IEEE Access
Multidisciplinary : Rapid Review : Open Access Journal

Received June 7, 2018, accepted July 15, 2018, date of publication August 10, 2018, date of current version September 7, 2018.

Digital Object Identifier 10.1109/ACCESS.2018.2864553

Dynamic Clustering Scheme for Evolving Data Streams Based on Improved STRAP

JINPING SUI[1], ZHEN LIU[1], ALEXANDER JUNG[2], LI LIU[3,4], AND XIANG LI[1]
[1]College of Electronic Science, National University of Defense Technology, Changsha 410073, China
[2]Department of Computer Science, Aalto University, 02150 Espoo, Finland
[3]College of System Engineering, National University of Defense Technology, Changsha 410073, China
[4]Center for Machine Vision and Signal analysis, University of Oulu, 90014 Oulu, Finland

Corresponding author: Zhen Liu (zhen_liu@nudt.edu.cn)

ABSTRACT A key problem within data mining is clustering of data streams. Most existing algorithms for data stream clustering are based on quite restrictive models for the cluster dynamics. In an attempt to overcome the limitations of existing methods, we propose a novel data stream clustering method, which we refer to as improved streaming affinity propagation (ISTRAP). The ISTRAP is based on an integrated evolution detection framework which ensures that the new emerging clusters are recognized timely. Moreover, within ISTRAP, outdated clusters are removed and recurrent clusters are efficiently detected rather than being treated as novel clusters. The proposed ISTRAP is non-parametric in the sense of not requiring any prior information about the number or the centers of clusters. The effectiveness of ISTRAP is evaluated using numerical experiments.

INDEX TERMS Data stream clustering, evolving data streams, affinity propagation (AP), on-line clustering.

I. INTRODUCTION

We consider data streams as sequences of data points that are continuously generated at rapid rates (such as individual frames of a movie or the amplitudes of a sound signal) [1]–[4]. In the last decade tremendous research efforts have been devoted to developing data stream mining techniques [5]–[7]. There is a trend towards non-parametric data stream clustering (DSC) methods which do not require prior information which is rarely available in practice [1]–[4], [8]–[14].

A drawback of existing DSC algorithms is that they cannot handle well rapid cluster evolution patterns [3]. Having an algorithm with improved ability to track the cluster evolution in data streams is desirable at least for two reasons: (i) the evolution pattern provides useful information (such as time of intrusion in a cybersecurity application); (ii) it can help users to make immediate decisions.

The data streams generated in many important applications have a time-evolving cluster structure [1], [15]–[20]. For example, within social media networks such as Twitter, timely topics appear quickly (cluster emergence) while older topics lose relevance over time (cluster disappearance). Moreover, some topics which disappeared will be popular again (cluster reoccurrence), such as periodical topics *e.g.,* festivities [21], [22]. Similar cluster evolution patterns will also be found in monitoring or observing systems in astronomy, medicine, finance or network [7], [22], [23]. In what follows, we focus on three typical types of cluster evolution: emergence, disappearance and reoccurrence.

Prior Art. Most existing DSC algorithms (see [26]–[31]) can detect the emergence of clusters [24]. In particular, the streaming affinity propagation (STRAP) is an efficient data stream clustering method based on the static clustering algorithm affinity propagation (AP) [32], [33]. Compared with other DSC algorithms, such as Den-stream [26], Clustream [25], STRAP is more accurate, stable and computationally efficient by selecting exemplars (representative data points of clusters) continuously [31]. The STRAP has been applied successfully for monitoring and discovering anomalies within a grid computing infrastructure [28]–[31]. However, the ability of STRAP to cope with disappearing and re-occurring clusters has not been studied so far. While the two algorithms MEC [17] and MONIC [18] aim at tracking cluster transitions, the reoccurrence of clusters is not targeted explicitly by them.

Contributions. By extending the basic STRAP method with a particular cluster evolution tracking, we propose a novel method termed improved STRAP (ISTRAP) in this paper. The ISTRAP method inherits some of the advantages of STRAP while gaining more satisfactory performance when tracking the cluster evolution. First, we design evolution detectors for each type of evolution. Then, a strategy is proposed to integrate the three types of cluster evolution detection. Two reservoirs are introduced: one (outlier reservoir) is for storing the outliers and the other (remove reservoir) is for collecting inactive clusters. Specifically, the former is related to the emergence detection and the latter is for the reoccurrence detection. Thirdly, the parameter sensitivity of ISTRAP is also analyzed, putting emphasis on analyzing its sensitivity to the parameters related to three types of cluster evolution. Finally, simulations on artificial and real data streams are carried out. The validity and superiority of ISTRAP in reacting to cluster evolution are shown by comparing its processing results with those of original STRAP.

Outline. The remainder of this paper is organized as follows. Section II provides the problem formulation and basic models which are used by ISTRAP. In Section III, we present ISTRAP. In Section IV, we evaluate the performance of the ISTRAP method using illustrative numerical experiments.

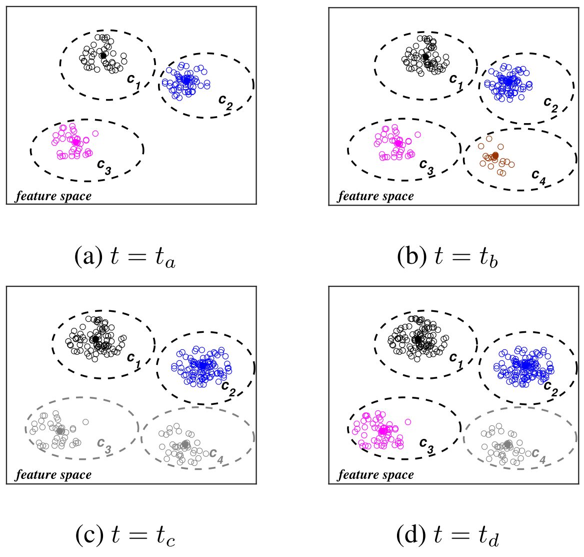

FIGURE 1. An illustration of how clusters evolve at timestamps $t_d > t_c > t_b > t_a$. (a) c_1, c_2 and c_3 exist at t_a; (b) c_4 emerges at t_b; (c) c_3 and c_4 disappear at t_c; (4) c_3 recurs at t_d. We use grey color to indicate disappeared clusters.

II. PROBLEM FORMULATION

A data stream is defined as a sequence of data points $\mathcal{X} = \{\boldsymbol{x}^t\}_{t=1}^N$, $(N \to \infty)$, each data point $\boldsymbol{x}^t \in \mathbb{R}^{n\times 1}$ being an n-dimensional vector received at timestamp t. For each timestamp t, DSC methods group the stream of data points into meaningful clusters [10]. The clustering for timestamp t is a set denoted as $\mathcal{C}^t$, *i.e.*, $\mathcal{C}^t = \{\boldsymbol{c}_1, \boldsymbol{c}_2, \ldots, \boldsymbol{c}_{k^t}\}$, constituted by k^t vectors $\boldsymbol{c}_i$. Each vector is a compressed representation of the corresponding cluster i at timestamp t. This allows to cope with memory restriction which are relevant in big data applications [3]. In Fig. 1(a), we depict a snapshot of a data stream at timestamp t_a, where three clusters are obtained, *i.e.*, $\mathcal{C}^{t_a} = \{\boldsymbol{c}_1, \boldsymbol{c}_2, \boldsymbol{c}_3\}$.

We consider non-stationary data streams with a time varying cluster structure. Therefore, the cluster model $\mathcal{C}^t$ changes over time. In Fig. 1, we illustrate four snapshots of a data stream, which show how its clusters evolve as data points flow in. Concretely, Fig. 1(b) corresponds to the emergence of $\boldsymbol{c}_4$. Then, $\boldsymbol{c}_3$ and $\boldsymbol{c}_4$ disappear at $t = t_c$, as shown in Fig. 1(c). When $t = t_d$, the cluster $\boldsymbol{c}_4$ recurs due to it is visited by data points again. The definitions of three typical types of cluster evolution are given as follows.

Emergence. Cluster emergence refers to the occurrence of a new cluster at timestamp t. In particular, a cluster $\boldsymbol{c}$ emerges at timestamp t if $\boldsymbol{c} \notin \mathcal{C}^1 \cup \mathcal{C}^2 \cup \cdots \cup \mathcal{C}^{t-1}$ and $\boldsymbol{c} \in \mathcal{C}^t$.

Disappearance. Cluster disappearance is defined as an existing cluster that is not visited by the recently arrived data points. Disappeared clusters need to be removed from the model to get the model non-redundant. A cluster $\boldsymbol{c}$ disappears if $\boldsymbol{c} \in \mathcal{C}^{t_0} \cap \mathcal{C}^{t_0+1} \cap \cdots \cap \mathcal{C}^{t-1}$ and $\boldsymbol{c} \notin \mathcal{C}^t$, where $1 \le t_0 < t$. That is, the cluster $\boldsymbol{c}$ is present in the cluster model before it is removed at timestamp t. Such a cluster is called a disappeared cluster.

Reoccurrence. Cluster reoccurrence means the situation where a previously disappeared cluster recurs at timestamp t. Formally, a cluster $\boldsymbol{c}$ recurs at timestamp t if $\boldsymbol{c} \in \mathcal{C}^{t_1} \cap \mathcal{C}^{t_1+1} \cap \cdots \cap \mathcal{C}^{t_2-1}$, $\boldsymbol{c} \notin \mathcal{C}^{t_2} \cup \mathcal{C}^{t_2+1} \cup \cdots \cup \mathcal{C}^{t-1}$, and $\boldsymbol{c} \in \mathcal{C}^t$, where $1 \le t_1 < t_2 < t$. That is, the cluster $\boldsymbol{c}$ is present from timestamp t_1 to timestamp $t_2 - 1$ and is removed from timestamp t_2. At timestamp t, it appears in the cluster model again.

A DSC algorithm assigns each arriving data point $\boldsymbol{x}^t$ to the cluster in $\mathcal{C}^{t-1}$ with the highest similarity. Most DSC algorithms need the so-called initialization phase which amounts to a static clustering problem. Assume the number of data points used for model initialization is denoted as T_0. Then the initialization phase can be formulated as the problem

$$\boldsymbol{m}^* = \underset{\boldsymbol{m}\in\mathbb{R}^{T_0}}{\arg\max} \left(\sum_{t=1}^{T_0} s\big(\boldsymbol{x}^t, \boldsymbol{c}_{m_t}\big) \right), \quad t \le T_0 \qquad (1)$$

where $\boldsymbol{c}_{m_t} \in \mathcal{C}^{T_0}$ and $\boldsymbol{m} = (m_1, \ldots, m_{T_0})$ denotes the mapping between data points $\{\boldsymbol{x}^t\}_{t=1}^{T_0}$ and $\mathcal{C}^{T_0}$. $s(\bullet)$ is the similarity between data point $\boldsymbol{x}^t$ and $\boldsymbol{c}_{m_t}$. Note that the initialization phase is a batch-mode clustering. The goal of the optimization is to find an optimal value $\boldsymbol{m}^*$ that maximizes the similarity between the T_0 data points and their respective clusters. We get an initial cluster model $\mathcal{C}^{T_0}$ through the initialization phase.

Based on the initial model, the DSC algorithm can perform real-time processing on each arriving data point $\boldsymbol{x}^t$ $(t > T_0)$. That is, the stream processing phase starts. The goal of this phase is to update the cluster model as the data points flow in, *i.e.*,

$$\mathcal{C}^t \leftarrow g(\mathcal{C}^{t-1}, \boldsymbol{x}^t), \quad t > T_0 \qquad (2)$$

where $g(\mathcal{C}^{t-1}, \boldsymbol{x}^t)$ denotes a function which updates $\mathcal{C}^{t-1}$ with the arriving data point $\boldsymbol{x}^t$ based on certain update rules. There are many kinds of update rules, which vary with different DSC methods. But a basic requirement is to find a suitable cluster for $\boldsymbol{x}^t$ in $\mathcal{C}^{t-1}$. For example, assigning $\boldsymbol{x}^t$ to a cluster in $\mathcal{C}^{t-1}$ with the smallest Euclidean distance.

The differences among different DSC algorithms are mainly reflected in the clustering method utilized in the initialization phase and the update strategy in the stream processing phase. Our focus is on designing novel detection strategies for the typical cluster evolution mentioned above, making the DSC algorithms able to handle evolving data streams and thus becoming more practical. We firstly provide a brief overview of STRAP and its basic AP model [33] here. AP is a message passing-based clustering algorithm proposed by Frey and Dueck [33]. It divides a dataset by finding a set of data points from the dataset, *i.e.*, exemplars, and associates each data point with one exemplar [34]. In contrast to many other clustering methods, AP does not require specification of the number of clusters nor any initialization of cluster centers. It achieves a stable performance with respect to the clustering results [31]. Note that AP is a batch-mode method, *i.e.*, it requires to have all data available before it starts processing.

Assume that there is a dataset $\mathcal{X}'$ which consists of I data points, *i.e.*, $\mathcal{X}' = \{\boldsymbol{x}_i\}_{i=1}^{I}$. Here, in order to avoid confusion with data stream $\mathcal{X}$, we use $\mathcal{X}'$ and i to denote the datasets and the index of its data point, respectively. Formally, the AP solves the following clustering problem [31], [34].

$$\boldsymbol{m}^* = \underset{\boldsymbol{m}\in\mathbb{R}^I}{\arg\min}\left(-\sum_{i=1}^{I} s(\boldsymbol{x}_i, \boldsymbol{x}_{m_i}) - \sum_{i=1}^{I} ln\,\chi(m_i)\right) \tag{3}$$

where $\boldsymbol{m} = (m_1, \ldots, m_I)$ is a mapping which maps each data point $\boldsymbol{x}_i$ to its exemplar $\boldsymbol{x}_{m_i}$. Here, $\chi(\bullet)$ is a penalty function which constrains each selected exemplar to be its own exemplar, *i.e.*,

$$\chi(m_i) = \begin{cases} 0, & \text{if } \exists\, m_j : \boldsymbol{x}_{m_j} = \boldsymbol{x}_i, \text{ and } \boldsymbol{x}_{m_i} \neq \boldsymbol{x}_i \\ 1, & \text{otherwise} \end{cases} \tag{4}$$

Function s evaluates the similarity between $\boldsymbol{x}_i$ and its exemplar $\boldsymbol{x}_{m_i}$. s is usually set to negative squared Euclidean distance, *i.e.*, $-d(\boldsymbol{x}_i, \boldsymbol{x}_{m_i})^2$, if $i \neq m_i$ [28], [33]. AP introduces a vector $\boldsymbol{p} = (p_1, \ldots, p_i, \ldots, p_I)$, the so-called preference parameter, to indicate the cost for each data point when it is chosen as an exemplar, that is $s(\boldsymbol{x}_i, \boldsymbol{x}_i) = p_i$. Generally, p_i is set to a constant which is the same for all data points.

STRAP extends AP which applies to batch data, to an online-processing of data streams [28]–[31]. Zhang *et al.* [29] firstly proposed a weighted AP (WAP) algorithm to make it possible for directly calculating the similarity between a single data point and a cluster which consists of a couple of data points. Based on WAP, STRAP was then proposed which contains three main steps, *i.e.*, initialization, emergence detection and model rebuild, as shown in Fig. 2. Concretely, AP is applied to compute the first exemplars and initialize the stream model. Then, as the stream flows in, each data $\boldsymbol{x}^t$ is compared with exemplars in the model. If $\boldsymbol{x}^t$ is judged as a normal one it would be assigned to a cluster. Otherwise, it will be judged as an outlier *i.e.*, the point that does not belong to any cluster and put into the outlier reservoir. The reservoir is checked at each timestamp to ensure if new clusters emerge. If emergence is detected, the model will be rebuilt to absorb the new clusters.

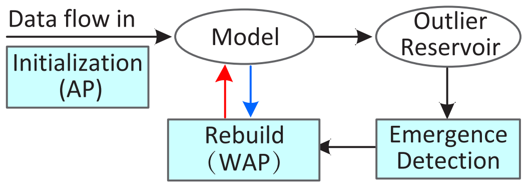

FIGURE 2. The framwork of STRAP algorithm.

STRAP has gained impressive performance compared to other data stream clustering methods [28]–[31]. The emergence of clusters is accurately detected under such framework. However, it cannot detect other typical types of cluster evolution (*e.g.,* disappearance and reoccurrence). In this paper, we mainly combine our evolution detection strategy with original STRAP to improve its ability when facing more complicated evolving data streams.

III. PROPOSED ALGORITHM

In this section, we introduce and discuss the ISTRAP algorithm. We first discuss the detection of the three cluster evolution patterns (*i.e.,* emergence, disappearance and reoccurrence) in Section III-A. Then we formulate the detailed algorithm. Finally, we discuss the parameter sensitivity in Section III-C.

A. DETECTION OF DYNAMIC CLUSTER EVOLUTION

In ISTRAP, we represent each cluster by a vector $\boldsymbol{c}_j = (\boldsymbol{e}_j^t, n_j^t, \sum_j^t, t_j, p_j^t)$, where

- $\boldsymbol{e}_j^t$ is the exemplar of cluster j;
- n_j^t is the total number of data points assigned to cluster j up to timestamp t;
- $\sum_j^t$ is the total similarity between the data points in cluster j and their exemplar $\boldsymbol{e}_j^t$;
- t_j is the last timestamp when a point was assigned to cluster j;
- p_j^t is a counter whose initial value is set as 0;

Depending on whether the clusters can effectively represent the current pattern of data streams, the ISTRAP divides the clusters into two states, *i.e.,* active and inactive, at each timestamp. Active state indicates that the cluster is still valid for the data stream at current timestamp because at least one data point in the data stream is assigned to it during a given interval. Inactive state reveals that the corresponding clusters are expired because they cannot represent the current pattern of the data stream with no recent point assigned in. Note that the state of a cluster at different timestamps may switch between the two mentioned states. At each timestamp, all the active clusters are saved in the cluster model $\mathcal{C}^t$, *i.e.,* $\mathcal{C}^t = \{\boldsymbol{c}_j\}_{j=1}^{k^t}$. Correspondingly, all the inactive clusters are saved in another set $\mathcal{C}'^t = \{\boldsymbol{c}_{j'}\}_{j'=1}^{k'^t}$, which we refer to as remove reservoir. Note that k'^t is the number of inactive clusters at timestamp t.

At each timestamp, the ISTRAP determines whether each new data point $\boldsymbol{x}^t$ is an outlier or a normal point. This determination is based on a following two-step process. First, the shortest Euclidean distance from $\boldsymbol{x}^t$ to $\mathcal{C}^{t-1}$ and $\mathcal{C}'^{t-1}$ is found, *i.e.*,

$$d_{min}^t = \min_{j=1\ldots k^{t-1}, j'=1\ldots k'^{t-1}} \{d(\boldsymbol{e}_j^{t-1}, \boldsymbol{x}^t), d(\boldsymbol{e}_{j'}^{t-1}, \boldsymbol{x}^t)\} \quad (5)$$

It should be mentioned that the shortest distance between $\boldsymbol{x}^t$ and $\mathcal{C}^{t-1}$ (or $\mathcal{C}'^{t-1}$) is defined as the the shortest distance between $\boldsymbol{x}^t$ and $\{\boldsymbol{e}_j^{t-1}\}_{j=1}^{k^{t-1}}$ (or $\{\boldsymbol{e}_{j'}^{t-1}\}_{j'=1}^{k'^{t-1}}$).

Then, d_{min}^t is compared with a preset threshold θ. The $\boldsymbol{x}^t$ will be considered as a normal point if d_{min}^t is smaller than θ. Otherwise, the $\boldsymbol{x}^t$ will be regarded as an outlier which is temporarily saved in a set $\mathcal{O}^t$, named outlier reservoir. At timestamp t, the number of outliers is denoted as r^t. Namely, $\mathcal{O}^t = \{\boldsymbol{x}^{t_1}, \boldsymbol{x}^{t_2}, \ldots, \boldsymbol{x}^{t_{r^t}}\}$.

Generally, r^t, t_j and p_j^t are related to the detection of emergence, disappearance and reoccurrence, respectively. All three types of evolution are checked at timestamp t.

Three evolution detectors have been specially designed in the algorithm to deal with three typical cluster evolution patterns, *i.e.*, emergence, disappearance and reoccurrence detectors. The emergence detector in ISTRAP is similar to that of STRAP. If parameter r^t is larger than a preset threshold α, the model will be rebuilt by applying WAP method [28] on exemplars in cluster model $\mathcal{C}^{t-1}$, *i.e.*, $\{\boldsymbol{e}_j^t\}_{j=1}^{k^{t-1}}$, and the outliers in $\mathcal{O}^t$.

For each active cluster $\boldsymbol{c}_j$, we calculate the time interval between the last timestamp at which it was visited and the current timestamp, *i.e.*,

$$\Delta_{tj} = t - t_j \quad (6)$$

Then, Δ_{tj} is compared with a threshold β to judge the state of cluster j at timestamp t. Namely, the cluster is marked with 'inactive' if its Δ_{tj} is larger than β. The inactive clusters are regarded as the clusters which have disappeared. The disappeared clusters will be removed from cluster model $\mathcal{C}^t$ to the remove reservoir $\mathcal{C}'^t$ to get $\mathcal{C}^t$ non-redundant.

The data streams arising in many important applications contain clusters which are reoccurring after inactive periods [21]. If the reoccurrence property could be utilized, one will directly regain the information of those temporarily disappeared clusters, thereby avoiding consuming much more time and computational cost compared with regenerating these clusters through the whole clustering process. In proposed ISTRAP, the disappeared clusters are stored in remove reservoir $\mathcal{C}'^t$. For all disappeared clusters, parameter $p_{j'}^t$ is applied to record the number of data points assigned to the cluster after their being judged disappeared. The parameter $p_{j'}^t$ is compared with a threshold γ at timestamp t to check if cluster j' has recurred. A disappeared cluster j' is regarded as a recurrent cluster if $p_{j'}^t$ is larger than γ. The whole dynamic cluster evolution detection process is elaborated in the following.

Algorithm 1 ISTRAP Algorithm

```
Input: $\mathcal{X} = \{x^t\}_{t=1}^N (N \to \infty), T_0, \theta, \alpha, \beta, \gamma$
 1: $r^0 = 0, \mathcal{O}^0 \longleftarrow \emptyset, \mathcal{C}'^0 \longleftarrow \emptyset$
 2: $\mathcal{C}^{T_0} \longleftarrow$ AP($x^1, x^2, \ldots, x^{T_0}$)
 3: for $t > T_0$ do
 4:     Calculate Equ. (5)
 5:     if $d_{min}^t < \theta$ then
 6:         if $d_{min}^t = \min_{j=1\ldots k^{t-1}}\{d(e_j^{t-1}, x^t)\}$ then
 7:             Update $c_j$
 8:         else
 9:             Update $c_{j'}$
10:             for $j'=1\ldots k'^t$ do
11:                 if $p_{j'}^t > \gamma$ then
12:                     $p_{j'}^t = 0$
13:                     $\mathcal{C} \longleftarrow c_{j'}$
14:                     Delete $c_{j'}$ in $\mathcal{C}'$
15:                 end if
16:             end for
17:         end if
18:     else
19:         $\mathcal{O} \longleftarrow x^t$
20:         $r^t = r^{t-1} + 1$
21:         if $r^t > \alpha$ then
22:             $\mathcal{C}^t \longleftarrow$ WAP($\mathcal{O}^t$,$\mathcal{C}^{t-1}$)
23:             $\mathcal{O}^t \longleftarrow \emptyset$
24:             $r^t = 0$
25:         end if
26:     end if
27:     for $j=1\ldots k^t$ do
28:         Calculate Equ. (6)
29:         if $\Delta_{tj} > \beta$ then
30:             $\mathcal{C}' \longleftarrow c_j$
31:             Delete $c_j$ in $\mathcal{C}$
32:         end if
33:     end for
34: end for
```

B. THE FRAMEWORK OF ISTRAP

The framework of ISTRAP is depicted in Fig. 3. From Fig. 3, we can find that four decision functions (one is for outlier detection and the rest are for the three kinds of evolution detection) and two reservoirs (denoted as outlier reservoir and remove reservoir) are included in the framework. All active clusters are saved in the cluster model (denoted as model) to represent the current pattern of the data stream. The remove reservoir collects all inactive clusters. Generally, the procedure of ISTRAP can be summarized as follows and the pseudocode of ISTRAP algorithm is illustrated in Algorithm 1.

- *Step 1:* The first bunch of data is processed by AP algorithm to extract representative data points as the initial exemplars and an initial model $\mathcal{C}^{T_0}$ is thus generated.
- *Step 2:* As the data points flow in, each data point will go through the outlier detection firstly. Specifically, the d_{min}^t

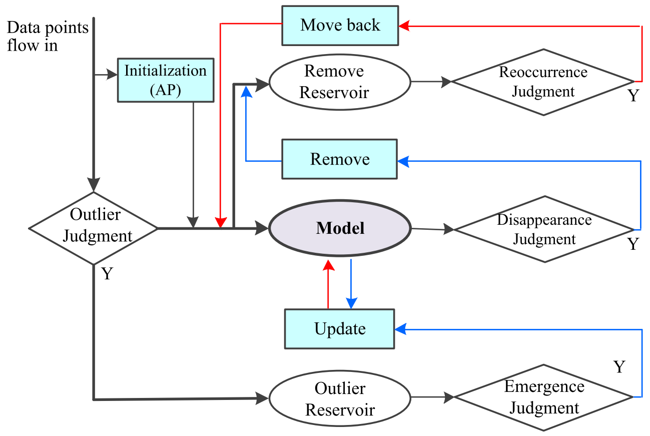

FIGURE 3. The framwork of ISTRAP algorithm.

is calculated by applying Equ. (5). The d_{min}^t is then compared with the given threshold θ. The data point could be assigned to the nearest exemplar of a certain cluster in model or remove reservoir if d_{min}^t is less than θ. Otherwise, it will be put into the outlier reservoir $\mathcal{O}$. Note that if the data point is assigned to a cluster in remove reservoir, the corresponding counter p_{j}^{t-1} of this cluster will then increase by 1.

- *Step 3:* Meanwhile, all clusters in remove reservoir $\mathcal{C}'$ are checked if some of them recur. The recurrent clusters will be moved back into the model $\mathcal{C}$.
- *Step 4:* The emergence criterion is triggered if new clusters need to be generated. Then, the model will be rebuilt from the current exemplars and the outlier reservoir, by using WAP algorithm [29].
- *Step 5:* All active clusters in model $\mathcal{C}$ will be checked if they are still active at timestamp t. The inactive clusters will be removed from the model into the remove reservoir $\mathcal{C}'$.
- *Step 6:* Back to *Step 2* until the stop condition is satisfied.

C. THE KEY PARAMETER SENSITIVITY ANALYSIS OF ISTRAP

There are several key parameters in ISTRAP which can affect the clustering quality, such as θ, α, β, and γ. Here, some qualitative analysis are given to show how these parameters impact on the performance of the algorithm.

Firstly, θ has an indirect effect on the detection of three kinds of evolution by controlling the number of outliers. Generally, a smaller θ results in that a point is easier to be judged as an outlier. Thus, more clusters tend to be inactive and more frequent rebuilt of the model will be caused. A recommended value is the average Euclidean distance between data points and exemplars in the initial model [31].

Secondly, α represents the threshold used in emergence detection. Emergence detector will trigger the rebuilt of the model. Consequently, α affects the stability and processing time of the ISTRAP.

Thirdly, β means the maximal duration tolerance that an active cluster in model is not visited by the arriving data points. Hence, a larger β will cause less possibility that clusters are judged as inactive.

The parameter γ represents the minimum number of data points assigned if a recurrent cluster needs to be sent to current model. The smaller γ is, the more recurrent clusters are found.

It should be pointed that ISTRAP can be viewed as a generalization model of the standard STRAP with two new parameters β and γ introduced. STRAP can be seen as a particular case of ISTRAP when β and γ equal to infinity.

IV. SIMULATION

We verify the effectiveness of ISTRAP in this section. Firstly, the data sets we used are introduced and then the performance of ISTRAP on dealing with typical evolving data streams is tested. Finally, the sensitivity against β and γ of ISTRAP is analyzed.

A. DATA SETS

We evaulate the performance of ISTRAP (Algorithm 1) using numerical experiments involving both artificial and real-world datasets. In particular, we have generated two artifial data streams and four data streams based on the MNIST database [35]. The datasets used in our experiments are described in Table 1.

TABLE 1. General information of evolving data streams occupied in experiments.

Data Stream	n	N	Clusters	Evolution type
AS 1	3	17200	12	Emergence Disappearance
AS 2	3	22000	12	Emergence Disappearance Reoccurrence
MNIST 1	784	20000	10	Emergence Disappearance
MNIST 2	784	20000	10	Emergence Disappearance Reoccurrence
MNIST 3	784	20000	10	Emergence Disappearance
MNIST 4	784	20000	10	Emergence Disappearance Reoccurrence

Artificial Data Streams: We created two artificial data streams, denoted AS 1 and AS 2, which contain different types of evolution. The data spatial distributions of AS 1 and AS 2 in their feature spaces are depicted in Fig. 4 and Fig. 5, respectively. The data points of two data streams are all derived from 12 clusters which are located in 3-dimensional real space. As shown in Table 1, AS 1 consists of 17200 data points and AS 2 has 22000 data points in total.

Real-World Data Streams: The MNIST (modified NIST) database is selected to test the performance of ISTRAP. We used the MNIST database provided by LeCun [35]. The MNIST database contains 70000 images of handwritten digits from 0 to 9. All digits have been size-normalized and centered to 28 × 28 gray-level images, so each image can be denoted as a 784-dimensional

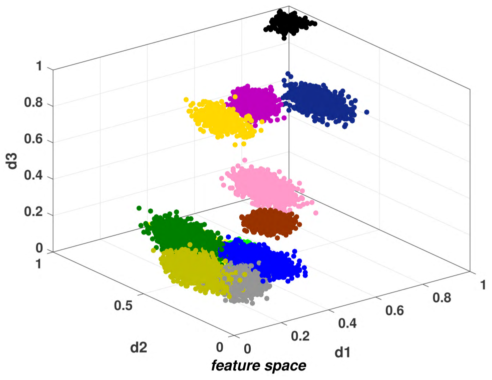

FIGURE 4. Spatial distribution of AS 1 in feature space.

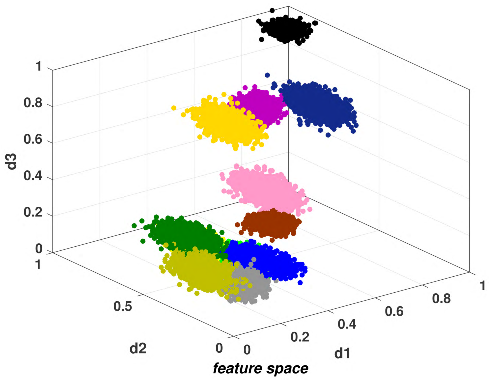

FIGURE 5. Spatial distribution of AS 2 in feature space.

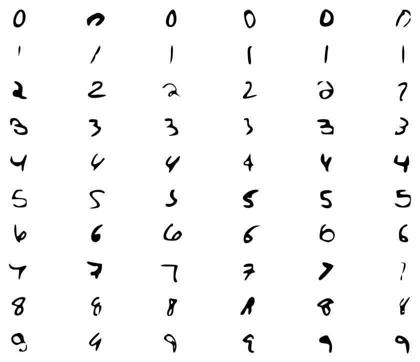

FIGURE 6. Sample images in MNIST database.

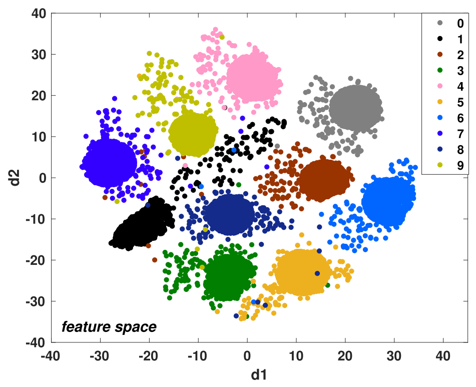

FIGURE 7. Spatial distribution in 2-dimensional feature space of the data set selected from MNIST database.

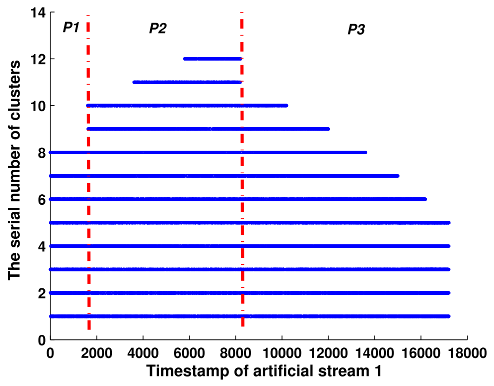

FIGURE 8. Temporal distribution and evolving property of AS 1.

vector [36]. Some samples in this database are illustrated in Fig. 6, provided by Niu and Suen [41]. It has been shown that the MNIST dataset is actually located a much lower-dimensional feature space. Hence, the t-SNE (t-distributed stochastic neighbor embedding) approach [37]–[40] is applied to reduce dimensionality from 784 to 2. We select 2000 data points (images) for each digit to form a data set. The spatial distribution in 2-dimensional feature space of the selected data set is depicted in Fig. 7. The selected data set is re-arranged to be added to the evolution property. As shown in Table 1, the MNIST data streams 1 to 4 are four evolving data streams derived from the selected data set (denoted as MNIST streams 1 to 4 in this paper). In particular, MNIST streams 3 and 4 are two data streams which are applied to analyze the parameter sensitivity of ISTRAP.

B. DETECTING DISAPPEARED CLUSTERS

We first apply ISTRAP to AS 1 and MNIST stream 1 to test its effectiveness for emergence and disappearance detection on artificial and real-world evolving data streams. The results of the standard STRAP will be provided for comparison. The data spatial distributions and evolving properties of AS 1 and MNIST stream 1 have been depicted in Fig. 8 and Fig. 9 respectively. As can be seen from Fig. 8 and Fig. 9, the two evolving streams have experienced three different evolving phases, *i.e.* initialization phase (denoted as P1), new clusters emerge in the following phase (P2), and some clusters successively disappear in the final phase (P3).

For a fair comparison, the parameters of ISTRAP and STRAP, for example, preference $\boldsymbol{p}$, α and θ, are set to be the same. The parameters settings are given as follows. We set $\alpha = 130$, $\beta = 150$ for both streams, $\theta = 0.1$ for artificial streams, $\theta = 12$ for MNIST streams. The comparison results between two algorithms are shown in Fig. 10 and Fig. 11.

From Fig. 10 and Fig. 11, we find that both ISTRAP and STRAP can detect the emergence (in P2) cases accurately. However, the disappearance (in P3) is not detected by STRAP

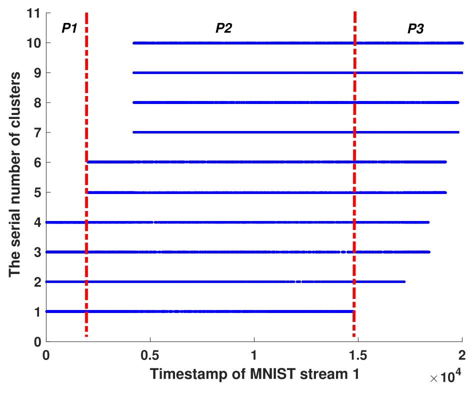

FIGURE 9. Temporal distribution and evolving property of MNIST stream 1.

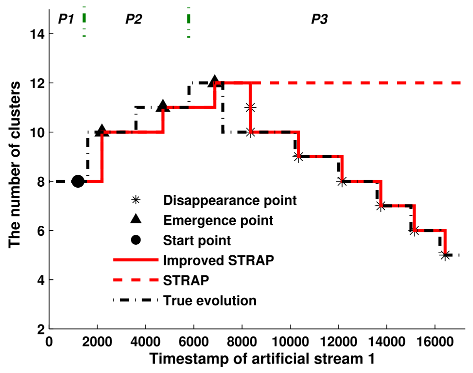

FIGURE 10. The results of ISTRAP and STRAP applied to AS 1.

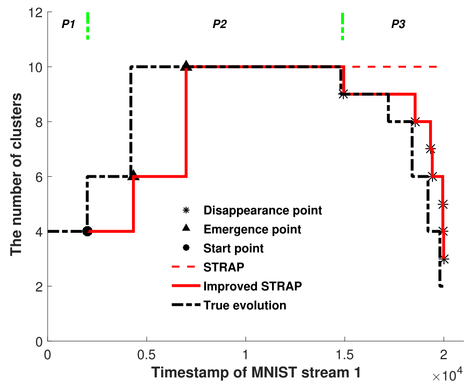

FIGURE 11. The results of ISTRAP and STRAP applied to MNIST stream 1.

as the number of clusters does not decrease with the disappearance of clusters, which reveals that STRAP can only find the emergence of clusters. In contrast, the ISTRAP accurately detects the disappearance of clusters.

C. DETECTING RECURRING CLUSTERS

We test the recurring cluster detection performance of ISTRAP using AS 2 and MNIST stream 2. The data spatial distributions and the evolving properties of these streams are shown in Fig. 12 and Fig. 13. As can be seen from Fig. 12 and Fig. 13, AS 2 and MNIST stream 2 can be divided into 4 phases according to their evolving property, *i.e.,* emergence in the first phase (denoted as P1), disappearance and reoccurrence in the following three phases (P2-P4). For comparison, STRAP is also tested on the data streams. It should be pointed out that all parameters remain the same with those of the experiments in the above sub-section.

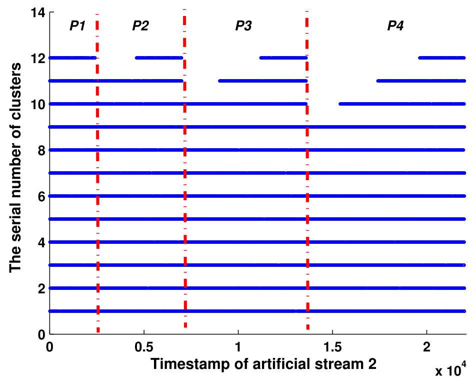

FIGURE 12. Temporal distribution and evolving property of AS 2.

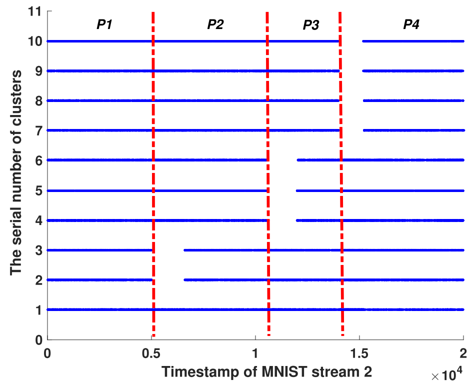

FIGURE 13. Temporal distribution of and evolving property of MNIST stream 2.

In Fig. 14 and Fig. 15, we illustrate the results of ISTRAP and STRAP on AS 2 and MNIST stream 2, respectively. As expected, we find that STRAP is unable to detect reoccurrence. In contrast, the ISTRAP shows its powerful effectiveness in tracking the reoccurrence of the clusters with a little delay after the true evolution curve. Thus we find that ISTRAP is able to accurately detect the disappearance and reoccurrence of evolving streams.

D. PARAMETER ANALYSIS

The key parameters θ, α, β, and γ have impact on the performance of ISTRAP. The θ, α are analyzed in detail in [28]–[31]. Here we mainly demonstrate how β and γ affect the algorithm according to simulation results.

We test the sensitivity of Algorithm 1 on the precise choice for β by applying it to MNIST stream 3. MNIST stream 3 consists of 20000 data points in total and the main evolution of the stream is the disappearance. Specifically, there are

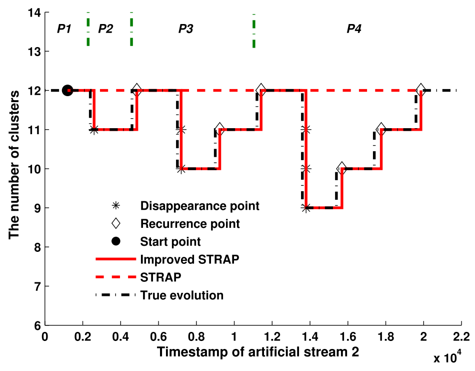

FIGURE 14. The results of ISTRAP and STRAP applied to AS 2.

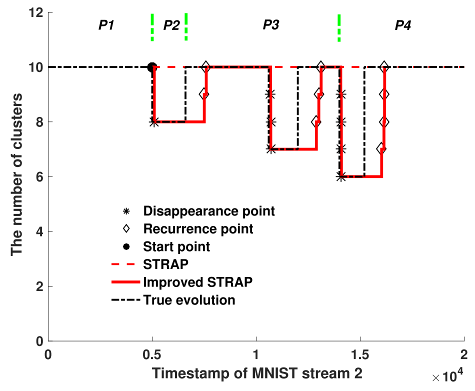

FIGURE 15. The results of ISTRAP and STRAP applied to MNIST stream 2.

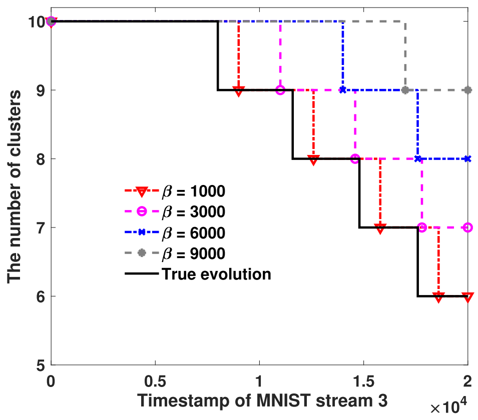

FIGURE 16. Disappearance detection results by ISTRAP under four typical β settings.

10 clusters in the beginning and then four clusters disappear one by one after reaching its minimum (six active clusters in the final phase). Fig. 16 shows the results under four typically different β settings.

From Fig. 16, the algorithm detects 4, 3, 2 and 1 times disappearance when the β is set 1000, 3000, 6000 and 9000 respectively. The larger β is set, the less possibility that the algorithm can detect all the disappearance.

Similar results can also be found when we apply the proposed ISTRAP on MNIST stream 4 to test its sensitivity to γ.

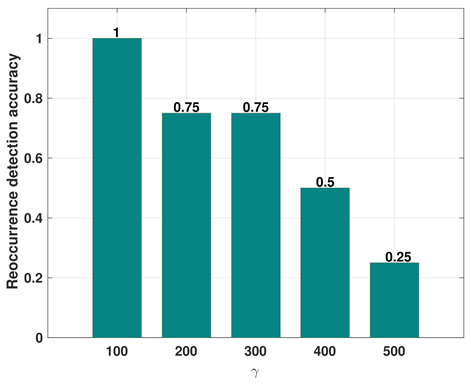

FIGURE 17. Reocurrence detection accuracy under five typical γ settings.

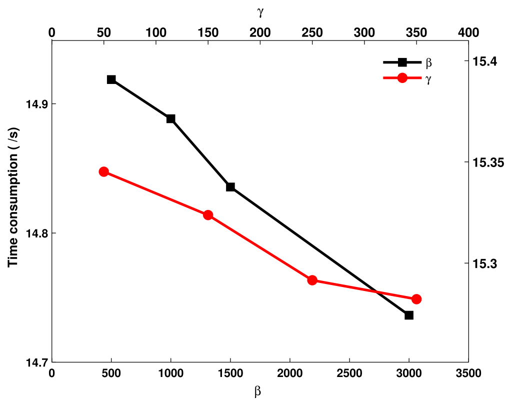

FIGURE 18. Time consumption of processing MNIST stream 4 under different β and γ settings.

MNIST stream 4 consists of 20000 data points in total and the main evolution of the stream is the reoccurrence. There are 4 times of reoccurrence existing in MNIST stream 4. We test the accuracy of the reoccurrence detection of the proposed ISTRAP under different γ values. As depicted in Fig. 17, the reoccurrence detection accuracy is gained by calculating the ratio of the number of reoccurrences detected to the true number of reoccurrences. We can observe that larger γ is less sensitive in monitoring the reoccurrence of clusters.

Smaller values of β and γ can ensure that the proposed method is more sensitive to the cluster evolution. However, the parameters cannot be set too small when we take the time consumption into account. in Fig. 18, we show the time consumption under different parameter settings of ISTRAP when processing MNIST stream 4. Concretely, we fix $\gamma = 50$ firstly and increase β from 600 to 3000. The time consumption curve shows a downward trend as a whole.When we fix $\beta = 600$ and increase γ from 50 to 350, the corresponding time consumption curve still shows a decreasing trend. This is because smaller parameter value makes the scheme more sensitive to the cluster evolution, resulting in more reactions or even false-detection. The scheme needs more time to react to the evolution. By contrast, larger parameter values make the scheme be more insensitive to the cluster evolution. Hence, less time will be consumed.

V. CONCLUSION

This paper proposes a novel data stream clustering approach, named ISTRAP, which can detect and monitor the cluster evolution *i.e.* emergence, disappearance, and reoccurrence. An integrated evolution detection framework is proposed which ensures that new emerging clusters are timely added to the current model, outdated clusters are removed and recurrent clusters are efficiently detected rather than being treated as novel clusters. Simulation results on artificial and real-world data streams show that ISTRAP is able to detect the evolution of data streams. The ISTRAP still suffers from some drawbacks. For example, some parameter settings are still needed to be set manually. Meanwhile, as a Euclidean distance-based algorithm, ISTRAP cannot handle well with high-dimensional data streams. Potential future work would be to expand ISTRAP to overcome these difficulties.

ACKNOWLEDGMENT

The authors would like to thank the Machine learning for Big Data research group in Department of Computer Science, School of Science, Aalto University.

REFERENCES

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

[2] M. Mousavi, A. A. Bakar, and M. Vakilian, "Data stream clustering algorithms: A review," *Int. J. Adv. Soft Comput. Appl.*, vol. 7, no. 3, p. 13, 2015.

[3] J. A. Silva, E. R. Faria, R. C. Barros, E. R. Hruschka, A. C. P. L. F. de Carvalho, and J. Gama, "Data stream clustering: A survey," *ACM Comput. Surv.*, vol. 46, no. 1, Oct. 2013, Art. no. 13.

[4] S. Guha, A. Meyerson, N. Mishra, R. Motwani, and L. O'Callaghan,"Clustering data streams: Theory and practice," *IEEE Trans. Knowl. Data Eng.*, vol. 15, no. 3, pp. 515–528, May/Jun. 2003.

[5] J. Gama, *Knowledge Discovery From Data Streams*. Boca Raton, FL, USA: CRC Press, 2010.

[6] S. Ramírez-Gallego, B. Krawczyk, S. García, M. Woźniak, and F. Herrera, "A survey on data preprocessing for data stream mining: Current status and future directions," *Neurocomputing*, vol. 239, pp. 39–57, May 2017.

[7] K. Gao and Y. Zhu, "Deep data stream analysis model and algorithm with memory mechanism," *IEEE Access*, vol. 5, pp. 84–93, 2017.

[8] B. Krawczyk, L. L. Minkub, J. Gama, J. Stefanowski, and M. Woźniak, "Ensemble learning for data stream analysis: A survey," *Inf. Fusion*, vol. 37, pp. 132–156, Sep. 2017.

[9] S. Ding, F. Wu, J. Qian, H. Jia, and F. Jin, "Research on data stream clustering algorithms," *Artif. Intell. Rev.*, vol. 43, no. 4, pp. 593–600, Apr. 2015.

[10] A. Amini, T. Y. Wah, and H. Saboohi, "On density-based data streams clustering algorithms: A survey," *J. Comput. Sci. Technol.*, vol. 29, no. 1, pp. 116–141, 2014.

[11] C.-D. Wang, J.-H. Lai, D. Huang, and W.-S. Zheng, "SVStream: A support vector-based algorithm for clustering data streams," *IEEE Trans. Knowl. Data Eng.*, vol. 25, no. 6, pp. 1410–1424, Jun. 2013.

[12] C. C. Aggarwal, J. Han, J. Wang, and P. S. Yu, "On high dimensional projected clustering of data streams," *Data Mining Knowl. Discovery*, vol. 10, no. 3, pp. 251–273, May 2005.

[13] A. Zhou, F. Cao, Y. Yan, C. Sha, and X. He, "Distributed data stream clustering: A fast EM-based approach," in *Proc. IEEE 23rd Int. Conf. Data Eng.*, Apr. 2007, pp. 736–745.

[14] P. Wang, H. Wang, X. Wu, W. Wang, and B. Shi, "A low-granularity classifier for data streams with concept drifts and biased class distribution," *IEEE Trans. Knowl. Data Eng.*, vol. 19, no. 9, pp. 1202–1213, Sep. 2007.

[15] G. I. Webb, R. Hyde, H. Cao, H. L. Nguyen, and F. Petitjean, "Characterizing concept drift," *Data Mining Knowl. Discovery*, vol. 30, no. 4, pp. 964–994, 2016.

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

[17] M. Oliveira and J. Gama, "MEC—Monitoring clusters' transitions," in *Proc. 5th Starting AI Researchers' Symp. (STAIRS)*, vol. 222, 2010, pp. 212–224.

[18] M. Spiliopoulou, I. Ntoutsi, Y. Theodoridis, and R. Schult, "The MONIC framework for cluster transition detection," in *Proc. 5th Hellenic Data Manage. Symp. (HDMS)*, Thessaloniki, Hellas, 2006.

[19] P. Lee, L. V. S. Lakshmanan, and E. E. Milios, "Incremental cluster evolution tracking from highly dynamic network data," in *Proc. IEEE 30th Int. Conf. Data Eng.*, Mar./Apr. 2014, pp. 3–14.

[20] J. de Andrade Silva, E. R. Hruschka, and J. Gama, "An evolutionary algorithm for clustering data streams with a variable number of clusters," *Expert Syst. Appl.*, vol. 67, pp. 228–238, Jan. 2017.

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

[22] J. Xu, F. Li, K. Chen, F. Zhou, J. Choi, and J. Shin, "Dynamic chameleon authentication tree for verifiable data streaming in 5G networks," *IEEE Access*, vol. 5, pp. 26448–26459, 2017.

[23] L. Shi, L. Liu, Y. Wu, L. Jiang, and J. Hardy, "Event detection and user interest discovering in social media data streams," *IEEE Access*, vol. 5, pp. 20953–20964, 2017.

[24] J. Gama, I. Žliobaitė, A. Bifet, M. Pechenizkiy, and A. Bouchachia, "A survey on concept drift adaptation," *ACM Comput. Surv.*, vol. 46, no. 4, Apr. 2014, Art. no. 44.

[25] C. C. Aggarwal, J. Han, J. Wang, and S. Y. Philip, "A framework for clustering evolving data streams," in *Proc. 29th VLDB Conf.*, 2003, pp. 81–92.

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

[27] P. Kranen, I. Assent, C. Baldauf, and T. Seidl, "The ClusTree: Indexing micro-clusters for anytime stream mining," *Knowl. Inf. Syst.*, vol. 29, no. 2, pp. 249–272, Nov. 2011.

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

[29] X. Zhang, C. Furtlehner, and M. Sebag, "Data streaming with affinity propagation," in *Proc. Joint Eur. Conf. Mach. Learn. Knowl. Discovery Databases*, 2008, pp. 628–643.

[30] X. Zhang, C. Furtlehner, J. Perez, C. Germain-Renaud, and M. Sebag, "Toward autonomic grids: Analyzing the job flow with affinity streaming," in *Proc. 15th ACM SIGKDD Int. Conf. Knowl. Discovery Data Mining*, 2009, pp. 987–996.

[31] X. Zhang, "Contributions to large scale data clustering and streaming with affinity propagation. Application to autonomic grids," Ph.D. dissertation, Dept. Comput. Sci., Univ. Paris-Sud, Orsay, France, 2010.

[32] D. Dueck, "Affinity propagation: Clustering data by passing messages," Ph.D. dissertation, Dept. Elect. Comput. Eng., Univ. Toronto, Toronto, ON, Canada, 2009.

[33] B. J. Frey and D. Dueck, "Clustering by passing messages between data points," *Science*, vol. 315, no. 5814, pp. 972–976, Feb. 2007.

[34] I. Givoni, C. Chung, and B. J. Frey. (Feb. 2012). "Hierarchical affinity propagation." [Online]. Available: https://arxiv.org/abs/1202.3722

[35] Y. LeCun, L. Bottou, Y. Bengio, and P. Haffner, "Gradient-based learning applied to document recognition," *Proc. IEEE*, vol. 86, no. 11, pp. 2278–2324, Nov. 1998.

[36] Z. Zhang, T. W. S. Chow, and M. Zhao, "Trace ratio optimization-based semi-supervised nonlinear dimensionality reduction for marginal manifold visualization," *IEEE Trans. Knowl. Data Eng.*, vol. 25, no. 5, pp. 1148–1161, May 2013.

[37] L. van der Maaten and G. Hinton, "Visualizing non-metric similarities in multiple maps," *Mach. Learn.*, vol. 87, no. 1, pp. 33–55, Apr. 2012.

[38] L. van der Maaten, "Accelerating t-SNE using tree-based algorithms," *J. Mach. Learn. Res.*, vol. 15, no. 1, pp. 3221–3245, Oct. 2014.

[39] L. van der Maaten, "Learning a parametric embedding by preserving local structure," in *Proc. 12th Int. Conf. Artif. Intell. Statist.*, 2009, pp. 384–391.

[40] L. van der Maaten and G. Hinton, "Visualizing data using t-SNE," *J. Mach. Learn. Res.*, vol. 9, pp. 2579–2605, Nov. 2008.

[41] X. X. Niu and C. Y. Suen, "A novel hybrid CNN–SVM classifier for recognizing handwritten digits," *Pattern Recognit.*, vol. 45, no. 4, pp. 1318–1325, Apr. 2012.

JINPING SUI was born in Dunhua, Jilin, China, in 1990. He received the B.S. degree in communication engineering from Northeastern University in 2013, and the M.S. degree in information and communication of engineering from the College of Electronic Science, National University of Defense Technology (NUDT), Changsha, China, in 2015, where he is currently pursuing the Ph.D. degree. He is also a Visiting Doctoral Student with the Machine Learning for Big Data research group, Department of Computer Science, Aalto University. His research interests include data streaming and machine learning.

ZHEN LIU was born in Taixing, Jiangsu, China, in 1983. He received the B.S. degree from Zhejiang University, Hangzhou, China, in 2006, and the Ph.D. degree from the National University of Defense Technology (NUDT), Changsha, China, in 2013. Since 2013, he has been a Lecturer with the College of Electronic Science, NUDT. His research interests include signal processing, compressed sensing, and machine learning.

ALEXANDER JUNG received the Diplom-Ingenieur and Dr.techn. degrees in electrical engineering from the Vienna University of Technology, Vienna, Austria, in 2008 and 2011, respectively. Since 2015, he has been an Assistant Professor with the Department of Computer Science, Aalto University. His research interests include big data over networks and semi-supervised learning.

LI LIU received the B.S. degree in communication engineering, the M.S. degree in photogrammetry and remote sensing, and the Ph.D. degree in information and communication engineering from the National University of Defense Technology (NUDT), China, in 2003, 2005, and 2012, respectively. She joined as a faculty member at NUDT in 2012, where she is currently an Associate Professor with the College of System Engineering. During her Ph.D. study, she spent over two years as a Visiting Student with the University of Waterloo, Canada, from 2008 to 2010. From 2015 to 2016, she spent ten months visiting the Multimedia Laboratory, The Chinese University of Hong Kong. Since 2016, she has been visiting the Machine Vision Group, University of Oulu, Finland. Her papers have currently over 1500 citations in Google Scholar. Her current research interests include texture analysis, image classification, object detection, and scene understanding. She was the Co-Chair of International Workshops at ACCV2014, CVPR2016, ICCV2017, and ECCV2018. She was a Guest Editor of special issues for the IEEE TPAMI, IJCV, and *Neurocomputing*. She is currently an Associate Editor of the *Visual Computer Journal*.

XIANG LI was born in Liuyang, Hunan, China, in 1967. He received the B.S. degree from Xidian University, Xi'an, China, in 1989, and the Ph.D. degree from the National University of Defense Technology (NUDT) in 1998. He is currently a Professor with the College of Electronic Science, NUDT. His research interests include signal processing, automation target recognition, and machine learning.