BLOCK-DIAGONAL STRUCTURE LEARNING FOR SUBSPACE CLUSTERING

Anonymous authors
Paper under double-blind review

ABSTRACT

Finding the informative subspaces of high-dimensional ordered datasets is at the core of numerous applications in computer vision, where spectral-based subspace clustering is arguably the most commonly studied method due to its solid empirical performance. Such algorithms compute an affinity matrix to construct a self-representation for each sample utilizing other samples as a dictionary, and spectral clustering is employed to identify the clustering structure based on the affinity matrix. The affinity matrix is shown in a block-diagonal form due to the nature of the ordered dataset. However, almost all the works focus on self-representation matrix construction with block diagonal prior, and none resort to the block diagonal partition of the self-representation matrix. Despite the fact that the block-diagonal structure learning embedded in self-representation plays a vital role in effective ordered subspace clustering, direct optimization with block-diagonal priors is challenging due to the sparsity and connectivity of self-representation. In this paper, we propose a technique, namely block-diagonal structure representation learning, to directly identify the optimal clustering structure of an ordered dataset instead of employing spectral clustering based on a given graph. The proposed algorithm can theoretically achieve the globally optimal solution to the proposed discrete block-diagonal partition problem. We test the proposed clustering method on several types of ordered databases, such as human face recognization, video scene clip partition, motion tracking, and dynamic 3-D facial expression cutting. The experiments illustrate that the proposed method can improve the performance of all of the state-of-the-art subspace clustering methods by roughly 30% 40% on the ordered dataset.

1 INTRODUCTION

High-dimensional data has widely emerged in the areas of image processing, computer vision, pattern recognition, bioinformatics, etc. The foundation of subspace clustering is that most data often have intrinsic subspace structures and can be regarded as samples drawn from multiple subspaces. For example, face images under varying illumination and multiple instances of handwritten digits with different translations can be approximately represented by subspaces, with each subspace corresponding to a category. Recovering the low-dimensional subspace of data is not only beneficial to the storage and management of data but also reduces the impact of high dimensionality for pattern recognition, thereby improving the performance of the algorithm. Also, it is well known that recovering low-dimensional subspaces of data plays a vital role in subspace clustering. Subspace clustering groups data into clusters to which each subspace belongs according to the low-dimensional subspace structure of data.

With excellent performance and simplicity, spectral-based subspace clustering algorithms have shown great success in recent years. This type of method first learns the representation coefficient matrix of the input data and constructs a similarity matrix by the coefficient matrix. Then, the clustering structure is identified by spectral clustering such as Ncut (Normalized cut). Obviously, the essential step of this subspace clustering algorithm is to seek a suitable representation coefficient matrix for data. However, the existing general subspace clustering methods show poor performance on the ordered dataset, even with different priors, e.g., block-diagonal Lu et al. (2018), sparse Xu et al. (2015), low-rank Liu et al. (2012).

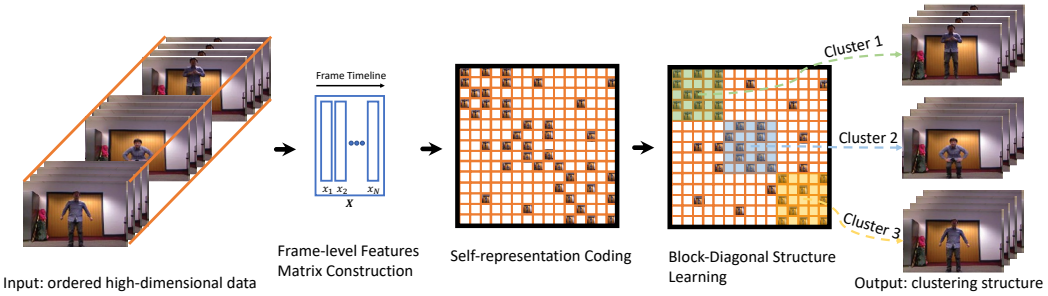


Figure 1: The framework of the proposed approach. We first extract frame-level features from the ordered dataset (e.g., video sequence) and learn the representation coding. Then, the block-diagonal structure of the self-representation is learned by the proposed partition method, and the clustering structure is identified simultaneously in the form of generating multiple temporal segments.

In addition, the works designed with an ordered prior show a little better performance than the general subspace clustering approaches. However, all the works focus on utilizing the order information embedded in sequential data by complicating the affinity matrix optimization problem. For example, the work Tierney et al. (2014) segments data drawn from a sequentially ordered union of subspaces and finds a sparse representation but includes a new penalty term to take care of sequential data. The work Li et al. (2015) exploits temporal information for time series data (e.g., human motion) and designs a temporal Laplacian regularization, which encodes the sequential relationships in time series data. The work Guo et al. (2014) focuses on the hyperspectral data taken from a drill hole and produces stable and continuous segments. The works Wang et al. (2018) propose a graph regularizer to incorporate temporal information to preserve more sequence knowledge into the learned representation. The work Wu et al. (2015) adopts a quadratic normalizer for the data sparse representation to model the correlation among the sparse data coefficients. The work Dimiccoli et al. (2021) introduces a regularization term for this auxiliary data matrix that preserves the local geometrical structure present in the high-dimensional space. The work Wang et al. (2022) learns the support structure representation of sequence, which can extract sufficient information about instances and get the compact structure of sequential data.

We found that the self-representation of the ordered dataset is shown in a block-diagonal form. If each subspace is independent and noise-free, the coefficient matrix $\mathbf{Z}$ learned by spectral-based subspace methods often exhibits a K-block diagonal structure, and K denotes that data is grouped into K classes, i.e.,

$$\mathbf{Z} = \begin{bmatrix} \mathbf{Z}_1 & \cdots & 0 \\ \vdots & \ddots & \vdots \\ 0 & \cdots & \mathbf{Z}_K \end{bmatrix}, \mathbf{Z}_k \in \mathbb{R}^{n_k \times n_k}. \tag{1}$$

The above $\mathbf{Z}$ reveals the true membership of data $\mathbf{X}$. If we apply spectral clustering on the affinity matrix defined as $(|\mathbf{Z}| + |\mathbf{Z}^{\mathrm{T}}|)/2$, then we may get correct clustering. So the block diagonal matrix plays a central role in the analysis of subspace clustering, though there has no "ground-truth" Z (or it is not necessary). We formally give the following definition.

Definition 1.1 (Block Diagonal Property). Given a set of sample vectors $\mathbf{X} = [\mathbf{X}_1, ..., \mathbf{X}_K] = [\mathbf{x}_1, ..., \mathbf{x}_N] \in \mathbb{R}^{D \times N}$ drawn from a union of K subspaces, where $\mathbf{X}_k \in \mathbb{R}^{D \times n_k}$ is a collection of n_k samples drawn from the k-th subspace, and $N = \sum_{k=1}^{K} n_k$. We say that $\mathbf{Z}$ obeys the Block Diagonal Property if $\mathbf{Z}$ is K-block diagonal as in (1), where the nonzero entries $\mathbf{Z}_k$ correspond to only $\mathbf{X}_k$.

Although the ordered subspace clustering methods maintain the successive property of sequential data well, redundant connections exist in the intersection of two subsequences, which will destroy the integrity of a cluster and easily result in the chained partition of the sequence. In addition, when the independent subspace assumption does not hold, the fragile block diagonal structure will be destroyed. The performance of subspace segmentation may be severely degraded. So it is necessary to learn a more specific and direct structure representation (block-diagonal) of a sequence to preserve

both sequential information and efficient connections. Besides, it is significant to learn the block-diagonal composition of the self-representation and identify the clustering structure simultaneously.

In this paper, we propose an effective approach to learning the block-diagonal structure of the self-representation of an ordered dataset. As shown in Figure 1, the proposed method is adopted as a substitute for the second stage of the spectral-based subspace clustering method to improve the performance of all subspace clustering methods on the ordered dataset. In summary, the main contributions of this paper are threefold.

1) We are the first to learn the block-diagonal composition of the self-representation and identify the clustering structure simultaneously and directly. Under the block-diagonal assumption of self-representation, we propose a block-diagonal partition problem with a given affinity matrix.

2) We proposed two partition problems for the uniform and non-uniform density cases. Based on theoretical analysis, an effective method is developed to compute the solution of the proposed two partition problems, and the global optimality will be given by the approach theoretically.

3) We test our method on seven challenging clustering tasks, including face recognization, car detection, motion capture, and video segmentation. The proposed method can be applied to any self-representation given by the existing subspace clustering method, thus improving their performance on the ordered dataset. Experimental result shows that our method can improve 30%$\sim$40% performance on clustering accuracy over the state-of-the-art methods.

2 Review of spectral Clustering

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 a $N \times N$ adjacency matrix (also similarity matrix or weight matrix) of the finite undirected graph $\mathcal{G}$ on N vertices, where the non-diagonal 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 report is not concerned with reflexive connections). The adjacency matrix is symmetric for undirected graphs. In the following, 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 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 partitioning 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 (long distance) and the edges within a group have high weight (short distance). The first requirement for the partitioning 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, usually, only a small group of points is isolated. For this reason, the famous cutting methods, which are widely used in spectral clustering, are introduced: $RadioCut(\{\mathcal{A}_k\}_{k=1}^{K}) = \sum_{k=1}^{K} \frac{cut(\mathcal{A}_k, \bar{\mathcal{A}}_k)}{|\mathcal{A}_k|}$ Hagen & Kahng (1992), and $Ncut(\{\mathcal{A}_k\}_{k=1}^{K}) = \sum_{k=1}^{K} \frac{cut(\mathcal{A}_k, \bar{\mathcal{A}}_k)}{vol(\mathcal{A}_k)}$ Shi & Malik (2000), where $|\mathcal{A}|$ denotes the number of vertices in $\mathcal{A}$, and $vol(\mathcal{A})$ measures the size of $\mathcal{A}$ by the weights of its edges, i.e. $vol(\mathcal{A}_k) = \sum_{i \in \mathcal{A}_k} \sum_{j \in \mathcal{V}} w_{ij}$. As for the second requirement, within-cluster similarity means optimizing $\sum_{k=1}^{K} \sum_{i,j \in \mathcal{A}_k} w_{ij}$. Within-cluster similarity is maximized if $cut(\mathcal{A}_k, \bar{\mathcal{A}}_k)$ is small and $vol(\mathcal{A}_k)$ is big. Therefore, Ncut implements the second criterion. Ncut can be interpreted as cutting through edges rarely transitioned by a random walk. RadioCut maximizes $|\mathcal{A}_k|$ which does not implement this requirement, as the within-cluster similarity is not related to the number of vertices in $\mathcal{A}_k$.

3 Block-Diagonal Structure Learning

For good performance in dealing with complex local correlation and high-dimensional structure of sequential data, representation-based methods have become one of the hot topics for sequential data

clustering, in which subspace clustering is a representative tool. However, instead of using spectral clustering to identify the clustering structure based on the given affinity matrix, we propose a graph partition method, which can not only learn the block-diagonal structure of the affinity matrix but also identify the clustering structure directly. Specifically, a set of $K-1$ indexes is denoted by a calligraphic letter: $\mathcal{T} = \{t_1, t_2, ..., t_{K-1}\} \subset \{1, 2, ..., N\}$, and its cardinal is $|\mathcal{T}|$. We assume that the partition indexes are ordered and there is at least one sample in each cluster such that $t_{k-1} < t_k$. The dummy indexes $t_0 := 0$ and $t_K := N$ are implicitly available. Suppose that the data are segmented by $K-1$ indexes in $\mathcal{T}$. Ideally, the weight matrix $\mathbf{W}$ of the ordered graph should be in a block-diagonal form. Firstly, due to the ordered nature of the data, we only need to care about the relationship between adjacent clusters, i.e., maximizing the sum of the adjacent clusters' self association $sasso(\mathcal{A}_k) + sasso(\mathcal{A}_{k+1})$, where $sasso(\mathcal{A}_k) = vol(\mathcal{A}_k) - cut(\mathcal{A}_k, \bar{\mathcal{A}}_k)$. Due to the partition nature, the self association can be expressed as $sasso(\mathcal{A}_k) = \sum_{i,j\in[t_{k-1}+1,t_k]} w_{ij}$. Secondly, following the maxi-cut requirement, usually only a little group of points is isolated. Thus, an average item is needed to balance the sample number in each cluster. Finally, we find that, for most existing clustering datasets, the density of each cluster is uniform, so, we propose uniform block average cut (UBAcut): $UBAcut(\{\mathcal{A}_k\}_{k=1}^K) = \frac{\sum_{k=1}^K sasso(\mathcal{A}_k)}{\sum_{k=1}^K |\mathcal{A}_k|^2}$. For datasets with different intra-cluster densities, we also propose non-uniform block average cut (NUBAcut): $NUBAcut(\{\mathcal{A}_k\}_{k=1}^K) = \sum_{k=1}^K \frac{sasso(\mathcal{A}_k)}{|\mathcal{A}_k|}$. We propose two new graph cut methods instead of using the famous Ncut and RadioCut. The reason will be given in Section 4.1.

Then, the proposed UBAcut problem, and NUBAcut problem are given by

$$\underset{t_1<t_2<...<t_{K-1}}{\text{maximize}} \quad \sum_{k=1}^{K} f(t_{k-1}, t_k) = \sum_{k=1}^{K} \frac{\sum_{i,j\in[t_{k-1}+1,t_k]} w_{ij}}{t_k - t_{k-1}} \tag{2}$$

and

$$\underset{t_1<t_2<...<t_{K-1}}{\text{maximize}} \quad h(t_1, t_2, ..., t_{K-1}) = \frac{\sum_{k=1}^{K} \sum_{i,j\in[t_{k-1}+1,t_k]} w_{ij}}{\sum_{k=1}^{K} (t_k - t_{k-1})^2} \tag{3}$$

3.1 Non-uniform Block Average Cut

Since there are N^{K-1} combinations of $K-1$ unknown variables $\{t_k\}_{k=1}^{K-1}$, the global optimal solution of Problem (2) can be searched exhaustively with a $\mathcal{O}(N^{K-1})$ complexity. In order to explore a low-complexity approach to solve the problem (2), we give the following theoretical analysis. Consider partitioning a K-component graph matrix into a two-block-diagonal form. Problem (2) can be written as

$$\underset{t}{\text{maximize}} \quad \frac{\sum_{i,j\in[1,t]} w_{ij}}{t} + \frac{\sum_{i,j\in[t+1,N]} w_{ij}}{N-t} \tag{4}$$

Given a K-component self-representation with $K-1$ partition indexes $\{c_k\}_{k=1}^{K-1}$, we suppose the within-cluster weight w_{ij}, $i, j \in [c_{k-1}+1, c_k]$ follow the Gaussian distribution $\mathcal{N}(\mu_k, \sigma^2)$, where μ_k is the average within-cluster weight of the k-th cluster measuring the k-th within-cluster density, and σ^2 is the variance of the noise simulating the uncertainty of the similarity between points. In addition, suppose the without-cluster weight w_{ij}, $i \in [c_{k-1}+1, c_k]$, $j \notin [c_{k-1}+1, c_k]$ follow the zero means Gaussian distribution $\mathcal{N}(0, \sigma^2)$. In order to remove the effect of the random noise on the data, expectation operation is used in the theoretical Analysis. Denote the expectation of the objective function of the problem (4) as $F(t)$. We have the following observations:

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

1) If $K = 1$, then $F(t) = N\mu_1$.

2) If $K \geq 2$, then $F(t)$ increase for $t \leq c_1$ and decrease for $t > c_{K-1}$.

3) If $K \geq 3$, then for the interval $[c_{k-1}+1, c_k]$, $k = 2, ..., K-1$, there are existing α and β to make $F(t)$ decrease for $t \in (c_k, \hat{t})$ and increase for $t \in (\hat{t}, c_{k+1})$, where $\hat{t} = \frac{N\alpha + N\sqrt{\alpha\beta}}{(\alpha-\beta)}$ or $\hat{t} = \frac{N\alpha - N\sqrt{\alpha\beta}}{(\alpha-\beta)}$.

Proof. See Appendix A □

Theorem 3.1 illustrate that one of the true partition $\{c_k\}_{k=1}^{K-1}$ will maximize $F(t)$. Denote the bounded function as $F_B(t_i, t, t_j) = \mathbb{E}[f(t_i, t) + f(t, t_j)]$, $0 < t_i < t_j < N$. Obviously, $F_B(0, t, N) = F(t)$. Then, we also have the following corollary.

Corollary 3.1.1. For any interval $(t_i, t_j] \subseteq (0, N]$, the function $F_B(t_i, t, t_j)$ has the same property as the function $F(t)$:

1) If none c_k in $(t_i + 1, t_j)$, then $F_B(t_i, t, t_j) = (t_j - t_i)\mu$, where μ is the mean of weight w_{ij}, $i, j \in [t_i + 1, t_j]$.

2) If there existing partition indexes $c_p, c_{p+1}, ..., c_q$, $1 \leq p \leq q \leq K-1$ in $(t_i + 1, t_j)$, then one of them will maximize $F_B(t_i, t, t_j)$.

Denote the binary paritition loss function as

$$\Phi(t_i, t_j) = \max_{t \in (t_i+1, t_j)} F_B(t_i, t, t_j) - \mathbb{E}\{f(t_i, t_j)\}$$

Corollary 3.1.1 indicates that $\Phi(\tau_1, \tau_2) > \Phi(\tau_3, \tau_4)$ if there existing c_k in $(\tau_1 + 1, \tau_2)$, and there are none c_k in $(\tau_3 + 1, \tau_4)$. In addition, one of $\{c_K\}_{k=1}^{K-1}$ in the interval $(\tau_1 + 1, \tau_2)$ will maximize $\Phi(\tau_1, \tau_2)$ for any τ_1, and τ_2.

Corollary 3.1.2. Given an ordered partition index set $\{t_k\}_{k=1}^{K-1}$, if any t_m, $m \in [1, K-1]$, satisfies $m = \underset{k \in [1, K-1]}{\operatorname{argmax}} \Phi(T_{k-1}, T_k)$, and $t_m = \underset{t_1 \in (T_{m-1}+1, T_m)}{\operatorname{argmax}} F_B(T_{k-1}, t, T_k)$, where $T = \{0, \{t_k\}_{k=1, t \neq m}^{K-1}, N\}$, then we can say $\{t_k\}_{k=1}^{K-1}$ maximize $F(t_1, t_2, ..., t_{K-1})$, and $\{t_k\}_{k=1}^{K-1} = \{c_k\}_{k=1}^{K-1}$.

Corollary 3.1.1 illustrates that only one of $\{c_k\}_{k=1}^{K-1}$ can satisfies the condition in Corollary 3.1.2. Thus, given any $\{t_k\}_{k=1}^{K-1}$, if each item in $\{t_k\}_{k=1}^{K-1}$ satisfies the condition in Corollary 3.1.2, we can say $\{t_k\}_{k=1}^{K-1} = \{c_k\}_{k=1}^{K-1}$.

The expectation analysis of the objective function gives us the intuation to design an alternative partition updating algorithm in order to obtain the solution of the problem (2). We first remove t_m in $\{t_k\}_{k=1}^{K-1}$, and assign the ordered $\{t_k\}_{k=1}^{K-1}$ to $\{\tau_k\}_{k=1}^{K-2}$. We hope to inset one index into $\{\tau_k\}_{k=1}^{K-2}$ to make the objective function value of the problem (2) becoming become larger. Since the additional structure of the problem (2), we only need to consider inserting the partition index in each interval $(\tau_{k-1} + 1, \tau_k)$, $k \in [1, K-1]$. The vector $\mathbf{g}$ is defined to save the optimal insert partition index for each interval $(\tau_{k-1} + 1, \tau_k)$, then the global optimal partition index is picked, and replace t_m in $\{t_k\}_{k=1}^{K-1}$. Alg. 1 shows the detail of the proposed scheme. The whole updating process will be terminated until satisfying Corollary 3.1.2, which illustrates the condition of reaching the optimal partition.

Alg.1 will converge to the true partition index $\{c_k\}_{k=1}^{K-1}$ if each cluster is far away from its adjacent two clusters, the density of each cluster is uniform and the noise is negligible. As we all know, severe noise can lead to the situation that the true partition index $\{c_k\}_{k=1}^{K-1}$ can not maximize the objective function. Although exhaustive searching or dynamic programming-based searching can obtain the solution which can maximize the objective function, the complexity is too high, e.g., exhaustive searching needs $\mathcal{O}(N^{K-1})$, and dynamic programming-based searching needs $\mathcal{O}(KN^2)$. However, our proposed method only need $\mathcal{O}(KN)$, and converge to $\{c_k\}_{k=1}^{K-1}$ with a high probability.

3.2 Uniform Block Average Cut

Given a K-component self-representation with $K-1$ partition indexes $\{c_k\}_{k=1}^{K-1}$, we suppose the within-block weight w_{ij}, $i, j \in [t_{k-1} + 1, t_k]$, $\forall\, k \in [1, K]$, follow Gaussian distribution, i.e.,

$N(\mu, \sigma^2)$, and the without-block weight w_{ij}, $i \in [t_{k-1}+1, t_k]$, $j \notin [t_{k-1}+1, t_k]$, $\forall\, k \in [1, K]$, follow zero means Gaussian distribution, i.e., $N(0, \sigma^2)$, we have the following observation. Denote the expectation of the objective function of problem (3) as $H(t_1, t_2, ..., t_{K-1}) = \mathbb{E}[h(t_1, t_2, ..., t_{K-1})]$.

Theorem 3.2. Given $K-2$ ordered partition index $\{\tau_k\}_{k=1}^{K-2}$, $\tau_0 = 0$, and $\tau_{K-1} = N$, for any interval $(\tau_{m-1}+1, \tau_m)$, $m \in [1, K-1]$, one of the true partition index $\{c_k\}_{k=1}^{K-1}$ in the interval $(\tau_{m-1}+1, \tau_m)$ will maximizing $H(\tau_1, \tau_2, ..., \tau_{m-1}, t, \tau_m, ..., \tau_{K-2})$.

Proof. See Appendix B □

Corollary 3.2.1. Given an ordered partition index set $\{t_k\}_{k=1}^{K-1}$, if any t_m, $m \in [1, K-1]$, satisfies

$$m = \underset{k \in [1, K-1]}{\operatorname{argmax}} \left\{ \max_{t \in (T_k+1, T_{k+1})} H(T_2, T_3, ..., T_k, t, T_{k+1}, ..., T_{K-2}) \right\},$$
$$t_m = \underset{t \in (T_m+1, T_{m+1})}{\operatorname{argmax}} H(T_2, T_3, ..., T_k, t, T_{k+1}, ..., T_{K-2})$$

where $T = \{0, \{t_k\}_{k=1, t \neq m}^{K-1}, N\}$, then we can say $\{t_k\}_{k=1}^{K-1}$ maximize $H(t_1, t_2, ..., t_{K-1})$, and $\{t_k\}_{k=1}^{K-1} = \{c_k\}_{k=1}^{K-1}$.

Theorem 3.2 illustrates that only one of $\{c_k\}_{k=1}^{K-1}$ can satisfies the condition in Corollary (3.2.1). Thus, given any $\{t_k\}_{k=1}^{K-1}$, if each item in $\{t_k\}_{k=1}^{K-1}$ satisfies the condition in Corollary (3.2.1), we can say $\{t_k\}_{k=1}^{K-1} = \{c_k\}_{k=1}^{K-1}$.

We use the updating progress, which is the same as NUBAcut. The only difference is the updating of t_m in $\{t_k\}_{k=1}^{K-1}$. The convergence is still guaranteed by Corollary 3.2.1.

Algorithm 1 Partition Index Optimization.

Input: the graph $\mathbf{W}$, and an initial partition index set $\{t_k\}_{k=1}^{K-1}$
Repeat
  for $m = 1 : K-1$ **do**
    Sort the elements $\{t_k\}_{k=1, t \neq m}^{K-1}$ in ascending order, and assign to $\{\tau_k\}_{k=1}^{K-2}$. Initialize two $(K-1)$-length empty vector $\mathbf{l}$ and $\mathbf{g}$. $\tau_0 = 0$, and $\tau_{K-1} = N$
    for $k = 1 : K-1$ **do**
      if NUBAcut
        $l_k = \underset{t_1 \in (\tau_{k-1}+1, \tau_k)}{\operatorname{argmax}} f(\tau_{k-1}, t) + f(t, \tau_k)$
        $g_k = f(\tau_{k-1}, l_k) + f(l_k, \tau_k) - f(\tau_{k-1}, \tau_k)$
      else if UBAcut
        $l_k = \underset{t \in (\tau_{k-1}+1, \tau_k)}{\operatorname{argmax}} h(\tau_1, \tau_2, ..., \tau_{k-1}, t, \tau_k, ..., \tau_{K-2})$
        $g_k = h(\tau_1, \tau_2, ..., \tau_{k-1}, l_k, \tau_k, ..., \tau_{K-2})$
      end if
    end for
    $\hat{k} = \underset{k}{\operatorname{argmin}}\, g_k$
    $t_m = l_{\hat{k}}$
  end for
until $\{t_k\}_{k=1}^{K-1}$ is unchanged
Output: $\{t_k\}_{k=1}^{K-1}$

4 EXPERIMENTS

In this section, extensive experiments are performed on different types of datasets to demonstrate the effectiveness and superiority of the proposed method in comparison with the state-of-the-art clustering methods. First, several synthetic graphs with different clustering densities and noise levels are generated to evaluate the proposed method for the block-diagonal partition task. Secondly, utilizing six famous ordered clustering datasets, we replace the spectral clustering operation of the famous state-of-the-art subspace clustering methods with our proposed BDSL algorithm to show the effectiveness of BDSL.

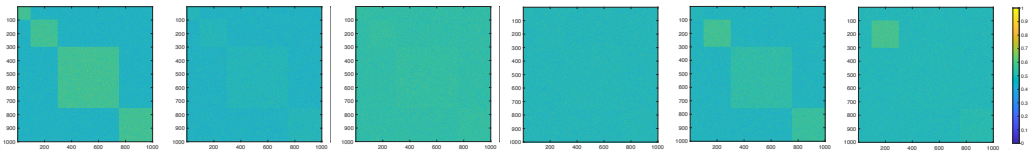


Figure 2: Example synthetic graphs.

Cluster Density $[\mu_1, \mu_2, \mu_3, \mu_4]$	Noise Variance σ^2	RadioCut (Spectral)	NCut (Spectral)	RadioCut (BDSL)	NCut (BDSL)	NUBAcut	UBAcut
$[1, 1, 1, 1]$	1	82.53	100	100	100	100	100
	25	24.61	27.63	74.82	78.92	99.24	100
	100	25.65	26.65	45.20	73.50	98.79	99.80
	400	24.42	27.42	45.10	52.00	97.90	99.60
$[0.1, 0.9, 0.5, 0.7]$	1	72.25	80.25	55.14	68.71	100	100
	25	21.67	26.76	50.52	54.60	99.80	99.87
	100	23.12	27.19	46.70	52.30	99.80	99.83
	400	22.87	26.82	45.10	50.10	98.50	98.54
$[0.01, 0.9, 0.05, 0.2]$	1	57.73	63.76	45.95	91.24	100	72.71
	25	24.58	26.54	45.10	73.80	99.90	69.90
	100	21.54	26.59	45.30	54.80	99.70	68.25
	400	23.15	26.13	45.60	52.50	65.80	65.20

Table 1: Clustering Accuracy Rate [%] on the synthetic graph.

4.1 Synthetic Experiment

In order to verify the validity of the algorithm and the correctness of the theory, we generate the graphs with different clustering densities and noise levels. As shown in Figure 2, we consider four clusters with different cluster number (i.e., 100, 200, 450, 250), cluster densities (e.g., $[\mu_1, \mu_2, \mu_3, \mu_4] = [1, 1, 1, 1], [0.1, 0.9, 0.5, 0.7], [0.01, 0.9, 0.05, 0.2]$), and noise levels (e.g., $\sigma^2 = 1, 25, 100, 400$). At the same time, the value in the graph is linearly normalized to $[0, 1]$. As shown in Figure 2, the partition task becomes more and more difficult with the increase of the noise variance.

We compare the proposed NUBAcut and UBAcut with RadioCutHagen & Kahng (1992) and NcutShi & Malik (2000). Since NCut can be written as $K - \sum_{k=1}^{K} \frac{\sum_{i,j\in[t_{k-1}+1,t_k]} w_{ij}}{\sum_{i\in[t_{k-1}+1,t_k],j\in[1,N]} w_{ij}}$, minimizing $Ncut(\mathcal{A}_1, \mathcal{A}_2, ..., \mathcal{A}_K)$ is equivalent to maximizing $\sum_{k=1}^{K} \frac{\sum_{i,j\in[t_{k-1}+1,t_k]} w_{ij}}{\sum_{i\in[t_{k1}+1,t_k],j\in[1,N]} w_{ij}}$ with fixed K. Since RadioCut can be expressed as $\sum_{k=1}^{K} \frac{\sum_{i\in[t_{k-1}+1,t_k],j\in[1,N],j\notin[t_{k-1}+1,t_k]} w_{ij}}{t_k - t_{k-1}}$, minimizing $RadioCut(\mathcal{A}_1, \mathcal{A}_2, ..., \mathcal{A}_K)$ is equivalent to maximizing $-\sum_{k=1}^{K} \frac{\sum_{i\in[t_{k-1}+1,t_k],j\in[1,N],j\notin[t_{k-1}+1,t_k]} w_{ij}}{t_k - t_{k-1}}$. Thus, if we define $f(t_{k-1}, t_k)$ in problem (2) as $\frac{\sum_{i,j\in[t_{k-1}+1,t_k]} w_{ij}}{\sum_{i\in[t_{k-1}+1,t_k],j\in[1,N]} w_{ij}}$ for Ncut, or $-\frac{\sum_{i\in[t_{k-1}+1,t_k],j\in[1,N],j\notin[t_{k-1}+1,t_k]} w_{ij}}{t_k - t_{k-1}}$ for RadioCut, the NCut and RadioCut can also be used in the proposed BDSL algorithm. These two methods are named as RadioCut (BDSL) and NCut (BDSL). In addition, in order to show the poor performance of the traditional spectral clustering methods, we also use RadioCut (Spectral) and NCut (Spectral) as the baseline. Table 1 illustrates the performance of the above methods on the synthetic graph. We have the following observation:

1) Comparing NUBAcut, UBAcut, RadioCut (BDSL) and NCut (BDSL) with RadioCut (Spectral) and NCut (Spectral), we found that the proposed BDSL algorithm is more effective on the ordered dataset than the traditional spectral clustering methods. In addition, the performance of NCut is always better than that of RadioCut. Even for the graph with uniform density and low-level noise, the RadioCut only has an 82.53% clustering accuracy.

Method	MAD Dataset		Keck Dataset		Weizmann Dataset		Animation Dataset		Ballet Dataset		BU-4DFE Dataset	
	Acc	NMI	Acc	NMI	Acc	NMI	Acc	NMI	Acc	NMI	Acc	NMI
Approaches Designed for General Dataset												
LRR	0.2397	0.2249	0.4297	0.4862	0.3638	0.4382	0.7855	0.7632	0.6799	0.7123	0.6467	0.6723
LRR+BDSL	0.5834	0.6843	0.6042	0.6643	0.5432	0.5042	0.9321	0.9423	0.9142	0.9024	0.8953	0.8842
PSCN	0.3208	0.3412	0.4192	0.3245	0.5321	0.5125	0.8241	0.8624	0.6210	0.6732	0.7224	0.7032
PSCN+BDSL	0.6234	0.6743	0.6245	0.6024	0.6783	0.6642	0.9742	0.9623	0.9221	0.9053	0.9153	0.9035
SSC	0.3817	0.4758	0.3137	0.3858	0.4576	0.6009	0.8842	0.8324	0.6042	0.6242	0.6723	0.6523
SSC+BDSL	0.6632	0.6924	0.6023	0.5923	0.6208	0.6342	0.9642	**0.9742**	0.9421	0.9253	0.9362	0.9143
LSR	0.3979	0.3667	0.4894	0.4548	0.5091	0.5093	0.7632	0.7842	0.5973	0.6412	0.6611	0.7214
LSR+BDSL	0.6345	0.6324	0.6734	0.6412	0.6453	0.6542	0.9735	0.9643	0.9042	0.9142	0.9053	0.8953
L2-Graph	0.4124	0.5214	0.4432	0.4102	0.5232	0.6232	0.7198	0.7412	0.7244	0.7132	0.7122	0.7232
L2-Graph+BDSL	0.6743	0.7142	0.6834	0.6724	0.6743	0.6632	0.8935	0.9042	0.9242	0.9157	0.8953	0.8853
iPursuit	0.5211	0.5124	0.4623	0.4353	0.5324	0.6421	0.8932	0.8832	0.7823	0.8044	0.8314	0.8512
iPursuit+BDSL	0.7142	0.7435	0.6724	0.6642	0.6245	0.6324	0.9424	0.9353	0.9425	0.9253	0.9255	0.9015
FGNSC	0.4825	0.5924	0.5124	0.4824	0.5488	0.6524	0.9211	0.9324	0.8242	0.8421	0.8721	0.8424
FGNSC+BDSL	0.6824	0.7943	**0.6933**	**0.7024**	0.6943	0.7042	**0.9743**	0.9543	0.9521	0.9527	0.9415	0.9253
BDRZ	0.6042	0.5624	0.5024	0.4531	0.5201	0.6297	0.9053	0.9252	0.8421	0.8632	0.8912	0.8843
BDRZ+BDSL	0.7142	0.7842	0.6534	0.6724	0.6843	0.6921	0.9663	0.9732	0.9532	0.9461	0.9462	0.9515
ORGEN	0.5925	0.5543	0.4822	0.4721	0.5823	0.6832	0.8842	0.8932	0.8632	0.8432	0.8814	0.9043
ORGEN+BDSL	0.7534	**0.8142**	0.6843	0.6425	**0.7144**	0.7242	0.9643	0.9532	**0.9632**	**0.9572**	**0.9735**	**0.9621**
RSSC	0.5204	0.4756	0.4324	0.4098	0.5053	0.6424	0.7832	0.7932	0.7823	0.8124	0.8215	0.8034
RSSC+BDSL	**0.7842**	0.7942	0.6234	0.6742	0.7043	**0.7344**	0.9242	0.9351	0.9234	0.9066	0.8924	0.9015
Approaches Designed for Ordered Dataset												
OSC	0.4327	0.5589	0.4393	0.4931	0.5216	0.5047	0.8923	0.9042	0.8724	0.8842	0.9244	0.9214
OSC+BDSL	0.6842	0.8433	0.6943	0.7242	0.7435	0.7742	0.9423	**0.9622**	0.9714	0.9653	0.9526	0.9416
TSC	0.5556	0.7721	0.4781	0.5329	0.6111	0.6199	0.9142	0.8823	0.9022	0.8924	0.9024	0.9043
TSC+BDSL	0.7342	0.8843	0.7042	0.6842	0.7743	0.8123	0.9633	0.9523	0.9623	0.9469	0.9242	0.9316
LTS	0.5736	0.8202	0.5395	0.5049	0.6208	0.6509	0.9233	0.9023	0.9142	0.9124	0.9453	0.8943
LTS+BDSL	0.7623	**0.8934**	**0.7345**	**0.7524**	**0.8134**	0.7942	**0.9653**	0.9525	0.9623	0.9528	0.9526	**0.9616**
QOSC	0.5234	0.7834	0.4923	0.4833	0.5832	0.6242	0.9043	0.8924	0.9412	0.9246	0.9124	0.9231
QOSC+BDSL	0.6932	0.8764	0.6843	0.6942	0.7932	0.7824	**0.9653**	0.9345	0.9773	0.9617	0.9267	0.9415
SpatSC	0.5034	0.7134	0.4421	0.4023	0.5432	0.5024	0.8924	0.9042	0.9024	0.8546	0.9354	0.9052
SpatSC+BDSL	0.7142	0.8244	0.6712	0.6642	0.7742	0.7242	0.9253	0.9353	0.9456	0.9278	0.9563	0.9416
GCRL	0.5732	0.7421	0.4723	0.4524	0.6124	0.5823	0.8832	0.8934	0.9242	0.8954	0.8734	0.9104
GCRL+BDSL	**0.7743**	**0.8934**	0.6942	0.6242	0.7824	**0.8142**	0.9035	0.9152	0.9742	0.9628	0.9435	0.9318
SSRC	0.4732	0.6923	0.5012	0.4823	0.6431	0.5621	0.8924	0.9143	0.8924	0.9255	0.8943	0.8521
SSRC+BDSL	0.7242	0.8452	0.6624	0.6542	0.8042	0.7624	0.9134	0.9273	**0.9853**	**0.9793**	**0.9627**	0.9571

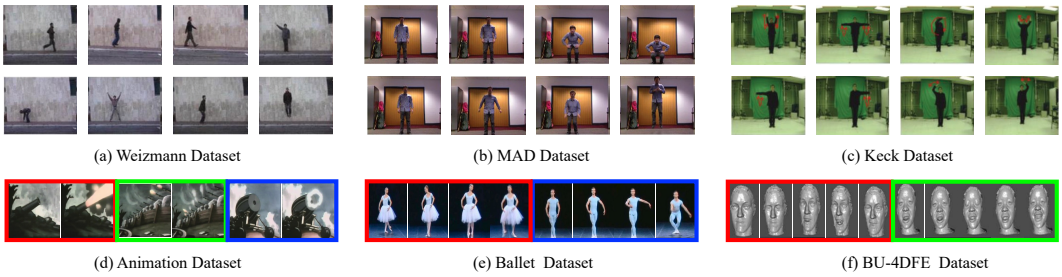
Table 2: Quantitative (the clustering accuracies and NMI) comparisons with state-of-the-art methods on the six ordered datasets.

2) Comparing NUBAcut and UBAcut with RadioCut (BDSL) and NCut (BDSL), although they all run based on the proposed BDSL algorithm, the proposed two graph cutting methods show better performance on the noisy graph.

3) Comparing NUBAcut with UBAcut, UBAcut is a little better than NUBAcut on the uniform density graph, i.e., $[\mu_1, \mu_2, \mu_3, \mu_4] = [1, 1, 1, 1]$. On the little bit non-uniform density graph, i.e., $[\mu_1, \mu_2, \mu_3, \mu_4] = [0.1, 0.9, 0.5, 0.7]$, their performances are similar. However, UBAcut performs worse than NUBAcut for severe non-uniform density graph, i.e., $[\mu_1, \mu_2, \mu_3, \mu_4] = [0.01, 0.9, 0.05, 0.2]$.

4.2 Experiments on sequential high-dimensional datasets

There are many subspace clustering methods designed for the ordered dataset, including OSCTierney et al. (2014), submoTSCLi et al. (2015), LTSWang et al. (2018), QOSCWu et al. (2015), SpatSCGuo et al. (2014), GCRLDimiccoli et al. (2021), and SSRCWang et al. (2022). However, all of these methods focus on the construction of the self-representation matrix, and we are the first to learn the block-diagonal structure of the self-representation directly. In order to evaluate the fact that our proposed strategy can improve the performance of the existing subspace clustering method on the ordered dataset. We use the state-of-the-arts to generate the self-represented affinity matrix, and then, apply the proposed BDSL on them. In another word, we replace the spectral clustering with proposed BDSL. We compare the traditional version of these state-of-the-arts with the improved version. In addition, our proposed BDSL is also effective on the subspace clustering methods designed for general datasets. In order to prove this, we also apply the proposed BDSL algorithm on the method which is designed for the general dataset clustering task, including LRRLiu et al. (2012), PSCNCui et al. (2021), SSCXu et al. (2015), LSRLu et al. (2012), L2-GraphPeng et al. (2016), iPursuitRahmani & Atia (2017), FGNSCYang et al. (2019), BDRZLu et al. (2018), ORGENYou et al. (2016), and RSSCXu et al. (2015).



(a) Weizmann Dataset (b) MAD Dataset (c) Keck Dataset

(d) Animation Dataset (e) Ballet Dataset (f) BU-4DFE Dataset

Figure 3: Example frames of six high-dimensional datasets.

We conduct the experiments on the famous video segmentation or motion segmentation dataset, including MAD DatasetHuang et al. (2014), Keck DatasetJiang et al. (2012), Weizmann DatasetGorelick et al. (2007), Animation DatasetTierney et al. (2014), Ballet DatasetFathi & Mori (2008), and BU-4DFE DatasetZhang et al. (2013). Figure **??** shows the frame samples of these datasets. The detail of these datasets is introduced as follows: 1) The Weizmann dataset contains 90 video sequences include 10 different actions performed by nine subjects in outdoor environments. The resolution is 180×144 with the frame rate of 50 fps. The subjects preform ten regular actions including run, walk, skip and so on. 2) The MAD dataset contains multi-modal actions of 20 subjects recorded by Microsoft Kinect in three formats. First is regular RGB image in resolution of 640×480. Second is 3D depth image. The third is human skeleton information. Each subject performs 35 actions in two different indoor environments. 3) The Keck dataset consists of 14 different actions from military signals. The resolution is 640×480. Three subjects preforms the 14 gestures and each action is repeatedly preformed 3 times by each subject. Thus, $3\times3\times14 = 126$ human action videos are obtained. 4) The animation sequences are around 10 seconds in length (approximately 300 frames) containing three scenes each. There are 24 sequences from the video. The scenes to be segmented can contain significant translation and morphing of objects within the scene and sometimes camera or perspective changes. Scene changes (or keyframes) were collected manually to form ground truth data. 5) The Ballet dataset used here contains 44 videos collected from an instructional ballet digital video disk Fathi & Mori (2008). The data set consists of eight complex action patterns performed by three subjects. These actions include: " left-to-right hand opening", " right-to-left hand opening", " standing hand opening", " leg swinging", " jumping"," turning", " hopping", and " standing still". 6) The BU-4DFE database consists of 101 subjects (58 female and 43 male) with a variety of ethnic/racial ancestries, including White, Black, East-Asian, Middle-East Asian, Indian, and Hispanic/Latino. Each subject was requested to perform six basic expressions (anger, disgust, happiness, fear, sadness and surprise) and its face image sequences and dynamic 3-D face shapes are captured simultaneously by a specific device.

We tune the parameters for the comparison methods to achieve the best results. The experiments are repeated ten times, and the mean clustering accuracy and NMI are computed to show the performance. There is no parameter in our proposed BDSL algorithm. Since the graph given by the comparisons is almost density uniform, we just use UBAcut to identify the clustering structure. The experimental result is shown in Table 2. Each traditional method's performance can be improved by our BDSL, and the clustering accuracy is improved by roughly 30% 40%.

5 Conclusions

In this work, we proposed an effective graph partition method that can be applied to each spectral-based subspace clustering method and improve their performance on the ordered dataset. Our proposed method can also be applied to the non-ordered clustering job, which can be achieved by using an ordering pre-processing method (e.g., DBSCAN Ester et al. (1996), OPTICSAnkerst et al. (1999)) to make the data ordered. In addition, our method is not limited to the subspace clustering task because the common distance measurement can also generate the block-diagonal form graph.

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.

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.

Mariella Dimiccoli, Lluís Garrido, Guillem Rodriguez-Corominas, and Herwig Wendt. Graph constrained data representation learning for human motion segmentation. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pp. 1460–1469, 2021.

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.

Alireza Fathi and Greg Mori. Action recognition by learning mid-level motion features. In *IEEE conference on computer vision and pattern recognition*, pp. 1–8, 2008.

Lena Gorelick, Moshe Blank, Eli Shechtman, Michal Irani, and Ronen Basri. Actions as space-time shapes. *IEEE transactions on pattern analysis and machine intelligence*, 29(12):2247–2253, 2007.

Yi Guo, Junbin Gao, and Feng Li. Spatial subspace clustering for drill hole spectral data. *Journal of Applied Remote Sensing*, 8(1):083644, 2014.

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.

Dong Huang, Shitong Yao, Yi Wang, and Fernando De La Torre. Sequential max-margin event detectors. In *European conference on computer vision*, pp. 410–424, 2014.

Zhuolin Jiang, Zhe Lin, and Larry Davis. Recognizing human actions by learning and matching shape-motion prototype trees. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 34(3):533–547, 2012.

Sheng Li, Kang Li, and Yun Fu. Temporal subspace clustering for human motion segmentation. In *Proceedings of the IEEE international conference on computer vision*, pp. 4453–4461, 2015.

Guangcan Liu, Zhouchen Lin, Shuicheng Yan, Ju Sun, Yong Yu, and Yi Ma. Robust recovery of subspace structures by low-rank representation. *IEEE transactions on pattern analysis and machine intelligence*, 35(1):171–184, 2012.

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.

Xi Peng, Zhiding Yu, Zhang Yi, and Huajin Tang. Constructing the l2-graph for robust subspace learning and subspace clustering. *IEEE transactions on cybernetics*, 47(4):1053–1066, 2016.

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.

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

Stephen Tierney, Junbin Gao, and Yi Guo. Subspace clustering for sequential data. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 1019–1026, 2014.

Lichen Wang, Zhengming Ding, and Yun Fu. Learning transferable subspace for human motion segmentation. In *Proceedings of the AAAI conference on artificial intelligence*, volume 32, 2018.

Xiumei Wang, Dingning Guo, and Peitao Cheng. Support structure representation learning for sequential data clustering. *Pattern Recognition*, 122:108326, 2022.

Fei Wu, Yongli Hu, Junbin Gao, Yanfeng Sun, and Baocai Yin. Ordered subspace clustering with block-diagonal priors. *IEEE transactions on cybernetics*, 46(12):3209–3219, 2015.

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

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.

Xing Zhang, Lijun Yin, Jeffrey F Cohn, Shaun Canavan, Michael Reale, Andy Horowitz, and Peng Liu. A high-resolution spontaneous 3d dynamic facial expression database. In *IEEE international conference and workshops on automatic face and gesture recognition (FG)*, pp. 1–6, 2013.

6 APPENDIX

6.1 A PROOF OF THEOREM 3.1

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

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

$$\begin{aligned} F(t) &= \frac{t^2\mu_1}{t} + \frac{(c_1 - t)^2\mu_1 + (c_2 - c_1)^2\mu_2 + ... + (N - c_{K-1})^2\mu_K}{N - t} \\ &= t\mu_1 + \frac{(c_1 - t)^2\mu_1 + (N - c_1)^2 b(2, K)}{N - t} \\ &= \frac{(N - 2c_1)t\mu_1 + c_1^2\mu_1 + (N - c_1)^2 b(2, K)}{N - t} \end{aligned} \tag{5}$$

where the mean weight of the ith, $i \in \{2, 3, ..., K\}$ cluster with zero covariance is $b(2, K) = \frac{(c_2-c_1)^2\mu_2+...+(N-c_{K-1})^2\mu_K}{(N-c_1)^2}$. Since $F^{'}(t) = \frac{(N-2c_1)\mu_1 N+c_1^2\mu_1+(N-c_1)^2 b(2,K)}{(N-t)^2} > 0$, $F(t)$ will increase with the increase of $t \in (0, c_1]$. For the symmetric case $t \in (c_{K-1}, N]$, $F(t)$ will decrease with the increase of $t \in (c_{K-1}, N]$.

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

$$\begin{aligned} F(t) = &\frac{c_1^2\mu_1 + ..., + (c_{k-1} - c_{k-2})^2 \mu_{k-1} + (t - c_{k-1})^2\mu_k}{t} \\ &+ \frac{(c_{l+2} - c_k)^2 \mu_{l+2} + ..., + (N - c_{K-1})^2 \mu_K + (c_k - t)^2\mu_k}{N - t} \end{aligned}$$

Then

$$F^{'}(t) = \frac{At^2 + Bt + C}{C_0}$$

where $A = \alpha - \beta$, $B = -2N\alpha$, $C_0 > 0$, $C = N^2\alpha$, $\alpha = c_1^2\mu_1 + ..., + (c_{k-1} - c_{k-2})^2 \mu_{k-1} + c_{k-1}^2\mu_k$, and $\beta = (N - c_k)^2\mu_k + (c_{k+1} - c_k)^2 \mu_{k+1} + ..., + (N - c_{K-1})^2 \mu_K$.

If $\alpha < \beta$, then $c_k < \frac{N\alpha+N\sqrt{\alpha\beta}}{(\alpha-\beta)}$, and $F(t)$ decrease in (c_{k-1}, t_0), and increase in (t_0, c_k), where $t_0 = \frac{N\alpha-N\sqrt{\alpha\beta}}{(\alpha-\beta)}$. If $\alpha > \beta$, then $c_{k-1} > \frac{N\alpha-N\sqrt{\alpha\beta}}{(\alpha-\beta)}$, and $F(t)$ decrease in (c_{k-1}, t_0), and increase in (t_0, c_k), where $t_0 = \frac{N\alpha+N\sqrt{\alpha\beta}}{(\alpha-\beta)}$. Thus, $t = \frac{N\alpha+N\sqrt{\alpha\beta}}{(\alpha-\beta)}$ or $t = \frac{N\alpha-N\sqrt{\alpha\beta}}{(\alpha-\beta)}$ will minimize $F(t)$ for $t \in (c_{k-1}, c_k]$.

A Proof of Theorem 3.2

Consider the true partition index $\{c_k\}_{k=1}^{K-1}$, $c_0 = 0$, and $c_K = N$. Fix $K - 2$ partition index $\{\tau_k\}_{k=1}^{K-2}$, $\tau_0 = 0$, and $\tau_{K-1} = N$, for the interval (τ_{m-1}, τ_m), $m \in [1, K - 1]$.

Case 1: suppose there are no c_k in (τ_{m-1}, τ_m), suppose $c_{l-1} < \tau_{m-1}, \tau_m < c_l$, $l \in [1, K]$, we have

$$F(t) = \frac{(t - \tau_{m-1})^2\mu + (\tau_m - t)^2\mu + L_1}{(t - \tau_{m-1})^2 + (\tau_m - t)^2 + L_2}$$

Then

$$F^{'}(t) = \frac{Bt + C}{[(t - \tau_{m-1})^2 + (\tau_m - t)^2 + L_2]^2}$$

where $B = L_2\mu - L_1$, and $C = \frac{\tau_{m-1}+\tau_m}{2}(L_1 - L_2\mu)$. Since $\frac{L_1}{L_2} < \mu$, we have $B > 0$. Thus, $F(t)$ decrease in (τ_{m-1}, t_0), and increase in (t_0, τ_m), where $t_0 = \frac{\tau_{m-1}+\tau_m}{2}$.

Case 2: suppose there are at least one c_k in (τ_{m-1}, τ_m), e.g., $\tau_{m-1} < c_l < ... < c_r < \tau_m, r-l \geq 0$. Consider the interval (τ_{m-1}, c_l), (c_{l+1}, c_{l+2}),...,(c_r, τ_m).

1) For the interval $t \in (\tau_{m-1}, c_l)$

$$F(t) = \frac{(t-\tau_{m-1})^2\mu + (c_l - t)^2\mu + L_1}{(t-\tau_{m-1})^2 + (\tau_m - t)^2 + L_2}$$

Then

$$F^{'}(t) = \frac{At^2 + Bt + C}{\left[(t-\tau_{m-1})^2 + (\tau_m - t)^2 + L_2\right]^2}$$

where $A = (c_l - \tau_m)\mu < 0$, $B = (L_2 - L_1/\mu - c_l^2 + \tau_m^2)\mu$, and $C = \frac{1}{2}\mu\left[(L_1/\mu + \tau_{m-1}^2 + c_l^2)(\tau_{m-1} + \tau_m) - (L_2 + \tau_{m-1}^2 + \tau_m^2)(\tau_{m-1} + c_l)\right]$. Since $c_l < \frac{-B-\sqrt{B^2-4AC}}{2A}$, $F(t)$ decrease in (τ_{m-1}, t_0), and increase in (t_0, c_l), $F(c_l) > F(\tau_{m-1})$, where $t_0 = \frac{-B-\sqrt{B^2-4AC}}{2A}$.

2) For the interval $t \in (c_r, \tau_m)$

$$F(t) = \frac{(t-c_r)^2\mu + (\tau_m - t)^2\mu + L_1}{(t-\tau_{m-1})^2 + (\tau_m - t)^2 + L_2}$$

Then

$$F^{'}(t) = \frac{At^2 + Bt + C}{\left[(t-c_r)^2 + (\tau_m - t)^2 + L_2\right]^2}$$

where $A = (c_l - \tau_{m-1})\mu > 0$, $B = (L_2 - L_1/\mu - c_r^2 + \tau_{m-1}^2)\mu$, and $C = \frac{1}{2}\mu\left[(L_1/\mu + \tau_m^2 + c_r^2)(\tau_{m-1} + \tau_m) - (L_2 + \tau_{m-1}^2 + \tau_m^2)(\tau_m + c_r)\right]$. Since $c_r > \frac{-B+\sqrt{B^2-4AC}}{2A}$, $F(t)$ decrease in (c_r, t_0), and increase in (t_0, τ_m), $F(c_r) > F(\tau_m)$, where $t_0 = \frac{-B+\sqrt{B^2-4AC}}{2A}$.

3) For the middle interval (c_{l+1}, c_{l+2}),

$$F(t) = \frac{(t-c_{l+1})^2\mu + (c_{l+2} - t)^2\mu + L_1}{(t-\tau_{m-1})^2 + (\tau_m - t)^2 + L_2}$$

Then

$$F^{'}(t) = \frac{At^2 + Bt + C}{\left[(t-c_r)^2 + (\tau_m - t)^2 + L_2\right]^2}$$

where $A = (c_{l+1} + c_{l+2} - \tau_{m-1} - \tau_m)\mu$, $B = (L_2 - L_1/\mu - c_{l+1}^2 - c_{l+2}^2 + \tau_{m-1}^2 + \tau_m^2)\mu$, and $C = \frac{1}{2}\mu\left[(L_1/\mu + c_{l+1}^2 + c_{l+2}^2)(\tau_{m-1} + \tau_m) - (L_2 + \tau_{m-1}^2 + \tau_m^2)(c_{l+1} + c_{l+2})\right]$. If $A < 0$, then $c_{l+2} < \frac{-B+\sqrt{B^2-4AC}}{2A}$, and $F(t)$ decrease in (c_r, t_0), and increase in (t_0, τ_m), where $t_0 = \frac{-B-\sqrt{B^2-4AC}}{2A}$. If $A > 0$, then $c_{l+1} > \frac{-B-\sqrt{B^2-4AC}}{2A}$, and $F(t)$ decrease in (c_r, t_0), and increase in (t_0, τ_m), where $t_0 = \frac{-B+\sqrt{B^2-4AC}}{2A}$.

Thus, there is always existing a c_k, $k = [1, K-1]$, minimizing $F(t)$ with any $\{\tau_m\}_{m=1}^{K-2}$.