CLUSTERING STRUCTURE IDENTIFICATION WITH ORDERING GRAPH

Anonymous authors
Paper under double-blind review

ABSTRACT

In machine learning, data is often presented in the form of a graph or similarity (or distance) values between samples. Graph construction research has developed significantly for decades, but the graph-based partition study still requires more attention because of its poor performance. For example, spectral clustering needs a post-processing (e.g., K-Means) step to uncover the clustering indicators. However, K-Means is sensitive to the initial center setting and easily falls into a local optimum. In this paper, we investigate an integrated density and graph clustering approach. Firstly, we introduce a matrix ordering algorithm, namely DBGO, for cluster analysis that does not explicitly produce a clustering of a dataset but instead creates an augmented graph representing its density-based ordered clustering structure. Secondly, we find that the graph matrix is shown in a block-diagonal form because of the nature of ordered clusters. We proposed a partition-based graph-cut clustering model to learn the block-diagonal structure of the graph matrix and identify the clustering structure simultaneously. If the between- and within-cluster weight of an ordered graph follow Gaussian distribution, then the true cluster segmentation will maximize the expectation of the graph-cut function, and our proposed BDSL algorithm can converge to the true cluster segmentation by maximizing the expectation of the graph-cut function, which has been guaranteed theoretically. We test the proposed method on synthetic datasets and five high-dimensional real datasets. Experimental results show that the proposed method outperforms state-of-the-art clustering methods and improves the performance of subspace clustering approaches by roughly 10%$\sim$50%.

1 INTRODUCTION

Larger and larger amounts of data are collected and stored in databases, increasing the need for efficient and effective analysis methods to implicitly use the information in the data. One of the primary data analysis tasks is cluster analysis which is intended to help a user to understand the natural grouping or structure in a dataset. Therefore, improved clustering algorithms have received much attention in the last few years. The goal of a clustering algorithm is to group the objects of a database into a set of meaningful subclasses. A clustering algorithm can be used either as a standalone tool to get insight into the distribution of a dataset, e.g., to focus further analysis and data processing or as a preprocessing step for other algorithms that operate on the detected clusters.

Many recent works pay attention to the effectivity of clustering, i.e., the quality or usefulness of the clustering result. However, their characteristic of effectiveness is still unsatisfactory. There are three interconnected reasons: 1) Almost all clustering algorithms require values for input parameters that are hard to determine, especially for real-world datasets containing high-dimensional objects. 2) The algorithms are susceptible to these parameter values, often producing very different segmentations of the dataset, even for slightly different parameter settings. 3) High-dimensional real datasets usually have a skewed distribution that can not be revealed by a clustering algorithm using only one global parameter setting.

State-of-the-art clustering methods based on graphical representations of the relationships among data points have been widely used for decades. For example, spectral clustering Von Luxburg (2007), normalized cut Shi & Malik (2000), and radio cut Hagen & Kahng (1992) all transform the data into a weighted, undirected graph based on pairwise similarities. Clustering is then accomplished by spectral or graph-theoretic optimization procedures. See Ding et al. (2005; 2006) for a discussion of the relations among these graph-based methods and also the connections to nonnegative matrix factorization. All of these methods involve a two-stage process in which a data graph is formed from the data, and then various optimization procedures are invoked on this fixed input data graph. On the one hand, the clustering results depend on the quality of the input data graph (i.e., they are sensitive to the particular graph construction methods). Thus, almost all researchers focus on the graph construction task, including good neighbor pruning Yang et al. (2019); Cui et al. (2021), multi-view affinity learning Xiao et al. (2020); Tang et al. (2018); Kang et al. (2020; 2021), sparse representation learning Elhamifar & Vidal (2013b), low-rank representation learning Xu et al. (2021), self-expression training based on neural network Lv et al. (2021); Cai et al. (2021); Zhang et al. (2021a). However, the final clustering structures are not represented explicitly in the data graph (e.g., graph-cut methods often use the K-Means algorithm to post-process the results to get the clustering indicators). More and more researchers have discovered that it is not the graph construction but the graph partition that limits improving clustering accuracy. Although a lot of graph partition approaches have been proposed recently, e.g., MST Şaar & Topcu (2022), PrimZhu & Chen (2021), Kruskal Battaglia & Pensa (2021), SNN Liu et al. (2022), Jarvis-Patrick Wang et al. (2022), Extended-DBSCAN Chen et al. (2021), Chameleon Zhang et al. (2021b), and Extended-Spectral Sharma & Seal (2021). However, their characteristics of effectiveness and robustness on real datasets are still unsatisfactory. A clustering technique is considered to be good if it satisfies the following requirements Fahim et al. (2008): a) Minimal requirements of the domain knowledge to determine the values of its input parameters, which is a significant problem, especially for large datasets. b) Discovery of arbitrarily shaped clusters. c) Good efficiency on large datasets. It is still challenging to design a clustering method that has the above three features simultaneously.

In this paper, we introduce an effective and robust approach for the purpose of cluster analysis which does not produce a clustering explicitly; but instead permutes a given graph representing the density-based clustering structure of the data. This ordered graph contains information similar to the density-based clusterings corresponding to a broad range of parameter settings. It is a versatile basis for both automatic and interactive cluster analysis. We proposed an ordering algorithm to find ordered density-based clusters in spatial data and detect meaningful clusters in data of varying densities. The graph weight matrix is shown in block-diagonal form because of the nature of ordered clusters. Thus, we proposed a novel partition-based normalized cut clustering model that identifies the clustering structure from the ordered similarity matrix with a K-block-diagonal form, where K is the number of clusters. In order to identify the clustering structure directly without requiring any post-processing, we impose an efficient binary segmentation algorithm to learn the proposed graph partition model. The ordered graph will be cut into a K-block-diagonal form by $K-1$ partition indexes. If the between- and within-cluster weight follows Gaussian distribution, then the true cluster segmentation will maximize the expectation of the objective function, and our algorithm can converge to the true cluster partition from an expectation view theoretically. We conduct empirical studies on four simulated and five real-world benchmark datasets to validate our proposed methods. Our integrated density and graph clustering method consistently outperforms other related methods in general low-dimensional clustering tasks. For the high-dimensional clustering task, the performance of conventional subspace clustering methods can be improved by roughly 10%$\sim$50% by replacing spectral clustering with our proposed method.

2 Review of Graph Construction and Partition

The graph-based method includes three phases: graph construction, graph refinement (e.g., prune and generate), and graph partition. Since the graph refinement method is complicated and various, we only introduce the graph construction and the graph partition technique.

2.1 Graph Construction

The general graph is often given by the distance measurement between points. The usage of distance depends on the property of the data. The traditional distance measurements include Euclidean, Mahalanobis, Cityblock, Minkowski, Chebychev, Cosine, Correlation, Hamming, Jaccard, and Spearman Choi et al. (2010). For example, the work Menon et al. (2020) supposes the data in each cluster are located in a high-dimensional subspace and use the cosine angle between points to measure the similarity of the samples in the same or different clusters. Specifically, the point $\mathbf{x}_i$ is normalized by $\bar{\mathbf{x}}_i = \frac{\mathbf{x}_i}{||\mathbf{x}_i||_2}$ firstly, where $||\cdot||$ denotes the l_2 norm. These points $\bar{\mathbf{x}}_i \in \mathbb{R}^D, \forall\, i \in \{1, 2, ..., N\}$ will lie in the high dimensional hypersphere $\mathbb{S}^{D-1}$. Let θ_{ij} denote the angle between two data points $\mathbf{x}_i$ and $\mathbf{x}_j$, i.e., $\theta_{ij} = \cos^{-1}(\bar{\mathbf{x}}_i^{\mathrm{T}}\bar{\mathbf{x}}_j), \quad \theta_{ij} \in [0, \pi]$. We convert the radian θ_{ij} to its degree form $\tilde{\theta}_{ij}$, i.e., $\tilde{\theta}_{ij} = \theta_{ij}180/\pi$, $\tilde{\theta}_{ij} \in [0, 180]$. To make similar features more prominent, people usually perform the exponential normalization on the original measurement, and the closer the distance, the larger the value. The weight is given by $w_{ij} = \exp(-\tilde{\theta}_{ij}^2/\sigma^2)$, where $\sigma^2 = \mathrm{Var}[\tilde{\theta}_{ij}]$, which is a parameter to control the attenuation degree with the increasing of the difference, and we usually set it as the variance of the weight directly for convenience in this paper.

Another type of method of constructing the similarity graph between points is mainly used for the data located in high-dimensional subspaces. These works use the self-expression property to construct an optimization problem based on the sparse feature Elhamifar & Vidal (2013b), block-diagonal constraint Lu et al. (2018), or low-rank property Vidal & Favaro (2014). Based on these models, deep neural network Ji et al. (2017), multi-view combination Elhamifar & Vidal (2013b), and neighbor pruning Yang et al. (2019); Cui et al. (2021) are used to improve the self-representation performance of the similarity matrix. For example, consider a matrix of column-wise samples $\mathbf{X} = [\mathbf{x}_1, \mathbf{x}_2, ..., \mathbf{x}_N] \in \mathbb{R}^{D\times N}$. Each sample or datum can be represented by a linear combination of n atoms in a dictionary $\mathbf{A} = [\mathbf{a}_1, \mathbf{a}_2, ..., \mathbf{a}_n] \in \mathbb{R}^{D\times n}$: $\mathbf{X} = \mathbf{AZ}$, where $\mathbf{Z} = [\mathbf{z}_1, \mathbf{z}_2, ..., \mathbf{z}_N] \in \mathbb{R}^{n\times N}$ is a coefficient matrix. Under this representation, each column $\mathbf{z}_i$ of the coefficient matrix $\mathbf{Z}$ can be interpreted as a new representation for each data sample $\mathbf{x}_i$ under the dictionary. Suppose the column vectors of $\mathbf{X}$ are drawn from a union of K subspaces $\{Q_k\}_{k=1}^K$ of unknown dimensions $\{d_k\}_{k=1}^K$. Without knowing the dictionary $\mathbf{A}$, one can use the self-expressiveness property of data to find the subspaces: each data point in a union of subspaces can be efficiently reconstructed by a combination of other points in the data. For instance, the work Elhamifar & Vidal (2013a) uses the data samples themselves as the dictionary, i.e., $\mathbf{A} = \mathbf{X}$. In this case, the coefficient matrix $\mathbf{Z}$ becomes a square matrix of size $N \times N$, i.e., $\mathbf{X} = \mathbf{XZ}$. In general, $\mathbf{N} > \mathbf{D}$, in this unrestricted case, there are near infinite possibilities for the coefficient matrix $\mathbf{Z}$. The choice of $\mathbf{Z}$ is the main difference among subspace clustering techniques. The work Elhamifar & Vidal (2013a) aims to find the sparsest representation using l_1 approximation. More specifically, every point in the data is a set of sparse linear combinations of other points from the same subspace. Mathematically the sparse formulation is written as

$$\begin{aligned} \underset{\mathbf{E},\mathbf{Z}}{\text{minimize}} \quad & \lambda||\mathbf{E}||_F^2 + ||\mathbf{Z}||_1 \\ \text{subject to} \quad & \mathbf{X} = \mathbf{XZ} + \mathbf{E}, \mathrm{diag}(\mathbf{Z}) = \mathbf{0} \end{aligned} \tag{1}$$

where $\mathbf{E}$ is the noise matrix. From these sparse representations, an affinity matrix $\mathbf{Z}$ is compiled. The problem can be solved by ADMM effectively Elhamifar & Vidal (2013a). In general, the graph is given by $\mathbf{W} = \frac{|\mathbf{Z}|+|\mathbf{Z}|^{\mathrm{T}}}{2}$ because of the symmetric and positive properties of the graph.

2.2 Graph Partition

Graph-based clustering approaches typically optimize their objectives based on a given data graph associated with a symmetric affinity matrix $\mathbf{W} \in \mathbb{R}^{N\times N}$, where N is the number of nodes (data points) in the graph. A weighted undirected complete graph $\mathcal{G}$ (i.e., fully connected graph) can be written as an ordered pair $\mathcal{G} := (\mathcal{V}, \mathcal{E}, \mathbf{W})$, where $\mathcal{V}$ is a set of vertices or nodes, $\mathcal{E}$ is a set of pairs (unordered) of distinct vertices, called edges or lines, and $\mathbf{W}$ is an $N \times N$ adjacency matrix (also similarity matrix or weight matrix) of the finite undirected graph $\mathcal{G}$ on N vertices, where the nondiagonal entry w_{ij} is the number of edges from vertex i to vertex j, and the diagonal entry w_{ii} is either twice the number of loops at vertex i or just the number of loops (usages differ, depending on the mathematical needs; this paper is not concerned with reflexive connections). Recall the symmetric and positive properties of the graph. We assume $w_{ij} = w_{ji} \geq 0$.

Given $\mathcal{A} \subset \mathcal{V}$, its complement, $\mathcal{V}\backslash\mathcal{A}$ will be denoted as $\bar{\mathcal{A}}$. Intuitively, the subset $\mathcal{A}$ is connected if paths between any two points in $\mathcal{A}$ need only points in $\mathcal{A}$. $\mathcal{A}$ is called a connected component with respect to $\bar{\mathcal{A}}$ if $\mathcal{A}$ is connected, and there are no edges between vertices in $\mathcal{A}$ and $\bar{\mathcal{A}}$. Subsets $\mathcal{A}_1, \mathcal{A}_2, ..., \mathcal{A}_K$ represent a partition of $\mathcal{V}$, if, for all $i, j \in [1, 2, ..., K]$, $\mathcal{A}_i \cap \mathcal{A}_j = \emptyset$, and $\cup_{k=1}^K \mathcal{A}_k = \mathcal{V}$. Generally speaking, clustering means partitioning a graph so that the edges between different groups have low similarity (low distance) and the edges within a group have high weight (low distance). The first requirement for the partition can be stated as the mini cut criterion, which has to be minimized: $cut(\mathcal{A}_1, \mathcal{A}_2, ..., \mathcal{A}_K) = \sum_{k=1}^{K} cut(\mathcal{A}_k, \bar{\mathcal{A}}_k)$ where $cut(\mathcal{A}_k, \bar{\mathcal{A}}_k) = \sum_{i \in \mathcal{A}_k, j \in \bar{\mathcal{A}}_k} w_{ij}$. Following the mini cut requirement, often only a little group of points is isolated. For this reason, the famous cutting methods, which are widely used in spectral clustering, are introduced: $Ncut(\mathcal{A}_1, \mathcal{A}_2, ..., \mathcal{A}_K) = \sum_{k=1}^{K} \frac{cut(\mathcal{A}_k, \bar{\mathcal{A}}_k)}{vol(\mathcal{A}_k)}$ Shi & Malik (2000) where $vol(\mathcal{A})$ measures the size of $\mathcal{A}$ by the weights of its edges, i.e. $vol(\mathcal{A}_k) = \sum_{i \in \mathcal{A}_k} \sum_{j \in \mathcal{V}} w_{ij}$. Within-cluster similarity is maximized if $cut(\mathcal{A}_k, \bar{\mathcal{A}}_k)$ is small and $vol(\mathcal{A}_k)$ is large. Therefore, Ncut can be interpreted as cutting through edges rarely transitions by a random walk. Denote the converse Ncut as $CNcut(\mathcal{A}_1, \mathcal{A}_2, ..., \mathcal{A}_K) = \sum_{k=1}^{K} \frac{sasso(\mathcal{A}_k)}{vol(\mathcal{A}_k)}$ where $sasso(\mathcal{A}_k) = vol(\mathcal{A}_k) - cut(\mathcal{A}_k, \bar{\mathcal{A}}_k)$. Since $Ncut(\mathcal{A}_1, \mathcal{A}_2, ..., \mathcal{A}_K) = K - CNcut(\mathcal{A}_1, \mathcal{A}_2, ..., \mathcal{A}_K)$, minimizing $Ncut(\mathcal{A}_1, \mathcal{A}_2, ..., \mathcal{A}_K)$ is equivalent to maximize $CNcut(\mathcal{A}_1, \mathcal{A}_2, ..., \mathcal{A}_K)$ with fixed K.

3 Clustering Algorithm

An essential property of many real datasets is that different local densities may be needed to reveal clusters in different regions of the data space. Thus, the density-based clustering method is a commonly used clustering strategy. DBSCAN Ester et al. (1996) is a well-known density-based clustering method. However, it can only provide flat labeling of the data objects based on a global density threshold. Using a single density threshold can often not properly characterize common data sets with clusters of very different densities or nested clusters. Although many methodologies have been proposed for solving this problem, e.g., HDBSCANCampello et al. (2013), their performance is still unsatisfactory. NcutShi & Malik (2000) is the most famous graph-based clustering method. However, Ncut is an NP-hard problem, and the famous spectral clustering algorithm can only give us a local clustering by applying Kmeans on the eigenvector.

We propose a two-stage clustering method. It remains the advantages of DBSCAN and Ncut, i.e., the usage of the density-based relationship and the graph construction, refinement, and partition technique. In addition, it overcomes the disadvantage of DBSCAN and Ncut. Specifically, CNcut is used to identify the clustering structure instead of using a single density threshold as DBSCAN, and Ncut does not face the uneven density clustering problem of DBSCAN. Due to the density-based ordering step, the Ncut becomes a non-NP-hard problem from the partition view. We proposed the BDSL algorithm with $\mathcal{O}(NlogN)$ complexity to solve the partition-based CNcut problem, and the proposed BDSL can converge to the true clustering partition from the expectation view. Note that we can replace Ncut with RadiocutHagen & Kahng (1992) and replace DBSCAN with OPTICSAnkerst et al. (1999). This paper only chooses DBSCAN and Ncut to describe the proposed integrated density and graph clustering framework.

3.1 Density-Based Graph Ordering

Due to the superiority of graph construction methodology, we start from a graph matrix, which can be a distance measurement or sparse representation, as said in Section 2.1. In this subsection, we consider combining the knowledge of density and graph to permute the graph into a block diagonal form, which implies the clustering structure. Specifically, since the similarity between each sample has been given by the entry of the graph matrix, the density-based greedy searching strategy has the potential ability to iterate through each sample one cluster by one cluster. Then, the graph matrix can be permuted according to the iteration order. Since the nature of ordered clusters, the ordering graph should be in a block-diagonal form.

We use the same greedy ordering strategy as DBSCAN Ester et al. (1996) to iterate through each sample based on the similarity given by the weight matrix. Specifically, given a complete undirected graph $\mathcal{G}$ with a weight matrix $\mathbf{W}$. The element w_{ij} measures the similarity between the ith sample and the jth sample. The more similar the ith sample and the jth sample are, the greater the weight w_{ij}. Denote $N_\delta(i)$ as the **δ-neighborhood index set** of the ith object. We sort the weight w_{ij}, $j = 1, 2, ..., N$, connected to the ith sample in descending order. Since DBSCAN cannot cluster data sets well with large differences in densities, a ratio threshold is proposed instead of using the constant threshold used in DBSCAN. Specifically, the index of the largest values will be put into the set $N_\delta(i)$ one by one until the sum of the weights in $N_\delta(i)$ over the total weight is larger than δ, i.e., $\frac{\sum_{l \in N_\delta(i)} w_{il}}{\sum_{j \in [1,N]} w_{ij}} > \delta$, $\delta \in (0, 1)$. The proposed ratio threshold rule is more suitable for the dataset containing clusters with uneven within-cluster weight density than the constant threshold rule in DBSCAN. Our method can use the ratio threshold rule because we only need to consider the order. However, DBSCAN can only use the constant threshold rule because it still relies on this constant threshold rule to identify the clustering structure. Based on the definition of δ-neighborhood index set, the ith object satisfying $|N_\delta(i)| \geq \varepsilon$ is called a **ε-core sample**, where $|X|$ denotes the number of the element in the set X.

Definition 3.1 (Density-reachable). The jth sample is directly density-reachable from the ith sample w.r.t. δ and ε in a set of samples X if 1) $j \in N_\delta(i)$; 2) $|N_\delta(i)| \geq \varepsilon$.

Instead of ordering the graph based on density-reachable directly, we denote the **core-weight** $c_\varepsilon(i)$ as the weight between ith sample and its εth-largest-weight neighbor. The core-weight of the ith object is simply the largest weight between ith sample and an object in its δ-neighborhood index set such that ith sample would be a core sample with respect to ε if this neighbor is contained in $N_\delta(i)$. Then, the reachability score is defined as follows.

Definition 3.2 (Reachability-score). Suppose $i, j \in [1, N]$, where N is the sample number. Let ε be a natural number. Then, the reachability-score of jth sample with respect to ith sample is defined as

$$s_\varepsilon(j, i) = \begin{cases} \min\left(c_\varepsilon(i), w_{ij}\right), & if\ |N_\delta(i)| \geq \varepsilon \\ Undefined, & otherwise \end{cases}$$

Algorithm 1 Density-based graph ordering (DBGO).

```
Input: the graph $\mathbf{W}$, and the parameter $\delta$, $\varepsilon$.
Initialize an empty set $\mathcal{C}$, and put the index of all $\varepsilon$-core samples into $\mathcal{C}$. Calculate core-weight $c_\varepsilon(i), i \in [1, N]$. Initialize an empty order list $O$, and a null reachability-score recording vector $s \in \mathbb{R}^N$.
for each unprocessed element $i$ in $\mathcal{C}$
    Mark $i$ as processed, and $O = [O; i]$.
    [$Q$, $s$]=QueueUpdate($i$, $Q$, $s$)
    repeat
        Pop $j$ from $Q$.
        Mark $j$ as processed, and $O = [O; j]$.
        if $|N_\delta(j)| \geq \varepsilon$
            [$Q$, $s$]=QueueUpdate($j$, $Q$, $s$)
        end if
    until $Q$ is empty
end for
Permute $\mathbf{W}$ based on $O$.
Output: $\mathbf{W}$
```

Based on the above definitions, we propose a density-based graph ordering approach, namely DBGO, which permutes a random graph into the block-diagonal form. Alg. 1 illustrates the main loop of the algorithm DBGO, and Alg. 2 is a subfunction used in Alg. 1.

Algorithm 2 Queue updating.

```
Input: the processing sample $i$, and the queue $Q$, the recording vector $s$.
for each unprocessed $j$ in $N_\delta(i)$
    $\hat{s} = \min\left(c_\varepsilon(i), w_{ij}\right)$
    if $s_j$ is unassigned
        $s_j = \hat{s}$
        Push $j$ into $Q$
    else
        if $\hat{s} > s_j$
            $s_j = \hat{s}$
        end if
    end if
end for
Output: $Q$, $s$
```

Each ε-core sample is handed over to a procedure of expanding cluster order if the core-sample is not yet processed. In the beginning, one of the unprocessed core samples is marked as processed and appended to the order list O. Then, an empty priority queue Q is initialized, and the samples that may be in the same cluster as the picked core sample will be put in the queue. This operation is accomplished by the QueueUpdate function in Alg.2, i.e., put the index of all of the δ-neighborhood of the i sample into the queue Q. The queue Q stores the index and sorts the index in descending order of their existing-reachability-score.

We use s_j to save the existing-reachability-score of the index j in Q. The existing-reachability-score s_j of the index j will be updated if the reachability-score of j to the processing sample m is larger than the current value of s_j. The pop operation of the queue Q will provide us with the element in Q, which has the largest existing-reachability-score. The push operation of the queue Q will append an element to Q, and the elements in Q will be sorted according to their existing-reachability-score automatically. We pop one of the element m from Q, and push all of the δ-neighborhood of the element m to Q if mth sample is a core sample. These two steps will repeat until Q is empty. The processing order of the sample will be recorded in the list O. Finally, the graph matrix $\mathbf{W}$ is adjusted according to O.

3.2 Block-Diagonal Structure Learning Based on Graph Partition

Recall the proposed converse Ncut in Section 2.2: $CNcut(\{\mathcal{A}_k\}_{k=1}^{K}) = \sum_{k=1}^{K} \frac{sasso(\mathcal{A}_k)}{vol(\mathcal{A}_k)}$. Since the nature of ordered clusters, the graph matrix is shown in a block-diagonal form as illustrated in Fig.1. Thus, the CNut problem can be seen as partitioning an ordered graph matrix into a K-block-diagonal form. Specifically, since the graph is ordered, the graph can be clustered by a set of $K-1$ indexes, which is denoted by a calligraphic letter: $\mathcal{T} = \{t_1, t_2, ..., t_{K-1}\} \subset \{1, 2, ..., N\}$. The dummy indexes $t_0 := 0$ and $t_K := N$ are implicitly available. We have the constraint that $t_{k-1} < t_k$ for any $k \in [1, K]$. Thus, a complete graph $\mathcal{G} = (\mathcal{V}, \mathcal{E}, \mathbf{W})$ can be partitioned into K disjoint subgraph with vertex sets $\mathcal{V}_{t_{k-1},t_k} = \{t_{k-1}+1, t_{k-1}+2, ..., t_k\}$, $k = 1, 2, ..., K$, by simply removing edges connecting the K parts. We propose the following partition-based CNut clustering model:

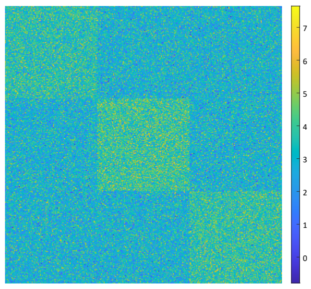

Figure 1: Ordered graph matrix of three clusters.

$$\underset{\{t_k\}_{k=1}^{K-1}, t_{k-1}<t_k}{\text{maximize}} \quad \sum_{k=1}^{K} \frac{sasso(\mathcal{A}_k)}{vol(\mathcal{A}_k)} = \sum_{k=1}^{K} f(t_{k-1}, t_k) = \sum_{k=1}^{K} \frac{\sum_{i,j\in[t_{k-1}+1,t_k]} w_{ij}}{\sum_{i\in[t_{k-1}+1,t_k]} \sum_{j\in[1,N]} w_{ij}} \tag{2}$$

We learn the block-diagonal structure of the graph matrix by solving (2), and the clustering structure will be identified by the $K-1$ partition indexes simultaneously. Since there are N^{K-1} combinations of $K-1$ variables $\{t_k\}_{k=1}^{K-1}$, the global optimal solution of (2) can be searched exhaustively with a $\mathcal{O}(N^{K-1})$ complexity. In order to explore a low-complexity approach to solve (2), we give the following theoretical analysis. Consider partitioning a K-component graph matrix (e.g., the matrix shown in Fig.1) into a two-block-diagonal form. Problem (2) can be written as

$$\underset{t\in(0,N)}{\text{maximize}} \quad f(0,t) + f(t,N) = \frac{\sum_{i,j\in[1,t]} w_{ij}}{\sum_{i\in[1,t]} \sum_{j\in[1,N]} w_{ij}} + \frac{\sum_{i,j\in[t+1,N]} w_{ij}}{\sum_{i\in[t+1,N]} \sum_{j\in[1,N]} w_{ij}} \tag{3}$$

Suppose there exists a K-component graph matrix $\mathbf{W} \in \mathbb{R}^{N\times N}$ which is ordered correctly, and the kth component corresponds to the kth cluster $[c_{k-1}+1, c_k]$, where $c_{k-1} < c_k, \forall k \in [1, K]$, and the dummy indexes $c_0 := 0$, $c_K := N$. Suppose the within-cluster weight of the kth cluster, w_{ij}, $i, j \in [c_{k-1}+1, c_k]$, follows the Gaussian distribution $\mathcal{N}(\mu_k + \beta, \sigma^2)$, $\mu_k > 0$, $\forall k \in [1, K]$, and the between-cluster weight, w_{ij}, $i \in [c_{k-1}+1, c_k]$, $j \notin [c_{k-1}+1, c_k], \forall k \in [1, K]$, follows the Gaussian distribution $\mathcal{N}(\beta, \sigma^2)$, where $\beta > 0$ is a large enough number to make sure all of the weights are non-negative. Denote the expectation of the objective function of (3) as $F(t)$. We have the following observation:

Theorem 3.1. The function $F(t)$ has the following property:

1) If $K = 1$, then $F(t) = 1$ for $t \in (0, N)$.

2) If $K \geq 2$, then $F(t)$ increase for $t \in (0, c_1]$ and decrease for $t \in [c_{K-1}, N)$.

3) If $K \geq 3$, then, for the interval $[c_k, c_{k+1}]$, $\forall k = 1, ..., K-2$, $F(t)$ decreases for $t \in [c_k, \hat{t}]$ and increases for $t \in [\hat{t}, c_{k+1}]$ if $\hat{t} \in (c_k, c_{k+1})$; $F(t)$ decreases for $t \in [c_k, c_{k+1}]$ if $\hat{t} \geq c_{k+1}$; $F(t)$ increases for $t \in [c_k, c_{k+1}]$ if $\hat{t} \leq c_k$, where

$$\hat{t} = \begin{cases} \frac{\left(\sqrt{B_1(c_{k+1}-c_k)^2+1}+\sqrt{B_2(c_{k+1}-c_k)^2+1}\right)^2}{2(c_{k+1}-c_k)(B_2-B_1)} + \frac{1}{2}(c_{k+1}+c_k) & B_1 \neq B_2 \\ \frac{1}{2}(c_{k+1}+c_k) & B_1 = B_2 \end{cases}$$

$B_1 = \frac{\mu_{k+1}+\beta}{\sum_{i=1}^{k}(c_i-c_{i-1})^2(\mu_i+\beta)}$, and $B_2 = \frac{\mu_{k+1}+\beta}{\sum_{i=k+2}^{K}(c_i-c_{i-1})^2(\mu_i+\beta)}$.

The proof of Theorem 3.1 is shown in Appendix A. Theorem 3.1 illustrate that $F(t)$ will be constant if none c_k in $(0, N)$ ($K = 1$), and one of $\{c_k\}_{k=1}^{K-1}$ will maximize $F(t)$ if there exists partition indexes $\{c_k\}_{k=1}^{K-1}$ in $(0, N)$ ($K \geq 2$). We denote the bounded binary-segmentation function as $F_B(c_i, t, c_j) = \mathbb{E}\left[f(c_i, t) + f(t, c_j)\right]$, where c_i, c_j are two of the true partition $\{c_k\}_{k=0}^{K}$ and $c_i < c_j$. Obviously, $F_B(0, t, N) = F(t)$. Then, we have the following corollary.

Corollary 3.1.1. For the interval $(c_i, c_j) \subseteq (0, N)$, where c_i, c_j are two of the partition $\{c_k\}_{k=0}^{K}$, and $c_i < c_j$, the function $F_B(c_i, t, c_j)$ has the same property as the function $F(t)$:

1) If none of $\{c_k\}_{k=1}^{K-1}$ in (c_i, c_j), then $F_B(c_i, t, c_j) = 1$

2) If there exists partition indexes $c_p, c_{p+1}, ..., c_q \in \{c_k\}_{k=1}^{K-1}$, $1 \leq p \leq q \leq K-1$, in (c_i, c_j), then one of $c_p, c_{p+1}, ..., c_q$ will maximize $F_B(c_i, t, c_j)$.

Denote the binary-segmentation reward function as

$$h(c_i, t, c_j) = f(c_i, t) + f(t, c_j) - f(c_i, c_j)$$

The function $h(c_i, t, c_j)$ represents the reward when we replace one segment $f(c_i, c_j)$ with two segments $f(c_i, t)$ and $f(t, c_j)$. The expectation of $h(c_i, t, c_j)$ is

$$\mathbb{E}(h(c_i, t, c_j)) = F_B(c_i, t, c_j) - \mathbb{E}\{f(c_i, c_j)\}$$

Since $\mathbb{E}\{f(c_i, c_j)\}$ is constant with given c_i and c_j, $\max_{t \in (c_i, c_j)} \mathbb{E}(h(c_i, t, c_j))$ is equivalent to $\max_{t \in (c_i, c_j)} F_B(c_i, t, c_j)$. Thus, the expectation function $\mathbb{E}(h(c_i, t, c_j))$ has the same property as $F_B(c_i, t, c_j)$, i.e., Corollary 3.1.1 also holds for $\mathbb{E}(h(c_i, t, c_j))$.

The property of function $F_B(\cdot)$ implies the character of the objective function of the problem (2). If we partition the graph matrix into two-block-diagonal matrix by solving (3), Theorem 3.1 points out that one of the true partition $\{c_k\}_{k=1}^{K-1}$ will maximize (3) from the expectation view. Suppose the first estimated partition is c_1, and the indexes $\tau_0 = 0$, $\tau_1 = c_1$, $\tau_2 = N$. If we want to obtain the second partition by inserting a new partition among $\{\tau_i\}_{i=1}^{2}$, in other words, we want to choose one of the two intervals (τ_0, τ_1), and (τ_1, τ_2) to insert a new partition. Then, we have the following problem.

$$\underset{j \in [1,2]}{\text{maximize}} \; \underset{t \in (\tau_{j-1}, \tau_j)}{\text{maximize}} \sum_{k=1}^{2} f(\tau_{k-1}, \tau_k) + h(\tau_{j-1}, t, \tau_j) \tag{4}$$

Since $\sum_{k=1}^{2} f(\tau_{k-1}, \tau_k)$ is constant, Problem (4) is equivalent to

$$\underset{j \in [1,2]}{\text{maximize}} \; \underset{t \in (\tau_{j-1}, \tau_j)}{\text{maximize}} \; h(\tau_{j-1}, t, \tau_j) \tag{5}$$

As said before, Corollary 3.1.1 also holds for $\mathbb{E}(h(c_i, t, c_j))$. Corollary 3.1.1 indicates that one of $\{c_k\}_{k=1}^{K-1}$ will maximize the expectation of $h(\tau_{j-1}, t, \tau_j)$. If c_2 maximizes (5), then c_2 also maximizes (4). Suppose the indexes $\tau_0 = 0$, $\tau_1 = c_1$,$\tau_2 = c_2$ $\tau_3 = N$. We choose one of the three intervals to insert the third partition. The problem will be shown in a similar form as (5). We repeat the updating process until we obtain $K-1$ optimal partition index $\{c_k\}_{k=1}^{K-1}$ for the problem (2). The pseudo-code for the procedure of graph partition is depicted in Alg. 3, namely BDSL, which is of the order of $\mathcal{O}(N\log N)$.

Algorithm 3 Block-diagonal structure learning (BDSL).

```
Input: an ordered graph $\mathbf{W}$, and the cluster number $K$
Initialize: $\mathcal{T} = \{\}$
repeat
    Initialize $L = |\mathcal{T}| + 1$, $t_0 = 0$, $t_L = N$,
    $\mathcal{T} = \{t_1, ..., t_{L-1}\}$, and two empty vector $v, g \in \mathbb{R}^L$.
    for $k = 0 : (L-1)$ do
        $v_k = \underset{t \in (t_k+1, t_{k+1})}{\text{argmax}} \; f(t_k, t) + f(t, t_{k+1})$
        $g_k = f(t_k, v_k) + f(v_k, t_{k+1}) - f(t_k, t_{k+1})$
    end for
    $\hat{k} = \underset{k}{\text{argmax}} \, g_k$, and $\hat{t} = v_{\hat{k}}$
    $\mathcal{T} = \mathcal{T} \cup \{\hat{t}\}$
until $L = K - 1$
Output: $\{\mathcal{T}\}$
```

Theorem 3.2. If the within-cluster weight of the kth cluster $\{c_{k-1}+1, c_{k-1}+2, ..., c_k\}$ follows the Gaussian distribution $\mathcal{N}(\mu_k + \beta, \sigma^2)$, $\mu_k > 0, \forall k \in [1, K]$, and the between-cluster weight follows the Gaussian distribution $\mathcal{N}(\beta, \sigma^2)$, where $\beta > 0$ is a large enough number to make sure all of the weight are non-negative. Then, our BDSL algorithm will converge to $\{c_k\}_{k=1}^{K-1}$ by maximizing the expectation of the objective function of Problem (2).

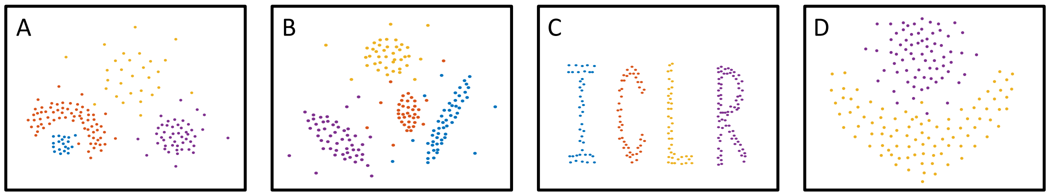

Figure 2: Four synthetic datasets.

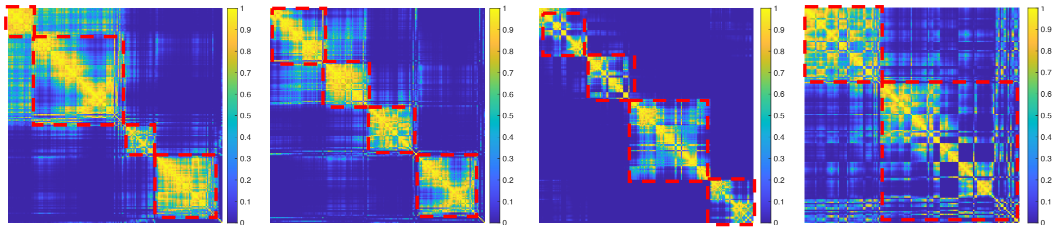

Figure 3: The affinity matrix of four synthetic datasets, which has been ordered by the proposed DBGO algorithm.

4 Experimental Results and Applications

In this section, extensive experiments are performed on different types of datasets to demonstrate the effectiveness of the proposed method in comparison with the state-of-the-art clustering methods. First, a synthetic dataset is generated to show the performance of the proposed method on the dataset with different densities, distributions, and connectivity. Then the widely used image datasets are used to show the effectiveness of the proposed method on the subspace clustering task.

4.1 Synthetic Experiments

We generate four basic datasets to test the effectiveness and robustness of the proposed methods on the general clustering task, which is shown in Figure 2. Dataset A contains four clusters with different densities and shapes. Dataset B contains four clusters following different Gaussian distributions. Dataset C contains four oddly shaped clusters with clear boundaries. Dataset D contains two coupled clusters without clear boundaries. As said in Section 2.1, different distance measurements can be used to show the similarity between samples. We pick Euclidean distance after exponential normalization as the weight. We compare the proposed approach with six types of clustering methods, including K-means MacQueen (1967), STREAM O'callaghan et al. (2002), BIRCH Zhang et al. (1996), DBSCAN Ester et al. (1996) OPTICS Ankerst et al. (1999), HDBSCAN Campello et al. (2013), ST-DBSCAN Birant & Kut (2007), STING Wang et al. (1997), GMM Rasmussen (1999),COBWEB Fisher (1987), spectral clustering Von Luxburg (2007), CLICK Sharan & Shamir (2000), and Chinese Whisper (CW) Biemann (2006). We tune the parameters for these methods to achieve the best results. The experiments are repeated ten times. The mean clustering accuracy is computed to show the performance.

Table 1: Clustering accuracy of synthetic datasets.

Dataset	A	B	C	D
Graph-based approach				
SpectralC	87.29	82.41	92.13	82.55
CLICK	90.88	75.56	83.57	73.66
CW	86.94	88.14	88.61	72.11
Density-based approach				
DBSCAN	79.41	89.72	93.15	87.81
OPTICS	84.21	89.47	95.24	77.45
HDBSCAN	89.15	92.77	93.24	88.26
ST-DBSCAN	77.87	81.24	94.36	75.92
Partition approach				
K-means	74.15	57.43	60.40	58.26
STREAM	71.43	60.76	52.40	80.77
Hierarchical approach				
BIRCH	60.08	81.94	64.40	77.55
Model-based approach				
GMM	75.54	70.39	72.34	82.52
COBWEB	83.31	90.07	75.37	78.29
Grid-based approach				
STING	70.44	72.68	70.76	83.68
Proposed	**97.29**	**99.38**	**99.90**	**99.96**

The experimental results are shown in Tab.1. We provide a visual comparison of graph weight matrices in Fig.3. It can be observed that our method DBGO provides weight matrices that are in a block-diagonal form roughly. The red dotted boxes show the graph partition result provided by BDSL. As anticipated, our proposed DBGO-BDSL outperforms all of the comparisons. This experiment highlights the robustness of our methods to noises and densities.

4.2 Experiments on High-dimensional Datasets

For good performance in dealing with complex local correlation and high-dimensional structure of the image data, representation-based methods have become one of the hot topics for high-dimensional data clustering, in which spectral-based subspace clustering is a representative tool. However, all of these state-of-the-art spectral-based subspace clustering methods use the spectral clustering method to solve the Ncut problem and identify the clustering structure by K-Means. The spectral clustering method shows poor performance on the non-uniform density and distribution dataset, which is shown in Section 4.1. Thus, we replace the spectral clustering with our proposed DBGO-BDSL in the conventional spectral-based subspace clustering methods to show the effectiveness of our DBGO-BDSL.

Table 2: Clustering accuracy of high-dimensional datasets. LSR means the original version of the conventional subspace clustering technique, and LSR-DB means we apply our DBGO-BDSL on the affinity matrix generated by LSR.

Dataset	AR	COIL-20	USPS	E-Yale B	MNIST
Subjects	100	20	10	38	10
LSR	74.15	57.43	60.40	58.26	57.80
LSR-DB	82.11(**+7.96**)	94.74(**+37.31**)	90.59(**+30.19**)	88.39(**+30.13**)	96.29(**+38.49**)
NSN	42.18	81.94	44.40	77.55	55.50
NSN-DB	80.79(**+38.61**)	92.14(**+10.2**)	94.18(**+49.78**)	91.27(**+13.72**)	95.66(**+40.16**)
SMR	68.85	75.49	68.83	67.91	68.87
SMR-DB	79.38(**+10.53**)	92.47(**+16.98**)	98.52(**+29.69**)	77.69(**+9.78**)	94.74(**+25.87**)
SSC	24.54	50.39	42.34	32.52	40.16
SSC-DB	75.28(**+50.74**)	90.19(**+39.80**)	92.56(**+50.22**)	82.43(**+49.91**)	90.26(**+50.10**)
QOSC	63.41	89.72	83.15	57.81	65.30
QOSC-DB	79.55(**+16.14**)	94.67(**+4.95**)	**98.84**(**+15.69**)	84.29(**+26.48**)	95.57(**+30.27**)
ORGEN	25.88	75.56	63.57	53.66	61.50
ORGEN-DB	80.52(**+54.64**)	**95.47**(**+19.91**)	93.59(**+30.02**)	93.64(**+39.98**)	91.32(**+29.82**)
iPursuit	65.88	73.47	43.30	84.24	75.40
iPursuit-DB	82.92(**+17.04**)	91.01(**+17.54**)	93.83(**+50.53**)	91.27(**+7.03**)	95.99(**+20.59**)
FGNSC	68.31	90.11	75.37	78.29	52.40
FGNSC-DB	73.88(**+5.57**)	95.38(**+5.27**)	95.82(**+20.45**)	**96.47**(**+18.18**)	92.69(**+40.29**)
BDRZ	51.43	71.67	52.40	80.77	71.80
BDRZ-DB	**88.77**(**+37.34**)	90.42(**+18.75**)	92.58(**+40.18**)	90.76(**+9.99**)	**99.36**(**+27.56**)

We test our method on five benchmark datasets (as shown in Fig.4) for famous tough high-dimensional clustering tasks, including face recognition (E-Yale B Georghiades et al. (2001), AR Martinez & Benavente (1998)), object detection (COIL-20 Nene et al. (1996)), and handwritten digits identification (MNIST LeCun et al. (1998), USPS Hastie et al. (2009)). Moreover, We use state-of-the-art spectral-based subspace clustering methods to generate the affinity matrix and then apply the proposed DBGO-BDSL to the affinity matrix. We compare the traditional version of these state-of-the-art methods with the improved version. The comparison includes representation-based methods (SRC Xiao et al. (2020) LLC Tang et al. (2018), CRC Wang et al. (2022), LRC Zhang et al. (2021b)), and low-rank criterion-based methods (LSRLu et al. (2012), NSNPark et al. (2014), SMRHu et al. (2014), SSCXu et al. (2015), QOSCLu et al. (2018), ORGENYou et al. (2016), iPursuit Rahmani & Atia (2017), FGNSCYang et al. (2019), and BDRZLu et al. (2018)).

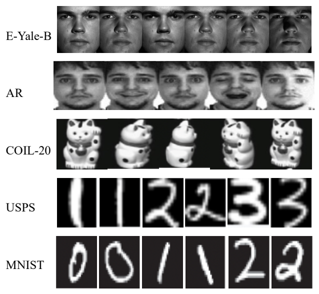

Figure 4: Example samples.

We tune the parameters for the comparison methods to achieve the best results. The experiments are repeated ten times, and the mean clustering accuracy is computed to show the performance. The experimental result is shown in Tab.2. We compare the original version of the conventional subspace clustering technique with our improved version, Each state-of-the-art spectral-based subspace clustering method's performance can be improved by our DBGO-BDSL, and their clustering accuracy is improved by roughly 10%∼50%. The performance of the proposed DBGO-BDSL depends on the representation quality of the affinity matrix given by the state-of-the-art method.

5 Conclusions

In this paper, we proposed a new two-stage integrated density and graph clustering method for low- or high-dimensional clustering tasks. The proposed clustering framework can overcome the drawback of the traditional density- and graph-based clustering strategies, and remains their advantages. Comprehensive experiments were conducted on synthetic datasets and real datasets. The proposed method not only shows the best performance on the synthetic low-dimensional dataset but also improves the clustering accuracy of the conventional spectral-based subspace clustering method by roughly 10%∼50% on the real high-dimensional dataset. The proposed clustering structure can be improved in many aspects in the future. For instance. we can add a graph refinement step to adjust the ordered matrix, refine the DBGO step using a more reasonable density-based search strategy, and replace the BDSL step with a more precise partition method.

REFERENCES

Mihael Ankerst, Markus M Breunig, Hans-Peter Kriegel, and Jörg Sander. Optics: Ordering points to identify the clustering structure. *ACM Sigmod record*, 28(2):49–60, 1999.

Elena Battaglia and Ruggero G Pensa. A parameter-less algorithm for tensor co-clustering. *Machine Learning*, pp. 1–43, 2021.

Chris Biemann. Chinese whispers-an efficient graph clustering algorithm and its application to natural language processing problems. In *Proceedings of TextGraphs: the first workshop on graph based methods for natural language processing*, pp. 73–80, 2006.

Derya Birant and Alp Kut. St-dbscan: An algorithm for clustering spatial–temporal data. *Data & knowledge engineering*, 60(1):208–221, 2007.

Yaoming Cai, Meng Zeng, Zhihua Cai, Xiaobo Liu, and Zijia Zhang. Graph regularized residual subspace clustering network for hyperspectral image clustering. *Information Sciences*, 578:85–101, 2021.

Ricardo JGB Campello, Davoud Moulavi, and Jörg Sander. Density-based clustering based on hierarchical density estimates. In *Pacific-Asia conference on knowledge discovery and data mining*, pp. 160–172, 2013.

Yewang Chen, Lida Zhou, Nizar Bouguila, Cheng Wang, Yi Chen, and Jixiang Du. Block-dbscan: Fast clustering for large scale data. *Pattern Recognition*, 109:107624, 2021.

Seung-Seok Choi, Sung-Hyuk Cha, and Charles C Tappert. A survey of binary similarity and distance measures. *Journal of systemics, cybernetics and informatics*, 8(1):43–48, 2010.

Zhihua Cui, Xuechun Jing, Peng Zhao, Wensheng Zhang, and Jinjun Chen. A new subspace clustering strategy for ai-based data analysis in iot system. *IEEE Internet of Things Journal*, 8(16): 12540–12549, 2021.

Chris Ding, Xiaofeng He, and Horst D Simon. On the equivalence of nonnegative matrix factorization and spectral clustering. In *Proceedings of the 2005 SIAM international conference on data mining*, pp. 606–610, 2005.

Chris Ding, Tao Li, Wei Peng, and Haesun Park. Orthogonal nonnegative matrix t-factorizations for clustering. In *Proceedings of the 12th ACM SIGKDD international conference on Knowledge discovery and data mining*, pp. 126–135, 2006.

Ehsan Elhamifar and René Vidal. Sparse subspace clustering: Algorithm, theory, and applications. *IEEE transactions on pattern analysis and machine intelligence*, 35(11):2765–2781, 2013a.

Ehsan Elhamifar and René Vidal. Sparse subspace clustering: Algorithm, theory, and applications. *IEEE transactions on pattern analysis and machine intelligence*, 35(11):2765–2781, 2013b.

Martin Ester, Hans-Peter Kriegel, Jörg Sander, Xiaowei Xu, et al. A density-based algorithm for discovering clusters in large spatial databases with noise. In *kdd*, volume 96, pp. 226–231, 1996.

AM Fahim, AM Salem, FA Torkey, and MA Ramadan. Density clustering based on radius of data (dcbrd). *International Journal of Computer and Information Engineering*, 2(10):3464–3469, 2008.

Douglas H Fisher. Knowledge acquisition via incremental conceptual clustering. *Machine learning*, 2(2):139–172, 1987.

Athinodoros S. Georghiades, Peter N. Belhumeur, and David J. Kriegman. From few to many: Illumination cone models for face recognition under variable lighting and pose. *IEEE transactions on pattern analysis and machine intelligence*, 23(6):643–660, 2001.

Lars Hagen and Andrew B Kahng. New spectral methods for ratio cut partitioning and clustering. *IEEE transactions on computer-aided design of integrated circuits and systems*, 11(9):1074–1085, 1992.

Trevor Hastie, Robert Tibshirani, Jerome H Friedman, and Jerome H Friedman. *The elements of statistical learning: data mining, inference, and prediction*, volume 2. 2009.

Han Hu, Zhouchen Lin, Jianjiang Feng, and Jie Zhou. Smooth representation clustering. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 3834–3841, 2014.

Pan Ji, Tong Zhang, Hongdong Li, Mathieu Salzmann, and Ian Reid. Deep subspace clustering networks. *Advances in neural information processing systems*, 30, 2017.

Zhao Kang, Xinjia Zhao, Chong Peng, Hongyuan Zhu, Joey Tianyi Zhou, Xi Peng, Wenyu Chen, and Zenglin Xu. Partition level multiview subspace clustering. *Neural Networks*, 122:279–288, 2020.

Zhao Kang, Zhiping Lin, Xiaofeng Zhu, and Wenbo Xu. Structured graph learning for scalable subspace clustering: From single view to multiview. *IEEE Transactions on Cybernetics*, 2021.

Yann LeCun, Léon Bottou, Yoshua Bengio, and Patrick Haffner. Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, 86(11):2278–2324, 1998.

Qiliang Liu, Jie Yang, Min Deng, Ci Song, and Wenkai Liu. Snn_flow: a shared nearest-neighbor-based clustering method for inhomogeneous origin-destination flows. *International Journal of Geographical Information Science*, 36(2):253–279, 2022.

Can-Yi Lu, Hai Min, Zhong-Qiu Zhao, Lin Zhu, De-Shuang Huang, and Shuicheng Yan. Robust and efficient subspace segmentation via least squares regression. In *European conference on computer vision*, pp. 347–360. Springer, 2012.

Canyi Lu, Jiashi Feng, Zhouchen Lin, Tao Mei, and Shuicheng Yan. Subspace clustering by block diagonal representation. *IEEE transactions on pattern analysis and machine intelligence*, 41(2): 487–501, 2018.

Juncheng Lv, Zhao Kang, Xiao Lu, and Zenglin Xu. Pseudo-supervised deep subspace clustering. *IEEE Transactions on Image Processing*, 30:5252–5263, 2021.

J MacQueen. Classification and analysis of multivariate observations. In *5th Berkeley Symp. Math. Statist. Probability*, pp. 281–297, 1967.

Aleix Martinez and Robert Benavente. The ar face database: Cvc technical report, 24. 1998.

Vishnu Menon, Gokularam Muthukrishnan, and Sheetal Kalyani. Subspace clustering without knowing the number of clusters: A parameter free approach. *IEEE Transactions on Signal Processing*, 68:5047–5062, 2020.

Sameer A Nene, Shree K Nayar, Hiroshi Murase, et al. Columbia object image library (coil-100). 1996.

Liadan O'callaghan, Nina Mishra, Adam Meyerson, Sudipto Guha, and Rajeev Motwani. Streaming-data algorithms for high-quality clustering. In *Proceedings 18th International Conference on Data Engineering*, pp. 685–694, 2002.

Dohyung Park, Constantine Caramanis, and Sujay Sanghavi. Greedy subspace clustering. *Advances in neural information processing systems*, 27, 2014.

Mostafa Rahmani and George Atia. Innovation pursuit: A new approach to the subspace clustering problem. In *International conference on machine learning*, pp. 2874–2882. PMLR, 2017.

Carl Rasmussen. The infinite gaussian mixture model. *Advances in neural information processing systems*, 12, 1999.

Fadi Şaar and Ahmet E Topcu. Minimum spanning tree-based cluster analysis: A new algorithm for determining inconsistent edges. *Concurrency and Computation: Practice and Experience*, 34(9): e6717, 2022.

Roded Sharan and Ron Shamir. Click: a clustering algorithm with applications to gene expression analysis. In *Proc Int Conf Intell Syst Mol Biol*, volume 8, pp. 16, 2000.

Krishna Kumar Sharma and Ayan Seal. Multi-view spectral clustering for uncertain objects. *Information Sciences*, 547:723–745, 2021.

Jianbo Shi and Jitendra Malik. Normalized cuts and image segmentation. *IEEE Transactions on pattern analysis and machine intelligence*, 22(8):888–905, 2000.

Chang Tang, Xinzhong Zhu, Xinwang Liu, Miaomiao Li, Pichao Wang, Changqing Zhang, and Lizhe Wang. Learning a joint affinity graph for multiview subspace clustering. *IEEE Transactions on Multimedia*, 21(7):1724–1736, 2018.

René Vidal and Paolo Favaro. Low rank subspace clustering (lrsc). *Pattern Recognition Letters*, 43: 47–61, 2014.

Ulrike Von Luxburg. A tutorial on spectral clustering. *Statistics and computing*, 17(4):395–416, 2007.

Wei Wang, Jiong Yang, Richard Muntz, et al. Sting: A statistical information grid approach to spatial data mining. In *Vldb*, volume 97, pp. 186–195, 1997.

Wei Wang, Xiaohui Hu, and Yao Du. Algorithm optimization and anomaly detection simulation based on extended jarvis-patrick clustering and outlier detection. *Alexandria Engineering Journal*, 61(3):2106–2115, 2022.

Xiaolin Xiao, Yue-Jiao Gong, Zhongyun Hua, and Wei-Neng Chen. On reliable multi-view affinity learning for subspace clustering. *IEEE Transactions on Multimedia*, 23:4555–4566, 2020.

Jun Xu, Kui Xu, Ke Chen, and Jishou Ruan. Reweighted sparse subspace clustering. *Computer Vision and Image Understanding*, 138:25–37, 2015.

Yesong Xu, Shuo Chen, Jun Li, Lei Luo, and Jian Yang. Learnable low-rank latent dictionary for subspace clustering. *Pattern Recognition*, 120:108142, 2021.

Jufeng Yang, Jie Liang, Kai Wang, Paul L Rosin, and Ming-Hsuan Yang. Subspace clustering via good neighbors. *IEEE transactions on pattern analysis and machine intelligence*, 42(6):1537–1544, 2019.

Chong You, Chun-Guang Li, Daniel P Robinson, and René Vidal. Oracle based active set algorithm for scalable elastic net subspace clustering. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 3928–3937, 2016.

Shangzhi Zhang, Chong You, René Vidal, and Chun-Guang Li. Learning a self-expressive network for subspace clustering. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 12393–12403, 2021a.

Tian Zhang, Raghu Ramakrishnan, and Miron Livny. Birch: an efficient data clustering method for very large databases. *ACM sigmod record*, 25(2):103–114, 1996.

Yuru Zhang, Shifei Ding, Lijuan Wang, Yanru Wang, and Ling Ding. Chameleon algorithm based on mutual k-nearest neighbors. *Applied Intelligence*, 51(4):2031–2044, 2021b.

Yuanjing Zhu and Xiaolong Chen. Design and research of a new clustering algorithm for wireless sensor network. In *IOP Conference Series: Earth and Environmental Science*, volume 769, pp. 042072, 2021.

A Appendix

A.1 Proof of Theorem 3.1

Case 0: $K = 1$, we have $F(t) = \frac{t_1^2(\mu_1+\beta)}{t_1 N(\mu_1+\beta)} + \frac{(N-t_1)^2(\mu_1+\beta)}{(N-t_1)N(\mu_1+\beta)} = 1$.

Case 1: $K \geq 2$. Consider $t \in (0, c_1]$, the $F(t)$ is given by

$$F(t) = \frac{t_1^2\mu_1 + t_1^2\beta}{t_1c_1\mu_1 + t_1N\beta} + \frac{(c_1-t_1)^2\mu_1 + (c_2-c_1)^2\mu_2 + ... + (N-c_{K-1})^2\mu_K + (N-t_1)^2\beta}{(c_1-t_1)c_1\mu_1 + (c_2-c_1)^2\mu_2 + ... + (N-c_{K-1})^2\mu_K + N(N-t_1)\beta} \tag{6}$$

$$= \frac{t_1}{c_1} + \frac{1+(c_1-t_1)^2C_1}{1+(c_1-t_1)c_1C_1}$$

$$= 1 + \frac{t_1}{c_1 + c_1^2C_1(c_1-t_1)}$$

$$= 1 + \frac{1}{(1+c_1^2C_1)c_1t_1^{-1} - c_1^2C_1}$$

where the mean weight of the ith, $i \in \{2, 3, ..., K\}$ cluster with zero covariance is $b(2, K) = \frac{(c_2-c_1)^2(\mu_2+\beta)+...+(N-c_{k-1})^2(\mu_K+\beta)}{(N-c_1)^2}$, and $C_1 = \frac{\mu_1+\beta}{(N-c_1)^2b(2,K)}$. $F(t)$ will increase with the increase of $t \in (0, c_1]$.

Case 2: $K \geq 2$. Consider $t \in [c_{K-1}, N)$, the $F(t)$ is given by

$$F(t) = \frac{(N-t_1)^2\mu_1 + t_1^2\beta}{(N-t_1)(N-c_{K-1})\mu_1 + t_1N\beta}$$
$$+ \frac{c_1^2\mu_1 + (c_2-c_1)^2\mu_2 + ... + (c_{K-1}-c_{K-2})^2\mu_{K-1} + (t_1-c_{K-1})^2\mu_K + (N-t_1)^2\beta}{c_1^2\mu_1 + (c_2-c_1)^2\mu_2 + ... + (c_{K-1}-c_{K-2})^2\mu_{K-1} + (t_1-c_{K-1})(N-c_{K-1})\mu_K + N(N-t_1)\beta}$$

$$= \frac{(N-t_1)^2\mu_1}{(N-t_1)(N-c_{K-1})\mu_1} + \frac{c_{K-1}^2b(1,K-1) + (t_1-c_{K-1})^2\mu_K}{c_{K-1}^2b(1,K-1) + (t_1-c_{K-1})(N-c_{K-1})\mu_K}$$

$$= \frac{N-t_1}{N-c_{K-1}} + \frac{1+(t_1-c_{K-1})^2C_2}{1+(t_1-c_{K-1})(N-c_{K-1})C_2}$$

$$= 1 + \frac{N-t_1}{N-c_{K-1} + (N-c_{K-1})^2C_2(t_1-c_{K-1})}$$

$$= 1 + \frac{1}{(N-c_{K-1})\left[1+(N-c_{K-1})^2C_2\right](N-t_1)^{-1} - (N-c_{K-1})^2C_2}$$

where $b(1, K-1) = \frac{c_1^2(\mu_1+\beta)+(c_2-c_1)^2(\mu_2+\beta)+...+(c_{K-1}-c_{K-2})^2(\mu_{K-1}+\beta)}{c_{K-1}^2}$, and $C_2 = \frac{\mu_K+\beta}{c_{K-1}^2b(1,K-1)}$.
$F(t)$ will decrease with the increase of $t \in [c_{K-1}, N)$.

Case 3: $K \geq 3$. Consider $t \in [c_k, c_{k+1}]$, $k = 1, 2, ..., K-2$.

$$F(t) = \frac{c_1^2\mu_1 + ..., + (c_k-c_{k-1})^2\mu_k + (t_1-c_k)^2\mu_{k+1} + t_1^2\beta}{c_1^2\mu_1 + ..., + (c_k-c_{k-1})^2\mu_k + (c_{k+1}-c_k)(t_1-c_k)\mu_{k+1} + t_1N\beta} \tag{7}$$
$$+ \frac{(c_{l+2}-c_{k+1})^2\mu_{l+2} + ..., + (N-c_{k-1})^2\mu_K + (c_{k+1}-t_1)^2\mu_{k+1} + (N-t_1)^2\beta}{(c_{l+2}-c_{k+1})^2\mu_{l+2} + ..., + (N-c_{k-1})^2\mu_K + (c_{k+1}-t_1)(c_{k+1}-c_k)\mu_2 + N(N-t_1)\beta}$$
$$= \frac{c_k^2b(1,k) + (t_1-c_k)^2\mu_{k+1}}{c_k^2b(1,k) + (c_{k+1}-c_k)(t_1-c_k)\mu_{k+1}} + \frac{(N-c_{k+1})^2b(k+2,K) + (c_{k+1}-t_1)^2\mu_{k+1}}{(N-c_{k+1})^2b(k+2,K) + (c_{k+1}-t_1)(c_{k+1}-c_k)\mu_{k+1}}$$

where

$$b(1,k) = \frac{c_1^2(\mu_1+\beta) + ..., + (c_k-c_{k-1})^2(\mu_k+\beta)}{c_k^2}$$

and

$$b(k+2,K)=\frac{\left(c_{k+2}-c_{k+1}\right)^2\left(\mu_{k+2}+\beta\right)+...,+\left(N-c_{k-1}\right)^2\left(\mu_K+\beta\right)}{\left(N-c_{k+1}\right)^2}$$

Denote $d=c_{k+1}-c_k$, and $x=t_1-\frac{c_{k+1}+c_k}{2}$. Since $t\in(c_{k+1},c_k)$, we have $x\in(-\frac{d}{2},\frac{d}{2})$. Denote $B_1=\frac{\mu_{k+1}+\beta}{c_k^2 b(1,k)}$, and $B_2=\frac{\mu_{k+1}+\beta}{(N-c_{k+1})^2 b(k+2,K)}$. So, $F(t)$ can be written as

$$F(x)=\frac{x^2(-d^2B_1B_2+B_1+B_2)+2dx(B_1-B_2)+\frac{3}{4}d^2(B_1+B_2)+\frac{d^4}{4}B_1B_2+2}{-x^2d^2B_1B_2+xd(B_1-B_2)+\frac{d^4}{4}B_1B_2+\frac{1}{2}d^2(B_1+B_2)+1}$$

$F(x)$ is convex for $x\in(-\frac{d}{2},\frac{d}{2})$. If we set the differentiate of $F(x)$ w.r.t. x as zero, we can obtain x_0 and we can say $F(x)$ is minimized at x_0, where

$$x_0=\begin{cases}\frac{\left(\sqrt{B_1d^2+1}+\sqrt{B_2d^2+1}\right)^2}{2d(B_2-B_1)} & B_1\neq B_2\\ 0 & B_1=B_2\end{cases}$$

Recall the definition of d, and x, $F(t)$ is minimized at

$$\hat{t}=\begin{cases}\frac{\left(\sqrt{B_1(c_{k+1}-c_k)^2+1}+\sqrt{B_2(c_{k+1}-c_k)^2+1}\right)^2}{2(c_{k+1}-c_k)(B_2-B_1)}+\frac{1}{2}\left(c_{k+1}+c_k\right) & B_1\neq B_2\\ \frac{1}{2}\left(c_{k+1}+c_k\right) & B_1=B_2\end{cases}$$

Thus, for the interval $[c_k,c_{k+1}]$, $k=1,...,K-2$, $F(t)$ decreases for $t\in[c_k,\hat{t}]$ and increases for $t\in[\hat{t},c_{k+1}]$ if $\hat{t}\in(c_k,c_{k+1})$; $F(t)$ decreases for $t\in[c_k,c_{k+1}]$ if $\hat{t}\geq c_{k+1}$; $F(t)$ increases for $t\in[c_k,c_{k+1}]$ if $\hat{t}\leq c_k$.