Multi-view representation learning for data stream clustering*

Jie Chen[a], Shengxiang Yang[b,*] and Zhu Wang[c]

[a]College of Computer Science, Sichuan University, Chengdu, 610065, Sichuan, P.R. China
[b]School of Computer Science and Informatics, De Montfort University, Leicester, LE1 9BH, U.K.
[c]Law School, Sichuan University, Chengdu, 610065, Sichuan, P.R. China

ARTICLE INFO

Keywords:
Data stream clustering
representation learning
multi-view data
high-dimensional data

ABSTRACT

Data stream clustering provides valuable insights into the evolving patterns of long sequences of continuously generated data objects. Most existing clustering methods focus on single-view data streams. In this paper, we propose a multi-view representation learning (MVRL) method for multi-view clustering of data streams. We first introduce an integrated representation learning model to learn a fused sparse affinity matrix across multiple views for spectral clustering. Motivated by the optimization procedure of the integrated representation learning model, we propose three consecutive stages: collaborative representation, the construction of individual global affinity matrices using a mapping function, and the calculation of a fused sparse affinity matrix using Euclidean projection. These stages allow the effective capture of the global and local structures of high-dimensional data objects. Moreover, each stage has a closed-form solution, which determines the upper bound of the computational cost and memory consumption. We then employ the construction residuals of the collaborative representation to adaptively update a dynamic set, which is used to preserve the representative data objects. The dynamic set efficiently transfers previously learned useful knowledge to the arriving data objects. Extensive experimental results on multi-view data stream datasets demonstrate the effectiveness of the proposed MVRL method.

1. Introduction

With the advance of electronic device technology, large amounts of data in data streams are continuously generated at high rates [33, 39]. For example, the amount of network traffic generated by computers has increased over time. Various types of attacks can be detected in network traffic: for example, denial of service, malware spreading, scanning, and command-and-control attacks. In addition, surveillance cameras collect large quantities of object trajectory data. Trajectory analysis can be implemented by trajectory clustering and used for applications such as traffic monitoring, understanding the activity of humans and vehicles, and the discovery of abnormal actions. A data stream is a potentially unbounded, ordered sequence of data objects, which may be generated from multiple signal sources or described by different modalities. Such data streams are referred to as multi-view data streams [19]. In contrast to streams with a single view, consistency information and complementary information characterizing the relationship between the data objects is typically provided in streams with multiple views [11, 18, 42, 45].

Data stream clustering refers to the task of efficiently partitioning the arriving data objects into several clusters according to a particular similarity measure [39, 48]. It is a fundamental technique for exploring the structures underlying multi-view data streams and provides valuable information for real-time decision-making [19, 20, 34, 43]. Traditional clustering techniques usually assume that there are a finite number of data objects generated by an unknown, stationary probability distribution, where the number of clusters is known [10, 26, 31, 36, 44]. The intrinsic nature of data streams requires the development of clustering algorithms capable of performing real-time incremental processing. They also need to comprise a single-pass method, be able to detect concept drift, and work with a limited amount of memory. Because the generation of data objects is unknown and possibly nonstationary, the probability distribution of the arriving data objects may change over time; this phenomenon is known as concept drift [48]. This implies that data stream clustering models should be able to dynamically evolve to characterize the intrinsic structures of the arriving data objects over time. In addition, the data objects should be discarded according to the limited size of the memory resource s once they have been processed. This requires a forgetting mechanism for the data stream clustering models

*This work was supported in part by National Key Project under Grant GJXM92579, in part by National Natural Science Foundation of China (NSFC) under Grant 61303015, and in part by Sichuan Science and Technology Program under Grant 2021YJ0078.
*Corresponding author.
chenjie2010@scu.edu.cn (J. Chen); syang@dmu.ac.uk (S. Yang); wangzhu@scu.edu.cn (Z. Wang)

so that they may discard outdated data objects at a later time. Moreover, real-time processing of a data stream requires a process that can continuously cluster data objects within a specified time [48].

To address the aforementioned problems from various perspectives, a variety of data stream clustering algorithms have been proposed, such as hierarchy-based [47], density-based [13, 22], and partitioning-based [1–3] clustering algorithms. These algorithms employ various data structures that preserve statistical summaries of data streams to capture the dynamic evolution associated with concept drift of the arriving data objects and discover clusters of arbitrary shapes. For example, balanced iterative reducing and clustering using hierarchies (BIRCH) is a classical hierarchical clustering algorithm that constructs a height-balanced tree using a clustering feature (CF) vector [47]. A variant of BIRCH called CluStream employs CF vectors to create micro-clusters that incrementally update the summary cluster information about the data streams [2]. Similarly, DenStream is a density-based data stream clustering algorithm that uses micro-clusters to store statistical information about data streams [8]. The CF vector has the ability to effectively produce statistical summaries for the data objects when the dimensionality of data objects is relatively low. In addition, Lughofer and Sayed-Mouchaweh [29] constructed an extension of evolving vector quantization by incrementally updating ellipsoidal clusters in arbitrary positions. Pehlivan and Turksen [35] presented a multiplicative fuzzy regression function, which corresponds to each cluster, determined by the transformations of membership values. Borlea *et al.* [5] presented the unified form clustering algorithm, which treats the fuzzy c-means and k-means algorithms as a single configurable algorithm. These algorithms often work well under specific circumstances. However, high-dimensional data often lie in low-dimensional structures in practice [17]. In particular, the high-dimensional data objects in a low-dimensional subspace that belong to the same cluster are often distributed arbitrarily and not around a centroid. Consequently, data structures that take advantage of the spatial proximity of the data objects may not be suitable for statistical summaries of high-dimensional data streams. Several classical dimensionality reduction techniques, such as principal component analysis (PCA) [41], may be used to preprocess high-dimensional data streams. However, dimensionality reduction requires an extra parameter to determine the number of dimensions. In addition, these approaches focus mainly on single-view data stream clustering. Huang *et al.* [19] presented a multi-view data stream clustering method to integrate information from multiple views and abstract summary statistics from the integrated features simultaneously, by performing iterative computations.

Subspace clustering methods typically attempt to seek the intrinsic low-dimensional structures of high-dimensional data [12, 46]. For example, two classes of representative algorithms, low-rank representation (LRR) [27], sparse subspace clustering (SSC) [17], and their extensions [7] take advantage of the self-expressiveness property of high-dimensional data. LRR and SSC determine low-dimensional structures by considering the global and local geometric structures of high-dimensional data, respectively. In addition, subspace learning is an intuitive way to develop an adaptive model that is able to simultaneously capture the global and local geometric structures to improve the performance of subspace clustering. For example, Chen *et al.* [9] extended LRR by integrating a symmetric constraint into the low-rankness property of high-dimensional data representation. Brbić *et al.* [7] proposed a low-rank SSC (LRSSC) method to encourage low-rank and sparse representations by introducing S_0 and l_0 pseudo-norm regularizations, respectively. LRRSC employs two corresponding penalty parameters to maintain the convexity of the sparse and low-rank constrained subproblems. However, it is an intractable problem to theoretically determine which factor (rank or sparsity) plays a more important role without any prior knowledge of the data distribution. In addition, Sui *et al.* proposed an evolutionary dynamic SSC algorithm for evolving high-dimensional data streams [40]. This method uses sparse representation to cope with the time-varying attributes of subspaces, such as subspace emergence, disappearance, and recurrence, in the evolving data streams. Hence, capturing the global and local geometric structures of high-dimensional data simultaneously remains a challenge.

Traditional subspace clustering methods are not suitable for data stream clustering, although they often achieve impressive results on stationary datasets. Similarly, traditional data clustering methods based on the Euclidean distance metric cannot effectively measure the relationships between data objects when the data objects are high-dimensional. These drawbacks motivated us to take advantage of the self-expressiveness property of high-dimensional data to perform data stream clustering by seeking intrinsic low-dimensional structures of high-dimensional data objects in data streams. In this paper, we present our proposed multi-view representation learning (MVRL) method for multi-view data stream clustering. We start with an integrated representation learning (IRL) model, which consists of three critical componentsa collaborative representation, mapping function, and sparsity regularizationto learn a fused sparse affinity matrix across multiple views. The optimization problem of the IRL model can be solved by an alternating optimization method, which requires iterative computations before convergence. Motivated by the optimization procedure, we transform the three components from the IRL to three consecutive stages: collaborative representation, the construction

Table 1
Definitions of symbols.

Symbol	Definition
n	Number of data objects
d	Dimension of data objects
$\mathbf{X} \in \mathbb{R}^{d \times n}$	Data matrix
$\mathbf{X}^T \in \mathbb{R}^{n \times d}$	The transpose of $\mathbf{X}$
$\mathbf{X}^{-1}$	The inverse of $\mathbf{X}$
$diag(\mathbf{X})$	The vector containing the n diagonal elements of $\mathbf{X}$
$tr(\mathbf{X})$	The trace of $\mathbf{X}$
$\|\mathbf{X}\|_0$	The number of nonzero elements in $\mathbf{X}$
$\|\mathbf{X}\|_1$	The l_1-norm of $\mathbf{X}$
$\|\mathbf{X}\|_F$	The Frobenius norm of $\mathbf{X}$
$\|\mathbf{X}\|_*$	The nuclear norm of $\mathbf{X}$
$\mathbf{W} \in \mathbb{R}^{n \times n}$	Affinity matrix

of individual global affinity matrices using a mapping function, and the calculation of the fused sparse affinity matrix using Euclidean projection. Thereby, we capture the global and local structures of high-dimensional data objects in data streams. The consistency information across the multiple views of the data objects is collected in the first two stages. The complementary information across the multiple views is exploited in the third stage. Each stage has a closed-form solution, which enables MVRL to satisfy the requirements for real-time processing and limited memory consumption in data stream clustering. We further take advantage of the construction residuals of the collaborative representation to adaptively update a dynamic set that is used to preserve the representative data objects over time. The dynamic set efficiently incorporates previously learned useful knowledge in the processing for arriving data objects. Simultaneously, the changes in the dynamic set are employed to detect concept drift. The original aspect of our work is that the proposed approach takes advantage of the self-expressiveness of the collaborative representation to transfer previously learned useful knowledge to the subsequent windows for multi-view data stream clustering.

Our major contributions are summarized as follows:

1. Three consecutive stages calculate individual affinity matrices that simultaneously explore global and local structures of high-dimensional data objects in data streams.

2. Each stage has a closed-form solution that determines the upper bound of the computational cost and memory consumption.

3. The representative data objects that transfer previously learned useful knowledge to later data stream processing are preserved in the dynamic set, which can be used to detect concept drift.

4. Extensive experimental results on real data stream datasets demonstrate the effectiveness and efficiency of MVRL.

The remainder of this paper is organized as follows. We briefly review some related work in Section 2. In Section 3, we present the proposed MVRL method in detail. Extensive experiments were conducted to evaluate the clustering performance of the proposed method; the results are presented in Section 4. Finally, we conclude the paper in Section 5.

2. Related work

In this section, we briefly review subspace clustering methods and several critical components of data stream clustering methods. For consistency, we define the symbols that we use in Table 1.

2.1. Subspace clustering

Consider a set of n data samples $\mathbf{X} = \left[\mathbf{x}_1, \mathbf{x}_2, ..., \mathbf{x}_n\right] \in \mathbb{R}^{d \times n}$ approximately drawn from a union of multiple linear subspaces. Subspace clustering is intended to group the data samples into their respective subspaces. The first step of the clustering process is the construction of an affinity matrix $\mathbf{W} \in \mathbb{R}^{n \times n}$, whose elements measure the similarity between data samples.

Some subspace clustering algorithms take advantage of the self-expressiveness property of high-dimensional data and typical matrix norm constraints to employ the affinity matrix for spectral clustering. For example, the LRR

algorithm [27] seeks a low-rank data representation $\mathbf{Z} \in \mathbb{R}^{n \times n}$ of $\mathbf{X}$ by solving the following optimization problem:

$$\min_{\mathbf{Z}} rank\,(\mathbf{Z}) \quad s.t. \quad \mathbf{X} = \mathbf{XZ}, \tag{1}$$

where $rank\,(\cdot)$ denotes the rank of a matrix. Because of the discrete nature of the rank function, the nuclear norm is often considered to be a good surrogate for the rank function. Hence, (1) can be rewritten as

$$\min_{\mathbf{Z}} \|\mathbf{Z}\|_* \quad s.t. \quad \mathbf{X} = \mathbf{XZ}. \tag{2}$$

The closed-form solution of (1) is uniquely obtained by

$$\mathbf{Z} = \mathbf{V}\mathbf{V}^T, \tag{3}$$

where $\mathbf{X} = \mathbf{U}\Sigma\mathbf{V}^T$ is the singular value decomposition (SVD) of $\mathbf{X}$.

The SSC algorithm assumes that each data sample is represented as a sparse linear combination of the other data samples from the same subspace [17]. A sparse representation $\mathbf{Z}$ is obtained by solving the following l_1-minimization optimization:

$$\min_{\mathbf{Z}} \|\mathbf{Z}\|_1 \quad s.t. \quad \mathbf{X} = \mathbf{XZ},\ diag\,(\mathbf{Z}) = 0. \tag{4}$$

Equation (4) can be solved by l_1-norm-based optimization techniques, for example, a feature-sign search algorithm [24]. The LRSSC method combines sparse and low-rank constraints:

$$\min_{\mathbf{Z}} \frac{1}{2} \|\mathbf{X} - \mathbf{XZ}\|_F^2 + \alpha\|\mathbf{Z}\|_{S_0} + \beta\|\mathbf{Z}\|_1 \quad s.t. \quad diag\,(\mathbf{Z}) = 0, \tag{5}$$

where α and β are parameters and $\|\cdot\|_{S_0}$ is a rank function [7]. Equation (5) can be solved using the alternating direction method of multipliers (ADMM) framework [6]. After $\mathbf{Z}$ is obtained, the affinity matrix $\mathbf{W}$ can be calculated using an absolute symmetrization step as

$$\mathbf{W} = \left(\mathbf{Z} + \mathbf{Z}^T\right)/2. \tag{6}$$

A spectral clustering algorithm, for example, NCuts, can be applied to the affinity matrix $\mathbf{W}$ to obtain the memberships of data samples [30].

2.2. Data stream clustering

A data stream $\mathcal{S} \in \mathbb{R}^{d \times N}$ is a massive sequence of data objects $\mathcal{S} = \left\{\mathbf{s}_1, \mathbf{s}_2, ..., \mathbf{s}_n\right\}$, where d is the dimensionality of the data objects and $N \to \infty$. It is not possible to store all data objects from a data stream because they are infinite in number. For data stream clustering, it is essential to develop special data structures that enable the data objects to be incrementally summarized. The most commonly used data structures are CF vectors, micro-clusters, prototype arrays, coreset trees, and grids [2, 47]. CF vectors preserve a summary of the data objects in an incremental manner. Micro-clusters retain the basic CF components and extend the summary by adding two more componentsthe sum of the timestamps and the sum of the squares of the timestampsso that temporal cluster metrics can be computed. Prototype arrays store the data partition of a set of representative data objects, and coreset trees store the summary in a binary tree. A grid constitutes a summary of the data objects. Clusters can be described by these data structures using various cluster properties, for example, the cluster centroid. In particular, these data structures preserve the summary statistics of long sequences of data objects without the need to store the actual objects.

Several window-based models have been developed for efficiently handling recent data objects [48]. There are three classical window-based models: damped window-based, landmark window-based, and sliding window-based models. In the damped window-based model, more recent data objects are given more weight than less recent ones. The importance of the data objects slowly decreases over time. In the landmark window-based model, the window length represents the number of data objects, and all data objects are equally distributed between a number of windows according to window length. In the damped window-based model, the oldest data object is removed from the window when a new data object is added. In the landmark and damped window-based models, all data objects in a window are given equal weight.

Data stream clustering methods are usually divided into two categoriesincremental learning and two-phase learningaccording to the method of data processing [39]. Incremental learning usually performs clustering in a single-pass over the dynamic set of data objects at a certain time and simultaneously maintains clusters using a particular adaptive strategy. Two-phase learning consists of two components: online and offline components. The online component summarizes the data stream in a real-time manner with the help of particular data structures. The offline component uses the summary statistics to perform clustering at a high level. For example, the CluStream algorithm is composed of these two components [2]. In the online phase, CluStream maintains a set of q micro-clusters as the data objects continually arrive. In the offline phase, it applies the k-means algorithm to the q micro-clusters to obtain the clustering results.

3. Multi-view data stream clustering

In this section, we present the MVRL algorithm for multi-view clustering of data streams. The algorithm applies a landmark window-based model to the data streams, where each data stream is separated by landmarks with a fixed number of data objects. The arriving data objects from the current landmark remain in the window until a new landmark is reached. We first focus on the multi-view clustering of the arriving data objects observed in the window. We then present an adaptive strategy to update a dynamic set of the representative data objects; this transfers previously learned knowledge to the processing of subsequent windows.

3.1. Integrated representation learning (IRL)

We consider a set of multi-view data $\mathbf{X}_t = \left\{ \mathbf{X}_t^{(v)} \in \mathbb{R}^{d_v \times n} \right\} \left(1 \le v \le n_v\right)$ containing n data objects in the tth window, where $\mathbf{X}_t^{(v)}$ is the vth view of the multi-view data, d_v denotes the dimensionality of features in the vth view, and n_v is the number of views. Each data object has an individual feature in each view, and all features of each data object are strictly aligned in multiple views. Each view $\mathbf{X}_t^{(v)}$ contains n features of data objects: $\mathbf{X}_t^{(v)} = \left[\mathbf{x}_{1,t}^{(v)}, \mathbf{x}_{2,t}^{(v)}, ..., \mathbf{x}_{n,t}^{(v)}\right]$ $\left(\mathbf{x}_{i,t}^{(v)} \in \mathbb{R}^{d_v}, 1 \le i \le n\right)$. To explore the consistency information and complementary information across the multiple views, we capture the global and local structures of high-dimensional data objects simultaneously. Specifically, we introduce an IRL model, which determines the intrinsic structures of high-dimensional data objects to learn the individual affinity matrices for the multiple views of the data objects. The objective function of the IRL model is expressed as follows:

$$\begin{aligned} &\min_{\mathbf{Z}_t^{(v)}, \mathbf{W}_t^{(v)}} \sum_{v=1}^{n_v} \left\| \mathbf{X}_t^{(v)} - \mathbf{X}_t^{(v)} \mathbf{Z}_t^{(v)} \right\|_F^2 + \sum_{v=1}^{n_v} \alpha^{(v)} \left\| \mathbf{Z}_t^{(v)} \right\|_F^2 + \sum_{v=1}^{n_v} \beta^{(v)} \left\| \mathbf{W}_t^{(v)} \right\|_1 + \sum_{v=1}^{n_v} \lambda^{(v)} \left\| \mathbf{W}_t^{(v)} - f\left(\mathbf{Z}_t^{(v)}\right) \right\|_F^2 \\ &s.t. \quad W_{ij}^{(v)} \ge 0, \ \left(\mathbf{W}_i^{(v)}\right)^T \mathbf{1} = 1, \end{aligned} \tag{7}$$

where $\alpha^{(v)}$, $\beta^{(v)}$, and $\lambda^{(v)}$ are positive parameters and $f(\cdot)$ is a mapping function. The function $f(\cdot)$ is used to construct the global affinity matrices for multiple views using collaborative representation matrices. In (7), $\mathbf{Z}_t^{(v)} \in \mathbb{R}^{n \times n}$ and $\mathbf{W}_t^{(v)} \in \mathbb{R}^{n \times n}$ represent the collaborative representation matrix and the individual affinity matrix for the vth view, respectively. In the constraints of (7), $\mathbf{W}_i^{(v)}$ is the ith column of $\mathbf{W}^{(v)}$ and $W_{ij}^{(v)}$ is the jth element of $\mathbf{W}_i^{(v)}$.

There are three critical components in (7): a collaborative representation, the mapping function $f(\cdot)$, and a sparsity regularization. All variables in (7) related to each view can be updated independently. For simplicity, (7) can be rewritten with respect to the vth view as follows:

$$\begin{aligned} &\min_{\mathbf{Z}_t^{(v)}, \mathbf{W}_t^{(v)}} \left\| \mathbf{X}_t^{(v)} - \mathbf{X}_t^{(v)} \mathbf{Z}_t^{(v)} \right\|_F^2 + \alpha^{(v)} \left\| \mathbf{Z}_t^{(v)} \right\|_F^2 + \beta^{(v)} \left\| \mathbf{W}_t^{(v)} \right\|_1 + \lambda^{(v)} \left\| \mathbf{W}_t^{(v)} - f\left(\mathbf{Z}_t^{(v)}\right) \right\|_F^2 \\ &s.t. \quad W_{ij}^{(v)} \ge 0, \ \left(\mathbf{W}_i^{(v)}\right)^T \mathbf{1} = 1. \end{aligned} \tag{8}$$

The collaborative representation matrix $\mathbf{Z}_t^{(v)}$ and individual affinity matrix $\mathbf{W}_t^{(v)}$ for the vth view are learned by minimizing (8), which can be effectively performed using an alternating optimization method. Each variable is iteratively updated while the other variables are fixed, until convergence.

When $\mathbf{W}_t^{(v)}$ is fixed, (8) is equivalent to the following problem:

$$\min_{\mathbf{Z}_t^{(v)}} \left\| \mathbf{X}_t^{(v)} - \mathbf{X}_t^{(v)} \mathbf{Z}_t^{(v)} \right\|_F^2 + \alpha^{(v)} \left\| \mathbf{Z}_t^{(v)} \right\|_F^2 + \lambda^{(v)} \left\| \mathbf{W}_t^{(v)} - f\left(\mathbf{Z}_t^{(v)} \right) \right\|_F^2. \tag{9}$$

Here, $\mathbf{Z}_t^{(v)}$ can be calculated by solving (9), which can be achieved by the ADMM framework [6]. The collaborative representation $\mathbf{Z}_t^{(v)}$ is used to capture the global structures of high-dimensional data objects.

When $\mathbf{Z}_t^{(v)}$ is fixed, (8) is equivalent to the following problem:

$$\min_{\mathbf{W}_t^{(v)}} \beta^{(v)} \left\| \mathbf{W}_t^{(v)} \right\|_1 + \lambda^{(v)} \left\| \mathbf{W}_t^{(v)} - f\left(\mathbf{Z}_t^{(v)} \right) \right\|_F^2 \quad s.t. \quad W_{ij}^{(v)} \geq 0, \ \left(\mathbf{W}_i^{(v)} \right)^T \mathbf{1} = 1. \tag{10}$$

Here, $\mathbf{W}_t^{(v)}$ is updated by solving (10), which can be achieved by l_1-based optimization techniques [14, 24]. The sparsity regularization is integrated into the IRL model to explore the local structure of high-dimensional data objects.

The two steps are repeated until convergence for each view. Individual affinity matrices $\mathbf{W}_t^{(v)} \ \left(1 \leq v \leq n_v\right)$ are finally obtained for the multiple views of the data objects. However, this optimization procedure requires several iterations before convergence. Because the number of iterations is unknown, it is not easy to theoretically estimate the upper bound of the computational cost. As a result, this approach may not be suitable for data stream clustering because of the requirement for real-time processing. Hence, we omit the entire procedure for solving (9) and (10).

Motivated by the above analysis, we convert the three components of (8) to three consecutive stages to calculate individual affinity matrices for multiple views. In the first stage, the collaborative representation enables $\mathbf{Z}_t^{(v)}$ to preserve the global structures of high-dimensional data objects. By setting $\lambda^{(v)} = 0$, we reformulate (9) as a collaborative representation problem with respect to $\mathbf{Z}_t^{(v)}$:

$$\min_{\mathbf{Z}_t^{(v)}} \left\| \mathbf{X}_t^{(v)} - \mathbf{X}_t^{(v)} \mathbf{Z}_t^{(v)} \right\|_F^2 + \alpha^{(v)} \left\| \mathbf{Z}_t^{(v)} \right\|_F^2. \tag{11}$$

For convenience, we use parameter α instead of $\alpha^{(v)}$ for all views. Equation (11) can be solved with a closed-form solution:

$$\mathbf{Z}_t^{(v)} = \left(\left(\mathbf{X}_t^{(v)} \right)^T \mathbf{X}_t^{(v)} + \alpha \cdot \mathbf{I} \right)^{-1} \left(\mathbf{X}_t^{(v)} \right)^T \mathbf{X}_t^{(v)}. \tag{12}$$

The parameter $\lambda^{(v)}$ in (9) is set to zero for two reasons. First, $\mathbf{Z}_t^{(v)}$ is expected to be low-rank with respect to the vth view. From the perspective of linear representation, the collaborative representation matrix $\mathbf{Z}_t^{(v)}$ is considered to be the linear spatial transformation result under the original data $\mathbf{X}_t^{(v)}$. According to (12), $\mathbf{Z}_t^{(v)}$ is a symmetric matrix. Moreover, $\mathbf{Z}_t^{(v)}$ must be low-rank if the dimensionality of the original data is far less than the number of data objects, that is, $d_v \ll n$. One of several dimensionality reduction techniques, such as PCA, can be applied to $\mathbf{X}_t^{(v)}$ if this condition is not satisfied (i.e.,$d_v \geq n$) in practice. Second, $\mathbf{Z}_t^{(v)}$ can be obtained in a closed-form solution, which is beneficial for the improvement of the computational efficiency of data stream clustering. Consequently, (9) is relaxed to (11) by setting $\lambda^{(v)} = 0$.

Because the collaborative representation matrix $\mathbf{Z}_t^{(v)}$ is low-rank, the angular information of the principal directions of $\mathbf{Z}_t^{(v)}$ is calculated in the second stage. Let the SVD of $\mathbf{Z}_t^{(v)}$ be $\mathbf{Z}_t^{(v)} = \mathbf{U}\mathbf{\Sigma}\mathbf{V}^T$ and $\mathbf{C}_t^{(v)} = \mathbf{U}\mathbf{\Sigma}^{\frac{1}{2}}$ [27], where $\mathbf{c}_i^{(v)}$ and $\mathbf{c}_j^{(v)}$ denote the ith and jth columns of $\mathbf{C}_t^{(v)}$, respectively. Here, the specific definition the mapping function $f(\cdot)$ is defined as follows:

$$f\left(\mathbf{z}_i^{(v)}, \mathbf{z}_j^{(v)} \right) = \left(\frac{\mathbf{c}_i^{(v)} \left(\mathbf{c}_j^{(v)} \right)^T}{\left\| \mathbf{c}_i^{(v)} \right\|_2 \left\| \mathbf{c}_j^{(v)} \right\|_2} \right)^2, \tag{13}$$

Algorithm 1 The projection of vector $\mathbf{v}$ onto a simplex [16]

Input: A vector $\mathbf{v} \in \mathbb{R}^n$ and a scalar $\omega > 0$
Sort $\mathbf{v}$ into $\mathbf{q}$: $\mathbf{q}_1 \geq \mathbf{q}_2 \geq ... \geq \mathbf{q}_n$;
Search $m = \max \left\{ j : \mathbf{q}_j - \frac{1}{j} \left(\sum_{i=1}^{j} \mathbf{q}_i - \omega \right) > 0, j \in [n] \right\}$;
Define $\varphi = \frac{1}{m} \left(\sum_{i=1}^{m} \mathbf{q}_i - \omega \right)$;
Output: $\mathbf{r}$: $\mathbf{r}_i = max\left(\mathbf{v}_i - \varphi, 0\right), i \in [n]$.

where $\mathbf{z}_i^{(v)}$ and $\mathbf{z}_j^{(v)}$ denote the ith and jth columns of $\mathbf{Z}_t^{(v)}$, respectively. The angular information of the principal directions of rows is consistent with that of columns in $\mathbf{Z}_t^{(v)}$ because $\mathbf{Z}_t^{(v)}$ is symmetric. The output of the mapping function is regarded as the global affinity matrices for multiple views.

The last stage considers the sparsity of the affinity matrix, which characterizes the local structures of the features in each view. Let $h\left(\mathbf{Z}_t^{(v)}\right)$ be a matrix of size $n \times n$, where each element can be calculated using the corresponding two columns of $\mathbf{Z}_t^{(v)}$ and (13):

$$\left[h\left(\mathbf{Z}_t^{(v)}\right)\right]_{ij} = f\left(\mathbf{z}_i^{(v)}, \mathbf{z}_j^{(v)}\right). \tag{14}$$

The initial fused affinity matrix $\mathbf{Z}_t$ can be obtained by simply averaging the individual affinity matrices. Thus, $\mathbf{Z}_t$ can be calculated by

$$\mathbf{Z}_t = \frac{1}{n_v} \sum_{v=1}^{n_v} h\left(\mathbf{Z}_t^{(v)}\right). \tag{15}$$

To exploit the sparsity of $\mathbf{Z}_t$, the optimization problem is formulated as

$$\min_{\mathbf{W}_t} \left\|\mathbf{W}_t\right\|_1 + \eta \left\|\mathbf{W}_t - \mathbf{Z}_t\right\|_F^2 \quad s.t. \quad W_{ij} \geq 0,\ \left(\mathbf{W}_i\right)^T \mathbf{1} = 1, \tag{16}$$

where $\mathbf{W}_t$ is the fused sparse affinity matrix and η is a parameter. The consistency and complementary information across multiple views are explored via the individual affinity matrices in the last stage. Similarly to (10), the optimization procedure of (16) still requires iterative computations. Fortunately, the constraints $W_{ij} \geq 0$ and $\mathbf{W}_i\mathbf{1} = 1$ make the sparsity regularization $\left\|\mathbf{W}_t\right\|_1$ a constant in (16), according to the definition of the l_1-norm of a matrix. Consequently, (16) is reformulated as

$$\min_{\mathbf{W}_t} \left\|\mathbf{W}_t - \mathbf{Z}_t\right\|_F^2 \quad s.t. \quad W_{ij} \geq 0,\ \left(\mathbf{W}_i\right)^T \mathbf{1} = 1. \tag{17}$$

It is reasonable to convert (16) to (17) under the constraints, but the reverse is not necessarily true. That is, the sparsity of the solution of (17) cannot be guaranteed after the sparsity regularization $\left\|\mathbf{W}_t\right\|_1$ is eliminated. Interestingly, each $\mathbf{W}_i$ of (17) can be solved by the Euclidean projection method while guaranteeing the sparsity of the solution [16]. For completeness, the details of the Euclidean projection method are presented in Algorithm 1, whose computational complexity is $O(n \log(n))$. Because of the requirements of Algorithm 1, $\mathbf{Z}_t$ is normalized using the following equation:

$$\mathbf{Z}_t \leftarrow normalize_{(0,1]}\left(\mathbf{Z}_t\right). \tag{18}$$

Algorithm 2 summarizes the complete procedure for calculating the fused affinity matrix $\mathbf{W}_t$ for multiple views. It is easy to determine the upper bound of the computational cost of Algorithm 2 because each stage has a closed-form solution. As a result, the algorithm satisfies the requirements of data stream clustering for real-time processing and limited memory consumption in theory. Moreover, the sparsity of the fused affinity matrix $\mathbf{W}_t$ enriches the relationships between data objects by exploring the local structure of high-dimensional data objects. Calculating the fused affinity matrix $\mathbf{W}_t$ by the three consecutive stages distinguishes MVRL from existing clustering approaches, which use an intuitive combination of low-rank and sparsity regularizations.

Algorithm 2 Multi-view data stream clustering in the tth window

Input: Data matrices $\mathbf{X}_t = \left\{ \mathbf{X}_t^{(v)} \right\}_{v=1}^{n_v}$, number of clusters c, and parameter $\alpha > 0$.

1: **for** $v = 1$ to n_v **do**
2: Calculate $\mathbf{Z}_t^{(v)}$ by solving (11) using (12);
3: Calculate $f\left(\mathbf{z}_i^{(v)}, \mathbf{z}_j^{(v)}\right)$ for each pair of data objects using (13);
4: **end for**
5: Construct $\mathbf{Z}_t$ using (15);
6: $\mathbf{Z}_t \leftarrow normalize_{(0,1]}\left(\mathbf{Z}_t\right)$;
7: Calculate $\mathbf{W}_t$ by solving (17) using Algorithm 1 with $\omega = 1$;
8: Apply $\mathbf{W}_t$ to perform NCuts.

Output:
The c clustering and $\mathbf{Z}_t^{(v)}$ $\left(v \in \left(1, 2, ...n_v\right)\right)$.

3.2. Adaptively updating the representative data objects in a dynamic set

Data objects in data streams may evolve. This means that the underlying structures of the data objects may evolve and change substantially over time. The self-expressiveness capability of the collaborative representation could be improved by adaptively updating the representative data objects in a dynamic set. In MVRL, the maximum size of the dynamic set is the same as that of the window. We illustrate two technical aspects of the importance of the dynamic set in multi-view data stream clustering: collaborative representation and the estimation of the evolving number of clusters:

1. The current data objects of the window are used to form the collaborative representation in addition to the representative data objects in the dynamic set; this approach is beneficial for capturing the global structures of data streams. For example, a valid data object can be represented well by the representative data objects even if it occurs in the window alone. Similarly, the representative data objects can help to achieve a compact representation of a few arriving data objects that belong to the same class. Moreover, new representative data objects are continuously added to the dynamic set while the outdated data objects are discarded from it. In this manner, the dynamic set shares the previously learned knowledge with the processing of the subsequent windows by incrementally improving the quality of the collaborative representation.

2. Determining an accurate number of clusters is a challenging problem even for a stationary dataset, and a fixed number of clusters is not able to effectively capture the dynamic evolution of data objects over time. We assume that the maximum number of dynamic clusters in the data streams is known. A dynamic set of representative data objects will eventually cover all potential clusters over time, and it can be employed to help to estimate the evolving number of clusters over time.

We next present the adaptive strategy to update a dynamic set of representative data objects in detail. Let be the dynamic set at the $(t-1)$th window. First, the dynamic set is initialized to null when Algorithm 2 is used to cluster the data objects in the first window. Second, all data objects in the first window are used to initialize the dynamic set, whose representative data objects are used for the next window. Let $\mathbf{X}_u$ be a matrix, whose columns consist of two parts: the data objects $\mathbf{X}_2$ in the second window and the representative data objects in the dynamic set $\mathcal{D}_1$. Third, the multi-view clustering of $\mathbf{X}_u$ is continued using Algorithm 2. Each data object in $\mathbf{X}_u$ has its own label according to the clustering results. Assume that $\mathbf{X}_u$ has c clusters. The computed partition is $\mathbf{X}_u = \left[\mathbf{X}_{u_1}, \mathbf{X}_{u_2}, ..., \mathbf{X}_{u_c}\right]$, which is divided by the respective labels of the data objects. Finally, the dynamic set $\mathcal{D}_2$ is adaptively updated using the divide and conquer strategy.

Reconstruction residuals of all data objects are calculated individually in each cluster. These residuals are employed to evaluate the importance of each data object to the collaborative representation in the corresponding cluster. For the collaborative representation in (11), the sum of the reconstruction residuals of multiple views of a data object $\mathbf{x}_t^i$ is

Algorithm 3 Adaptive update of the representative data objects in a dynamic set

Input: Data matrices $\mathbf{X}_u = \left[\mathbf{X}_{u_1}, \mathbf{X}_{u_2}, ..., \mathbf{X}_{u_c}\right]$ and the number of representative data objects n.

Initialize: Dynamic set $\mathcal{D} = \{\}$.

```
l = ⌊(n/c)⌋
for k = 1 to c do
    for each data object x^i in X_{u_c} do
        Calculate a reconstruction error of x^i: g(x^i) using (19);
    end for
    Add the top l data objects with the highest reconstruction errors to D;
end for
```

Output:

Dynamic set $\mathcal{D}$.

defined as follows:

$$g\left(\mathbf{x}_t^i\right) = \sum_{v=1}^{n_v} \left\| \mathbf{X}_t^{(v)'} - \mathbf{X}_t^{(v)'} \mathbf{Z}_t^{(v)'} \right\|_F^2, \tag{19}$$

where $\mathbf{X}_t^{(v)'}$ denotes $\mathbf{X}_t^{(v)}$ with its ith column replaced with zeros and $\mathbf{Z}_t^{(v)'}$ denotes $\mathbf{Z}_t^{(v)}$ with its ith row replaced with zeros. A higher value of $f\left(\mathbf{x}_t^i\right)$ indicates an $\mathbf{x}_t^i$ that is more important in the reconstruction of the collaborative representation. The top l data objects with the highest number of reconstruction errors are selected as the representative data objects of cluster $\mathbf{X}_{u_i}$ $(1 \leq i \leq c)$. Thus, $\mathcal{D}_2$ consists of not more than $l \times c$ representative data objects chosen from all clusters. The complete procedure for adaptively updating the representative data objects is outlined in Algorithm 3.

3.3. MVRL method

Dynamic set $\mathcal{D}$ is employed in the multi-view clustering of data streams. However, the proposed method needs to predict the number of clusters dynamically before $\mathcal{D}$ covers all potential clusters over time. For completeness , we estimate the number of clusters dynamically using the normalized Laplacian matrix $\mathbf{L}$ of $\mathbf{Z}$:

$$\mathbf{L} = \mathbf{I} - \mathbf{Q}^{-\frac{1}{2}} \mathbf{Z} \mathbf{Q}^{-\frac{1}{2}}, \tag{20}$$

where $\mathbf{I} \in \mathbb{R}^{n \times n}$ is the identity matrix and $\mathbf{Q}$ is a diagonal matrix whose diagonal entries are defined as $q_i = \sum_{j=1}^{n} z_{ij}$. Let $\left\{\lambda_i\right\}_{i=1}^{n}$ be the singular values of the Laplacian matrix $\mathbf{L}$. The singular values are sorted in ascending order: $0 \leq \lambda_1 \leq \lambda_2 \leq ... \leq \lambda_n$. The current number of clusters c is estimated by

$$c = n - \rho, \tag{21}$$

where $\rho = \max\left\{i : abs\left(\lambda_i - \lambda_{(i+1)}\right), i \in [1, n-1]\right\}$ [30].

Outlier detection is one of the most important aspects of data stream clustering. Potential outliers are data objects that deviate from the normal data distribution in the window. Ideally, an outlier detection mechanism is able to distinguish between potential outliers and cluster evolution. For a given data object , we consider the reconstruction residuals of multiple views on the data objects $\mathbf{X}_t^{(v)}$ that are from the same cluster k as follows:

$$h\left(\mathbf{x}_t^i\right) = \sum_{v=1}^{n_v} \left\| \mathbf{x}_t^i - \mathbf{X}_t^{(v)} \mathbf{z}_t^i \right\|_F^2, \tag{22}$$

where $\mathbf{z}_t^i$ is a coefficient vector in $\mathbf{Z}_t^{(v)}$ corresponding to $\mathbf{x}_t^i$.

Concept drift causes the subspace structures of high-dimensional data objects to evolve over time. Several specific types of subspace evolution in data streams are handled by the proposed method: subspace emergence, disappearance,

Algorithm 4 MVRL algorithm

Input: Data matrices $\mathbf{X}_t = \left\{\mathbf{X}_t^{(v)}\right\}_{v=1}^{n_v}$, parameters $\alpha > 0$ and $\sigma > 0$, and maximum number of clusters c_{max}.

Initialize: Dynamic set $\mathcal{D} = \{\}$ and $c = 0$.

```
if t == 1 then
    X = X_1 and add each data object of X to D;
else
    X = [X_t, X_t'], where the columns of X_t' correspond to the elements of D;
end if
for each x_i ∈ X do
    Compute h(x_t^i) using (22);
    if h(x_t^i) < σ then
        remove x_i from X because it is regarded as an outlier;
    end if
end for
if c < c_max then
    update c by (21);
else
    c = c_max;
end if
Perform multi-view data stream clustering on X using Algorithm 2 to obtain c clusters and Z_t^(v) (v ∈ (1, 2, ...n_v));
X_u = [X_u1, X_u2, ..., X_uc] consists of the c clusters;
Update the dynamic set D_t (t > 1) using Algorithm 3;
```

Output:

The c clusters and $\mathcal{D}_t$.

and recurrence. The changes in the representative data objects of the dynamic set can be detected according to the clustering results of Algorithm 4. Hence, changes in the representative data objects of the clusters reflect the subspace evolution in data streams. For example, subspace emergence is recognized when a completely new cluster appears and the number of clusters increases in the dynamic set. Similarly, a previous subspace is considered to have disappeared if the representative data objects of the previous cluster remain unchanged during several successive windows.

Algorithm 4 summarizes the complete procedure of the proposed method. In particular, single-view data stream clustering (i.e., $n_v = 1$) is a special case of multi-view data stream clustering in Algorithm 4. This enables the proposed method to perform data stream clustering with any number of views. It is worth noting that the use of the dynamic set inevitably increases the computational cost. However, the upper bound of the computational cost is still definite, even with the dynamic set. We believe that the extra computational cost will become less important with continuing advances in computer hardware.

3.4. Complexity analysis

The first stage of Algorithm 2 involves matrix inversion and matrix multiplication in (12); its computational complexity is $\mathcal{O}\left(d^3\right)$. The second stage of Algorithm 2 requires the computation of SVD and the multiplication of each pair of vectors in (13); its computational complexity is $\mathcal{O}\left(n^3\right)$. The computational complexity of the last stage of Algorithm 2 is $\mathcal{O}\left(n^2 \log(n)\right)$. In addition, the computational complexity of NCuts is $\mathcal{O}\left(n^3\right)$ in Algorithm 2. Hence, the computational complexity of Algorithm 2 is $\mathcal{O}\left(n^3 + d^3\right)$. Moreover, the computational complexity of Algorithm 3 is $\mathcal{O}\left(cd^2n^2\right)$, where c denotes the number of clusters in the window. Therefore, the overall complexity of Algorithm 4 is $\mathcal{O}\left(cd^2n^2 + d^3 + n^3\right)$ in each window. In particular, the complexity of Algorithm 4 is $\mathcal{O}\left(n^3\right)$ if $c \ll n$ and $d \ll n$.

4. Experiments

In this section, we report experiments to evaluate the effectiveness and efficiency of MVRL on benchmark datasets. The source code of MVRL, which is implemented in MATLAB 2019b, is available online[1]. The experiments were conducted on a Windows 10 platform with an Intel i7-10700 CPU and 32 GB of RAM.

[1] https://github.com/chenjie20/MVRL

Table 2
Statistics of the five data stream datasets.

Dataset	Classes	Data objects	Views	Features
Forest Cover	7	580,000	1	54
MNIST	10	70,000	1	784
Network Intrusion	41	4,898,431	1	23
Reuters	6	18,758	5	115, 155, 215, 248, 342
Handwritten	10	8, 677	6	6, 47, 64, 76, 216, 240

4.1. Experimental settings

4.1.1. Datasets

Five benchmark datasets were used in the experiments. The statistics of the datasets are summarized in Table 2. The first three datasets contain a single view, whereas the other two datasets consist of multiple views.

- **Forest Cover Dataset** [15]. This dataset consists of 54 cartographic variables. The last variables of the dataset represent the truth label of the corresponding data object.
- **MNIST Dataset** [23]. This dataset contains 70,000 images of ten handwritten digits (09), with 7,000 images of each digit. The grayscale images are 28×28 pixels in size.
- **Network Intrusion Dataset** [15]. This dataset contains 4,898,431 records of network traffic data that belong to 23 types of connection. Each instance is described by 41 features.
- **Reuters Dataset** [15]. This dataset consists of 18,758 documents written in five languages and their translations over a common set of six categories.
- **Handwritten Dataset** [15]. This dataset contains 2,000 images of handwritten digits (09) with six views. The dataset was downloaded from the UCI repository.

4.1.2. Comparison methods

We compared MVRL with three single-view data stream clustering algorithmsClusTree [21], MBSCAN [37], and ESA-Stream [25]and three multi-view clustering algorithmsthe online multi-view clustering (OMVC) algorithm [38], LRSSC [7], and the self-representation subspace clustering (SRSC) algorithm [28]. The implementation of ClusTree was provided by massive online analysis, which is a widely used open-source tool for data stream mining [4]. The source code of the other five algorithmsMBSCAN, ESA-Stream, OMVC, LRSSC, and SRSCwas provided by their respective authors. For the multi-view datasets, we ran the single-view data stream clustering algorithms on each individual view and chose the best clustering result for comparison. For the multi-view clustering algorithms, we estimated the number of clusters using (21).

4.1.3. Evaluation metrics

To measure the clustering quality of all competing algorithms, we employed three metrics: the clustering purity, normalized mutual information (NMI), and F-measure. We calculated the mean and standard definition for each metric [32]. The clustering purity represents how often each cluster is assigned to the class that appears most frequently within the cluster. Given two sets $\mathcal{A}$ and $\mathcal{B}$ of $\mathbf{X}$, $\mathcal{A} = \{a_1, a_2, ..., a_p\}$ is the set of clusters and $\mathcal{B} = \{b_1, b_2, ..., b_q\}$ is the set of classes, where p and q denote the numbers of clusters and classes, respectively. The purity of $\mathcal{A}$ and $\mathcal{B}$ is defined as follows:

$$Purity\left(\mathcal{A}, \mathcal{B}\right) = \frac{1}{n} \sum_{i} \max_{j} \left| a_i \cap b_j \right|, \tag{23}$$

where n denotes the number of data objects in $\mathbf{X}$. The NMI of $\mathcal{A}$ and $\mathcal{B}$ is defined as follows:

$$NMI\left(\mathcal{A}, \mathcal{B}\right) = \frac{I\left(\mathcal{A}, \mathcal{B}\right)}{\sqrt{H\left(\mathcal{A}\right) H\left(\mathcal{B}\right)}}, \tag{24}$$

where $I\left(\cdot, \cdot\right)$ is the mutual information metric and $H\left(\cdot\right)$ is the entropy metric. The precision of the clusters is the number of true positive results divided by the number of all positive results, and the recall is the number of true positive results divided by the number of all samples that should have been identified as positive. The F-measure is the harmonic mean of the values of precision and recall of the clusters [10, 32].

Table 3
Clustering results – mean and standard deviation (%) – over the given number of windows on the five datasets.

Datasets	Metrics (%)	MVRL	ClusTree	MBSCAN	ESA-Stream	LRSSC	SRSC	OMVC
Forest Cover	Purity	**77.12 (16.69)**	67 (17.24)-	67.85 (16.06)-	70.05 (16.54)-	73.84 (18.16)-	73.52 (18.58)-	68.39 (17.46)-
	NMI	**26.31 (11.23)**	21 (13.23)-	16.15 (11.93)-	17.47 (11.85)-	22.38 (12.2)-	21.5 (11.96)-	15.69 (11.43)-
	F-measure	**56.49 (10.14)**	39 (12.53)-	40.05 (12.05)-	42.85 (10.11)-	43.21 (10.76)-	42.5 (10.17)-	40.3 (16.09)-
MNIST	Purity	**74.82 (2.62)**	60 (7.31)-	67.9 (5.15)-	64.08 (3.38)-	57.82 (3.55)-	61.78 (4.37)-	42.38 (5.18)-
	NMI	**72.34 (2.54)**	26 (9.71)-	27.03 (9.27)-	18.19 (6.12)-	49.83 (3.19)-	54.89 (3.75)-	33.15 (5.76)-
	F-measure	**73.85 (2.84)**	51 (5.4)-	68.1 (6.22)-	57.89 (4.29)-	56.47 (3.62)-	60.6 (4.42)-	42.12 (5.6)-
Network Intrusion	Purity	**97.96 (0.49)**	66.23 (8.99)-	95.73 (1.4)-	95.88 (2.86)-	97.21 (0.76) ≈	97.05 (0.74)-	91.05 (6.01)-
	NMI	**67.41 (1.23)**	27.14 (7.68)-	55.49 (1.57)-	55.77 (3.42)-	56.54 (1.65)-	57.06 (1.38)-	56.28 (8.35)-
	F-measure	**73.11 (2.06)**	26.17 (10.04)-	58.54 (3.82)-	58.19 (2.8)-	59.73 (4.3)-	60.76 (2.19)-	57.77 (10.46)-
Reuters	Purity	**51.36 (4.12)**	48 (5.05)-	49.08 (4.23)-	47.19 (4.48)-	49.31 (4.47)-	49.58 (4.84)-	47.8 (4.56)-
	NMI	**29.26 (4.34)**	19 (4.56)-	18.29 (4.71)-	21.14 (4.61)-	24.01 (4.85)-	26.86 (4.75)-	20.24 (4.52)-
	F-measure	**44.68 (2.99)**	40 (4.1)-	39.29 (3.59)-	41.27 (3.49)-	43.03 (4.52)-	43.3 (5.5)-	42.07 (4.07)-
Handwritten	Purity	**90.9 (3.63)**	75 (5.09)-	44.4 (4.27)-	68.95 (4.54)-	70.5 (4.21)-	75.05 (3.98)-	70.15 (5.89)-
	NMI	**87.1 (3.73)**	73 (4.79)-	50.64 (4.8)-	64.72 (3.9)-	69.36 (4.01)-	73.22 (4.23)-	67.62 (6.23)-
	F-measure	**90.94 (3.66)**	75 (5.15-)	46.35 (3.99)-	67.61 (4.23)-	71.96 (4.12)-	76.2 (4.5)-	70.38 (5.59)-

Table 4
Average computational costs of a window (seconds) on the five datasets.

Databases	MVRL	ClusTree	MBSCAN	ESA-Stream	LRSSC	SRSC	OMVC
Forest Cover	1.62	**0.35**	2.56	1.69	1.81	1.71	1.97
MNIST	1.49	**0.68**	2.83	1.74	1.06	5.56	2.08
Network Intrusion	1.29	**0.16**	1.51	1.47	3.59	17.16	2.64
Reuters	0.61	**0.17**	0.29	0.19	0.49	1.3	0.91
Handwritten	0.08	**0.02**	0.05	0.03	0.09	0.95	1.86

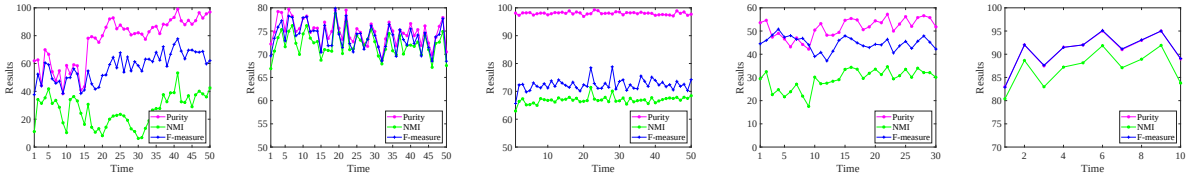
4.1.4. Parameter settings

For a fair comparison, we adopted the parameter settings of the baseline algorithms that were reported in their respective papers and manually adjusted the parameters to obtain the best results. The MVRL method has two parameters, $\alpha > 0$ and $\sigma > 0$, which are mutually independent. The parameter α was initially chosen from $\left[10^{-3}, 10^{-2}, 5^{-2}, 0.1, 0.5, 1, 2, 5, 10, 50, 500\right]$ without considering outlier detection, and we recorded the best result on each dataset. In the tables , the best and second-best clustering results are highlighted in bold and underlined text, respectively.

4.2. Clustering quality evaluation

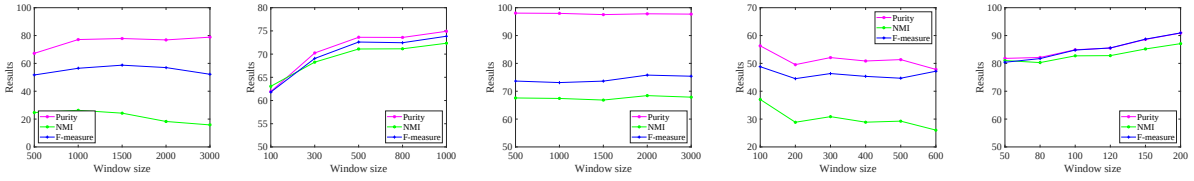
The Forest Cover, MNIST, and Network Intrusion datasets (listed in Table 2) contain only one view, whereas the Reuters and Handwritten datasets consist of multiple views. Because we use a landmark window-based model on the data stream datasets, the numbers of windows and the window sizes of the Forest Cover, MNIST, Network Intrusion, Reuters, and Handwritten datasets were set to (1) 50, 1000, (2) 50, 1000, (3) 50, 1000, (4) 30, 500, and (5) 10, 200, respectively. We used a PCA algorithm to preprocess the original features of the data objects. Specifically, the dimensionality of the original features of the data objects was reduced to 30, 50, 30, 10, and 30 for the Forest Cover, MNIST, Network Intrusion, Reuters, and Handwritten datasets, respectively, if the dimensionality of the original features was greater than the reduced size in each view. This can be regarded as a preprocessing step in MVRL. We then normalized the features obtained by dimensionality reduction in the multi-view data. The parameter settings of MVRL were (1) $\alpha = 10^{-3}$, (2) $\alpha = 2$, (3) $\alpha = 500$, (4) $\alpha = 0.05$, and (5) $\alpha = 10$ for the Forest Cover, MNIST, Network Intrusion, Reuters, and Handwritten datasets, respectively. The parameters of MVRL remained unchanged for all windows.

Table 3 shows the clustering results (means and standard deviations) of the compared clustering methods on the five datasets. The symbols "+", "-", and "≈" indicate that the mean clustering results of the other competing methods were significantly better than, significantly worse than, and comparable with those of MVRL, respectively. MVRL performed better than all competing methods in terms of the three metrics (i.e., purity, NMI, and F-measure). For example, the average purity of MVRL was higher than that of the second-best method by 3.28%, 6.92%, 0.75%, 1.78%, and 9.35% on the Forest Cover, MNIST, Network Intrusion, Reuters, and Handwritten datasets, respectively. Similar improvements were observed in terms of the other two metrics used for measuring clustering performance. Moreover,



(a) Forest Cover (b) MNIST (c) Network Intrusion (d) Reuters (e) Handwritten

Figure 1: Online clustering results varying with the given number of the windows on the five datasets.



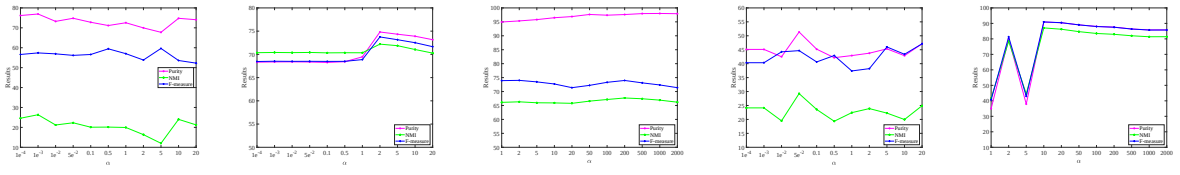
(a) Forest Cover (b) MNIST (c) Network Intrusion (d) Reuters (e) Handwritten

Figure 2: Average clustering results varying with window size on the five datasets.

the standard deviations of the clustering results over the given number of windows indicate that the clustering results of MVRL remained stable on all datasets except the Forest Cover dataset. These results verify the effectiveness of the proposed method. In addition, the multi-view clustering methods, which consider the subspace structures in high-dimensional data, consistently outperformed the other comparison methods. For instance, the average clustering results of MVRL, LRRSC, and SRSC were substantially higher than those of the other competing methods. This demonstrates the advantages of exploiting subspace structures in high-dimensional data. Moreover, MVRL consistently achieved better clustering results than LRRSC and SRSC. This indicates that MVRL effectively improves clustering performance by capturing the local and global structures of high-dimensional data.

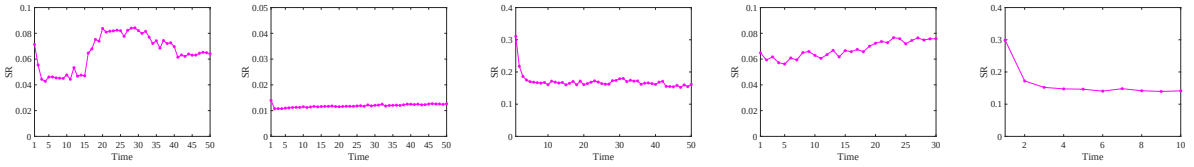
The average computational costs of a landmark window for all algorithms are shown in Table 4. ClusTree performed more efficiently than the other methods. It uses the Euclidean distance metric for measuring the relationship between data objects. In addition, the proposed method performed better than all other algorithms except for ClusTree on the single-view (i.e., Forest Cover, MNIST, and Network Intrusion) datasets. The single-view data stream clustering algorithms have a lower computational cost than the multi-view clustering algorithms on the multi-view (i.e., Reuters and Handwritten) datasets. This is because the multi-view clustering algorithms process the multiple views simultaneously. Our method exhibits the same advantages with respect to performance when compared with the other multi-view clustering methods on the multi-view datasets. These results show that the computational efficiency of the proposed method is comparable with all the comparison algorithms except for ClusTree. This is mainly because the proposed method calculates an SVD of the collaborative representation matrix in Algorithm 2, which is a time-consuming operation. In particular, this does not affect the determination of the upper bound of the computational cost of the proposed method.

Fig. 1 shows how the three metrics (purity, NMI, and F-measure) varied with respect to the number of landmark windows on the five datasets. For the Forest Cover dataset, the clustering results fluctuated wildly in the first 15 landmark windows, and then increased quickly. This is why the standard deviations of the clustering results are relatively large on the Forest Cover dataset. Similarly, the clustering results increased and then remained relatively stable after the second landmark window on the Handwritten dataset. Conversely , the clustering metrics showed small fluctuations over time on the other three datasets. MVRL always achieved relatively stable clustering performance after several landmark windows in the experiments. These improvements and stability in the clustering performance were achieved for two reasons. First, the number of clusters can be accurately estimated (i.e., as the maximum number of clusters) after several landmark windows. Second, the representative data objects that are used in the collaborative representation effectively improve the clustering performance.



(a) Forest Cover (b) MNIST (c) Network Intrusion (d) Reuters (e) Handwritten

Figure 3: Average clustering results varying with α on the five datasets.



(a) Forest Cover (b) MNIST (c) Network Intrusion (d) Reuters (e) Handwritten

Figure 4: Changes in the SR in different windows.

4.3. Window size scalability

We next report the results of experiments to investigate the influence of window size in a landmark window-based model. We adopted the same parameter settings as those used in Section 4.2; the clustering results are shown in Fig. 2. An increase in window size usually enhances the capability of collaborative representation. However, the accuracy of the clustering results may be sensitive to the number of data objects in a window. Fig. 2 shows that the clustering results increased slowly as the window size increased for all five datasets. In addition, the clustering results fluctuated slightly when the window size exceeded 100 for the Reuters dataset. The above observations imply that the clustering performance of the proposed method is stable for a large range of fixed window sizes.

4.4. Parameter sensitivity analysis

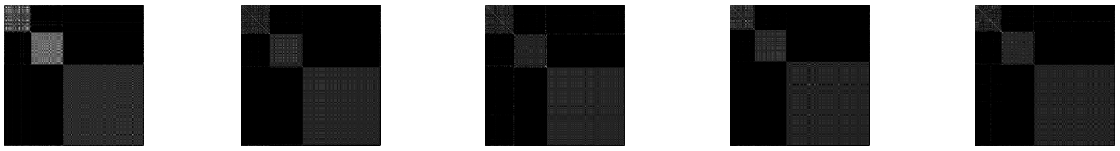
In this section, we investigate the sensitivity of parameter α on the clustering performance of MVRL without considering outlier detection. Parameter α was varied in the range $[1, 2, 5, 10, 20, 50, 100, 200, 500, 1,000, 2,000]$ for the Network Intrusion and Handwritten datasets and in the range $[10^{-4}, 10^{-3}, 10^{-2}, 0.05, 0.1, 0.5, 1, 2, 5, 10, 20]$ for the other datasets. Fig. 3 shows the influence of parameter α on the average clustering results on the five datasets. Good clustering performance was achieved for different values of parameter α on different datasets. Hence, it is still an open problem to determine an appropriate value of parameter α with no prior knowledge of the data distribution. However, Fig. 3 shows that the proposed method achieved satisfactory clustering performance over a relatively large range of α. For example, it performed stably on the MNIST and Handwritten datasets with $\alpha \in [2, 20]$ and $\alpha \in [10, 2,000]$, respectively. In addition, the average clustering results of the proposed method fluctuated slightly over the given range of α on the other three datasets.

4.5. Sparsity stability analysis

The sparsity of the fused affinity matrix $\mathbf{W}_t$ is crucial for capturing the local structures of high-dimensional data objects in Algorithm 2. The sparsity ratio (SR) of $\mathbf{W}_t$ is defined as follows:

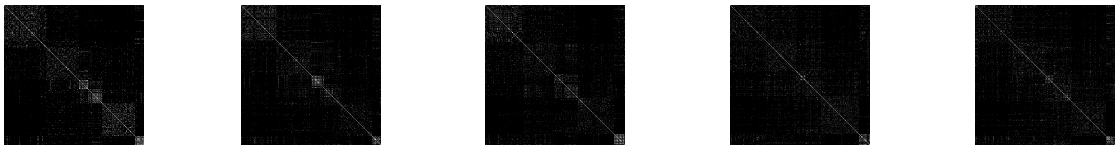
$$SR\left(\mathbf{W}_t\right) = \frac{\left\|\mathbf{W}_t\right\|_0}{size\left(\mathbf{W}_t\right)}, \tag{25}$$

where $size\left(\mathbf{W}_t\right)$ denotes the number of elements of $\mathbf{W}_t$. We conducted experiments to evaluate whether the fused affinity matrix obtained by the proposed method usually remains sparse in different windows. Specifically, we calculated the SR in different windows corresponding to the online clustering experiments reported in Section 4.2. Fig. 4 shows the changes in the SR over the landmark windows of the five datasets. First, the fused affinity matrix was sparse for all the windows of the five datasets. Moreover, the SR of the fused affinity matrix usually remained stable



(a) Window 1 (b) Window 10 (c) Window 20 (d) Window 30 (e) Window 40

Figure 5: Fused affinity matrix produced by MVRL on the Network Intrusion dataset in different windows.



(a) Window 1 (b) Window 5 (c) Window 10 (d) Window 20 (e) Window 30

Figure 6: Fused affinity matrix produced by MVRL on the Reuters dataset in different windows.

over successive windows. This indicates that the proposed method shows robust stability, in terms of the sparsity of the fused affinity matrix, over time.

Finally, we present some intuitive examples of fused affinity matrices produced by MVRL on the Network Intrusion and Reuters datasets. The fused affinity matrices in various landmark windows are illustrated in Figs. 5 and 6. All entries of the matrices are arranged in order of their ground-truth labels. Each fused affinity matrix reveals a distinct block-diagonal structure. For example, Fig. 5(a) depicts the fused affinity matrix in the first window on the Network Intrusion dataset. According to the ground-truth labels of the data objects, there are three main clusters in the first window. The three clusters comprise 18.50%, 22.50%, and 57.20% of the data objects. The three corresponding distinct block-diagonal structures are clearly visible in Fig. 5(a). Similarly, we can observe several distinct block-diagonal structures in Fig. 6. These examples intuitively show that low-dimensional structures of high-dimensional data objects are effectively explored by MVRL.

4.6. Discussion

Finding the low-dimensional structures of high-dimensional data objects is critical to evaluating the memberships of data objects. Therefore, the Euclidean distance metric may be unsuitable for measuring the relationships between high-dimensional data objects in data stream clustering methods, such as ClusTree, MBSCAN, and ESA-Stream. General clustering algorithms, such as LRSSC and SRSC, employ sparsity, low-rank regularizations, or an intuitive combination of them to seek low-dimensional structures of high-dimensional data objects. However, in previous studies, the exploitation of the global and local structures of high-dimensional data objects was insufficient or ambiguous. In contrast, MVRL effectively exploits the intrinsic low-dimensional structures of high-dimensional data objects preserved in the final fused sparse affinity matrix. Consequently, MVRL performs better than existing algorithms for data stream clustering.

General clustering algorithms often require iterative computations before convergence when solving their respective optimization problems. Such computations may incur a high computational cost when the number of iterations is large. Computational cost may be of secondary importance when clustering is performed on stationary datasets. However, one of the important requirements of clustering data streams is to continuously cluster objects in real time. Each stage of MVRL has an individual closed-form solution. In general, the upper bound of its computational cost is determined by closed-form solutions. Compared with the competing subspace clustering algorithms, whose complexity is $\mathcal{O}(tn^3)$, the complexity of MVRL is $\mathcal{O}(n^3)$, where t represents the number of iterations. This explains why MVRL incurred a relatively low computational cost in the experiments.

5. Conclusion

In this paper, we proposed the MVRL algorithm for the multi-view clustering of data streams. MVRL makes full use of the consistency information and complementary information across multiple views. It successively exploits the global and local structures of high-dimensional data objects in data streams using three consecutive stages: collaborative representation, construction of individual global affinity matrices using the mapping function, and calculation of the fused sparse affinity matrix using Euclidean projection. The global structures of high-dimensional data objects are captured by the first two stages, where the individual global affinity matrices contain the consistency information across multiple views. The local structures of high-dimensional data objects are exploited by the third stage, where the fused sparse affinity matrix incorporates the complementary information across multiple views. Because each stage has a closed-form solution, it is easy to determine the upper bound of the computational cost and memory consumption. Consequently, the requirements of data stream clustering for real-time processing and limited memory consumption are satisfied. In addition, MVRL shows robust stability in sparsity in successive windows, thereby guaranteeing stable clustering performance. Furthermore, the dynamic set, which stores the representative data objects over time, can be adaptively updated using the construction residuals of the collaborative representation. This efficiently enables previously learned useful knowledge to be used on the arriving data objects in MVRL, thereby improving the clustering performance in subsequent windows. Moreover, changes in the dynamic set can be employed to detect concept drift. Extensive experiments on five benchmark datasets demonstrated the efficiency and effectiveness of the proposed method and the stability of its clustering performance over time.

Adaptively choosing an optimal value for parameter α in MVRL is still an open challenge, which we intend to address in future work. In addition, we intend to produce an accurate estimate of the number of current clusters before the maximum number of clusters is reached. Such an estimate should lead to significant improvements in data stream clustering.

References

[1] Ackermann, M.R., Märtens, M., Raupach, C., Lammersen, C., Sohler, C., 2012. StreamKM++: A clustering algorithm for data streams. J. Exp. Algorithmics 17, 1–30.

[2] Aggarwal, C.C., Han, J., Wang, J., Yu, P.S., 2003. A framework for clustering evolving data streams, in: in Proc. 29th Int. Conf. Very Large Data Bases, Berlin, Germany. pp. 81–92.

[3] Bagozi, A., Bianchini, D., Antonellis, V.D., 2021. Multi-level and relevance-based parallel clustering of massive data streams in smart manufacturing. Inf. Sci. 577, 805–823.

[4] Bifet, A., Holmes, G., Kirkby, R., Pfahringer, B., 2010. MOA: massive online analysis. J. Mach. Learn. Res. 11, 1601–1604.

[5] Borlea, I.D., Precup, R.E., Borlea, A.B., Iercan, D., 2021. A unified form of fuzzy c-means and k-means algorithms and its partitional implementation. Knowl.-Based Syst. 214, 106731.

[6] Boyd, S., Parikh, N., Chu, E., Peleato, B., Eckstein, J., 2011. Distributed optimization and statistical learning via the alternating direction method of multipliers. Found. Trends Mach. Learn. 3, 1–122.

[7] Brbić, M., Kopriva, I., 2020. l_0-motivated low-rank sparse subspace clustering. IEEE Trans. Cybern. 50, 1711–1725.

[8] Cao, F., Estert, M., Qian, W., Zhou, A., 2006. Density-based clustering over an evolving data stream with noise, in: Proc. SIAM Int. Conf. Data Min., Bethesda, MD, USA. pp. 328–339.

[9] Chen, J., Mao, H., Sang, Y., Yi, Z., 2017. Subspace clustering using a symmetric low-rank representation. Knowledge-Based Systems 127, 46–57.

[10] Chen, J., Mao, H., Wang, Z., Zhang, X., 2021a. Low-rank representation with adaptive dictionary learning for subspace clustering. Knowl.-Based Syst. 223, 107053. doi:10.1016/j.knosys.2021.107053.

[11] Chen, J., Yang, S., Mao, H., Fahy, C., 2021b. Multiview subspace clustering using low-rank representation. IEEE Trans. Cybern. , 1–15doi:10.1109/TCYB.2021.3087114.

[12] Chen, J., Yang, S., Wang, Z., Mao, H., 2021c. Efficient sparse representation for learning in high-dimensional data. IEEE Trans. Neural Netw. Learn. Syst. , 1–15doi:10.1109/TNNLS.2021.3119278.

[13] Degirmenci, A., Karal, O., 2022. Efficient density and cluster based incremental outlier detection in data streams. Inf. Sci. 607, 901–920.

[14] Donoho, D.L., 2006. For most large underdetermined systems of linear equations the minimal $l_1$norm solution is also the sparsest solution. Commun. Pure Appl. Math. 59, 797–829.

[15] Dua, D., Graff, C., 2013. UCI machine learning repository. http://archive.ics.uci.edu/ml.

[16] Duchi, J., Shalev-Shwartz, S., Singer, Y., Chandra, T., 2008. Efficient projections onto the l_1-ball for learning in high dimensions, in: Proc. 25th Int. Conf. Mach. Learn. (ICML), Helsinki, Finland. pp. 272–279.

[17] Elhamifar, E., Vidal, R., 2013. Sparse subspace clustering algorithm, theory, and applications. IEEE Trans. Pattern Anal. Mach. Intell. 35, 2765–2781.

[18] Hajjar, S.E., Dornaika, F., Abdallah, F., 2022. One-step multi-view spectral clustering with cluster label correlation graph. Inf. Sci. 592, 97–111.

[19] Huang, L., Wang, C.D., Chao, H.Y., Yu, P.S., 2020. Mvstream: Multiview data stream clustering. IEEE Trans. Neural Netw. Learn. Syst. 31, 3482–3496.
[20] Huang, R., Xiao, R., W, W.Z., Gong, P., Chen, J., Rida, I., 2021. Towards an efficient real-time kernel function stream clustering method via shared nearest-neighbor density for the IIoT. Inf. Sci. 566, 364–378.
[21] Kranen, P., Assent, I., Baldauf, C., Seidl, T., 2011. The clustree: indexing micro-clusters for anytime stream mining. Knowl. Inf. Syst. 29, 249–272.
[22] Laohakiat, S., Sa-ing, V., 2021. An incremental density-based clustering framework using fuzzy local clustering. Inf. Sci. 547, 404–426.
[23] LeCun, Y., Bottou, L., Bengio, Y., Haffner, P., 1998. Gradient-based learning applied to document recognition. in Proc. IEEE 86, 22782324.
[24] Lee, H., Battle, A., Raina, R., Ng, A.Y., 2007. Efficient sparse coding algorithms, in: Adv. Neural. Inf. Process. Syst., Vancouver, British Columbia, Canada. pp. 801–808.
[25] Li, Y., Li, H., Wang, Z., Liu, B., Cui, J., Fei, H., 2022a. ESA-stream: Efficient self-adaptive online data stream clustering. IEEE Trans. Knowl. Data Eng. 34, 617–630.
[26] Li, Y., Yang, M., Peng, D., Li, T., Huang, J., Peng, X., 2022b. Twin contrastive learning for online clustering. Int. J. Comput. Vis. 130, 22052221.
[27] Liu, G., Lin, Z., Yan, S., Sun, J., Yu, Y., Ma, Y., 2013. Robust recovery of subspace structures by low-rank representation. IEEE Trans. Pattern Anal. Mach. Intell. 35, 171–184.
[28] Liu, J., Liu, X., Zhang, Y., Zhang, P., W.Tu, Wang, S., Zhou, S., Liang, W., Wang, S., Yang, Y., 2021. Self-representation subspace clustering for incomplete multi-view data, in: ACM Int. Conf. Multimedia, Chengdu, China. pp. 1–11.
[29] Lughofer, E., Sayed-Mouchaweh, M., 2015. Autonomous data stream clustering implementing split-and-merge conceptstowards a plug-and-play approach. Inf. Sci. 304, 54–79.
[30] Luxburg, U.V., 2007. A tutorial on spectral clustering. Stat. Comput. 17, 395–416.
[31] Ma, X., Yan, X., Liu, J., Zhong, G., 2022. Simultaneous multi-graph learning and clustering for multiview data. Inf. Sci. 593, 472–487.
[32] Manning, C.D., Raghavan, P., Sch´utze, H., 2008. Introduction to Information Retrieval. Cambridge University Press, New York, NY, USA.
[33] Nguyen, H.L., Woon, Y.K., Ng, W.K., 2015. A survey on data stream clustering and classification. Knowl. Inf. Syst. 45, 535–569.
[34] Otero, A., Félix, P., Márquez, D.G., García, C.A., Caffarena, G., 2022. A fault-tolerant clustering algorithm for processing data from multiple streams. Inf. Sci. 584, 649–664.
[35] Pehlivan, N.Y., Turksen, I.B., 2021. A novel multiplicative fuzzy regression function with a multiplicative fuzzy clustering algorithm. Romanian J. Inf. Sci. Technol. 24, 79–98.
[36] Peng, X., Li, Y., Tsang, I.W., H. Zhu, J.L., Zhou, J.T., 2022. Xai beyond classification: Interpretable neural clustering. J. Mach. Learn. Res. 23, 1–28.
[37] Qin, X., Ting, K.M., Zhu, Y., Lee., V.C., 2019. Nearest-neighbour-induced isolation similarity and its impact on density-based clustering, in: in Proc. AAAI Conf. Artif. Intell., pp. 4755–4762.
[38] Shao, W., He, L., Lu, C., Yu, P.S., 2016. Online multi-view clustering with incomplete views, in: 2016 IEEE Int. Conf. Big Data, Washington D.C., USA. pp. 1012–1017.
[39] Silva, J.A., Faria, E.R., Barros, R.C., Hruschka, E.R., de Carvalho, A.C.P.L.F., Gama, J., 2013. Data stream clustering: A survey. ACM Comput. Surv. 46, 1–31.
[40] Sui, J., Liu, Z., Liu, L., Jung, A., Li, X., 2022. Dynamic sparse subspace clustering for evolving high-dimensional data streams. IEEE Trans. Cybern. 52, 4173–4186.
[41] Turk, M.A., Pentland, A.P., 1991. Face recognition using eigenfaces, in: in Proc. IEEE Conf. Comput. Vis. Pattern Recognit., Lahaina, Maui, Hawaii, USA. pp. 586–587.
[42] Wang, S., Xiao, S., Zhu, W., Guo, Y., 2022. Multi-view fuzzy clustering of deep random walk and sparse low-rank embedding. Inf. Sci. 586, 224–238.
[43] Yang, M., Huang, Z., Hu, P., Li, T., Lv, J., Peng, X., 2022a. Learning with twin noisy labels for visible-infrared person re-identification, in: in Proc. IEEE Conf. Comput. Vis. Pattern Recognit., New Orleans, Louisiana, USA. pp. 14308–14317.
[44] Yang, M., Li, Y., Hu, P., Bai, J., Lv, J., Peng, X., 2022b. Robust multi-view clustering with incomplete information. IEEE Trans. Pattern Anal. and Mach. Intell. , 1–14doi:10.1109/TPAMI.2022.3155499.
[45] Zhang, C., Cui, Y., Han, Z., Zhou, J.T., Fu, H., Hu, Q., 2020a. Deep partial multi-view learning. IEEE Trans. Pattern Anal. Mach. Intell. , 1–14doi:10.1109/TPAMI.2020.3037734.
[46] Zhang, C., Fu, H., Wang, J., Li, W., Cao, X., Hu, Q., 2020b. Tensorized multi-view subspace representation learning. Int. J. Comput. Vis. 128, 2344–2361.
[47] Zhang, T., Ramakrishnan, R., Livny, M., 1997. Birch: A new data clustering algorithm and its applications. Data Min. Knowl. Discov. 1, 2215–2228.
[48] Zubaroğlu, A., Atalay, V., 2021. Data stream clustering: a review. Artif. Intell. Rev. 54, 1201–1236.