# Two-stage Sparse Representation Clustering for Dynamic Data Streams

Jie Chen, Member, IEEE, Zhu Wang, Shengxiang Yang, Senior Member, IEEE, Hua Mao

Abstract—Data streams are a potentially unbounded sequence of data objects, and the clustering of such data is an effective way of identifying their underlying patterns. Existing data stream clustering algorithms face two critical issues: evaluating the relationship among data objects with individual landmark windows of fixed size, and passing useful knowledge from previous landmark windows to the current landmark window. Based on sparse representation techniques, this paper proposes a two-stage sparse representation clustering (TSSRC) method. The novelty of the proposed TSSRC algorithm comes from evaluating the effective relationship among data objects in the landmark windows with an accurate number of clusters. First, the proposed algorithm evaluates the relationship among data objects using sparse representation techniques. The dictionary and sparse representations are iteratively updated by solving a convex optimization problem. Second, the proposed TSSRC algorithm presents a dictionary initialization strategy that seeks representative data objects by making full use of the sparse representation results. This efficiently passes previously learned knowledge to the current landmark window over time. Moreover, the convergence and sparse stability of TSSRC can be theoretically guaranteed in continuous landmark windows under certain conditions. Experimental results on benchmark datasets demonstrate the effectiveness and robustness of TSSRC.

Index Terms—Data stream, clustering, sparse representation, dictionary learning.

## I. Introduction

WITH the rapid development of software and hardware technologies, large amounts of data are being collected in dynamic environments [1]–[4]. Such online data are usually referred to as data streams. A data stream is a potentially unbounded, continuous, massive sequence of data objects in a dynamic environment. The goal of data stream clustering is to place data objects into their respective groups according to particular similarity measures.

Manuscript received June 1, 2021; revised April 26, 2022; accepted September 4, 2022. This work was supported in part by National Natural Science Foundation of China (NSFC) under Grants 61303015 and 61673331, in part by National Key Project under Grant GJXM92579, and in part by Sichuan Science and Technology Program under Grant 2021YJ0078. (Corresponding author: Zhu Wang.)

J. Chen is with the College of Computer Science, Sichuan University, Chengdu 610065, China (E-mail: chenjie2010@scu.edu.cn).

Z. Wang is with the Law School, Sichuan University, Chengdu 610065, China (E-mail: wangzhu@scu.edu.cn).

S. Yang is with the School of Computer Science and Informatics, De Montfort University, Leicester LE1 9BH, U. K. (e-mail: syang@dmu.ac.uk).

H. Mao is with the Department of Computer and Information Sciences, Northumbria University, Newcastle, NE1 8ST, U. K. (e-mail: hua.mao@northumbria.ac.uk).

The unique characteristics of data streams typically lead to specific challenges in data stream clustering [3]. For instance, the probability distribution of data streams may continuously evolve over time, which implies that the underlying clustering models should be adaptively updated to characterize the intrinsic structure of the current data stream at any time, and also eliminate the effects of outdated data and noise. As data objects often arrive continuously, the clustering of data streams should satisfy certain memory requirements and time restrictions, with rapid detection of the presence of potential outliers. A number of clustering algorithms often perform well on a finite amount of data generated by an unknown, stationary probability distribution [5]–[9]. However, the additional considerations outlined above mean that these algorithms may be insufficient for the clustering of data streams.

Several traditional clustering approaches have been proposed for data streams [10]–[16]. Balanced iterative reducing and clustering using hierarchies (BIRCH) is a typical hierarchical clustering algorithm that uses a clustering feature (CF) vector to construct a height-balanced tree, with new clusters generated by pruning the leaves of this tree [16]. Moreover, the CF vector or its variants were adopted to incrementally summarize the data streams. For example, Fahy et al. presented an ant colony stream clustering (ACSC) algorithm that employs local density and local similarity produced by artificial ants to probabilistically pick and drop microclusters created by the CF vectors [12]. Huang et al. presented a multiview support vector domain description model for multiview data stream clustering [13]. These algorithms typically employ a local similarity strategy using the Euclidean distance between data objects to preserve local neighborhood information. However, estimating an appropriate size for the neighborhood remains an open question.

The nature of clustering implies that highly correlated data objects from the same cluster contain similar structural characteristics. Traditional clustering approaches take advantage of the spatial proximity of data objects, i.e., the computation of the Euclidean distance, to retain information in the cluster components [16]–[21]. However, it is insufficient to characterize the intrinsic structures of high-dimensional data objects in data streams. This motivates the development of alternative techniques for exploiting the intrinsic characteristics of the high-dimensional data objects [10], [22], [23], [23], [24]. For example, Krleža1 et al. proposed a statistical hierarchical clustering (SHC) algorithm that uses statistical inference

based on statistical distances to estimate statistical distributions on data streams [10]. Sui et al. presented an evolutionary dynamic sparse subspace clustering (EDSSC) algorithm based on sparse representation to address the problem of concept drift [22]. Li et al. proposed an efficient self-adaptive online data stream (ESA-Stream) clustering algorithm that self-adaptively learns parameters to cluster data streams in a dynamic online environment [25]. Data streams dynamically change as new clusters appear or others disappear. This would enable the number of clusters to change dynamically in each individual landmark window. Determining the exact number of clusters is essential in improving the quality of clustering, and is one of the difficulties encountered in previous algorithms [26]. Moreover, this also requires the clustering algorithm to be capable of incremental learning from large-scale dynamic data streams. Therefore, there is a need for a new learning strategy that adaptively passes previously learned knowledge to the current data stream over time.

Sparse representation has been widely studied and has achieved encouraging results in fields such as image processing, computer vision, and pattern recognition [8], [27]–[29]. Wang et al. proposed a robust rank constrained sparse learning (RRCSL) algorithm that combines sparse representation with an $l_{2,1}$-norm to learn a similarity graph for single view or multiview clustering [27]. Ding et al. presented a sparse representation-based intuitionistic fuzzy clustering (SRIFC) algorithm that clusters decision makers into several groups according to relation sparsity [28]. Gu et al. presented a fuzzy double $c$-means based on sparse self-representation (FDCM_SSR) algorithm that can simultaneously cluster two datasets with different dimensions [29]. Sparse representation essentially takes advantage of the self-expressiveness property of the data, enabling each data point to be effectively represented as a sparse linear combination of other points from the same class. Ideally, the sparse representation of a data object identifies highly correlated data objects from the same class, where the values of the nonzero elements correspond to a weight for each pair of data objects. This avoids the need to specify the proper neighborhood size when evaluating the similarity among data points.

Dictionary learning techniques have been widely combined with sparse representation models [30], [31]. For example, a number of online or batch dictionary learning methods have been developed for dealing with large-scale data objects [30]–[32]. These dictionary learning methods learn a compact dictionary from the large-scale data objects by incrementally updating the learned dictionary in the online process. However, data stream clustering must adapt rapidly to the changing dynamics of the data. This means that algorithms should store some of the data objects for a given period of time and later discard them, such as by employing a forgetting mechanism. As a result, incrementally updating the learned dictionary is currently unable to deal with the clustering of data streams. As the above discussion shows, there are few sparse representation techniques with online dictionary learning that can be used for the clustering of data streams.

In this paper, we present a two-stage sparse representation clustering (TSSRC) method for clustering high-dimensional data streams. TSSRC finds a discriminative sparse representation of the high-dimensional data that unravels the intrinsic relationship among data objects. In the first stage, TSSRC integrates the sparse constraint into a data object representation in order to learn a sparse matrix, which preserves the relationship among the data objects. We propose an iterative optimization method to efficiently improve the convergence speed of the TSSRC model. The learned sparse matrix is then employed as an affinity matrix for spectral clustering. In the second stage, a dictionary initialization strategy is employed to efficiently pass previously learned knowledge to the current landmark window over time. The convergence and sparse stability of TSSRC are theoretically analyzed, as these aspects are crucial in practical applications. Finally, we report the results of extensive experiments that demonstrate the performance of our proposed method through comparisons with several state-of-the-art data stream clustering methods.

The contributions of this study can be summarized as follows:

1) A sparse representation approach is incorporated into data stream clustering, enriching the measurements among data objects for robust data stream clustering.
2) A dictionary initialization strategy is proposed that effectively retains previously learned knowledge and provides an exact number of clusters for the current landmark window over time.
3) The convergence and sparse stability of TSSRC are theoretically guaranteed under certain conditions in continuous landmark windows.
4) Our extensive experimental results using benchmark databases demonstrate the effectiveness, stable convergence, and sparse stability of TSSRC for data stream clustering.

The remainder of this paper is organized as follows. In Section II, we present a brief review of sparse representation and data stream clustering. Section III introduces the TSSRC method and analyzes the proposed approach. In Section IV, we evaluate the TSSRC method through extensive experiments using real and synthetic data streams. Finally, we draw together the conclusions from this study in Section V.

## II. Related work

### A. Sparse Representation Theory

Sparse representation is an extremely successful technique for representing high-dimensional data [33], [34]. Let the dictionary $\mathbf{D} = [\mathbf{d}_1, \mathbf{d}_2, ..., \mathbf{d}_n] \in \mathbb{R}^{d\times n}$ be a set of $n$ vectors, where each column of the set is of length $d$. Given a vector $\mathbf{x} \in \mathbb{R}^d$, its sparsest representation over the dictionary $\mathbf{D}$ can be approximately represented by a

sparse linear combination of only a few columns of $\mathbf{D}$. Specifically, a sparse representation vector $\mathbf{z} \in \mathbb{R}^n$ of $\mathbf{x}$ can be calculated via the following $l_0$-norm minimization optimization problem:

$$\min_{\mathbf{z}} \|\mathbf{x} - \mathbf{D}\mathbf{z}\|_2 + \lambda \|\mathbf{z}\|_0 \tag{1}$$

where $\lambda > 0$ is used as a trade-off parameter, and $\|\cdot\|_0$ counts the number of nonzero entries in a vector.

Dictionary learning is particularly useful for sparse representation. Considering a vector $\mathbf{x}_i$, several existing online or batch dictionary learning methods are usually based on minimizing the following objective function [30]–[32]:

$$\min_{\mathbf{D},\mathbf{z}_i} \sum_{i=1}^{n} \left( \frac{1}{2} \|\mathbf{x}_i - \mathbf{D}\mathbf{z}_i\|_2^2 + \lambda \|\mathbf{z}_i\|_0 \right) \tag{2}$$

where $\mathbf{z}_i \in \mathbb{R}^n \, (1 \leq i \leq n)$ represents a sparse coefficient vector. Iteratively updating each column of the dictionary for each data object incurs a heavy computational cost. Moreover, these dictionary learning methods make use of each data object to incrementally update the learned dictionary without any forgetting mechanisms. Therefore, these disadvantages make them unsuitable for the real-time clustering of data streams.

### B. Data Stream Clustering Techniques

The conventional data stream clustering algorithms often fall into four categories: hierarchy-based clustering [10], [16], [35], density-based clustering [12], [14], [18], [20], [25], [36], [37], partitioning-based clustering [21], [38], [39], and model-based clustering [11], [15], [19]. BIRCH constructs a height-balanced tree using the CF vector [16], where $CF = (N, LS, SS)$ is composed of $N$ (the number of data objects), $LS$ (the linear sum of the data objects), and $SS$ (the sum of squared data objects). The Euclidean distances between new objects and each centroid of the CF entries in BIRCH are calculated and compared with a threshold to determine whether the new object belongs to a new class. The fully online clustering algorithm consists of two separate phases for clustering evolving data streams into arbitrary shaped clusters [20]. The first stage uses a graph structure to represent microclusters, and the second stage merges these microclusters into macroclusters. ACSC identifies clusters as groups of micro-clusters by introducing artificial ants, which sort clusters by probabilistically picking and dropping microclusters [12]. The fuzzy incremental density-based clustering algorithm clusters evolving data streams using a two-phase scheme [18]. This scheme first produces microclusters using fuzzy local clustering, and then determines the final clusters by estimating the valley threshold from each density peak. These algorithms first consider the CF vector or its variants as data structures in order to create microclusters. Then, they perform the merge operation of microclusters to obtain macroclusters using respective criteria. Essentially, these algorithms share the principle of the local similarity measure based on the Euclidean distance for clusters.

The collection of high-dimensional data from multiple classes often lies in a union of low-dimensional subspaces [40]. Bahri et al. provided a survey on dimensionality reduction approaches for evolving data streams [41]. This survey claimed that stream data mining algorithms usually provide good accuracy for evolving data streams when combined with data-dependent dimensionality reduction techniques. Moreover, these techniques are naturally adapted to the evolving environment of data streams. In addition, EDSSC introduces a subspace structure evolution detection model to detect the appearing, disappearing, and recurring subspaces [22]. In particular, it uses a singular-based Laplacian matrix decomposition to automatically estimate the number of subspaces. However, the accurate estimation of the number of clusters for data objects in the landmark window remains an open problem. Therefore, there is a need for further research to develop data stream clustering algorithms that take into account the subspace structures of high-dimensional data.

## III. Two-stage sparse representation clustering

In this section, we describe the proposed TSSRC method. Sparse representation learning and a discriminative dictionary initialization are achieved through a two-stage data stream clustering structure.

### A. Problem Formulation

A data stream $\mathcal{S}$ is a massive sequence of data objects that arrive continuously over time, i.e., $\mathbf{X} = \{\mathbf{x}_1, \mathbf{x}_2, .., \mathbf{x}_N\} \, (N \to \infty)$, where each data object $x_i \in \mathbb{R}^d$ $(1 \leq i \leq N)$ is associated with a particular time stamp $i$. Because data streams are potentially infinite, we consider a landmark window model, in which data objects are separated by landmarks. When a new landmark is reached, all data objects previously kept in the window are discarded, and new data objects starting from the current landmark are then kept in the window until the next landmark is reached. Without loss of generality, let $\mathbf{X} = [\mathbf{x}_1, \mathbf{x}_2, ..., \mathbf{x}_n] \in \mathbb{R}^{d \times n}$ be a representative example of the data objects in a landmark window. Data stream clustering aims to divide these data objects into their corresponding groups to discover the intrinsic structure of each landmark window, where the number of clusters in the landmark window is unknown.

### B. Sparse Representation-Based Dictionary Learning for Relationship Discovery

TSSRC makes full use of sparse representation to measure their relationship. For example, given a set of data objects $\mathbf{X} = [\mathbf{x}_1, \mathbf{x}_2, ..., \mathbf{x}_n]$ in a landmark window, we represent each data object as a linear combination of the others, where the matrix coefficients should be sparse. The coefficients of the sparse representation of individual data objects are adopted to measure the relationship among the data objects. Hence, designing an affinity matrix is

the key step towards evaluating the relationship among data objects.

Considering a data object $\mathbf{x}_i$ $(1 \leq i \leq n)$, TSSRC uses the following $l_0$-norm minimization for the sparse representation:

$$\underset{\mathbf{z}_i}{arg\min} \|\mathbf{z_i}\|_0 \qquad s.t. \qquad \|\mathbf{x}_i - \mathbf{X}\mathbf{z}_i\|_l \leq \varepsilon \tag{3}$$

where $\|\cdot\|_l$ characterizes the noise term $\varepsilon$. The columns of matrix $\mathbf{Z} \in \mathbb{R}^{n\times n}$ consist of the sparse coefficients, i.e., $\mathbf{Z} = [\mathbf{z}_1, \mathbf{z}_2, ..., \mathbf{z}_n]$. Further, $\mathbf{Z}^*$ is used to define an affinity matrix, i.e., $\mathbf{Z}^* = |\mathbf{Z}| + \left|\mathbf{Z}^T\right|$, where the value of element $\mathbf{z}_{ij}$ in the affinity matrix represents the similarity between the data objects $\mathbf{x}_i$ and $\mathbf{x}_j$. The corresponding value of the sparse representation between each pair of data objects indicates their similarity, and the number of nonzero sparse representation coefficients indicates the potential relationship among the corresponding data objects. If any two data objects $\mathbf{x}_i$ and $\mathbf{x}_j$ are close in the intrinsic geometry of the data distribution, then the representations of these two data objects, namely, $\mathbf{z}_i$ and $\mathbf{z}_j$ with respect to the same basis $\mathbf{X}$, are close to each other [42]. For a larger degree of similarity between $\mathbf{x}_i$ and $\mathbf{x}_j$, the distance between $\mathbf{z}_i$ and $\mathbf{z}_j$ should be smaller. Hence, it is reasonable that sparse representation is adopted to measure the relationship among data objects in the data stream.

Clustering data streams requires a process that can continuously cluster objects within some time restrictions. However, determining the sparsest representation of each data object individually in (3) leads to a high computational cost. To accelerate the optimization, we apply a batch mode to obtain sparse solutions by solving a sparse representation optimization problem. Each element $z_{ij}$ of the matrix $\mathbf{Z}$ represents the similarity between data objects $i$ and $j$. To maintain the interpretability of the sparse representation results, the diagonal elements of $\mathbf{Z}$ must be zero. This indicates that there is no relationship between any data object and itself. In addition, this avoids the trivial solution of $\mathbf{Z}$ being the identity matrix from the optimization perspective. Given an initial dictionary $\mathbf{D}$, we consider the following convex optimization problem to seek a sparse representation $\mathbf{Z}$:

$$\min_{\mathbf{Z}} \|\mathbf{Z}\|_0 + \frac{\alpha}{2}\|\mathbf{X} - \mathbf{D}\mathbf{Z}\|_F^2 \quad s.t. \quad diag\,(\mathbf{Z}) = \mathbf{0} \tag{4}$$

where $\alpha$ is a scalar constant and $diag\,(\mathbf{Z})$ represents the vector containing the $n$ diagonal elements of $\mathbf{Z}$. In particular, $\mathbf{X}$ can be represented as the linear combination:

$$\mathbf{X} = \mathbf{D}\mathbf{Z} + \mathbf{E}. \tag{5}$$

This indicates that $\mathbf{D}$ is a specific dictionary that linearly spans the data space. The regularization $\|\mathbf{D}\|_F^2$ is incorporated to prevent $\mathbf{D}$ from being arbitrarily large when $\mathbf{D}$ is iteratively updated during optimization. Assuming that $\mathbf{Z}$ is sparse, the preliminary objective function is defined as:

$$\begin{aligned} &\min_{\mathbf{Z},\mathbf{D}} \|\mathbf{Z}\|_0 + \frac{\alpha}{2}\|\mathbf{X} - \mathbf{D}\mathbf{Z}\|_F^2 + \frac{\beta}{2}\|\mathbf{D}\|_F^2 \\ &s.t. \quad diag\,(\mathbf{Z}) = \mathbf{0} \end{aligned} \tag{6}$$

where $\beta > 0$ is a regularization parameter and $\mathbf{D}$ is the dictionary to be learnt.

## C. Optimization

The coefficients of sparse representation and the dictionary are iteratively updated by solving (6). We first convert this problem to the following equivalent problem by introducing an auxiliary variable $\mathbf{J}$:

$$\begin{aligned} &\min_{\mathbf{Z},\mathbf{D},\mathbf{J}} \|\mathbf{Z}\|_0 + \frac{\alpha}{2}\|\mathbf{X} - \mathbf{D}\mathbf{J}\|_F^2 + \frac{\beta}{2}\|\mathbf{D}\|_F^2 \\ &s.t. \quad \mathbf{J} = \mathbf{Z} - d\,(\mathbf{Z}) \end{aligned} \tag{7}$$

where $d\,(\mathbf{Z}) \in \mathbb{R}^{n\times n}$ represents a diagonal matrix whose diagonal entries correspond to that of $\mathbf{Z}$. The augmented Lagrangian function of (7) is:

$$\begin{aligned} &\min_{\mathbf{Z},\mathbf{D},\mathbf{J},\mathbf{Y}} \|\mathbf{Z}\|_0 + \frac{\alpha}{2}\|\mathbf{X} - \mathbf{D}\mathbf{J}\|_F^2 + \frac{\beta}{2}\|\mathbf{D}\|_F^2 + \\ &tr\left(\mathbf{Y^T}\left(\mathbf{J} - \mathbf{Z} + d\,(\mathbf{Z})\right)\right) + \frac{\mu}{2}\|\mathbf{J} - \mathbf{Z} + d\,(\mathbf{Z})\|_F^2 \end{aligned} \tag{8}$$

where $\mathbf{Y} \in \mathbb{R}^{n\times n}$ is a Lagrange multiplier and $\mu > 0$ is a penalty parameter. The above optimization problem can be formulated as follows:

$$\begin{aligned} &\min_{\mathbf{Z},\mathbf{D},\mathbf{J},\mathbf{Y}} \|\mathbf{Z}\|_0 + \frac{\alpha}{2}\|\mathbf{X} - \mathbf{D}\mathbf{J}\|_F^2 + \frac{\beta}{2}\|\mathbf{D}\|_F^2 \\ &+ \frac{\mu}{2}\left\|\mathbf{Z} - d\,(\mathbf{Z}) - \left(\mathbf{J} + \frac{\mathbf{Y}}{\mu}\right)\right\|_F^2. \end{aligned} \tag{9}$$

**Algorithm 1** Solving (6) using an inexact ALM framework

Input: data matrices $\mathbf{X} \in \mathbb{R}^{d\times n}$ and $\mathbf{D} \in \mathbb{R}^{d\times n}$, parameters $\alpha > 0$, $\beta > 0$, $\mu > 0$ and $\rho > 1$.

Initialize: $k = 1$, $\mathbf{Z}_1 = \mathbf{J}_1 = \mathbf{Y}_1 = \mathbf{0}$, $\mathbf{D}_1 = \mathbf{D}$, $\mu_1 = \mu$, $\mu_{\max} = 10^6$ and $\varepsilon = 10^{-2}$.

```
while not converged do
    update J_{k+1} using (10);
    update Z_{k+1} using (11);
    update D_{k+1} using (17);
    update the multiplier Y_{k+1} using (18);
    update the parameter μ: μ_{k+1} = min(ρμ_k, μ_max);
    check the convergence condition:
        ||Z_{k+1} − J_{k+1}||_max < ε;
    k ← k + 1;
end while
```

Output: $\mathbf{Z}_k, \mathbf{D}_k$

Equation (9) can be effectively solved using the inexact augmented Lagrange multipliers (ALM) framework [43]. The variables $\mathbf{J}$, $\mathbf{Z}$ and $\mathbf{D}$ can be updated alternately while the other two variables are fixed. The variable $\mathbf{J}$ has a closed-form solution, and the update scheme for $\mathbf{J}_{k+1}$ is:

$$\begin{aligned} &\mathbf{J}_{k+1} = \left(\alpha \cdot \mathbf{D}_k^T\mathbf{D}_k + \mu_k \cdot \mathbf{I}\right)^{-1}\left(\alpha \cdot \mathbf{D}_k^T\mathbf{X} + \mu_k \cdot \mathbf{Z}_k - \mathbf{Y}_k\right), \\ &\mathbf{J}_{k+1} \leftarrow normalize_{(0,1]}\left(\mathbf{J}_{k+1}\right) \end{aligned} \tag{10}$$

where $normalize_{(0,1]}(\cdot)$ represents a matrix with columns normalized to a length of one. Given the fixed $\mathbf{J}_{k+1}$, $\mathbf{Z}_{k+1}$ is updated by the following scheme:

$$\begin{aligned}&\mathbf{Z}_{k+1}=\min_{\mathbf{Z}_{k+1}}\frac{1}{\mu_k}\|\mathbf{Z}_{k+1}\|_0+\frac{1}{2}\left\|\mathbf{Z}_{k+1}-\left(\mathbf{J}_{k+1}+\frac{\mathbf{Y}_k}{\mu_k}\right)\right\|_F^2,\\&\mathbf{Z}_{k+1}\leftarrow\mathbf{Z}_{k+1}-d\left(\mathbf{Z}_{k+1}\right).\end{aligned}\tag{11}$$

The closed-form solution of the first part of (11) is obtained using the hard thresholding operator $\mathcal{H}$:

$$\mathbf{Z}_{k+1}=\mathcal{H}_{\sqrt{\frac{1}{\mu_k}}}\left(\mathbf{J}_{k+1}+\frac{\mathbf{Y}_k}{\mu_k}\right)\tag{12}$$

where the operator $\mathcal{H}_{\sqrt{\lambda}}(x)$ is defined as follows [44]:

$$\mathcal{H}_{\sqrt{\lambda}}(x)=\begin{cases}0, & if \quad |x|\le\sqrt{\lambda}\\ x, & if \quad |x|>\sqrt{\lambda}\end{cases}.\tag{13}$$

When $\mathbf{Z}$ is fixed, Problem (6) is equivalent to the following problem:

$$\min_{\mathbf{D}}\frac{\alpha}{2}\|\mathbf{X}-\mathbf{D}\mathbf{Z}\|_F^2+\frac{\beta}{2}\|\mathbf{D}\|_F^2.\tag{14}$$

For a fixed $\mathbf{Z}_{k+1}$ in the $(k+1)$-th iteration, $\mathbf{D}_{k+1}$ is updated by minimizing the following surrogate function:

$$\begin{aligned}\phi\left(\mathbf{D}_k\right)=&tr\left(\mathbf{D}_k^T\mathbf{D}_k\mathbf{Z}_{k+1}\mathbf{Z}_{k+1}^T-2\mathbf{D}_k^T\mathbf{X}_s\mathbf{Z}_{k+1}^T\right)\\&+\frac{\beta}{\alpha}tr\left(\mathbf{D}_k^T\mathbf{D}_k\right).\end{aligned}\tag{15}$$

We employ a classical block coordinate descent algorithm that is made up of the gradient computation and the Euclidean projection on the $l_2$-norm to minimize $\phi\left(\mathbf{D}_k\right)$ on the columns of $\mathbf{D}$ [32], [45]. The gradient of $\phi\left(\mathbf{D}_k\right)$ can be computed as:

$$\frac{\partial\phi\left(\mathbf{D}_k\right)}{\partial\mathbf{D}_k}=\mathbf{D}_k\left(\mathbf{Z}_k\mathbf{Z}_k^T+\frac{\beta}{\alpha}\cdot\mathbf{I}\right)-\mathbf{X}_s\mathbf{Z}_k^T.\tag{16}$$

The updating scheme for $\mathbf{D}_{k+1}$ is:

$$\mathbf{D}_{k+1}=\mathcal{P}\left(\mathbf{D}_k-\frac{1}{\delta}\cdot\frac{\partial\phi\left(\mathbf{D}_k\right)}{\partial\mathbf{D}_k}\right)\tag{17}$$

where $\frac{1}{\delta}$ is the gradient step and $\mathcal{P}$ represents the Euclidean projection on the $l_2$-norm. In practice, we usually set $\delta=\mathbf{A}[j,j]$ while calculating the $j$-th column of $\mathbf{D}_{k+1}$, where $\mathbf{A}=\mathbf{Z}_{k+1}\mathbf{Z}_{k+1}^T+\frac{\beta}{\alpha}\cdot\mathbf{I}$ [32]. For fixed $\mathbf{J}_{k+1}$ and $\mathbf{Z}_{k+1}$, the Lagrange multiplier $\mathbf{Y}_{k+1}$ is updated using the step size $\mu$ as:

$$\mathbf{Y}_{k+1}=\mathbf{Y}_k+\mu_k\left(\mathbf{Z}_{k+1}-\mathbf{J}_{k+1}\right).\tag{18}$$

The convergence condition is satisfied when $\|\mathbf{Z}_k-\mathbf{J}_k\|_{\max}<\varepsilon$ holds, where $\|\cdot\|_{\max}$ represents the maximum absolute value among all elements in a matrix. In practice, we usually set $\varepsilon=10^{-2}$. The complete procedure for solving (6) is outlined in Algorithm 1.

### D. Proposed Dictionary Initialization Strategy

Choosing an appropriate dictionary in (6) leads to effective algorithms for the evaluation of the sparse representation. An intuitive approach is to consider the data objects $\mathbf{X}$ as a dictionary; this is often adopted in traditional clustering algorithms based on sparse representation. However, various levels of noise are ubiquitous in data streams, and this may result in unsatisfactory results. Moreover, data streams have particular characteristics, e.g., possibly infinite volume and dynamical changes, that differentiate them from traditional static datasets. Data objects in the current window must be processed before the next landmark window of stream data arrives. There is a need for a dictionary initialization strategy for sparse representations that adaptively passes previously learned knowledge to the current data stream over time. Consequently, we pursue a more appropriate dictionary rather than using the data objects themselves as the dictionary. This reveals the true relationship among the data objects in the current landmark window.

To design a discriminative dictionary, the key step is to preserve valuable data objects for a given period of time, and replace them with a number of new valuable data objects later. Assume that we have representative data objects $\mathbf{X}_s\in\mathbb{R}^{d\times m}$ that were previously learned from data streams before time $t$, where $c$ represents the number of clusters, $n_c=\lfloor n/c\rfloor$, and $m=c\times n_c$. When $t=1$, the problem of data stream clustering is considered to be a traditional clustering problem, i.e., $\mathbf{D}=\mathbf{X}_1$. Then, $\mathbf{X}_1$ is considered to be $\mathbf{X}_s$ for the current landmark window. When $\mathbf{X}_t$ arrives at time $t>1$, $\mathbf{D}=[\mathbf{X}_s,\mathbf{X}_t]\in\mathbb{R}^{d\times(m+n)}$ and $\mathbf{X}=[\mathbf{X}_s,\mathbf{X}_t]$ are employed to represent a set of data objects and its dictionary, respectively, for (6). The sparse representation result $\mathbf{Z}$ and the corresponding dictionary $\mathbf{D}$ can be obtained by Algorithm 1.

We further elaborate the implications of the above sparse representation $\mathbf{Z}$ and the corresponding dictionary $\mathbf{D}$ for two critical issues in data stream clustering. First, the learned affinity matrix is constructed using $\mathbf{Z}$, i.e., $\mathbf{Z}^*=|\mathbf{Z}|+\left|\mathbf{Z}^T\right|$, which represents the final similarity of data objects in the current landmark window. TSSRC considers $\mathbf{Z}^*$ as an affinity for spectral clustering algorithms, e.g., NCuts [46]. Consequently, we obtain the final clustering results for $\mathbf{X}$, which contain the clusters of $\mathbf{X}_t$ in the current window. Moreover, $\mathbf{Z}$ and $\mathbf{D}$ are crucial for the exact construction of a new $\mathbf{X}_s$ for the current landmark window. The $c$ submatrices are extracted from the sparse representation $\mathbf{Z}$, i.e., $\mathbf{Z}_1,\mathbf{Z}_2,...,\mathbf{Z}_c$. Considering a data object $\mathbf{x}_i$ from cluster $j$, we can easily choose its sparse representation $\mathbf{Z}_j$ from $\mathbf{Z}$. Each coefficient matrix $\mathbf{Z}_j\in\mathbb{R}^{n_j\times n_j}$ $(j=1,2,...,c)$ is a block matrix associated with the $j$-th cluster, where $\sum_{j=1}^{c}n_j=m+n$. The dictionary $\mathbf{D}$ for all $c$ clusters can be considered as $\mathbf{D}=[\mathbf{D}_1,\mathbf{D}_2,...,\mathbf{D}_c]$, where each $\mathbf{D}_j$ is a sub-dictionary containing $n_j$ elements. Ideally, the nonzero entries in $\mathbf{Z}_j$ will all be associated with the columns of $\mathbf{D}_j$. In other words, $\mathbf{x}_i$ should be

Algorithm 2 TSSRC algorithm

Input: data matrices $\mathbf{X}_t = [\mathbf{x}_1, \mathbf{x}_2, ..., \mathbf{x}_n] \in \mathbb{R}^{d\times n}$ and $\mathbf{X}_s \in \mathbb{R}^{d\times m}$, parameters $\alpha > 0$, $\beta > 0$, number of clusters $c$.

```
if t == 1 then
    D = X_1, X = X_1 and X_s = X_1;
else
    D = [X_s, X_t] and X = [X_s, X_t];
end if
Solve Problem (6) using Algorithm 1, and obtain the optimal solution (Z, D).
Compute the affinity matrix Z* = |Z| + |Z^T|;
Apply Z* in the NCuts algorithm, and obtain c clusters containing the clusters of X_t;
if t > 1 then
    n_c = ⌊n/c⌋;
    /* Add at most n_c data objects chosen from each cluster into X_s as the representative data objects */
    for each cluster C_j do
        /* n_j represents the number of data objects in C_j */
        if n_j > n_c then
            for each element d_k in D_j do
                Compute AR(d_k) using (19);
            end for
            Add n_c data objects into X_s corresponding to the largest n_c approximate residuals;
        else
            Add all n_j data objects into X_s
        end if
    end for
end if
```

Output: clusters of $\mathbf{X}_t$ and the new $\mathbf{X}_s$.

represented as a linear combination of the elements from $\mathbf{D}_j$, where $\mathbf{Z}_j$ denotes the vector of sparse representation coefficients associated with the $j$-th cluster. However, corruptions and insufficient modeling accuracy may lead to small nonzero entries associated with a few data objects from other clusters. Using only the coefficients associated with the corresponding cluster $j$, we can approximate the data objects $\mathbf{X}_j$ as $\widetilde{\mathbf{X}_j} = \mathbf{D}_j\mathbf{Z}_j$. The definition of the approximate residual (AR) of each element of $\mathbf{D}$, i.e., $\mathbf{d}_k$, is given as follows:

$$AR(\mathbf{d}_k) = \left\| normalize_{(0,1]}\left(\widetilde{\mathbf{X}_j}\right) - \sum_{i=1, i\neq k}^{n_j} \mathbf{d}_i \cdot \left(\mathbf{Z}_j^i\right)^T \right\|_F^2, \quad k = 1, 2, ..., n_j, \qquad j = 1, 2, ..., c \tag{19}$$

where $\mathbf{Z}_j^i$ represents the $i$-th column of $\mathbf{Z}_j$. $AR(\mathbf{d}_k)$ denotes the error of all dictionary elements of the $j$-th cluster when $\mathbf{d}_k$ is removed. We employ $AR(\mathbf{d}_k)$ to measure the degree of importance of $\mathbf{d}_k$ in the sparse representation. A higher value of $AR(\mathbf{d}_k)$ indicates that $\mathbf{d}_k$ is more important. The representative data objects of each cluster $\mathbf{C}_j$ are then determined as those with the largest $m_j$ residuals. Algorithm 2 summarizes the complete data stream clustering procedure.

In practice, there are often several data objects that belong to more than one cluster. These may be considered as outliers in traditional clustering problems, and are usually removed before clustering to prevent poor clustering performance because of insufficient representative data objects of the dictionary in the sparse representation. However, most such objects may be valid in a data stream. TSSRC overcomes a fundamental limitation of insufficient representative data objects in traditional sparse representation algorithms, because $\mathbf{X}_s$ contains representative data objects from all clusters over time.

## E. Theoretical Analysis

1) Convergence Analysis: In this section, we estimate the convergence condition in Algorithm 1. Algorithm 1 has a single convergence condition, i.e., $\|\mathbf{Z}_{k+1} - \mathbf{J}_{k+1}\|_{\max} < \varepsilon$. Theorem 1 shows that the convergence condition of Algorithm 1 is satisfied under certain conditions. The proof of Theorem 1 can be found in the supplementary material.

Theorem 1 The convergence condition $\|\mathbf{Z}_{k+1} - \mathbf{J}_{k+1}\|_{\max} < \varepsilon$ will eventually be satisfied as $k$ increases if $\rho$ and $\mu$ satisfy the following conditions:

$$\rho > 2 \quad and \quad \mu > 0$$

where $k$ represents the number of iterations and $\varepsilon$ is a small positive number, e.g., $\varepsilon = 10^{-4}$.

2) Sparse Stability Analysis: Sparse representation with respect to a learned dictionary of base elements is of great importance in identifying and modeling the intrinsic structures of data streams. With the appropriate regularization parameters $\alpha$ and $\beta$ in (6), we usually strike a balance between the sparse approximation error of $\mathbf{X}$ and the sparsity of $\mathbf{Z}$ in a single landmark window. However, large amounts of data objects are being continuously generated by an unknown, stationary probability distribution. Maintaining the sparse stability of $\mathbf{Z}$ with fixed $\alpha$ and $\beta$ in TSSRC is crucial for data stream clustering as data objects arrive in successive landmark windows.

To keep the sparsity of $\mathbf{Z}$ stable, we investigate the sparse stability condition in a single landmark window. Theorem 2 shows that the sparsity ratio of $\mathbf{Z}$ always remains stable after several iterative computations satisfying a certain condition in Algorithm 1. This provides a theoretical guarantee for the stability of the sparsity ratio in successive landmark windows. The proof of Theorem 2 is given in the supplementary material.

Theorem 2 Suppose that convergence is achieved after the $k$-th iteration in Algorithm 1. The sparsity ratio (SR) of a matrix $Z$ is defined as $SR(\mathbf{Z}_k) = \frac{\|\mathbf{Z}_k\|_0}{num(\mathbf{Z}_k)}$, where num($\mathbf{Z}_k$) is the number of elements in $\mathbf{Z}_k$. The SR of $\mathbf{Z}$

will always remain stable, i.e., $|SR(\mathbf{Z}_{k+1}) - SR(\mathbf{Z}_k)| < \varepsilon$, after $k$ iterative computations, if

$$\mu_{k-1} > 1 \quad and \quad \rho > 1$$

where $\|\mathbf{Z}_k\|_0$ counts the number of nonzero entries in the matrix $\mathbf{Z}_k$, $\varepsilon = 1e^{-6}$ and $k > 1$.

The sparsity condition in Theorem 2 is slightly stricter than the convergence condition in Theorem 1. Note that $\mu$ in Theorem 1 represents the initial value at the beginning of the iterative computations while $\mu_{k-1}$ in Theorem 2 is the result of $(k-1)$ iterative computations using $\mu_k = \rho\mu_{k-1}$. Consequently, the SR of $\mathbf{Z}$ must reach a stable state before convergence is achieved with appropriate values of $\rho$ and $\mu$. This demonstrates that the proposed method offers a theoretical guarantee of the feasibility of the sparse representation.

3) Parameter Analysis: The convergence and sparsity of Algorithm 1 are theoretically guaranteed by Theorems 1 and 2. In particular, the combination of $\mu$ and $\rho$ effectively controls the convergence speed and sparsity ratio in Algorithm 1. In theory, a larger value of $\rho$ implies faster convergence. The desired sparsity can be obtained if the initial value of $\mu$ is relatively small (e.g., $\mu < 1$). In addition, estimating the number of clusters $c$ is one of the main challenges in data clustering tasks. The the normalized Laplacian matrix $\mathbf{L}$ can be constructed by $\mathbf{Z}$. While $\mathbf{Z}$ is a strictly block-diagonal affinity matrix, the number of clusters $c$ can be obtained by counting the number of zero singular values of $\mathbf{L}$ [47]. However, the various noise and corruption in the data stream means that $\mathbf{Z}$ is only nearly a block-diagonal matrix. Assume that the maximum number of clusters in a data stream is $C_{max}$. There are probably $C_{max}$ clusters in $\mathbf{X}_s$ over time. We introduce two schemes to automatically determine the number of clusters $c$. In the first scheme, we can prepare $\mathbf{X}_s$ for data stream clustering tasks if some prior knowledge of the data stream is available, where $\mathbf{X}_s$ consists of fully representative data objects for all clusters. This is an intuitive and effective scheme, and $c$ is always the maximum number of clusters, i.e., $c = C_{max}$. The second scheme is based on a theoretical analysis of the spectral properties of block-diagonal affinity matrices. The value of $c$ is determined such that the gap between each two consecutive eigenvalues of $\mathbf{L}$ from $\lambda_1$ to $\lambda_p$ is very small, but the gap between $\lambda_p$ and $\lambda_{p+1}$ is relatively large, where $p$ is a positive integer [48]. This heuristic scheme does not take advantage of any prior knowledge of the data stream. We will set $c = C_{max}$ if $p \geq C_{max}$. Simultaneously, this means that there are fully representative data objects for all clusters in $\mathbf{X}_s$. The scheme will be abandoned once $c = C_{max}$ in successive landmark windows. It is obvious that TSSRC will benefit from knowing an accurate number of clusters.

## F. Computational Complexity

We assume that a landmark window $\mathbf{X}$ has $n$ data objects belonging to $c$ clusters, where the size of $\mathbf{X}$ is $d \times n$. Algorithm 2 summarizes the complete data stream clustering algorithm of TSSRC. We use an inexact ALM framework in Algorithm 1 [43]. The computational complexity of the first step in Algorithm 1 is $\mathcal{O}(n^3)$ because it requires the inverse of an $n \times n$ matrix to be computed. The second and third steps in Algorithm 1 have a computational complexity of $\mathcal{O}(n^2)$ and $\mathcal{O}(d \times n^2)$, respectively, as they involve matrix multiplication. The overall computational complexity of Algorithm 1 is $\mathcal{O}(tn^3)$ if $d < n$, where $t$ is the number of iterations. The spectral clustering and online dictionary learning steps in Algorithm 2 have a computational complexity of $\mathcal{O}(n^3)$ and $\mathcal{O}(n^2)$, respectively. Therefore, the final overall complexity of TSSRC is $\mathcal{O}(tn^3)$ if $d < n$.

## G. Comparison with Online Dictionary Learning Schemes

A number of online dictionary learning methods have been developed and combined with sparse representation models for signal reconstruction and classification [30]–[32]. For an online data object, these methods typically perform dictionary learning in two stages: first, the sparse coefficients are calculated with an updated dictionary by solving an $l_1$-regularized linear least-squares problem; second, each column of the dictionary is sequentially updated with the sparse coefficients while the other columns are held fixed under the particular constraint [31], [32]. The two stages of online dictionary learning methods involve iterative computations, which may not satisfy the real-time requirements of clustering tasks for data streams. Moreover, these methods often use random variables to initialize the dictionary, where the number of columns in the impact dictionary is different from that of the data objects. As a result, the sparse coefficients do not contain any meanings that can measure the relationship among data objects. Additionally, online dictionary learning methods take advantage of complete history information to incrementally update the dictionary by processing one element at a time. However, there is no need to preserve the whole meaning of all original data objects in data stream clustering, as the distribution of data objects in data streams may vary over time. Conversely, plausible forgetting mechanisms should ideally be employed to consider cluster evolution.

TSSRC learns a dictionary in a different way. The tasks of finding the sparse coefficients and learning the dictionary of a group of data object are integrated into a single stage in Algorithm 1. In particular, a forgetting mechanism is implemented using the dictionary initialization strategy. Specifically, all data objects are adopted as an initialized dictionary in the first window. Then, the initialized dictionary is changed across dynamic data streams by updating the set of representative data objects. This satisfies the requirement of obsolescence for previous data objects in data stream clustering. Hence, TSSRC provides sparse representation model-based dictionary learning for relationship discovery, which works radically differently from previous online dictionary learning schemes.

TABLE I
Description of streaming datasets.

| Dataset | Classes | Data objects | Features | Type |
|---|---|---|---|---|
| UG2C5D | 2 | 200,000 | 5 | Synthetic |
| Network Intrusion | 2 | 494,000 | 42 | Real |
| Keystroke | 4 | 1,600 | 10 | Real |
| Forest Cover | 7 | 580,000 | 54 | Real |
| USPS | 10 | 7,291 | 256 | Real |
| COIL-100 | 100 | 7,200 | 1,024 | Real |

## IV. Experimental Study

In this section, we evaluate the performance of the proposed TSSRC[1] on publicly available datasets by comparing it against existing popular data stream clustering algorithms, namely, CluStream [21], ClusTree [49], CluStreamKM [50], StreamKM++ [39], multiview stream (MVStream) clustering [13], ESA-Stream [25], RRCSL [27], and FDCM_SSR [29]. TSSRC is implemented in MATLAB and all experiments are conducted on a Windows platform with an Intel i5-2300 CPU and 16 GB RAM. The implementation of the first four algorithms is provided by Massive Online Analysis (MOA), which is a popular open source tool for data stream mining [50]. The source codes of the latter four algorithms are provided by their authors or implemented according to their respective theories.

### A. Experimental Settings

1) Assessment Criteria: The TSSRC algorithm is evaluated in terms of its clustering quality and time efficiency. The quality of clustering is measured by five metrics, namely, the clustering purity (Purity), normalized mutual information (NMI), $F$-measure, within-cluster sum of square error (SSE), and silhouette coefficient (SC). The first three metrics and the latter two metrics are commonly used as external and internal measures for clustering evaluation, respectively. The five metrics follow their standard definitions [2], [25], [51]. A higher value indicates better clustering performance for the metrics except SSE.

2) Datasets: We consider one synthetic and five real streaming datasets, all of which are publicly available [52], [53]. Table I summarizes the details of the six datasets. The UG2C5D and Keystroke datasets have drift intervals of 2,000 and 200, respectively. The drift intervals of the Network Intrusion, United States Postal Service (USPS), and COIL-100 datasets are unknown. For the sake of simplicity, their drift intervals are set to 1,000 in the experiments. We selected the first 50 landmark windows of the UG2C5D, Network Intrusion, Forest Cover datasets and all landmark windows of the other datasets for clustering evaluation.

3) Parameter Settings: All parameters of the benchmark algorithms were manually tuned using MOA to ensure good results. Algorithm 2 has three main parameters: $\alpha$, $\beta$ and $c$. Empirically speaking, $\lambda$ should be relatively large if the data are slightly contaminated by noise, and vice versa. The parameters $\alpha$, $\beta$ were chosen from $\left\{1e^{-3}, 5e^{-3}, 1e^{-2}, 5e^{-2}, 0.1, 0.5, 1\right\}$ with the grid search strategy. For parameter $c$, we employed the second scheme mentioned in Section III-E3. In addition, there are two additional parameters involved in the optimization procedure of TSSRC, i.e., $\mu$ and $\rho$. According to Theorems 1 and 2, we set $\mu$ and $\rho$ to 0.05 and 8, respectively. Further details of the parameters are given in each experiment.

[1]https://codeocean.com/capsule/5525574/tree/v2

### B. Clustering Quality Evaluation

We evaluate the performance of the proposed algorithm with streaming datasets. The results are presented in Table II. For the UG2C5D and Network Intrusion datasets, a large number of landmark windows only contain one cluster. Therefore, the NMIs and SCs of these two datasets are omitted from Table II. For this experiment, the six groups of TSSRC parameters were (1) $\alpha = 0.5$, $\beta = 0.01$, (2) $\alpha = 1$, $\beta = 1e^{-2}$, (3) $\alpha = 1e^{-2}$, $\beta = 0.1$, (4) $\alpha = 5e^{-2}$, $\beta = 0.1$, (5) $\alpha = 5e^{-3}$, $\beta = 5e^{-3}$, and (6) $\alpha = 1e^{-3}$, $\beta = 5e^{-2}$.

Table II shows that TSSRC consistently obtained the best results in terms of the five metrics. This confirms that our proposed method is very effective against a varying number of clusters regarding dynamic data stream clustering. For example, TSSRC achieved a high ACC of 92.57% on the UG2C5D dataset, an improvement of at least 2.57% over the other algorithms. We observed the same advantage when more features were contained in the data objects. For example, TSSRC obtained a high ACC of 98.59%, 80.94%, 76.88%, 78.5%, and 54.4% on the other five datasets. Moreover, Table II indicates that TSSRC consistently outperformed the other eight algorithms in terms of the other metrics. For example, using TSSRC, the F-measure for each dataset is increased by at least 2.75%, 5.01%, 6.15%, 3.59%, 17.89%, and 5.08% respectively, compared with the competing methods. The NMI values achieved by TSSRC are 2.13%, 2.4%, 10.4%, and 6.8% higher than the second-best results on the Keystroke, Forest Cover, USPS, and COIL-100 datasets, respectively. Moreover, TSSRC produces excellent results that are much better than the other methods in terms of both SSE and SC. These results confirm that the relationship calculated from the sparse representation significantly improves the clustering performance, especially when the data objects contain more features.

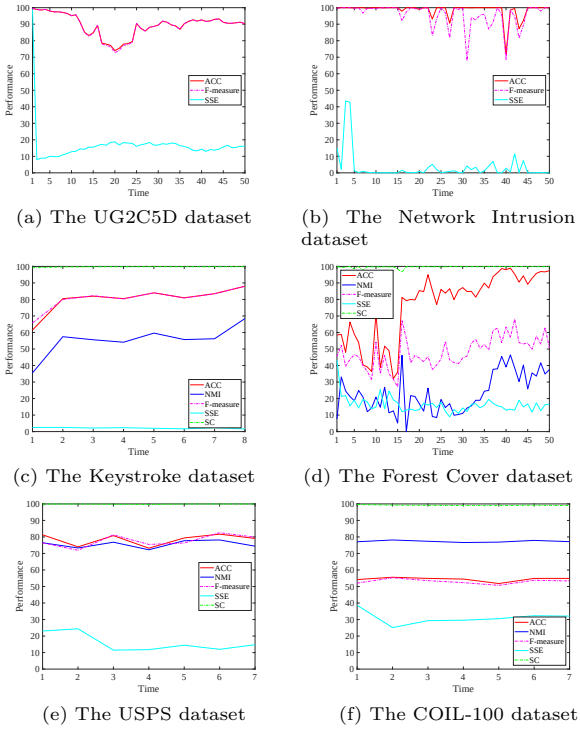
The time efficiency of the proposed algorithm was also evaluated and the results are presented in Table III. From Table III, it can be observed that CluStreamKM was faster than the other algorithms on all datasets except Keystroke. TSSRC obtained an encouraging result with the UG2C5D dataset, where there are 2,000 data objects in a single landmark window. However, the proposed method achieved the best result with the Keystroke dataset, where a single landmark window contains 200 data objects. This means that the size of a single landmark window has a large impact on the computational cost of data stream clustering by the proposed TSSRC. In addition,

TABLE II
Average clustering results over a fixed size of landmark windows on the streaming datasets. Best and second-best results are shown in bold and underlined, respectively.

| Datasets | Metrics | TSSRC | CluStream | ClusTree | CluStreamKM | StreamKM++ | MVStream | ESA-Stream | RRCSL | FDCM_SSR |
|---|---|---|---|---|---|---|---|---|---|---|
| UG2C5D | Purity (%) | 92.57 | 86 | 84 | 89 | 90 | 85.5 | 88.81 | 82.15 | 89.82 |
| | F-measure (%) | 92.56 | 78 | 75 | 81 | 84 | 83.67 | 86.31 | 81.76 | 89.81 |
| | SSE | 166.83 | 231 | 276 | 235 | 217 | 292.1 | 196.31 | 356.07 | 281.54 |
| Network Intrusion | Purity (%) | 98.59 | 93 | 91 | 94 | 92 | 92.56 | 93.67 | 93.1 | 94.33 |
| | F-measure (%) | 95.73 | 72 | 70 | 78 | 75 | 82.68 | 72.78 | 63.42 | 90.72 |
| | SSE | 33.51 | 130 | 133 | 106 | 127 | 113.51 | 92.7 | 179.89 | 113.92 |
| Keystroke | Purity (%) | 80.94 | 64 | 73 | 71 | 69 | 64.75 | 68.98 | 68.81 | 74.75 |
| | F-measure (%) | 81.42 | 74 | 75 | 72 | 70 | 68.76 | 65.83 | 68.57 | 75.27 |
| | NMI (%) | 57.3 | 37 | 49 | 53 | 47 | 38.31 | 42.13 | 47.55 | 55.17 |
| | SSE | 21.37 | 41 | 36 | 39 | 43 | 37.92 | 38.29 | 27.4 | 25.72 |
| | SC | 1.00 | 0.72 | 0.84 | 0.76 | 0.68 | 0.63 | 0.81 | 0.97 | 0.95 |
| Forest Cover | Purity (%) | 76.88 | 61 | 66 | 53 | 56 | 68.5 | 70.05 | 69.25 | 71.6 |
| | F-measure (%) | 48.2 | 37 | 33 | 36 | 38 | 40.32 | 42.85 | 41.7 | 44.61 |
| | NMI (%) | 23.11 | 20 | 16 | 14 | 18 | 16.6 | 17.47 | 17.39 | 20.71 |
| | SSE | 163.38 | 267 | 208 | 243 | 227 | 212.6 | 190.63 | 186.95 | 197.07 |
| | SC | 1.00 | 0.53 | 0.68 | 0.56 | 0.62 | 0.67 | 0.72 | 0.88 | 0.99 |
| USPS | Purity (%) | 78.5 | 53 | 58 | 60 | 62 | 63.37 | 64.04 | 63.1 | 68.5 |
| | F-measure (%) | 77.69 | 51 | 56 | 58 | 59 | 58.86 | 59.65 | 58.81 | 59.8 |
| | NMI (%) | 75.6 | 48 | 54 | 53 | 56 | 55.620 | 60.31 | 51.93 | 65.2 |
| | SSE | 159.77 | 312 | 324 | 293 | 283 | 374.19 | 366.41 | 239.93 | 175.31 |
| | SC | 1.00 | 0.67 | 0.63 | 0.69 | 0.73 | 0.78 | 0.59 | 0.56 | 0.98 |
| COIL-100 | Purity (%) | 54.4 | 41 | 39 | 44 | 43 | 45.4 | 46.86 | 42.4 | 51.1 |
| | F-measure (%) | 53.01 | 40 | 42 | 38 | 39 | 39.3 | 47.86 | 40.5 | 47.93 |
| | NMI (%) | 77.31 | 51 | 58 | 62 | 57 | 47.23 | 44.34 | 69.1 | 70.51 |
| | SSE | 310.83 | 797 | 689 | 636 | 563 | 529.05 | 614.52 | 390.41 | 340.44 |
| | SC | 0.99 | 0.58 | 0.64 | 0.5 | 0.73 | 0.86 | 0.66 | 0.96 | 0.95 |

TABLE III
Average computational cost of a landmark window (S) for each dataset.

| Datasets | TSSRC | CluStream | ClusTree | CluStreamKM | StreamKM++ | MVStream | ESA-Stream | RRCSL | FDCM_SSR |
|---|---|---|---|---|---|---|---|---|---|
| UG2C5D | 6.16 | 0.14 | 1.22 | 0.13 | 0.33 | 9.16 | 9.11 | 43.18 | 28.32 |
| Network Intrusion | 2.05 | 0.47 | 2.7 | 0.24 | 0.44 | 3.68 | 2.77 | 15.36 | 5.24 |
| Keystroke | 0.06 | 0.15 | 0.9 | 0.09 | 0.1 | 0.11 | 0.12 | 0.43 | 0.26 |
| Forest Cover | 1.98 | 0.29 | 0.72 | 0.27 | 0.48 | 4.44 | 2.69 | 13.11 | 3.75 |
| USPS | 2.22 | 0.69 | 1.23 | 0.53 | 0.91 | 3.72 | 3.65 | 14.63 | 5.51 |
| COIL-100 | 2.85 | 1.71 | 2.21 | 1.55 | 2.45 | 3.82 | 4.02 | 16.4 | 6.97 |




(a) The UG2C5D dataset (b) The Network Intrusion dataset

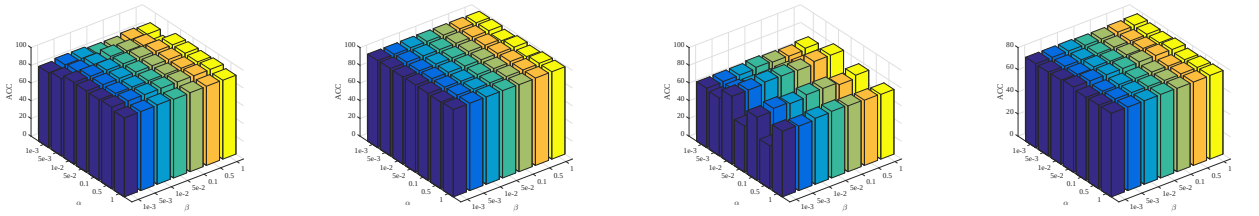
(c) The Keystroke dataset (d) The Forest Cover dataset

(e) The USPS dataset (f) The COIL-100 dataset

Fig. 1. Online performance with the six datasets over continuous landmark windows.

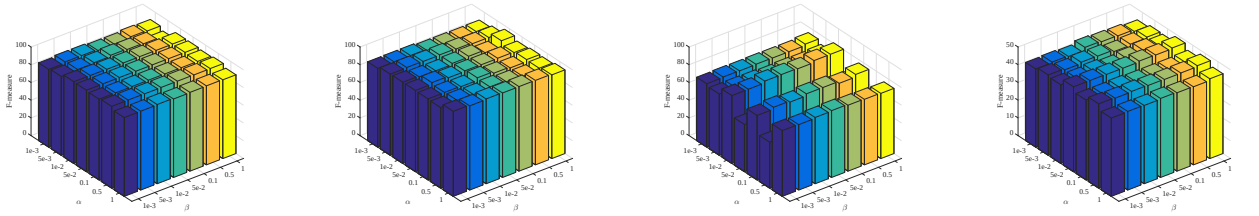
TSSRC runs faster than several competing methods such as MVStream, ESA-Stream, RRCSL and FDCM_SSR on all datasets. Overall, the computational cost of TSSRC is medium among all the competing methods. With the enhancement of computing capabilities, the computational cost may become of secondary importance to the improved performance achievable by sparse representation methods.

The online performance of TSSRC with all six datasets is displayed in Fig. 1. Each figure represents the changes in three or five evaluation metrics over time. In order to fit the scale size of the figures, the values of SSE are narrowed 10 times and the values of SC are amplified 100 times. We can see that the fluctuating range of performance is smaller in most windows for the UG2C5D and Network Intrusion datasets. In addition, ACC, $F$-measure, and NMI eventually become relatively stable after a small number of windows with the Forest Cover, Keystroke, USPS, and COIL-100 datasets. Similarly, the same phenomena can be observed from Fig. 1 in terms of SSE. This comparison demonstrates that the dictionary initialization strategy plays a critical role in ensuring the performance stability of TSSRC. Moreover, Fig. 1 shows that the values of SC are always close to 1 for the Keystroke, Forest Cover, USPS, and COIL-100 datasets. This indicates that the cluster results of the proposed method are reasonable.



(a) The UG2C5D dataset (b) The Network Intrusion dataset (c) The Keystroke dataset (d) The Forest Cover dataset

Fig. 2. ACC with different $\alpha$ and $\beta$ combinations for the four datasets.



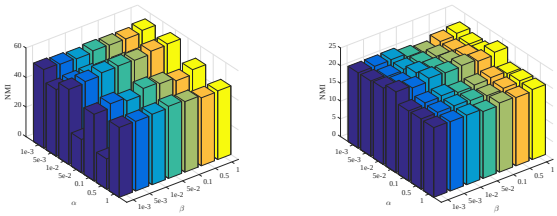
(a) The UG2C5D dataset (b) The Network Intrusion dataset (c) The Keystroke dataset (d) The Forest Cover dataset

Fig. 3. $F$-measure with different $\alpha$ and $\beta$ combinations for the four datasets.

From the experimental results, we can see TSSRC and FDCM_SSR often achieve much better results than all other methods on all datasets. However, the traditional data stream clustering methods, such as CluStream, CluTree, CluStreamKM and StreamKM++, often obtain similar clustering performance to TSSRC and FDCM_SSR on the UG2C5D, Network Intrusion and Keystroke datasets. This is because the dimensionality of the data objects or the number of the clusters is relatively small. In addition, the gap performance of clustering widens as the dimensionality of the data objects dramatically increases. The major disadvantage of these traditional methods is a lack of the ability of characterizing intrinsic structure for high-dimensional data. Simultaneously, this also verifies the effectiveness of sparse representation in exploring the subspace structures of high-dimensional data. Moreover, TSSRC performs better than the other sparse representation-based clustering methods in all datasets. In particular, the clustering metrics of TSSRC become much more stable after the implementation of several landmark windows on the Forest Cover, Keystroke, USPS, and COIL-100 datasets. This is attributed to dictionary learning that efficiently passes previously learned knowledge to the current landmark window over time. Consequently, the improvements from the proposed method show the importance of sparse representation and dictionary learning in data stream clustering.

## C. Parameter Sensitivity Analysis

TSSRC has two main parameters: $\alpha$ and $\beta$. We examined the sensitivity of TSSRC to $\alpha$ and $\beta$ with fixed values of $\mu = 0.05$ and $\rho = 8$. Specifically, we selected $\alpha$ and $\beta$ from the set $\{1e^{-3}, 5e^{-3}, 1e^{-2}, 5e^{-2}, 0.1, 0.5, 1\}$. We select four representative datasets with different $\alpha$ and $\beta$ combinations for parameter sensitivity analysis.



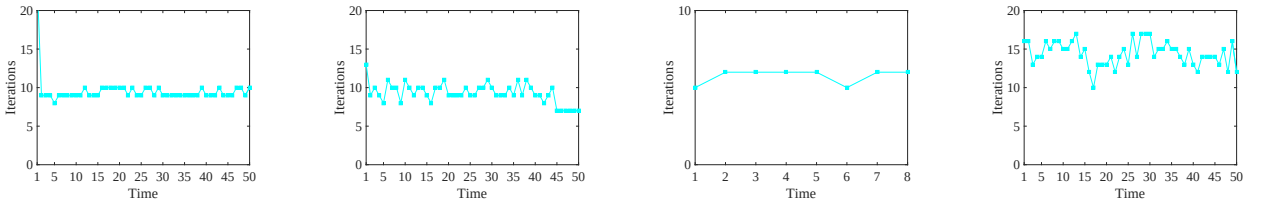
(a) The UG2C5D dataset (b) The Network Intrusion dataset

Fig. 4. NMI with different $\alpha$ and $\beta$ combinations for the two datasets.

Figs. 2 and 3 show the influence of $\alpha$ and $\beta$ on ACC and $F$-measure for the four datasets. In addition, Fig. 4 shows the influence of $\alpha$ and $\beta$ on NMI for the Keystroke and Forest Cover datasets. We can observe that TSSRC performs well for a large range of $\alpha$ and $\beta$ on all datasets except Keystroke. As $\beta$ increases from $1e^{-3}$ to 1, ACC, $F$-measure, and NMI remain relatively stable for each different value of $\alpha$ with the Keystroke dataset. This indicates that the proposed TSSRC method is only slightly sensitive to $\alpha$. The drift interval of the Keystroke dataset is 200, whereas the other three datasets have drift intervals of not less than 1,000. The comparison results indicate that the size of a single landmark window is an important factor in obtaining the desired clustering results.

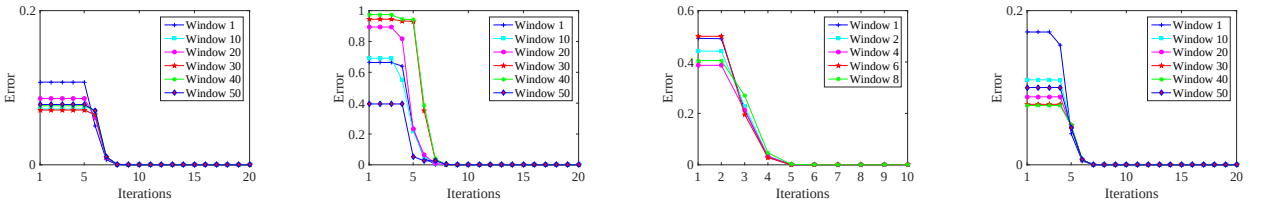
## D. Convergence Study and Sparsity Stability Analysis

First, we evaluated the number of iterations required for TSSRC to converge in the experiments. Fig. 5 shows the results for continuous landmark windows with the four datasets. The number of iterations is typically less than 20 in the experiments. Moreover, the iteration curves remain relatively stable over time. The experimental results indicate that TSSRC offers rapid convergence in data stream clustering.



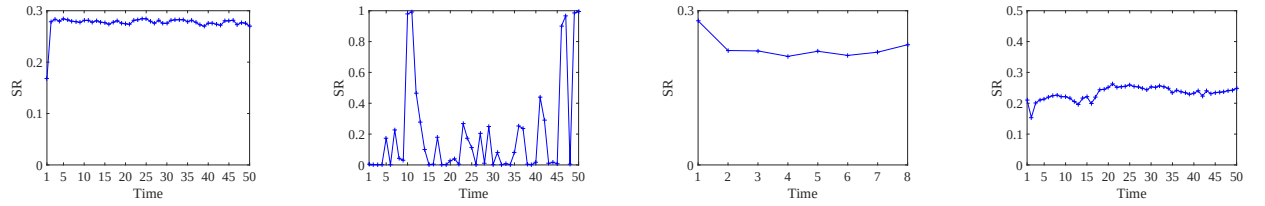

(a) The UG2C5D dataset (b) The Network Intrusion dataset (c) The Keystroke dataset (d) The Forest Cover dataset

Fig. 5. Changes in number of iterations over continuous landmark windows.




(a) The UG2C5D dataset (b) The Network Intrusion dataset (c) The Keystroke dataset (d) The Forest Cover dataset

Fig. 6. Changes in convergence time over continuous landmark windows.




(a) The UG2C5D dataset (b) The Network Intrusion dataset (c) The Keystroke dataset (d) The Forest Cover dataset

Fig. 7. Changes in SR over continuous landmark windows.

We then validated the convergence of Algorithm 1. There are a large number of landmark windows to be evaluated in the experiments. Because of space limitations, we chose a subset of the tested landmark windows to demonstrate the convergence of Algorithm 1. Fig. 6 shows the changes in the objective value of the convergence condition generated by Algorithm 1 on each iteration. The experimental results illustrate that TSSRC quickly converges to a relatively small error. Generally, for a fixed $\mu$, using a larger value of $\rho$ produces better convergence than a relatively small value. This is because the sparsity becomes stable more quickly. However, this may result in worse clustering performance.

According to Theorem 2, the sparsity stability of TSSRC can be guaranteed in a single landmark window if certain conditions are satisfied. The sparsity stability of TSSRC is considered to be a critical factor reflecting the performance stability of TSSRC over entire landmark windows. Therefore, we investigated the sparsity stability of TSSRC over time. Fig. 7 shows the changes in SR with continuous landmark windows for the four datasets. The SR results remain at a stable low rate in all datasets except Network Intrusion. Moreover, the SR results given by TSSRC are often low in the landmark windows of the Network Intrusion dataset. This verifies that the sparse representation is an effective method of characterizing the relationship among data objects in data stream clustering.

### E. Discussion

Compared with $k$-means-based methods such as CluStreamKM and StreamKM++, TSSRC seeks clusters using spectral clustering techniques involving $k$-means. The critical difference is that TSSRC adopts sparse representation techniques to capture the intrinsic characteristics of data objects, whereas CluStreamKM and StreamKM++ calculate the centroid of each CF vector using the original data objects. The affinity matrix that encodes the membership of data objects in the data stream can be constructed using sparse representation techniques. The experimental SR results show the importance of sparse representation in data stream clustering.

The number of clusters plays an important role in evaluating the clustering performance for TSSRC. The representative data objects are chosen according to the sparse representation results, where columns of the dictionary consist of representative data objects from previous landmark windows. This means that an exact number of clusters for the current landmark window may be equal to the maximum number of clusters over time. This is why the performance curve initially fluctuates before the performance improves and remains stable.

## V. Conclusion

This paper has described the TSSRC algorithm for clustering dynamic data streams, which provides insights into their underlying structure. This work overcomes the

two fundamental problems of data stream clustering by sparse representation learning, namely, evaluating the relationship among data objects and learning useful knowledge from previous landmark windows. In contrast with existing clustering techniques involving the computation of the Euclidean distance, TSSRC makes full use of sparse representation techniques to exploit the intrinsic characteristics of the data objects. TSSRC automatically selects the number of neighboring data objects, which guarantees that highly correlated data objects are represented together. Moreover, the dictionary initialization strategy has been introduced to pass previously learned knowledge to the current landmark window. In particular, the exact number of clusters for the current landmark window can be determined over time, which significantly enriches the relationship among data objects and improves the clustering performance for data streams. Although TSSRC involves iterative computations, relatively few iterations are required. Moreover, the average computational cost of TSSRC is approximately proportional to the size of a single landmark window. Appropriate dictionary and landmark window sizes can be used to control the overall computational cost of TSSRC. Different from $l_1$-norm based techniques, a hard thresholding operator is adopted to ensure a sparse representation. This dramatically reduces the computational cost of the iterative process. As our extensive experiments on stationary and dynamic datasets have shown, TSSRC has better clustering ability than several existing data stream clustering algorithms.

## References


[1] A. Zubaroğlu and V.Atalay, "Data stream clustering: a review," Artif. Intell. Rev., vol. 54, no. 2, pp. 1201–1236, Oct. 2021.

[2] S. Mansalis, E. Ntoutsi, B. Pelekis, and Y. Theodoridis, "An evaluation of data stream clustering algorithms," Stat. Anal. Data. Min.: ASA Sci. J., vol. 11, no. 4, pp. 167–187, Aug. 2018.

[3] H. L. Nguyen, Y. K. Woon, and W. K. Ng, "A survey on data stream clustering and classification," Knowl. Inf. Syst., vol. 45, no. 3, pp. 535–569, Oct. 2015.

[4] J. A. Silva, E. R. Faria, R. C. Barros, E. R. Hruschka, A. C. P. L. F. de Carvalho, and J. Gama, "Data stream clustering: a survey," ACM Comput. Surv., vol. 46, no. 1, pp. 1–31, Oct. 2013.

[5] P. Hu, X. Peng, H. Zhu, L. Zhen, J. Lin, Y. H, and D. Peng, "Deep semisupervised multiview learning with increasing views," IEEE Trans. Cybern., pp. 1–12, Sep. 2021.

[6] X. Peng, J. Feng, J. T. Zhou, Y. Lei, and S. Yan, "Deep subspace clustering," IEEE Trans. Neural Netw. Learn Syst., vol. 31, no. 12, pp. 5509–5521, Dec. 2020.

[7] C. Zhang, H. Fu, Q. Hu, X. Cao, Y. Xie, D. Tao, and D. Xu, "Generalized latent multi-view subspace clustering," IEEE Trans. Pattern Anal. and Mach. Intell., vol. 42, no. 1, pp. 1–1, Jan. 2020.

[8] J. Chen, H. Mao, H. Zhang, and Z. Yi, "Symmetric low-rank preserving projections for subspace learning," Neurocomputing, vol. 315, no. 13, pp. 381–393, Nov. 2018.

[9] J. Chen, H. Zhang, H. Mao, Y. Sang, and Z. Yi, "Symmetric low-rank representation for subspace clustering," Neurocomputing, vol. 173, no. 15, pp. 1192–1202, Jan. 2016.

[10] D. Krleža1, B. Vrdoljak, and M. Brčić, "Statistical hierarchical clustering algorithm for outlier detection in evolving data streams," Mach. Learn., vol. 110, no. 1, pp. 139–184, Jan. 2021.

[11] T. Pham, D. Kottke, G. Krempl, and B. Sick, "Stream-based active learning for sliding windows under the influence of verification latency," Mach. Learn., pp. 1–26, Nov. 2021.

[12] C. Fahy, S. Yang, and M. Gongora, "Ant colony stream clustering: A fast density clustering algorithm for dynamic data streams," IEEE Trans. Cybern., vol. 49, no. 6, pp. 2215–2228, Jun. 2019.

[13] L. Huang, C. D. Wang, H. Y. Chao, and P. S. Yu, "MVStream: Multiview data stream clustering," IEEE Trans. Neural Netw. Learn Syst., vol. 31, no. 9, pp. 3482–3496, Oct. 2019.

[14] A. Amini, H. Saboohi, T. Herawan, and T. YingWah, "Mudistream: A multi density clustering algorithm for evolving data stream," J. Netw. Comput. Appl., vol. 59, no. 1, pp. 370–385, Mar. 2016.

[15] A. Zhou, F. Cao, Y. Yan, C. Sha, and X. He, "Distributed data stream clustering: A fast em-based approach," in Int. Conf. Adv. Data Min. Appl., Istanbul, Turkey, Apr. 2007, pp. 736–745.

[16] T. Zhang, R. Ramakrishnan, and M. Livny, "Birch: A new data clustering algorithm and its applications," Data Min Knowl. Discov., vol. 1, no. 2, pp. 2215–2228, Jun. 1997.

[17] Y. Wu, L. Chen, S. Li, and J. Chen, "An adaptive algorithm for dealing with data stream evolution and singularity," Inf. Sci., vol. 545, pp. 312–330, Feb. 2021.

[18] S. Laohakiat and V. Sa-Ing, "An incremental density-based clustering framework using fuzzy local clustering," Inf. Sci., vol. 547, pp. 404–426, Feb. 2021.

[19] M. O. Attaoui, H. Azzag, M. Lebbah, and N. Keskes, "Subspace data stream clustering with global and local weighting models," Neural. Comput. Appl., vol. 33, pp. 3691–3712, 2021.

[20] R. Hyde, P. Angelov, and A. R. MacKenzie, "Fully online clustering of evolving data streams into arbitrarily shaped clusters," Inf. Sci, vol. 382, no. 1, pp. 96–114, Mar. 2017.

[21] C. C. Aggarwal, J. Han, J. Wang, and P. S. Yu, "A framework for clustering evolving data streams," in in Proc. Int. Conf. VLDB, Berlin, Germany, Sept. 2003, pp. 81–92.

[22] J. Sui, Z. Liu, L. Liu, A. Jung, and X. Li, "Dynamic sparse subspace clustering for evolving high-dimensional data streams," IEEE Trans. Cybern., pp. 1–14, Nov. 2020.

[23] C. Zhang, Y. Cui, Z. Han, J. T. Zhou, H. Fu, and Q. Hu, "Deep partial multi-view learning," IEEE Trans. Pattern Anal. and Mach. Intell., pp. 86–99, Nov. 2020.

[24] Z. Bu, H. J. Li, J. Cao, Z. Wang, and G. Gao, "Dynamic cluster formation game for attributed graph clustering," IEEE Trans. Cybern., vol. 49, no. 1, pp. 328–341, Jan. 2019.

[25] Y. Li, H. Li, Z. Wang, B. Liu, J. Cui, and H. Fei, "ESA-stream: Efficient self-adaptive online data stream clustering," IEEE Trans. Knowl. Data Eng., pp. 1–13, Apr. 2020.

[26] S. Zhou, Z. Xu, and F. Liu, "Method for determining the optimal number of clusters based on agglomerative hierarchical clustering," IEEE Trans. Neural Netw. Learn Syst., vol. 28, no. 12, pp. 3007–3017, Dec. 2017.

[27] Q. Wang, R. Liu, M. Chen, and X. Li, "Robust rank-constrained sparse learning: a graph-based framework for single view and multiview clustering," IEEE Trans. Cybern., pp. 1–12, Apr. 2021.

[28] R. X. Ding, X. Wang, K. Shang, B. Liu, and F. Herrera, "Sparse representation-based intuitionistic fuzzy clustering approach to find the group intra-relations and group leaders for large-scale decision making," IEEE Trans. Fuzzy Syst., vol. 27, no. 3, pp. 559–573, Mar. 2019.

[29] J. Gu, L. Jiao, S. Yang, and F. Liu, "Fuzzy double $c$-means clustering based on sparse self-representation," IEEE Trans. Fuzzy Syst., vol. 26, no. 2, pp. 612–626, Apr. 2018.

[30] C. Bao, H. Ji, Y. Quan, and Z. Shen, "Dictionary learning for sparse coding: Algorithms and convergence analysis," IEEE Trans. Signal Process., vol. 38, no. 7, pp. 1356–1369, Jul. 2016.

[31] J. Mairal, F. Bach, H. Ponce, and G. Sapiro, "Online dictionary learning for sparse coding," in in Proc. 26th Int. Conf. Mach. Learn. (ICML), Montreal, Canada, Jun. 2009, pp. 689–696.

[32] A. Mensch, J. Mairal, B. Thirion, and G. Varoquaux, "Dictionary learning for massive matrix factorization," in in Proc. 34th Int. Conf. Mach. Learn. (ICML), New York City, NY, USA, Jun. 2016, pp. 1737–1746.

[33] J. Chen, S. Yang, Z. Wang, and H. Mao, "Efficient sparse representation for learning with high-dimensional data," IEEE Trans. Neural Netw. Learn Syst., pp. 1–15, Oct. 2021.

[34] D. L. Donoho, "For most large underdetermined systems of linear equations the minimal $l_1$-norm solution is also the sparsest solution," Commun. Pure. Appl. Math., vol. 59, no. 6, pp. 797–829, Mar. 2006.

[35] S. Luhr and M. Lazarescu, "Incremental clustering of dynamic data streams using connectivity based representative points," Data Knowl. Eng., vol. 68, no. 1, pp. 1–27, Jan. 2009.
[36] L. Tu and Y. Chen, "Stream data clustering based on grid density and attraction," ACM Trans. Knowl. Discov. Data, vol. 3, no. 3, pp. 1–27, Jul. 2009.
[37] F. Cao, M. Estert, W. Qian, and A. Zhou, "Density-based clustering over an evolving data stream with noise," in in Proc. SIAM Int. Conf. Data Min., Bethesda, MD, USA, Apr. 2006, pp. 328–339.
[38] 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.
[39] M. R. Ackermann, M. Märtens, C. Raupach, C. Lammersen, and C. Sohler, "Streamkm++: A clustering algorithm for data streams," J. Exp. Algorithmic, vol. 17, no. 2, pp. 1–30, May 2012.
[40] E. Elhamifar and R. Vidal, "Sparse subspace clustering algorithm, theory, and applications," IEEE Trans. Pattern Anal. and Mach. Intell., vol. 35, no. 11, pp. 2765–2781, Nov. 2013.
[41] M. Bahri, A. Bifet, S. Maniu, and H. M. Gome, "Survey on feature transformation techniques for data streams," in in Proc. 29th Int. Conf. Int. Jt. Conf. Artif. Intell, Montreal-themed Virtual Reality, Aug. 2021, pp. 4796–4802.
[42] M. Zheng, J. Bu, C. Chen, C. Wang, L. Zhang, G. Qiu, and D. Cai, "Graph regularized sparse coding for image representation," IEEE Trans. Image Process., vol. 20, no. 5, pp. 1327–1336, May 2011.
[43] Z. Lin, R. Liu, and Z. Su, "Linearized alternating direction method with adaptive penalty for low-rank representation," in Adv. Neural. Inf. Process. Syst., Vancouver, British Columbia, Canada, Dec. 2011, pp. 612–620.
[44] T. Blumensath and M. E. Davies, "Iterative thresholding for sparse approximations," J. Fourier Anal. Appl., vol. 14, no. 5, pp. 629–654, Sept. 2008.
[45] J. Mairal, F. Bach, J. Ponce, and G. Sapiro, "Online learning for matrix factorization and sparse coding," J. Mach. Learn. Res., vol. 11, no. 1, pp. 19–60, Jul. 2010.
[46] J. Shi, J. Malik, and S.Sastry, "Normalized cuts and image segmentation," IEEE Trans. Pattern Anal. and Mach. Intell., vol. 22, no. 8, pp. 181–214, Aug. 2000.
[47] 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. and Mach. Intell., vol. 35, no. 1, pp. 171–184, Jan. 2013.
[48] U. V. Luxburg, "A tutorial on spectral clustering," Stat. Comput., vol. 17, no. 4, pp. 395–416, Aug. 2007.
[49] P. Kranen, I. Assent, C. Baldauf, and T. Seidl, "The clustree: indexing micro-clusters for anytime stream mining," Knowl. Inf. Syst., vol. 29, no. 2, pp. 249–272, Nov. 2011.
[50] A. Bifet, G. Holmes, R. Kirkby, and B. Pfahringer, "MOA: massive online analysis," J. Mach. Learn. Res., vol. 11, no. 1, pp. 1601–1604, May 2010.
[51] C. D. Manning, P. Raghavan, and H. Sch'utze, Introduction to Information Retrieval. New York, NY, USA: Cambridge University Press, 2008.
[52] D. Dua and C. Graff. (2017, Apr.) UCI machine learning repository. [Online]. Available: http://archive.ics.uci.edu/ml
[53] V. M. A. Souza, D. F. Silva, J. Gama, and G. E. A. P. A. Batista, "Data stream classification guided by clustering on nonstationary environments and extreme verification latency," in in Proc. SIAM Int. Conf. Data Min., Vancouver, British Columbia, Canada, Apr. 2015, pp. 873–881.

Jie Chen received the BSc degree in Software Engineering, MSc degree and PhD degree in Computer Science from Sichuan University, Chengdu, China, in 2005, 2008 and 2014, respectively. He is currently an Associate Professor in the College of Computer Science, Sichuan University, China. His current research interests include machine learning, big data analysis and deep neural networks.

Zhu Wang received the B.M., MSc., and LLD. degrees in Civil and Commercial Law from Renmin University of China, China in 2003, 2006, and 2009, respectively. He is currently a Professor in Law and Director of Institute of Rule of Law of Market Economy, Sichuan University, Chengdu, China. His research interests are mainly in Tort, insurance law, constitution and big data analysis of law.

Shengxiang Yang (Senior Member, IEEE) received the Ph.D. degree from Northeastern University, Shenyang, China, in 1999. He is currently a Professor of Computational Intelligence and the Deputy Director of the Institute of Artificial Intelligence, School of Computer Science and Informatics, De Montfort University, Leicester, U.K. He has over 360 publications with an H-index of 64 according to Google Scholar. His current research interests include evolutionary computation, swarm intelligence, artificial neural networks, data mining and data stream mining, and relevant real-world applications. Dr. Yang serves as an Associate Editor/Editorial Board Member of a number of international journals, such as the IEEE Transactions on Evolutionary Computation, IEEE Transactions on Cybernetics, Information Sciences, and CAAI Transactions on Intelligence Technology.

Hua Mao received the B.S. degree and M.S. degree in Computer Science from University of Electronic Science and Technology of China (UESTC) in 2006 and 2009, respectively. She received her Ph.D. degree in Computer Science and Engineering from Aalborg University, Denmark in 2013. She is currently a Senior Lecturer in Department of Computer and Information Sciences, Northumbria University, U.K. Her current research interests include Deep Neural Networks and Big Data.