SVStream: A Support Vector Based Algorithm for Clustering Data Streams

Chang-Dong Wang, Student Member, IEEE, Jian-Huang Lai, Member, IEEE, Dong Huang, and Wei-Shi Zheng, Member, IEEE

Abstract—In this paper, we propose a novel data stream clustering algorithm, termed SVStream, which is based on support vector domain description and support vector clustering. In the proposed algorithm, the data elements of a stream are mapped into a kernel space, and the support vectors are used as the summary information of the historical elements to construct cluster boundaries of arbitrary shape. To adapt to both dramatic and gradual changes, multiple spheres are dynamically maintained, each describing the corresponding data domain presented in the data stream. By allowing for bounded support vectors (BSVs), the proposed SVStream algorithm is capable of identifying overlapping clusters. A BSV decaying mechanism is designed to automatically detect and remove outliers (noise). We perform experiments over synthetic and real data streams, with the overlapping, evolving and noise situations taken into consideration. Comparison results with state-of-the-art data stream clustering methods demonstrate the effectiveness and efficiency of the proposed method.

Index Terms—Data stream clustering, support vector, clusters of arbitrary shape, overlapping, evolving, noise.

1 INTRODUCTION

In the past few years, a huge amount of streaming data, such as network flows, phone records, sensor data and web click streams, have been generated due to the progress in hardware technology. Analyzing these data has been a hot research topic [1], [2], [3], [4], [5]. The goal of data stream analysis is to make decisions based on the summary information gathered over the past observed data elements. Common techniques include classification [6], [7], clustering [1], [8], model ensemble [9], [10], multiple or distributed data stream clustering [11], [12], change diagnosis [13], [14], query processing [15], etc. Among them, clustering is one of the most effective means of summarizing data streams and building a model for visualization and analysis [1], [8], [16], [17], [18]. For example, clustering sales log streams helps advertisers discover market segments, and clustering telephone-call records exposes fraudulent telephone use in real time. In this paper, we propose an effective and efficient data stream clustering algorithm, which is based on support vector domain description [19] and support vector clustering [20].

1.1 Related Work

One of the earliest and best known clustering algorithms for very large datasets is the BIRCH approach [21], [22], which relies on CF-tree [21] to perform hierarchical clustering. However, it is not designed for clustering data streams and cannot address the concept drift problem. The first well-known algorithm aiming at performing clustering over entire data streams is the STREAM algorithm proposed by Guha et al. [16], [23]. The STREAM algorithm extends the classical $k$-median algorithm in a divide-and-conquer fashion to cluster data streams in a single pass. Such methods simply view data stream clustering as a variant of the one-pass clustering algorithm, and also cannot deal with concept drift. Babcock et al. [24] proposed to extend the STREAM algorithm from one-pass clustering to the sliding window model, where data elements arrive in a stream and only the last $N$ elements are considered relevant at any moment. However, as pointed out in [18], [25], this model only partially addresses the evolving characteristics of stream data.

The CluStream framework proposed in [1] is effective in handling evolving data streams. It divides the clustering process into an online component which periodically uses micro-clusters to store detailed summary statistics and an offline component which uses this summary statistics in conjunction with other user input to produce clusters. For high-dimensional data stream clustering, Aggarwal et al. [17], [26] proposed HPStream, which reduces the dimensionality of the data stream via data projection before clustering. For automatically estimating the cluster number within the data stream, Zhang et al. [27] extended the affinity propagation algorithm [28] to a data stream clustering version called StrAP.

One common limitation of the above algorithms is that, they cannot find clusters of arbitrary shape, since most of them are based on $k$-means-like algorithms. However, nonlinearly separable clusters of arbitrary shape are seen in many applications [8], [18], [25], [29], [30], [31]. To discover clusters of arbitrary shape, Cao et al. [8] proposed a DenStream algorithm, which extends DBSCAN [29] by introducing micro-clusters to the density based connectivity search. Independently, Chen and Tu [18], [25] also proposed a density-based method termed D-Stream. Rather than using micro-clusters, D-Stream partitions
the data space into grids and maps new data points into the corresponding grid to store density information, which are further clustered based on the density. RepStream [32] is another algorithm capable of identifying clusters of arbitrary shape. It is a sparse-graph-based approach that employs representative cluster points to incrementally process incoming data and uses a graph-based description to model the spatio-temporal relationships. Compared with the linear clustering methods, these nonlinear approaches can identify clusters of arbitrary shape; however, they are computationally more expensive, requiring more memory to store nonlinear cluster structures and more time to update them.

1.2 The Proposed Work

In this paper, inspired by support vector domain description (SVDD) [19] and support vector clustering (SVC) [20], we propose a novel data stream clustering algorithm termed SVStream (Support Vector based Stream clustering). According to the SVDD theory [19], support vectors (SVs) can obtain flexible and accurate data descriptions by mapping the data into a kernel space. These SVs are used to construct cluster boundaries of arbitrary shape in SVC [20].

Compared to the original SVC algorithm, one major challenge of extending SVC to data stream clustering is how to maintain SVs so as to adapt to dramatic and gradual changes in streaming. To this end, we propose a multi-sphere representation, where multiple spheres are dynamically maintained in a sphere set. Each sphere describes the corresponding data domain presented in a data stream. When a new data chunk arrives, if a dramatic change occurs, a new sphere is created; otherwise, only the existing spheres are updated to take into account the new chunk. The data elements of this new chunk are assigned with cluster labels according to the cluster boundaries constructed by the sphere set. Another challenge of utilizing SVC to data stream clustering lies in properly discarding bounded support vectors (BSVs). Since as new data arrive, more and more BSVs are generated, causing a dramatic demand for storage and computation, but cannot be directly discarded because some of them would become new SVs in later steps. To tackle this problem, a BSV decaying mechanism is proposed, which assigns each BSV with a BSV age, through which appropriate BSVs are discarded as outliers.

Compared with the existing data stream clustering algorithms, which would fail in identifying arbitrary-shaped clusters, or cannot adapt to the dynamic data streams, or require expensive computational resources to achieve these goals, the proposed SVStream algorithm has the following advantages:

- SVStream accurately discovers clusters of arbitrary shape by constructing cluster boundaries using SVs.
- It can adapt to dramatic and gradual changes in an evolving stream by dynamically maintaining multiple spheres.
- It can discover overlapping clusters by allowing for BSVs. It is also effective in detecting and removing outliers (noise) via the BSV decaying mechanism.
- It is very efficient and requires very small memory space due to the compact representation of the summary information by multiple spheres.

The remainder of this paper is organized as follows. In section 2, we briefly review the background including SVDD and SVC, which form the basis of our approach. Section 3 describes in detail the proposed SVStream algorithm. The experimental results are reported in section 4. We conclude this paper in section 5.

2 BACKGROUND

2.1 Support Vector Domain Description

In domain description, the task is to give a description of a set of objects, which should cover the positive objects and reject the negative ones in the object space [19]. Support vector domain description (SVDD) is a sphere shaped data description. By using a nonlinear transformation, SVDD can obtain a very flexible and accurate data description relying on only a small number of SVs.

Let \( \mathcal{X} = \{ \mathbf{x}_i \in \mathbb{R}^d | i = 1, \ldots, N \} \) be a given data set of \( N \) points. Using the nonlinear transformation \( \phi \) from the input space to a Gaussian kernel feature space\(^1\), we look for the smallest sphere that encloses most of the data points in the feature space, which is described by the center \( \mu \) and the radius \( R \). That is, we minimize the radius

\[
F(R, \mu, \xi_i) = R^2 + C \sum_{i=1}^{N} \xi_i,
\]

under the constraints \( ||\phi(\mathbf{x}_i) - \mu||^2 \leq R^2 + \xi_i, \forall i = 1, \ldots, N \), where the parameter \( C \) gives the trade-off between the sphere volume and the accuracy of data description, and \( \xi_i \geq 0 \) are some slack variables. By introducing the Lagrangian, we have

\[
L(R, \mu, \xi_i, \beta_i, \alpha_i) = R^2 + C \sum_{i=1}^{N} \xi_i - \sum_{i=1}^{N} \beta_i (R^2 + \xi_i - ||\phi(\mathbf{x}_i) - \mu||^2) - \sum_{i=1}^{N} \alpha_i \xi_i,
\]

where \( \beta_i \geq 0, \alpha_i \geq 0 \) are Lagrange multipliers. Setting to 0 the derivative of \( L \) w.r.t. \( R, \mu \) and \( \xi_i \) respectively leads to

\[
\sum_{i=1}^{N} \beta_i = 1, \quad \mu = \sum_{i=1}^{N} \beta_i \phi(\mathbf{x}_i), \quad \beta_i = C - \alpha_i.
\]

The KKT complementarity conditions result in

\[
\alpha_i \xi_i = 0, \quad \beta_i (R^2 + \xi_i - ||\phi(\mathbf{x}_i) - \mu||^2) = 0.
\]

By eliminating the variables \( R, \mu, \xi_i \) and \( \alpha_i \), the Lagrangian can be turned into the Wolfe dual form

\[
\max_{\beta_i} W = \sum_{i=1}^{N} \beta_i K(\mathbf{x}_i, \mathbf{x}_i) - \sum_{i,j=1}^{N} \beta_i \beta_j K(\mathbf{x}_i, \mathbf{x}_j)
\]

subject to \( \sum_{i=1}^{N} \beta_i = 1, \; 0 \leq \beta_i \leq C, \forall i = 1, \ldots, N \).
where the dot products $\phi(x_i) \cdot \phi(x_j)$ are replaced by an appropriate Gaussian kernel $K(x_i, x_j) = \exp(-q\|x_i - x_j\|^2)$ with the width parameter $q$. In [33], a dynamic dissimilarity measure was proposed to eliminate the sensitivity to the choice of this kernel parameter.

According to the values of the Lagrange multipliers $\beta_i, i = 1, \ldots, N$, the data points are classified into three types:

- Inner points (IPs): $\beta_i = 0$, which lie either inside or on the sphere surface.
- Support vectors (SVs): $0 < \beta_i < C$, which lie on the sphere surface.
- Bounded support vectors (BSVs): $\beta_i = C$, which lie either outside or on the sphere surface.

It is obvious that setting $C \geq 1$ will result in no BSV.

The kernel radius function, defined by the Euclidean distance $\|x - \mu\|$, is given by

$$R(x) = \sqrt{1 - 2\sum_{i=1}^{N} \beta_i K(x_i, x) + \sum_{i,j=1}^{N} \beta_i \beta_j K(x_i, x_j)}.$$  \hspace{1cm} (6)

The radius of the sphere is defined as

$$R = \max\{R(x_i) | x_i \text{ is a SV}, i.e. 0 < \beta_i < C\}.$$  \hspace{1cm} (7)

However, in some extreme case where the parameter $C$ is set so small that all data points are BSVs, there would be no SV. For instance, if $C = 0.1$ and we have $N = 10$ data points, since $\sum_{i=1}^{N} \beta_i$ must equal 1, automatically all $\beta_i, i = 1, \ldots, N$ must equal $C$ (which means all the data points are BSVs), so the definition for the sphere radius fails—it is undefined in this case. To handle this problem, we provide an alternative definition for the sphere radius as follows.

$$R = \begin{cases} \max \{R(x_i) | 0 < \beta_i < C\} & \text{if } \{x_i | 0 < \beta_i < C\} \neq \emptyset \\ \min \{R(x_i)\} & \text{otherwise}. \end{cases}$$  \hspace{1cm} (8)

The reason for this definition is that although there is no data point satisfying $0 < \beta_i < C$ due to $\beta_i = C, i = 1, \ldots, N$, according to the definition of BSVs, some BSVs may lie on the sphere surface giving a proper definition to the sphere radius. The contours enclosing most of the points in the data space are defined by the set $\{x | R(x) = R\}$.

### 2.2 Cluster Labeling

To achieve cluster labeling, an adjacency matrix $A = [A_{ij}]_{N \times N}$ is computed as

$$A_{ij} = \begin{cases} 1 & \text{if } \forall y \text{ on the line segment connecting } x_i \text{ and } x_j, R(y) \leq R \\ 0 & \text{otherwise}. \end{cases}$$  \hspace{1cm} (9)

Clusters are defined as the connected components of the graph induced by $A$. Checking the line segment is implemented by sampling a number of points (usually 10 points are used). This is called the complete graph (CG) labeling method [20]. BSVs are unclassified by this procedure since their feature space images always lie outside the sphere. One may either leave them unclassified, or assign them to the closest cluster as we do in our method.

To speedup cluster labeling, several cluster labeling methods have been developed [34], [35], [36]. For instance, in [34], three cluster labeling methods were introduced, including the delaunay diagram (DD), minimum spanning tree (MST) and $K$-nearest neighbor (KNN). In [35], a SEP-based complete graph (SEP-CG) labeling method was developed based on some invariant topological properties of the kernel radius function. Another similar cluster labeling method, termed E-SVC, is based on the equilibrium vector [36]. In our approach, the CG cluster labeling method is used, which produces the most accurate clustering results.

### 3 The Proposed SVStream Algorithm

In this paper, we consider the problem where the data elements of a stream arrive over time in chunks [37], [38], [39]. Without loss of generality, the chunks are of equal size, each containing $M$ data elements. Let $X^t$ denote the $t$-th chunk, $X^t = \{x_1^t, \ldots, x_M^t\}$, where $x_i^t$ is the $i$-th data element of $X^t$.

The data stream has the form,

$$x_1^t, \ldots, x_M^t, \ldots, x_1^{t+1}, \ldots, x_M^{t+1}, \ldots, x_1^{t+2}, \ldots, x_M^{t+2}, \ldots,$$

#### 3.1 Multi-Sphere Definition

To adapt to both dramatic and gradual changes, multiple spheres are dynamically maintained in a sphere set $S$. We begin by defining the concept of the sphere structure $S$.

**Definition 1.** Given a set of $M$ data elements, the Gaussian kernel parameter $q$ and the trade-off parameter $C$, the sphere structure $S$ is defined as

$$S = \{SV, BSV, \|\mu\|^2, R_{SV}, R_{BSV}\}.$$  \hspace{1cm} (11)

The definition of each entry is as follows:

- $SV$ is a support vector set
  $$SV = \{(x_i, \beta_i, L_i, T_i) | 0 < \beta_i < C, i = 1, \ldots, M\}.$$  \hspace{1cm} (12)
- $BSV$ is a bounded support vector set
  $$BSV = \{(x_i, \beta_i, L_i, T_i) | \beta_i = C, i = 1, \ldots, M\}.$$  \hspace{1cm} (13)
- $\|\mu\|^2$ is the squared length of the sphere center $\mu$
  $$\|\mu\|^2 = \sum_{(x_i, \beta_i, \ldots) \in SV \cup BSV} \beta_i \beta_j K(x_i, x_j).$$  \hspace{1cm} (14)
- $R_{SV}$ is the radius of the sphere
  $$R_{SV} = \begin{cases} \max \{R(x_i) | (x_i, \ldots) \in SV\} & \text{if } SV \neq \emptyset \\ \min \{R(x_i) | (x_i, \ldots) \in BSV\} & \text{otherwise}. \end{cases}$$  \hspace{1cm} (15)
- $R_{BSV}$ is the maximum Euclidian distance of the bounded support vectors from the sphere center $\mu$
  $$R_{BSV} = \max \{R(x_i) | (x_i, \ldots) \in BSV\}.$$  \hspace{1cm} (16)

For convenience, $R_{SV}$ and $R_{BSV}$ are termed $S$-radius and $B$-radius respectively, with their corresponding sphere surfaces called $S$-surface and $B$-surface.
In the above definition, a \((d + 3)\) tuple \((x_i, \beta_i, L_i, T_i)\) is used to represent a SV or BSV, where \(x_i\) is a \(d\)-dimensional vector, \(\beta_i\) is the Lagrange multiplier, \(L_i\) is the cluster label, and \(T_i\) is the BSV age defined as the accumulated times for \(x_i\) to be a BSV. When some entries of this tuple are unknown or when the computation is independent of them, they will be denoted as ". . . " like in equations (14), (15) and (16).

**Definition 2.** The multi-sphere set \(SS\) is defined as a set consisting of multiple spheres, that is, \(SS = \{S^1, \ldots, S^{||SS||}\}\), where the superscript denotes the index of a sphere.

In clustering process, if two spheres \(S\) and \(\overline{S}\) are too close to each other, they should be merged. The problem is how to measure the "closeness" between two spheres. According to Fig. 1(a), a natural measurement of the similarity between two spheres \(S = \{SV, BSV\}, \|\mu\|^2, R_{SV}, R_{BSV}\}\) and \(\overline{S} = \{\overline{SV}, \overline{BSV}, \|\mu\|^2, \overline{R}_{SV}, \overline{R}_{BSV}\}\) should take into account both of their sphere center distance and their S-radiusi. The sphere distance \(SDist(S, \overline{S})\) is defined as the ratio of the sphere center distance \(|\mu - \overline{\mu}|\) to the sum of the S-radiusi \(R_{SV} + \overline{R}_{SV}\).

**Definition 3.** The sphere distance between two spheres \(S\) and \(\overline{S}\) is defined as \(SDist(S, \overline{S}) = ||\mu - \overline{\mu}|| / (R_{SV} + \overline{R}_{SV})\), where \(||\mu - \overline{\mu}||\) is computed by

\[
||\mu - \overline{\mu}||^2 = ||\mu||^2 + ||\overline{\mu}||^2 - 2 \sum_{(\xi_i, \beta_i, \ldots) \in SV \cup BSV} (\xi_i, \beta_i, \ldots) \in SV \cup BSV \beta_i\beta_j K(x_i, x_j).
\]

### 3.2 Multi-Sphere Evolution

#### 3.2.1 Overall Procedure

For the first chunk \(\lambda^1\), the first sphere \(S\) is generated by Definition 1 and the data elements of \(\lambda^1\) are directly assigned with cluster labels by the CG cluster labeling method [20].

When a new data chunk \(\lambda^t\) arrives, it is necessary to determine whether a new sphere should be created according to the amount of the data elements of \(\lambda^t\) lying outside all the existing B-surfaces. A data element is lying outside the B-surface if its distance from the sphere center is larger than the B-radius. Let \(O\lambda^t\) denote the subset of \(\lambda^t\) consisting of data elements lying outside all the existing B-surfaces. That is,

\[
O\lambda^t = \{x^t_i \in \lambda^t | \forall t = 1, \ldots, ||SS||, R^t(x^t_i) > R^t_{BSV}\}.
\]

According to the ratio of \(O\lambda^t\) to \(\lambda^t\), there are two cases in updating the sphere set \(SS\):

1. If the ratio of \(O\lambda^t\) to \(\lambda^t\) is larger than the outer-sphere threshold \(\delta\), the existing spheres are incapable of describing the major part of this new chunk. A new sphere is created for \(O\lambda^t\) and each data element of \(O\lambda^t \cap \lambda^t\) is incorporated into the nearest data domain by updating the corresponding sphere.

To create a new sphere for \(O\lambda^t\), the Lagrange multipliers are obtained by solving (5). The sphere structure \(S\) is obtained by Definition 1. The BSV age of each SV and BSV is set to 0 and 1 respectively. The cluster labels of SVs and BSVs are assigned by the CG cluster labeling [20], and these labels start with the current maximum label plus one.

For each of the remaining data elements, i.e., \(\forall x^t_i \in \lambda^t \setminus O\lambda^t\), the closest sphere \(S^m\) is found by \(m = \arg \min_{t = 1, \ldots, ||SS||} R^t(x^t_i) / R^t_{BSV}\). Let \(\hat{t}^l, t = 1, \ldots, ||SS||\) denote the subset of \(\lambda^t\) consisting of the data elements of which the closest sphere is \(S^m\),

\[
\hat{t}^l = \{x^t_i \in \lambda^t | m = \arg \min_{t = 1, \ldots, ||SS||} R^t(x^t_i) / R^t_{BSV}\},
\]

\[
\lambda^t \setminus O\lambda^t = \bigcup_{t = 1, \ldots, ||SS||} \hat{t}^l,\lambda^t \setminus O\lambda^t = \bigcup_{t = 1, \ldots, ||SS||} \hat{t}^l,\lambda^t \setminus O\lambda^t = \bigcup_{t = 1, \ldots, ||SS||} \hat{t}^l.
\]

The data elements of \(\hat{t}^l\) are used to update \(S^m\), as will be discussed in subsubsection 3.2.2.

II. If the ratio of \(O\lambda^t\) to \(\lambda^t\) is not larger than the outer-sphere threshold \(\delta\), there is no need of creating a new sphere. Instead, each data element is directly incorporated into the nearest data domain. Let \(\tilde{t}^l, t = 1, \ldots, ||SS||\) denote the subset of \(\lambda^t\) consisting of the data elements of which the closest sphere is \(S^m\),

\[
\tilde{t}^l = \{x^t_i \in \lambda^t | m = \arg \min_{t = 1, \ldots, ||SS||} R^t(x^t_i) / R^t_{BSV}\},
\]

\[
\lambda^t \setminus O\lambda^t = \bigcup_{t = 1, \ldots, ||SS||} \tilde{t}^l,\lambda^t \setminus O\lambda^t = \bigcup_{t = 1, \ldots, ||SS||} \tilde{t}^l,\lambda^t \setminus O\lambda^t = \bigcup_{t = 1, \ldots, ||SS||} \tilde{t}^l.
\]

The data elements of \(\tilde{t}^l\) are used to update \(S^m\). After updating the sphere set \(SS\) to take into account \(\lambda^t\), some spheres may become so close to each other that the data domains described by them are strongly overlapping. For any two spheres \(S\) and \(\overline{S}\), if their SDist\((S, \overline{S})\) is less than the sphere merging threshold \(\eta\), \(S\) and \(\overline{S}\) are merged as a new sphere \(\overline{S}\). We will describe the sphere merging process in subsubsection 3.2.3.

In this way, we have successfully updated the summary information represented by the multi-sphere set \(SS\), and the cluster structure has also been formed by constructing the cluster boundaries according to the SVs. As a result, each data element of this new chunk \(\lambda^t\) can be directly assigned to the closest cluster.

In the above sphere updating and merging processes, some BSVs may become SVs, and new BSVs may be generated. Keeping all the BSVs forever after their presences may cause the dramatic demand for storage and computation. On the other hand, directly deleting a BSV once it is detected in creating a new sphere may result in the loss of some potential SVs in later steps. Therefore, a rational choice is to eliminate an old BSV if its BSV age is older than the BSV decaying threshold \(\zeta\). That is, for each sphere \(S\) in \(SS\), \(\forall (x_i, \beta_i, L_i, T_i) \in BSV\), if \(T_i \geq \zeta\), the corresponding \((x_i, \beta_i, L_i, T_i)\) is deleted from BSV and \(x_i\) is labeled as an outlier by re-assigning its cluster label as 0. Two advantages of this BSV decaying mechanism are that, 1) eliminating old BSVs can help generate more accurate cluster boundaries and detect the trend for one cluster to shrink or split which is often the case in data stream clustering; 2) it provides a natural way of detecting and handling outliers.

3. For example, if the current maximum label is 10 (the SVStream algorithm has detected 10 clusters so far) and the new sphere \(S\) is partitioned into 3 clusters, the first cluster of these 3 clusters is labeled as cluster 11, the second as cluster 12 and the third as cluster 13.
3.2.2 Sphere Update

The goal of sphere update is to update the sphere \( S \) such that the resulting sphere describes the data domain comprising of the original one and the one formed by data elements of \( \mathcal{X} \).

The set \( \mathcal{X} \) is first partitioned into two subsets \( \mathcal{I} \) and \( \mathcal{O} \), such that the subset \( \mathcal{I} \) consists of the data elements lying inside the S-surface of \( S \), and the subset \( \mathcal{O} \) is the complementary set consisting of the data elements lying on or outside the S-surface. That is,

\[
\mathcal{I} = \{ x_i \in \mathcal{X} | R(x_i) < R_{SV} \}, \quad \mathcal{O} = \mathcal{X} \setminus \mathcal{I}. \tag{21}
\]

According to the SVDD theory [19], inner points have no influence over the resulting SVs or BSVs. Therefore, the data elements of \( \mathcal{I} \) do not contribute to the update of the sphere \( S \). Only the subset \( \mathcal{O} \) needs to be considered. Since the data elements of \( \mathcal{O} \) lie on or outside the S-surface of \( S \), the S-surface of the updated sphere taking into account \( \mathcal{O} \) must be larger than its original counterpart, as shown in Fig. 1(b). Consequently, it is impossible for the old inner points enclosed by the old S-surface to become SVs or BSVs. It implies that the SVs and BSVs of the updated sphere \( S' \) are only from the data elements of \( SV, BSV \) and \( O \). For simplicity, we denote the overall data elements of \( SV, BSV \) and \( O \) in a temporary data set \( T \). That is, \( T = \{ x_i | \{x_i, \ldots \} \in SV \cup BSV \} \cup O = \{ x_i | i = 1, \ldots, |T| \} \), where \( |T| = |SV| + |BSV| + |O| \) is the number of data elements of \( T \). The Lagrange multipliers are obtained for \( T \) by solving (5). The sphere structure \( S' \) is generated by Definition 1 with the BSV ages and the cluster labels determined as follows.

\[
\forall (x'_i, \beta'_i, L'_i, T'_i) \in SV', \text{ if } x'_i \in O, \text{ i.e., } x'_i \text{ is a new data element, } T'_i \text{ is set to } 0; \text{ otherwise, } T'_i \text{ is set to the original } T_i, \text{ i.e., kept unchanged. } \forall (x'_i, \beta'_i, L'_i, T'_i) \in BSV', \text{ if } x'_i \in O, \text{ T}'_i \text{ is set to } 1; \text{ otherwise, } T'_i \text{ is set to the original } T_i \text{ plus one, i.e., its BSV age increases by one.}
\]

To assign cluster labels to the data elements of \( SV' \) and \( BSV' \), an adjacency matrix \( A \) is constructed for the data elements lying inside or on the S-surface of \( S' \) (i.e., \( \beta'_i < C \)) based on the kernel radius function \( R'(x) \) and the S-radius \( R_{SV} \) via (9). The connected components are obtained. For each connected component, there are four cases:

I. If all data elements of this component are from the same old cluster, which implies that this old cluster (or part of it) is kept unchanged, the data elements of this component are assigned with the same cluster label as the old cluster.

II. If some data elements of this component are from the same old cluster and the remaining are from \( O \), which implies that this old cluster (or part of it) is enlarged, the data elements of this component are assigned with the same cluster label as the old cluster.

III. If all data elements of this component are from \( O \), which implies that a new cluster emerges, the data elements of this component are assigned with a new cluster label, i.e. the current maximum label plus one.

IV. If this component contains data elements from different old clusters, which implies that some old clusters (or parts of them) are merged to generate a new cluster, the data elements of this component are assigned with a new cluster label, i.e. the current maximum label plus one.

For the overall connected components, another possible phenomenon is that, some old cluster may be split into two or more clusters. However, from case I and case II, these split clusters are assigned with the same cluster label as the old one. Therefore, it is necessary to check the uniqueness of the above assigned cluster labels of these connected components. If there exist more than one components with the same cluster label, which implies that the old cluster with this label is split into more than one new clusters, each of these components should be re-assigned with a distinct new cluster label, i.e. the current maximum label plus one. In this way, the data elements of \( SV' \) are assigned with meaningful cluster labels. For the data elements of \( BSV' \), they are assigned into the closest clusters.

3.2.3 Sphere Merging

The goal of sphere merging is to merge \( \overline{S} \) with \( S \), such that the resulting sphere obtains an accurate data description of the sum of two data domains described by \( \overline{S} \) and \( S \) respectively. Figure 1(c) illustrates the merging of two spheres.

Similar to that in the sphere update, we denote the overall data elements of \( \overline{SV}, \overline{BSV} \) and \( SV, BSV \) in a temporary
Algorithm 1 The SVStream Algorithm

1: **Input:** $DS$: data stream, $M$: chunk size, $q$: Gaussian kernel parameter, $C$: trade-off parameter, $\delta$: outer-sphere threshold, $\eta$: sphere merging threshold, $\zeta$: BSV decaying threshold.
2: Initialize an empty sphere set $SS = \emptyset$, $t = 1$.
3: Obtain $X^1$ of $M$ elements from $DS$.
4: Create a sphere $S$ for $X^1$ by Definition 1.
5: Assign cluster labels to the data elements of $X^1$ by the CG labeling method.
6: Add $S$ to $SS$, i.e., $SS ← \{S\}$.
7: **repeat**
8: \hspace{1em} $t ← t + 1$.
9: \hspace{1em} Obtain $X^t$ of $M$ elements from $DS$.
10: Compute $O\Delta X^t$ via (18).
11: \hspace{1em} if $|O\Delta X^t|/|X^t| > \delta$ then
12: \hspace{2em} Create a new sphere $S$ for $O\Delta X^t$.
13: \hspace{2em} Incorporate each element of $X^t \setminus O\Delta X^t$ into the nearest data domain by updating the corresponding sphere.
14: \hspace{1em} Add the new sphere $S$ to $SS$, i.e., $SS ← SS \cup \{S\}$.
15: \hspace{1em} else
16: \hspace{2em} Incorporate each element of $X^t$ into the nearest data domain by updating the corresponding sphere.
17: \hspace{1em} **end if**
18: Merge any two spheres $S$ and $\bar{S}$ if $SDist(S, $\bar{S}$) < $\eta$.
19: Assign the data elements of $X^t$ to the closest clusters.
20: Delete any old BSV with the BSV age older than $\zeta$.
21: **until** No data chunk arrives.

The rings cannot be separated when no BSV is allowed. The separation gap between two overlapping rings is clear when BSVs are allowed.

and each of the remaining data elements of this chunk is incorporated into the nearest data domain.

- If a gradual change occurs or the stream remains stable, each data element of this chunk is directly incorporated into the nearest data domain.

In both cases, some spheres are possibly needed to merge with each other as a result of enlarging or creating spheres. In addition, eliminating old BSVs by the BSV decaying mechanism would help detect the trend for one cluster to shrink or split. In this way, our algorithm is capable of detecting both dramatic and gradual changes in a data stream.

There may be overlap between any two clusters, the separation gap between which is disturbed by the data points lying between them. According to the SVC theory [20], if no BSV is allowed (i.e., setting $C ≥ 1$), the SVs will lie in the overlapping region such that the overlapping clusters cannot be separated, as shown in Fig. 2(a). However, if the data points in the overlapping region are taken as BSVs, the separation gap becomes very clear, as shown in Fig. 2(b). Therefore, by setting the input trade-off parameter $C < 1$, the proposed SVStream algorithm is effective in dealing with data streams consisting of overlapping clusters.

A by-product of allowing for BSVs is the ability to handle outliers (noise) [20]. Since when BSVs are allowed, the kernel sphere only contains the data points of relatively high density, leaving the BSVs (i.e., those far away from the core of the clusters) outside the S-surfaces of spheres. By introducing the BSV decaying mechanism, the old BSVs are taken as outliers and automatically eliminated. The ability to handle outliers is especially beneficial in data stream clustering, where random outliers appear occasionally due to various factors in the generation source of a stream.

4 Experimental Results

In this section, we evaluate the effectiveness and efficiency of SVStream and compare it with several well-known data stream clustering algorithms, including StreamKM (STREAM K-Means) [16], CluStream [1], DenStream [8], RepStream [32] and StrAP [27]. All the experiments are conducted on a PC with Core(TM)2 Duo 2.4GHz processor and 2GB memory, which runs Windows XP professional operating system.

Unless particularly mentioned, the optimal values of the outer-sphere threshold $\delta$ and the sphere merging threshold $\eta$ are set...
4.1 Data Sets and Evaluation

Two synthetic data streams are first generated to test the effectiveness of SVStream on overlapping and evolving streams respectively. The first synthetic stream, called Ring-Ball, consists of a series of 2000 data elements belonging to two overlapping classes. The separation gap between the two classes is not clear in streaming due to the disturbance caused by the data points of one class. A randomized order that does not change during the experiment is taken as the order of streaming. The overall distribution of the stream is shown in Fig. 3(a). The second synthetic stream, called Smileface-Twomoons, is a data stream characterized by both dramatic and gradual changes with noise. It consists of a series of 7000 data elements with 3.4% noise as shown in Fig. 3(b). In order to test the performance with consideration to both dramatic and gradual changes, these data elements are arranged in such an order that the three classes belonging to "smile face" are presented first (Fig. 7(a)), then the two classes belonging to "two moons" emerge (Fig. 8(a)), and finally the upper two classes belonging to "smile face" are gradually merging (Fig. 9(a)).

Two real-world data streams from the UCI KDD Archive [40] are used, which are the KDD-CUP’99 Network Intrusion Detection stream data set (KDDCUP99) and the Forest CoverType data set (Forest-CoverType) respectively. The KDDCUP99 data set is a real dataset that evolves significantly over time and has been widely used to evaluate data stream clustering algorithms [1], [8], [16], [17], [32]. It consists of a series of TCP connection records of LAN network traffic managed by MIT Lincoln Labs. The complete data set contains approximately 4.9 million records, and as in the previous work [1], [17], [26], [32], a sub-sampled subset of length 494020 is used. Each connection is classified into a normal connection or an intrusion (attack). The attacks fall into four main categories: DOS (denial-of-service, e.g., syn flood), R2L (unauthorized access from a remote machine, e.g., guessing password), U2R (unauthorized access to local superuser privileges, e.g., various “buffer overflow” attacks), and PROBING (surveillance and other probing, e.g., port scanning). Each connection record in this dataset contains 42 attributes, and as in [1], [8], [16], [17], [32], all 34 continuous attributes are used for clustering and one outlier point has been removed. The Forest-CoverType data set contains totally 581012 observations belonging to seven forest cover types. Each observation consists of 54 geological and geographical features that describe the environment in which trees are observed, including 10 quantitative variables, 4 binary wilderness areas and 40 binary soil type variables. As in [17], [26], all the 10 quantitative variables are used. These two real-world datasets are converted into data streams by taking the data input order as the order of streaming.

The clustering quality is evaluated by Rand index (RI) [41], which measures how accurately a clusterer can classify data elements by comparing cluster labels with the underlying class labels. Given \( N \) data points, the total \( \binom{N}{2} \) distinct pairs of data points can be categorized into four categories: (1) pairs having the same cluster label and the same class label, with their number denoted as \( N^{11} \); (2) pairs having different cluster labels and different class labels, with their number denoted as \( N^{00} \); (3) pairs having the same cluster label but different class labels, with their number denoted as \( N^{10} \); (4) pairs having different cluster labels but the same class label, with their number denoted as \( N^{01} \). The well-known Rand index (RI) is defined as [41]

\[
RI = \frac{N^{11} + N^{00}}{\binom{N}{2}}.
\]

The value of Rand index lies between 0 and 1. A higher Rand index indicates better clustering results. In data stream clustering, we report both the Rand index for each chunk and the average Rand index for the whole data stream.

4.2 Data Streams of Overlapping Clusters

For demonstrating the effectiveness of SVStream in discovering overlapping clusters in data streams, we first compare the performances of SVStream on the Ring-Ball data stream with \( C \) set at different values. The parameters are set that the chunk size \( M = 100 \), the Gaussian kernel parameter \( q = 9 \) and the BSV decaying threshold \( \zeta = 2 \).

Figure 4 compares the performances of SVStream on Ring-Ball with and without BSVs by setting \( C = 0.25 \) and \( C = 1 \) respectively. By taking the data elements in the overlapping region as BSVs, the separation gap between the two overlapping clusters becomes very clear and an accurate cluster structure is obtained, i.e., SVs, BSVs and the corresponding contours which directly form the cluster boundaries (Fig. 4(b)). Therefore, it achieves a good classification of the data elements of the current chunk (Fig. 4(c)). On the other hand, when no BSV is allowed by setting \( C = 1 \), the overlapping clusters are mistakenly merged into one cluster due to the disturbance caused by the data points lying between them. Please note that, in addition to the disturbed data points of the current chunk, the maintained information (i.e., SVs) of the historical data points may aggravate the failure to separate the overlapping clusters. Since by allowing for no BSV, the maintained SVs may contain some data points lying between clusters. However, by allowing for BSVs, this problem can be effectively tackled.

Figure 5 plots the Rand index as a function of the chunk step with \( C \) set to 0.25. Amongst 20 chunks, only in the first
The first 1100 elements

(a) $\mathcal{X}^{11}$ spheres, $C = 0.25$
(b) $\mathcal{X}^{11}$ clustering, $C = 0.25$
(c) $\mathcal{X}^{11}$ spheres, $C = 1$
(d) $\mathcal{X}^{11}$ clustering, $C = 1$

Fig. 4. Comparing the performances of SVStream on Ring-Ball with $(C = 0.25)$ and without $(C = 1)$ BSVs.

The Rand index as a function of the chunk step by SVStream on the Ring-Ball data stream.

(a) $\mathcal{X}^{11} \cup \mathcal{X}^{2} \cup \ldots \cup \mathcal{X}^{10}$
(b) Spheres till $\mathcal{X}^{10}$
(c) $\mathcal{X}^{10}$ clustering

(a) $\mathcal{X}^{11} \cup \mathcal{X}^{2} \cup \ldots \cup \mathcal{X}^{12}$
(b) Spheres till $\mathcal{X}^{12}$
(c) $\mathcal{X}^{12}$ clustering

(a) $\mathcal{X}^{11} \cup \mathcal{X}^{2} \cup \ldots \cup \mathcal{X}^{51}$
(b) Spheres till $\mathcal{X}^{51}$
(c) $\mathcal{X}^{51}$ clustering

(a) All deleted BSVs
(b) Spheres till $\mathcal{X}^{70}$
(c) $\mathcal{X}^{70}$ clustering

Fig. 5. The Rand index as a function of the chunk step by SVStream on the Ring-Ball data stream.

4.3 Evolving Data Streams with Noise

In this subsection, we perform SVStream on the Smileface-Twomoons data stream to demonstrate its effectiveness in clustering evolving data streams with noise. The parameters are set that $M = 100$, $q = 9$, $C = 0.25$ and $\zeta = 2$. As aforementioned, the data elements of Smileface-Twomoons are arranged in such an order that both dramatic and gradual changes will occur in streaming.

Figures 7 to 10 plot the results in four key steps (10, 12, 51 and 70) respectively. Figure 7 shows that, the proposed algorithm is capable of discovering clusters of arbitrary shape by constructing accurate cluster boundaries from SVs (Fig. 7(b)). Figure 8 shows when a dramatic change occurs that two new classes are emerging (Fig. 8(a)), the proposed algorithm is also effective in detecting this change (Fig. 8(b)) and identifying the new classes (Fig. 8(c)). In step 51, the data chunk $\mathcal{X}^{51}$ has brought a new trend that two old classes are merging (Fig. 9(a)). Figure 9 shows that SVStream can effectively discover this change, forming the adaptive cluster boundaries (Fig. 9(b)) and making accurate cluster assignments to the data elements of this chunk (Fig. 9(c)). Figure 10 shows the result of the last chunk, as well as the overall deleted BSVs. In particular, by comparing the overall deleted BSVs in Fig. 10(a) and the distribution of all data elements in Fig. 3(b), it is obvious that most of the outliers scattered in the data space have been correctly detected and removed. As will be discussed soon, amongst 70 chunks, SVStream has correctly classified 65 chunks, achieving a Rand index higher than 0.95, which is a very satisfactory result.

Figure 11 plots the Rand index as a function of the chunk step. From this figure, amongst 70 chunks, only 5 chunks have been mistakenly clustered by either merging or splitting...
and removed); however, the false detection rate will also increase (i.e., more non-outliers would be falsely detected as outliers). The reason is that, by decreasing $\zeta$, the BSVs will be maintained in a much shorter time in spheres, and be more likely detected as outliers. Similarly, as $C$ decreases from 0.34 to 0.26, the detection rate and the false detection rate will increase. This phenomenon relies on the SVDD theory \cite{19} itself. As the trade-off parameter $C$ decreases, more and more $\beta_i$ will approach the value of $C$, i.e., become BSVs. So there are more potential outliers. Another SVDD theory-based parameter affecting the detection of outliers is the Gaussian kernel width parameter $q$. As shown in this figure, the detection rate as well as the false detection rate will increase, as $q$ increases. According to the SVDD theory \cite{19}, the increase of $q$ will directly decrease the smoothness of the kernel sphere, and hence more BVSs will be generated. Overall, there exists a common inflection point for the three parameters (i.e., $\zeta = 2$, $C = 0.3$, $q = 11$), in which a proper trade-off between the detection rate and the false detection rate is obtained. It should be pointed out that, unlike $C$ and $q$, the parameter $\zeta$ is independent of the feature space of datasets, i.e., the attribute type and range, since it only controls when to transfer BSVs to outliers. Therefore, there exists a common value for this parameter. According to the inflection point in this analysis, we will set $\zeta$ to 2 as default in all experiments.

4.4 Parameter Analysis

In this subsection, we report and compare the clustering results by SVStream on the four testing streams when respectively using different chunk size $M$, Gaussian kernel parameter $q$, and trade-off parameter $C$. When analyzing one of the three parameters, the default values of the other two parameters listed in Table 1 are used. Moreover, this subsection will also analyze the connection between the performance and the parameters involved in multi-sphere representation, i.e., the out-sphere threshold $\delta$ and the sphere merging threshold $\eta$.

### Table 1

<table>
<thead>
<tr>
<th>Data streams</th>
<th>$M$</th>
<th>$q$</th>
<th>$C$</th>
</tr>
</thead>
<tbody>
<tr>
<td>Ring-Ball</td>
<td>100</td>
<td>9</td>
<td>0.25</td>
</tr>
<tr>
<td>Smileface-Twomoons</td>
<td>100</td>
<td>11</td>
<td>0.3</td>
</tr>
<tr>
<td>KDDCup99</td>
<td>100</td>
<td>0.011</td>
<td>0.2</td>
</tr>
<tr>
<td>Forest-CoverType</td>
<td>100</td>
<td>16</td>
<td>0.2</td>
</tr>
</tbody>
</table>

4. When analyzing one of the three parameters, the default values of the other two parameters are $\zeta = 2$, $C = 0.3$ and $q = 11$. 
Fig. 14. Average Rand index, execution time, number of SVs and BSVs, and memory usages vs. chunk size $M$.

### 4.4.1 Chunk Size $M$

We first analyze the performance in terms of average Rand index, execution time and memory usage when using different chunk size $M$ on the four testing data streams. Figure 14 reports the results. From Fig. 14(a), it is clear that on all testing streams except Ring-Ball, the average Rand index remains stable when different $M$ are used, with the highest average Rand index achieved in $M = 100$ in most cases. The execution time however strongly depends on the chunk size $M$ as shown in Fig. 14(b). The memory usage is mainly due to the storage of the data elements of the current chunk, the SVs and the BSVs. Figure 14(c) shows the maximum number of SVs plus BSVs in clustering the whole data streams, which is always less than 200. By comparing Fig. 14(c) and Fig. 14(d), we can see that the memory usage depends on both the number of SVs plus BSVs and the data dimensionality. On KDDCUP99, the number of SVs plus BSVs is less than half of that on Ring-Ball and Smileface-Twomoons. However, the memory usage on KDDCUP99 is about four times higher due to the fact that the data dimensionality of KDDCUP99 is 33, much higher than that of Ring-Ball and Smileface-Twomoons. Based on the above analysis, considering both the effectiveness and efficiency, a rational choice for $M$ is 100, which is used as the default parameter in this paper.

### 4.4.2 Gaussian Kernel Parameter $q$

We analyze how the average Rand index and the detected cluster number depend on the Gaussian kernel parameter $q$. To this end, we report the cluster detection rate, that is, the ratio of the chunks in which the correct cluster number is detected to the overall chunks. As discussed in [20], the Gaussian kernel width parameter $q$ controls the smoothness of the contour generated by SVDD, and its value should be chosen larger than the multiplicative inverse of the maximum squared Euclidean distance between all data points. In the data stream model, this can be selected in the first chunk by setting $q$ slightly larger than $\frac{1}{\max_{i,j} \|x_i - x_j\|^2}$. Figure 15 shows the average Rand index on the four testing streams using different kernel parameter $q$. As in SVM, there exists a relatively wide range of $q$ that can generate good clustering.

On the Ring-Ball data stream, by setting $q$ to the values ranging between 7 and 13, SVStream generates acceptable clustering results, achieving the minimum average Rand index as high as 0.85 (Fig. 15(a)) and the average cluster detection rate of 0.83. In particular, it achieves the highest average Rand index of 0.9267 when $q$ is set to 9. On Smileface-Twomoons, by setting $q$ to the values ranging from 10 to 15, the minimum average Rand index of 0.94 (Fig. 15(b)) and the average cluster detection rate of 0.68 are obtained. And it obtains the highest average Rand index of 0.98 when $q$ is set to 11. On the real KDDCUP99 data stream, the minimum average Rand index of 0.658 is achieved when $q$ is set to the range of $[0.006, 0.096]$ as shown in Fig. 15(c), and in most cases it can detect the right cluster number with the average cluster detection rate higher than 0.7. From the figure, we can see that, the most suitable kernel parameter on this data stream is 0.011. Similarly, on the real Forest-CoverType, setting $q$ to the range of $[11, 19]$ can generate the minimum average Rand index higher than 0.705 (Fig. 15(c)) with the average cluster detection rate as high as 0.75. And the most suitable kernel parameter achieving the highest average Rand index is 16.

### 4.4.3 Trade-off Parameter $C$

This subsubsection analyzes the connections between the trade-off parameter $C$ and the clustering results in terms of the average Rand index and cluster number detection rate. As discussed in [20], the trade-off parameter $C$ mainly controls the size of the sphere, as well as the number of BSVs. If it is too small, very large amount of data elements may become
BSVs; on the other hand, if it is larger than 1, no BSV will be generated. According to the previous work on SVC [20], [33], [34], [35], [36], the value of $C$ should be chosen from $[0.1, 0.3]$ such that a moderate number of BSVs can be generated. The average Rand indices as a function of $C$ on the four testing streams have validated this empirical conclusion, as shown in Fig. 16. From the figure, it is clear that on both synthetic and real data streams, there exist wide ranges of $C$ where satisfactory clustering results have been obtained.

On Ring-Ball and Smileface-Twomoons, when setting $C$ to the range $[0.22, 0.3]$, SVStream has obtained average Rand indices higher than 0.8 and 0.9 respectively, as shown in Fig. 16(a). Moreover, in this range, the average cluster detection rates are 0.77 and 0.57 respectively. On Ring-Ball, the most suitable value of $C$ is 0.25, where the average Rand index obtained is 0.9267, and the cluster detection rate is 0.85. On Smileface-Twomoons, the most suitable value of $C$ is 0.3, where SVStream obtains the average Rand index as high as 0.98. On the two real data streams, setting $C$ to the range from 0.1 and 0.3 can generate stable results, with the average Rand indices being at least 0.65 and 0.68 respectively. The cluster detection rates on both two real streams with $C$ setting between 0.1 and 0.3 are at least 0.7, which is a relatively good result. On the two real streams, the most suitable values of $C$ are 0.2, as shown in Fig. 16(b).

4.4.4 Multi-sphere Parameters

In this subsubsection, we will analyze the connection between the clustering performance of SVStream and the parameters involved in multi-sphere representation, i.e., the out-sphere threshold $\delta$ and the sphere merging threshold $\eta$. Figure 17 plots the average Rand indices vs. $\delta$ and $\eta$ on the four data streams. In general, the clustering performance is stable at the pair ranges $[\delta, \eta]$ with $\delta$ and $\eta$ being within $[0.5, 0.6]$ and $[0.9, 1.0]$ respectively. In particular, on the Smileface-Twomoons and the two real data streams, a pair of $(\delta, \eta)$ with $\delta$ smaller than 0.5 and $\eta$ smaller than 0.9 would result in not so good clustering results. Please note that, a new sphere is created only if there are more than $\delta$ percentage of data elements in the new chunk lying outside the existing B-surfaces. If setting $\delta$ too small, i.e., smaller than 0.5, the condition $|O\lambda^t|/|\lambda^t| > \delta$ can be easily satisfied, therefore too many spheres would be generated. In conjunction with small $\eta$, i.e., smaller than 0.9, which means that these too many spheres are merged only if they are too close (with their sphere distance less than 0.9), the consequence is that these too many spheres would exist for many chunks, leading to incorrect clustering results of these chunks. From this analysis, a good choice for these two parameters is to take the two bounds respectively, i.e., $\delta = 0.6$ and $\eta = 1.0$, which are used as default in all experiments.

4.5 Clustering Quality Comparison

In this subsection, we compare the effectiveness and efficiency of SVStream with the well-known data stream clustering algorithms in terms of average Rand index, execution time and memory usage.

4.5.1 Compared Algorithms and Experimental Setting

The optimal parameters analyzed in the aforementioned subsection (Table 1) are used in the SVStream algorithm. The five compared algorithms and their parameter settings are summarized below.

1. StreamKM (STREAM K-Means) [16], which processes data streams in chunks of $M$ data points. It summarizes each chunk by clustering its data into $k$ centers, each weighted by the number of assigned data points. Once $M$ centers have been collected, they are clustered to generate a smaller set of $2k$ centers, with each being weighted by the sum of the weights of the assigned centers. After one pass of the stream data, these intermediate centers are further clustered into $k$ final centers. Each data point is assigned to the nearest center. The chunk sizes $M$ are set to 500 on the four streams.

2. CluStream [1], which separates the clustering process into an online micro-clustering and an offline macro-clustering. At the beginning, the first $InitNumber$ data points are collected and clustered by $k$-means to create $q$ initial micro-clusters. When a new data point arrives, it is added to the nearest micro-cluster if this point is within its maximum boundary, which is defined as a factor of $t$ of the root-mean-square deviation of its data points from the centroid; otherwise, a new micro-cluster is created. When a new micro-cluster is created, one old micro-cluster is deleted if its least relevance stamp is below a user-defined threshold $\delta$; otherwise, two closest micro-clusters are merged. At each clock time divisible by $\alpha^i$ for any integer $i$, the current set of micro-clusters is stored on disk referring as snapshots using pyramidal time frame techniques. In the offline component, the desired $k$ macro-clusters are generated at each clock time using the micro-clusters within the time horizon $h$. As in [1], the parameters are set as follows: the number of micro-clusters $q = 10 \times k$, $InitNumber = 1000$, the maximum boundary factor $t = 2$, the user-defined threshold $\delta = 512$, the pyramidal time frame parameters $\alpha = 2$ and $l = 10$, and the time horizon $h = 100$.

3. DenStream [8], which divides the clustering process into an online part and an offline part. Each data point is weighted by a fading function $f(t) = 2^{-tM}$ with $t$ being its elapsed time and $\lambda$ being the fading factor. Micro-clusters are defined according to the sum $w$ of the weights of data points, and the radius $r$...
(which cannot be larger than the maximum radius threshold $\epsilon$). Micro-clusters are classified into $p$-micro-clusters ($w \geq \beta \cdot \mu$) and $o$-micro-clusters ($w < \beta \cdot \mu$). The first $InitNumber$ data points are collected and clustered by DBSCAN to create initial $p$-micro-clusters. When a new data point arrives, we try to merge it to the nearest $p$-micro-cluster; or merge it to the nearest $o$-micro-cluster and check whether this $o$-micro-cluster would become a $p$-micro-cluster; or directly create a new $o$-micro-cluster by this data point. In the offline part, clusters are generated by applying a variant of DBSCAN [29] to $p$-micro-clusters. As in [8], the parameters are set as follows: $InitNumber = 1000$, the fading factor $\lambda = 0.25$, the maximum radius threshold $\epsilon = 16$, the weighting threshold $\mu = 10$ and the outlier threshold $\beta = 0.2$.

4. RepStream [32], which updates two sparse $k$-NN graphs. The first sparse graph (SG) is used to select new representative vertices. The second representative sparse graph (RSG) is used to track the connectivity between all the representative vertices. When a new data point arrives, it is first added to SG. This data point will join an existing cluster if it is reciprocally connected to the corresponding representative vertex; otherwise, it will itself become a predictor vertex, which will later become a representative vertex and form a new cluster. The unused representative vertices are discarded by the first-in-first-out mechanism according to a user specified decay rate $\lambda$. Larger clusters are formed when two or more representative vertices are density-related, depending on a user specified density scaler $\alpha$. According to [32], the parameters are set as follows: the $k$-NN graph parameter $k = 9$, the decay rate $\lambda = 0.99$ and the density scaler $\alpha = 2.0$.

5. StrAP [27], which extends affinity propagation (AP) [28] to data stream. A hierarchical AP (Hi-AP) is introduced, which first randomly splits the dataset into some subsets, followed by performing AP on each subset to generate exemplars with each being weighted by the number of assigned data points. These weighted exemplars are clustered using weighted AP (WAP) [27]. The stream clustering model is initialized by applying Hi-AP to the first $InitNumber$ data points. When a new data point arrives, if its distance to the closest exemplar is less than the threshold $\epsilon$, this data point joins the corresponding cluster containing this closest exemplar; otherwise, this data point is considered as an outlier and put in the reservoir which gathers the last $M$ outliers. At every consecutive time of length $\Delta$, if one cluster has received no data point, it is taken as old cluster and is deleted. At each step, a Page Hinkley (PH) statistical test [42], [43] is applied to the outlier rate. If the PH value is larger than a user specified parameter $\lambda$, the stream clustering model is rebuilt using WAP from the weighted exemplars and the outliers in the reservoir. According to [27], the parameters are set as follows: $InitNumber = 1000$, the outlier threshold $\epsilon$ is set to the average distance of data points in the initialization step, the maximum number of outliers stored in reservoir $M = 100$, the time length $\Delta = 50$ and the PH test threshold $\lambda = 0.01$.

4.5.2 Rand index

We first compare the average Rand indices by the six algorithms, as shown in Fig. 18(a). In general, SVStream obtains the highest Rand indices on the four testing streams. On the two synthetic data streams, SVStream has significantly outperformed the linear clustering methods such as StrAP, CluStream and StreamKM, and it is still better than the state-of-the-art nonlinear clustering methods such as DenStream and RepStream. For instance, on the Ring-Ball data stream, our approach has obtained the average Rand index as high as 0.9267, which is 0.0604 higher than the second winner RepStream and is 0.2826 higher than the most classical StreamKM algorithm (achieving a 43.8% improvement). Similarly, on the Smileface-Twomoons data stream, our approach has obtained the average Rand index of 0.9857, which is 0.0565 higher than the second winner RepStream and is 0.3145 higher than StrAP (a 46.8% improvement).

On the two real data streams, SVStream also outperforms the other five methods by an average 5% improvement. On the KDDCUP99 data stream, the proposed SVStream has obtained the average Rand index of 0.6604, which is slightly better than the second winner DenStream by 0.0237 and makes a significant improvement compared with the three linear clustering methods (i.e., StrAP, CluStream and StreamKM) by at least 12.5%. On the other real data stream Forest-CoverType, relatively greater improvement has been obtained, i.e., 5.8% compared with the second winner and 32.1% compared with StreamKM. In general, although SVStream has not made such significant improvements on the two real data streams as on the two synthetic data streams, they are still encouraging improvements, especially compared with the linear clustering methods. The main reason is that, compared with the linear clustering methods such as StrAP, CluStream and StreamKM, the nonlinear SVStream forms accurate cluster boundaries of arbitrary shape; meanwhile, compared with the existing nonlinear clustering methods such as DenStream and RepStream, the proposed SVStream is more effective in dealing with both dramatic and gradual changes and outliers. Additionally, the cluster boundaries constructed from the support vectors are more accurate than those by the compared algorithms due to the generation of BSVs.

4.5.3 Execution Time

This subsubsection compares the execution time consumed by the six algorithms. The execution time includes the time used to update the summary information (or cluster structure) and the time used to realize cluster labeling, over the whole data stream. For instance, in StreamKM, the execution time includes two parts: the time used to generate the specified number of cluster centers at the first pass of the stream, and the time used to assign each data point to the nearest cluster center at the second pass of the stream. In CluStream and DenStream, the computational time includes both the online micro-clustering component and the offline macro-clustering component. In RepStream, the execution time includes the time used to construct graphs and the time used in cluster labeling. In StrAP, the execution time is measured as the time elapsed from the beginning of the streaming to the end, including the time for the PH statistical test [42], [43].

Figure 18(b) plots the execution time in seconds by the six algorithms. From the figure, the proposed SVStream algorithm
Comparing the execution time, that is, it is faster than some of them, i.e., RepStream, StrAP and StreamKM; meanwhile the same as or slower than CluStream and DenStream. For instance, although it is a little bit slower than DenStream by on average 3%, SVStream is much faster than another nonlinear clustering method RepStream by at least 10%. The main computational component that consumes the major part of SVStream execution time is cluster labeling, which includes constructing an adjacency matrix and finding connected components. However, as introduced in the Background section, there exist some recent algorithms aiming at speeding up the labeling process, the fastest ones of which are almost linear, e.g. [35]. These efforts may contribute to the proposed approach to make SVStream more efficient in computational time complexity.

4.5.4 Memory Usage
We also compare the memory used by the six algorithms. The memory usage is directly measured by the peak memory usage of each algorithm during the stream clustering process. Figure 18(c) shows that the proposed SVStream algorithm has a significant advantage over its competitors in memory usage. For instance, on the two synthetic 2-D data streams, SVStream requires respectively 62.4 and 67.6 KB memory, which are only about 10% of the memory used by the second winner. On KDDCUP99, the proposed SVStream approach consumes only 9% of the memory used by RepStream and no more than 5% of the memory used by StreamKM. The comparison result on the Forest-CoverType data stream is similar to that on the two synthetic data streams, which demonstrates that SVStream outperforms existing data stream clustering algorithms in terms of memory usage. The main reason is that, the multi-spheres provide a very compact representation of the summary information of data streams. For instance, from Fig. 14(c), in all cases, the number of SVs and BSVs is bounded by 200. That is, no more than 200 data elements are stored as the summary information in SVStream. On the other hand, the other algorithms take more memory to store the summary information, where more than 1000 intermediate points/micro-clusters should be stored.

5 CONCLUSIONS
In this paper, we have proposed an effective and efficient algorithm, called SVStream, for clustering evolving data streams consisting of overlapping clusters with noise. The algorithm dynamically maintains multiple spheres in a multi-sphere set, which are used to represent the summary information of the historical data elements. According to the SVDD theory, the multi-sphere representation provides a very compact and accurate data description of the historical data chunks, so it has a limited memory consumption with precision guaranteed. The support vectors are used to construct cluster boundaries of arbitrary shape. To adapt to both dramatic and gradual changes of a stream, when a new data chunk arrives, the multi-sphere set is updated accordingly (e.g., creating a new sphere when a dramatic change occurs). By allowing for BSVs, the proposed algorithm is capable of partitioning overlapping clusters. In addition, the outliers (noise) can be effectively detected and removed via the BSV decaying mechanism. Experimental results over synthetic and real data streams demonstrate the effectiveness and efficiency of the proposed algorithm.

ACKNOWLEDGMENTS
This project was supported by the NSFC-GuangDong (U0835005) and the NSFC (60803083). The authors would like to thank the associate editor and reviewers for their comments which are very helpful in improving the manuscript.

REFERENCES

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication.
Q. Zhang, J. Liu, and W. Wang, "Incremental subspace clustering over multiple data streams," in Proc. of the 7th Int. Conf. on Data Mining, 2007.


C. C. Aggarwal, J. Han, J. Wang, and P. S. Yu, "On high dimensional projected clustering of data streams," Data Mining and Knowledge Discovery, vol. 10, pp. 251–273, 2005.


