Centerless Clustering: An Efficient Variant of K-means Based on K-NN Graph

Shenfei Pei, Huimin Chen, Feiping Nie, Rong Wang, and Xuelong Li, *Fellow, IEEE*

Abstract—Although lots of clustering models have been proposed recently, k-means and the family of spectral clustering methods are both still drawing a lot of attention due to their simplicity and efficacy. We first reviewed the unified framework of k-means and graph cut models, and then proposed a clustering method called k-sums where a k-nearest neighbor (k-NN) graph is adopted. The main idea of k-sums is to minimize directly the sum of the distances between points in the same cluster. To deal with the situation where the graph is unavailable, we proposed k-sums-x that takes features as input. The computational and memory overhead of k-sums are both $O(nk)$, indicating that it can scale linearly w.r.t. the number of objects to group. Moreover, the costs of computational and memory are Irrelevant to the product of the number of points and clusters. The computational and memory complexity of k-sums-x are both linear w.r.t. the number of points. To validate the advantage of k-sums and k-sums-x on facial datasets, extensive experiments have been conducted on 10 synthetic datasets and 17 benchmark datasets. While having a low time complexity, the performance of k-sums is comparable with several state-of-the-art clustering methods.

Index Terms—Efficient clustering, fast, k-nearest graph, spectral clustering, k-means.

1 Introduction

With the popularity of the Internet, large-scale datasets can be found anywhere in our lives, such as shopping records generated by e-commerce sites and photos (videos) uploaded by users on social media platforms. Take Facebook as an example, there are over 2.5 billion monthly active users in the United States and Canada, about 1 million links are shared, and 3 million messages are sent, every 20 minutes, 100 million hours of video are watched on Facebook every day, as of April 2020, reported by statista[1]. Although these large-scale datasets are relatively easy to obtain, further analysis is essential to get useful information.

As one of the fundamental unsupervised technologies in the fields of machine learning, data mining, pattern recognition, and others, clustering is drawing more and more attention over the last few years. Up to now, a series of models have been presented for cluster analysis and put into practice successfully in various fields, such as face recognition [1], [2], document clustering [3], [4], social networks [5], [6], and image segmentation [7], [8].

Among various clustering algorithms, the family of spectral clustering (SC) and k-means are still drawing lots of attention, due to their simplicity and efficacy. The former takes graphs as input and the latter takes features as input.

The models proposed in this article are closely related to these traditional clustering algorithms. Therefore, a brief review of them is presented first. For convenience, we use $\mathbf{X} = [\mathbf{x}_1, \cdots, \mathbf{x}_n]^T \in \mathbb{R}^{n\times d}$, $\mathbf{W} \in \mathbb{R}^{n\times n}$, and c to denote the dataset with n samples, the similarity matrix, and the number of clusters to construct, respectively.

Taking the spectral clustering method introduced in [9] as an example, it can be seen as a relaxation of the normalized-cut problem. The objective function of normalized-cut can be formulated as

$$\min_{\mathcal{A}_1,\cdots,\mathcal{A}_c} \sum_{k=1}^{c} \frac{\sum_{\mathbf{x}_i\in\mathcal{A}_i,\mathbf{x}_j\in\bar{\mathcal{A}}_i} w_{ij}}{vol(\mathcal{A}_k)}, \tag{1}$$

where $vol(\mathcal{A}_k) = \sum_{\mathbf{x}_i\in A_k}\sum_{j=1}^{n} w_{ij}$. The spectral clustering method [9] transforms the problem in Eq. (1) into an eigendecomposition problem by relaxing the discrete condition and often yields more superior experimental performance. However, the computational cost of the eigendecomposition involved increases quadratically with the number of samples, which is very time-consuming.

With these notations mentioned above, the objective function of k-means can be formulated as

$$\min_{\mathcal{A}_1,\cdots,\mathcal{A}_c} \sum_{k=1}^{c}\sum_{\mathbf{x}_i\in\mathcal{A}_k} \|\mathbf{x}_i - \mathbf{m}_k\|_2^2. \tag{2}$$

where $\mathbf{m}_k = \frac{1}{n_k}\sum_{\mathbf{x}_i\in\mathcal{A}_k}\mathbf{x}_i$ denotes the centroid of cluster $\mathcal{A}_k$ containing n_k samples. The goal of k-means is to minimize the sum of distances between the samples and the center in each cluster. In general, the computational overhead of k-means is linear with respect to the number of samples, but it cannot separate clusters non-linearly separable in input space, which is a major disadvantage.

In this paper, we are aiming at designing a fast clustering algorithm whose computational complexity increases

- *S. Pei, H. Chen, F. Nie (corresponding author) are with the School of Computer Science, School of Artificial Intelligence, Optics and Electronics (iOPEN), and the Key Laboratory of Intelligent Interaction and Applications (Ministry of Industry and Information Technology), Northwestern Polytechnical University, Xi'an 710072, Shaanxi, P. R. China. E-mail: shenfeipei@gmail.com, chenhuimin@mail.nwpu.edu.cn, feipingnie@gmail.com.*
- *R. Wang and X. Li are with the School of Artificial Intelligence, Optics and Electronics (iOPEN), and the Key Laboratory of Intelligent Interaction and Applications (Ministry of Industry and Information Technology), Northwestern Polytechnical University, Xi'an 710072, Shaanxi, P. R. China.*
  E-mail: wangrong07@tsinghua.org.cn, li@nwpu.edu.cn

Code release: https://github.com/ShenfeiPei/KSUMS2

1. https://www.statista.com/

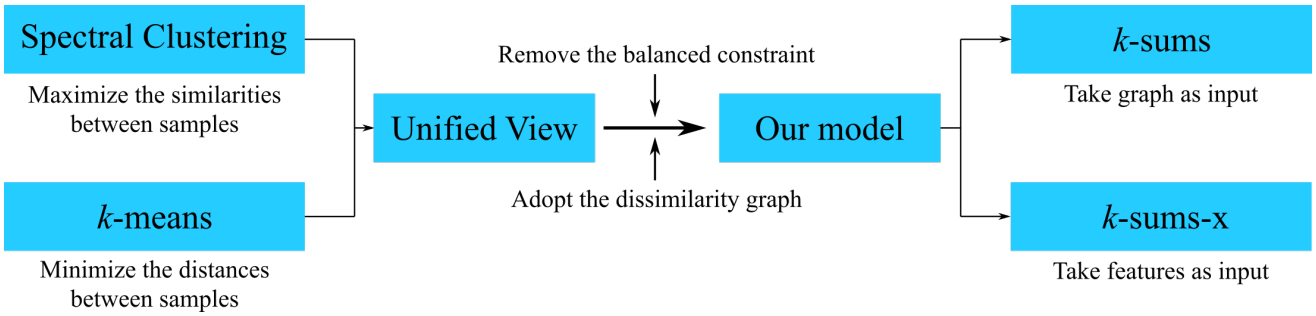


Fig. 1: The motivation of the proposed algorithms (k-sums and k-sums-x). (1) The unified view of k-means and spectral clustering is firstly revisited. (2) Then, our model is proposed by making some modifications to this unified view. (3) Two versions of the proposed method have been developed, called k-sums and k-sums-x. The former (k-sums) takes a k-NN graph as input and the latter takes the features as input.

linearly with the number of samples. Moreover, its performance is comparable to traditional spectral clustering.

The main idea of the proposed model comes from the unified view of k-means and spectral clustering. Specifically, we found that the unified view has a relatively simple form if we focus on the situation where the number of samples in each cluster is strictly equal, and the model (without the constraint) naturally tends to produce more balanced clustering results, if a dissimilarity matrix is used. The motivation of the algorithm is shown in Fig. 1.

Based on these findings, a novel clustering method that is dedicated to directly minimizing the sum of the distances between points in the same cluster is proposed and called k-sums. It is worth mentioning that it takes the nearest neighbor graph as input and only cares about the distance between points that are neighbors of each other. The standard coordinate descent (CD) algorithm is adopted to optimize our model, and an extremely efficient optimization algorithm is designed for the sub-problems involved in the CD algorithm. Therefore, the computational complexity of k-sums is greatly reduced.

In addition, in order to apply the proposed clustering algorithm in the situation where the nearest neighbour graph is not available, a version that takes features as input is also discussed and called k-sums-x. Again, the coordinate descent algorithm is used to optimize the problem of k-sums-x, and the property of squared Euclidean distance is used to reduce the computational overhead.

It is worthwhile to highlight the main contributions of this article as follows:

- Based on the unified view of k-means and spectral clustering, a novel clustering method that is dedicated to directly minimizing the sum of the distances between points in the same cluster is proposed. It cares about the squared Euclidean distance between points only, so, the trouble of measuring the similarity between points can be avoided.
- The optimization algorithm employed, i.e., coordinate descent method, guarantees that no empty cluster will appear in the clustering result, with any initialization. In particular, our model naturally tends to produce a more balanced partition, and a detailed explanation is provided.
- For k-sums, the version of the proposed model that takes a k-nearest neighbor graph as input, its computational and memory overhead are both linear with respect to the number of points. More importantly, the time complexity is independent of the product of the number of points and clusters. These properties mean that it is easily scalable and applicable to large-scale problems.
- To apply our model in the situation where a k-NN graph is not available, k-sums-x, the version of the proposed model that takes features as input, is also provided. With some pre-processing, its computational complexity is still linear with respect to the number of samples.
- In order to validate the superiority of our algorithms, extensive experiments have been conducted on 10 synthetic datasets, 13 middle-scale real-world datasets, and 4 large-scale datasets, and our algorithms have shown satisfactory results compared to the state-of-the-art algorithms.

The rest of this paper is organized as follows: In Section 2, we briefly introduce some algorithms related to spectral clustering or k-means. In Section 3, the unified view of k-means and spectral clustering is briefly revisited, and our model is presented based on the unified view. In Section 4, a variant of our model is proposed, and the computational and memory overhead are analyzed. The experimental results are reported in Section 5 and the conclusions are given in Section 6.

Notations: All matrices are denoted by boldface uppercase letters, and all vectors are represented by boldface lowercase letters. For matrix $\mathbf{M} = [\mathbf{m}_1, \cdots, \mathbf{m}_n]^T \in \mathbb{R}^{n\times d}$, m_{ij} and $\mathbf{m}_i$ are used to represent the (i,j)-th entry and transpose of the i-th row of matrix, $\mathbf{M}$, respectively. The squared l_2 norm of the vector $\mathbf{v} \in \mathbb{R}^n$ is defined as $\|\mathbf{v}\|_2^2 = \sum_{i=1}^n v_i^2$. $\mathbf{X} \in \mathbb{R}^{n\times d}$ is adopted to denote the dataset with n samples, where $\mathbf{x}_i$ denotes the i-the point, and c is used the number of clusters to construct. For matrix $\mathbf{Y}$, if there is only one element equal to 1 in $\mathbf{y}_i, \forall i = 1, \cdots, n$, and the rest elements are all 0, then $\mathbf{Y}$ is referred to cluster indicator matrix. We use $\Phi^{n\times c}$ to denote the set of all cluster indicator matrices. $\mathbf{1}$ is the column vector of all ones. $\mathbf{x}_j^{(i)}$ denotes the j-th sample in cluster i.

2 Related Work

Our model comes from the unified view of k-means and spectral clustering, so, we mainly introduce the related algorithms of the two.

2.1 Algorithms Related to Spectral Clustering

The workflow of SC can be summarized as follows: 1. Construct an undirected graph, i.e., $\mathbf{G} =< \mathbf{V}, \mathbf{W} >$, where $\mathbf{V}$ and $\mathbf{W}$ denote the collection of input patterns and the adjacency matrix, respectively. In general, w_{ij} represents the similarity between samples $\mathbf{x}_i$ and $\mathbf{x}_j$, and $\mathbf{W} \in \mathbb{R}^{n\times n}$ can be determined by the heat kernel as

$$w_{ij} = \begin{cases} e^{-\frac{\|\mathbf{x}_i - \mathbf{x}_j\|_2^2}{t}} & \text{if } \mathbf{x}_i \in \mathcal{N}_k(\mathbf{x}_j) \text{ or } \mathbf{x}_j \in \mathcal{N}_k(\mathbf{x}_i) \\ 0 & \text{otherwise} \end{cases} \quad (3)$$

where $\mathcal{N}_k(\mathbf{x}_i)$ denotes the set of k nearest neighbors of $\mathbf{x}_i$. $t > 0$ is a parameter. Obviously, $\mathbf{W}$ is a symmetric matrix. 2. Compute the Laplacian matrix $\mathbf{L} = \mathbf{\Delta} - \mathbf{W}$, where $\mathbf{\Delta}$ is a diagonal matrix and defined as $\Delta_{jj} = \sum_{i=1}^{n} w_{ij}$ 3. Compute the first c eigenvectors of the normalized Laplacian matrix $\mathbf{L}_{norm}$, with

$$\mathbf{L}_{norm} = \begin{cases} \mathbf{\Delta}^{-1/2}\mathbf{L}\mathbf{\Delta}^{-1/2} & \text{Ng et al [9]} \\ \mathbf{\Delta}^{-1}\mathbf{L} & \text{Shi and Malik [10]} \end{cases} \quad (4)$$

Despite its promising performance, it has a high computational overhead, which can be seen from the two stages of graph construction and eigendecomposition. To this end, A lot of effort has been made. Among these studies, Nyström-based and anchor-based methods gain the most popularity.

The Nystrom method can be seen as an algorithm for approximately solving eigendecomposition problems. The approximation works by first solving the eigendecomposition problem for a small random subset of points and then extrapolating this solution to the full set of data points. [11], [12], [13] applied this algorithm to accelerate the calculation of spectral clustering. [14] proposed a variational Nystrom method. Using fewer sampled points, it achieved a lower approximation error.

The anchor-based strategy is also a very popular method for accelerating the spectral clustering algorithm. These methods learn a sparse representation for each data point firstly. The similarity matrix is then designed as

$$\mathbf{W} = \mathbf{Z}^T\mathbf{\Delta}^{-1}\mathbf{Z}, \quad (5)$$

where $\mathbf{Z} \in \mathbb{R}^{m\times n}$ is the sparse representation, m denotes the number of anchors, $\mathbf{\Delta}$ is a diagonal matrix with $\Delta_{kk} = \sum_{j=1} z_{jk}$. Generally, the computational overhead of these methods is $O(nm^2)$.

[15] proposed a fast implementation of spectral clustering, where random projection and sampling methods are used for reducing the dimensionality and cardinality of data. [16] proposed a fast method, where the spectral clustering is performed on a hierarchical bipartite graph that much smaller than the similarity graph. In addition, the method is able to deal with the out-of-sample problem. For coping with large-scale clustering problems in the case of limited memory, [17], [18] proposed sequential SC algorithms. The time complexity of them is nearly linear with respect to the number of points, and the memory overhead is independent of the number of points.

2.2 Algorithms Related to k-means

k-means, as one of the most popular clustering algorithms, is dedicated to minimizing within-cluster variances (squared Euclidean distances). The main idea of k-means is to assign each point to the cluster whose center is closest to the point. With the help of the aforementioned notations, the objective function of k-means can be expressed as

$$\min_{\mathbf{Y}\in\Phi^{n\times c},\mathbf{M}} \|\mathbf{X} - \mathbf{Y}\mathbf{M}\|_F^2, \quad (6)$$

where $\mathbf{Y} \in \Phi^{n\times c}$ is a cluster indicator matrix, $\mathbf{M} = [\mathbf{m}_1, \cdots, \mathbf{m}_c]^T \in \mathbb{R}^{c\times d}$, $\mathbf{m}_i$ denotes the center (mean) of i-th cluster. The problem (6) is NP-hard, but many efficient optimization methods [27, 37, 13] are proposed.

Although problem (6) can be optimized efficiently, k-means has the following disadvantages:

- It cannot separate clusters that are not separable linearly in input space, which is a major disadvantage of it.
- The quality of the clustering result will be greatly affected by the initialization of the clustering partition.

To solve the first problem, kernel-based k-means was proposed [19], [20], where a feature function $\phi : \mathbb{R}^d \to \mathcal{F}$ is used to map the input features non-linearly into output vectors, and the traditional k-means is then performed in these vectors instead of the original features. However, most kernel-based k-means algorithms cannot be applied directly to large-scale problems for their high computational complexity. To this end, by employing a randomized approach, Chitta et al. [21] proposed an approximation scheme, termed approximate kernel k-means, where the computational and memory overhead are both reduced. [47, 17, 7] adopted the incomplete Cholesky factorization and Nystrom methods to solve approximately kernel learning problems. In order to overcome the influence of the initialization on performance, k-means++, a variant of k-means [22] which is widely used, is proposed. It seeded the initial centers by a specific rule and the initialization is provably competitive with the optimal solution. Despite its promising performance, k-means++ requires c full passes over the whole dataset. To this end, Bachem et al. [23] adopted Markov Chain Monte Carlo (MCMC) method to approximate k-means++ with a lower time complexity. In addition, Bachem et al. [24] proposed a fast seeding method, where the quality of the clustering result is provably competitive with the optimal solution.

From the above discussion, it can be seen that the k-means based and spectral-based clustering algorithms are loosely related. In order to establish the connection spectral clustering and k-means, a lot of effort has been done, which narrowed the gap between the two to some extent. An explicit theoretical connection between kernel k-means and spectral clustering has been proposed in [25], [26]. In addition, Ding et al. [27] has shown that Nonnegative Matrix Factorization (NMF) of $\mathbf{W}$, i.e.,

$$\min_{\mathbf{H}\geq 0} \|\mathbf{W} - \mathbf{H}\mathbf{H}^T\|_F^2 \quad (7)$$

is equivalent mathematically to kernel k-means with the strict orthogonality relation relaxed.

2.3 The Unified View

As mentioned before, the graph partitioning problem from which spectral clustering is derived can be formulated as

$$\min_{\mathbf{Y}\in\Phi^{n\times c}} Tr\left(\left(\mathbf{Y}^T\mathbf{Y}\right)^{-1}\mathbf{Y}^T\mathbf{L}\mathbf{Y}\right). \tag{8}$$

where $\mathbf{L}=\mathbf{\Delta}-\mathbf{W}$, $\mathbf{\Delta}$ is the degree matrix, $\Delta_{jj}=\sum_{i=1}^{n} w_{ij}$. In practice, a normalized Laplacian matrix $\mathbf{L}_{norm}$ is usually used, and the meaning of $\mathbf{L}_{norm}$ is different in different versions, as mentioned above. If the similarity matrix $\mathbf{W}$ is doubly stochastic, that is to say

$$\sum_{i=1}^{n} w_{ij}=1, \sum_{j=1}^{n} w_{ij}=1, \forall i,j=1,2,\cdots,n, \tag{9}$$

then, we have $\mathbf{\Delta}=\mathbf{I}$. Therefore the objective function in Eq. (8) can be simplified as

$$\max_{\mathbf{Y}\in\Phi^{n\times c}} Tr\left(\left(\mathbf{Y}^T\mathbf{Y}\right)^{-1}\mathbf{Y}^T\mathbf{W}\mathbf{Y}\right) \tag{10}$$

It's worth noting that many studies [28], [29] have shown that the performance of SC can be enhanced by employing the doubly stochastic matrix. Thus, the simplified model (10) is meaningful. Next, we pay attention to k-means and transform its objective function into a form similar to Eq. (10).

For convenience, we focus on the matrix form of k-means. It is not difficult to find that the following equations are hold.

$$\min_{\mathbf{Y}\in\Phi^{n\times c},\mathbf{M}} \|\mathbf{X}-\mathbf{Y}\mathbf{M}\|_F^2 \tag{11}$$

$$\Leftrightarrow \min_{\mathbf{Y}\in\Phi^{n\times c},\mathbf{M}} Tr\left(-2\mathbf{X}^T\mathbf{Y}\mathbf{M}+\mathbf{M}^T\mathbf{Y}^T\mathbf{Y}\mathbf{M}\right) \tag{12}$$

Taking the derivation of Eq. (12) with respect to $\mathbf{M}$, we have

$$-2\mathbf{Y}^T\mathbf{X}+2\mathbf{Y}^T\mathbf{Y}\mathbf{M}=0 \tag{13}$$

$$\mathbf{M}=\left(\mathbf{Y}^T\mathbf{Y}\right)^{-1}\mathbf{Y}^T\mathbf{X} \tag{14}$$

We call the cluster "empty cluster" if it contains no samples. Obviously, if no empty cluster appears in the clustering result, $\mathbf{Y}^T\mathbf{Y}$ is reversible.

Here, we only focus on the situation where $\mathbf{Y}^T\mathbf{Y}$ is reversible since the optimization algorithm adopted in this paper guarantees that no empty cluster will appear in the clustering result, in the proposed model.

Replacing $\mathbf{M}$ with $(\mathbf{Y}^T\mathbf{Y})^{-1}\mathbf{Y}^T\mathbf{X}$, we arrive at

$$\min_{\mathbf{Y}\in\Phi^{n\times c},\mathbf{M}} Tr\left(-2\mathbf{X}^T\mathbf{Y}\mathbf{M}+\mathbf{M}^T\mathbf{Y}^T\mathbf{Y}\mathbf{M}\right) \tag{15}$$

$$\Leftrightarrow \min_{\mathbf{Y}\in\Phi^{n\times c}} Tr\left(\left(-2\mathbf{X}^T\mathbf{Y}+\mathbf{X}^T\mathbf{Y}\right)\mathbf{M}\right) \tag{16}$$

$$\Leftrightarrow \max_{\mathbf{Y}\in\Phi^{n\times c}} Tr\left(\mathbf{X}^T\mathbf{Y}\left(\mathbf{Y}^T\mathbf{Y}\right)^{-1}\mathbf{Y}^T\mathbf{X}\right) \tag{17}$$

$$\Leftrightarrow \max_{\mathbf{Y}\in\Phi^{n\times c}} Tr\left(\left(\mathbf{Y}^T\mathbf{Y}\right)^{-1}\mathbf{Y}^T\mathbf{X}\mathbf{X}^T\mathbf{Y}\right) \tag{18}$$

It is not difficult to find that both the spectral clustering and k-means can be unified into the framework below.

$$\max Tr\left(\left(\mathbf{Y}^T\mathbf{Y}\right)^{-\frac{1}{2}}\right)\mathbf{Y}^T\mathbf{G}\mathbf{Y}\left(\mathbf{Y}^T\mathbf{Y}\right)^{-\frac{1}{2}}\Big). \tag{19}$$

Equation (19) denotes the objective function of spectral clustering if $\mathbf{G}=\mathbf{W}$, and the objective function of k-means if $\mathbf{G}=\mathbf{X}\mathbf{X}^T$.

The unified view discussed here is different from that discussed in [25], [26]. Specifically, they mainly explored the relationship between kernel k-means and spectral clustering and proved the equivalence of the two. In our article, we are committed to discussing the essential difference between traditional k-means and spectral clustering with doubly stochastic matrix.

2.4 The Difference and Connection Between k-means and Spectral Clustering

Here, we write the two into another unified mathematical form. Based on it, we found that the essential difference between the two is that the measurements they employed are different.

According to [30], k-means (2) can be rewritten as

$$\min_{\mathcal{A}_1,\cdots,\mathcal{A}_c} \sum_{k=1}^{c}\frac{1}{|\mathcal{A}_k|}\sum_{\mathbf{x}_i,\mathbf{x}_j\in A_k} d_{ij}^2 \tag{20}$$

where $d_{ij}^2=\|\mathbf{x}_i-\mathbf{x}_j\|_2^2$.

It is not difficult to find that the objective function of spectral clustering can be written similarly as

$$\max_{\mathcal{A}_1,\cdots,\mathcal{A}_c} \sum_{k=1}^{c}\frac{1}{|\mathcal{A}_k|}\sum_{\mathbf{x}_i,\mathbf{x}_j\in A_k} w_{ij} \tag{21}$$

Based on the discussion above, we have the following observations: 1. Spectral clustering adopted doubly stochastic matrix shares the same framework as the k-means, as shown in Eq. (19). 2. For these two algorithms, the only difference is that different metrics are used to measure the similarity between samples. 3. From another view, the goal of SC is to **maximize** the mean of the similarities between samples in the same cluster, while the goal of k-means is to **minimize** the mean of distances between samples in the same cluster, as shown in Eq. (20) and Eq. (21).

3 An Efficient Clustering Model

Taking the unified framework as starting point, we developed our model. Interestingly, our model has two forms: k-sums that take graph as input, and k-sums-x that take features as input. k-sums can be regard as a competitor proposed for the family of spectral clustering and k-sums-x can be seen as a competitor proposed for k-means-like algorithms, as shown in Figure 1.

3.1 Our Model

Although those clustering algorithms where a k-nearest neighbor graph is adopted usually yield better experimental performance, in terms of commonly used metrics, such as ACC, NMI, compared to other methods, their optimization algorithms are time-consuming usually.

Fortunately, we found that a simple model can be derived from the unified view shown in Eq. (19), if we focus on the situation where the number of samples in each cluster is strictly equal. In other words, we have the following equations

$$\max_{\mathbf{Y}\in\Phi^{n\times c}} Tr\left(\left(\mathbf{Y}^T\mathbf{Y}\right)^{-1}\mathbf{Y}^T\mathbf{G}\mathbf{Y}\right)\ \ s.t.\mathbf{Y}^T\mathbf{Y}=\bar{n}\mathbf{I} \tag{22}$$

$$\Leftrightarrow \max_{\mathbf{Y}\in\Phi^{n\times c}} Tr\left(\mathbf{Y}^T\mathbf{G}\mathbf{Y}\right)\quad s.t.\mathbf{Y}^T\mathbf{Y}=\bar{n}\mathbf{I} \tag{23}$$

We remove the balance constraint to make the problem easy to solve and adopted a distance matrix instead of a similarity matrix to avoid the trivial solution. Therefore the final objective function of our model is as follows

$$\min_{\mathbf{Y}\in\Phi^{n\times c}} Tr\left(\mathbf{Y}^T\widetilde{\mathbf{D}}\mathbf{Y}\right) \tag{24}$$

with

$$\tilde{d}_{ij} = \begin{cases} \|\mathbf{x}_i - \mathbf{x}_j\|_2^2, & x_i \leftrightarrow x_j \\ \gamma, & \text{otherwise} \end{cases} \tag{25}$$

where $\mathbf{x}_i \leftrightarrow \mathbf{x}_j$ means $\mathbf{x}_i \in \mathcal{N}_k(\mathbf{x}_j)$ and $\mathbf{x}_j \in \mathcal{N}_k(\mathbf{x}_i)$, γ means the maximum of the set $\mathcal{A}$ consisting of defined values, that is to say

$$\mathcal{A} = \{\|\mathbf{x}_i - \mathbf{x}_j\|_2^2 \mid \mathbf{x}_i \leftrightarrow \mathbf{x}_j, i = 1, \cdots, n, j = 1 \cdots, n\}. \tag{26}$$

The proposed model aims to minimize the sum of the squared Euclidean distances between points in the same cluster, so we term our model as k-sums.

3.2 Analysis

To begin with, some notations are presented here. Note that these notations only work in this subsection. Let $\mathcal{A}(\mathbf{x}_i)$ be the set of $\{\mathbf{x}_l \mid \mathbf{x}_l \in \mathcal{N}_k(\mathbf{x}_i) \text{ and } \mathbf{x}_i \in \mathcal{N}_k(\mathbf{x}_l)\}$, $\mathcal{F}_i$ be the set of samples in cluster i, and $\mathbf{x}_j^{(i)}$ be the j-th sample in cluster i, we use the following notations,

$$\begin{cases} k_i^* = |\mathcal{A}(\mathbf{x}_i)|, \\ k_{ij}^* = |\mathcal{A}(\mathbf{x}_j^{(i)})|, \\ k_{ij} = |\mathcal{A}(\mathbf{x}_j^{(i)}) \cap \mathcal{F}_i|, \\ \bar{k}_{ij} = |\mathcal{A}(\mathbf{x}_j^{(i)}) - \mathcal{F}_i|, \end{cases}$$

to represent the number of samples in corresponding sets. Then, we have

$$k_{ij}^* = k_{ij} + \bar{k}_{ij}. \tag{27}$$

It means that the samples in $\mathcal{A}(\mathbf{x}_i)$ can be divided into two categories (in or not in the i-th cluster, $\mathcal{F}_i$).

Similarly, using the following notations

$$\begin{cases} s_i^* = \sum_{\mathbf{x}_j\in\mathcal{B}} d(\mathbf{x}_i, \mathbf{x}_j), \mathcal{B} = \mathcal{A}(\mathbf{x}_i), \\ s_{ij}^* = \sum_{\mathbf{x}_l\in\mathcal{B}} d(\mathbf{x}_j^{(i)}, \mathbf{x}_l), \mathcal{B} = \mathcal{A}(\mathbf{x}_j^{(i)}), \\ s_{ij} = \sum_{\mathbf{x}_l\in\mathcal{B}} d(\mathbf{x}_j^{(i)}, \mathbf{x}_l), \mathcal{B} = \mathcal{A}(\mathbf{x}_j^{(i)}) \cap \mathcal{F}_i, \\ \bar{s}_{ij} = \sum_{\mathbf{x}_l\in\mathcal{B}} d(\mathbf{x}_j^{(i)}, \mathbf{x}_l), \mathcal{B} = \mathcal{A}(\mathbf{x}_j^{(i)}) - \mathcal{F}_i, \end{cases}$$

with

$$d(\mathbf{x}_i, \mathbf{x}_j) = \|\mathbf{x}_i - \mathbf{x}_j\|_2^2,$$

to represent the sum of distances in these sets, we have

$$s_{ij}^* = s_{ij} + \bar{s}_{ij}. \tag{28}$$

That is to say, for distance $d(\mathbf{x}_j^{(i)}, \mathbf{x}_l)$, there are two cases: (1) $\mathbf{x}_l$ belongs to $\mathcal{F}_i$; (2) $\mathbf{x}_l$ does not belong to $\mathcal{F}_i$, which is similar to Equation (27).

With these notations, we can rewrite our model Eq. (24) into a more detailed form, from which we can see how our model works.

$$\sum_{i=1}^{c}\sum_{j=1}^{n_i}(s_{ij} + (n_i - k_{ij})\gamma) \tag{29}$$

$$=\gamma\sum_{i=1}^{c} n_i^2 + \sum_{i=1}^{c}\sum_{j=1}^{n_i}(s_{ij}^* - \bar{s}_{ij}) - \sum_{i=1}^{c}\sum_{j=1}^{n_i}(k_{ij}^* - \bar{k}_{ij})\,\gamma \tag{30}$$

$$=\gamma\sum_{i=1}^{c} n_i^2 + \sum_{i=1}^{n}(s_i^* - \gamma k_i^*) - \sum_{i=1}^{c}\sum_{j=1}^{n_i}\bar{s}_{ij} + \gamma\sum_{i=1}^{c}\sum_{j=1}^{n_i}\bar{k}_{ij} \tag{31}$$

Because s_i^* and k_i^* are constants, the problem (24) can be written equivalently as

$$\min_{\mathcal{F}_1,\cdots,\mathcal{F}_c} \gamma\sum_{i=1}^{c} n_i^2 - \sum_{i=1}^{c}\sum_{j=1}^{n_i}\bar{s}_{ij} + \gamma\sum_{i=1}^{c}\sum_{j=1}^{n_i}\bar{k}_{ij} \tag{32}$$

Assuming $d(\mathbf{x}_i, \mathbf{x}_j) = e, \forall \mathbf{x}_i \in \mathbf{X}, \mathbf{x}_j \in \mathcal{A}(\mathbf{x}_i)$, the problem (32) can be simplified further as

$$\min_{\mathcal{F}_1,\cdots,\mathcal{F}_c} \gamma\sum_{i=1}^{c} n_i^2 + (\gamma - e)\sum_{i=1}^{c}\sum_{j=1}^{n_i}\bar{k}_{ij} \tag{33}$$

To make the clustering results meaningful, $\gamma > e$ is required.

It can be seen from Eq. (32) that (i). our model tends to produce more balanced clustering results (the first term), (ii). the proposed model tends to group $\mathbf{x}_i$ and its neighbors into the same cluster (the third term), (iii). if grouping $\mathbf{x}_i$ and its neighbors into different clusters will result in a smaller objective function value, then the model will give priority to the neighbors that are far away from the sample.

The simplified model (33) retains the properties (i) and (ii) of the model (32), which means the proposed model is also applicable to unweighted graphs.

3.3 Graph Construction

In practical applications, a weighted graph containing similarities between samples is more common. In other words, the squared Euclidean distance between two points that are connected in the graph can not be obtained directly. In this situation, it is recommended to use the following formula to convert the similarities into the distances.

$$\tilde{d}_{ij} = -log\left(s_{ij}\right), \tag{34}$$

where s_{ij} represents the similarity between points $\mathbf{x}_i$ and $\mathbf{x}_j$. In some situations, only features are provided, at this time, it is recommended to construct a k-NN graph using fast approximate nearest neighbor algorithms. In this article, the k-d tree and EFANNA are employed to construct the approximate nearest neighbor graph required by the model for synthetic and real-world datasets, respectively.

3.4 Initialization

An intuitive way is to group the closer samples (the distance between the two is less than γ, for example) into one cluster. However, it is difficult for us to find an appropriate value for γ. In this paper, we treat the nearest neighbours as connected points and establish a path for each sample by breadth-first search method. Then group the first n/c points of the path into one cluster. If the number of clusters obtained is more than c, the clusters with the smaller number of samples are randomly assigned to other clusters.

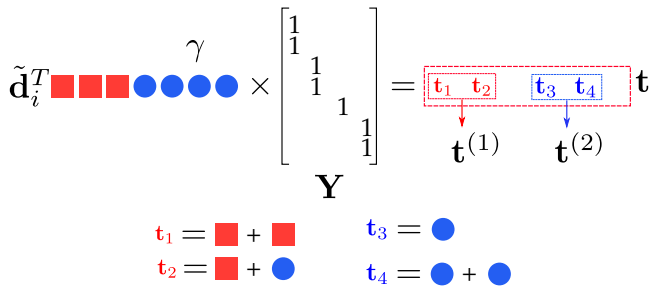


Fig. 2: Schematic diagram of calculating the product of $\tilde{\mathbf{d}}_i^T$ and $\mathbf{Y}$. Red squares represent the squared Euclidean distances between samples, and blue circles denote the elements that are equal to γ.

3.5 Optimization

To solve the i-th row of $\mathbf{Y}$, we fixed the other rows as constants. To begin with, let $\mathbf{U}_{(i)}$ be the permutation matrix to swap the i-th and first rows and $\mathbf{Y}_r^T$ be the matrix $\mathbf{U}_{(i)}\mathbf{Y}$ with the first row removed, we have

$$\min_{\mathbf{y}_i} Tr\left(\mathbf{Y}^T\tilde{\mathbf{D}}\mathbf{Y}\right) \tag{35}$$

$$\Leftrightarrow \min_{\mathbf{y}_i} Tr\left(\mathbf{Y}^T\mathbf{U}_{(i)}^T\mathbf{U}_{(i)}\tilde{\mathbf{D}}\mathbf{U}_{(i)}^T\mathbf{U}_{(i)}\mathbf{Y}\right) \tag{36}$$

$$\Leftrightarrow \min_{\mathbf{y}_i} Tr\left(\begin{bmatrix}\mathbf{y}_i & \mathbf{Y}_r\end{bmatrix}\begin{bmatrix}\tilde{d}_{ii} & \mathbf{v}^T\\ \mathbf{v} & \tilde{\mathbf{D}}_r\end{bmatrix}\begin{bmatrix}\mathbf{y}_i^T\\ \mathbf{Y}_r^T\end{bmatrix}\right) \tag{37}$$

$$\Leftrightarrow \min_{\mathbf{y}_i} Tr\left(\mathbf{y}_i\tilde{d}_{ii}\mathbf{y}_i^T\right) + 2Tr\left(\mathbf{Y}_r\mathbf{v}\mathbf{y}_i^T\right) \tag{38}$$

where $\mathbf{v}^T = \begin{bmatrix}\tilde{d}_{i,2} & \cdots & \tilde{d}_{i,i-1} & \tilde{d}_{i,1} & \tilde{d}_{i,i+1} & \cdots & \tilde{d}_{i,n}\end{bmatrix}$. Since $\tilde{d}_{ii} = 0$, the model can be further simplified to

$$\min_{\mathbf{y}_i} Tr\left(\mathbf{Y}_r\mathbf{v}\mathbf{y}_i^T\right) \tag{39}$$

$$\Leftrightarrow \min_{\mathbf{y}_i} \mathbf{y}_i^T\mathbf{Y}_r\mathbf{v} \tag{40}$$

Obviously $\mathbf{Y}_r\mathbf{v} = \mathbf{Y}^T\tilde{\mathbf{d}}_i$, so problem (40) is equivalent to

$$\min_{\mathbf{y}_i} Tr\left(\mathbf{y}_i^T\mathbf{Y}^T\tilde{\mathbf{d}}_i\right) \tag{41}$$

From Eq. (35) to Eq. (41), we know that our model (24) can be optimized by solving Eq. (41) ($\forall i = 1, \cdots, n$) iteratively until converging.

It is not difficult to find that the optimal solution of the i-th row of $\mathbf{Y}$, $\mathbf{y}_i$ can be obtained by

$$y_{il} = \begin{cases} 1 & l = \arg\min_j \left(\tilde{\mathbf{d}}_i^T\mathbf{Y}\right)_j \\ 0 & \text{otherwise}\end{cases} \tag{42}$$

3.6 Acceleration

Without any acceleration technique, it takes $O(nc)$ time to calculate the product of $\tilde{\mathbf{d}}_i^T$ and $\mathbf{Y}$. Note that $\mathbf{Y}$ is an indicator matrix, that is, there are only n non-zero elements in $\mathbf{Y}$. Therefore, calculating the product of $\tilde{\mathbf{d}}_i^T$ and $\mathbf{Y}$ can be completed in $O(n)$ time. However, the computational cost of the entire algorithm is $O(n^2)$, which means that the algorithm is difficultly scalable to large-scale problems

We denote the elements in $\tilde{\mathbf{d}}_i^T$ that are smaller than γ by red squares, and denote the elements that are equal to γ by blue circles, as shown in Fig. 2. Let $\mathbf{t}$ denote the product of $\tilde{\mathbf{d}}_i^T$ and $\mathbf{Y}$, i.e., $\mathbf{t} = \tilde{\mathbf{d}}_i^T\mathbf{Y}$, t_j can be regarded as the sum of some elements in $\tilde{\mathbf{d}}_i^T$. Therefore, there are k elements at most of $\mathbf{t}$, called $t^{(1)}$, that is related to the red squares, and the remaining elements, called $t^{(2)}$, are only related to the blue circles. We denote the j-th element in $\mathbf{t}^{(2)}$ by the i-th element in $\mathbf{t}$, i.e., $t_j^{(2)} = t_i^{(1)}$. Then we have

$$t_j^{(2)} = n_i\gamma. \tag{43}$$

If we record the number of samples of each cluster at the beginning of the optimization, then the elements in $\mathbf{t}^{(2)}$ can be directly obtained by Eq. (43) when solving $\mathbf{y}_i$. In addition, since the coordinate descent algorithm is adopted, only $O(1)$ time is needed to update the number of samples in each cluster if $\mathbf{y}_i$ is updated. So far, the computational overhead of our algorithm is $O(nk + nc)$. The detailed algorithm to solve the problem (24) is summarized in Algorithm 1.

Algorithm 1: Accelerated algorithm for solving problem (24).

Data: $\tilde{\mathbf{D}} \in \mathbb{R}^{n\times n}$, the number of cluster, c.
Result: Clustering result, $\mathbf{g}$.
Initialize $\mathbf{g}$ randomly;
Compute $\mathbf{n}$ (n_i is the number of points in cluster i);
Initialize vectors $\mathbf{u} \in \mathbb{R}^c$, $\mathbf{t} \in \mathbb{R}^c$, and $\mathbf{v} \in \mathbb{R}^c$ with 0;
while *not converge* **do**
  for $i = 1, \cdots, n$ **do**
    $\tilde{c} = g_i$;
    $\mathcal{A}(\mathbf{x}_i) = \{\mathbf{x}_j | \mathbf{x}_j \in \mathcal{N}_k(\mathbf{x}_i)$ and $\mathbf{x}_i \in \mathcal{N}_k(\mathbf{x}_j)\}$;
    $\mathbf{t}, \mathbf{v} = f_1(\tilde{\mathbf{d}}_i, \mathbf{g}, \mathbf{n}, \mathbf{u}, \mathbf{v}, \mathbf{t}, \mathcal{A}(\mathbf{x}_i))$;
    // Compute the elements in $\mathbf{t}^{(2)}$;
    for $l = 1 \cdots c$ **do**
      if $v_l == 0$ **then** $t_l = n_l * \gamma$;
    for $\mathbf{x}_j \in \mathcal{A}(\mathbf{x}_i)$ **do**
      $l = g_j$; $v_l = 0$;
    Find the minimum of $\mathbf{t}$, denote it by $t_{\hat{c}}$;
    $g_i = \hat{c}$; $n_{\tilde{c}} -= 1$; $n_{\hat{c}} += 1$;

3.6.1 Speed Up Further

Although the computational overhead of Algorithm 1 is linear with respect to the number of samples, it involves the product of the number of points and clusters. Therefore, it is very time-consuming, if the number of clusters to construct is very large.

We find the minimum value of $\mathbf{t}$ by

$$t^* = \begin{cases} \min(\mathbf{t}^{(1)}), & \text{if } \min(\mathbf{t}^{(1)}) <= \min(\mathbf{t}^{(2)}) \\ \min(\mathbf{t}^{(2)}), & \text{otherwise}\end{cases} \tag{44}$$

where t^* denotes the minimum of $\mathbf{t}$. Obviously, it takes $O(k)$ and $O(c-k)$ time to find the minimum of $\mathbf{t}^{(1)}$ and $\mathbf{t}^{(2)}$, respectively. Fortunately, the elements in $\mathbf{t}^{(2)}$ are all multiples of γ, thus, the cluster with the smallest number of samples in $\mathbf{t}^{(2)}$ corresponds to its minimum value, so we can find its minimum in a faster way.

Before introducing the acceleration algorithm, the following lemma is introduced.

Algorithm 2: The function used to calculate $\mathbf{t}^{(1)}$, called $f_1(\tilde{\mathbf{d}}_i, \mathbf{g}, \mathbf{n}, \mathbf{u}, \mathbf{v}, \mathbf{t}, \mathcal{A}(\mathbf{x}_i))$.

Data: $\tilde{\mathbf{d}}_i \in \mathbb{R}^n$, $\mathbf{g} \in \mathbb{R}^n, \mathbf{n}, \mathbf{u}, \mathbf{v}, \mathbf{t} \in \mathbb{R}^c, \mathcal{A}(\mathbf{x}_i)$
Result: Vector $\mathbf{t}$ (t_j has been calculated if $t_j \in \mathbf{t}^{(1)}$).
Vector $\mathbf{v}$ that is modified.
for $\mathbf{x}_j \in \mathcal{A}(\mathbf{x}_i)$ **do**
  $l = g_j$; $t_l = 0$; $u_l = 0$;
for $\mathbf{x}_j \in \mathcal{A}(\mathbf{x}_i)$ **do**
  $l = g_j$; $t_l += \tilde{d}_{ij}$; $u_l += 1$;
for $\mathbf{x}_j \in \mathcal{A}(\mathbf{x}_i)$ **do**
  $l = g_j$;
  if $v_l == 0$ **then**
    $v_l = 1$;
    $t_l += \gamma(n_l - u_l)$;

Lemma 1. *Let* $\mathbf{n} = [n_1, \cdots, n_c]^T \in \mathbb{R}^c$ *denote an ordered vector, where* $0 \leq n_i \leq N$, *and* $N < \infty$ *is a positive integer. It takes only* $O(1)$ *time to maintain its order if one element of it is increased or reduced by* 1.

The vector $\mathbf{n}$ representing the number of samples of each cluster just satisfies the conditions described Lemma 2, thus, it takes only $O(1)$ time to maintain its order if $\mathbf{y}_i$ is updated. In other words, if we sort $\mathbf{n}$ at the beginning of optimization, it takes $O(k)$ time at most to get the minimum value of $\mathbf{t}^{(2)}$.

So far, the computational overhead of our algorithm has been reduced to $O(nk)$. However, in practice, we find the minimum value of t through the simpler formula (45) instead of the formula (44).

$$\min(\mathbf{t}) = \begin{cases} \min(\mathbf{t}^{(1)}), & \text{if } \min(\mathbf{t}^{(1)}) <= \gamma q_1 \\ \gamma q_1, & \text{otherwise} \end{cases} \tag{45}$$

where q_1 is the number of samples in p_1 which is the cluster with the smallest number of samples in $\mathbf{t}$.

Equation (45) holds because of the following facts:

- γq_1 is an element of $\mathbf{t}^{(2)}$. In this case, $\gamma q_1 = \min(\mathbf{t}^{(2)})$, and Eq. (45) is then equivalent to Eq. (44).
- γq_1 does not belong to $\mathbf{t}^{(2)}$. In this case, we have

$$\min(\mathbf{t}^{(1)}) <= \gamma q_1 <= \min(\mathbf{t}^{(2)}).$$

Thus, $\min(\mathbf{t}^{(1)})$ is the minimum of $\mathbf{t}$, Eq. (45) holds.

The finally adopted algorithm for solving the problem (24) is summarized in Algorithm 3.

In short, we have carefully designed the optimization algorithm for problem (24), which reduces the computational complexity from $O(n^2c)$ to $O(nk)$, as shown in Table 1, where k is the number of neighborhoods in graph construction, not the number of clusters.

TABLE 1: The time complexity of algorithms

Cases	Time complexity
Original	$O(n^2c)$
Consider the sparsity of $\mathbf{Y}$	$O(n^2)$
Consider the property of $\tilde{\mathbf{d}}_i^T$	$O(n(k+c))$
The final optimization algorithm	$O(nk)$

Algorithm 3: The finally adopted algorithm for solving problem (24).

Data: $\tilde{\mathbf{D}} \in \mathbb{R}^{n \times n}$, the number of cluster, c.
Result: Clustering result, $\mathbf{g}$.
Initialize $\mathbf{g}$ randomly;
Initialize vectors $\mathbf{u} \in \mathbb{R}^c$, $\mathbf{t} \in \mathbb{R}^c$, and $\mathbf{v} \in \mathbb{R}^c$ with 0;
Compute $\mathbf{n}$ (n_i is the number of points in cluster i);
// Sort $\mathbf{n} \in \mathbb{R}^c$ by bucket sort algorithm, $n[p_i] = q_i$;
$\mathbf{p}, \mathbf{q} = sort(\mathbf{n})$;
while *not converge* **do**
  for $i = 1, \cdots, n$ **do**
    $\tilde{c} = g_i$;
    $\mathcal{A}(\mathbf{x}_i) = \{\mathbf{x}_j | \mathbf{x}_j \in \mathcal{N}_k(\mathbf{x}_i)$ and $\mathbf{x}_i \in \mathcal{N}_k(\mathbf{x}_j)\}$;
    $\mathbf{t}, \mathbf{v} = f_1(\tilde{\mathbf{d}}_i, \mathbf{g}, \mathbf{n}, \mathbf{u}, \mathbf{v}, \mathbf{t}, \mathcal{A}(\mathbf{x}_i))$;
    for $\mathbf{x}_j \in \mathcal{A}(\mathbf{x}_i)$ **do**
      $l = g_j$; $v_l = 0$;
    Get the minimum of $\mathbf{t}^{(1)}$;
    Obtain the minimum of $\mathbf{t}$ by judging whether $min(\mathbf{t}^{(1)}) < \gamma q_1$ holds, and denote it by $t_{\hat{c}}$;
    $g_i = \hat{c}$;
    $n_{\tilde{c}} -= 1$; $n_{\hat{c}} += 1$;
    Update $\mathbf{p}$ and $\mathbf{q}$ based on Lemma 1. ;

3.7 Time and space complexity

- Space complexity: In finally adopted algorithm, the memory is mainly occupied by matrix $\widetilde{\mathbf{D}}$, which is a dense matrix. But with the property shown in Eq. (32), we can find that there are at most nk elements in $\widetilde{\mathbf{D}}$ that need to be stored. Thus, the space complexity of the algorithm is $O(nk)$.
- Time complexity: $O(n)$ time is needed to sort $\mathbf{n}$ if the bucket sort algorithm is adopted. $O(k)$ time is needed to find the minimum of $\mathbf{t}^{(1)}$, as shown in Algorithm 2. According to Lemma 1, it takes only $O(1)$ time to update $\mathbf{p}$ and $\mathbf{q}$. Hence, the computational overhead of Algorithm 3 is $O(nk)$.

3.8 Advantages

Our model has the following advantages:

- **Efficient.** An extremely efficient optimization algorithm is designed for the problem (24), and the time and space complexity is greatly reduced. Specifically, the computational and memory overhead are both $O(nk)$, which has been discussed in subsection 3.7;
- **Effective.** Since a k-NN graph is adopted in k-sums, it can separate clusters that are not separable linearly in input space. Thus, it should be yield better experimental performance, which is verified in Section 5;
- **Robust.** The distance between $\mathbf{x}_i$ and $\mathbf{x}_j$ not calculated if $\mathbf{x}_i \notin \mathcal{N}_k(\mathbf{x}_j)$ or $\mathbf{x}_j \notin \mathcal{N}_k(\mathbf{x}_i)$, but a parameter γ is used instead, which means that our model is robust to outliers, and this point is verified by the experiment performed on "Outlier", in Section 5.

In addition, our model only involves the squared Euclidean distance, so it has no trouble measuring the similarities between samples.

4 A VARIANT OF K-SUMS

If the dimensionality of the data is high, even if the value of k is appropriate, the quality of the graph will be poor, which will lead to poor performance of the clustering algorithm. In addition, it is non-trivial to chose an appropriate k value for all datasets. Thus, it is desirable to get an algorithm that take features as input. We consider the following model and refer to it k-sums-x.

$$\min_{\mathbf{Y}\in\Phi^{n\times c}} Tr\left(\mathbf{Y}^T\mathbf{D}\mathbf{Y}\right), \tag{46}$$

where $\mathbf{D}\in\mathbb{R}^{n\times n}$ denotes the distance matrix, $d_{ij}=\|\mathbf{x}_i-\mathbf{x}_j\|_2^2$. Again, the standard CD algorithm is used to optimize the problem (46). Because the matrix $\mathbf{D}$ involved in the model is a dense matrix of size n by n, the following two issues that users care about are naturally raised.

- The dense matrix $\mathbf{D}$ is included in the proposed model. Although $\mathbf{D}$ is symmetric, there are still n elements with different values. Does the algorithm need to store the values of these elements? In that case, the algorithm will have a high memory overhead.
- If the standard CD algorithm is used to optimize the problem (46), the product of d and Y will still be involved in the sub-problem. However, all elements of di may not be equal, so the acceleration technique described earlier does obviously not work, which may cause the computational overhead to increase quadratically with the number of samples.

Considering the problems discussed above, we designed a new acceleration technology for the sub-problems involved in the coordinate descent algorithm. This method relies on the property of the squared Euclidean distance, which is one of the reasons why the squared Euclidean distance is used instead of other distances.

It is worth noting that by adopting the proposed acceleration algorithm, the computational and memory overhead are both reduced largely reduced. Specifically, The time and space complexity of the proposed optimization algorithm are both linear with respect to the number of points.

4.1 Analysis

Similar to the previously proposed model, model (46) also tends to produce a more balanced partition. Specifically, the following equations hold.

$$\min_{\mathbf{Y}\in\Phi^{n\times c}} Tr\left(\mathbf{Y}^T\mathbf{D}\mathbf{Y}\right) \tag{47}$$

$$\Leftrightarrow \min_{\mathbf{Y}\in\Phi^{n\times c}} \sum_{k=1}^{c}\sum_{\mathbf{x}_i,\mathbf{x}_j\in\mathcal{A}_k} d_{ij} \tag{48}$$

$$\Leftrightarrow \min_{\mathbf{Y}\in\Phi^{n\times c}} \sum_{k=1}^{c} n_i^2\bar{d}_i \tag{49}$$

where $\bar{d}_i$ denotes the average of the distances between samples in the i-th cluster, $\bar{d}_i=\frac{1}{n_i^2}\sum_{\mathbf{x}_i,\mathbf{x}_j\in\mathcal{A}_k} d_{ij}$. We consider the case where the average distance of each cluster is approximately equal, i.e., $\bar{d}_1\approx\cdots\approx\bar{d}_c$. Then the problem (49) can be written equivalently as

$$\min_{\mathbf{Y}\in\Phi^{n\times c}} \sum_{k=1}^{c} n_i^2. \tag{50}$$

Obviously, the $n_i=n/c,\forall i=1,\cdots,c$ is the optimal solution of the problem (50).

4.2 Connection with k-means

Theorem 2. *The problem (46) is equivalent to the problem of k-means 2, if we constrain the partitions must be balanced ($\mathbf{Y}^T\mathbf{Y}=\beta I$, where β is a constant).*

Proof. With the constraint of $\mathbf{Y}^T\mathbf{Y}=\beta I$, problem (46) can be rewritten as follows.

$$\min_{\mathbf{Y}\in\Phi^{n\times c}} Tr\left(\mathbf{Y}^T\mathbf{D}\mathbf{Y}\right) \tag{51}$$

$$\Leftrightarrow \min_{\mathbf{Y}\in\Phi^{n\times c}} Tr\left(\left(\mathbf{Y}^T\mathbf{Y}\right)^{-1}\mathbf{Y}^T\mathbf{D}\mathbf{Y}\right) \tag{52}$$

$$\Leftrightarrow \min_{\mathbf{Y}\in\Phi^{n\times c}} diag\left(\left(\mathbf{Y}^T\mathbf{Y}\right)^{-1}\right)^T diag\left(\mathbf{Y}^T\mathbf{D}\mathbf{Y}\right) \tag{53}$$

$$\Leftrightarrow \min_{\mathbf{Y}\in\Phi^{n\times c}} \sum_{k=1}^{c}\frac{1}{n_k}\sum_{\mathbf{x}_i,\mathbf{x}_j\in\mathcal{A}_k}\|\mathbf{x}_i-\mathbf{x}_j\|_2^2 \tag{54}$$

where, $diag(\mathbf{A})$ is the function to extract the diagonal vector from the matrix, $\mathbf{A}$, i.e, $diag(\mathbf{A})=[a_{1,1},\cdots,a_{n,n}]^T$. The Eq. (54) is exactly the problem of k-means as shown in Section 2.4, completing the proof. □

4.3 Optimization

Similar to k-sums, the coordinate descent algorithm is used again to solve problem (46). Therefore, the optimal solution of $\mathbf{y}_i$ can be obtained by

$$y_{il}=\begin{cases}1 & l=\arg\min_j\left(\mathbf{d}_i^T\mathbf{Y}\right)_j\\ 0 & \text{otherwise}\end{cases} \tag{55}$$

Let $\mathbf{t}$ represent the product of $\mathbf{d}_i^T$ and $\mathbf{Y}$, i.e, $\mathbf{t}=\mathbf{d}_i^T\mathbf{Y}$, then t_j can be regarded as the sum of some elements of $\mathbf{d}_i$. Fortunately, we have a simpler formula for summing these squared Euclidean distances as shown in Figure **??**.

$$\sum_{i=1}^{n}\|\mathbf{z}-\mathbf{x}_i\|_2^2=n\|z\|_2^2+\sum_{i=1}^{n}\|\mathbf{x}_i\|_2^2-2\mathbf{z}^T\sum_{i=1}^{n}\mathbf{x}_i \tag{56}$$

With the help of Formula (56), the k-th element of $\mathbf{t}$ can be expressed as

$$t_k=\sum_{\mathbf{x}_j\in\mathcal{A}_k}\|\mathbf{x}_i\|_2^2+\sum_{\mathbf{x}_j\in\mathcal{A}_k}\|\mathbf{x}_j\|_2^2-2\mathbf{x}_i^T\sum_{\mathbf{x}_j\in\mathcal{A}_k}\mathbf{x}_j \tag{57}$$

For convenience, let

$$\mathbf{S}\in\mathbb{R}^{c\times d},\mathbf{s}_k=\sum_{\mathbf{x}_j\in\mathcal{A}_k}\mathbf{x}_j, \tag{58}$$

$$\mathbf{u}\in\mathbb{R}^n,u_j=\|x_j\|_2^2, \tag{59}$$

$$\mathbf{n}\in\mathbb{R}^c,n_k=\sum_{\mathbf{x}_j\in\mathcal{A}_k}1 \tag{60}$$

$$\mathbf{v}\in\mathbb{R}^c,v_k=\sum_{\mathbf{x}_j\in\mathcal{A}_k}\|\mathbf{x}_j\|_2^2, \tag{61}$$

the vector $\mathbf{t}$ can be expressed as

$$t_k=n_ku_i+v_k-2\mathbf{x}_i^T\mathbf{s}_k, \tag{62}$$

and can be calculated in $O(dc)$ time, if the variables (n_k, u_k, v_k, and $\mathbf{s}_k$) are computed in advance. In addition, it is not difficult to find that we can update variables $\mathbf{s}_k$, v_k, and n_k in only $O(d)$ time, if $\mathbf{y}_i$ is updated. Therefore, the computational complexity of the whole algorithm is $O(ndc)$.

Algorithm 4: The finally adopted algorithm for solving problem (46).

Data: $\mathbf{X} \in \mathbb{R}^{n\times d}$, the number of cluster, c.
Result: Clustering result, $\mathbf{g}$.
Initialize $\mathbf{g}$ randomly;
Calculate the l_2 norm of each point, i.e., $\|\mathbf{x}_i\|, \forall i = 1, \cdots, n$);
Compute $\mathbf{n}$ (n_i is the number of points in cluster i);
Compute $\mathbf{v}$ and $\mathbf{s}$ according to Eq. (61) and Eq. (58);
while *not converge* **do**
  for $i = 1, \cdots, n$ **do**
    $\tilde{c} = g_i$;
    Compute $\mathbf{t}$ according to Eq. (57);
    Obtain the minimum of $\mathbf{t}$ and denote it by $t_{\hat{c}}$;
    // Update $\mathbf{g}$;
    $g_i = \hat{c}$;
    // Update $\mathbf{n}$;
    $n_{\tilde{c}} -= 1$; $n_{\hat{c}} += 1$;
    // Update $\mathbf{v}$;
    $v_{\tilde{c}} -= \|\mathbf{x}_i\|$; $v_{\hat{c}} += \|\mathbf{x}_i\|$;
    // Update $\mathbf{S}$;
    $s_{\tilde{c}} -= \mathbf{x}_i$; $s_{\hat{c}} += \mathbf{x}_i$;

4.4 Time and space complexity

- Space complexity: Variables $\mathbf{X} \in \mathbb{R}^{n\times d}$, $\mathbf{S} \in \mathbb{R}^{c\times d}$, $\mathbf{u} \in \mathbb{R}^n$, $\mathbf{n} \in \mathbb{R}^c$, and $\mathbf{v} \in \mathbb{R}^c$ need to be stored, and the temporary variables involved in the algorithm occupy much less memory than these variables, so the space complexity is $O(nd)$, which is linear with respect to the number of points.
- Time complexity: It takes $O(n)$, $O(nd)$, $O(n)$, and $O(nd)$ time to calculate variables $\mathbf{n}$, $\mathbf{u}$, $\mathbf{v}$, and $\mathbf{S}$ in advance, and it takes $O(1)$, $O(1)$, and $O(d)$ time to update variables $\mathbf{n}$, $\mathbf{v}$, and $\mathbf{S}$, if y_i is updated. In addition, the vector $\mathbf{t}$ can be computed in $O(dc)$ time if these variables are computed in advance. $O(c)$ time is need to find the minimum value of $\mathbf{t}$. Therefore, the computational overhead of the proposed algorithm is $O(ndc)$, which is the same as traditional k-means.

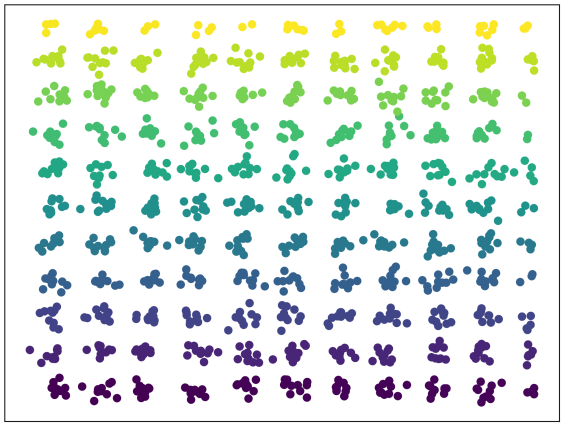
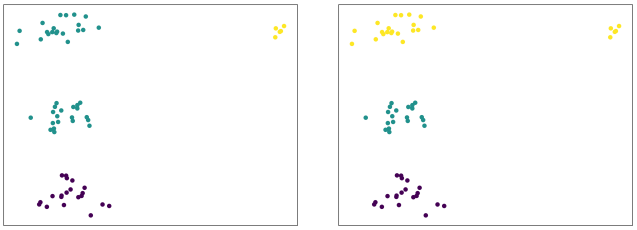


Fig. 3: Distribution of D_1.



(a) k-means (b) k-sums

Fig. 4: Performance on Outlier

TABLE 2: The performance on Grid-like datasets. (n: the number of samples, c: the number of clusters.

	Data info		FIN		k-means		k-sums	
	n	c	Time	F_1	Time	F_1	Time	F_1
D_1	100K	5K	4.07	0.895	14.40	0.892	0.49	0.992
D_2	100K	10K	4.31	0.961	24.55	0.882	0.36	0.991
D_3	100K	20K	3.57	0.872	36.87	0.867	0.31	0.986
D_4	200K	5K	7.60	0.951	34.96	0.898	1.73	0.992
D_5	200K	10K	7.44	0.893	59.71	0.891	1.03	0.992
D_6	200K	20K	7.74	0.961	96.80	0.881	1.01	0.992
D_7	300K	5K	11.73	0.954	55.16	0.900	3.66	0.992
D_8	300K	10K	11.86	0.874	98.36	0.896	2.05	0.992
D_9	300K	20K	11.72	0.940	168.03	0.888	1.83	0.991

5 Experiments

Extensive experiments on synthetic and benchmark datasets have been conducted, to validate the superiority of our algorithms, i.e., k-sums and k-sums-x. In this section, these experiments are introduced in detail.

5.1 Experiments conducted on synthetic datasets

5.1.1 Datasets

Ten synthetic datasets are used, including D_1, D_2, D_3, D_4, D_5, D_6, D_7, D_8, D_9, and Outlier. The first nine are grid-like datasets, specifically, their distribution is roughly as shown in Figure 3, and the statistics of these datasets are summarized in Table 2.

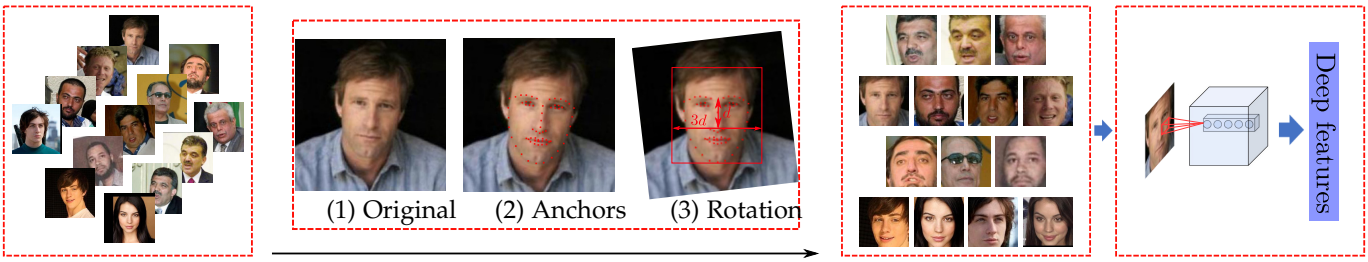
5.1.2 Experimental results

Experiments based on synthetic datasets are mainly used to verify the advantages mentioned in Section 3.8.

1. From the results in Table 2, it can be seen that the running time of k-sums (The time of constructing the k-NN graph by KD-Tree is included) is approximately linear with respect to the number of samples in the data, and is irrelevant to the number of clusters. So, for the datasets with more clusters, the advantage of our algorithm is more obvious, which verifies the first advantage.

2. It can be seen from Table 2 while having a low time complexity, k-sums shows high-quality clustering results, which verifies the second advantage.

3. As shown in Figure 4, the cluster composed of a few abnormal points heavily affected the performance of k-means but nearly has no effect on our algorithm. Therefore, our algorithm is robust to abnormal points.



(a) Dataset (b) Preprocessing (c) Cropped Dataset (d) Deep Feature

Fig. 5: Schematic diagram of face representation.

5.2 Experiments conducted on benchmark datasets

5.2.1 Datasets

Eighteen real-world datasets are involved for the verification of the clustering performance of the proposed methods (k-sums and k-sums-x). The description of them are as follows:

FaceV5[2] is a widely used dataset in face recognition, it contains $2,500$ images of 500 subjects under various illumination, pose, expression, and imaging distance.

CFPW [31] is a widely used face verification dataset, containing $7,000$ images of 500 celebrities in frontal and profile views. Each image contains 16×16 pixels.

FEI [32] dataset is a Brazilian face database. There are 14 images for each of 200 individuals (students or staff), adding up to a total number of $2,800$ images. We use a subset (700images of 50 individuals) in this paper.

Grimace [33] dataset contains 360 face images with 20 individuals. Images of individuals from various racial origins have different facial expressions on their beards and glasses.

IMM [34] dataset consisting of 240 annotated face images of 40 (7 females and 33 males) different individuals, each with 6 images. In this data, no one wears glasses.

MSRA25 [35] dataset consists of $1,799$ face gray images in 12 classes represented by 256 features, each class has 113—186 samples with different poses.

MUCT [36] consists of $3,755$ faces of 276 individuals, each image has 76 manual landmarks. This dataset provides a diversity of lighting, age, and ethnicity.

ORL [37] dataset contains 10 images of each of 40 distinct subjects. The images were taken at different times, varying the lighting, facial expressions and facial details.

YaleFace [38] contains 165 grayscale images of 15 individuals. There are 11 images per subject under different facial expressions or configurations.

Face96 [39] are the open source datasets of face recognition provided by the University of Essex. Face96 has $3,016$ images of 151 individuals. Each sample is cropped into 16×16 scale.

Isolet [40], **Mpeg7** [41], and **Palm**[3] are widely used datasets for clustering and classification. Isolet consists of 7797 samples of 26 letters represented by 617 features spoken by 150 speakers. Mpeg7 consists of 1400 shapes grouped into 70 classes. Palm contains 2000 fingerprint images generated by 100 different palms.

2. http://biometrics.idealtest.org/dbDetailForUser.do?id=9
3. http://www.escience.cn/people/chenxiaojun

TABLE 3: Benchmark datasets

	FaceV5	CFPW	FEI	Isolet	Grimace
# Samples	2500	7000	700	7797	360
# Features	256	256	147	617	256
# Subjects	500	500	50	26	18
	Mpeg7	Face96	IMM	YaleFace	MSRA25
# Samples	1400	3016	240	165	1799
# Features	728	256	256	256	256
# Subjects	70	151	40	15	12
	Palm	MUCT	ORL	PEAL	CACD
# Samples	2000	3755	400	30863	163446
# Features	256	256	256	256	256
# Subjects	100	276	40	1040	2000
	WebFace	CelebA			
# Samples	494414	202599			
# Features	256	256			
# Subjects	10575	10177			

To verify the efficiency of k-sums, four large-scale datasets are used. They are **PEAL** [42] [4], **CACD** [43], **WebFace** [44], and **CelebA** [45]. The statistics of all real-world datasets are summarized in Table 3.

5.2.2 Face representation

Because of the difference in lighting, posture, and background, it is very unwise to directly measure the Euclidean distance between two face images. Therefore, we use a pre-trained deep neural network model [46] to extract the features of the face images and then performing clustering algorithms in the feature space.

As shown in Figure 5, before feeding the pictures to the model, we use Dlib to find the location of 68 facial landmarks. Then the face image is rotated so that 20- and 25-th landmarks are located on the same horizontal line. Next, with 31-th landmark as the center, we cut out a square with a length of $3d$ (d is the vertical distance from the 31-th landmark to the 25-th landmark).

5.2.3 Baseline

To verify the effectiveness of the proposed k-sums, we compare it with five graph-based clustering methods including: Anchor-based Graph Clustering-improved algorithm

4. The subset named CAS-PEAL-R1 is used, which is publicly available.

TABLE 4: Performance of algorithms taking graph as input (mean (± std))

		AGCI	FCDMF	FINCH	RCAE	RCC	k-sums
FaceV5	ACC	0.734 (±9.0e-03)	0.546 (±1.2e-02)	0.866 (±0.0e+00)	0.834 (±2.1e-02)	0.389 (±0.0e+00)	**0.956 (±5.0e-03)**
	NMI	0.932 (±2.9e-03)	0.847 (±4.1e-03)	0.935 (±0.0e+00)	0.882 (±9.6e-03)	0.632 (±0.0e+00)	**0.986 (±1.5e-03)**
	ARI	0.626 (±2.0e-02)	0.342 (±1.4e-02)	0.236 (±0.0e+00)	0.115 (±1.6e-02)	0.029 (±0.0e+00)	**0.931 (±6.8e-03)**
	Time	16.823 (±1.2e-02)	5.530 (±7.4e-03)	4.943 (±0.0e+00)	1.7e+2 (±8.1e+1)	8.926 (±1.8e-02)	**0.295 (±4.4e-04)**
CFPW	ACC	0.541 (±9.5e-03)	0.357 (±9.1e-03)	0.665 (±0.0e+00)	0.726 (±3.3e-02)	0.749 (±0.0e+00)	**0.761 (±5.4e-03)**
	NMI	0.770 (±6.4e-03)	0.686 (±5.3e-03)	0.679 (±0.0e+00)	0.736 (±2.8e-02)	0.883 (±3.8e-04)	**0.895 (±1.8e-03)**
	ARI	0.206 (±1.7e-02)	0.158 (±9.1e-03)	0.016 (±0.0e+00)	0.029 (±8.3e-03)	0.323 (±5.8e-03)	**0.647 (±6.4e-03)**
	Time	50.718 (±1.1e-01)	17.581 (±3.1e-02)	40.038 (±0.0e+00)	1.4e+3 (±1.1e+3)	21.132 (±2.5e-01)	**2.346 (±4.0e-03)**
Face96	ACC	0.655 (±2.7e-02)	0.758 (±1.5e-02)	0.966 (±0.0e+00)	0.970 (±1.3e-02)	0.975 (±0.0e+00)	**0.992 (±3.3e-16)**
	NMI	0.875 (±1.5e-02)	0.921 (±4.6e-03)	0.979 (±0.0e+00)	0.980 (±5.6e-03)	0.989 (±0.0e+00)	**0.996 (±2.0e-16)**
	ARI	0.446 (±6.6e-02)	0.725 (±1.5e-02)	0.888 (±0.0e+00)	0.898 (±4.4e-02)	0.961 (±0.0e+00)	**0.986 (±2.2e-16)**
	Time	7.988 (±1.2e-02)	4.050 (±1.4e-02)	0.648 (±0.0e+00)	1.1e+2 (±5.2e+1)	9.095 (±6.9e-02)	**0.391 (±7.8e-04)**
FEI	ACC	0.459 (±2.1e-02)	0.420 (±1.7e-02)	0.516 (±0.0e+00)	0.518 (±6.0e-02)	0.043 (±0.0e+00)	**0.540 (±1.6e-02)**
	NMI	0.689 (±1.1e-02)	0.653 (±7.6e-03)	**0.734 (±0.0e+00)**	0.687 (±3.6e-02)	0.047 (±0.0e+00)	0.728 (±9.0e-03)
	ARI	0.337 (±1.8e-02)	0.289 (±1.4e-02)	0.363 (±0.0e+00)	0.316 (±6.4e-02)	0.003 (±0.0e+00)	**0.407 (±1.8e-02)**
	Time	0.330 (±7.9e-04)	0.215 (±6.8e-05)	0.034 (±0.0e+00)	3.452 (±1.4e+00)	1.239 (±1.6e-02)	**0.019 (±4.3e-04)**
Grimace	ACC	0.791 (±7.2e-02)	0.754 (±4.9e-02)	**1.000 (±0.0e+00)**	0.984 (±2.1e-02)	**1.000 (±0.0e+00)**	**1.000 (±0.0e+00)**
	NMI	0.908 (±4.7e-02)	0.868 (±2.2e-02)	**1.000 (±0.0e+00)**	0.988 (±1.1e-02)	**1.000 (±0.0e+00)**	**1.000 (±0.0e+00)**
	ARI	0.768 (±1.3e-01)	0.721 (±4.1e-02)	**1.000 (±0.0e+00)**	0.975 (±2.6e-02)	**1.000 (±0.0e+00)**	**1.000 (±0.0e+00)**
	Time	0.093 (±2.8e-04)	0.081 (±1.5e-04)	0.026 (±0.0e+00)	0.629 (±4.1e-01)	0.872 (±2.7e-02)	**0.004 (±8.9e-07)**
IMM	ACC	0.761 (±5.2e-02)	0.734 (±2.6e-02)	0.996 (±0.0e+00)	0.873 (±3.0e-02)	0.871 (±0.0e+00)	**1.000 (±0.0e+00)**
	NMI	0.919 (±2.4e-02)	0.887 (±1.2e-02)	0.997 (±0.0e+00)	0.915 (±2.3e-02)	0.953 (±0.0e+00)	**1.000 (±0.0e+00)**
	ARI	0.721 (±7.5e-02)	0.668 (±3.1e-02)	0.991 (±0.0e+00)	0.701 (±8.8e-02)	0.862 (±1.1e-16)	**1.000 (±0.0e+00)**
	Time	0.057 (±2.4e-04)	0.052 (±4.9e-05)	0.130 (±0.0e+00)	0.622 (±3.1e-01)	0.353 (±1.3e-02)	**0.002 (±1.4e-06)**
Isolet	ACC	0.535 (±2.5e-02)	0.260 (±1.5e-02)	0.469 (±0.0e+00)	0.555 (±1.6e-02)	0.038 (±0.0e+00)	**0.619 (±1.1e-02)**
	NMI	0.710 (±1.3e-02)	0.425 (±1.3e-02)	0.700 (±0.0e+00)	0.703 (±2.0e-02)	0.000 (±0.0e+00)	**0.733 (±5.2e-03)**
	ARI	0.476 (±2.9e-02)	0.183 (±1.4e-02)	0.415 (±0.0e+00)	0.456 (±2.7e-02)	0.000 (±0.0e+00)	**0.533 (±9.5e-03)**
	Time	23.445 (±4.8e-02)	20.949 (±2.7e-03)	**1.617 (±0.0e+00)**	1.5e+3 (±3.6e+2)	8.2e+2 (±3.6e+0)	3.370 (±9.5e-02)
Mpeg7	ACC	0.466 (±2.4e-02)	0.439 (±1.3e-02)	0.161 (±0.0e+00)	0.466 (±5.2e-02)	0.328 (±1.2e-01)	**0.541 (±7.1e-03)**
	NMI	0.662 (±1.6e-02)	0.653 (±8.1e-03)	0.218 (±0.0e+00)	0.629 (±4.3e-02)	0.667 (±3.5e-02)	**0.725 (±3.4e-03)**
	ARI	0.288 (±3.0e-02)	0.303 (±1.4e-02)	0.007 (±0.0e+00)	0.155 (±5.5e-02)	0.336 (±1.4e-01)	**0.415 (±6.1e-03)**
	Time	3.999 (±5.5e-03)	2.609 (±2.5e-04)	2.921 (±0.0e+00)	1.6e+1 (±1.0e+1)	10.938 (±1.3e-02)	**0.088 (±7.3e-04)**
MSRA25	ACC	0.520 (±4.4e-02)	0.353 (±2.2e-02)	0.444 (±0.0e+00)	0.514 (±2.2e-02)	0.139 (±0.0e+00)	**0.585 (±3.0e-02)**
	NMI	0.591 (±3.1e-02)	0.367 (±2.7e-02)	0.495 (±0.0e+00)	0.564 (±1.8e-02)	0.065 (±0.0e+00)	**0.592 (±1.8e-02)**
	ARI	0.368 (±3.8e-02)	0.200 (±3.3e-02)	0.220 (±0.0e+00)	0.328 (±3.6e-02)	0.034 (±0.0e+00)	**0.413 (±2.2e-02)**
	Time	1.978 (±3.4e-03)	1.564 (±1.5e-02)	**0.112 (±0.0e+00)**	26.848 (±8.9e+00)	34.898 (±3.5e-02)	0.167 (±8.3e-03)
MUCT	ACC	0.745 (±2.2e-02)	0.734 (±1.6e-02)	**0.986 (±0.0e+00)**	0.976 (±1.3e-02)	0.885 (±1.1e-16)	0.984 (±2.3e-03)
	NMI	0.935 (±1.1e-02)	0.918 (±4.2e-03)	**0.993 (±0.0e+00)**	0.988 (±6.3e-03)	0.963 (±0.0e+00)	**0.993 (±9.5e-04)**
	ARI	0.661 (±1.0e-01)	0.691 (±1.5e-02)	0.968 (±0.0e+00)	0.919 (±7.4e-02)	0.848 (±1.1e-16)	**0.979 (±2.5e-03)**
	Time	18.924 (±1.6e-02)	5.605 (±2.0e-02)	**0.473 (±0.0e+00)**	1.9e+2 (±1.1e+2)	10.260 (±7.5e-03)	0.632 (±8.0e-04)
ORL	ACC	0.687 (±7.8e-02)	0.758 (±3.4e-02)	**1.000 (±0.0e+00)**	0.988 (±9.9e-03)	0.998 (±1.1e-16)	**1.000 (±0.0e+00)**
	NMI	0.871 (±5.8e-02)	0.897 (±1.2e-02)	**1.000 (±0.0e+00)**	0.992 (±4.8e-03)	0.998 (±0.0e+00)	**1.000 (±0.0e+00)**
	ARI	0.578 (±1.9e-01)	0.714 (±3.6e-02)	**1.000 (±0.0e+00)**	0.977 (±1.7e-02)	0.997 (±0.0e+00)	**1.000 (±0.0e+00)**
	Time	0.134 (±3.0e-04)	0.112 (±1.4e-04)	0.031 (±0.0e+00)	0.924 (±6.1e-01)	0.852 (±1.8e-02)	**0.006 (±9.5e-07)**
Palm	ACC	0.695 (±2.1e-02)	0.593 (±1.3e-02)	0.591 (±0.0e+00)	**0.824 (±4.3e-02)**	0.020 (±0.0e+00)	0.807 (±1.9e-02)
	NMI	0.892 (±6.4e-03)	0.824 (±5.8e-03)	0.778 (±0.0e+00)	0.927 (±1.0e-02)	0.012 (±0.0e+00)	**0.931 (±6.6e-03)**
	ARI	0.649 (±2.1e-02)	0.516 (±1.3e-02)	0.148 (±0.0e+00)	0.736 (±3.8e-02)	0.000 (±0.0e+00)	**0.778 (±2.1e-02)**
	Time	3.473 (±3.3e-03)	2.092 (±1.8e-03)	0.648 (±0.0e+00)	3.1e+1 (±1.6e+1)	8.815 (±1.4e-02)	**0.167 (±8.3e-04)**
YaleFace	ACC	0.695 (±9.1e-02)	0.736 (±5.8e-02)	0.976 (±0.0e+00)	0.976 (±1.1e-16)	0.976 (±1.1e-16)	**0.994 (±3.3e-16)**
	NMI	0.811 (±6.4e-02)	0.834 (±2.4e-02)	0.968 (±0.0e+00)	0.968 (±0.0e+00)	0.968 (±0.0e+00)	**0.992 (±2.2e-16)**
	ARI	0.562 (±1.4e-01)	0.663 (±4.9e-02)	0.942 (±0.0e+00)	0.942 (±0.0e+00)	0.942 (±0.0e+00)	**0.986 (±2.2e-16)**
	Time	0.033 (±1.8e-04)	0.031 (±1.5e-04)	0.017 (±0.0e+00)	0.173 (±1.7e-01)	0.213 (±2.6e-02)	**0.001 (±1.0e-06)**

TABLE 5: Performance of algorithms taking features as input (mean (± std))

		k-means	CDKM	LKM	RKM	SPKM	k-sums-x
FaceV5	ACC	0.730 (±1.2e-02)	0.931 (±5.4e-03)	0.664 (±2.1e-01)	0.814 (±4.2e-02)	0.757 (±1.2e-02)	**0.963 (±3.0e-03)**
	NMI	0.930 (±2.6e-03)	0.978 (±1.2e-03)	0.897 (±7.5e-02)	0.947 (±1.1e-02)	0.941 (±3.8e-03)	**0.986 (±8.3e-04)**
	ARI	0.608 (±1.5e-02)	0.814 (±1.3e-02)	0.521 (±2.7e-01)	0.709 (±8.1e-02)	0.698 (±1.7e-02)	**0.915 (±6.3e-03)**
	Time	**0.084 (±2.5e-02)**	17.311 (±3.3e+00)	4.729 (±2.1e+00)	3.121 (±8.8e-01)	0.405 (±6.7e-02)	0.254 (±3.8e-02)
CFPW	ACC	0.547 (±1.0e-02)	**0.717 (±2.7e-03)**	0.480 (±2.4e-01)	0.659 (±1.6e-02)	0.623 (±1.0e-02)	0.673 (±4.8e-03)
	NMI	0.771 (±6.0e-03)	0.830 (±2.2e-03)	0.763 (±1.1e-01)	0.844 (±1.8e-02)	**0.865 (±2.8e-03)**	0.856 (±1.5e-03)
	ARI	0.208 (±1.6e-02)	0.318 (±1.2e-02)	0.289 (±1.9e-01)	0.450 (±9.4e-02)	**0.528 (±8.0e-03)**	0.518 (±4.8e-03)
	Time	**0.561 (±2.3e-01)**	1.8e+2 (±5.6e+1)	14.208 (±6.9e+00)	1.8e+1 (±3.7e+0)	1.355 (±2.0e-01)	0.944 (±8.9e-02)
Face96	ACC	0.666 (±3.0e-02)	0.949 (±6.3e-03)	0.620 (±2.1e-01)	0.898 (±7.7e-02)	0.789 (±1.5e-02)	**0.989 (±4.0e-03)**
	NMI	0.884 (±1.7e-02)	0.977 (±1.8e-03)	0.784 (±9.8e-02)	0.968 (±2.1e-02)	0.948 (±4.9e-03)	**0.993 (±1.1e-03)**
	ARI	0.477 (±7.8e-02)	0.915 (±9.6e-03)	0.581 (±2.5e-01)	0.890 (±7.6e-02)	0.796 (±1.8e-02)	**0.980 (±3.9e-03)**
	Time	**0.046 (±1.1e-02)**	6.413 (±2.8e+00)	10.960 (±1.1e+01)	1.465 (±4.6e-01)	0.218 (±4.9e-02)	0.091 (±1.8e-02)
FEI	ACC	0.465 (±2.6e-02)	0.490 (±1.8e-02)	0.413 (±2.8e-02)	0.465 (±1.6e-02)	0.460 (±1.6e-02)	**0.515 (±1.8e-02)**
	NMI	0.692 (±1.6e-02)	0.713 (±1.0e-02)	0.631 (±2.6e-02)	0.677 (±1.1e-02)	0.684 (±9.4e-03)	**0.721 (±7.5e-03)**
	ARI	0.342 (±2.8e-02)	0.372 (±1.9e-02)	0.298 (±1.9e-02)	0.329 (±1.3e-02)	0.322 (±1.4e-02)	**0.396 (±1.6e-02)**
	Time	**0.007 (±6.2e-04)**	0.567 (±1.5e-01)	1.081 (±1.2e-01)	0.100 (±1.5e-02)	0.023 (±3.4e-03)	0.048 (±5.6e-03)
Grimace	ACC	0.727 (±9.9e-02)	0.992 (±2.3e-02)	0.653 (±3.1e-01)	**1.000 (±0.0e+00)**	0.790 (±5.0e-02)	**1.000 (±0.0e+00)**
	NMI	0.871 (±7.5e-02)	0.997 (±8.0e-03)	0.710 (±3.2e-01)	**1.000 (±0.0e+00)**	0.919 (±1.9e-02)	**1.000 (±0.0e+00)**
	ARI	0.680 (±1.7e-01)	0.993 (±2.2e-02)	0.607 (±3.8e-01)	**1.000 (±0.0e+00)**	0.803 (±4.5e-02)	**1.000 (±0.0e+00)**
	Time	**0.006 (±2.4e-04)**	0.084 (±2.8e-02)	0.240 (±1.0e-01)	0.016 (±2.3e-03)	0.008 (±1.5e-03)	0.022 (±3.0e-03)
IMM	ACC	0.748 (±5.7e-02)	0.970 (±2.5e-02)	0.648 (±2.3e-01)	0.932 (±4.4e-02)	0.800 (±5.0e-02)	**1.000 (±0.0e+00)**
	NMI	0.914 (±2.0e-02)	0.991 (±7.8e-03)	0.790 (±1.4e-01)	0.979 (±1.4e-02)	0.937 (±1.6e-02)	**1.000 (±0.0e+00)**
	ARI	0.703 (±7.0e-02)	0.966 (±2.8e-02)	0.572 (±3.3e-01)	0.927 (±4.7e-02)	0.792 (±4.8e-02)	**1.000 (±0.0e+00)**
	Time	**0.006 (±2.9e-04)**	0.100 (±2.2e-02)	0.169 (±3.4e-02)	0.020 (±3.2e-03)	**0.006 (±1.3e-03)**	0.026 (±3.2e-03)
Isolet	ACC	0.529 (±3.8e-02)	0.536 (±3.0e-02)	0.450 (±1.2e-01)	0.592 (±3.5e-02)	0.546 (±4.8e-02)	**0.610 (±1.3e-02)**
	NMI	0.709 (±1.5e-02)	0.711 (±1.0e-02)	0.594 (±1.3e-01)	0.720 (±1.1e-02)	0.711 (±1.8e-02)	**0.731 (±7.4e-03)**
	ARI	0.468 (±3.7e-02)	0.471 (±2.5e-02)	0.355 (±1.4e-01)	0.506 (±3.0e-02)	0.475 (±4.0e-02)	**0.524 (±1.3e-02)**
	Time	**0.196 (±5.1e-02)**	31.072 (±8.1e+00)	1.851 (±7.4e-01)	1.603 (±7.4e-01)	0.746 (±2.1e-01)	0.349 (±6.5e-02)
Mpeg7	ACC	0.460 (±1.8e-02)	0.517 (±1.6e-02)	0.276 (±1.9e-01)	0.516 (±2.4e-02)	0.480 (±1.5e-02)	**0.551 (±9.9e-03)**
	NMI	0.661 (±1.6e-02)	0.721 (±7.9e-03)	0.500 (±1.7e-01)	0.709 (±1.5e-02)	0.689 (±7.6e-03)	**0.738 (±5.5e-03)**
	ARI	0.278 (±3.1e-02)	0.364 (±2.1e-02)	0.156 (±1.7e-01)	0.383 (±2.8e-02)	0.342 (±1.7e-02)	**0.434 (±9.4e-03)**
	Time	**0.033 (±5.1e-03)**	7.875 (±2.0e+00)	1.338 (±9.7e-01)	0.267 (±1.4e-02)	0.214 (±3.8e-02)	0.120 (±1.3e-02)
MSRA25	ACC	0.522 (±5.9e-02)	0.499 (±3.2e-02)	0.404 (±1.1e-01)	0.517 (±4.1e-02)	0.474 (±3.1e-02)	**0.530 (±3.4e-02)**
	NMI	0.584 (±4.3e-02)	0.577 (±2.4e-02)	0.430 (±1.6e-01)	0.574 (±3.1e-02)	0.553 (±3.9e-02)	**0.600 (±2.7e-02)**
	ARI	0.345 (±5.5e-02)	0.344 (±2.9e-02)	0.250 (±1.1e-01)	0.360 (±3.2e-02)	0.323 (±5.5e-02)	**0.392 (±3.2e-02)**
	Time	**0.019 (±3.5e-03)**	0.694 (±1.3e-01)	0.135 (±9.3e-02)	0.179 (±4.8e-02)	0.037 (±1.1e-02)	0.034 (±6.3e-03)
MUCT	ACC	0.741 (±2.5e-02)	0.970 (±5.3e-03)	0.746 (±9.3e-02)	0.875 (±5.3e-02)	0.791 (±2.2e-02)	**0.975 (±4.1e-03)**
	NMI	0.935 (±1.4e-02)	0.990 (±1.1e-03)	0.868 (±6.5e-02)	0.963 (±1.1e-02)	0.954 (±5.1e-03)	**0.991 (±1.4e-03)**
	ARI	0.672 (±1.2e-01)	0.956 (±5.6e-03)	0.717 (±9.9e-02)	0.860 (±4.5e-02)	0.794 (±2.3e-02)	**0.969 (±4.6e-03)**
	Time	**0.074 (±1.4e-02)**	11.546 (±2.3e+00)	6.550 (±4.2e+00)	2.204 (±1.9e-01)	0.323 (±3.7e-02)	0.177 (±1.6e-02)
ORL	ACC	0.708 (±6.3e-02)	0.969 (±2.4e-02)	0.564 (±2.6e-01)	0.948 (±4.5e-02)	0.792 (±3.3e-02)	**0.998 (±7.5e-03)**
	NMI	0.887 (±4.3e-02)	0.992 (±6.1e-03)	0.696 (±1.8e-01)	0.984 (±1.4e-02)	0.935 (±1.1e-02)	**0.999 (±2.8e-03)**
	ARI	0.624 (±1.4e-01)	0.971 (±2.2e-02)	0.517 (±3.3e-01)	0.947 (±4.6e-02)	0.795 (±3.3e-02)	**0.997 (±8.5e-03)**
	Time	**0.006 (±3.1e-04)**	0.151 (±3.2e-02)	0.267 (±1.4e-01)	0.029 (±2.4e-03)	0.011 (±1.9e-03)	0.050 (±5.6e-03)
Palm	ACC	0.685 (±2.4e-02)	0.750 (±1.9e-02)	0.569 (±1.9e-01)	0.703 (±4.5e-02)	0.691 (±1.9e-02)	**0.799 (±1.8e-02)**
	NMI	0.890 (±8.4e-03)	0.916 (±5.2e-03)	0.770 (±1.6e-01)	0.889 (±1.9e-02)	0.891 (±7.8e-03)	**0.925 (±5.7e-03)**
	ARI	0.641 (±2.6e-02)	0.711 (±1.9e-02)	0.506 (±2.1e-01)	0.658 (±5.5e-02)	0.643 (±2.2e-02)	**0.766 (±1.8e-02)**
	Time	**0.023 (±3.3e-03)**	3.332 (±7.4e-01)	2.856 (±1.7e+00)	0.532 (±7.8e-02)	0.177 (±3.8e-02)	0.070 (±1.6e-02)
YaleFace	ACC	0.767 (±9.4e-02)	0.968 (±2.4e-02)	0.649 (±2.7e-01)	0.944 (±5.2e-02)	0.833 (±4.6e-02)	**0.988 (±0.0e+00)**
	NMI	0.864 (±5.6e-02)	0.964 (±1.2e-02)	0.700 (±2.6e-01)	0.963 (±2.8e-02)	0.925 (±2.1e-02)	**0.984 (±2.2e-16)**
	ARI	0.700 (±1.3e-01)	0.932 (±3.1e-02)	0.567 (±3.5e-01)	0.922 (±6.1e-02)	0.824 (±4.5e-02)	**0.972 (±1.1e-16)**
	Time	**0.005 (±1.8e-04)**	0.032 (±5.6e-03)	0.130 (±9.3e-02)	0.006 (±2.7e-03)	**0.005 (±2.3e-03)**	0.017 (±2.6e-03)

TABLE 6: Performance on large-scale datasets

Data	Met.	FINCH	k-sums	k-means	k-sums-x
CACD	ACC	0.745	**0.832**	0.68	0.819
	NMI	0.79	**0.903**	0.858	0.897
	ARI	0.021	**0.755**	0.461	0.736
	Time	2217.65	**20.119**	114.032	114.449
WebFace	ACC	-	**0.565**	0.529	0.489
	NMI	-	**0.847**	0.836	0.83
	ARI	-	**0.391**	0.329	0.334
	Time	-	**75.114**	1841.865	4232.793
CelebA	ACC	-	**0.467**	0.365	0.44
	NMI	-	**0.802**	0.52	0.555
	ARI	-	**0.298**	0	0
	Time	-	**7.476**	301.742	828.339
PEAL	ACC	**0.884**	0.877	0.696	0.849
	NMI	0.906	**0.942**	0.894	0.929
	ARI	0.125	**0.806**	0.587	0.747
	Time	75.844	**1.401**	3.112	9.706

- Out of memory

(AGCI) [47], Fast clustering with Co-clustering via Discrete non-negative Matrix Factorization (FCDMF) [48], First Integer Neighbor Clustering Hierarchy (FINCH) [49], Rank-Constrained clustering with Adaptive Embedding (RCAE), and Robust continuous clustering (RCC) [50].

We verify the superiority of k-sums-x by comparing it with five k-means based algorithms, which are: k-means (KM) [51], Coordinate Descent Method for k-means(CDKM) [52], lasso k-means (LKM) [53], regularized k-means (RKM) [54], and spherical k-means (SPKM) [55].

5.2.4 Settings

The parameters involved in these algorithms are described here. In all methods that required an anchor graph as input, i.e., AGCI and FCDMF, the anchors are always generated by k-means (the cluster center is regarded as an anchor point, in the final clustering result). The number of anchors, m, is set as follows:

$$m = \min(n/2, 1024) \tag{63}$$

For these methods that require a k-NN graph or anchor graph, heat-kernel is used to construct the graph, and the number of neighbours, k, is determined by $k = \lfloor n/c * 1.2 \rfloor$. In RCAE, the regularization parameter γ is tuned from $\{0.01, 0.1, 1, 10, 100\}$. the clustering threshold in RCC is tuned from $\{0.1, 0.3, 0.5, 0.7, 0.9\}$. For a fair comparison, the number of neighbours, k, is also determined by $k = \lfloor n/c * 1.2 \rfloor$, in RCC and k-sums.

The methods shown in Table 5 are all initialized in a random way. The regularization parameter λ is tuned from $\{10e-6, 10e-4, 10e-2, 10e0, 10e2, 10e4, 10e6\}$. RKM used here is the version with the strict balance constraint. Algorithms whose performance is affected by initialization were run 50 times, and average performance was reported.

5.2.5 Experimental results

From the results shown in Table 4, 5 and Figure 6, it is not difficult to find that:

1. First of all, it can be clearly seen that our algorithms have achieved the highest performance on most datasets showing the effectiveness and robustness of them. Take k-sums as an example, it achieves 32.4%, 30.5%, 7.9%, 5.7%, 4.5%, and 4.4% gain on CFPW, FaceV5, Mpeg7, Isolet, MSRA25, and YaleFace over the second-best method respectively, in terms of ARI score. Similar conclusions can be obtained with the metrics of recall and precision.
2. In Table 4, the time consumed by constructing the k-NN graph is also included. Here, we construct it by brute force since the size of these datasets is small. As can be seen, on medium-scale data sets, k-sums shows a speed advantage compared to other graph-based algorithms, but it is not as efficient as the k-means. This is why we do experiments on large-scale datasets.
3. Algorithms that require a hyperparameter have poor performance on some data, which may be caused by inappropriate parameters. k-sums algorithm requires only one parameter, k, which makes the algorithm easier to use. Moreover, it has satisfactory performance, even if the parameter is not tuned, as shown in Table 4.
4. For graph-based algorithms, their performance usually is effected by the value of k. However, from Table 4, we can see that k-sums has achieved satisfactory performance on Isolet ($n/c \approx 300$), MSRA25($n/c \approx 150$), CFPW($n/c \approx 14$) and YaleFace ($n/c \approx 11$), which shows that the formula used to determine the value of k is reasonable. The effect of parameter k on performance has also been studied, and the results are shown in Figure 6.
5. Again, the time of construction the k-NN graph is included in Table 6. As we have seen, in terms of efficiency, k-sums outperforms k-means on large-scale datasets. On large-scale datasets, most graph-based algorithms (AGCI, FCDMF, RCAE and RCC) cannot produce results because of their high computational complexity or high memory overhead. EFANNA [56] is the algorithm that is used to generate the k-nearest neighbors graph.

From the results shown in Table 4, 5, and 6, in most benchmarks, k-sums achieves higher performance over k-sums-x. It seems that k-sums-x is not very necessary. Therefore, we think it is necessary to list some scenarios where k-sums-x is applicable.

- In many practical scenarios, the cluster centers are also very important information. In other words, we sometimes need meaningful cluster centers instead of a partition [57], [58]. In k-sums-x, the means of each cluster can be used as cluster centers. However, in k-sums, the means of each cluster are meaningless.
- If some new training samples are obtained, it is difficult for k-sums to update the model, which is also a disadvantage of most graph-based algorithms. Usually, it will be retrained on whole datasets. But it is easy for k-sums-x to update the model.

In summary, k-sums should be used if a partition is all you need, k-sums-x should be used if clustering centers are needed or you need to update the model often.

6 Conclusion

Based on the unified framework of k-means and graph cut models, we proposed a novel clustering algorithm,

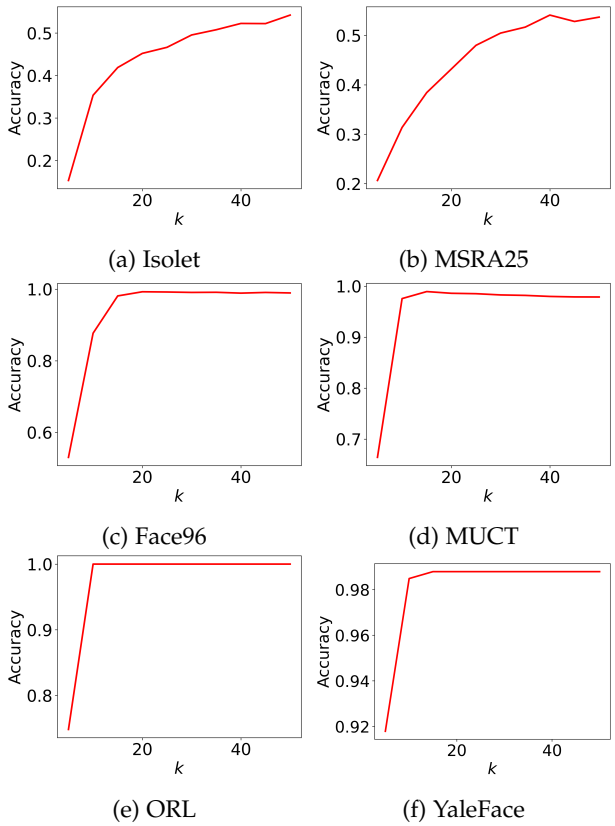


Fig. 6: Performance with different k

called k-sums. The exciting thing is that our model can be efficiently optimized if the standard coordinate descent algorithm is adopted, and its performance is comparable to most graph-based clustering algorithms. Specifically, both computational and memory overhead are $O(nk)$ that is irrelevant to the product of the number of samples and subjects.

To deal with the situation in which the k-NN graph is not available, we proposed a variant of k-sums, called k-sums-x. It is a k-means-like algorithm, and only takes features and the number of clusters as input. By the optimization method based on coordinate descent proposed in this paper, k-sums-x can also be solved very efficiently. The computational complexity of it is the same as that of k-means.

Extensive experiments have been conducted on 10 synthetic datasets, 13 middle-scale benchmark datasets, and 4 large-scale datasets. to verify the effectiveness and efficiency of the proposed models, i.e., k-sums and k-sums-x, and the properties of them are also analysed.

ACKNOWLEDGMENTS

This work was supported in part by the National Key Research and Development Program of China under Grant 2018AAA0101902, in part by the Natural Science Basic Research Program of Shaanxi (Program No. 2021JM-071), in part by the National Natural Science Foundation of China under Grant 62176212, Grant 61936014 and Grant 61772427, and in part by the Fundamental Research Funds for the Central Universities under Grant G2019KY0501.

REFERENCES

[1] M. Tapaswi, M. T. Law, and S. Fidler, "Video face clustering with unknown number of clusters," in *Proceedings of the IEEE/CVF International Conference on Computer Vision*, 2019, pp. 5027–5036.
[2] Y. Shi, C. Otto, and A. K. Jain, "Face clustering: representation and pairwise constraints," *IEEE Transactions on Information Forensics and Security*, vol. 13, no. 7, pp. 1626–1640, 2018.
[3] R. Janani and S. Vijayarani, "Text document clustering using spectral clustering algorithm with particle swarm optimization," *Expert Systems with Applications*, vol. 134, pp. 192–200, 2019.
[4] L. Krawczyk, "Comparing applicability of prevalent clustering algorithms for document clustering," B.S. thesis, Humboldt-Universität zu Berlin, 2019.
[5] H.-W. Lee, N. Malik, F. Shi, and P. Mucha, "Social clustering in epidemic spread on coevolving networks," *Physical Review E*, vol. 99, no. 6, p. 062301, 2019.
[6] M. Xu, Y. Li, R. Li, F. Zou, and X. Gu, "Eadp: An extended adaptive density peaks clustering for overlapping community detection in social networks," *Neurocomputing*, vol. 337, pp. 287–302, 2019.
[7] R. Jain and R. Sharma, "Image segmentation through fuzzy clustering: A survey," in *Harmony Search and Nature Inspired Optimization Algorithms*, 2019, pp. 497–508.
[8] T. Lei, X. Jia, Y. Zhang, S. Liu, H. Meng, and A. K. Nandi, "Superpixel-based fast fuzzy c-means clustering for color image segmentation," *IEEE Transactions on Fuzzy Systems*, vol. 27, no. 9, pp. 1753–1766, 2018.
[9] A. Ng, M. Jordan, and Y. Weiss, "On spectral clustering: Analysis and an algorithm," in *Advances in neural information processing systems*, 2002, pp. 849–856.
[10] J. Shi and J. Malik, "Normalized cuts and image segmentation," *IEEE Transactions on pattern analysis and machine intelligence*, vol. 22, no. 8, pp. 888–905, 2000.
[11] C. Fowlkes, S. Belongie, F. Chung, and J. Malik, "Spectral grouping using the nystrom method," *IEEE transactions on pattern analysis and machine intelligence*, vol. 26, no. 2, pp. 214–225, 2004.
[12] A. Choromanska, T. Jebara, H. Kim, M. Mohan, and C. Monteleoni, "Fast spectral clustering via the nyström method," in *Algorithmic Learning Theory*, 2013, pp. 367–381.
[13] G. Zhong and C.-M. Pun, "Revisiting nyström extension for hypergraph clustering," *Neurocomputing*, vol. 403, pp. 247–256, 2020.
[14] M. Li, J. Kwok, and B. Lü, "Making large-scale nyström approximation possible," in *ICML 2010-Proceedings, 27th International Conference on Machine Learning*, 2010, p. 631.
[15] T. Sakai and A. Imiya, "Fast spectral clustering with random projection and sampling," in *Machine Learning and Data Mining in Pattern Recognition*, 2009, pp. 372–384.
[16] X. Yang, W. Yu, R. Wang, G. Zhang, and F. Nie, "Fast spectral clustering learning with hierarchical bipartite graph for large-scale data," *Pattern Recognition Letters*, vol. 130, pp. 345 – 352, 2020.
[17] Y. Li, J. Huang, and W. Liu, "Scalable sequential spectral clustering," in *Proceedings of the Thirtieth AAAI Conference on Artificial Intelligence*, 2016, p. 1809–1815.
[18] A. Hassanzadeh, A. Kaarna, and T. Kauranne, "Sequential spectral clustering of hyperspectral remote sensing image over bipartite graph," *Applied Soft Computing*, vol. 73, pp. 727 – 734, 2018.
[19] B. Schölkopf, A. Smola, and K.-R. Müller, "Nonlinear component analysis as a kernel eigenvalue problem," *Neural computation*, vol. 10, pp. 1299–1319, 1998.
[20] D. Kim, K. Lee, D. Lee, and K. Lee, "Evaluation of the performance of clustering algorithms in kernel-induced feature space," *Pattern Recognition*, vol. 38, pp. 607–611, 2005.
[21] R. Chitta, R. Jin, T. Havens, and A. Jain, "Approximate kernel k-means: Solution to large scale kernel clustering," in *Proceedings of the 17th ACM SIGKDD international conference on Knowledge discovery and data mining*, 2011, pp. 895–903.
[22] D. Arthur and S. Vassilvitskii, "k-means++: The advantages of careful seeding," Stanford, Tech. Rep., 2006.
[23] O. Bachem, M. Lucic, S. Hassani, and A. Krause, "Approximate k-means++ in sublinear time," in *Proceedings of the Thirtieth AAAI Conference on Artificial Intelligence*, 2016, pp. 1459–1467.
[24] O. Bachem, M. Lucic, H. Hassani, and A. Krause, "Fast and provably good seedings for k-means," in *Advances in Neural Information Processing Systems 29*, D. D. Lee, M. Sugiyama, U. V. Luxburg, I. Guyon, and R. Garnett, Eds., 2016, pp. 55–63.
[25] I. Dhillon, Y. Guan, and B. Kulis, "Kernel k-means: spectral clustering and normalized cuts," in *Proceedings of the tenth ACM SIGKDD*

international conference on Knowledge discovery and data mining, 2004, pp. 551–556.
[26] I. S. Dhillon, Y. Guan, and B. Kulis, *A unified view of kernel k-means, spectral clustering and graph cuts*. Citeseer, 2004.
[27] C. Ding, X. He, and H. Simon, "On the equivalence of nonnegative matrix factorization and spectral clustering," in *Proceedings of the 2005 SIAM international conference on data mining*, 2005, pp. 606–610.
[28] R. Zass and A. Shashua, "Doubly stochastic normalization for spectral clustering," in *Advances in Neural Information Processing Systems*, B. Schölkopf, J. Platt, and T. Hoffman, Eds., vol. 19, 2007.
[29] X. Wang, F. Nie, and H. Huang, "Structured doubly stochastic matrix for graph based clustering: Structured doubly stochastic matrix," in *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, 2016, p. 1245–1254.
[30] P. E. Hart, D. G. Stork, and R. O. Duda, *Pattern classification*. Wiley Hoboken, 2000.
[31] C. C. V. P. R. C. D. J. S. Sengupta, J.C. Cheng, "Frontal to profile face verification in the wild," in *Proc. WACV*, 2016, pp. 1–9.
[32] C. E. Thomaz and G. A. Giraldi, "A new ranking method for principal components analysis and its application to face image analysis," *Image Vision Comput.*, vol. 28, no. 6, pp. 902–913, 2010.
[33] U. A. Kamerikar and M. Chavan, "Experimental assessment of lda and klda for face recognition," *International Journal*, vol. 2, no. 2, pp. 137–146, 2014.
[34] M. Nordstrom, M. Larsen, J. Sierakowski, and M. Stegmann, *The IMM Face Database - An Annotated Dataset of 240 Face Images*. Technical University of Denmark, DTU Informatics, 2004.
[35] L. Zhang, Q. Zhang, B. Du, J. You, and D. Tao, "Adaptive manifold regularized matrix factorization for data clustering." in *Proc. IJCAI*, 2017, pp. 3399–3405.
[36] S. Milborrow, J. Morkel, and F. Nicolls, "The muct landmarked face database," *Pattern Recognition Association of South Africa*, pp. 32–34, 2010.
[37] D. Cai, X. He, Y. Hu, J. Han, and T. Huang, "Learning a spatially smooth subspace for face recognition," in *Proc. CVPR*, 2007, pp. 1–7.
[38] A. S. Georghiades, P. N. Belhumeur, and D. J. Kriegman, "From few to many: illumination cone models for face recognition under variable lighting and pose," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 23, no. 6, pp. 643–660, 2001.
[39] D. Virmani, P. Girdhar, P. Jain, and P. Bamdev, "Fdrenet: Face detection and recognition pipeline," *Engineering, Technology and Applied Science Research*, vol. 9, no. 2, pp. 3933–3938, 2019.
[40] M. Fanty and R. Cole, "Spoken letter recognition," *Proc. NIPS*, vol. 3, pp. 220–226, 1990.
[41] L. J. Latecki and R. Lakamper, "Shape similarity measure based on correspondence of visual parts," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 22, no. 10, pp. 1185–1190, 2000.
[42] W. Gao, B. Cao, S. Shan, X. Chen, D. Zhou, X. Zhang, and D. Zhao, "The cas-peal large-scale chinese face database and baseline evaluations," *IEEE Transactions on Systems, Man, and Cybernetics - Part A: Systems and Humans*, vol. 38, no. 1, pp. 149–161, 2008.
[43] B.-C. Chen, C.-S. Chen, and W. H. Hsu, "Face recognition and retrieval using cross-age reference coding with cross-age celebrity dataset," *IEEE Transactions on Multimedia*, vol. 17, no. 6, pp. 804–815, 2015.
[44] D. Yi, Z. Lei, S. Liao, and S. Z. Li, "Learning face representation from scratch," *arXiv preprint arXiv:1411.7923*, 2014.
[45] Z. Liu, P. Luo, X. Wang, and X. Tang, "Deep learning face attributes in the wild," in *Proceedings of International Conference on Computer Vision (ICCV)*, 2015, pp. 3730–3738.
[46] X. Wu, R. He, Z. Sun, and T. Tan, "A light cnn for deep face representation with noisy labels," *IEEE Transactions on Information Forensics and Security*, vol. 13, no. 11, pp. 2884–2896, 2018.
[47] Y. Zhao, Y. Yuan, and Q. Wang, "Fast spectral clustering for unsupervised hyperspectral image classification," *Remote Sensing*, vol. 11, no. 4, p. 399, 2019.
[48] F. Nie, S. Pei, R. Wang, and X. Li, "Fast clustering with co-clustering via discrete non-negative matrix factorization for image identification," in *Proc. ICASSP*, 2020, pp. 2073–2077.
[49] V. S. M. Saquib Sarfraz and R. Stiefelhagen, "Efficient parameter-free clustering using first neighbor relations," in *Proc. CVPR*, 2019, pp. 8934–8943.
[50] S. A. Shah and V. Koltun, "Robust continuous clustering," *Proceedings of the National Academy of Sciences*, vol. 114, no. 37, pp. 9814–9819, 2017.
[51] J. A. Hartigan and M. A. Wong, "Algorithm as 136: A k-means clustering algorithm," *Journal of the Royal Statistical Society*, vol. 28, no. 1, pp. 100–108, 1979.
[52] F. Nie, J. Xue, D. Wu, R. Wang, H. Li, and X. Li, "Coordinate descent method for k-means," *IEEE Trans. Pattern Anal. Mach. Intell.*, 2021.
[53] Z. Li, F. Nie, X. Chang, Z. Ma, and Y. Yang, "Balanced clustering via exclusive lasso: A pragmatic approach," in *Proc. AAAI*, 2018, pp. 3596–3603.
[54] W. Lin, Z. He, and M. Xiao, "Balanced clustering: A uniform model and fast algorithm," in *Proc. IJCAI*, 7 2019, pp. 2987–2993.
[55] C. Buchta, M. Kober, I. Feinerer, and K. Hornik, "Spherical k-means clustering," *Journal of statistical software*, vol. 50, no. 10, pp. 1–22, 2012.
[56] C. Fu and D. Cai, "Efanna: An extremely fast approximate nearest neighbor search algorithm based on knn graph," *arXiv preprint arXiv:1609.07228*, 2016.
[57] Y.-N. Zhu and Y.-F. Li, "Semi-supervised streaming learning with emerging new labels," in *Proceedings of the AAAI Conference on Artificial Intelligence*, vol. 34, no. 04, 2020, pp. 7015–7022.
[58] H. Jegou, M. Douze, and C. Schmid, "Product quantization for nearest neighbor search," *IEEE transactions on pattern analysis and machine intelligence*, vol. 33, no. 1, pp. 117–128, 2010.

Shenfei Pei is now working toward his PhD degree in Northwestern Polytechnical University, Xi'an, China. His research interests include machine learning and its application fields, such as data mining, computer vision and Clustering.

Huimin Chen Huimin Chen received the bachelor's degree in software engineering from Northwestern Polytechnical University, Xi'an, China, in 2019. Her current research interests in machine learning and its applications, including clustering and dimensionality reduction.

Feiping Nie received the Ph.D. degree in Computer Science from Tsinghua University, China in 2009, and currently is full professor in Northwestern Polytechnical University, China. His research interests are machine learning and its applications, such as pattern recognition, data mining, computer vision, image processing and information retrieval. He has published more than 100 papers in the following journals and conferences: TPAMI, IJCV, TIP, TNNLS, TKDE, ICML, NIPS, KDD, IJCAI, AAAI, ICCV, CVPR, ACM MM. His papers have been cited more than 20000 times and the H-index is 78. He is now serving as Associate Editor or PC member for several prestigious journals and conferences in the related fields.

Rong Wang received the B.S. degree in information engineering, the M.S. degree in signal and information processing, and the Ph.D. degree in computer science from Xi'an Research Institute of Hi-Tech, Xi'an, China, in 2004, 2007 and 2013, respectively. During 2007 and 2013, he also studied in the Department of Automation, Tsinghua University, Beijing, China for his Ph.D. degree. He is currently an associate professor at the School of Cybersecurity and School of Artificial Intelligence, Optics and Electronics (iOPEN), Northwestern Polytechnical University, Xi'an, China. His research interests focus on machine learning and its applications.

Xuelong Li (Fellow, IEEE) is a Full Professor with the school of Artificial Intelligence, Optics and Electronics (iOPEN), Northwestern Polytechnical University, Xi'an, China.