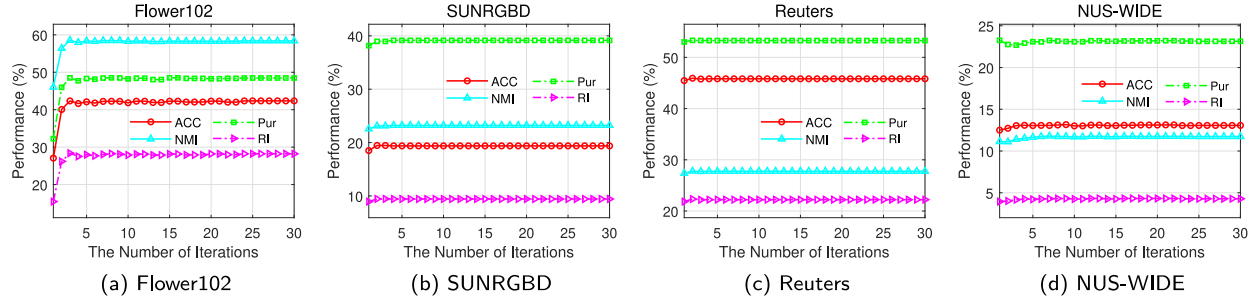

Fig. 7. The clustering performance with iterations learned by proposed algorithm on four large-scale datasets.

significantly from the average kernel weight, thereby negatively affecting the clustering task. Notably, our proposed SMKKM-KWR algorithm outperforms it by significant margins, namely 0.9%, 4.7%, 1.2%, 13.1%, 0.4%, 0.6%, 1.1%, 8.6% and 5.5% in terms of ACC. The improvements of the other performance criteria are similar, which demonstrates the importance of using prior information while SMKKM has already got the good performance.

- Our proposed SMKKM-KWR algorithm exceeds almost all the compared methods across various performance metrics. Specifically, in the MSRA, Pima, and Segment datasets, its ACC exceeds that of suboptimal algorithms by 2.1%, 1.5%, and 2.9%, respectively. This improvement can be attributed to the framework of min–max optimization and the effective integration of prior information concerning the average kernel coefficients.
- Except for SMKKM, LFMVC achieved good clustering performance in most cases, thanks to its idea of post-merging the partition matrix. However, our proposed SMKKM-KWR algorithm outperformed it in the ACC metric on eight datasets by 8.7%, 3.4%, 0.7%, 4.0%, 2.2%, 12.8%, 2.9% and 0.2%, respectively.

In summary, our proposed algorithm exhibits superior performance when compared to the contrast algorithm. This is attributed to the validity of the min–max alignment criterion and the effective guidance provided by the average kernel coefficients for clustering. Furthermore, the exceptional experimental results provide compelling evidence supporting the use of the average partition as prior information.

4.2.2. Learning procedure

The learning progress of our proposed SMKKM-KWR is illustrated in Figs. 1 and 2, which display the loss and clustering performance, respectively, across iterations. It is evident that the optimization formula loss monotonically decreases, while the clustering performance exhibits a consistent upward trend with the increase in the number of iterations.

4.2.3. Kernel weights

We investigate the weight coefficients of the kernel learned by various algorithms on benchmark datasets, and plot the distribution in Fig. 3. It is observed that some datasets, such as Mfeature and Pima, exhibit extremely sparse kernel weights learned by MKKM due to alternating optimization, which prevents the full utilization of multiple

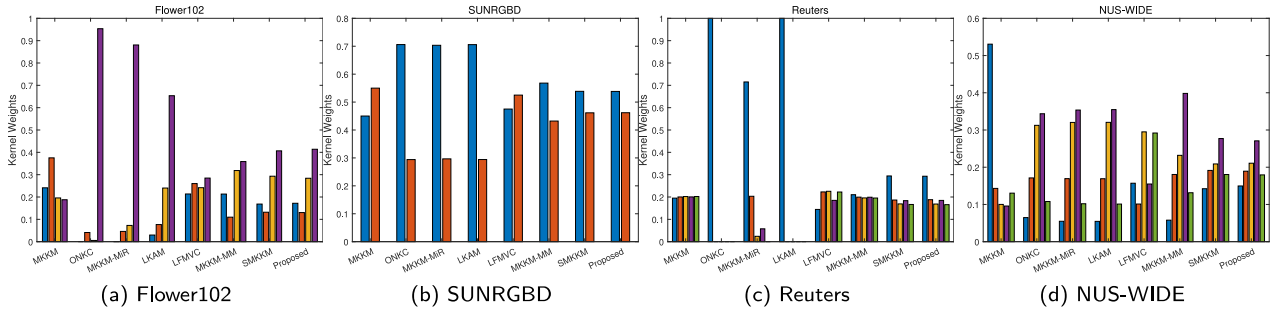

Fig. 8. Comparison of the weight coefficients obtained by the algorithm on four large-scale datasets.

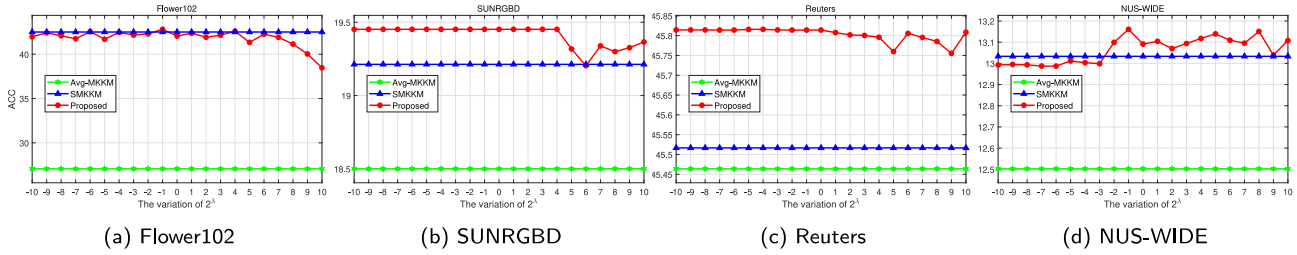

Fig. 9. The sensitivity analysis of λ on four large-scale datasets.

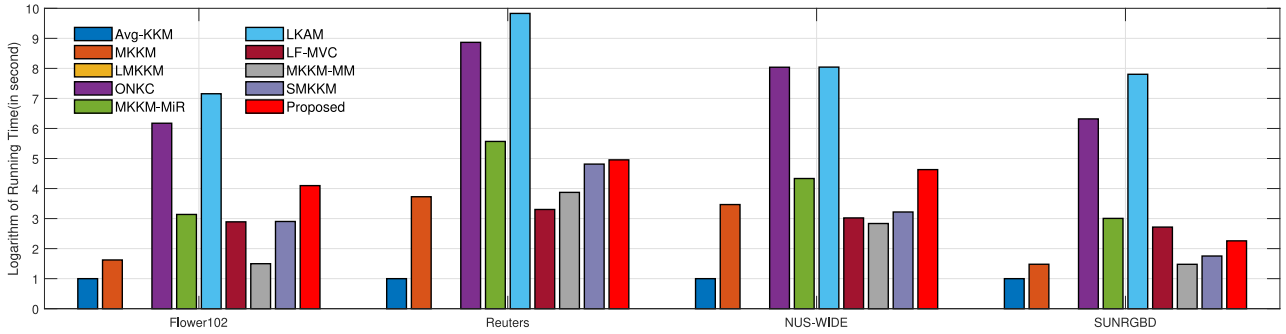

Fig. 10. Time cost of tested algorithms on four large-scale datasets.

Table 3
Specification of used four large-scale datasets.

Dataset	Number of		
	Samples	Kernels	Clusters
Flower102	8189	4	102
SUNRGBD	10 335	2	45
Reuters	18 758	5	6
NUS-WIDE	23 593	5	31

kernel matrices and leads to poor performance. For instance, MKKM achieves ACC of only 50.7% and 77.8% on Pima and Mfeature, respectively. In contrast, the weight coefficients of the kernel obtained by our proposed SMKKM-KWR algorithm are all non-sparse, which greatly benefits the clustering performance. For example, our algorithm achieves clustering accuracies of 67.4% and 95.4% on Mfeature and Pima, respectively. This improvement can be attributed to the l_2 norm constraint on $\boldsymbol{\gamma}$ and the use of a reduced gradient descent algorithm to ensure reasonable weight coefficients.

4.2.4. Hyper-parameter sensitivity

We carry out the experiments to examine the sensitivity of the proposed algorithm to hyper-parameter settings by comparing various values of λ. The results, as illustrated in Fig. 4, demonstrate that the algorithm remains stable for values of λ smaller than 2^0 and 2^6 on Sonar and MSRA datasets, respectively. This preliminary investigation suggests that the proposed algorithm is relatively insensitive to changes in hyper-parameter values.

4.2.5. Running time

To provide a more comprehensive illustration of the computational efficiency of our proposed SMKKM-KWR, we have presented the execution time of all tested algorithms on each dataset in Fig. 5. The results indicate that our algorithm maintains the average computational efficiency without significantly increasing the time complexity while simultaneously enhancing the clustering performance.

4.3. Experiment on large-scale datasets

Furthermore, we conduct the experiment on four extra large-scale datasets, including *Flower102*,[7] *SUNRGBD*,[8] *Reuters*,[9] *NUS-WIDE* [31]. The detailed information regarding these benchmarks is summarized in Table 3.

[7] www.robots.ox.ac.uk/~vgg/data/flowers/102/.
[8] http://rgbd.cs.princeton.edu/.
[9] https://kdd.ics.uci.edu/databases/reuters21578/.

Table 4
Clustering performance comparison on the four large-scale datasets.

Dataset	Avg-KKM	MKKM	LMKKM	ONKC	MKKM-MiR	LKAM	LF-MVC	MKKM-MM	SMKKM	Proposed
ACC										
Flower102	27.1 ± 0.8	22.4 ± 0.5	OOM	39.5 ± 0.7	40.2 ± 0.9	41.4 ± 0.8	38.4 ± 1.2	27.1 ± 0.8	42.5 ± 0.8	42.8 ± 0.8
SUNRGBD	18.5 ± 0.5	17.2 ± 0.6	OOM	19.8 ± 0.5	19.5 ± 0.5	19.6 ± 0.5	18.6 ± 0.6	18.5 ± 0.5	19.2 ± 0.5	19.5 ± 0.6
Reuters	45.5 ± 1.5	45.4 ± 1.5	OOM	41.8 ± 1.2	46.2 ± 1.4	45.5 ± 0.0	45.7 ± 1.6	45.5 ± 1.5	45.5 ± 0.7	45.8 ± 1.0
NUS-WIDE	12.5 ± 0.4	12.7 ± 0.2	OOM	13.1 ± 0.3	12.9 ± 0.2	13.7 ± 0.2	13.2 ± 0.4	12.5 ± 0.4	13.0 ± 0.3	13.2 ± 0.2
NMI										
Flower102	46.0 ± 0.5	42.7 ± 0.2	OOM	56.1 ± 0.4	56.7 ± 0.5	56.9 ± 0.3	54.9 ± 0.4	46.0 ± 0.5	58.6 ± 0.5	58.7 ± 0.4
SUNRGBD	22.6 ± 0.3	21.2 ± 0.4	OOM	23.6 ± 0.2	23.5 ± 0.3	23.9 ± 0.3	22.6 ± 0.4	22.6 ± 0.3	23.1 ± 0.4	23.1 ± 0.3
Reuters	27.4 ± 0.4	27.3 ± 0.4	OOM	22.3 ± 0.4	25.3 ± 0.7	29.9 ± 0.0	27.4 ± 0.4	27.4 ± 0.4	27.7 ± 0.2	27.8 ± 0.2
NUS-WIDE	11.1 ± 0.1	11.3 ± 0.2	OOM	11.2 ± 0.2	11.0 ± 0.2	13.4 ± 0.2	11.3 ± 0.2	11.1 ± 0.1	11.4 ± 0.2	11.7 ± 0.2
Purity										
Flower102	32.3 ± 0.6	27.8 ± 0.4	OOM	45.6 ± 0.7	46.3 ± 0.8	48.0 ± 0.6	44.6 ± 0.8	32.3 ± 0.6	48.6 ± 0.7	48.8 ± 0.8
SUNRGBD	38.2 ± 0.7	36.2 ± 0.7	OOM	39.6 ± 0.6	39.4 ± 0.6	39.6 ± 0.4	38.1 ± 0.6	38.2 ± 0.7	39.0 ± 0.6	39.0 ± 0.6
Reuters	53.0 ± 0.4	52.9 ± 0.5	OOM	52.6 ± 0.3	52.2 ± 0.6	55.4 ± 0.0	53.2 ± 0.4	53.0 ± 0.4	53.3 ± 0.0	53.3 ± 0.0
NUS-WIDE	23.3 ± 0.3	24.2 ± 0.4	OOM	22.6 ± 0.4	22.2 ± 0.4	25.0 ± 0.4	23.5 ± 0.4	23.3 ± 0.3	22.9 ± 0.3	23.1 ± 0.4
Rand index										
Flower102	15.5 ± 0.5	12.1 ± 0.4	OOM	24.9 ± 0.5	25.5 ± 0.6	27.2 ± 0.6	25.5 ± 1.0	15.5 ± 0.5	28.5 ± 0.8	28.4 ± 0.6
SUNRGBD	8.9 ± 0.3	8.1 ± 0.3	OOM	9.7 ± 0.2	9.6 ± 0.3	9.9 ± 0.3	9.0 ± 0.2	8.9 ± 0.3	9.4 ± 0.3	9.4 ± 0.2
Reuters	21.8 ± 1.4	21.8 ± 1.4	OOM	20.3 ± 0.3	23.1 ± 0.6	24.1 ± 0.0	22.1 ± 1.6	21.8 ± 1.4	22.1 ± 0.8	22.2 ± 0.8
NUS-WIDE	3.9 ± 0.2	4.0 ± 0.1	OOM	4.3 ± 0.2	4.2 ± 0.1	5.3 ± 0.2	4.5 ± 0.2	3.9 ± 0.2	4.3 ± 0.2	4.3 ± 0.1

The aggregative clustering performance are shown in Table 4, and visible results are plotted in Figs. 6, 7, 8, 9, 10, respectively. Note that "OOM" denotes "OUT OF MEMORY".

From the experimental results, we can observe that the clustering performance of our proposed algorithm is still stable, always achieving encouraging improvement. Furthermore, the running time consumed by our proposed algorithm is acceptable.

5. Conclusion

In this paper, we develop a new multiple kernel K-means algorithm termed as SMKKM-KWR. To prevent the learned unified partition being far from the average partition,we add the average alignment as a regularization term on the basis of SMKKM. Moreover, we propose an efficient optimization algorithm to solve the new resultant problem. Comprehensive experimental results on nine datasets show the effectiveness of our proposed SMKKM-KWR algorithm.

CRediT authorship contribution statement

Miaomiao Li: Conceptualization, Writing – original draft. **Yi Zhang:** Experiment, Methodology, Manuscript proof-reading. **Suyuan Liu:** Methodology, Manuscript proof-reading. **Zhe Liu:** Methodology, Writing–review. **Xinzhong Zhu:** Conceptualization, Writing–review.

Declaration of competing interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

Data availability

Data will be made available on request.

References

[1] M. Filippone, F. Camastra, F. Masulli, S. Rovetta, A survey of kernel and spectral methods for clustering, Pattern Recognit. 41 (1) (2008) 176–190.

[2] B. Zhao, J.T. Kwok, C. Zhang, Multiple kernel clustering, in: SDM, 2009, pp. 638–649.

[3] X.P. Blanco Valencia, M. Becerra, A. Castro Ospina, M. Ortega Adarme, D. Viveros Melo, D.H. Peluffo-Ordóñez, et al., Kernel-based framework for spectral dimensionality reduction and clustering formulation: A theoretical study, 2017.

[4] D. Marin, M. Tang, I.B. Ayed, Y. Boykov, Kernel clustering: Density biases and solutions, IEEE Trans. Pattern Anal. Mach. Intell. 41 (1) (2017) 136–147.

[5] L. Houthuys, R. Langone, J.A. Suykens, Multi-view kernel spectral clustering, Inf. Fusion 44 (2018) 46–56.

[6] M. Gönen, E. Alpaydın, Multiple kernel learning algorithms, J. Mach. Learn. Res. 12 (2011) 2211–2268.

[7] A. Kumar, P. Rai, H. Daumé, Co-regularized multi-view spectral clustering, in: NIPS, 2011, pp. 1413–1421.

[8] A. Kumar, H. Daumé, A Co-training approach for multi-view spectral clustering, in: ICML, 2011, pp. 393–400.

[9] R. Chitta, R. Jin, A.K. Jain, Efficient kernel clustering using random fourier features, in: 2012 IEEE 12th International Conference on Data Mining, IEEE, 2012, pp. 161–170.

[10] B. Yan, P. Sarkar, On robustness of kernel clustering, Adv. Neural Inf. Process. Syst. 29 (2016).

[11] W. Yan, J. Xu, J. Liu, G. Yue, C. Tang, Bipartite graph-based discriminative feature learning for multi-view clustering, in: Proceedings of the 30th ACM International Conference on Multimedia, 2022, pp. 3403–3411.

[12] S. Yu, L.-C. Tranchevent, X. Liu, W. Glänzel, J.A.K. Suykens, B.D. Moor, Y. Moreau, Optimized data fusion for kernel k-means clustering, IEEE TPAMI 34 (5) (2012) 1031–1039.

[13] M. Gönen, A.A. Margolin, 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, Montreal, Quebec, Canada, 2014, pp. 1305–1313, December 8-13 2014.

[14] X. Peng, Z. Huang, J. Lv, H. Zhu, J.T. Zhou, COMIC: multi-view clustering without parameter selection, in: Proceedings of the 36th International Conference on Machine Learning, ICML, 2019, pp. 5092–5101.

[15] S. Zhou, E. Zhu, X. Liu, T. Zheng, Q. Liu, J. Xia, J. Yin, Subspace segmentation-based robust multiple kernel clustering, Inf. Fusion 53 (2020) 145–154.

[16] Y. Zhang, W. Liang, X. Liu, S. Dai, S. Wang, L. Xu, E. Zhu, Sample weighted multiple kernel K-means via min-max optimization, in: Proceedings of the 30th ACM International Conference on Multimedia, 2022, pp. 1679–1687.

[17] R. Wang, J. Lu, Y. Lu, F. Nie, X. Li, Discrete and parameter-free multiple kernel k-means, IEEE Trans. Image Process. 31 (2022) 2796–2808.

[18] C. Tang, Z. Li, W. Yan, G. Yue, W. Zhang, Efficient multiple kernel clustering via spectral perturbation, in: Proceedings of the 30th ACM International Conference on Multimedia, 2022, pp. 1603–1611.

[19] L. Du, P. Zhou, L. Shi, H. Wang, M. Fan, W. Wang, Y.-D. Shen, Robust multiple kernel k-means using l21-norm, in: Twenty-Fourth International Joint Conference on Artificial Intelligence, 2015.

[20] X. Liu, Y. Dou, J. Yin, L. Wang, E. Zhu, Multiple kernel k-means clustering with matrix-induced regularization, in: Proceedings of the Thirtieth AAAI Conference on Artificial Intelligence, Phoenix, Arizona, USA, 2016, pp. 1888–1894, February 12-17, 2016.

[21] M. Li, X. Liu, L. Wang, Y. Dou, J. Yin, E. Zhu, Multiple kernel clustering with local kernel alignment maximization, in: Proceedings of the Twenty-Fifth International Joint Conference on Artificial Intelligence, IJCAI 2016, New York, NY, USA, 2016, pp. 1704–1710, 9-15 July 2016.

[22] X. Liu, S. Zhou, Y. Wang, M. Li, Y. Dou, E. Zhu, J. Yin, Optimal neighborhood kernel clustering with multiple kernels, in: Proceedings of the Thirty-First AAAI Conference on Artificial Intelligence, San Francisco, California, USA, 2017, pp. 2266–2272, February 4-9, 2017.

[23] Y. Zhang, X. Liu, J. Liu, S. Dai, C. Zhang, K. Xu, E. Zhu, Fusion multiple kernel k-means, in: Proceedings of the AAAI Conference on Artificial Intelligence, 2022, pp. 9109–9117.
[24] R. Yin, Y. Liu, W. Wang, D. Meng, Scalable kernel *k*-means with randomized sketching: From theory to algorithm, IEEE Trans. Knowl. Data Eng. (2022).
[25] X. Liu, E. Zhu, J. Liu, Simplemkkm: Simple multiple kernel k-means, 2020, arXiv preprint arXiv:2005.04975.
[26] J.F. Bonnans, A. Shapiro, Optimization problems with perturbations: A guided tour, SIAM Rev. 40 (2) (1998) 228–264.
[27] A. Rakotomamonjy, F.R. Bach, S. Canu, Y. Grandvalet, Simplemkl, JMLR 9 (2008) 2491–2521.
[28] H. Huang, Y. Chuang, C. Chen, Multiple kernel fuzzy clustering, IEEE Trans. Fuzzy Syst. 20 (1) (2012) 120–134.
[29] S. Wang, X. Liu, E. Zhu, C. Tang, J. Liu, J. Hu, J. Xia, J. Yin, Multi-view clustering via late fusion alignment maximization, in: Proceedings of the Twenty-Eighth International Joint Conference on Artificial Intelligence, IJCAI 2019, Macao, China, 2019, pp. 3778–3784, August 10-16, 2019.
[30] S. Bang, Y. Yu, W. Wu, Robust multiple kernel k-means clustering using min-max optimization, 2018, arXiv:1803.02458.
[31] T.-S. Chua, J. Tang, R. Hong, H. Li, Z. Luo, Y.-T. Zheng, NUS-WIDE: A real-world web image database from national university of Singapore, in: Proc. of ACM Conf. on Image and Video Retrieval (CIVR'09), Santorini, Greece, 2009, July 8-10.

Miaomiao Li received her Ph.D. degree at National University of Defense Technology, China. She is now Lecture of Changsha University, Changsha, China. Her current research interests include kernel learning and multi-view clustering. Miaomiao Li has published several peer-reviewed papers such as IEEE T-PAMI, IEEE T-NNLS, AAAI, IJCAI, Neurocomputing, etc. She serves on the Technical Program Committees of IJCAI 2017–2020.

Yi Zhang is pursuing his Ph.D. degree in National University of Defense Technology (NUDT) China. His current research interests include kernel learning and unsupervised multi-view learning. He has published several peer-reviewed papers in journals and conferences such as NeurIPS, ICML, ICCV, ACM MM, AAAI, IEEE T-IP, IEEE T-NNLS, TOMM, etc.

Suyuan Liu is pursuing his Ph.D. degree in National University of Defense Technology (NUDT) China His current research interests include multi-view learning and scalable clustering.

Zhe Liu received the B.S. and M.S. degrees from Shandong University, in 2008 and 2011, respectively, and the Ph.D. degree from the Laboratory of Algorithmics, Cryptology and Security (LACS), University of Luxembourg, Luxembourg, in 2015. He is a professor with the College of Computer Science and Technology, Nanjing University of Aeronautics and Astronautics (NUAA), China, and he is the founder of Trust Intelligent Computing research group in Zhejiang Lab. Before joining NUAA, he was a researcher with the SnT, University of Luxembourg, Luxembourg. His Ph.D. thesis has received the prestigious FNR Awards 2016 — Outstanding Ph.D. Thesis Award for his contributions in cryptographic engineering on IoT devices. His research interests include computer arithmetic and information security. He has co-authored more than 70 research peer-reviewed journal and conference papers.

Xinzhong Zhu is a professor at College of Mathematics and Computer Science, Zhejiang Normal University, and also the president of Research Institute of Ningbo Cixing Co. Ltd, PR, China. He received his Ph.D. degree at XIDIAN University, China. His research interests include machine learning, computer vision, manufacturing informatization, robotics and system integration, and intelligent manufacturing. He is a member of the ACM and certified as CCF Senior Member. Dr. Zhu has published 30+ peer-reviewed papers, including those in highly regarded journals and conferences such as IEEE T-PAMI, IEEE T-MM, IEEE T-KDE, AAAI, IJCAI, etc.