# Adaptive Consensus Clustering for Multiple K-Means Via Base Results Refining

Peng Zhou, Liang Du, and Xuejun Li

***Abstract*—Consensus clustering, which learns a consensus clustering result from multiple weak base results, has been widely studied. However, conventional consensus clustering methods only focus on the ensemble process while ignoring the quality improvement of the base results, and thus they just use the fixed base results for consensus learning. In this paper, we provide an alternative idea to improve the final consensus clustering performance by considering the base results refining. In our framework, we adaptively refine the base results in the process of the ensemble. In more detail, on one hand, we ensemble multiple K-means results to learn the consensus one by considering the consensus and diversity; on the other hand, we apply the consensus result to design a graph filter to learn a more cluster-friendly embedding for refining the base K-means results. In our framework, the consensus learning and base results refining are integrated into one unified objective function so that these two tasks can be boosted by each other. Then we design an effective iterative algorithm to optimize the carefully designed objective function. The extensive experiments on benchmark data sets demonstrate that the proposed method can outperform both the single clustering and the state-of-the-art consensus clustering methods. The codes of this paper are released in http://Doctor-Nobody.github.io/codes/ACMK.zip.**

***Index Terms*—Consensus clustering, graph filter, multiple k-means.**

## I. Introduction

CLUSTERING is a fundamental unsupervised learning problem and is widely studied in recent decades. Among various clustering methods, K-means is one of the most famous methods because of its easiness and effectiveness. However, K-means also has some limitations on its stableness and robustness [1], [2]. For example, K-means is sensitive to its initialization and different initializations often lead to very different clustering results. Moreover, it can only handle sphere-shaped data and often fails to partition data on complex manifolds.

To overcome these limitations, consensus clustering is proposed [3]. Consensus clustering first runs K-means multiple times to obtain multiple base clustering results. Although these base results may be imperfect and unreliable, consensus clustering then ensembles them to learn a consensus clustering result which is more stable and robust than the base ones. Since consensus clustering can often improve the clustering performance, it has attracted much attention [4], [5], [6], [7], [8], [9], [10]. Despite the benefits of these methods, we observe that all these methods try to ensemble multiple *fixed* pre-given base results. However, as introduced before, these base results are imperfect and unreliable, and thus they may also mislead the ensemble process.

Different from conventional consensus clustering methods, which only apply the ensemble method to improve the clustering performance, in this paper, we provide an alternative idea to achieve this. Since the base results are unreliable, why not try to improve the base results in the process of the ensemble? To fulfill this idea, we propose a novel *Adaptive Consensus Multiple K-means* (ACMK) method. In this method, instead of integrating fixed base results, we adaptively update them in the process of consensus learning. The benefits are twofold: on one hand, by consensus learning, we can obtain a more stable and robust result from base ones; and on the other hand, the consensus result can further guide us to refine the base results in turn. Therefore, the consensus learning and base results refining can be boosted by each other.

In the process of consensus learning, we provide an ensemble method to consider both the consensus and diversity properties. In the base results refining, with the consensus result, we apply the graph filter method to learn a more cluster-friendly embedding of the original data, which meets the manifold assumption of clustering, and then run the multiple K-means on this embedding to obtain multiple new base results. Since the new embedding is more cluster-friendly than the embedding of the previous iterations, the base results can also be increasingly more reliable. Different from conventional graph filter methods such as [11], [12], which use a fixed filter, our method designs a *learnable* graph filter, i.e., the graph structure of the filter is also automatically updated in the process of consensus learning, which can be more flexible to handle some complex data. We carefully design an objective function to integrate the two tasks (i.e., consensus learning and base results refining) into a unified framework seamlessly. Then, we provide an effective

Manuscript received 28 February 2022; revised 7 December 2022; accepted 1 April 2023. Date of publication 6 April 2023; date of current version 15 September 2023. This work was supported by the National Natural Science Foundation of China under Grants 62176001, 61806003, 61976129, and 61972001. Recommended for acceptance by P. Bogdanov. *(Corresponding author: Peng Zhou.)*

Peng Zhou is with the Anhui Provincial International Joint Research Center for Advanced Technology in Medical Imaging, School of Computer Science and Technology, Anhui University, Hefei, Anhui 230601, China, and also with the State Key Laboratory of Computer Science, Institute of Software, Chinese Academy of Sciences, Beijing 100190, China (e-mail: zhoupeng@ahu.edu.cn).

Xuejun Li is with the Anhui Provincial International Joint Research Center for Advanced Technology in Medical Imaging, School of Computer Science and Technology, Anhui University, Hefei 230601, China (e-mail: xjli@ahu.edu.cn).

Liang Du is with the School of Computer and Information Technology, Shanxi University, Taiyuan, Shanxi 030006, China (e-mail: duliang@sxu.edu.cn).

This article has supplementary downloadable material available at https://doi.org/10.1109/TKDE.2023.3264970, provided by the authors.

Digital Object Identifier 10.1109/TKDE.2023.3264970

ADMM [13] method to iteratively optimize the introduced objective function. Finally, we conduct extensive experiments by comparing it with the state-of-the-art consensus clustering methods on benchmark data sets. The experimental results demonstrate the effectiveness and superiority of the proposed method.

The main contributions of this paper are summarized as follows:

- We propose a novel consensus clustering method. Different from conventional methods which just ensemble fixed base results, the proposed one integrates the base results refining into the consensus learning, which can improve the quality of the base results and further improve the final clustering performance.
- Different from the conventional graph filter methods, which use a pre-given and fixed graph filter, we design a learnable graph filter to learn a cluster-friendly embedding which can guide the base results refining for consensus learning. In our framework, the consensus learning approach can in turn guide us to learning a more appropriate graph filter.
- The extensive experiments show that the proposed method not only improves the single clustering performance but also achieves better performance compared with the state-of-the-art consensus clustering methods.

## II. Related Work and Preliminaries

In this section, we briefly introduce some related work and preliminaries. First, we introduce some notations. We use bold uppercase and lowercase letters to denote matrices and vectors, respectively. $M_{ij}$ denotes the $(i,j)$-th element of matrix $\mathbf{M}$ and $v_i$ denotes the $i$-th element of vector $\mathbf{v}$.

### A. K-Means

K-means [14] is one of the most famous clustering methods and has been widely studied. Given a data set $\mathbf{X} \in \mathbb{R}^{n\times d}$ with $n$ instances and $d$ features, supposing that we want to partition them into $c$ clusters, K-means uses the following formula to generate the clustering result:

$$\begin{aligned} &\min_{\mathbf{G},\mathbf{F}} \quad \|\mathbf{X}-\mathbf{G}\mathbf{F}\|_F^2, \\ &s.t. \quad \mathbf{G}\in\{0,1\}^{n\times c}, \quad \forall i: \sum_{j=1}^{c} G_{ij}=1, \end{aligned} \tag{1}$$

where $\mathbf{G} \in \mathbb{R}^{n\times c}$ is a cluster indicator matrix, and $\mathbf{F} \in \mathbb{R}^{c\times d}$ contains the $c$ cluster centroids. Each row of $\mathbf{G}$ contains only one 1 and other elements are 0 s. $G_{ij} = 1$ means that the $i$-th instance belongs to the $j$-th cluster.

Since (1) is non-convex, it is hard to find the global optima. Traditional K-means can just find the local optima. That is why we often get different clustering results if we use different initializations. Therefore, K-means suffers from the stable problem [15]. Moreover, since K-means applies the Euclidean distance to measure the distance between the instance and the cluster center, it is designed to handle only the sphere-shaped data. If we use it to handle some other data on some complex manifolds, it may lead to poor clustering results [16].

### B. Consensus Clustering

Consensus clustering [3], also known as clustering ensemble, learns a consensus clustering result from multiple base clusterings. Since consensus clustering can provide a more stable and robust clustering result, it attracts much attention in recent years. One related work of consensus clustering is multi-view clustering. Multi-view clustering takes the multiple groups of features (a.k.a. views) of data as inputs and integrates the multiple features to obtain a consensus result. For example, Wang et al. characterized the local manifold structure of each view and proposed a multi-view spectral clustering to preserve such structures to obtain the consensus result [17], [18]; Zhou et al. ensembled multiple views incrementally for multi-view spectral clustering [19]; Liu et al. tried to ensemble multiple incomplete views of data and proposed an effective and efficient incomplete multi-view clustering method [20]; Wu et al. applied the deep multi-view learning to unsupervised person re-identification [21]; most recently, Wu et al. adopted recursive Hermite polynomial networks to extract multi-scale features of data for representation learning [22]. Different from multi-view learning whose inputs are multiple groups of features, consensus clustering often takes one group of features as input and applies one or multiple base clustering methods to generate multiple base clustering results for ensemble. Therefore, consensus clustering often ensembles at the decision level instead of the data or model level.

One natural idea of consensus clustering is to reformulate the consensus clustering problem to a new clustering problem, where each instance is represented by its base clustering result instead of the original features [23], [24], [25]. For example, Topchy et al. first regarded the base clustering results as the new categorical features and applied the expectation-maximization clustering method on them to obtain the final clustering result [23]; Nguyen et al. also viewed the base results as the new categorical data and performs K-modes method for consensus clustering [24]; Bai et al. provided an information theoretical framework to do consensus clustering [25].

Some methods relabel each data by label alignment methods based on the multiple clustering results [26], [27], [28]. For example, Zhou et al. designed an alignment method for multiple K-means [26]; Hore et al. proposed a scalable relabel algorithm for consensus clustering [27]; Li et al. designed a Dempster-Shafer evidence theory based relabel method to learn a consensus clustering result [28].

Another common strategy for consensus clustering is to transform the base clustering results to graphs and learn the consensus result from the graphs [29], [30], [31], [32], [33], [34]. For example, Fern et al. constructed bipartite graph from base results for consensus learning [35]; Iam-On et al. constructed the graph with the similarity metric based on the link of data for consensus learning [36], [37]; Tao et al. applied spectral clustering on the graph constructed from base results [29]; Huang et al. proposed an ultra-scalable spectral clustering method on the graph for

consensus clustering [30]; Zhou et al. designed a tri-level robust clustering ensemble method by multiple graph learning [32]; Strehl et al. and Zhou et al. learned the consensus results on the carefully constructed hyper-graph [3], [33].

Although consensus clustering has demonstrated promising performance, all of these methods use fixed pre-given base results and never update the base results. Different from these conventional methods, in this paper, we adaptively adjust the base results and learn the consensus result from the refined base results. By making the two tasks, i.e. base results refining and ensemble, be boosted by each other, we can further improve the clustering performance.

### C. Graph Filter

Given a graph $\mathcal{G}$ with $n$ nodes $\{v_1, \ldots, v_n\}$, whose adjacency matrix is $\mathbf{W} \in \mathbb{R}^{n\times n}$ where $W_{ij} \geq 0$ and $\mathbf{W} = \mathbf{W}^T$, its normalized Laplacian matrix is $\mathbf{L} = \mathbf{I} - \mathbf{D}^{-\frac{1}{2}}\mathbf{W}\mathbf{D}^{-\frac{1}{2}}$, where $\mathbf{I}$ is an identity matrix, and $\mathbf{D}$ is a diagonal matrix whose diagonal element $D_{ii} = \sum_{j=1}^{n} W_{ij}$. Then we can obtain its eigenvalue decomposition $\mathbf{L} = \mathbf{U}\mathbf{\Sigma}\mathbf{U}^T$, where $\mathbf{U} \in \mathbb{R}^{n\times n}$ contains $n$ eigenvectors of $\mathbf{L}$ and $\mathbf{\Sigma} \in \mathbb{R}^{n\times n}$ is a diagonal matrix whose diagonal elements are the eigenvalues of $\mathbf{L}$. Obviously, all diagonal elements of $\mathbf{\Sigma}$ are in the range [0,2]. According to spectral graph theory [38], the eigenvectors of $\mathbf{L}$ are the Fourier bases of the graph and the eigenvalues can be viewed as the associated frequencies.

Given a graph signal $\mathbf{s} = [s(v_1), \ldots, s(v_n)]^T$ on $\mathcal{G}$, graph filter aims to provide an operation (relative with $\mathcal{G}$) on $\mathbf{s}$. In our clustering scenario, the data matrix $\mathbf{X} \in \mathbb{R}^{n\times d}$ can be viewed as a kind of graph signal. If $\mathbf{X}$ has a clear clustering structure, it should follow the cluster and manifold assumption, i.e., the data in a cluster should be close to each other. According to [39], [40], a smoother graph signal can lead to a clearer clustering structure of data, which follows such cluster and manifold assumption.

Therefore, we need a graph filter to make the signal $\mathbf{X}$ smoother. Some recent research [11], [39] shows that the smooth signal should contain more low-frequency bases than the high-frequency bases. To achieve this, they suggest constructing the filtered signal $\hat{\mathbf{s}}$ as

$$\hat{\mathbf{s}} = \mathbf{U}\left(\mathbf{I} - \frac{\mathbf{\Sigma}}{2}\right)^k \mathbf{U}^T\mathbf{s} = \left(\mathbf{I} - \frac{\mathbf{L}}{2}\right)^k \mathbf{s}, \tag{2}$$

where $k$ is a positive integer to capture the $k$-hop neighborhood relation [41]. Notice that small eigenvalues of $\mathbf{L}$, which corresponds to the low-frequency parts, will lead to large eigenvalues of $(\mathbf{I} - \frac{\mathbf{L}}{2})^k$, which can preserve the low-frequency parts and remove the high-frequency parts. Here it uses $(\mathbf{I} - \frac{\mathbf{L}}{2})^k$ instead of $(\mathbf{I} - \mathbf{L})^k$, because the eigenvalues of $\mathbf{L}$ are in the range [0,2], and we transform all eigenvalues into the range [0,1] by the filter $(\mathbf{I} - \frac{\mathbf{L}}{2})^k$, or otherwise $(\mathbf{I} - \mathbf{L})^k$ is not a positive semi-definite matrix and may contain negative eigenvalues.

The conventional graph filter methods, such as [11], [12], often construct a pre-defined fixed graph filter, which may be inappropriate for the complex real-world data. To tackle this issue, in this work, we design a learnable graph filer on $\mathbf{X}$ and obtain the smoother embedding $(\mathbf{I} - \frac{\mathbf{L}}{2})^k\mathbf{X}$.

## III. Adaptive Consensus Multiple K-Means

In this section, we introduce our ACMK method in more detail.

### A. Multiple K-Means

Given a data set $\mathbf{X} \in \mathbb{R}^{n\times d}$ with $n$ instances and $d$ features, we can use (1) to obtain the clustering result matrix $\mathbf{G}$. However, as introduced before, K-means is not stable, i.e., with different initializations, K-means may obtain very different clustering results. To tackle this problem, we run $v$ times K-means with different initializations and learn a consensus one from the multiple results. To run $v$ times K-means, we can use the following formula:

$$\begin{aligned} \min_{\mathbf{G}^{(m)},\mathbf{F}^{(m)}} \quad & \sum_{m=1}^{v} \|\mathbf{X} - \mathbf{G}^{(m)}\mathbf{F}^{(m)}\|_F^2, \\ s.t. \quad & \mathbf{G}^{(m)} \in \{0,1\}^{n\times c_m}, \quad \forall i : \sum_{j=1}^{c_m} G_{ij}^{(m)} = 1. \end{aligned} \tag{3}$$

where $\mathbf{G}^{(m)}$ and $\mathbf{F}^{(m)}$ are the $m$-th cluster indicator matrix and cluster centroid matrix, respectively. $c_m$ is the number of clusters in the $m$-th K-means result.

### B. Base Results Refining With Graph Filter

Before ensembling multiple $\mathbf{G}^{(m)}$ to obtain a consensus result, we observe that traditional consensus clustering methods often directly fuse fixed $\mathbf{G}^{(m)}$. However, since single K-means may be imperfect, directly ensembling these weak base results may also lead to an unreliable consensus result. To address this issue, we adaptively update the base results $\mathbf{G}^{(m)}$ and ensemble the refined base results instead.

To this end, we wish to obtain a cluster-friendly embedding of $\mathbf{X}$ and learn the base results $\mathbf{G}^{(m)}$ from such cluster-friendly embedding as the refined base results. As introduced before, a cluster-friendly embedding should satisfy the cluster and manifold assumption, i.e., the embeddings of data which are in the same cluster should be close to each other. According to [39], since graph filter can provide a smooth and cluster-friendly embedding of the original data, we can use graph filter to generate a better embedding of $\mathbf{X}$ and perform K-means on such embedding. Due to the cluster-friendly embedding, the learned $\mathbf{G}^{(m)}$ will be more reliable.

To perform the graph filter, we need a graph $\mathcal{G}$ of the data. Conventional graph filter methods often use a pre-defined graph to construct the filter. However, the pre-defined fixed graph may not be flexible enough to handle some complex data $\mathbf{X}$. To tackle this problem, we design a learnable graph filter, whose structure will be adaptively updated in the process of the ensemble. Here we assume that we already have such graph $\mathcal{G}$ and we will introduce how to learn it later.

Suppose that the adjacency matrix of $\mathcal{G}$ is $\mathbf{W} \in [0,1]^{n\times n}$. We then construct its normalized Laplacian matrix $\mathbf{L} = \mathbf{I} - \mathbf{D}^{-\frac{1}{2}}\mathbf{W}\mathbf{D}^{-\frac{1}{2}}$. As introduced before, we perform the high-order graph filter $(\mathbf{I} - \frac{\mathbf{L}}{2})^k$ on $\mathbf{X}$, which can characterize the high-order relationship among instances, where $k$ is the number of

hops to capture the high-order relation of instances and in our implementation we fix it to 3 for simplicity. By the graph filter, we obtain a high-order smooth embedding $(\mathbf{I}-\frac{\mathbf{L}}{2})^k\mathbf{X}$. Then, we run multiple K-means on this embedding:

$$\min_{\mathbf{G}^{(m)},\mathbf{F}^{(m)}} \sum_{m=1}^{v}\left\|\left(\mathbf{I}-\frac{\mathbf{L}}{2}\right)^k\mathbf{X}-\mathbf{G}^{(m)}\mathbf{F}^{(m)}\right\|_F^2, \quad s.t.\ \ \mathbf{G}^{(m)}\in\{0,1\}^{n\times c_m},\ \ \forall i:\sum_{j=1}^{c_m}G_{ij}^{(m)}=1. \tag{4}$$

### C. Consensus Learning for the Graph Structure

Now we consider how to construct the adaptive adjacency matrix $\mathbf{W}$. Different from the conventional methods which directly construct a static graph, we learn a dynamic graph $\mathbf{W}$ from base clustering results. The goal is twofold: first, $\mathbf{W}$ should fuse the information in base clustering results ($\mathbf{G}^{(m)}$); second, $\mathbf{W}$ should be adaptively updated because that, with better base clusterings, we can construct a more accurate graph.

To achieve this, we impose a simple yet effective fusion term $\|\mathbf{W}-\sum_{m=1}^{v}\alpha_m\mathbf{G}^{(m)}\mathbf{G}^{(m)T}\|_F^2$ on (4), where $\alpha_m\in[0,1]$ is the weight of the $m$-th base result. Although this term seems simple, it can leverage the two important properties of consensus clustering, i.e. *consensus* and *diversity*. Notice that, when minimizing this term, we have

$$\begin{aligned}
&\min_{\mathbf{W},\mathbf{G}^{(m)}}\left\|\mathbf{W}-\sum_{m=1}^{v}\alpha_m\mathbf{G}^{(m)}\mathbf{G}^{(m)T}\right\|_F^2\\
&=\min_{\mathbf{W},\mathbf{G}^{(m)}}\|\mathbf{W}\|_F^2-2\left\langle\mathbf{W},\sum_{m=1}^{v}\alpha_m\mathbf{G}^{(m)}\mathbf{G}^{(m)T}\right\rangle\\
&\quad+\sum_{m=1}^{v}\sum_{p=1}^{v}\alpha_m\alpha_p\left\langle\mathbf{G}^{(m)}\mathbf{G}^{(m)T},\mathbf{G}^{(p)}\mathbf{G}^{(p)T}\right\rangle,\\
&=\min_{\mathbf{W},\mathbf{G}^{(m)}}\|\mathbf{W}\|_F^2-\underbrace{2\left\langle\mathbf{W},\sum_{m=1}^{v}\alpha_m\mathbf{G}^{(m)}\mathbf{G}^{(m)T}\right\rangle}_{Consensus}\\
&\quad+\underbrace{\sum_{m\neq p}^{v}\alpha_m\alpha_p\left\langle\mathbf{G}^{(m)}\mathbf{G}^{(m)T},\mathbf{G}^{(p)}\mathbf{G}^{(p)T}\right\rangle}_{Diversity}\\
&\quad+\sum_{m=1}^{v}\alpha_m^2\|\mathbf{G}^{(m)}\mathbf{G}^{(m)T}\|_F^2,
\end{aligned}\tag{5}$$

where $\langle\cdot,\cdot\rangle$ is the inner production of two matrices. The first term in (5) is a Frobenius norm regularizer of $\mathbf{W}$. When minimizing the second term, which is equivalent to maximizing the inner production of $\mathbf{W}$ and $\sum_{m=1}^{v}\alpha_m\mathbf{G}^{(m)}\mathbf{G}^{(m)T}$, which makes $\mathbf{W}$ to preserve the information of all $\mathbf{G}^{(m)}$'s. Obviously, it captures the *consensus* property. Minimizing the third term is to make any two base results $\mathbf{G}^{(m)}$ and $\mathbf{G}^{(p)}$ far away from each other, which makes the refined base clusterings to be different from each other, and it characterizes the *diversity* property. Moreover, let us take a closer look at the fourth term $\sum_{m=1}^{v}\alpha_m^2\|\mathbf{G}^{(m)}\mathbf{G}^{(m)T}\|_F^2$. Suppose there are $c$ clusters in $\mathbf{G}^{(m)}$, and there are $n_1,\ldots,n_c$ instances in the $c$ clusters, respectively. Since there is only one 1 in each row of $\mathbf{G}^{(m)}$, it is easy to verify that $\|\mathbf{G}^{(m)}\mathbf{G}^{(m)T}\|_F^2=\sum_{i=1}^{c}n_i^2$. Since $\sum_{i=1}^{c}n_i=n$ which is a constant, to minimize $\|\mathbf{G}^{(m)}\mathbf{G}^{(m)T}\|_F^2$ which is equivalent to minimizing $\sum_{i=1}^{c}n_i^2$, will lead to a more balanced clustering result which can avoid the clustering collapse (i.e., most data are put in one cluster) to some extent.

Now, we obtain our final objective function:

$$\begin{aligned}
\min_{\mathbf{W},\boldsymbol{\alpha},\mathbf{G}^{(m)},\mathbf{F}^{(m)}}\ &\sum_{m=1}^{v}\left\|\left(\frac{\mathbf{I}+\mathbf{D}^{-\frac{1}{2}}\mathbf{W}\mathbf{D}^{-\frac{1}{2}}}{2}\right)^k\mathbf{X}-\mathbf{G}^{(m)}\mathbf{F}^{(m)}\right\|_F^2\\
&+\lambda\left\|\mathbf{W}-\sum_{m=1}^{v}\alpha_m\mathbf{G}^{(m)}\mathbf{G}^{(m)T}\right\|_F^2,\\
s.t.\quad &\mathbf{G}^{(m)}\in\{0,1\}^{n\times c_m},\ \ \forall i:\sum_{j=1}^{c_m}G_{ij}^{(m)}=1,\\
&\forall i,j:\ 0\le W_{ij}\le 1,\ \ \mathbf{W}=\mathbf{W}^T,\\
&\forall m:\ 0\le\alpha_m\le 1,\ \ \sum_{m=1}^{v}\alpha_m=1,
\end{aligned}\tag{6}$$

where the constraints on $\mathbf{W}$ make sure that the values in the adjacency matrix are between 0 and 1 and the adjacency matrix should be symmetric, and $\lambda$ is a balanced hyper-parameter.

Notice that on one hand, when optimizing $\mathbf{G}^{(m)}$ and $\mathbf{F}^{(m)}$, we are doing the multiple K-means on the smooth embedding of the data and moreover it can force the base clusterings to be diverse from each other. On the other hand, when optimizing $\mathbf{W}$ and $\boldsymbol{\alpha}$, we are ensembling multiple base results to construct the consensus graph $\mathbf{W}$.

### D. Optimization

In this subsection, we will introduce how to optimize (6) in detail. Note that the graph filter also involves $\mathbf{W}$ and makes (6) difficult to be solved directly. To address this issue, we apply an ADMM method to optimize it iteratively. First, by introducing two auxiliary variables $\mathbf{A}=\frac{\mathbf{I}+\mathbf{D}^{-\frac{1}{2}}\mathbf{W}\mathbf{D}^{-\frac{1}{2}}}{2}$ and $\mathbf{V}=\mathbf{W}$, we can rewrite (6) as following:

$$\begin{aligned}
\min_{\boldsymbol{\theta}}\ &\sum_{m=1}^{v}\left\|\mathbf{A}^k\mathbf{X}-\mathbf{G}^{(m)}\mathbf{F}^{(m)}\right\|_F^2\\
&+\lambda\left\|\mathbf{W}-\sum_{m=1}^{v}\alpha_m\mathbf{G}^{(m)}\mathbf{G}^{(m)T}\right\|_F^2,\\
s.t.\ \ &\mathbf{G}^{(m)}\in\{0,1\}^{n\times c_m},\ \ \forall i:\sum_{j=1}^{c_m}G_{ij}^{(m)}=1,\\
&\mathbf{A}=(\mathbf{I}+\mathbf{D}^{-\frac{1}{2}}\mathbf{W}\mathbf{D}^{-\frac{1}{2}})/2,\\
&\forall i,j:\ 0\le W_{ij}\le 1,\ \ \mathbf{W}=\mathbf{V},\ \ \mathbf{V}=\mathbf{V}^T,\\
&\forall m:\ 0\le\alpha_m\le 1,\ \ \sum_{m=1}^{v}\alpha_m=1.
\end{aligned}\tag{7}$$

where $\boldsymbol{\theta} = \{\mathbf{W}, \mathbf{A}, \mathbf{V}, \boldsymbol{\alpha}, \mathbf{G}^{(m)}, \mathbf{F}^{(m)}\}$ is the set of all learnable parameters.

By introducing the Lagrange multipliers $\boldsymbol{\Lambda}_1 \in \mathbb{R}^{n\times n}$ and $\boldsymbol{\Lambda}_2 \in \mathbb{R}^{n\times n}$, we can obtain its Lagrange formula:

$$\begin{aligned}\min_{\boldsymbol{\theta}} \quad & \sum_{m=1}^{v} \left\|\mathbf{A}^k\mathbf{X} - \mathbf{G}^{(m)}\mathbf{F}^{(m)}\right\|_F^2 \\ & +\lambda \left\|\mathbf{W} - \sum_{m=1}^{v} \alpha_m \mathbf{G}^{(m)}\mathbf{G}^{(m)T}\right\|_F^2 \\ & + \left\langle \boldsymbol{\Lambda}_1, \mathbf{A} - (\mathbf{I} + \mathbf{D}^{-\frac{1}{2}}\mathbf{W}\mathbf{D}^{-\frac{1}{2}})/2 \right\rangle + \langle \boldsymbol{\Lambda}_2, \mathbf{W} - \mathbf{V}\rangle \\ & + \frac{\mu}{2}\left(\|\mathbf{A} - (\mathbf{I} + \mathbf{D}^{-\frac{1}{2}}\mathbf{W}\mathbf{D}^{-\frac{1}{2}})/2\|_F^2 + \|\mathbf{W} - \mathbf{V}\|_F^2\right), \\ s.t. \quad & \mathbf{G}^{(m)} \in \{0,1\}^{n\times c_m}, \quad \forall i : \sum_{j=1}^{c_m} G_{ij}^{(m)} = 1, \\ & \forall i,j : \ 0 \le W_{ij} \le 1, \quad \mathbf{V} = \mathbf{V}^T, \\ & \forall m : \ 0 \le \alpha_m \le 1, \quad \sum_{m=1}^{v} \alpha_m = 1. \end{aligned} \tag{8}$$

where $\mu > 0$ is an adaptive parameter. Notice that, by introducing the Lagrange multipliers $\boldsymbol{\Lambda}_1$ and $\boldsymbol{\Lambda}_2$, we move the constraints $\mathbf{A} = (\mathbf{I} + \mathbf{D}^{-\frac{1}{2}}\mathbf{W}\mathbf{D}^{-\frac{1}{2}})/2$ and $\mathbf{W} = \mathbf{V}$ to the objective function, and $\mathbf{A}$, $\mathbf{W}$, and $\mathbf{V}$ are independent variables, which can be optimized by fixing others.

We initialize $\mathbf{G}^{(m)}$ and $\mathbf{F}^{(m)}$ by running K-means on $\mathbf{X}$ with different initializations. We initialize $\alpha_m = 1/v$, $\mathbf{W} = \frac{1}{v}\sum_{m=1}^{v} \mathbf{G}^{(m)}\mathbf{G}^{(m)T}$, $\mathbf{V} = \mathbf{W}$, $\boldsymbol{\Lambda}_1 = \boldsymbol{\Lambda}_2 = \mathbf{0}$, and $\mu = 1$. Then, we minimize (8) by optimizing one variable while fixing other variables.

*1) Optimize* $\mathbf{A}$*:* When optimizing $\mathbf{A}$, we drop the terms irrelative to $\mathbf{A}$ and obtain:

$$\begin{aligned}\min_{\mathbf{A}} \mathcal{J}_1 = & \, v tr\left(\mathbf{A}^k\mathbf{X}\mathbf{X}^T\mathbf{A}^{kT}\right) - 2tr\left(\sum_{m=1}^{v}\mathbf{G}^{(m)}\mathbf{F}^{(m)}\mathbf{X}^T\mathbf{A}^k\right) \\ & + tr\left(\boldsymbol{\Lambda}_1^T\mathbf{A}\right) + \frac{\mu}{2}\|\mathbf{A} - (\mathbf{I} + \mathbf{D}^{-\frac{1}{2}}\mathbf{W}\mathbf{D}^{-\frac{1}{2}})/2\|_F^2. \end{aligned} \tag{9}$$

Notice that (9) is a non-constraint optimization problem and it can be solved by the standard Quasi-Newton method. In our implementation, we use L-BFGS algorithm [42]. To apply the Quasi-Newton method, we need the partial derivative of $\mathcal{J}_1$ w.r.t. $\mathbf{A}$. Denoting $\mathbf{B} = \sum_{m=1}^{v}\mathbf{X}\mathbf{F}^{(m)T}\mathbf{G}^{(m)T}$ and $\mathbf{C} = (\mathbf{I} + \mathbf{D}^{-\frac{1}{2}}\mathbf{W}\mathbf{D}^{-\frac{1}{2}})/2$, according to the chain rule of derivative, we have

$$\begin{aligned}\frac{\partial \mathcal{J}_1}{\partial \mathbf{A}} = & \, 2v\sum_{r=0}^{k-1}\left(\mathbf{A}^{k-r-1}\mathbf{X}\mathbf{X}^T\mathbf{A}^{kT}\mathbf{A}^r\right)^T \\ & - 2\sum_{r=0}^{k-1}\left(\mathbf{A}^r\mathbf{B}\mathbf{A}^{k-r-1}\right)^T + \boldsymbol{\Lambda}_1 + \mu(\mathbf{A} - \mathbf{C}). \end{aligned} \tag{10}$$

The detailed derivation can be found in Appendix, which can be found on the Computer Society Digital Library at http://doi.ieeecomputersociety.org/10.1109/TKDE.2023.3264970.

*2) Optimize* $\mathbf{W}$*:* When optimizing $\mathbf{W}$, by denoting $\mathbf{E} = \mathbf{A} - \frac{\mathbf{I}}{2}$ and $\mathbf{F} = \sum_{m=1}^{v}\alpha_m\mathbf{G}^{(m)}\mathbf{G}^{(m)T}$, we rewrite the objective function as follows:

$$\begin{aligned}\min_{\mathbf{W}} \ \mathcal{J}_2 = & \, \lambda\|\mathbf{W} - \mathbf{F}\|_F^2 - \frac{1}{2}tr(\boldsymbol{\Lambda}_1^T\mathbf{D}^{-\frac{1}{2}}\mathbf{W}\mathbf{D}^{-\frac{1}{2}}) \\ & + tr(\boldsymbol{\Lambda}_2^T\mathbf{W}) + \frac{\mu}{8}tr(\mathbf{D}^{-\frac{1}{2}}\mathbf{W}^T\mathbf{D}^{-1}\mathbf{W}\mathbf{D}^{-\frac{1}{2}}) \\ & - \frac{\mu}{2}tr\left(\mathbf{E}^T\mathbf{D}^{-\frac{1}{2}}\mathbf{W}\mathbf{D}^{-\frac{1}{2}}\right) + \frac{\mu}{2}\|\mathbf{W} - \mathbf{V}\|_F^2, \\ s.t. \ & \forall i,j : \ 0 \le W_{ij} \le 1. \end{aligned} \tag{11}$$

(11) can also be solved by L-BFGS algorithm. However, when computing the partial derivative of $\mathcal{J}_2$ w.r.t. $\mathbf{W}$, we should be careful of $\mathbf{D}$ which is also relative with $\mathbf{W}$. Taking $D_{ii} = \sum_{j=1}^{n} W_{ij}$ into (11) and performing the chain rule, we have

$$\begin{aligned}\frac{\partial \mathcal{J}_2}{\partial \mathbf{W}} = & \, 2\lambda(\mathbf{W} - \mathbf{F}) - \frac{1}{2}\mathbf{D}^{-\frac{1}{2}}(\boldsymbol{\Lambda}_1 + \mu\mathbf{E})\mathbf{D}^{-\frac{1}{2}} \\ & + \frac{1}{4}diag\left(\mathbf{D}^{-\frac{3}{2}}\left((\boldsymbol{\Lambda}_1 + \mu\mathbf{E})^T\mathbf{D}^{-\frac{1}{2}}\mathbf{W}\right.\right. \\ & \left.\left. + \ (\boldsymbol{\Lambda}_1 + \mu\mathbf{E})\mathbf{D}^{-\frac{1}{2}}\mathbf{W}^T\right)\right)\mathbf{1}^T \\ & + \boldsymbol{\Lambda}_2 + \frac{\mu}{4}\mathbf{D}^{-1}\mathbf{W}\mathbf{D}^{-1} + \mu(\mathbf{W} - \mathbf{V}) \\ & - \frac{\mu}{8}diag\left(\mathbf{D}^{-2}\left(\mathbf{W}^T\mathbf{D}^{-1}\mathbf{W} + \mathbf{W}\mathbf{D}^{-1}\mathbf{W}^T\right)\right)\mathbf{1}^T, \end{aligned} \tag{12}$$

where $\mathbf{1}$ is a vector whose elements are all 1 s, and $diag(\mathbf{M})$ is the diagonal vector of matrix $\mathbf{M}$. The detailed derivation can also be found in Appendix, available in the online supplemental material.

*3) Optimize* $\mathbf{G}^{(m)}$*:* When optimizing $\mathbf{G}^{(m)}$, we reformulate the objective function as:

$$\begin{aligned}\min_{\mathbf{G}^{(m)}} \quad & \left\|\mathbf{A}^k\mathbf{X} - \mathbf{G}^{(m)}\mathbf{F}^{(m)}\right\|_F^2 - 2\lambda\alpha_m tr(\mathbf{W}^T\mathbf{G}^{(m)}\mathbf{G}^{(m)T}) \\ & + \lambda\alpha_m^2 tr(\mathbf{G}^{(m)}\mathbf{G}^{(m)T}\mathbf{G}^{(m)}\mathbf{G}^{(m)T}) \\ & + \lambda\alpha_m\sum_{p\neq m}\alpha_p tr(\mathbf{G}^{(m)}\mathbf{G}^{(m)T}\mathbf{G}^{(p)}\mathbf{G}^{(p)T}), \\ s.t. \quad & \mathbf{G}^{(m)} \in \{0,1\}^{n\times c_m}, \quad \forall i : \sum_{j=1}^{c_m} G_{ij}^{(m)} = 1. \end{aligned} \tag{13}$$

Notice that there is only one 1 in each row of $\mathbf{G}^{(m)}$. Therefore, we can optimize $\mathbf{G}^{(m)}$ row by row. In more detail, when optimizing the $i$-th row, we define $\mathbf{g}_1, \ldots, \mathbf{g}_{c_m}$ where $\mathbf{g}_j \in \{0,1\}^{1\times c_m}$ and only the $j$-th element in $\mathbf{g}_j$ is 1 and other elements are 0 s. Then, we set the $i$-th row as $\mathbf{g}_1, \ldots, \mathbf{g}_{c_m}$ respectively and find the one which leads to the minimum of the objective function, and set the $i$-th row of $\mathbf{G}^{(m)}$ as it.

*4) Optimize* $\mathbf{F}^{(m)}$*:* When optimizing $\mathbf{F}^{(m)}$, we can rewrite the objective function as:

$$\min_{\mathbf{F}^{(m)}} \quad \left\|\mathbf{A}^k\mathbf{X} - \mathbf{G}^{(m)}\mathbf{F}^{(m)}\right\|_F^2. \tag{14}$$

By setting its partial derivative w.r.t. $\mathbf{F}^{(m)}$ to zero, we can obtain its closed-form solution:

$$\mathbf{F}^{(m)} = \left(\mathbf{G}^{(m)T}\mathbf{G}^{(m)}\right)^{-1}\mathbf{G}^{(m)T}\mathbf{A}^{k}\mathbf{X}. \tag{15}$$

Notice that $\mathbf{G}^{(m)T}\mathbf{G}^{(m)}$ is a diagonal matrix whose inverse can be computed in $O(n)$ time.

*5) Optimize* $\mathbf{V}$*:* By fixing other variables, we obtain the objective function w.r.t. $\mathbf{V}$ as follows:

$$\begin{aligned} \min_{\mathbf{V}} \quad & \left\|\mathbf{V} - \left(\mathbf{W} + \frac{\mathbf{\Lambda}_2}{\mu}\right)\right\|_F^2, \\ s.t. \quad & \mathbf{V} = \mathbf{V}^T. \end{aligned} \tag{16}$$

This is a projection problem and its closed-form solution is:

$$\mathbf{V} = \left(\mathbf{W} + \mathbf{W}^T\right)/2 + \left(\mathbf{\Lambda}_2 + \mathbf{\Lambda}_2^T\right)/(2\mu). \tag{17}$$

*6) Optimize* $\boldsymbol{\alpha}$*:* When optimizing $\boldsymbol{\alpha}$, we reformulate the objective function as the following convex quadratic programming:

$$\begin{aligned} \min_{\boldsymbol{\alpha}} \quad & \boldsymbol{\alpha}^T\mathbf{H}\boldsymbol{\alpha} + \mathbf{f}^T\boldsymbol{\alpha}, \\ s.t. \quad & \forall m:\ 0 \le \alpha_m \le 1, \quad \sum_{m=1}^{v}\alpha_m = 1, \end{aligned} \tag{18}$$

where $H_{mp} = tr(\mathbf{G}^{(m)}\mathbf{G}^{(m)T}\mathbf{G}^{(p)}\mathbf{G}^{(p)T})$ and $f_m = -2tr(\mathbf{W}^T\mathbf{G}^{(m)}\mathbf{G}^{(m)T})$. Eq(18) can be solved by any standard quadratic programming method. In our implementation, we solve it by *quadprog* function provided by Matlab.

*7) Update Lagrange Multipliers:* We update the Lagrange multipliers as follows:

$$\begin{cases} \mathbf{\Lambda}_1 \leftarrow \mathbf{\Lambda}_1 + \mu\left(\mathbf{A} - (\mathbf{I} + \mathbf{D}^{-\frac{1}{2}}\mathbf{W}\mathbf{D}^{-\frac{1}{2}})/2\right), \\ \mathbf{\Lambda}_2 \leftarrow \mathbf{\Lambda}_2 + \mu\left(\mathbf{W} - \mathbf{V}\right), \\ \mu \leftarrow 1.05 * \mu. \end{cases} \tag{19}$$

Algorithm 1 summarizes the whole process. After obtaining the consensus graph adjacency matrix $\mathbf{W}$, we can use it to generate the final clustering result in many ways. For example, we can run spectral clustering on $\mathbf{W}$, or we can run K-means on the final embedding data $(\frac{\mathbf{I}+\mathbf{D}^{-\frac{1}{2}}\mathbf{W}\mathbf{D}^{-\frac{1}{2}}}{2})^k\mathbf{X}$.

**Algorithm 1:** Adaptive Consensus Multiple K-Means.

**Input:** Data matrix $\mathbf{X}$, $v$ times of running K-means, hyper-parameter $k$ and $\lambda$.
**Output:** Final consensus clustering result.

```
Initialize G^(m), F^(m), W, V, α, Λ_1, Λ_2, and μ as introduced in Section Optimization.
while not converge do
    Update A by solving (9).
    Update W by solving (11).
    for m = 1, ..., v do
        Update G^(m) by solving (13) row by row.
        Update F^(m) by (15).
    end for
    Update V by (17).
    Update α by solving (18).
    Update Lagrange multipliers by (19).
end while
Obtain the final clustering result from W with K-means or spectral clustering.
```

## E. Discussion

Now we analyze the time complexity of Algorithm 1. Since the order $k$ in our method is set as 3, it can be regarded as a very small constant. When updating $\mathbf{A}$, we need to compute the gradient by (10). In (10), we first compute $\mathbf{A}^{k-r-1}\mathbf{X}\mathbf{X}^T\mathbf{A}^{kT}\mathbf{A}^r$. Denotes $n$ as the number of instances and $d$ as the number of features. If $n > d$, by using the associative property of matrix multiplication, we spend $O(n^2\,d)$ time to compute $\mathbf{A}^{k-r-1}\mathbf{X}$ and $\mathbf{X}^T\mathbf{A}^{kT}\mathbf{A}^r$. Then we spend $O(n^2\,d)$ time to multiply these two terms to obtain $\mathbf{A}^{k-r-1}\mathbf{X}\mathbf{X}^T\mathbf{A}^{kT}\mathbf{A}^r$. If $n < d$, we can calculate $\mathbf{X}\mathbf{X}^T$ in $O(n^2\,d)$ time in advance (i.e., just need to compute it once), and in the following computation, we just need $O(n^3)$ time for matrix multiplication, which is faster than $O(n^2\,d)$. When computing the second term $\mathbf{A}^r\mathbf{B}\mathbf{A}^{k-r-1}$, we take the definition of $\mathbf{B}$ into it and obtain $\mathbf{A}^r\sum_{m=1}^{v}\mathbf{X}\mathbf{F}^{(m)T}\mathbf{G}^{(m)T}\mathbf{A}^{k-r-1}$. Also by the associative property of matrix multiplication, we compute it in $O(v(ndc + n^2c))$ time, where $v$ is the number of bases and $c$ is the number of clusters, which are both small in practice.

When updating $\mathbf{W}$, we need to compute the gradient by (12). In (12), we need to compute $diag(\mathbf{D}^{-3/2}((\mathbf{\Lambda}_1 + \mu\mathbf{E})^T\mathbf{D}^{-1/2}\mathbf{W} + (\mathbf{\Lambda}_1 + \mu\mathbf{E})\mathbf{D}^{-1/2}\mathbf{W}^T)$. Since we just need the diagonal vector of this matrix, it takes $O(n^2)$ time. Similarly, it also takes $O(n^2)$ time to compute $diag(\mathbf{D}^{-2}(\mathbf{W}^T\mathbf{D}^{-1}\mathbf{W} + \mathbf{W}\mathbf{D}^{-1}\mathbf{W}^T))$.

When optimizing $\mathbf{G}^{(m)}$, we can compute $\mathbf{A}^k\mathbf{X}$ in $O(n^2\,d)$ time in advance and just compute it once. Then we update each row of $\mathbf{G}^{(m)}$ to decide the location of the unique 1. It takes $O(nc)$ time. Updating $\mathbf{F}^{(m)}$ is the same as the conventional K-means, which takes $O(n)$ time. Updating $\mathbf{V}$ only involves matrix addition without any matrix multiplication. When optimizing $\boldsymbol{\alpha}$, we first spend $O(nv^2)$ to compute $\mathbf{H}$. Then, we solve the convex quadratic programming in $O(v^3)$ time.

To sum up, the time complexity is $O(n^2d + vndc + vn^2c + nv^2 + v^3)$. Since in practice, the number of bases and clusters $v$ and $c$ are often small, the bottleneck is about $O(n^2\,d)$. It is linear with the number of dimensions. It is square with the number of instances, which makes it hard to handle very large data sets. However, it is comparable with other graph-based consensus clustering methods, because the time complexity of graph-based methods is often square with the number of instances. In the future, we will continue to study how to further speed up this method and reduce the time complexity.

Then, we try to explain why the refinement can improve the clustering theoretically. We explain it from two aspects. First, we analyze it from the perspective of the graph filter. As claimed by previous literature [43], given a graph Laplacian matrix, its low eigenvalues correspond to the large-scale structure (i.e., clusters) in the graph, and its high eigenvalues correspond to the details but also the noises. Therefore, to obtain a clearer clustering structure, we need a *low-pass* graph filter which can

suppress the high eigenvalues and preserve the low eigenvalues. Wai et al. give a definition of low-pass graph filter [44]. Given a graph and its Laplacian matrix $\mathbf{L}$, the eigenvalues of $\mathbf{L}$ are $0 = \sigma_1 \leq \sigma_2 \leq \cdots \leq \sigma_n \leq 2$. $\mathcal{H} : \mathbb{R}^{n\times n} \to \mathbb{R}^{n\times n}$ is a transform on the Laplacian matrix, and the eigenvalues of matrix $\mathcal{H}(\mathbf{L})$ are $h(\sigma_1), h(\sigma_2), \ldots, h(\sigma_n)$, where $h$ is the corresponding transform of the eigenvalues. Then, according to [44], the low-pass graph filter has the following definition.

*Definition 1.* [44] $\mathcal{H}(\mathbf{L})$ is a $(K, \eta)$ low-pass graph filter if

$$\eta := \frac{\max\left(|h(\sigma_{K+1})|, |h(\sigma_{K+2})|, \ldots, |h(\sigma_n)|\right)}{\min\left(|h(\sigma_1)|, |h(\sigma_2)|, \ldots, |h(\sigma_K)|\right)} < 1 \quad (20)$$

Here $\eta$ is the low-pass coefficient. From Definition 1, given a graph filter $\mathcal{H}(\mathbf{L})$, if there exists an integer $1 \leq K < n$ and a low-pass coefficient $\eta < 1$, then it is a low-pass graph filter. Theorem 1 shows that the learned graph filter of our method $(\mathbf{I} - \frac{\mathbf{L}}{2})^k$ is a low-pass graph filter.

*Theorem 1.* For the learned graph filter of our method $(\mathbf{I} - \frac{\mathbf{L}}{2})^k$, there exists an integer $1 \leq K < n$ such that the low-pass coefficient $\eta$ is smaller than 1, and thus the learned graph filter is a low-pass graph filter.

*Proof.* Given the learned graph filter $(\mathbf{I} - \frac{\mathbf{L}}{2})^k$, its transform on eigenvalues is $h(\sigma_i) = (1 - \frac{\sigma_i}{2})^k$. Then, we compute its low-pass coefficient $\eta$:

$$\begin{aligned}\eta &= \frac{\max(|h(\sigma_{K+1})|, |h(\sigma_{K+2})|, \ldots, |h(\sigma_n)|)}{\min(|h(\sigma_1)|, |h(\sigma_2)|, \ldots, |h(\sigma_K)|)} \\ &= \frac{h(\sigma_{K+1})}{h(\sigma_K)} \\ &= \frac{(1 - \frac{\sigma_{K+1}}{2})^k}{(1 - \frac{\sigma_K}{2})^k} \\ &= \left(\frac{2 - \sigma_{K+1}}{2 - \sigma_K}\right)^k, \end{aligned} \quad (21)$$

where the second equation holds because $h(\sigma_i)$ is a monotonically decreasing function of $\sigma_i$. Since $\sigma_1 = 0$ and as long as there exists a non-zero eigenvalue of $\mathbf{L}$, there exsits a $K$ such that $\sigma_K < \sigma_{K+1}$, and thus $\eta = (\frac{2-\sigma_{K+1}}{2-\sigma_K})^k < 1$. Therefore, our learned graph filter $(\mathbf{I} - \frac{\mathbf{L}}{2})^k$ is a low-pass graph filter. □

Since our learned graph filter is a low-pass graph filter, as introduced before, it can suppress the high eigenvalues which can alleviate the noises on the graph and thus highlight the low eigenvalues which can reveal a clearer clustering structure.

Second, we analyze it from the perspective of spectral graph theory. In the clustering scenario, the data matrix $\mathbf{X}$ can be regarded as graph signals. $\mathbf{X}$ with a clear clustering structure should follow the cluster and manifold assumption, i.e., the data in the same cluster should be close to each other. According to [39], [40], a smooth graph signal tends to follow the cluster and manifold assumption.

To show our method can improve the base results, we can show that, with our learned graph $\mathbf{W}$ and its Laplacian matrix $\mathbf{L}$ and the graph filter $(\mathbf{I} - \frac{\mathbf{L}}{2})^k$, the graph signals $(\mathbf{I} - \frac{\mathbf{L}}{2})^k\mathbf{X}$ are smoother than the original graph signals $\mathbf{X}$. To show this, we need the metric of the smoothness of the graph signal. Here we use the metric defined in [45], [46], that is, given a graph signal $\mathbf{x} \in \mathbb{R}^n$ and the graph $\mathbf{W}$ and its Laplacian matrix $\mathbf{L}$, the smoothness of $\mathbf{x}$ is evaluated by

$$\mathbf{x}^T\mathbf{L}\mathbf{x} = \frac{1}{2}\sum_{i,j=1}^{n} W_{ij}(x_i - x_j)^2. \quad (22)$$

The smaller $\mathbf{x}^T\mathbf{L}\mathbf{x}$ is, the smoother the signal is. It means that, if the $i$-th node and the $j$-th node are close in the graph, i.e., $W_{ij}$ is large, then smooth signal requires that the $i$-th and $j$-th elements in the signal $\mathbf{x}$ (i.e., $x_i$ and $x_j$) should be similar. It is consistent with the cluster and manifold assumption. Given this smoothness metric, we have the following Theorem:

TABLE I
DESCRIPTION OF THE DATA SETS

| | #instances | #features | #classes |
|---|---|---|---|
| 20NG | 3970 | 1000 | 4 |
| Lung | 203 | 3312 | 5 |
| Orlraws | 100 | 10304 | 10 |
| Pendigit | 10992 | 16 | 10 |
| Tr11 | 414 | 6429 | 9 |
| Tr12 | 313 | 5804 | 8 |
| Tr31 | 927 | 10128 | 7 |
| Yale | 165 | 1024 | 15 |

*Theorem 2.* With the learned graph filter $(\mathbf{I} - \frac{\mathbf{L}}{2})^k$ of our method, the signals $(\mathbf{I} - \frac{\mathbf{L}}{2})^k\mathbf{X}$ are smoother than the original signals $\mathbf{X}$.

*Proof.* Here we consider the $i$-th signal $\mathbf{X}_{\cdot i}$ (i.e., the $i$-th column of $\mathbf{X}$), and the analysis of other signals is similar. Given the metric defined before, we try to prove that $((\mathbf{I} - \frac{\mathbf{L}}{2})^k\mathbf{X}_{\cdot i})^T\mathbf{L}((\mathbf{I} - \frac{\mathbf{L}}{2})^k\mathbf{X}_{\cdot i}) \leq \mathbf{X}_{\cdot i}^T\mathbf{L}\mathbf{X}_{\cdot i}$.

Suppose the eigenvalue decomposition of $\mathbf{L}$ is $\mathbf{L} = \mathbf{U}\mathbf{\Sigma}\mathbf{U}^T$. We have

$$\begin{aligned}&\left(\left(\mathbf{I} - \frac{\mathbf{L}}{2}\right)^k\mathbf{X}_{\cdot i}\right)^T\mathbf{L}\left(\left(\mathbf{I} - \frac{\mathbf{L}}{2}\right)^k\mathbf{X}_{\cdot i}\right) \\ &= \mathbf{X}_{\cdot i}^T\left(\mathbf{I} - \frac{\mathbf{L}}{2}\right)^k\mathbf{L}\left(\mathbf{I} - \frac{\mathbf{L}}{2}\right)^k\mathbf{X}_{\cdot i} \\ &= \mathbf{X}_{\cdot i}^T\mathbf{U}\left(\mathbf{I} - \frac{\mathbf{\Sigma}}{2}\right)^k\mathbf{U}^T\mathbf{U}\mathbf{\Sigma}\mathbf{U}^T\mathbf{U}\left(\mathbf{I} - \frac{\mathbf{\Sigma}}{2}\right)^k\mathbf{U}^T\mathbf{X}_{\cdot i} \\ &= \left(\mathbf{U}^T\mathbf{X}_{\cdot i}\right)^T\left(\mathbf{I} - \frac{\mathbf{\Sigma}}{2}\right)^k\mathbf{\Sigma}\left(\mathbf{I} - \frac{\mathbf{\Sigma}}{2}\right)^k\left(\mathbf{U}^T\mathbf{X}_{\cdot i}\right). \end{aligned} \quad (23)$$

Denoting $\mathbf{s} = \mathbf{U}^T\mathbf{X}_{\cdot i}$, we have $((\mathbf{I} - \frac{\mathbf{L}}{2})^k\mathbf{X}_{\cdot i})^T\mathbf{L}((\mathbf{I} - \frac{\mathbf{L}}{2})^k\mathbf{X}_{\cdot i}) = \mathbf{s}^T(\mathbf{I} - \frac{\mathbf{\Sigma}}{2})^k\mathbf{\Sigma}(\mathbf{I} - \frac{\mathbf{\Sigma}}{2})^k\mathbf{s}$, and similarly we also have $\mathbf{X}_{\cdot i}^T\mathbf{L}\mathbf{X}_{\cdot i} = \mathbf{s}^T\mathbf{\Sigma}\mathbf{s}$. Supposing the eigenvalues in $\mathbf{\Sigma}$ are $\sigma_1, \ldots, \sigma_n$, we have

$$\begin{aligned}&\mathbf{s}^T\left(\mathbf{I} - \frac{\mathbf{\Sigma}}{2}\right)^k\mathbf{\Sigma}\left(\mathbf{I} - \frac{\mathbf{\Sigma}}{2}\right)^k\mathbf{s} - \mathbf{s}^T\mathbf{\Sigma}\mathbf{s} \\ &= \sum_{i=1}^{n}(1 - \frac{\sigma_i}{2})^{2k}\sigma_i s_i^2 - \sum_{i=1}^{n}\sigma_i s_i^2 \\ &= \sum_{i=1}^{n}\left((1 - \frac{\sigma_i}{2})^{2k} - 1\right)\sigma_i s_i^2 \leq 0, \end{aligned} \quad (24)$$

TABLE II
ACC RESULTS ON ALL THE DATA SETS

| Methods | 20NG | Lung | Orlraws | Pendigit | Tr11 | Tr12 | Tr31 | Yale |
|---|---|---|---|---|---|---|---|---|
| KM-avg | 0.2577 ±0.0021 | 0.7001 ±0.0260 | 0.6709 ±0.0219 | 0.6846 ±0.0120 | 0.5176 ±0.0225 | 0.4772 ±0.0210 | 0.3982 ±0.0054 | 0.4655 ±0.0091 |
| KM-best | 0.2701 ±0.0118 | 0.8483 ±0.0363 | 0.7780 ±0.0399 | 0.7608 ±0.0136 | 0.6302 ±0.0611 | 0.5786 ±0.0453 | 0.4155 ±0.0014 | 0.5321 ±0.0242 |
| SC [53] | 0.2525 ±0.0006 | 0.5271 ±0.1188 | 0.6850 ±0.0354 | 0.7340 ±0.0617 | 0.6500 ±0.0305 | 0.4246 ±0.0337 | 0.4379 ±0.0386 | 0.5430 ±0.0401 |
| CSPA [3] | 0.3037 ±0.0197 | 0.4207 ±0.0260 | 0.7180 ±0.0305 | 0.6950 ±0.0300 | 0.4749 ±0.0281 | 0.5278 ±0.0286 | 0.1917 ±0.0072 | 0.5394 ±0.0297 |
| HGPA [3] | 0.2573 ±0.0027 | 0.4778 ±0.0353 | 0.7540 ±0.0425 | 0.1114 ±0.0000 | 0.4616 ±0.0485 | 0.4843 ±0.0359 | 0.2309 ±0.0114 | 0.4927 ±0.0277 |
| MCLA [3] | 0.2639 ±0.0062 | 0.6596 ±0.0559 | 0.7770 ±0.0564 | 0.7274 ±0.0398 | 0.4937 ±0.0510 | 0.5550 ±0.0700 | 0.2436 ±0.0146 | 0.5158 ±0.0231 |
| NMFC [54] | 0.2676 ±0.0117 | 0.6286 ±0.1111 | 0.7630 ±0.0400 | 0.7168 ±0.0403 | 0.5469 ±0.0358 | 0.5425 ±0.0381 | 0.3636 ±0.0019 | 0.5255 ±0.0346 |
| LWEA [55] | 0.2534 ±0.0021 | 0.7089 ±0.1218 | 0.8100 ±0.0429 | 0.7088 ±0.0442 | **0.6959** ±0.0516 | 0.5939 ±0.0899 | 0.4093 ±0.0153 | 0.5255 ±0.0300 |
| LWGP [55] | 0.2592 ±0.0019 | 0.6709 ±0.1071 | 0.7790 ±0.0479 | 0.7574 ±0.0346 | 0.5894 ±0.0531 | 0.6163 ±0.0513 | 0.3819 ±0.0244 | 0.5339 ±0.0250 |
| RSEC [29] | 0.2673 ±0.0180 | 0.6108 ±0.1069 | 0.6160 ±0.0790 | 0.6116 ±0.0756 | 0.4273 ±0.0822 | 0.3578 ±0.0322 | 0.3738 ±0.0228 | 0.4042 ±0.0241 |
| DREC [56] | 0.2538 ±0.0018 | 0.8192 ±0.0575 | 0.5590 ±0.0654 | 0.4740 ±0.1004 | 0.6326 ±0.0693 | 0.5211 ±0.1014 | 0.3647 ±0.0235 | 0.3915 ±0.0181 |
| SPCE [10] | 0.2564 ±0.0023 | 0.8571 ±0.0369 | 0.7170 ±0.0422 | 0.4159 ±0.1528 | 0.4150 ±0.1217 | 0.4425 ±0.0538 | 0.3889 ±0.0263 | 0.4733 ±0.0218 |
| SCCBG [31] | 0.2520 ±0.0004 | 0.8089 ±0.0949 | 0.7510 ±0.0363 | 0.6718 ±0.0393 | 0.6196 ±0.0566 | 0.5706 ±0.0799 | 0.4136 ±0.0017 | 0.5024 ±0.0527 |
| TRCE [32] | 0.2565 0.0022 | 0.7323 ±0.1107 | 0.7656 ±0.0508 | 0.7407 ±0.0437 | 0.6249 ±0.0495 | 0.5879 ±0.0468 | 0.4080 ±0.0150 | 0.5545 ±0.0289 |
| CESHL [33] | 0.2546 ±0.0020 | 0.6764 ±0.1022 | 0.7750 ±0.0065 | 0.5101 ±0.0452 | 0.7312 ±0.1564 | 0.5741 ±0.0994 | 0.3996 ±0.0185 | 0.5073 ±0.0528 |
| ACMK-SC | **0.3114** ±0.0191 | 0.5818 ±0.0743 | **0.8150** ±0.0435 | **0.7788** ±0.0167 | 0.5490 ±0.0339 | 0.5588 ±0.0498 | **0.4503** ±0.0550 | **0.5697** ±0.0218 |
| ACMK-KM | 0.2902 ±0.0091 | **0.8783** ±0.0324 | **0.8110** ±0.0202 | 0.7399 ±0.0361 | 0.6708 ±0.0494 | **0. 6786** ±0.0616 | **0.4503** ±0.0550 | **0.5679** ±0.0128 |

where the last inequality holds because that the eigenvalues $\sigma_i$s of $\mathbf{L}$ satisfy $0 \leq \sigma_i \leq 2$. It means that $((\mathbf{I} - \frac{\mathbf{L}}{2})^k \mathbf{X}_{.i})^T \mathbf{L}((\mathbf{I} - \frac{\mathbf{L}}{2})^k \mathbf{X}_{.i}) \leq \mathbf{X}_{.i}^T \mathbf{L} \mathbf{X}_{.i}$, and thus $(\mathbf{I} - \frac{\mathbf{L}}{2})^k \mathbf{X}$ are smoother than the original signals $\mathbf{X}$. □

According to Theorem 2, with the learned graph filter $(\mathbf{I} - \frac{\mathbf{L}}{2})^k$, we can obtain a smoother embedding, and it leads to a clearer clustering structure, which follows the cluster and manifold assumption.

## IV. EXPERIMENTS

In this section, we compare the proposed method with single clustering and consensus clustering methods on benchmark data sets.

### A. Data Sets

We use 8 data sets, including 20NG [47], Lung [48], Orlraws [49], Pendigit [50], Tr11 [51], Tr12 [51], Tr31 [51], Yale [52]. The detailed information is summarized in Table I.

### B. Experimental Setup

In our method, we run K-means with different initializations 10 times to generate 10 base results (i.e., $v = 10$). We fix $k = 3$ and tune $\lambda$ in the range $\{10^{-3}, \ldots, 10^3\}$. After obtaining the consensus adjacency matrix $\mathbf{W}$, we generate the final consensus clustering results in two ways: 1) we run spectral clustering on $\mathbf{W}$, which is denoted as **ACMK-SC**; 2) we run K-means on the final embedding $((\mathbf{I} + \mathbf{D}^{-\frac{1}{2}} \mathbf{W} \mathbf{D}^{-\frac{1}{2}})/2)^k \mathbf{X}$, which is denoted as **ACMK-KM**. To show the effectiveness of the proposed method, we compare it with the following methods:

- **KM-avg**, which is the average result of the 10 base K-means results.
- **KM-best**, which is the best result of the 10 base K-means results.
- **SC** [53], which is the spectral clustering on the original data $\mathbf{X}$. In more detail, we construct $k$-nn graph with a Gaussian kernel of $\mathbf{X}$ and run spectral clustering on such a graph.
- **CSPA** [3], which is a cluster-based similarity partitioning method for consensus clustering.
- **HGPA** [3], which is a hypergraph partitioning algorithm for consensus clustering.
- **MCLA** [3], which is a meta-clustering algorithm for consensus clustering.
- **NMFC** [54], which is a consensus clustering with nonnegative matrix factorization.
- **LWEA** [55], which is a locally weighted consensus clustering method with hierarchical agglomerative.
- **LWGP** [55], which is a locally weighted consensus clustering method with graph partitioning.
- **RSEC** [29], which is a robust spectral consensus clustering method.

TABLE III
NMI RESULTS ON ALL THE DATA SETS

| Methods | 20NG | Lung | Orlraws | Pendigit | Tr11 | Tr12 | Tr31 | Yale |
|---|---|---|---|---|---|---|---|---|
| KM-avg | 0.0089 ±0.0019 | 0.5161 ±0.0218 | 0.7426 ±0.0170 | 0.6623 ±0.0057 | 0.4726 ±0.0257 | 0.3931 ±0.0218 | 0.0610 ±0.0035 | 0.5133 ±0.0075 |
| KM-best | 0.0196 ±0.0105 | 0.6104 ±0.0428 | 0.8265 ±0.0322 | 0.6923 ±0.0068 | 0.5741 ±0.0479 | 0.5141 ±0.0349 | 0.0754 ±0.0047 | 0.5596 ±0.0123 |
| SC [53] | 0.0031 ±0.0004 | 0.4106 ±0.1004 | 0.7457 ±0.0252 | **0.7368** ±0.0407 | 0.6010 ±0.0394 | 0.3711 ±0.0267 | **0.1166** ±0.0187 | 0.5562 ±0.0245 |
| CSPA [3] | **0.0209** ±0.0112 | 0.3581 ±0.0219 | 0.7668 ±0.0193 | 0.6293 ±0.0156 | 0.5098 ±0.0380 | 0.4607 ±0.0460 | 0.0240 ±0.0034 | 0.5637 ±0.0167 |
| HGPA [3] | 0.0003 ±0.0003 | 0.2951 ±0.0394 | 0.8003 ±0.0315 | 0.0014 ±0.0000 | 0.4526 ±0.0317 | 0.4058 ±0.0466 | 0.0193 ±0.0049 | 0.5395 ±0.0181 |
| MCLA [3] | 0.0051 ±0.0029 | 0.4899 ±0.0307 | 0.8024 ±0.0407 | 0.6690 ±0.0254 | 0.4915 ±0.0611 | 0.4890 ±0.0638 | 0.0441 ±0.0102 | 0.5532 ±0.0188 |
| NMFC [54] | 0.0147 ±0.0082 | 0.4722 ±0.0511 | 0.8028 ±0.0270 | 0.6653 ±0.0213 | 0.5685 ±0.0368 | 0.5162 ±0.0476 | 0.0835 ±0.0103 | 0.5599 ±0.0160 |
| LWEA [55] | 0.0046 ±0.0030 | 0.5148 ±0.0850 | 0.8365 ±0.0252 | 0.6557 ±0.0251 | 0.6396 ±0.0362 | 0.5087 ±0.0674 | 0.0595 ±0.0037 | 0.5340 ±0.0180 |
| LWGP [55] | 0.0109 ±0.0019 | 0.4957 ±0.0575 | 0.8200 ±0.0313 | 0.6861 ±0.0135 | 0.5759 ±0.0371 | 0.5549 ±0.0514 | 0.0656 ±0.0070 | 0.5592 ±0.0168 |
| RSEC [29] | 0.0154 ±0.0146 | 0.4094 ±0.0611 | 0.6981 ±0.0738 | 0.5967 ±0.0511 | 0.3245 ±0.0851 | 0.2279 ±0.0530 | 0.0252 ±0.0215 | 0.4363 ±0.0169 |
| DREC [56] | 0.0045 ±0.0024 | 0.5265 ±0.1554 | 0.6787 ±0.0771 | 0.4661 ±0.1069 | 0.5697 ±0.0614 | 0.4109 ±0.0984 | 0.0562 ±0.0183 | 0.4470 ±0.0322 |
| SPCE [10] | 0.0083 ±0.0030 | 0.6275 ±0.0618 | 0.7884 ±0.0336 | 0.3660 ±0.1682 | 0.2140 ±0.1983 | 0.3908 ±0.0637 | 0.0659 ±0.0051 | 0.5782 ±0.0088 |
| SCCBG [31] | 0.0029 ±0.0008 | 0.5912 ±0.0870 | 0.8018 ±0.0257 | 0.6416 ±0.0254 | 0.5199 ±0.0862 | 0.4563 **0**.0885 | 0.0574 ±0.0011 | 0.5231 ±0.0423 |
| TRCE [32] | 0.0081 ±0.0029 | 0.5339 ±0.0684 | 0.8046 ±0.0472 | 0.6837 ±0.0131 | 0.5905 ±0.0801 | 0.5227 ±0.0433 | 0.0575 ±0.0043 | 0.5657 ±0.0174 |
| CESHL [33] | 0.0059 ±0.0031 | 0.5075 ±0.0677 | 0.8107 ±0.0441 | 0.6753 ±0.0208 | 0.3493 ±0.2326 | 0.4400 ±0.0945 | 0.0472 ±0.0196 | 0.5334 ±0.0393 |
| ACMK-SC | 0.0149 ±0.0059 | 0.3443 ±0.0805 | **0.8438** ±0.0278 | 0.7012 0.0092 | 0.5262 ±0.0480 | 0.4628 ±0.0504 | **0.1107** ±0.0307 | 0.5756 ±0.0087 |
| ACMK-KM | 0.0123 ±0.0048 | **0.6716** ±0.0593 | **0.8441** ±0.0150 | 0.6750 ±0.0201 | **0.6559** ±0.0466 | **0.6185** ±0.0541 | **0.1107** ±0.0307 | **0.5945** ±0.0167 |

- **DREC** [56], which is a consensus clustering based on dense representation.
- **SPCE** [10], which is a self-paced clustering ensemble method.
- **SCCBG** [31], which is a self-paced consensus clustering method with bipartite graph learning.
- **TRCE** [32], which is a tri-level robust consensus clustering method.
- **CESHL** [33], which is a consensus clustering method with structured hyper-graph learning.

Among the compared methods, KM-avg, KM-best, and SC are single clustering methods, and the others are all consensus clustering methods. We repeat experiments 10 times with different 10 base results generated by K-means and report the mean results and the standard deviation. For all compared methods and our method, the number of clusters is set to the true number of classes of each data set. The base clustering results used in compared consensus clustering methods are the same as ours, i.e., they also ensemble the initial $\{\mathbf{G}^{(1)}, \ldots, \mathbf{G}^{(10)}\}$. We use two popular metrics to measure the clustering performance: Accuracy (ACC) and Normalized Mutual Information (NMI).

### C. Experimental Results

Tables II and III show the ACC and NMI of all methods on all data sets. The bold font means the result is significantly different from others in the $t$-test (i.e., the $p$-value is smaller than 0.05). From these tables, we find that the proposed method performs significantly better than the average base K-means results, which demonstrates that our consensus learning can indeed improve the clustering performance. Even compared with the best base results KM-best, ours also outperforms them at most time. On most data sets, ACMK-SC performs better than SC, which means the learned graph structure can reveal a clearer clustering structure than the pre-defined one. When compared with other state-of-the-art consensus clustering methods, ours achieves a better result on most data sets. Notice that all these compared consensus clustering methods except ours use fixed base results. Our ACMK refines the base results and learns the consensus result with the refined base ones, and thus it often achieves better performance.

To further illustrate the effect of the graph filter, we show the results with different orders of graph filter, i.e., we show the results with $k = \{1, 2, 3, 4\}$. Tables IV and V show the results of ACMK-KM and ACMK-SC, respectively. From Tables IV and V, we find that ACMK often achieves good results when $k = 3$ or $k = 4$. If $k$ is too small, the graph filter cannot effectively capture the high-order relationship among instances. If $k$ is too large, it will lead to the over-smooth problem [57], [58].

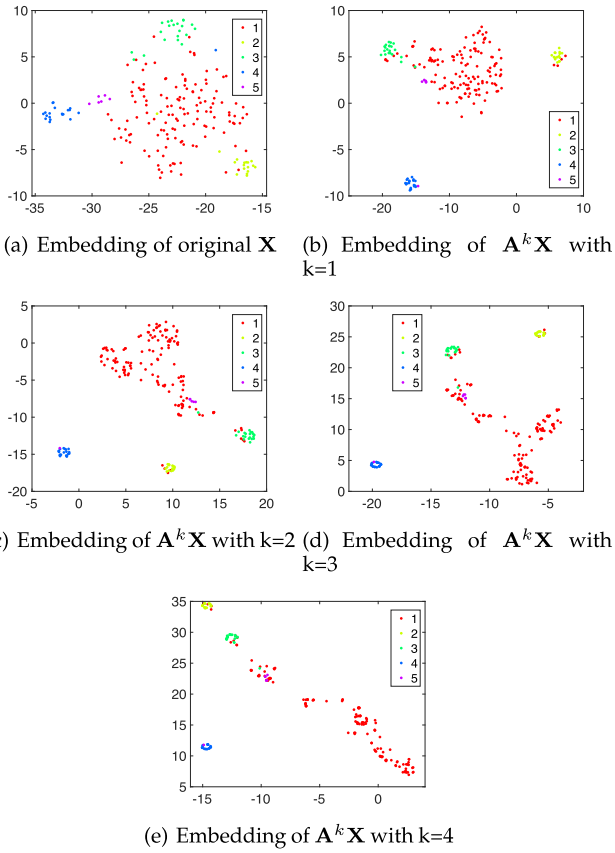
Fig. 1 shows the visualization results of t-SNE [59] on Lung data set. Fig. 1(a) is the t-SNE result of the original data $\mathbf{X}$. Figs. 1(b)–(e) are the t-SNE result of the embedding $\mathbf{A}^k\mathbf{X}$

TABLE IV
RESULTS OF ACMK-KM WITH DIFFERENT $k$

| $k$ | Metric | 20NG | Lung | Orlraws | Pendigit | Tr11 | Tr12 | Tr31 | Yale |
|---|---|---|---|---|---|---|---|---|---|
| $k=1$ | ACC | 0.2724 | 0.8665 | 0.8080 | 0.6811 | **0.6923** | 0.5843 | 0.4003 | 0.5491 |
| | NMI | 0.0059 | 0.6482 | **0.8451** | 0.6393 | 0.5842 | 0.4375 | 0.0379 | 0.5746 |
| $k=2$ | ACC | 0.2898 | 0.8759 | **0.8130** | 0.6771 | 0.6816 | 0.6419 | 0.3974 | 0.5788 |
| | NMI | 0.0108 | 0.6651 | 0.8417 | 0.6448 | 0.6394 | 0.5579 | 0.0559 | 0.5932 |
| $k=3$ | ACC | 0.2902 | **0.8783** | 0.8110 | **0.7399** | 0.6708 | **0.6786** | 0.4503 | 0.5679 |
| | NMI | 0.0123 | **0.6716** | 0.8441 | **0.6750** | **0.6559** | **0.6185** | 0.1107 | **0.5945** |
| $k=4$ | ACC | **0.3054** | 0.8207 | 0.7960 | 0.6781 | 0.6423 | 0.6553 | **0.4633** | **0.5848** |
| | NMI | **0.0193** | 0.6177 | 0.8324 | 0.6454 | 0.6443 | 0.6161 | **0.1436** | 0.5924 |

TABLE V
RESULTS OF ACMK-SC WITH DIFFERENT $k$

| $k$ | Metric | 20NG | Lung | Orlraws | Pendigit | Tr11 | Tr12 | Tr31 | Yale |
|---|---|---|---|---|---|---|---|---|---|
| $k=1$ | ACC | 0.3029 | **0.6350** | 0.8030 | 0.6847 | 0.5684 | 0.4240 | 0.4003 | 0.5394 |
| | NMI | 0.0154 | **0.4553** | 0.8396 | 0.6418 | 0.4477 | 0.2564 | 0.0379 | 0.5595 |
| $k=2$ | ACC | 0.3120 | 0.6296 | **0.8160** | 0.7061 | 0.5357 | 0.4642 | 0.3974 | 0.5515 |
| | NMI | 0.0149 | 0.3413 | 0.8429 | 0.6530 | 0.4642 | 0.3387 | 0.0559 | 0.56252 |
| $k=3$ | ACC | 0.3114 | 0.5818 | 0.8150 | **0.7788** | 0.5490 | 0.5588 | 0.4503 | 0.5697 |
| | NMI | 0.0149 | 0.3443 | **0.8438** | **0.7012** | 0.5262 | 0.4628 | 0.1107 | 0.5756 |
| $k=4$ | ACC | **0.3202** | 0.6069 | 0.7970 | 0.7578 | **0.5715** | **0.6070** | **0.4633** | **0.5733** |
| | NMI | **0.0192** | 0.4483 | 0.8290 | 0.6885 | **0.5722** | **0.5328** | **0.1436** | **0.5832** |



(a) Embedding of original $\mathbf{X}$ (b) Embedding of $\mathbf{A}^k\mathbf{X}$ with k=1

(c) Embedding of $\mathbf{A}^k\mathbf{X}$ with k=2 (d) Embedding of $\mathbf{A}^k\mathbf{X}$ with k=3

(e) Embedding of $\mathbf{A}^k\mathbf{X}$ with k=4

Fig. 1. T-SNE results on lung.

with $k = 1, 2, 3, 4$, respectively. We can see that the data in Fig. 1(a) are entangled together, and with the graph filter, the data are easily separated. It shows that the graph filter can lead to a cluster-friendly embedding, which meets the cluster and manifold assumption, i.e., the data in the same cluster should be close to each other and the data in different clusters should be far apart from each other.

To show the effectiveness of the refinement quantitatively, we compare the ACC of each base clustering result before and after refinement. The NMI results are similar. Fig. 2 shows the results. The blue bars represent the ACC results of 10 original base clustering (i.e., directly obtained from the initial $\mathbf{G}^{(i)}$); and the red bars represent the ACC of refined base clusterings (i.e., directly obtained from the learned $\mathbf{G}^{(i)}$). From Fig. 2, we can see that, on most data sets, our refinement can improve most base results more or less. It demonstrates that our graph filter method can indeed improve the performance of base clusterings, and further with better base results it can obtain a better consensus clustering result.

Fig. 3 shows the convergence curves of ACMK on Lung, Orlraws, Tr31, and Yale data sets. The results on other data sets are similar. It can be seen that ACMK can converge very fast, i.e., it often converges within 20 iterations.

### D. Ablation Study

For an ablation study, we compare with some degenerated versions of ACMK to show the effect of the consensus learning and base result refining. To show the effect of consensus learning, we compare with **ACMK-GF** without consensus learning. In more detail, we run K-means directly on $((\mathbf{I}+\mathbf{D}^{-\frac{1}{2}}\mathbf{W}\mathbf{D}^{-\frac{1}{2}})/2)^k\mathbf{X}$, where $\mathbf{W}=\mathbf{G}^{(m)}\mathbf{G}^{(m)T}$ constructed directly from the base results without the consensus learning. ACMK-GF reports the average results among the 10 base results. To show the effectiveness of the base result refining, we compare with **ACMK-Fix**, which does not use the graph filter and directly applies spectral clustering on $\mathbf{W}=\frac{1}{v}\sum_{m=1}^{v}\mathbf{G}^{(m)}\mathbf{G}^{(m)T}$ with fixed base results $\mathbf{G}^{(m)}$s.

Since ACMK-GF is the K-means based method, we compare it with KM-avg and ACMK-KM. Table VI shows the ACC and NMI results. From Table VI, we can see that ACMK-GF is often

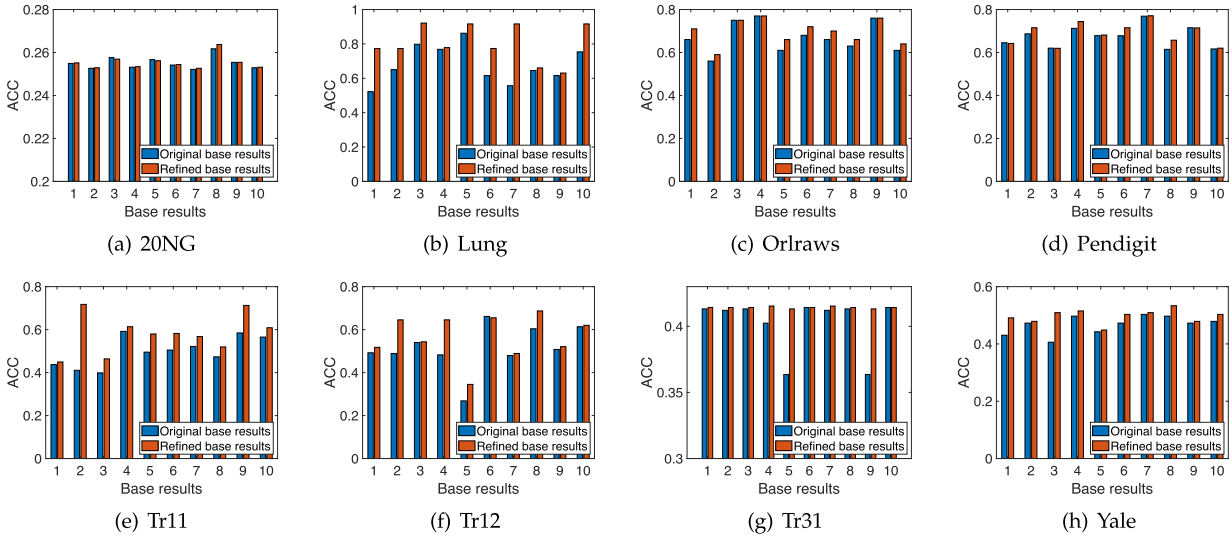



Fig. 2. ACC results of each base result before and after refinement.

TABLE VI
ABLATION STUDY: COMPARISON OF KM-AVG, ACMK-GF, AND ACMK-KM

| $k$ | Metric | 20NG | Lung | Orlraws | Pendigit | Tr11 | Tr12 | Tr31 | Yale |
|---|---|---|---|---|---|---|---|---|---|
| KM-avg | ACC | 0.2577 | 0.7001 | 0.6709 | 0.6846 | 0.5176 | 0.4772 | 0.3982 | 0.4655 |
| | NMI | 0.0089 | 0.5161 | 0.7426 | 0.6623 | 0.4726 | 0.3931 | 0.0610 | 0.5133 |
| ACMK-GF | ACC | 0.2675 | 0.5701 | 0.6453 | 0.7080 | 0.5156 | 0.4634 | 0.4109 | 0.5077 |
| | NMI | **0.0187** | 0.4010 | 0.7169 | 0.6648 | 0.4992 | 0.3946 | 0.0988 | 0.5524 |
| ACMK-KM | ACC | **0.2902** | **0.8783** | **0.8110** | **0.7399** | **0.6708** | **0.6786** | **0.4503** | **0.5679** |
| | NMI | 0.0123 | **0.6716** | **0.8441** | **0.6750** | **0.6559** | **0.6185** | **0.1107** | **0.5945** |

TABLE VII
ABLATION STUDY: COMPARISON OF SC, ACMK-FIX, AND ACMK-SC

| $k$ | Metric | 20NG | Lung | Orlraws | Pendigit | Tr11 | Tr12 | Tr31 | Yale |
|---|---|---|---|---|---|---|---|---|---|
| SC | ACC | 0.2525 | 0.5271 | 0.6850 | 0.7340 | **0.6500** | 0.4246 | 0.4379 | 0.5430 |
| | NMI | 0.0031 | 0.4106 | 0.7457 | **0.7368** | **0.6010** | 0.3711 | **0.1166** | 0.5562 |
| ACMK-Fix | ACC | 0.2717 | 0.5576 | 0.7800 | 0.7083 | 0.5529 | 0.5019 | 0.3746 | 0.5558 |
| | NMI | **0.0201** | **0.4111** | 0.8179 | 0.6672 | 0.5668 | 0.4459 | 0.0668 | 0.5642 |
| ACMK-SC | ACC | **0.3114** | **0.5818** | **0.8150** | **0.7788** | 0.5490 | **0.5588** | **0.4503** | **0.5697** |
| | NMI | 0.0149 | 0.3443 | **0.8438** | 0.7012 | 0.5262 | **0.4628** | 0.1107 | **0.5756** |

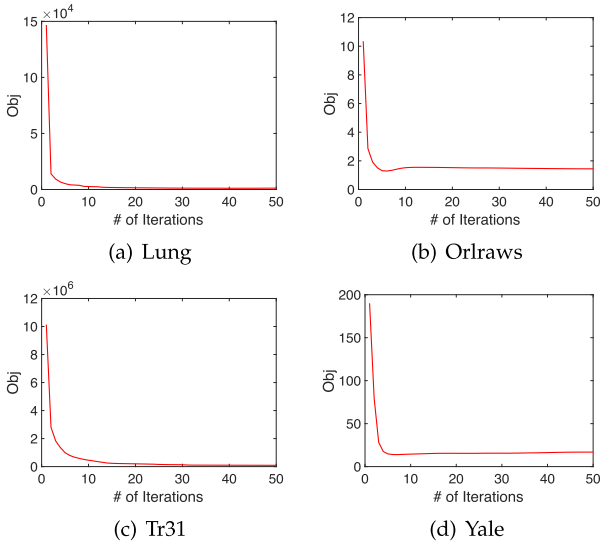



Fig. 3. Convergence curves.

better than KM-avg, which means with the graph filter, we obtain a more cluster-friendly embedding. ACMK-KM can further improve ACMK-GF, which demonstrates the effectiveness of consensus learning.

ACMK-Fix is a spectral clustering based method, and thus we compare it with SC and ACMK-SC in Table VII. ACMK-Fix often outperforms SC, which means the fixed graph constructed via consensus learning is more appropriate than the pre-defined graph used in SC. When compared with ACMK-SC, we find that ACMK-SC often achieves better performance, which means by refining the base results can further improve the performance of consensus learning.

### E. Parameter Study

In this subsection, we show the effect of the hyper-parameter $\lambda$. In Fig. 4, we show the clustering results of ACMK-KM and ACMK-SC with different values of $\lambda$ on all data sets. It can be

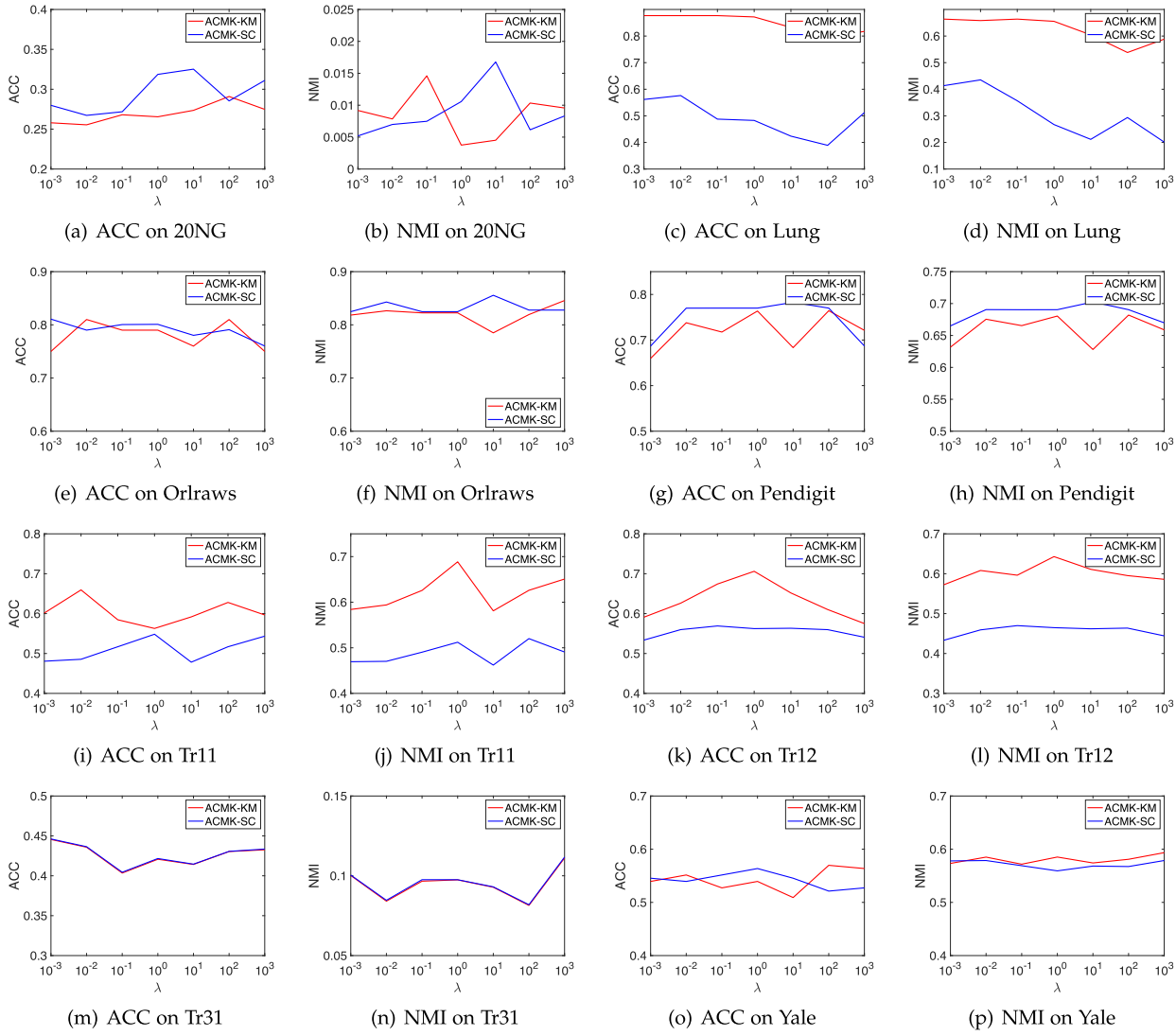



Fig. 4. Clustering results of ACMK-KM and ACMK-SC w.r.t. λ.

seen that the results are relatively insensitive to λ, and we often achieve a good result when λ is in the range $[10^{-2}, 10^{2}]$.

## V. Conclusion

In this paper, we provided another idea for consensus clustering to improve the clustering performance, i.e., we adaptively improved the quality of each base result in the process of consensus learning. To fulfill this idea, we proposed a novel adaptive consensus multiple K-means. In this method, to adaptively improve the quality of base results, we applied a learnable graph filter to learn a smoother and more cluster-friendly embedding for the original data and updated the base results by running K-means on such embedding. Meanwhile, we ensembled multiple base results to construct a consensus graph for the graph filter by considering the consensus and diversity. The proposed method integrated such two processes into a unified framework and obtained the final consensus clustering result from the consensus graph. The experimental results on benchmark data sets shew the effectiveness and superiority of the proposed method by comparing it with other state-of-the-art consensus clustering methods.

Although the proposed method achieves promising performance, since it performs a learnable graph filter, the time complexity is with the square of the number of instances. In the future, we will further study the scalable issue to design a lightweight learnable graph filter. Moreover, as pointed out by one of the anonymous reviewers, one possible improvement of this work is that we can use a more powerful polynomial graph filter $\mathbf{A} + \mathbf{A}^2 + \cdots + \mathbf{A}^k$ to refine the base clustering.

## Acknowledgments

We also would like to thank the anonymous reviewers for their constructive comments and suggestions.

## REFERENCES


[1] F. Wang, X. Wang, and T. Li, "Generalized cluster aggregation," in *Proc. 21st Int. Joint Conf. Artif. Intell.*, Pasadena, California, USA, 2009, pp. 1279–1284.

[2] L. Bai, J. Liang, and F. Cao, "A multiple k-means clustering ensemble algorithm to find nonlinearly separable clusters," *Inf. Fusion*, vol. 61, pp. 36–47, 2020.

[3] A. Strehl and J. Ghosh, "Cluster ensembles – a knowledge reuse framework for combining multiple partitions," *J. Mach. Learn. Res.*, vol. 3, no. 3, pp. 583–617, 2003.

[4] H. Liu, T. Liu, J. Wu, D. Tao, and Y. Fu, "Spectral ensemble clustering," in *Proc. 21th ACM SIGKDD Int. Conf. Knowl. Discov. Data Mining*, 2015, pp. 715–724.

[5] D. Huang, J. Lai, and C. Wang, "Robust ensemble clustering using probability trajectories," *IEEE Trans. Knowl. Data Eng.*, vol. 28, no. 5, pp. 1312–1326, May 2016.

[6] Z. Tao, H. Liu, and Y. Fu, "Simultaneous clustering and ensemble," in *Proc. AAAI Conf. Artif. Intell.*, 2017, pp. 1546–1552.

[7] S. Abbasi, S. Nejatian, H. Parvin, V. Rezaie, and K. Bagherifard, "Clustering ensemble selection considering quality and diversity," *Artif. Intell. Rev.*, vol. 52, no. 2, pp. 1311–1340, 2019.

[8] A. Bagherinia, B. Minaei-Bidgoli, M. Hossinzadeh, and H. Parvin, "Elite fuzzy clustering ensemble based on clustering diversity and quality measures," *Appl. Intell.*, vol. 49, no. 5, pp. 1724–1747, 2019.

[9] S. Wang et al., "Multi-view clustering via late fusion alignment maximization," in *Proc. 28th Int. Joint Conf. Artif. Intell.*, 2019, pp. 3778–3784.

[10] P. Zhou, L. Du, X. Liu, Y. Shen, M. Fan, and X. Li, "Self-paced clustering ensemble," *IEEE Trans. Neural Netw. Learn. Syst.*, vol. 32, no. 4, pp. 1497–1511, Apr. 2021.

[11] Z. Lin and Z. Kang, "Graph filter-based multi-view attributed graph clustering," in *Proc. 30th Int. Joint Conf. Artif. Intell.*, Montreal, Canada, 2021, 2021, pp. 2723–2729.

[12] P. Chen, L. Liu, Z. Ma, and Z. Kang, "Smoothed multi-view subspace clustering," in *Neural Computing for Advanced Applications*, H. Zhang, Z. Yang, Z. Zhang, Z. Wu, and T. Hao Eds., Singapore: Springer, 2021, pp. 128–140.

[13] S. Boyd et al., "Distributed optimization and statistical learning via the alternating direction method of multipliers," *Foundations Trends Mach. Learn.*, vol. 3, no. 1, pp. 1–122, 2011.

[14] J. MacQueen et al., "Some methods for classification and analysis of multivariate observations," in *Proc. 5th Berkeley Symp. Math. Statist. Probability*, 1967, pp. 281–297.

[15] P. Zhou, L. Du, H. Wang, L. Shi, and Y.-D. Shen, "Learning a robust consensus matrix for clustering ensemble via Kullback-Leibler divergence minimization," in *Proc. 24th Int. Joint Conf. Artif. Intell.*, 2015, pp. 4112–4118.

[16] H. Xiong, J. Wu, and J. Chen, "K-means clustering versus validation measures: A data-distribution perspective," *IEEE Trans. Systems, Man, Cybern. B, Cybern.*, vol. 39, no. 2, pp. 318–331, Apr. 2009.

[17] Y. Wang, W. Zhang, L. Wu, X. Lin, M. Fang, and S. Pan, "Iterative views agreement: An iterative low-rank based structured optimization method to multi-view spectral clustering," in *Proc. 25th Int. Joint Conf. Artif. Intell.*, New York, NY, USA, 2016, pp. 2153–2159.

[18] Y. Wang, L. Wu, X. Lin, and J. Gao, "Multiview spectral clustering via structured low-rank matrix factorization," *IEEE Trans. Neural Netw. Learn. Syst.*, vol. 29, no. 10, pp. 4833–4843, Oct. 2018.

[19] P. Zhou, Y.-D. Shen, L. Du, F. Ye, and X. Li, "Incremental multi-view spectral clustering," *Knowl.-Based Syst.*, vol. 174, pp. 73–86, 2019.

[20] X. Liu et al., "Efficient and effective regularized incomplete multi-view clustering," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 43, no. 8, pp. 2634–2646, Aug. 2021.

[21] L. Wu et al., "Pseudo-pair based self-similarity learning for unsupervised person re-identification," *IEEE Trans. Image Process.*, vol. 31, pp. 4803–4816, 2022.

[22] Y. L. Wu, D. Liu, X. Guo, R. Hong, L. Liu, and R. Zhang, "Multi-scale spatial representation learning via recursive hermite polynomial networks," in *Proc. 31t Int. Joint Conf. Artif. Intell.*, Vienna, Austria, 2022, pp. 1465–1473.

[23] A. P. Topchy, A. K. Jain, and W. F. Punch, "Clustering ensembles: Models of consensus and weak partitions," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 27, no. 12, pp. 1866–1881, Dec. 2005.

[24] N. Nguyen and R. Caruana, "Consensus clusterings," in *Proc. IEEE 7th Int. Conf. Data Mining*, Omaha, NE, USA, 2007, pp. 607–612.

[25] L. Bai, J. Liang, H. Du, and Y. Guo, "An information-theoretical framework for cluster ensemble," *IEEE Trans. Knowl. Data Eng.*, vol. 31, no. 8, pp. 1464–1477, Aug. 2019.

[26] Z. Zhou and W. Tang, "Clusterer ensemble," *Knowl. Based Syst.*, vol. 19, no. 1, pp. 77–83, 2006.

[27] P. Hore, L. O. Hall, and D. B. Goldgof, "A scalable framework for cluster ensembles," *Pattern Recognit.*, vol. 42, no. 5, pp. 676–688, 2009.

[28] F. Li, Y. Qian, J. Wang, and J. Liang, "Multigranulation information fusion: A dempster-shafer evidence theory-based clustering ensemble method," *Inf. Sci.*, vol. 378, pp. 389–409, 2017.

[29] Z. Tao, H. Liu, S. Li, Z. Ding, and Y. Fu, "Robust spectral ensemble clustering via rank minimization," *ACM Trans. Knowl. Discov. From Data*, vol. 13, no. 1, pp. 1–25, 2019.

[30] D. Huang, C. Wang, J. Wu, J. Lai, and C. Kwoh, "Ultra-scalable spectral clustering and ensemble clustering," *IEEE Trans. Knowl. Data Eng.*, vol. 32, no. 6, pp. 1212–1226, Jun. 2020.

[31] P. Zhou, L. Du, and X. Li, "Self-paced consensus clustering with bipartite graph," in *Proc. 29th Int. Joint Conf. Artif. Intell.*, 2020, pp. 2133–2139.

[32] P. Zhou, L. Du, Y.-D. Shen, and X. Li, "Tri-level robust clustering ensemble with multiple graph learning," in *Proc. 35th AAAI Conf. Artif. Intell.*, 2021, pp. 11 125–11 133.

[33] P. Zhou, X. Wang, L. Du, and X. Li, "Clustering ensemble via structured hypergraph learning," *Inf. Fusion*, vol. 78, pp. 171–179, 2022.

[34] P. Zhou, X. Liu, L. Du, and X. Li, "Self-paced adaptive bipartite graph learning for consensus clustering," *ACM Trans. Knowl. Discov. Data*, vol. 17, no. 5, 2022, Art. no. 62.

[35] X. Z. Fern and C. E. Brodley, "Solving cluster ensemble problems by bipartite graph partitioning," in *Proc. 21st Int. Conf. Mach. Learn.*, 2004, p. 36.

[36] N. Iam-On, T. Boongoen, S. M. Garrett, and C. J. Price, "A link-based approach to the cluster ensemble problem," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 33, no. 12, pp. 2396–2409, Dec. 2011.

[37] N. Iam-On, T. Boongoen, S. M. Garrett, and C. J. Price, "A link-based cluster ensemble approach for categorical data clustering," *IEEE Trans. Knowl. Data Eng.*, vol. 24, no. 3, pp. 413–425, Mar. 2012.

[38] D. I. Shuman, S. K. Narang, P. Frossard, A. Ortega, and P. Vandergheynst, "The emerging field of signal processing on graphs: Extending high-dimensional data analysis to networks and other irregular domains," *IEEE Signal Process. Mag.*, vol. 30, no. 3, pp. 83–98, May 2013.

[39] Z. Ma, Z. Kang, G. Luo, L. Tian, and W. Chen, "Towards clustering-friendly representations: Subspace clustering via graph filtering," in *Proc. 28th ACM Int. Conf. Multimedia*, Seattle, WA, USA, 2020, pp. 3081–3089.

[40] E. Pan and Z. Kang, "Multi-view contrastive graph clustering," in *Proc. Int. Conf. Neural Inf. Process. Syst.*, 2021, pp. 2148–2159.

[41] X. Zhang, H. Liu, Q. Li, and X. Wu, "Attributed graph clustering via adaptive graph convolution," in *Proc. 28th Int. Joint Conf. Artif. Intell.*, Macao, China, 2019, pp. 4327–4333.

[42] D. C. Liu and J. Nocedal, "On the limited memory BFGS method for large scale optimization," *Math. Prog.*, vol. 45, no. 1/3, pp. 503–528, 1989.

[43] J. Klicpera, S. Weißenberger, and S. Günnemann, "Diffusion improves graph learning," in *Proc. Annu. Conf. Neural Inf. Process. Syst.*, 2019, pp. 13 333–13 345.

[44] H. Wai, S. Segarra, A. E. Ozdaglar, A. Scaglione, and A. Jadbabaie, "Blind community detection from low-rank excitations of a graph filter," *IEEE Trans. Signal Process.*, vol. 68, pp. 436–451, 2020.

[45] X. Dong, D. Thanou, P. Frossard, and P. Vandergheynst, "Learning laplacian matrix in smooth graph signal representations," *IEEE Trans. Signal Process.*, vol. 64, no. 23, pp. 6160–6173, Dec. 2016.

[46] D. Zhou and B. Schölkopf, "A regularization framework for learning from graph data," in *Proc. ICML Workshop Statist. Relational Learn.*, 2004, pp. 132–137.

[47] D. Cai, X. He, W. V. Zhang, and J. Han, "Regularized locality preserving indexing via spectral regression," in *Proc. 16th ACM Conf. Inf. Knowl. Manage.*, 2007. pp. 741–750.

[48] Z. Hong and J. Yang, "Optimal discriminant plane for a small number of samples and design method of classifier on the plane," *Pattern Recognit.*, vol. 24, no. 4, pp. 317–324, 1991.

[49] F. Samaria and A. Harter, "Parameterisation of a stochastic model for human face identification," in *Proc. IEEE 2nd Workshop Appl. Comput. Vis.*, 1994, pp. 138–142.

[50] F. Alimoglu and E. Alpaydin, "Methods of combining multiple classifiers based on different representations for pen-based handwritten digit recognition," in *Proc. 5th Intell. Artif. Neural Netw. Symp.*, 1996, pp. 418–435.

[51] J. Cao, Z. Wu, J. Wu, and H. Xiong, "SAIL: Summation-based incremental learning for information-theoretic text clustering," *IEEE Trans. Cybern.*, vol. 43, no. 2, pp. 570–584, Apr. 2013.
[52] D. Cai, X. He, Y. Hu, J. Han, and T. Huang, "Learning a spatially smooth subspace for face recognition," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit. Mach. Learn.*, 2007, pp. 1–7.
[53] J. Shi and J. Malik, "Normalized cuts and image segmentation," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 22, no. 8, pp. 888–905, Aug. 2000.
[54] T. Li and C. H. Q. Ding, "Weighted consensus clustering," in *Proc. SIAM Int. Conf. Data Mining*, Atlanta, GA, USA, 2008, pp. 798–809.
[55] D. Huang, C. Wang, and J. Lai, "Locally weighted ensemble clustering," *IEEE Trans. Cybern.*, vol. 48, no. 5, pp. 1460–1473, May 2018.
[56] J. Zhou, H. Zheng, and L. Pan, "Ensemble clustering based on dense representation," *Neurocomputing*, vol. 357, pp. 66–76, 2019.
[57] Q. Li, Z. Han, and X. Wu, "Deeper insights into graph convolutional networks for semi-supervised learning," in *Proc. 32nd AAAI Conf. Artif. Intell.*, 2018, pp. 3538–3545.
[58] G. Li, M. Muller, A. Thabet, and B. Ghanem, "DeepGCNs: Can GCNs go as deep as CNNs?," in *Proc. IEEE/CVF Int. Conf. Comput. Vis.*, 2019, pp. 9267–9276.
[59] L. van der Maaten and G. Hinton, "Visualizing data using t-SNE," *J. Mach. Learn. Res.*, vol. 9, no. 86, pp. 2579–2605, 2008.

**Peng Zhou** received the BE degree in computer science and technology from the University of Science and Technology of China, in 2011, and the PhD degree in computer science from the Institute of Software, Chinese Academy of Sciences, in 2017. He is currently an associate professor with the School of Computer Science and Technology, Anhui University. He has published more than 30 papers in highly regarded conferences and journals, including *IEEE Transactions on Knowledge and Data Engineering*, *IEEE Transactions on Neural Networks and Learning Systems*, *IEEE Transactions on Cybernetics*, *ACM Transactions on Knowledge Discovery from Data*, IJCAI, AAAI, SDM, ICDM, etc. His research interests include machine learning, data mining, and artificial intelligence. More publications and codes can be found in his homepage: https://doctor-nobody.github.io/.

**Liang Du** received the BE degree in software engineering from Wuhan University, in 2007, and the PhD degree in computer science from the Institute of Software, University of Chinese Academy of Sciences, in 2013. From July 2013 to July 2014, he was a software engineer with Alibaba Group. He was also an assistant researcher with the State Key Laboratory of Computer Science, Institute of Software, Chinese Academy of Sciences. He is currently an associate professor with Shanxi University. He has published more than 40 papers in top conferences and journals, including KDD, IJCAI, AAAI, ICDM, *IEEE Transactions on Knowledge and Data Engineering*, SDM, and CIKM. His research interests include clustering with noise and heterogeneous data, ranking for feature selection, active learning, and document summarization.

**Xuejun Li** received the PhD degree from Anhui University, in 2008. He is currently a professor with the School of Computer Science and Technology, Anhui University, China. His major research interests include workflow systems, cloud computing, and intelligent software.