Sui, Jinping; Liu, Zhen; Liu, Li ; Jung, Alex; Li, Xiang

Dynamic Sparse Subspace Clustering for Evolving High-Dimensional Data Streams

Published in:
IEEE Transactions on Cybernetics

DOI:
10.1109/TCYB.2020.3023973

Published: 01/01/2022

Document Version
Publisher's PDF, also known as Version of record

Published under the following license:
CC BY

Please cite the original version:
Sui, J., Liu, Z., Liu, L., Jung, A., & Li, X. (2022). Dynamic Sparse Subspace Clustering for Evolving High-Dimensional Data Streams. *IEEE Transactions on Cybernetics*, *52*(6), 4173-4186.
https://doi.org/10.1109/TCYB.2020.3023973

Dynamic Sparse Subspace Clustering for Evolving High-Dimensional Data Streams

Jinping Sui, Zhen Liu, Li Liu, Alexander Jung, and Xiang Li

***Abstract*—In an era of ubiquitous large-scale evolving data streams, data stream clustering (DSC) has received lots of attention because the scale of the data streams far exceeds the ability of expert human analysts. It has been observed that high-dimensional data are usually distributed in a union of low-dimensional subspaces. In this article, we propose a novel sparse representation-based DSC algorithm, called evolutionary dynamic sparse subspace clustering (EDSSC). It can cope with the time-varying nature of subspaces underlying the evolving data streams, such as subspace emergence, disappearance, and recurrence. The proposed EDSSC consists of two phases: 1) static learning and 2) online clustering. During the first phase, a data structure for storing the statistic summary of data streams, called EDSSC summary, is proposed which can better address the dilemma between the two conflicting goals: 1) saving more points for accuracy of subspace clustering (SC) and 2) discarding more points for the efficiency of DSC. By further proposing an algorithm to estimate the subspace number, the proposed EDSSC does not need to know the number of subspaces. In the second phase, a more suitable index, called the average sparsity concentration index (ASCI), is proposed, which dramatically promotes the clustering accuracy compared to the conventionally utilized SCI index. In addition, the subspace evolution detection model based on the Page-Hinkley test is proposed where the appearing, disappearing, and recurring subspaces can be detected and adapted. Extinct experiments on real-world data streams show that the EDSSC outperforms the state-of-the-art online SC approaches.**

***Index Terms*—Data stream clustering (DSC), high-dimensional data stream, sparse representation, subspace clustering (SC).**

Manuscript received March 16, 2020; revised June 29, 2020; accepted September 9, 2020. Date of publication November 24, 2020; date of current version June 16, 2022. This work was supported in part by the National Natural Science Foundation of China under Grant 62022091, Grant 61921001, Grant 61872379, Grant 71701205, and Grant 91846301; in part by the Hunan Science and Technology Plan Project under Grant 2019GK2131; and in part by the Academy of Finland under Grant 331883. This article was recommended by Associate Editor L. Rutkowski. *(Corresponding author: Zhen Liu.)*

Jinping Sui is with the College of Electronic Science and Technology, National University of Defense Technology, Changsha 410073, China, and also with the Department of Computer Science, Aalto University, 02150 Espoo, Finland (e-mail: suijinping13@nudt.edu.cn).

Zhen Liu is with the College of Electronic Science and Technology, National University of Defense Technology, Changsha 410073, China (e-mail: zhen_liu@nudt.edu.cn).

Li Liu is with the College of System Engineering, National University of Defense Technology, Changsha 410073, China, and also with the Center for Machine Vision and Signal Analysis, University of Oulu, 02150 Oulu, Finland (e-mail: li.liu@oulu.fi).

Alexander Jung is with the Department of Computer Science, Aalto University, 02150 Espoo, Finland (e-mail: alex.jung@aalto.fi).

Xiang Li is with the Department of Electronic Science, National University of Defense Technology, Changsha 410073, China (e-mail: lixiang01@vip.sina.com).

Color versions of one or more figures in this article are available at https://doi.org/10.1109/TCYB.2020.3023973.

Digital Object Identifier 10.1109/TCYB.2020.3023973

I. Introduction

HIGH-DIMENSIONAL data streams are generated at an unprecedented scale in various realms, such as media, communication, finance, meteorology, etc., [1]–[4]. These data streams are often high dimensional, unlabeled, large scale, and evolving, which present huge challenges for data stream clustering (DSC). Most existing DSC algorithms, including the classic ones, such as CluStream [5] and DenStream [6], or even the more recent ones, such as STRAP [7], EDMStream [8], and CEDAS [9], are inadequate to address these challenges. In addition, most existing DSC algorithms are based on nonevolutionary models (e.g., CluStream) or simple evolutionary models (e.g., DenStream, STRAP, and CEDAS) which cannot adapt to the complicated dynamics of clusters' structures in the real world [8], [10]. Therefore, there is a pressing need to study more effective DSC algorithms to process evolving data streams that are high dimensional and large scale.

Motivations: It has been realized that many real-world high-dimensional data, such as motion [11], face [12], and texture [13], actually lie in a union of low-dimensional subspaces [1], [2], [14]–[18]. Subspace clustering (SC) refers to the problem of simultaneously clustering the data into multiple subspaces and finding a low-dimensional subspace to fit each group of points [19]–[22]. Currently, representation-based SC (RBSC) approaches have been dominating the field and represent the state of the art. They are based on the hypothesis that each data point in a union of subspaces can be represented as a linear combination of other points, that is, the so-called *self-expressiveness property*. Popular RBSC approaches include sparse SC (SSC) [1], low-rank representation (LRR) [16], and their variants.

RBSC approaches have been extensively studied in the static data clustering field. However, existing RBSC approaches cannot be directly applied to the DSC problem due to the following reasons.

1) Most RBSC approaches, such as SSC and LRR, are batch processing based [as shown in Fig. 1(a)] and, thus, cannot deal with evolving data streams. As contrasted in Fig. 1, for data streams, it is obviously unwise and even impracticable to collect all the data points and then process them due to the limitations in computing and storage resources. In addition, batch processing manners cannot achieve real-time processing and cannot meet the basic requirements of online learning scenarios.
2) Most existing online RBSC approaches are based on nonevolutionary models resulting in the information that

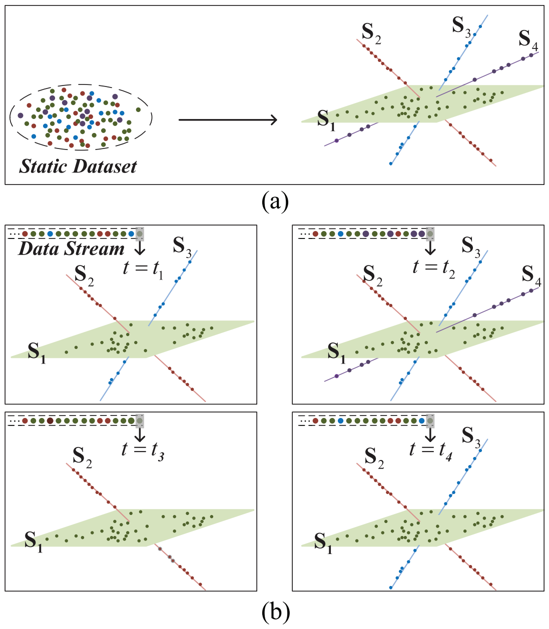

Fig. 1. (a) SC on a static dataset. (b) SC on an evolving data stream: $\mathbf{S}_1$, $\mathbf{S}_2$, and $\mathbf{S}_3$ exist at timestamp t_1; $\mathbf{S}_4$ emerges at timestamp t_2; at timestamp t_3, $\mathbf{S}_3$ and $\mathbf{S}_4$ disappear; and $\mathbf{S}_3$ recurs at timestamp t_4 ($t_4 > t_3 > t_2 > t_1$).

changes over time underlying the data streams [as shown in Fig. 1(b)] that cannot be revealed.

3) The difficulty is balancing two conflicting goals: a) saving more points for pursuing good clustering performance of SC or b) discarding more points for pursuing efficiency of DSC. On the one hand, the self-expressiveness property of data requires saving lots of data points to obtain more robust results, leading to an increase in computational complexity and storage assumption of most existing SC approaches [23]. On the other hand, the DSC algorithms usually discard many points to fulfill the restrictions on computing and storage usage.

Recently, research on online SC algorithms using the self-expressiveness property has grown in popularity with some representative algorithms, such as SSSC [23], SLRR [23], online LRR [24], OLRSC [25], and SLSR [23], proposed. Most of these approaches achieve online SC based on two-phase frameworks, that is, the static phase for global subspace learning and the online phase for subspace classifying (see Fig. 2). However, we think these approaches are too infant to deal with DSC problems from the following aspects: first, they are based on the assumption that subspaces as well as the subspace structure remain stationary and can be all learned in the static phase, without taking the evolving subspace structure into consideration (which is impractical in the DSC field). Second, they lack proper data structures for storing statistic summaries of data streams, which is one of the basic requirements for DSC goals. For example, SSSC, SLSR, and SLRR tend only to save the data points used in the static phase and discard all the data points in the second phase. Online LRR saves all the newly added data to refine the subspace structure learned in the first phase. Clearly, it is not reasonable to simply reserve or discard all the data points to solve DSC tasks.

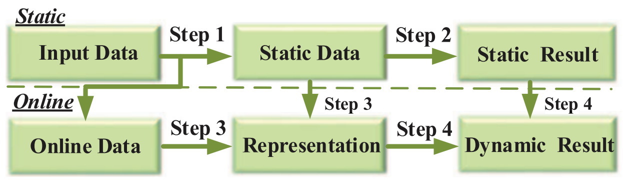

Fig. 2. Unified framework of state-of-the-art online RBSC algorithms.

Contributions: In this article, a novel online SC method, called evolutionary dynamic SSC (EDSSC), is proposed for evolving high-dimensional DSC tasks. Like the recent approaches SSSC [23], SLSR [23], SLRR [23], and online LRR [24], our proposed EDSSC consists of two phases: 1) static learning and 2) online clustering, but fundamental differences exist: EDSSC does not assume that the global subspace structure must be covered by the data points in the static learning phase and must remain unchanged in the online clustering phase. Instead, EDSSC can detect and adapt to the evolving subspace structure based on the proposed subspace evolution detection strategy. In the online clustering phase, EDSSC identifies the representative points of the newly appearing subspaces and then updates the models with these new points, instead of keeping all the points (such as online LRR [24]) or discarding all the points (such as SSSC [23]). The statistic information of the data streams with identified representative points are stored in a data structure that is referred to as *EDSSC summary*. The main contributions of this article are summarized as follows.

1) We are among the first to formulate and study online SC on evolving data streams, that is, the task of performing SC on data streams whose points lie in a union of evolving subspaces. We provide a mathematical formulation of online SC and propose a novel online SC method EDSSC by making use of the self-expressiveness property of data.
2) We propose a new index, called the average sparsity concentration index (ASCI), to help EDSSC to identify if the newly arriving point is a normal point or an outlier. This issue has not acquired sufficient attention in state-of-the-art online RBSC approaches. Compared to the widely adopted index, that is, the sparsity concentration index (SCI) [15], we contend ASCI is better than SCI, especially when the number of data points among subspaces is imbalanced (see Example 2).
3) We propose a subspace structure evolution detection model in EDSSC based on the Page–Hinkley (PH) test [7], [26] and three typical subspace evolutions, such as subspace emergence, disappearance, and recurrence, can be detected. In addition, unlike the state-of-the-art online RBSC algorithms, EDSSC does not require the number of subspaces to be known as prior information. Instead, a method to estimate the number of subspaces is proposed to ensure that EDSSC is more practical.
4) We perform extensive experiments on evolving data streams derived from real-world public datasets, including facial images (AR, ExYaleB, and MPIE) and handwritten digits or characters (USPS, PenDigits, EMNIST-letter, and MNIST), showing that EDSSC achieves

significant improvement in clustering quality [measured by accuracy and normalized mutual information (NMI)] over the existing representative DSC (CEDAS [9] and STRAP [7]) or online RBSC methods (SSSC [23], SLRR [23], SLSR [23], and OLRSC [25]). For example, on the ExYaleB data stream, EDSSC achieves 75.01% accuracy and 86.47% NMI compared with 57.14% accuracy and 74.43% NMI of OLRSC which has the best performance among all baseline algorithms.

This article is a substantial extension of our conference paper [27]. The proposed D-SSC model in [27] gets further improved in this article from the following aspects. First, we propose a new subspace evolution detection model based on the PH test which is different from the one adopted by D-SSC [27]. In comparison, the proposed subspace evolution detection model relies on fewer parameters, making EDSSC more adaptable to evolving data streams. Second, we solve the estimation of the number of problem subspaces that is not fully resolved in [27]. Third, as concluded above, a more proper index, that is, ASCI is proposed in this article to ensure EDSSC is more robust than D-SSC in processing the imbalanced data streams. In addition, we further analyze EDSSC in space and time complexity and parameter sensitivity. Finally, more extensive experiments are performed in this article where more real-world datasets and more state-of-the-art approaches are tested.

Organizations: The remainder of this article is organized as follows. Section II provides a review of the state of the art. Section III formulates the evolving high-dimensional DSC problem. Section IV presents the principles and methodology behind EDSSC and describes the corresponding pseudocode as well as the complexity analysis of the EDSSC. In Section V, we verify and analyze the performance of the proposed algorithm by comparing with SSSC [23], SLRR [23], SLSR [23], OLRSC [25], CEDAS [9], and STRAP [7]. Finally, the study is concluded in Section VI.

II. Related Work

Our approach draws on methods from DSC and SC. In this section, we will give a brief overview of the related work.

A. High-Dimensional Data Stream Clustering

Although a plethora of DSC algorithms has been proposed, the clustering of high-dimensional data streams is still in an immature stage. One of the main reasons is that in the high-dimensional space, all pairs of points tend to be almost equidistant from each other due to an effect, called "curse of dimensionality" [28]. Yet only a few approaches have been proposed to tackle the high-dimensional DSC problem. They can be divided into two categories: 1) *full-space-based ones* and 2) *subspace-based ones*.

Full-Space-Based DSC: Most partition-based methods cannot keep effective performance when dealing with high-dimensional data streams. However, STRAP [7], one of the latest methods of partition-based DSC algorithms, has been successfully employed to process the KDD'99 dataset (34D). Compared to partition-based DSC methods, density-based [6], [9] and synchronization-based ones [29] are more suitable for processing high-dimensional data streams because, in theory, they can find arbitrary-shaped clusters that exist in the entire feature space without requiring the number of clusters. For example, DenStream [6], CEDAS (density based) [9], and SyncTree (synchronization based) [29] are recently proposed and hold state-of-the-art performance among full-space DSC algorithms. They achieve high-performance DSC by combining the static clustering theories (i.e., density-based clustering or synchronization-based clustering [30]–[34]) with dynamic frames (a graph structure or a hierarchical tree). It should be noted that one of the major limitations of the full-space DSC algorithms is that they can only detect clusters existing in the full space. However, clusters are always being hidden since many irrelevant dimensions exist in the high-dimensional space. Therefore, the clustering accuracy of these algorithms cannot be guaranteed for high-dimensional data streams.

Subspace-Based DSC: As discussed above, the cluster structures of high-dimensional data are quite challenging to be directly discovered due to the curse of dimensionality. Luckily, it has been found that some of the high-dimensional data points may be possibly grouped together in certain subspaces. Recently, a few DSC algorithms [28], [35]–[39] are proposed to detect the cluster structures in low-dimensional subspaces. HPStream [28] first introduces the projected clustering method to process high-dimensional data streams. However, it needs a predefined number of clusters that is impractical in real scenarios. HDDStream [35] introduces the density-based projected clustering idea to the DSC field. PreDeConStream [36] extends a static projected-based clustering method, called PreDeCon [40], toward the data stream processing field. SubCluTree [38] uses an adaptive grid strategy to perform a bottom-up detection of the subspace candidates. In addition, it can adapt to the varying speeds of the streaming data. Generally, the subspace-based DSC algorithms cost considerable time for searching the preferred dimension in all possible subspaces of the full space, which cannot meet the low computational complexity requirement of data stream processing [36].

B. Subspace Clustering

The goal of SC is to reveal the data structures by detecting the clusters in the low-dimensional subspaces of the original feature space. Actually, SC is quite a general concept with tremendous algorithms proposed in different directions. For example, the axis-parallel SC methods, for example, [41]–[43], aim to find the clusters only in the axis-parallel subspaces. The arbitrarily oriented SC algorithms, such as 4C [44], CASH [45], and ORSC [46], make it possible to find the clusters in arbitrarily oriented subspaces. Particularly, the ORSC achieves state-of-the-art performance in accuracy and scalability based on the synchronization clustering theory [29]–[33]. Furthermore, the disjoint SC focuses on segmenting the entire space into a union of subspaces that are disjoint or even independent. Among these research directions of SC, the disjoint SC methods are closely related to our work in this article.

Meanwhile, the disjoint SC methods are accepting worldwide attention due to their wide application in machine-learning fields. Therefore, in the following, SC refers to disjoint SC methods unless otherwise stated. Currently, the RBSC methods utilizing the self-expressiveness property of data have become the state-of-the-art methods in the field of SC. The main idea of RBSC methods is to first learn an affinity graph for the data points (subspace recovery) and then apply spectral clustering to the graph (subspace segmentation) [14]. The RBSC algorithms can be divided into two categories: 1) static ones and 2) online ones.

Definition 1 (Self-Expressiveness Property): For a collection of N high-dimensional data points $\mathbf{X} = [\boldsymbol{x}_1 \ldots \boldsymbol{x}_N]_{d\times N} \in \mathbb{R}^{d\times N}$, with each point from a union of independent linear or affine subspaces, each data point $\boldsymbol{x}_i$ can be represented as a linear or affine combination of other points, that is

$$\boldsymbol{x}_i = \mathbf{X}\boldsymbol{c}_i \tag{1}$$

where $\boldsymbol{c}_i = [c_{i1}c_{i2}\ldots c_{iN}]^\top \in \mathbb{R}^{N\times 1}$ and $c_{ii} = 0$.

Static RBSC: If we have a static dataset $\mathbf{X}$, then we have

$$\mathbf{X} = \mathbf{X}\mathbf{C}, \quad \text{diag}(\mathbf{C}) = 0 \tag{2}$$

where $\mathbf{C} = [\boldsymbol{c}_1\boldsymbol{c}_2\ldots\boldsymbol{c}_N] \in \mathbb{R}^{N\times N}$ and diag($\mathbf{C}$) is the vector of the diagonal elements of $\mathbf{C}$. $\mathbf{C}$ is referred to as the *representation matrix*.

For a system of equations such as (2) with infinitely many solutions [1], one can restrict the set of solutions by minimizing an objective function, that is

$$\min f(\mathbf{C}) \ \text{ s.t. } \ \mathbf{X} = \mathbf{X}\mathbf{C}, \quad \text{diag}(\mathbf{C}) = 0 \tag{3}$$

where $f(\cdot)$ denotes the objective function.

Generally, static RBSC approaches vary from their objective functions in (3) and there exists two popular objective functions, that is, norm-based and low-rank-based ones. Typical approaches that utilize the two objective functions are SSC [1], LSR [47] (norm-based objective functions), and LRR (low-rank-based objective functions) [16], respectively. For SSC, $f(\mathbf{C}) = \|\mathbf{C}\|_0$, where $\|\cdot\|_0$ denotes the 0-norm. For LSR, $f(\mathbf{C}) = \|\mathbf{C}\|_F$, where $\|\cdot\|_F$ denotes the Frobenius norm. While for LRR [16], $f(\mathbf{C}) = \text{rank}(\mathbf{C})$. That is, SSC enforces $\mathbf{C}$ to be sparse, LSR tends to group highly correlated data together, while LRR encourages $\mathbf{C}$ to be low rank [47]. Actually, most RBSC algorithms so far are static SC algorithms and actually are variants or extensions of SSC, LSR, LRR, or a combination of them (e.g., LSS [48]). The optimal $\mathbf{C}^*$ obtained by (3) will then be utilized to build an affinity matrix that is denoted as $\mathbf{W}$. The SC results will be obtained after applying spectral clustering algorithms [49]–[52] to $\mathbf{W}$.

Now, we take LRR as an example to briefly introduce the solving of (3). Generally, considering that real-world datasets may contain noise corruption, (3) is usually converted to the following optimization problem:

$$\min_{\mathbf{Z},\mathbf{E}} \ \text{rank}(\mathbf{C}) + \lambda\|\mathbf{E}\|_\ell \ \text{ s.t. } \ \mathbf{X} = \mathbf{X}\mathbf{C} + \mathbf{E} \tag{4}$$

where $\lambda > 0$ is a balance parameter, $\mathbf{E}$ is an additional error matrix that is assumed to be sparse, and $\|\cdot\|_\ell$ denotes a certain regularization strategy. However, (4) is hard to optimize due to the rank function. In practice, one can solve (4) by using the $\ell_{2,1}$-norm to regularize the second term and the nuclear norm as a surrogate to replace the rank function

$$\min_{\mathbf{C},\mathbf{E}} \ \|\mathbf{C}\|_* + \lambda\|\mathbf{E}\|_{2,1} \ \text{ s.t. } \ \mathbf{X} = \mathbf{X}\mathbf{C} + \mathbf{E} \tag{5}$$

where $\|\cdot\|_*$ means the nuclear norm, which is the best convex envelope of the rank [24]. The optimization problem (5) is convex and can be solved by several methods. In this article, we utilize a state-of-the-art approach, called the inexact augmented Lagrange multiplier (IALM), for its accuracy and efficiency [53], [54]. Then, the skinny SVD of $\mathbf{C}$ is obtained

$$\mathbf{C}^* = \mathbf{U}^*\mathbf{S}^*(\mathbf{V}^*)^\top. \tag{6}$$

An affinity matrix $\mathbf{W}$ can be built as follows:

$$[\mathbf{W}]_{ij} = \left(\left[\widetilde{\mathbf{U}}\widetilde{\mathbf{U}}^\top\right]_{ij}\right)^2 \tag{7}$$

where $\widetilde{\mathbf{U}} = \mathbf{U}^*(\mathbf{S}^*)^{(1/2)}$. Finally, one can get the clustering (segmentations) of the data by applying spectral clustering algorithms (e.g., normalized cuts (NCuts) method [16]) to $\mathbf{W}$.

Online RBSC: Popular online RBSC includes SSSC, SLSR, SLRR [23], and online LRR [24]. However, these online SC algorithms, in general, fail to process evolving data streams as analyzed in Section I.

Here, we comprehensively summarize the architectures of state-of-the-art online RBSC algorithms in a unified framework depicted in Fig. 2. There are two distinct phases, that is, static phase and online phase, as well as four main steps included (i.e., steps 1–4 in Fig. 2): first, the input data are divided into two parts, that is, static data and online data. The static data and online data are used for static learning and online clustering, respectively. Second, a certain static SC approach, such as SSC, LSR, LRR, is performed over the static data to obtain the static result, which could be regarded as a preparation work for the following online clustering. Then, the representation matrix or representation coefficients are obtained based on the static data, which relies on the processing manner of the algorithm. Finally, the online data points will be assigned to the corresponding subspaces that are found in the static results.

III. Problem Formulation

A high-dimensional data stream is a sequence of timestamped high-dimensional data points that lie in a union of low-dimensional subspaces. Assume that at each timestamp t, we receive one data point $\boldsymbol{x}^t$, $\boldsymbol{x}^t \in \mathbb{R}^{d\times 1}$. We arrange t points which we have received in total in a matrix $\mathbf{X}^t = [\boldsymbol{x}^1 \ldots \boldsymbol{x}^t]_{d\times t}$.

Note that data streams have two features that are significantly different from static datasets: the first distinctive feature is the extremely limited number of accesses to the received points (most of the points should be immediately discarded after being accessed once or a few times for decreasing computing and storage consumption). The second main difference to static data is the potentially evolving property, that is, the *evolving data streams*.

The goal of this article is to perform DSC on the evolving high-dimensional data streams, that is, providing a time-varying SC result $\mathbb{S}^t$ at each timestamp t which reflects the partition of received points $\mathbf{X}^t$ such that the points belonging to the same subspace can be assigned to the same cluster.

The first feature of data streams aforementioned results in which we could not expect to obtain $\mathbb{S}^t$ by repeatedly performing static clustering on $\mathbf{X}^t$. Instead, the incremental learning manner should be applied to process data streams, that is

$$\mathbb{S}^t = g\left(\mathbb{S}^{t-1}, \boldsymbol{x}^t\right) \tag{8}$$

where $g(\mathbb{S}^{t-1}, \boldsymbol{x}^t)$ denotes a certain function that updates $\mathbb{S}^{t-1}$ with $\boldsymbol{x}^t$ at timestamp t.

The evolving feature of the data streams requires that no assumption should be made that the subspace structure remains unchanged over time. Generally, the evolution can be caused by three types of subspace evolution [55]–[58], that is, subspace emergence, disappearance, and recurrence, which are formulated as follows.

Subspace Emergence: It refers to the occurrence of a new subspace at timestamp t. In particular, a subspace $\mathcal{S}$ emerges at timestamp t if $\mathcal{S} \notin \mathbb{S}^1 \cup \mathbb{S}^2 \cup \cdots \cup \mathbb{S}^{t-1}$ and $\mathcal{S} \in \mathbb{S}^t$.

Subspace Disappearance: It is defined as an existing subspace that is not visited by the recently arrived data points. Formally, a subspace $\mathcal{S}$ disappears if $\mathcal{S} \in \mathbb{S}^{t_1} \cap \mathbb{S}^{t_1+1} \cap \cdots \cap \mathbb{S}^{t-1}$ and $\mathcal{S} \notin \mathbb{S}^t$, where $1 \le t_1 < t$.

Subspace Recurrence: It means the situation where a previously disappeared subspace recurs at timestamp t. Formally, a subspace $\mathcal{S}$ recurs at timestamp t if $\mathcal{S} \in \mathbb{S}^{t_2} \cap \mathbb{S}^{t_2+1} \cap \cdots \cap \mathbb{S}^{t_3-1}$, $\mathcal{S} \notin \mathbb{S}^{t_3} \cup \mathbb{S}^{t_3+1} \cup \cdots \cup \mathbb{S}^{t-1}$, and $\mathcal{S} \in \mathbb{S}^t$, where $1 \le t_2 < t_3 < t$.

IV. Proposed EDSSC Approach

Even though the self-expressiveness property has been successfully used to perform SC on static datasets, it is quite challenging to achieve dynamic SC on data streams based on the property because it is hard to balance two conflicting goals: 1) saving points for good SC performance and 2) discarding points for low computational complexity. In this section, an efficient algorithm, EDSSC, is proposed for balancing the two competing goals and addressing (8), that is, performing DSC on evolving data streams.

EDSSC consists of the following two phases: 1) static learning and 2) online clustering, which are presented in detail in Sections IV-A and IV-B, respectively. The subspace evolution problem is solved in Section IV-C where the subspace evolution detection strategy is given. Finally, the pseudocode and the complexity analysis of EDSSC are provided in Section IV-D.

A. Static Learning and EDSSC Summary Structure

As indicated in (8), the previous clustering result is needed to identify the clustering result at the current timestamp. Therefore, the ultimate goal of static learning is to learn an initial subspace structure based on a certain amount of points initially received to initialize the EDSSC model.

In the static learning phase, we mainly address two challenges: 1) proposing a method to know the number of subspaces lying in the data used in the static learning and 2) designing a data structure, called the EDSSC summary, for storing statistic summaries of the data stream. In a broad sense, the EDSSC summary here is the clustering result. Hence, we continue using $\mathbb{S}^t$ to denote the EDSSC summary.

Specifically, we denote the EDSSC summary as $\mathbb{S}^t = \{\mathcal{S}^t\}_{l=1}^{k^t}$. Each $\mathcal{S}_l^t = \{n_l^t, \mathbf{R}_l^t, \mathcal{T}_l^t, \Omega_l^t\}$ summarizes the information of the lth subspace, where:

1) n_l^t is the total number of data points assigned to subspace l up to timestamp t;
2) $\mathbf{R}_l^t$ is referred to as a *reserved data matrix* of subspace l, which saves selected points from subspace l up to t;
3) $\mathcal{T}_l^t$ records the timestamps of all the points that are assigned to subspace l up to t;
4) Ω_l^t records the ASCI (which will be described in Section IV-B) of all the points belonging to subspace l up to t.

EDSSC needs to be initialized at the static learning phase where the first batch of $T_0 (T_0 > d)$ data points $\mathbf{X}^{T_0}$ is processed, namely, EDSSC needs to cluster $\mathbf{X}^{T_0}$ to obtain the representation matrix $\mathbf{C}^{*T_0}$ first, which can be solved via the optimization problem in (5).

The affinity matrix at timestamp T_0, denoted as $\mathbf{W}^{T_0}$, is then built by (7). The clustering result will be obtained after applying spectral clustering algorithms on $\mathbf{W}^{T_0}$. Note that the number of subspaces, that is, k^{T_0}, is required to be input for most spectral clustering algorithms.

Generally, estimating the number of subspaces underlying the data matrix is still one of the most challenging problems in clustering [59]–[61]. This is partly because it is quite subjective to definite what a subspace is. Subsequently, most state-of-the-art online RBSC methods (including SSSC, SLRR, SLSR, OLRSC, etc.) assume that k^{T_0} is defined by users as prior, which limits their usage in real-world applications.

Currently, singular-based Laplacian matrix decomposition is one of the main techniques to solve the problem. Liu *et al.* [16] estimated the number of clusters by counting the small singular values of a normalized Laplacian matrix that should be smaller than a given cut-off threshold. However, the method is extremely sensitive to the selection of the threshold. This article proposes a new method that uses singular-based Laplacian matrix decomposition to automatically estimate the number of subspaces without requiring any input threshold. Assume we have obtained the similarity matrix $\mathbf{W}$ of the data matrix $\mathbf{X}_{d\times N}$ [via (6) and (7)]. Then, we can obtain the normalized Laplacian matrix $\mathbf{L}$ as follows:

$$\mathbf{L} = \mathbf{I} - \mathbf{D}^{-1/2}\mathbf{W}\mathbf{D}^{-1/2} \tag{9}$$

where $\mathbf{D} = \text{diag}(\sum_{j=1}^{N} [\mathbf{W}]_{1j}, \ldots, \sum_{j=1}^{N} [\mathbf{W}]_{Nj})$. Then, the eigenvalues of $\mathbf{L}$ are obtained, which are denoted as $\{\sigma_i\}_{i=1}^{N}$. Here, assume that the eigenvalues have been sorted in an increasing order, that is, $0 = \sigma_1 \le \sigma_2 \le \cdots \le \sigma_N$. Note that 0 is one of the eigenvalues.

Then, the estimation of k^{T_0} could be obtained by

$$\widehat{k} = \max_i \{|\phi_i|\}_{i=2}^{N-1} - k_0 \tag{10}$$

where

$$\phi_i = \log_a\left(\sigma_i^2/\sigma_{i+1}\sigma_{i-1}\right), (a \text{ is a constant and } a > 1) \quad (11)$$

and

$$k_0 = \begin{cases} 0, & \text{when } \sigma_{i-1}\sigma_{i+1} \geq \sigma_i^2 \\ 1, & \text{otherwise.} \end{cases} \quad (12)$$

After estimating the number of subspaces underlying $\mathbf{X}^{T_0}$, we can further pursue the eigenvectors corresponding to the smallest k^{T_0} eigenvalues. The clustering result of $\mathbf{X}^{T_0}$ can be then obtained by performing a basic clustering (e.g., k-means clustering) on the eigenvectors matrix. Note that $\mathcal{T}_l^{T_0}$ and $\Omega_l^{T_0}$ are empty for $l \in [1, k^{T_0}]$. Limited by the storage space, it is unwise to have all the points of each subspace in their respective $\mathbf{R}_l^t$. Therefore, it is reasonable to only select and save a small part of points as representatives of each subspace. Actually, representative selecting is another challenging topic that has not been well solved. There are a few algorithms designed to accomplish the task, such as DS3 [62] and ESC [14]. However, these methods are inefficient for large-scale data processing. Here, for each subspace, we adopt a uniform random sampling approach of which the time complexity is only $O(1)$ and the performance are comparable with the other complex sampling techniques [23]. We call the points processed in the static learning phase and online clustering as *supporting points* and *streaming points*, respectively. Since the number of supporting points in each subspace may vary significantly, we design a logarithmic function to control the number of points reserved in each subspace as follows. Assume that we have n_{sup} supporting points and n_{res} points will be reserved in $\mathbf{R}$, then n_{res} can be decided by the following:

$$n_{\text{res}} = \begin{cases} n_{\text{sup}}, & \text{when } n_{\text{sup}} < N_0 \\ \min\left\{n_{\text{sup}}, \log_{c_0}\left(a_0\left(n_{\text{sup}}\right)^{1/n_0} + b_0\right)\right\}, & \text{otherwise} \end{cases} \quad (13)$$

where N_0, a_0, b_0, c_0, and n_0 are constants and can be predefined by the users for different tasks. Our recommended setting for n_0 and N_0 is $n_0 = 2$ and $N_0 = d$. The basic principle for (13) is when $n_{\text{sup}} < N_0$, all the n_{sup} will be reserved. While when $n_{\text{sup}} \geq N_0$, a subset of the supporting points will be reserved and the larger the n_{sup}, the lesser the ratio $n_{\text{res}}/n_{\text{sup}}$. Generally, we expect that when $n_{\text{sup}} = d$, $n_{\text{res}} = d$ and when $n_{\text{sup}} = 4d$, $n_{\text{res}} = 2d$ (d is the dimension of the data point). This can be easily achieved by setting a_0 and b_0 according to the following equations:

$$\begin{cases} a_0 = \frac{c_0^{2d} - c_0^d}{d^{1/2}} \\ b_0 = 2c_0^d - c_0^{2d}. \end{cases} \quad (14)$$

It should be noted that we use (13) to balance the contradiction between the saving points for accuracy and discarding points for efficiency. Note that we set $n_0 = 2$, $N_0 = D$, and $c_0 = 1.005$ in this article. In specific, the initialization of the EDSSC summary is summarized in Algorithm 1.

Algorithm 1 Initialization of EDSSC Summary

```
Input: Data stream $x^1, \ldots, x^t, \ldots$; Initial batch size $T_0$;
Output: The initialization of EDSSC summary.
1: Obtaining $\mathbf{C}^{*T_0}$ by solving Eq. (5) and Eq. (6).
2: Building the affinity graph by solving Eq. (7).
3: Estimating the number of subspaces $\widetilde{k}^{T_0}$ by solving Eq. (10).
4: Obtaining the clustering results by performing spectral clustering
   algorithms on $\mathbf{W}^{T_0}$.
5: Applying the uniform random sampling approach to the points
   of each subspace.
6: Getting the EDSSC summary initialized.
```

B. Online Clustering Based on Sparse Representation

After the static learning phase, the EDSSC is ready to cluster the following points in an online manner, that is, the online clustering phase. For each arriving data point in the online clustering phase, EDSSC must decide if it is a normal point first (if the point belongs to one of the subspaces we have found in the EDSSC summary). The abnormal points are called *outliers* and should be rejected. Generally, the outlier detection is one of the most important research topics in the DSC research community [63], [64]. Here, with the help of the self-expressiveness property, EDSSC is capable of automatically accepting the normal points and rejecting the outliers.

Depending on whether the subspaces can effectively represent the current pattern of data streams, EDSSC divides the found subspaces into two states, that is, *active* and *inactive*, at each timestamp. The inactive state means the corresponding subspaces have expired. Both states can be converted to each other over time. Section IV-C will demonstrate how they convert to each other under the subspace evolution detection strategy of EDSSC. Note that only the active subspaces are stored in the EDSSC summary. For the inactive subspaces, EDSSC does not directly delete them but stores them to another reservoir $\mathbb{D}^t = \{\mathcal{D}_m^t\}_{m=1}^{h^t}$ which we refer to as the *remove reservoir*. Here, we assume that at timestamp t, there are h^t inactive subspaces in $\mathbb{D}^t$. The inactive subspace can be denoted as $\mathcal{D}_m^t = \{\widetilde{n}_m^t, \widetilde{\mathbf{R}}_m^t, \widetilde{\mathcal{T}}_m^t, \widetilde{\Omega}_m^t\}$. Meanwhile, we define $\mathbf{Z}^t = [\mathbf{R}_1^t \cdots \mathbf{R}_{k^t}^t \widetilde{\mathbf{R}}_1^t \cdots \widetilde{\mathbf{R}}_{h^t}^t]$ here. $\mathbf{Z}^t$ is made up by all reserved data matrix of active and inactive subspaces at timestamp t.

For a new arriving data point $\boldsymbol{x}^t (t > T_0)$, we first get its representation coefficients under $\mathbf{Z}^t$, that is

$$\min \ \left\|\boldsymbol{c}^t\right\|_0 \ \text{ s.t. } \boldsymbol{x}^t = \mathbf{Z}^t \boldsymbol{c}^t. \quad (15)$$

We can also use $\|\cdot\|_1$ norm to relax (15) to obtain the solution $\boldsymbol{c}^{*t}$, referred to as *representation coefficients*. Considering $\boldsymbol{x}^t$ could be either an outlier or a normal point, it is necessary to identify $\boldsymbol{x}^t$ before we assign $\boldsymbol{x}^t$ into a subspace. According to the self-expressiveness property, we know that $\boldsymbol{c}^{*t}$ has the block sparsity property if $\boldsymbol{x}^t$ is a normal point. That is, nonzero elements of $\boldsymbol{c}^{*t}$ are concentrated on a certain part (this part corresponds to the reserved matrix of its own subspace). While for outliers, their solutions $\boldsymbol{c}^{*t}$ do not have the block sparsity property. Without loss of generality, we further demonstrate the difference between the results of outlier and normal point in the following example.

Example 1 (Normal Points Versus Outliers): The Cropped Extended Yale B [65][1] database is utilized to demonstrate

[1]The Cropped Extended Yale B database can be accessed via http://vision.ucsd.edu/~leekc/ExtYaleDatabase/ExtYaleB.html.

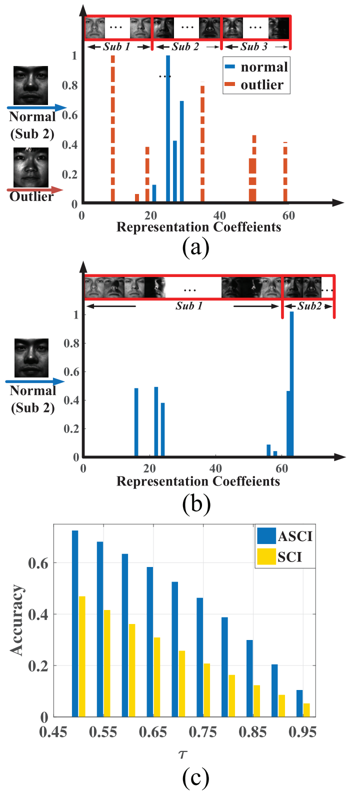

Fig. 3. Illustration of Example 1 and Example 2. (a) *Example 1*: Matrix $\mathbf{Z}$ consists of 60 images of three subjects (20 images/subject) from Cropped Extend Yale B. Then, we obtain the representation coefficients of a normal image and an outlier image. The nonzero coefficients of the normal image show a concentrate feature while those of the outlier image are dispersed. (b) *Example 2*: When $\mathbf{Z}$ is imbalanced, the nonzero coefficients of a normal point from the under-represented subject (Sub2) will not just correspond to the images from the same subject in $\mathbf{Z}$. The SCI index in this case is not effective anymore with almost equivalent sums of nonzero coefficients between Sub1 and Sub2. However, ASCI is still valid. (c) Comparison of the accuracy of the experiments performed in Example 2 being identified as normal points under ASCI and SCI indexes with different τ settings.

the difference of the coefficients of a normal and an outlier. The Cropped Extended Yale B database contains 2432 images from 38 subjects (64 for each subject) and the size of each image is 192×168 cropped from the original Extended Yale B database [66]. We further downsize each image to 48×42 for computational efficiency. Here, the 38 subjects are referred to as Sub1, Sub2, $\ldots$, Sub38, respectively. The first four subjects (Sub1–Sub4) are utilized in this example. We randomly select 20 images from Sub1, Sub2, and Sub3, respectively, (60 images in total) and treat each image as a column to form matrix $\mathbf{Z}_{2016\times 60}$. Obviously, for $\mathbf{Z}$, the new arriving images to be assigned will be normal points if they are from Sub1–Sub3. Otherwise, they would be outliers. We select one image from the remaining images of Sub2, which is denoted as $\boldsymbol{x}_{\text{normal}}$. Then, an image, denoted as $\boldsymbol{x}_{\text{outlier}}$, is randomly selected from Sub4. The representation coefficients for $\boldsymbol{x}_{\text{normal}}$ and $\boldsymbol{x}_{\text{outlier}}$ are obtained by (15) and are illustrated in Fig. 3(a). For the coefficients of the normal point $\boldsymbol{x}_{\text{normal}}$, the nonzero coefficients only correspond to the images from the same subspace (Sub2) of $\boldsymbol{x}_{\text{normal}}$. However, the nonzero coefficients for the outlier are very different.

In order to quantitatively measure the sparse concentration of $\boldsymbol{c}^{*t}$, we propose the ASCI to quantify how concentrated the coefficients are on a single subspace.

Definition 2 [Average Sparsity Concentration Index (ASCI)]: The ASCI of a coefficient vector $\boldsymbol{c} \in \mathbb{R}^n$ is defined as

$$\text{ASCI}(\boldsymbol{c}) \triangleq \frac{k \cdot \max_j\left(\dfrac{\left\|\delta_j(\boldsymbol{c})\right\|_1/\zeta_j}{\sum_{i=1}^{k}\|\delta_i(\boldsymbol{c})\|_1/\zeta_i}\right) - 1}{k-1} \in [0, 1] \quad (16)$$

where k is the number of internal subparts which $\boldsymbol{c}$ can be divided into. The function $\delta_j(\cdot) : \mathbb{R}^n \to \mathbb{R}^n$ selects the coefficients associated with the jth ($j \in [1, k]$) subpart in $\boldsymbol{c}$ and keeps its elements which correspond to other subsparts in $\boldsymbol{c}$ as 0. ζ_j is the length of the jth subspart in $\boldsymbol{c}$. ASCI($\boldsymbol{c}$) $\in [0, 1]$ and a higher ASCI($\boldsymbol{c}$) means the coefficients of $\boldsymbol{c}$ is more likely to concentrate on a single subpart.

Proposition 1: The ASCI index is equivalent to the SCI index[2] [15] when the length of all the subsparts of the vector to be measured is equal.

Proof of Proposition 1: Suppose we have a $q \times 1$ coefficient vector $\boldsymbol{c}$ containing k sequentially concatenated but disjoint subparts $\{\boldsymbol{c}_i\}_{i=1}^k$ (i.e., for $\forall i, j \in [1, k], \boldsymbol{c}_i \cap \boldsymbol{c}_j = \emptyset$ and $\boldsymbol{c}_1 \cup \boldsymbol{c}_2 \cdots \cup \boldsymbol{c}_k = \boldsymbol{c}$) and the length of each subpart $\boldsymbol{c}_i$ is denoted as ζ_i. When all the subparts are of equal length, we have $\zeta_i = q/k$ for $\forall i \in \{1, \ldots, k\}$. Then, the ASCI (16) can be further derived as

$$\begin{aligned}\text{ASCI}(\boldsymbol{c}) &= \frac{k \cdot \max_j\left(\dfrac{(q/k)\cdot\left\|\delta_j(\boldsymbol{c})\right\|_1}{(q/k)\cdot\sum_{i=1}^{k}\|\delta_i(\boldsymbol{c})\|_1}\right) - 1}{k-1} \\ &= \frac{k \cdot \max_j\left(\left\|\delta_j(\boldsymbol{c})\right\|_1/\|\boldsymbol{c}\|_1\right) - 1}{k-1}.\end{aligned}$$

Therefore, the SCI index can be viewed as a special form of the ASCI index. For more general cases when the subparts of the vector are of different length, the ASCI index takes the length of each subpart into consideration. We contend this could ensure that ASCI is more suitable to evaluate the sparsity concentration of a coefficient vector than the SCI index, especially when the length of subparts is highly imbalanced, as illustrated in Example 2.

Example 2 (ASCI Versus SCI): For ease of demonstration, we assume that $\mathbf{Z}$ consists of two subparts, $\mathbf{Z}_1$ and $\mathbf{Z}_2$ and the length of the two subparts are imbalanced. Without loss of generality, we assume ζ_1 is obviously larger than ζ_2. Accordingly, $\mathbf{Z}_1$ and $\mathbf{Z}_2$ are referred to as *over-represented subject* and *under-represented subject*, respectively. We randomly select 60 and six images from Sub1 and Sub2, respectively, to form $\mathbf{Z}_{2016\times 66}$. Then, an image, denoted as $\boldsymbol{x}_{\text{test}}$, is randomly selected from the rest images from Sub2. After being taken into (15), the representation coefficients $\boldsymbol{c}^*$ are obtained and demonstrated in Fig. 3(b). Interestingly, as can be found in Fig. 3(b), the coefficients do not ideally concentrate on the second subpart (which corresponds to the images from Sub2) as expected. The main reason is when there is an imbalance among internal subparts of $\mathbf{Z}$, the representation

[2]The definition of SCI is $\text{SCI}(\boldsymbol{c}) \triangleq (k \cdot \max_j \|\delta_j(\boldsymbol{c})\|_1/\|\boldsymbol{c}\|_1 - 1)/(k-1)$.

coefficients $\boldsymbol{c}^*$ is more likely to have nonzero entries corresponding to the columns in $\mathbf{Z}$ from the over-represented subject. Actually, a similar phenomenon is found recently in [14] which demonstrates that the imbalanced case is pretty common in practice. In this experiment, $\mathrm{ASCI}(\boldsymbol{c}^*) = 0.9021$ while $\mathrm{SCI}(\boldsymbol{c}^*) = 0.1002$. Obviously, under the SCI index, the test $\boldsymbol{x}_{\text{test}}$ (which should be identified as a normal point) is easier to be wrongly identified as an outlier.

We perform additional experiments by selecting other subjects as over-represented and under-represented subjects. Specifically, we sequentially select each of the 38 subjects as the over-represented subject and select one from the remaining subjects as the under-represented subject. The other settings are equal to the experiment above. We perform 40 774 experiments ($38 \times 37 \times 58$) in total and we obtain the corresponding $\mathrm{ASCI}(\boldsymbol{c}^*)$ and $\mathrm{SCI}(\boldsymbol{c}^*)$. Note that under a given τ, these $\mathrm{ASCI}(\boldsymbol{c}^*)$ and $\mathrm{SCI}(\boldsymbol{c}^*)$ will be compared to τ and then identified to be normal points or outliers. Here, with different τ, we further depict the accuracy of these 40 774 test samples being identified as normal points under ASCI and SCI indexes [see Fig. 3(c)]. As shown in Fig. 3(c), the ASCI index dramatically outperforms than SCI under different τ with a significant improvement on the accuracy. Even though it has been observed that such imbalanced case are quite more common in real life [14], fortunately, the imbalanced case can be better handled by ASCI because the length of each subpart in $\boldsymbol{c}^*$ is accounted in (16). Then, for the under-represented subject, the influence of the imbalanced case will be dramatically mitigated or even eliminated.

It should be noted that we denote $\mathrm{ASCI}(\boldsymbol{c})$ as $\omega(\boldsymbol{c})$ for brevity. For $\boldsymbol{c}^{*t}$ obtained by (15), we can then calculate the corresponding $\omega(\boldsymbol{c}^{*t})$. Here, we choose a threshold $\tau \in [0, 1]$ and accept $\boldsymbol{x}^t$ as a normal point if

$$\omega\left(\boldsymbol{c}^{*t}\right) \geq \tau \tag{17}$$

and otherwise reject as an outlier. The outlier will be saved in an *outlier reservoir*, denoted as $\mathbb{O}^t$. Precisely, $\mathbb{O}^t = \{\mathcal{O}^i\}_{i=1}^{n_o^t}$ and $\mathcal{O}^t = \{\boldsymbol{x}^i, \omega^i, t_i\}$. n_o^t is the number of outliers at timestamp t. While for a normal point, it can be either from an active subspace or from an inactive subspace. There are k^t active subspaces and h^t inactive subspaces whose reserved data matrix corresponds to the $k^t + h^t$ subparts in $\mathbf{Z}^t$. We then assign the normal point $\boldsymbol{x}^t$ based on the following optimization function:

$$\min_j r_j\left(\boldsymbol{x}^t\right) \triangleq \left\|\boldsymbol{x}^t - \mathbf{Z}^t\delta_j\left(\boldsymbol{c}^{*t}\right)\right\|_2 \tag{18}$$

where $r_j(\cdot)$ is the residual if $\boldsymbol{x}^t$ is assigned into the jth subspace. The optimal j^* will be obtained by (18) which corresponds to the j^*th subpart in $\mathbf{Z}^t$. If $j^* \leq k^t$, the corresponding active subspace $\mathcal{S}_{j^*}$ would be updated, otherwise the inactive subspace $\mathcal{D}_{j^*-k^t}$ would be updated.

C. Online Subspace Evolution Detection

In real-world application, most data streams have nonstationary properties, commonly known as *concept drift* [55]. The concept drift leads to a time-varying subspace structure, that is, subspace evolution. Depending on the drift speed, such subspace evolution can be further divided into abrupt and gradual subspace evolution. It should be pointed out that here we mainly focus on the abrupt case because it is relatively more ubiquitous and easy to be detected in unsupervised tasks [67]. Therefore, in this article, the subspace evolution mainly refers to the abrupt case unless otherwise stated. Concretely, three specific types of subspace evolution are considered, that is, subspace emergence, disappearance, and recurrence. An online subspace evolution detection strategy is designed to ensure that EDSSC can deal with these evolving cases. Specifically, the subspace emergence and recurrence detection are based on the PH test [26], [68].

The PH test is a scalar change point detection (CPD) test method and has been successfully proved and verified in [7]. Compared with other concept drift detection methods, such as PCA-based approaches [34], the PH test allows us to not directly detect the changes of data stream $\mathbf{X}^t$, to avoid increasing the computational cost and complicating the algorithm. Here, inspired by [7], we detect the emergence and recurrence through observing the variation tendency of the outlier rate and the recurring points rate, respectively. The subspace disappearance detection is based on a fading function.

1) Subspace Emergence and Recurrence Detection Based on PH Test: Considering high-dimensional DSC is an unsupervised task, we employ the PH test, which is one of the classical statistical hypothesis testing methods to detect the concept drift. We first give a brief demonstration of how the PH test works [7], [26], [68].

Assume the observed random variable is p. At each timestamp t, we get the empirical average of p

$$\bar{p}_t = \frac{1}{t}\sum_{i=1}^{t} p_i \tag{19}$$

and the sum of differences between p and $\bar{p}_t$

$$\gamma_t = \sum_{i=1}^{t}(p_i - \bar{p}_i + \delta) \tag{20}$$

during the time interval $[1, t]$. δ is a positive real value which controls the test model. Meanwhile, Γ_t records the historical maximum value of γ up to current timestamp t, that is, $\Gamma_t = \max\{\gamma_1, \ldots, \gamma_t\}$.

At each timestamp, the gap PH_t between Γ_t and γ_t is obtained and the PH test is triggered if the gap is above a threshold η, that is

$$\text{PH test triggered iff } \mathrm{PH}_t = \Gamma_t - \gamma_t > \eta. \tag{21}$$

The PH test has been theoretically verified to detect a negative drop in the mean of the Gaussian distribution [7]. Obviously, η controls the flexibility of the detection and an appropriate η setting should depend on the stream itself. In [7], a proper way about setting η is proposed and employed in our work. η can be set as

$$\eta = \begin{cases} 0, & \text{if } \mathrm{PH}_t = 0 \\ f * \bar{p}_{t_0}, & \text{otherwise} \end{cases} \tag{22}$$

where f is a constant that controls the sensitivity of the detection and t_0 is the first timestamp when $\mathrm{PH}_t \neq 0$.

Algorithm 2 EDSSC Algorithm

```
Input: Data stream $x^1, \ldots, x^t, \ldots$; Initial batch size $T_0$;
    Thresholds $\beta, , \tau, f$; $\mathbb{S} \leftarrow \emptyset$, $\mathbb{D} \leftarrow \emptyset$, $\mathbb{O} \leftarrow \emptyset$
Output: The online cluster membership of the data stream.
1: Apply Algorithm 1 to get the initial EDSSC summary $\mathbb{S}^{T_0}$.
2: Compute $\omega(c^{*t})$ for each arriving point $x^t$ ($t > T_0$) via
   Eq. (15)-Eq. (16).
3: Decide to accept $x^t$ as a normal point or reject $x^t$ as an outlier
   via Eq. (17). For the normal point, compute the residual and
   $j^*$ by Eq. (18) and update $\mathbb{S}$ or $\mathbb{D}$ accordingly. For the outlier,
   update $\mathbb{O}$.
4: Perform subspace evolution detection: compute $\dot{p}_m^t$, $p^t$ and detect
   the subspace recurrence, emergence via Eq. (21). If Eq. (21) is
   triggered, update $\mathbb{S}$, $\mathbb{D}$, $\mathbb{O}$ accordingly. Compute $\ddot{p}_l^t$ and detect
   the subspace disappearance via $\ddot{p}_l^t \geq 0.5$. If triggered, update $\mathbb{S}$
   and $\mathbb{D}$.
```

Now, we consider the subspace emergence. Subspace emergence will result in an obvious feature in the data stream, that is, a large number of outliers found in the data stream within a short time interval. Hence, we define a variable p which can quantitatively reflect whether outliers appear in large quantities in a short time. That is

$$p^t = \sqrt{\frac{1}{n_o^t}\sum_{i=1}^{n_o^t}(1+\log(t_i - t_{i-1}))\left(\omega_{t_i} - \frac{1}{n_o^t}\sum_{k=1}^{n_o^t}\omega_{t_k}\right)^2}. \quad (23)$$

p^t is monitored under the PH test at each timestamp. When outliers appear in large quantities in a short time, p^t will show a downward trend which can be detected by the PH test. It should be pointed out that when the subspace emergence is detected, the points in $\mathbb{O}$ will be taken into (5) to find the clustering result. The points will also be randomly sampled then. The new added subspaces summary $\mathbb{S}_*$ will be updated into the EDSSC summary finally.

Subspace recurrence is the case that once disappeared subspaces are active again, that is, their points are observed again in the data streams during the very recent timestamps. Actually, in this sense, subspace recurrence has a common evolving feature with subspace emergence. That is, subspace recurrence is equivalent to the emergence of the second time or even higher time. Hence, for each inactive subspace $\mathcal{D}_m^t$ in the remove reservoir $\mathbb{D}^t$, we define a variable $\dot{p}_m^t$, that is

$$\dot{p}_m^t = \sqrt{\frac{1}{\widetilde{n}_m^t}\sum_{i=1}^{\widetilde{n}_m^t}(1+\log(t_i - t_{i-1}))}. \quad (24)$$

2) Subspace Disappearance Detection Based on Fading Function: The data points of data streams are potentially infinite which implies that many of the previously active subspaces will gradually expire. These subspaces should be detected in time to ensure that the clustering results are more suitable to reflect the current patterns of data streams. EDSSC defines a variable $\ddot{p}_l^t$ for each active subspace, that is

$$\ddot{p}_l^t = 1 - \frac{1}{1+e^{-(t-\max\{\mathcal{T}_l^t\}-\beta)}} \quad (25)$$

where $\max\{\mathcal{T}_l^t\}$ is the last timestamp when a data point was assigned to subspace l and β is a preset parameter which controls the sensitivity of the EDSSC model to subspace disappearance detection. As can be inferred from (25), $\ddot{p}_l^t$ is based on a sigmoid function and $\ddot{p}_l^t$ will drop sharply when $t > \max\{\mathcal{T}_l\}+\beta$. Hence, EDSSC monitors $\ddot{p}_l^t$ for all active subspace at each timestamp and accept the subspace as an active subspace if $\ddot{p}_l^t \geq 0.5$. Inactive subspaces will be removed from the EDSSC summary to avoid the summary being redundant.

D. Algorithm Flow and Complexity Analysis of EDSSC

We summarize the complete procedure of EDSSC in Algorithm 2. Precisely, the EDSSC involves four main steps.

First, the first bunch of data is collected and processed to initialize the EDSSC summary (Section IV-A).

Second, for each of the arriving points, its ASCI value will be calculated by (15) and (16). Then, the point will be accepted as a normal point or rejected as an outlier according to (17). For normal points, they will be updated in the EDSSC summary or the remove reservoir by (18). While the outliers will be put into the outlier reservoir $\mathbb{O}^t$ (Section IV-B).

Third, the PH test will be utilized to check if subspaces recur in remove reservoir $\mathbb{D}^t$ and new subspaces emerge in outlier reservoir $\mathbb{O}^t$ (Section IV-C1).

Finally, all active subspaces in EDSSC summary $\mathbb{S}^t$ will be checked if they are still active at timestamp t. The inactive subspaces will be removed from $\mathbb{S}^t$ to the remove reservoir $\mathbb{D}^t$ (Section IV-C2).

Since the variables used for change detection consume ignorable memory, the memory usage of EDSSC is mainly dominated by the saving of representatives. According to (13), the number of saved points is approximately logarithmic to the total number of points. Therefore, for n points, the approximate space complexity will be $O(d\log^2(n))$, which is comparable to the state-of-the-art online RBSC algorithms (e.g., SSSC, SLSR, and SLRR) and much less than those of the static RBSC algorithms [e.g., $O(dn^2)$ of SSC and LRR].

In the static learning phase, the time complexity of EDSSC is mainly determined by the solving of (5), which is approximately $O(T_0^3)$ [23]. While in the second phase, EDSSC needs to compute (15) for the following $n - T_0$ points and (5) if subspace emergence is detected, which is approximately $O(\zeta n_o^3 + (n - T_0)\log^2(n))$. Here, ζ denotes the number of times that subspace emergence detection is triggered and n_o is the number of points in the outlier reservoir when the emergence detection is triggered. Due to $T_0, n_o \ll n$, the time complexity of EDSSC is much smaller than those of LRR ($O(d^2n + n^3)$) and SSC ($O(dn^3)$) [23] and relatively larger than those of SSSC, SLRR and online LRR due to the ζ times computing of (5). However, such an additional computational cost, that is, $O(\zeta n_o^3)$, is inevitable to find the new subspaces which are not learned in the static phase instead of assuming all the subspaces having been learned in the first phase. The time complexity of EDSSC would be comparable with the other algorithms when processing data streams without any new subspaces emergence.

V. Experiments

In this section, experiments are performed on publicly available datasets to evaluate the performance of EDSSC.

TABLE I
EVOLUTIONARY DATA STREAMS USED IN EXPERIMENTS

	Data sets	#samples	Dim.	#classes	ini.	emer.
Small	AR	1400	167	100	70	30
	Ex-YaleB	2432	2016	38	20	18
	MPIE	4400	115	100	70	30
Medium	USPS	9298	256	10	4	6
	PenDigits	10992	16	10	4	6
	EMNIST-letter	13000	784	26	18	8
Large	MNIST 30K	30000	784	10	4	6

Moreover, the state-of-the-art online SC algorithms, that is, SSSC [23], SLRR [23], SLSR [23], and OLRSC [25], as well as DSC algorithms CEDAS [9] and STRAP [7], are selected as baseline algorithms. Section V-B demonstrates the influence of parameters on the performance of the proposed algorithm. Section V-C compares the results of all the evaluated algorithms on small-scale evolving data streams. In Section V-D, we investigate the performance of these evaluated algorithms on medium-scale evolving data streams. In addition, the performance of EDSSC, as well as other baseline algorithms on evolving large-scale data streams, are reported in Section V-E.

A. Datasets and Experimental Setup

The experiments are performed on seven evolutionary streams generated from seven public datasets. Particularly, they are divided into three categories according to their scales, that is, small scale, medium scale, and large scale, as shown in Table I. For computational efficiency, some datasets have been preprocessed. For AR datasets, we select 1400 images equally from 100 persons and the dimension of each sample has been reduced from 19 800 to 167. Similar to AR, we also use a subset of the MPIE dataset which consists of 4400 images from 100 persons of the MPIE dataset. EMNIST-letter consists of 13 000 samples of 26 English letters and MNIST 30K is a subset whose samples are randomly selected from the MNIST dataset. In order to generate the evolutionary data stream, in the initial stage, we only select some samples from some of the classes and the samples of classes that are not selected emerge in the online phase.

We compare the proposed algorithm with four state-of-the-art online RBSC algorithms, that is, SSSC [23], SLRR [23], SLSR [23], OLRSC [25], and two state-of-the-art related DSC algorithms, CEDAS [9] and STRAP [7]. All the algorithms are implemented with MATLAB. For SSSC, SLRR, SLSR, and OLRSC, there exists a common and vital parameter λ which is utilized to balance the data fidelity and the regularization term when solving the ℓ_1-minimization problem. For CEDAS and STRAP, a parameter r is required as an input to control the radius of a cluster. For a fair comparison, the parameters of all algorithms are tuned for obtaining the best performance, as summarized in Table II.

All the algorithms are measured using accuracy and NMI between the results given by the algorithms and the ground truth. The values of accuracy and NMI are real numbers between 0 and 1. Particularly, larger values mean the given result matches the ground truth more. In our experiments, the accuracy and NMI are the average values obtained by running each program ten times on each data stream.

TABLE II
PARAMETER SETTINGS FOR DIFFERENT ALGORITHMS

Data sets	Ours	SSSC	SLRR	SLSR	OLRSC	CEDAS	STRAP
	τ, f	λ,ϵ	λ	λ	λ	r	r
AR	0.4,20	1e-3,1e-7	3.1	1e-3	0.0213	1200	1500
Ex-YaleB	0.5,10	0.5,0.3	2.5	1e-4	0.0223	2000	2000
MPIE	0.4,20	1e-3,1e-5	0.7	0.1	0.0933	0.1	0.1
USPS	0.4,15	1e-3,1e-6	3.1	1e-3	0.0625	10	8.5
PenDigits	0.5,25	1e-3,1e-7	0.6	1e-3	0.2500	100	100
EMNIST-letter	0.4,10	0.1,1e-4	1e-3	0.1	0.3162	100	100
MNIST 30K	0.4,30	1e-3,0.8	3.1	0.01	0.0357	7.5	7

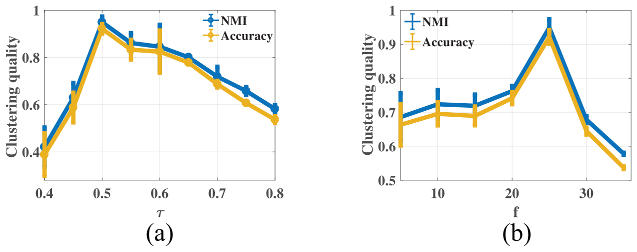

Fig. 4. Parameter sensitivity of EDSSC on parameters. (a) Influence of τ, where $f = 25$. (b) Influence of $f = 25$, where $\tau = 0.5$.

B. Impact of Parameters

Before comparing the performance of all algorithms, we would like to investigate the influence of parameters in EDSSC. There are two key parameters, that is, τ and f, which would impact EDSSC. Precisely, τ is used to identify the normal points from outliers and f in (22) controls the threshold η in (21) for change detection in PH tests. To study how these parameters impact on EDSSC, we chose different parameters settings of τ and f and run the EDSSC on the PenDigits stream. The corresponding results are illustrated in Fig. 4, from which the following observation could be obtained.

Fig. 4(a) depicts the clustering quality (accuracy and NMI) of EDSSC under different τ settings (we keep $f = 25$). When the τ is relatively small (e.g., $\tau < 0.5$) or large (e.g., $\tau > 0.75$), the clustering quality is lower. The reason is a smaller τ results in that more outliers are accepted as normal points and a larger τ gets more normal points rejected as outliers. Moreover, a larger τ is easier to trigger PH detection getting clustering quality decreased.

We report the results of the effect of f on the EDSSC method (we fix $\tau = 0.5$) in Fig. 4(b) from which we find that f will impact the clustering quality. If f is set too large for the PH test to get triggered, then the new emerging subspaces would not be found timely.

C. Online Subspace Clustering on Small-Scale and Evolving Data Streams

In this section, we focus on studying the performance of EDSSC and baseline algorithms on small-scale and evolving data streams. We report their performance in Table III from which we can find the following.

Our EDSSC succeeds in achieving high clustering accuracy and NMI in processing the three small-scale and evolving data

TABLE III
PERFORMANCE COMPARISON AMONG DIFFERENT ALGORITHMS ON THREE SMALL-SCALE DATA STREAMS (AR, EXYALEB, AND MPIE)

Datasets	AR		ExYaleB		MPIE	
Algorithm	Acc.(%)	NMI(%)	Acc.	NMI	Acc.	NMI
EDSSC	**61.96**	75.35	**75.01**	**86.47**	**79.71**	**89.34**
SSSC [23]	41.43	63.04	36.17	57.69	62.72	79.43
SLRR [23]	45.32	77.47	55.72	70.99	64.33	85.33
SLSR [23]	59.69	**84.35**	53.28	70.33	31.44	69.47
OLRSC [25]	60.57	70.54	57.14	74.43	60.93	80.00
CEDAS [9]	27.43	60.21	27.43	59.70	11.52	12.14
STRAP [7]	18.84	34.31	16.12	24.40	32.00	41.01

TABLE IV
PERFORMANCE COMPARISON DIFFERENT ALGORITHMS OVER THREE MEDIUM-SCALE DATA STREAMS (USPS, PENDIGITS, AND EMNIST-LETTER) AND ONE LARGE-SCALE DATA STREAM (MNIST 30K)

Datasets	USPS		PenDigits		EMNIST-letter		MNIST30K	
Algorithm	Acc.	NMI	Acc.	NMI	Acc.	NMI	Acc.	NMI
EDSSC	**67.01**	**78.65**	**80.67**	**87.09**	**67.62**	75.58	**69.73**	**85.37**
SSSC [23]	44.46	38.29	40.78	39.79	44.23	49.08	56.21	64.97
SLRR [23]	60.43	67.20	44.57	50.41	59.31	72.87	47.43	56.01
SLSR [23]	50.43	59.60	43.77	47.90	65.60	**83.50**	44.18	55.56
OLRSC [25]	56.12	55.84	57.54	49.10	28.53	38.69	34.91	49.91
CEDAS [9]	40.92	39.67	56.70	64.89	46.35	47.97	46.35	47.97
STRAP [7]	44.11	49.61	34.40	34.07	27.02	38.91	27.02	38.91

streams. For example, on the ExYaleB data stream, EDSSC achieves 75.01% accuracy and 86.47% NMI compared with 57.14% accuracy and 74.43% NMI of OLRSC which has the best performance among all baseline algorithms. It proves the feasibility of our EDSSC algorithm, especially the effectiveness of the detection strategy of subspace evolution, the estimation algorithm of the number of subspaces, and the proposal of the ASCI index.

It can be observed that our EDSSC achieves much higher clustering quality than the state-of-the-art RBSC algorithms, that is, SSSC, SLRR, SLSR, and OLRSC. Such superiority of EDSSC is more obvious on the ExYaleB and MPIE data streams. The reason is that these RBSC algorithms are based on the assumption that the subspace structure should be unchanged. This assumption leads to that they fail to detect and adapt the evolving subspace, which naturally results in the failure in the clustering. It should be pointed out that for the AR stream, the NMI value of EDSSC is slightly less than that of SLSR. One possible reason is that the spatial distribution of data points in the AR datasets is rather complex. Hence, it is relatively difficult to accurately estimate the number of subspaces underlying the AR stream without any prior information (EDSSC estimates 126 subspaces). However, SLSR, SSSC, SLRR, and OLRSC need an accurate number of subspaces as prior information. In the experiments of this article, we provide them with the accurate number of subspaces, which will promote their NMI values.

Compared with the two state-of-the-art DSC algorithms, that is, STRAP and CEDAS, EDSSC is more suitable in processing these high-dimensional evolutionary data streams. This is mainly because STRAP and CEDAS are based on traditional distance measurements which are not effective in high-dimensional space to indicate the aggregation of the points belonging to the same subspace. However, our EDSSC fully exploits the self-expressiveness property of the data points, thus achieves better performance in high-dimensional space.

D. Online Subspace Clustering on Medium-Scale and Evolving Data Streams

This section investigates the proposed method as well as baseline algorithms on three medium-scale handwritten digit (or character) streams, that is, USPS, PenDigits, EMNIST-letter. Table IV reports the accuracy and NMI of the tested algorithms, from which the following could be observed.

The proposed EDSSC achieves the best performance on the USPS and PenDigits streams among all the performed algorithms. For example, on the USPS data stream, the accuracy and NMI of EDSSC can reach 67.01% and 78.65% followed by SLRR whose accuracy and NMI only 60.43% and 67.20%. For the EMNIST-letter stream, EDSSC achieves the highest accuracy with 67.62%. The NMI of EDSSC on EMNIST-letter is 75.58%, which is less than SLSR (83.50%). However, as discussed before, it is mainly because the spatial distribution of the EMNIST-letter stream is complex. The number of subspaces estimated by EDSSC is 32, which is slightly larger than the actual.

It can be observed that the performance of our EDSSC is more stable when handling different kinds of evolving data streams. Nevertheless, the performance of the baseline algorithms fluctuates. For example, SLSR performs relatively acceptable on processing the EMNIST-letter stream with accuracy = 65.60% and NMI = 83.50%. However, it achieves only accuracy = 50.43% and NMI = 59.60% in USPS stream processing. The performance of SSSC on the USPS stream as well as the EMNIST-letter stream is not comparable to the other RBSC algorithms but its performance becomes relatively acceptable on the PenDigits stream processing. The possible reason is that these baseline algorithms are based on different theory to capture the subspace structure. For example. SSSC is based on sparse restriction while our EDSSC is based on the low-rank restriction in (3). The latter is more powerful to capture the structure among different kinds of streams [23].

E. Online Subspace Clustering on Large-Scale and Evolving Data Stream

We further evaluate the proposed and the baseline algorithms on a large-scale and evolving data stream. The the MNIST 30K stream is used in this experiment which consists of 30000 data points randomly selected from 10 classes in the MNIST database. Besides subspace emergence, we further add subspace disappearance, recurrence in the data stream to test the performance of the algorithms. The temporal distribution and evolving property of MNIST 30K are shown in Fig. 5.

As can be seen from Fig. 5, we divide the data stream into five phases denoted as P1–P5, respectively. There are five subspaces emerging in P1 followed by three subspaces and two subspaces emerging in the second and third phases,

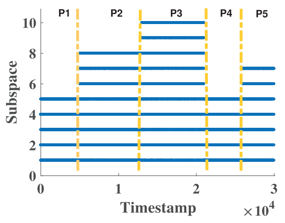

Fig. 5. Temporal distribution and the evolving property of MNIST 30K.

respectively. Then, in P4, five subspaces disappear and two of them recur in the next phase.

The performance of the algorithms on the MNIST 30K stream has been reported on the right side of Table IV. As shown in Table IV, the proposed EDSSC outperforms the baseline algorithms in the accuracy and NMI value of the result. Furthermore, in order to study the performance of each algorithm for subspace evolution, we depict the real-time number of subspace recovered by each algorithm on MNIST 30K in Fig. 6. From Fig. 6(a), it can be observed that EDSSC is effective in detecting and adapting the subspace evolution with the three types of subspace evolution being detected accurately. However, in Fig. 6(b), it can be found that the number of subspaces keeps as equal to 5 during the entire process. The reason is that SSSC, SLRR, SLSR, and OLRSC cannot deal with the subspace evolution and assume all the subspaces should be covered in the static learning phase. Therefore, the number of subspaces is unchanged. In addition, these four algorithms actually cannot estimate the number of subspaces in the static phase and they require the number of the subspace must be input as prior information. In Fig. 6(c), we can find that the CEDAS has the potential to deal with the subspace emergence and disappearance. However, it cannot recover the subspace accurately because CEDAS is based on traditional distance measurement which is not effective in high-dimensional space. Meanwhile, note that the subspace recurrence cannot be detected at all. As shown in Fig. 6(d), STRAP can only detect subspace emergence. Therefore, the number of subspaces can reach 9 from 5 around $t = 14\,000$. But after that, the number of subspaces keeps unchanged. This is because the subspace disappearance and recurrence can not be detected or acted by STRAP.

VI. Conclusion

In this article, we focused on one unsolved but vital problem, that is, how to cluster the evolving and high-dimensional data streams. To this end, a novel DSC algorithm, called EDSSC, was proposed which can perform online SC on the evolving and high-dimensional data streams. EDSSC is a two-phase algorithm, including a static learning phase and an online clustering phase. Compared with the state-of-the-art algorithms, EDSSC satisfactorily handles the following issues. First, in the static learning phase, our EDSSC was capable of estimating the number of subspaces by the proposed method (10). Second, by proposing a data structure, called EDSSC summary, EDSSC perfectly reached a balance between the two competing goals: saving more points for the accuracy or discarding more points for efficiency. Third,

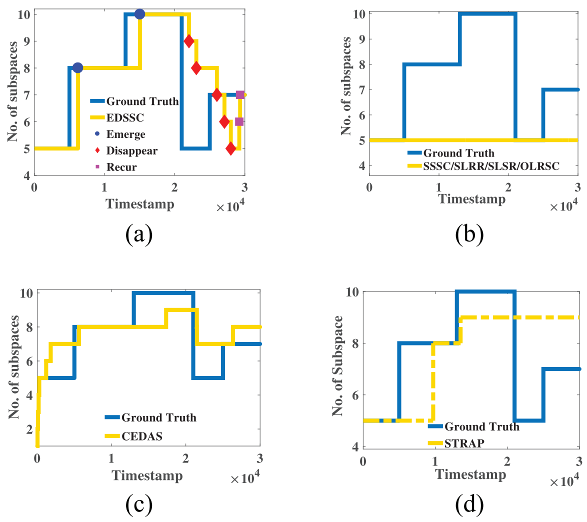

Fig. 6. Real-time number of the subspace recovered by algorithms on MNIST 30K. (a) EDSSC. (b) SSSC/SLRR/SLSR/OLRSC. (c) CEDAS. (d) STRAP.

EDSSC does not make the assumption that the subspace structure of the data streams must be unchanged. Instead, EDSSC can detect and act subspace evolution properly by the proposed subspace detection strategy based on the PH test. Finally, in the learning phase, the arriving point can be assigned to the proper subspace with the help of the proposed ASCI index. We have proved that ASCI is better than the commonly utilized SCI index.

In future studies, we will concentrate on exploring a proper strategy enabling EDSSC to detect gradual subspace evolution. In addition, we will focus on the automatic optimization of relevant parameters affecting the performance.

Acknowledgment

The authors would like to acknowledge Timo Huuhtanen for his help in polishing this article and Zhang *et al.* [7], Hyde *et al.* [9], and Peng *et al.* [23] for their providing source codes.

References

[1] E. Elhamifar and R. Vidal, "Sparse subspace clustering: Algorithm, theory, and applications," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 35, no. 11, pp. 2765–2781, Nov. 2013.

[2] R. Basri and D. W. Jacobs, "Lambertian reflectance and linear subspaces," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 25, no. 2, pp. 218–233, Feb. 2003.

[3] X. J. Hunt and R. Willett, "Online data thinning via multi-subspace tracking," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 41, no. 5, pp. 1173–1187, May 2019.

[4] J. Luo, L. Jiao, F. Liu, S. Yang, and W. Ma, "A Pareto-based sparse subspace learning framework," *IEEE Trans. Cybern.*, vol. 49, no. 11, pp. 3859–3872, Nov. 2019.

[5] C. C. Aggarwal, J. Han, J. Wang, and P. S. Yu, "A framework for clustering evolving data streams," in *Proc. 29th Int. Conf. Very Large Data Bases Vol. 29*, 2003, pp. 81–92.

[6] F. Cao, M. Estert, W. Qian, and A. Zhou, "Density-based clustering over an evolving data stream with noise," in *Proc. SIAM Int. Conf. Data Min.*, 2006, pp. 328–339.

[7] X. Zhang, C. Furtlehner, C. Germain-Renaud, and M. Sebag, "Data stream clustering with affinity propagation," *IEEE Trans. Knowl. Data Eng.*, vol. 26, no. 7, pp. 1644–1656, Jul. 2014.

[8] S. Gong, Y. Zhang, and G. Yu, "Clustering stream data by exploring the evolution of density mountain," *Proc. VLDB Endowm.*, vol. 11, no. 4, pp. 393–405, 2017.

[9] R. Hyde, P. Angelov, and A. R. MacKenzie, "Fully online clustering of evolving data streams into arbitrarily shaped clusters," *Inf. Sci.*, vols. 382–283, pp. 96–114, Mar. 2017.

[10] M. Oliveira and J. Gama, "A framework to monitor clusters evolution applied to economy and finance problems," *Intell. Data Anal.*, vol. 16, no. 1, pp. 93–111, 2012.

[11] G. Xia, H. Sun, L. Feng, G. Zhang, and Y. Liu, "Human motion segmentation via robust kernel sparse subspace clustering," *IEEE Trans. Image Process.*, vol. 27, no. 1, pp. 135–150, Jan. 2018.

[12] W. Deng, J. Hu, and J. Guo, "Face recognition via collaborative representation: Its discriminant nature and superposed representation," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 40, no. 10, pp. 2513–2521, Oct. 2018.

[13] S. Minaee and Y. Wang, "An ADMM approach to masked signal decomposition using subspace representation," *IEEE Trans. Image Process.*, vol. 28, no. 7, pp. 3192–3204, Jul. 2019.

[14] C. You, C. Li, D. P. Robinson, and R. Vidal, "A scalable exemplar-based subspace clustering algorithm for class-imbalanced data," in *Proc. Eur. Conf. Comput. Vis. (ECCV)*, Sep. 2018, pp. 68–85.

[15] J. Wright, A. Y. Yang, A. Ganesh, S. S. Sastry, and Y. Ma, "Robust face recognition via sparse representation," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 31, no. 2, pp. 210–227, Feb. 2009.

[16] G. Liu, Z. Lin, S. Yan, J. Sun, Y. Yu, and Y. Ma, "Robust recovery of subspace structures by low-rank representation," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 35, no. 1, pp. 171–184, Jan. 2013.

[17] A. Hashemi and H. Vikalo, "Evolutionary self-expressive models for subspace clustering," *IEEE J. Sel. Topics Signal Process.*, vol. 12, no. 6, pp. 1534–1546, Dec. 2018.

[18] J. Sui, Z. Liu, L. Liu, B. Peng, T. Liu, and X. Li, "Online non-cooperative radar emitter classification from evolving and imbalanced pulse streams," *IEEE Sensors J.*, vol. 20, no. 14, pp. 7721–7730, Jul. 2020.

[19] J. Yang, J. Liang, K. Wang, P. Rosin, and M.-H. Yang, "Subspace clustering via good neighbors," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 42, no. 6, pp. 1537–1544, Jun. 2020.

[20] B. A. Kelkar and S. F. Rodd, "Subspace clustering—A survey," in *Data Management, Analytics and Innovation*. Singapore: Springer, 2019, pp. 209–220.

[21] R. Pasunuri, V. C. Venkaiah, and A. Srivastava, "Clustering high-dimensional data: A reduction-level fusion of PCA and random projection," in *Recent Developments in Machine Learning and Data Analytics*. Singapore: Springer, 2019, pp. 479–487.

[22] X. Peng, Z. Yu, Z. Yi, and H. Tang, "Constructing the L2-graph for robust subspace learning and subspace clustering," *IEEE Transa. Cybern.*, vol. 47, no. 4, pp. 1053–1066, Apr. 2017.

[23] X. Peng, H. Tang, L. Zhang, Z. Yi, and S. Xiao, "A unified framework for representation-based subspace clustering of out-of-sample and large-scale data," *IEEE Trans. Neural Netw. Learn. Syst.*, vol. 27, no. 12, pp. 2499–2512, Dec. 2016.

[24] B. Li, R. Liu, J. Cao, J. Zhang, Y.-K. Lai, and X. Liu, "Online low-rank representation learning for joint multi-subspace recovery and clustering," *IEEE Trans. Image Process.*, vol. 27, no. 1, pp. 335–348, Jan. 2019.

[25] J. Shen, P. Li, and H. Xu, "Online low-rank subspace clustering by basis dictionary pursuit," in *Proc. 33rd Int. Conf. Mach. Learn.*, 2016, pp. 622–631.

[26] E. S. Page, "Continuous inspection schemes," *Biometrika*, vol. 41, nos. 1–2, pp. 100–115, 1954.

[27] J. Sui, Z. Liu, L. Liu, A. Jung, T. Liu, B. Peng, and X. Li, "Sparse subspace clustering for evolving data streams," in *Proc. IEEE Int. Conf. Acoust. Speech Signal Process. (ICASSP)*, Brighton, U.K., 2019, pp. 7455–7459.

[28] C. C. Aggarwal, J. Han, J. Wang, and P. S. Yu, "A framework for projected clustering of high dimensional data streams," in *Proc. 13th Int. Conf. Very Large Data Bases Vol. 30*, 2004, pp. 852–863.

[29] J. Shao, Y. Tan, L. Gao, Q. Yang, C. Plant, and I. Assent, "Synchronization-based clustering on evolving data stream," *Inf. Sci.*, vol. 501, pp. 573–587, Oct. 2019.

[30] C. Böhm, C. Plant, J. Shao, and Q. Yang, "Clustering by synchronization," in *Proc. 16th ACM SIGKDD Int. Conf. Knowl. Discover. Data Min.*, 2010, pp. 583–592.

[31] J. Shao, X. He, C. Böhm, Q. Yang, and C. Plant, "Synchronization-inspired partitioning and hierarchical clustering," *IEEE Trans. Knowl. Data Eng.*, vol. 25, no. 4, pp. 893–905, Apr. 2013.

[32] J. Shao, X. He, Q. Yang, C. Plant, and C. Böhm, "Robust synchronization-based graph clustering," in *Proc. Pac.-Asia Conf. Knowl. Discover. Data Min.*, 2013, pp. 249–260.

[33] J. Shao, C. Gao, W. Zeng, J. Song, and Q. Yang, "Synchronization-inspired co-clustering and its application to gene expression data," in *Proc. IEEE Int. Conf. Data Min. (ICDM)*, New Orleans, LA, USA, 2017, pp. 1075–1080.

[34] J. Shao, F. Huang, Q. Yang, and G. Luo, "Robust prototype-based learning on data streams," *IEEE Trans. Knowl. Data Eng.*, vol. 30, no. 5, pp. 978–991, May 2018.

[35] I. Ntoutsi, A. Zimek, T. Palpanas, P. Kröger, and H.-P. Kriegel, "Density-based projected clustering over high dimensional data streams," in *Proc. SIAM Int. Conf. Data Min.*, 2012, pp. 987–998.

[36] M. Hassani, P. Spaus, M. M. Gaber, and T. Seidl, "Density-based projected clustering of data streams," in *Proc. Int. Conf. Scalable Uncertainty Manag.*, 2012, pp. 311–324.

[37] H.-P. Kriegel, P. Kröger, I. Ntoutsi, and A. Zimek, "Towards subspace clustering on dynamic data: An incremental version of predecon," in *Proc. 1st Int. Workshop Novel Data Stream Pattern Min. Techn.*, 2010, pp. 31–38.

[38] M. Hassani, P. Kranen, R. Saini, and T. Seidl, "Subspace anytime stream clustering," in *Proc. 26th Int. Conf. Sci. Stat. Database Manag.*, 2014, p. 37.

[39] H.-P. Kriegel, P. Kröger, I. Ntoutsi, and A. Zimek, "Density based subspace clustering over dynamic data," in *Proc. 23rd Int. Conf. Sci. Stat. Database Manag.*, vol. 6809. Portland, OR, USA, Jul. 2011, pp. 387–404.

[40] C. Bohm, K. Railing, H.-P. Kriegel, and P. Kroger, "Density connected clustering with local subspace preferences," in *Proc. 4th IEEE Int. Conf. Data Min. (ICDM'04)*, Brighton, U.K., 2004, pp. 27–34.

[41] C. C. Aggarwal, J. L. Wolf, P. S. Yu, C. Procopiuc, and J. S. Park, "Fast algorithms for projected clustering," *ACM SIGMOD Rec.*, vol. 28, no. 2, pp. 61–72, 1999.

[42] R. Agrawal, J. Gehrke, D. Gunopulos, and P. Raghavan, "Automatic subspace clustering of high dimensional data for data mining applications," *ACM SIGMOD Rec.*, vol. 27, no. 2, pp. 94–105, 1998.

[43] K. Kailing, H.-P. Kriegel, and P. Kröger, "Density-connected subspace clustering for high-dimensional data," in *Proc. SIAM Int. Conf. Data Min.*, 2004, pp. 246–256.

[44] C. Böhm, K. Kailing, P. Kröger, and A. Zimek, "Computing clusters of correlation connected objects," in *Proc. ACM SIGMOD Int. Conf. Manag. Data*, 2004, pp. 455–466.

[45] E. Achtert, C. Böhm, J. David, P. Kröger, and A. Zimek, "Robust clustering in arbitrarily oriented subspaces," in *Proc. SIAM Int. Conf. Data Min.*, 2008, pp. 763–774.

[46] J. Shao, X. Wang, Q. Yang, C. Plant, and C. Böhm, "Synchronization-based scalable subspace clustering of high-dimensional data," *Knowl. Inf. Syst.*, vol. 52, no. 1, pp. 83–111, 2017.

[47] C.-Y. Lu, H. Min, Z.-Q. Zhao, L. Zhu, D.-S. Huang, and S. Yan, "Robust and efficient subspace segmentation via least squares regression," in *Proc. Eur. Conf. Comput. Vis.*, 2012, pp. 347–360.

[48] X. Zhu, S. Zhang, Y. Li, J. Zhang, L. Yang, and Y. Fang, "Low-rank sparse subspace for spectral clustering," *IEEE Trans. Knowl. Data Eng.*, vol. 31, no. 8, pp. 1532–1543, Aug. 2019.

[49] X. Zhu, S. Zhang, R. Hu, W. He, C. Lei, and P. Zhu, "One-step multi-view spectral clustering," *IEEE Trans. Knowl. Data Eng.*, vol. 31, no. 10, pp. 2022–2034, Oct. 2019.

[50] Y. Yang, F. Shen, Z. Huang, H. T. Shen, and X. Li, "Discrete nonnegative spectral clustering," *IEEE Trans. Knowl. Data Eng.*, vol. 29, no. 9, pp. 1834–1845, Sep. 2017.

[51] Q. Wang, Z. Qin, F. Nie, and X. Li, "Spectral embedded adaptive neighbors clustering," *IEEE Trans. Neural Netw. Learn. Syst.*, vol. 30, no. 4, pp. 1265–1271, Apr. 2019.

[52] M. Brbić and I. Kopriva, "ℓ_0-motivated low-rank sparse subspace clustering," *IEEE Trans. Cybern.*, vol. 50, no. 4, pp. 1711–1725, Apr. 2020.

[53] A. Sumarsono and Q. Du, "Low-rank subspace representation for estimating the number of signal subspaces in hyperspectral imagery," *IEEE Trans. Geosci. Remote Sens.*, vol. 53, no. 11, pp. 6286–6292, Nov. 2015.

[54] Z. Lin, M. Chen, and Y. Ma, "The augmented lagrange multiplier method for exact recovery of corrupted low-rank matrices," 2010. [Online]. Available: arXiv:1009.5055.

[55] J. Lu, A. Liu, F. Dong, F. Gu, J. Gama, and G. Zhang, "Learning under concept drift: A review," *IEEE Trans. Knowl. Data Eng.*, vol. 31, no. 12, pp. 2346–2363, Dec. 2019.

[56] V. M. Nidhi, V. Gupta, and R. Vig, “Methods to investigate concept drift in big data streams,” in *Knowledge Computing and Its Applications: Knowledge Manipulation and Processing Techniques*, vol. 1. Singapore: Springer, 2018, pp. 51–74.

[57] G. I. Webb, R. Hyde, H. Cao, H. L. Nguyen, and F. Petitjean, “Characterizing concept drift,” *Data Min. Knowl. Discover.*, vol. 30, no. 4, pp. 964–994, 2016.

[58] Y. Sun, K. Tang, L. L. Minku, S. Wang, and X. Yao, “Online ensemble learning of data streams with gradually evolved classes,” *IEEE Trans. Knowl. Data Eng.*, vol. 28, no. 6, pp. 1532–1545, Jun. 2016.

[59] Z. Li, L.-F. Cheong, S. Yang, and K.-C. Toh, “Simultaneous clustering and model selection: Algorithm, theory and applications,” *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 40, no. 8, pp. 1964–1978, Aug. 2018.

[60] J. Yang, J. Liang, K. Wang, Y.-L. Yang, and M.-M. Cheng, “Automatic model selection in subspace clustering via triplet relationships,” in *Proc. 32nd AAAI Conf. Artif. Intell.*, 2018, pp. 4358–4365.

[61] J. Liang, J. Yang, M.-M. Cheng, P. L. Rosin, and L. Wang, “Simultaneous subspace clustering and cluster number estimating based on triplet relationship,” *IEEE Trans. Image Process.*, vol. 28, no. 8, pp. 3973–3985, Aug. 2019.

[62] E. Elhamifar, G. Sapiro, and S. S. Sastry, “Dissimilarity-based sparse subset selection,” *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 38, no. 11, pp. 2182–2197, Nov. 2016.

[63] S. Ren, B. Liao, W. Zhu, Z. Li, W. Liu, and K. Li, “The gradual resampling ensemble for mining imbalanced data streams with concept drift,” *Neurocomputing*, vol. 286, pp. 150–166, Apr. 2018.

[64] J. Sui, Z. Liu, A. Jung, L. Liu, and X. Li, “Dynamic clustering scheme for evolving data streams based on improved strap,” *IEEE Access*, vol. 6, pp. 46157–46166, 2018.

[65] K.-C. Lee, J. Ho, and D. J. Kriegman, “Acquiring linear subspaces for face recognition under variable lighting,” *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 27, no. 5, pp. 684–698, May 2005.

[66] A. Georghiades, P. Belhumeur, and D. Kriegman, “From few to many: Illumination cone models for face recognition under variable lighting and pose,” *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 23, no. 6, pp. 643–660, Jun. 2001.

[67] X. Wu, P. Li, and X. Hu, “Learning from concept drifting data streams with unlabeled data,” *Neurocomputing*, vol. 92, pp. 145–155, Sep. 2012.

[68] D. V. Hinkley, “Inference about the change-point from cumulative sum tests,” *Biometrika*, vol. 58, no. 3, pp. 509–523, 1971.

Jinping Sui was born in Jilin, China, in 1990. He received the B.S. degree in communication engineering from Northeastern University, Shenyang, China, in 2013, and the M.S. degree in information and communication engineering from the College of Electronic Science and Technology, National University of Defense Technology (NUDT), Changsha, China, in 2015. He is currently pursuing the Ph.D. degree with the College of Electronic Science and Technology, NUDT, and is also with the Department of Computer Science, Aalto University, Espoo, Finland.

His research interests include data stream clustering and machine learning for big data.

Zhen Liu was born in Jiangsu, China, in 1983. He received the B.S. degree from Zhejiang University, Hangzhou, China, in 2006, and the Ph.D. degree from the National University of Defense Technology (NUDT), Changsha, China, in 2013.

He is an Associate Professor with the College of Electronic Science and Technology, NUDT. His research interests include radar signal processing, compressed sensing, and machine learning.

Li Liu received the B.Sc. degree in communication engineering, the M.Sc. degree in photogrammetry and remote sensing, and the Ph.D. degree in information and communication engineering from the National University of Defense Technology (NUDT), Changsha, China, in 2003, 2005, and 2012, respectively.

She is an Associate Professor with the College of System Engineering and the Center for Machine Vision and Signal Analysis, University of Oulu, Oulu, Finland. During her Ph.D. study, she was a visiting student with the University of Waterloo, Waterloo, ON, Canada, from 2008 to 2010. From 2015 to 2016, she visited the Multimedia Laboratory, Chinese University of Hong Kong, Hong Kong, for ten months. From 2016 to 2018, she was a Senior Researcher with the Machine Vision Group, University of Oulu. Her papers have currently over 2700 citations in Google Scholar. Her current research interests include computer vision and machine learning.

Dr. Liu was a Co-Chair of nine international workshops at CVPR, ICCV, and ECCV. She was a Guest Editor of special issues for IEEE TRANSACTIONS ON PATTERN ANALYSIS AND MACHINE INTELLIGENCE and *International Journal of Computer Vision*. She currently serves as an Associate Editor for *Pattern Recognition Letters* and *Visual Computer Journal*. She serves as an Area Chair for ICME2020 and ACCV2020.

Alexander Jung received the Diplom-Ingenieur and Dr.techn. degrees in electrical engineering from the Vienna University of Technology, Vienna, Austria, in 2008 and 2011, respectively.

Since 2015, he has been an Assistant Professor with the Department of Computer Science, Aalto University, Espoo, Finland. His research interests revolve around fundamental limits and efficient algorithms for learning from massive network-structured data (big data over networks).

Xiang Li was born in Hunan, China, in 1967. He received the B.S. degree from Xidian University, Xi’an, China, in 1989, and the Ph.D. degree from the National University of Defense Technology (NUDT), Changsha, China in 1998.

He is a Professor with NUDT. His research interests include signal processing, automation target recognition, and machine learning.