A Novel Data-driven Approach to Autonomous Fuzzy Clustering

Xiaowei Gu, Qiang Ni, and Guolin Tang

Abstract—In this paper, a new data-driven autonomous fuzzy clustering (AFC) algorithm is proposed for static data clustering. Employing a Gaussian-type membership function, AFC firstly uses all the data samples as micro-cluster medoids to assign memberships to each other and obtains the membership matrix. Based on this, AFC chooses these data samples that represent local models of data distribution as cluster medoids for initial partition. It then continues to optimize the cluster medoids iteratively to obtain a locally optimal partition as the algorithm output. Moreover, an online extension is introduced to AFC enabling the algorithm to cluster streaming data chunk-by-chunk in a “one pass” manner. Numerical examples based on a variety of benchmark problems demonstrate the efficacy of the AFC algorithm in both offline and online application scenarios, proving the effectiveness and validity of the proposed concept and general principles.

Index Terms—data-driven, fuzzy clustering, locally optimal partition, medoids, pattern recognition.

I. INTRODUCTION

CLUSTERING is a commonly-used unsupervised machine learning technique for statistical data analysis [1]. Its main objective is to group data into clusters such that data samples belonging to the same cluster share higher similarity than those belonging to other clusters. Thus, clustering is a tool of great importance for disclosing the underlying patterns and unveiling the natural geometry of data [2]. Due to the great demand, it has been a hot research area over the past decades, and a wide variety of clustering algorithms have been introduced and implemented for real-world applications, such as data mining [3] and image segmentation [4].

Generally, clustering algorithms mainly utilize the statistical properties and mutual distances of data for clustering. Different algorithms usually produce different partitions for the same data because of their unique operating mechanisms. Based on the way data samples are assigned to clusters, existing clustering algorithms can be broadly divided into two major categories, namely, 1) crisp clustering and 2) fuzzy clustering [5].

Crisp clustering algorithms assign each individual sample to only one cluster. Thus, clusters obtained by these algorithms are mutually exclusive. The majority of existing clustering algorithms in the literature belong to the first category. The classical crisp clustering algorithms include, but are not limited to, k-means [6], k-medoids [7], DBSCAN [8], BIRCH [9], affinity propagation [10], Gaussian mixture model [11], mean-shift [12] and density peak [13]. In recent years, many advanced clustering algorithms of this category have been proposed in literature, such as Gaussian density distance [14], local gravitation clustering [15], autonomous data partitioning [16] and fast density peak [17], etc.

Different from crisp clustering, fuzzy clustering algorithms assign each data sample to every cluster with a membership coefficient [18]. Fuzzy clustering algorithms naturally produce overlapping partitions, and they have shown better capability in capturing the data structure thanks to this additional flexibility [19]. The most well-known and widely used fuzzy clustering algorithm is the fuzzy c-means (FCM) algorithm proposed by Dunn [20] and Bezdek [21]. There have been many variants of FCM algorithm introduced in the past decades [22]. For example, the fuzzy c-medoids (FCMd) algorithm was developed on the basis of FCM by combining with the idea of medoids [23]. The kernel FCM algorithm was introduced by utilizing kernel functions to map original data into a higher dimensional kernel Hilbert space such that data can be clustered more easily [24], [25]. A generalized multiple-kernel FCM algorithm that employs a linear combination of multiple kernels was proposed in [26]. A regularized FCM method was presented in [27], which modifies the objective function of FCM by incorporating a graph regularization term constructed based on data correlations. The FCM algorithm was modified in [28] to enhance its ability of handling outliers by involving a robust loss function and a penalty term adding sparseness to the memberships of each individual sample with respect to different clusters. A FCMdd algorithm that employs the weighted sum of pairwise distances per attribute type as the dissimilarity measure was proposed in [5] for heterogeneous data clustering. By injecting data affinity into fuzzy clustering, a membership affinity regularized FCM algorithm was proposed in [29] for handling data with complex distribution. However, similar to some classical crisp clustering algorithms such as k-means, it is a challenging task for the FCM algorithm and its variants to self-determine the optimal number of clusters without prior knowledge of the problems [19]. Although there have been a few FCM variants that are capable of estimating the number of clusters through an iterative searching process, such algorithms are highly computationally expensive and their performance is subject to externally controlled parameters [19], [30], [31], [32]. Very importantly, these algorithms are not applicable...
for streaming data clustering. They have to repeat the entire clustering process again if new data samples are given.

In this paper, a new data-driven autonomous fuzzy clustering (AFC) algorithm is proposed. The proposed AFC algorithm adopts the well-known partitioning around medoids (PAM) strategy [33]. It firstly treats every sample in the data space as a micro-cluster with itself as the cluster medoid. Then, the algorithm calculates the memberships of each individual data sample with respect to all the micro-clusters and obtains the membership matrix. Based on the membership matrix, AFC selects out a much smaller number of highly representative samples as cluster medoids. Such samples represent the local models of data distribution and can be used to achieve a good initial partition of the data space. The proposed algorithm then continues to optimize these cluster medoids iteratively to achieve the locally optimal partition. Furthermore, a critical extension is introduced to AFC for online application scenarios by allowing the proposed algorithm to cluster streaming data on a chunk-by-chunk basis. With this extension, AFC is able to extract cluster medoids from each individual chunk of the data streams and fuse them with the cluster medoids identified from historical chunks together to produce the final outcome. To achieve more compact data partition, AFC only keeps the more representative and distinctive cluster medoids during the fusion process, while other cluster medoids that represent similar patterns are removed to avoid redundancy.

To summarize, key features that set the proposed AFC algorithm apart from existing approaches include:

1) a Gaussian type membership function to control the degree of fuzziness via a self-adjusting kernel width derived based on mutual distances of data;
2) an approach to self-determine the number of clusters and achieve high-quality initial data partition without computationally expensive searching, and;
3) a chunk-by-chunk clustering mechanism that enables the algorithm to cluster data streams with high computation-and-memory-efficiency.

The remainder of this paper is organized as follows. Section II describes the technical details of the AFC algorithm. The extension to streaming data clustering is presented in Section III. Section IV presents the detailed computational complexity analysis of the proposed algorithm. Numerical examples with discussions are given by Section V. Section VI concludes this paper and gives directions for future work.

II. PROPOSED ALGORITHM

This section describes the proposed AFC algorithm in detail. First of all, let \( \{ x \} = \{ x_1, x_2, ..., x_K \} \) be a static data set in a real \( N \)-dimensional data space, \( \mathbb{R}^N \), where \( x_k = [x_{k,1}, x_{k,2}, ..., x_{k,N}]^T \in \mathbb{R}^N \) (\( k = 1, 2, ..., K \)); the subscript \( k \) stands for the time instance at which \( x_k \) is observed. In this paper, the commonly used Euclidean distance is employed as the default distance measure, namely,

\[ \| x_i - x_j \| = \sqrt{\sum_{n=1}^{N} (x_{i,n} - x_{j,n})^2}. \]

A. Objective Function

The objective of the proposed algorithm is to partition the \( K \) data samples into \( C \) clusters. Note that \( C \) is not known a priori. Each cluster is represented by a medoid, \( p_i \) (\( i = 1, 2, ..., C \)). A medoid is an actual data sample at the cluster closest to the mean. Typically, cluster medoids can provide more intuitive information about the data than cluster means, which are used by many clustering approaches. This is because cluster means usually do not physically exist; meanwhile, cluster medoids are the most representative samples in the data space.

For data partitioning, the following objective function and optimization problem is proposed in this paper:

\[ J(U, P) = \sum_{k=1}^{K} \sum_{i=1}^{C} \bar{\mu}_{i,k}^2 \| x_k - p_i \|^2 \]  \( (1) \)

where \( U = [\bar{\mu}_{i,k}]_{i=1:K, k=1:C} \) is the membership matrix; \( P = [p_i]_{i=1:C} \) is the medoid matrix; \( \bar{\mu}_{i,k} \) is the normalized fuzzy membership of \( x_k \) in the \( i \)th cluster represented by the medoid, \( p_i \), subject to:

\[ \bar{\mu}_{i,k} > 0 \quad \text{and} \quad \sum_{i=1}^{C} \bar{\mu}_{i,k} = 1 \]  \( (2) \)

The normalized fuzzy membership, \( \bar{\mu}_{i,k} \) is obtained using Eqn. (3).

\[ \bar{\mu}_{i,k} = \frac{\mu(p_i, x_k, \sigma_G)}{\sum_{j=1}^{K} \mu(p_j, x_k, \sigma_G)} \]  \( (3) \)

where \( \mu(p_i, x_k, \sigma_G) \) is the Gaussian type fuzzy membership defined by Eqn. (4) [34].

\[ \mu(p_i, x_k, \sigma_G) = e^{-\frac{\| p_i - x_k \|^2}{\sigma_G^2}} \]  \( (4) \)

Gaussian kernel function is the most widely used type of membership functions by existing fuzzy rule-based systems. Gaussian is more compact and has stronger capability to neutralize the negative effects of outliers than other commonly used kernel functions, such as Cauchy and triangular, etc. Thus, using Gaussian type membership function can effectively improve the robustness of the AFC algorithm. However, one may choose to use other kernel functions instead as the best-performing membership function is always different from case to case depending on the nature of data.

The main aim of AFC is to identify a set of cluster medoids, \( P \) from these empirically observed data samples by minimizing Eqn. (1). It is worth noting that Eqn. (1) is a simplified version of the commonly used objective function by the FCM algorithm [21], [22], [35]. Unlike the conventional FCM algorithm, which controls the degree of fuzziness by employing a fuzzy weighting exponent, AFC controls the degree of fuzziness by adjusting the kernel width. The kernel width, \( \sigma_G \) is derived directly from data based on their mutual distances and the level of granularity (\( G \)) controlled by users [36]:

\[ \sigma^2_g = \frac{1}{\sum_{i=1}^{K-1} \sum_{j=i+1}^{K} \sum_{l=1}^{K-1} \sum_{m=l+1}^{K} w_{g,i,l} w_{g,j,m} \| x_i - x_j \|^2} \]  \( (5) \)

where \( g = 1, 2, ..., G \); \( G \) can be any non-negative integer chosen by users and \( w_{g,i,j} = \begin{cases} 1, & \text{if } \| x_i - x_j \| \leq \sigma_{g-1} \\ 0, & \text{else} \end{cases} \)

There is \( \sigma^2_0 = \frac{1}{K(K-1)} \sum_{i=1}^{K-1} \sum_{j=i+1}^{K} \| x_i - x_j \|^2 \).
The self-adjusting kernel width, $\sigma_G$ represents the radius of zone of influence around each cluster medoid under the $G^{th}$ level of granularity, and it can also be viewed as the maximum distance between any two neighbouring data samples under the $G^{th}$ level of granularity.

The main procedure of the proposed algorithm is described by the following section. By default, the $G^{th}$ level of granularity is considered.

### B. Algorithmic Procedure

For initialization, the proposed algorithm treats every individual sample as a micro-cluster with itself as the medoid. As a result, there are $K$ micro-clusters in total. Then, the corresponding membership matrix, denoted by $U_{micro}$ is obtained by using each micro-cluster medoid, $x_k$ to assign memberships to all the $K$ data samples in the data space (including itself):

$$U_{micro} = [\bar{\mu}_k^1]_{k=1:K}$$

where $\bar{\mu}_{k,j} = \frac{\mu(x_k, x_j, \sigma_G)}{\sum_{i=1}^K \mu(x_i, x_j, \sigma_G)}$; $\mu(x_k, x_j, \sigma_G)$ is calculated by Eqn. (4).

Based on $U_{micro}$, the cumulative membership of each micro-cluster medoid, $x_k$ is calculated by Eqn. (7) ($k = 1, 2, \ldots, K$):

$$\lambda(x_k) = \sum_{j=1}^{K} \bar{\mu}_{k,j} = \sum_{j=1}^{K} \frac{\mu(x_k, x_j, \sigma_G)}{\sum_{i=1}^{K} \mu(x_i, x_j, \sigma_G)}$$

The cumulative membership, $\lambda(x)$ is the sum of normalized membership degrees that each individual cluster medoid assigns to all data samples. It has the power to disclose the natural multimodal structure of data. The cumulative membership values of all data samples from S1 dataset (available at http://cs.joensuu.fi/sipu/datasets/) are depicted in Fig. 1 for illustration, where the level of granularity, $G$ is set to be 1, 3, 4 and 5 for Figs. 1(a)-1(d), respectively.

After this, Condition 1 (Eqn. (8)) is used to identify a small set of data samples from $\{x\}$, denoted by $\{p\}^0_C$ as the local maxima of $\lambda(x)$, representing local models of data distribution.

**Condition 1:** If $$(\lambda(x_k) > \max_{\lambda(x_j) : \|x_k - x_j\| \leq \sigma_G, k \neq j} (\lambda(x_j)))$$

Then $$(x_k \in \{p\}^0_C)$$

where $C$ is the cardinality of $\{p\}^0_C$. The local maxima of $\lambda(x)$ identified by Condition 1 are depicted in Fig. 1 as the red dots.

Next, $\{p\}^0_C$ are used as the cluster medoids for data partitioning. The algorithm then iteratively optimizes the positions of cluster medoids by minimizing the objective function, namely, Eqn. (1). It is worth noting that using the local maxima, $\{p\}^0_C$ identified by Condition 1 as the initial cluster medoids brings two attractive benefits: (1) the number of clusters is self-determined by the algorithm based on the spatial distribution of data instead of asking users to pre-set based on prior knowledge of the problems; 2) the algorithm is more robust than the traditional FCM algorithm because it is free from random initialization. The initial cluster medoids identified from S1 dataset under the levels of granularity set as 1, 3, 4 and 5 are presented in Figs. 2(a)-2(d), respectively, for illustration, where dots in different colours stand for samples of different discovered clusters with transparency proportional to the respective membership degrees. Note that unless specifically declared otherwise, clustering results presented in this paper are obtained after defuzzification. It can be seen from Fig. 2 that a greater value of $G$ helps the proposed algorithm to identify more cluster medoids in the data space, leading to finer partitioning result.

To minimize the objective function, $\{p\}^0_C$ is firstly reformulated as the cluster medoid matrix, denoted by $P^0$ and the corresponding membership matrix, $U^0$ is obtained by Eqn. (3). Based on $P^0$ and $U^0$, $J(U^0, P^0)$ can be calculated using Eqn. (1). The proposed algorithm iteratively optimizes the positions of cluster medoids by repeating the following two steps ($t \leftarrow t + 1$), which is standard for FCMdd [19], [23].

**Step 1.** Update $P^{t-1}$ to $P^t$ with fixed $U^{t-1}$ by Eqn. (9):

$$p_t^i = \arg \min_{z \in \{x\}} \left( \sum_{k=1}^K \bar{\mu}_{t-1,k} \|x_k - z\|^2 \right)$$
Then, update $U^{t-1}$ to $U^t$ with fixed $P^t$ by Eqn. (10):

$$\tilde{\mu}_{i,k}^t = \frac{\mu(p_i^t, x_k, \sigma_G)}{\sum_{j=1}^C \mu(p_j^t, x_k, \sigma_G)}$$

(10)

where $i = 1, 2, \ldots, C$; $k = 1, 2, \ldots, K$.

Step 2. Calculate $J(U^t, P^t)$ using Eqn. (1). Then, the algorithm goes back to Step 1 with $t \leftarrow t + 1$.

The iteration process continues until $J(U^t, P^t)$ reaches a locally minimum value. Once the iteration process is terminated, the final cluster medoid matrix, $P^t$ and membership matrix, $U^t$ are obtained as the algorithm’s output, re-denoted by $P$ and $U$. Following the example given by Figs. 1 and 2, the final cluster medoids obtained after the iterative optimization process from $S_1$ dataset under different levels of granularity are presented in Fig. 3. By comparing between Figs. 2 and 3, one may conclude that the cluster medoids identified by Condition 1 provide a good initialization for the later iterative optimization process.

It is worth noting that the robustness of the proposed AFC algorithm can be further improved by replacing the Euclidean distance with a robust dissimilarity measure and using a more robust objective function for optimization. One may find good examples of robust dissimilarity measures and objective functions for fuzzy clustering from [33]. Nevertheless, this is beyond the scope of this paper.

C. Summarization

The algorithmic procedure of the proposed AFC algorithm for static data clustering is summarized by Algorithm 1.

Algorithm 1 AFC for static data clustering.

| inputs: | i) static dataset, $\{x\}$; ii) level of granularity, $G$; |
| outputs: | cluster medoid matrix, $P$; |
| obtain the self-adjusting kernel width, $\sigma_G$ by (5); |
| obtain $U_{micro}$ by (6); |
| calculate $\lambda(x)$ by (7); |
| obtain $P^0$ by (8) and obtain $U^0$ by (3); |
| calculate $J(U^0, P^0)$ by (1); |
| $t \leftarrow 1$; |
| while true do |
| update $P^t$ and $U^t$ by (9) and (10); |
| calculate $J(U^t, P^t)$ by (1); |
| if ($J(U^t, P^t)$ converges) then |
| break; |
| else |
| $t \leftarrow t + 1$; |
| end if |
| end while |
| $P \leftarrow P^t$; |
| return $P$. |

III. Extension to Streaming Data Clustering

As many real-world applications concern streaming data processing, this section introduces an extension to the AFC algorithm for data stream clustering. This extension enables the proposed algorithm to continue clustering newly arrived data samples chunk-by-chunk on top of the original partition
initialized by a static dataset. To guarantee its memory-efficiency, AFC discards all the processed historical data chunks and keeps only the key information in the system.

The main algorithmic procedure for chunk-by-chunk streaming data clustering is described as follows. By default, it is assumed that there have been $C_{L-1}^L$ cluster medoids in total, denoted by $\{p\}_{L-1}^L = \{p_{L-1,1}^L, p_{L-1,2}^L, \ldots, p_{L-1,C_{L-1}^L}^L\}$, identified from the $L-1$ historical data chunks.

### A. Algorithmic Procedure

Given the $L^{th}$ data chunk, denoted by $\{x\}_L = \{x_{L,1}, x_{L,2}, \ldots, x_{L,C_L}\}$ ($K_L$ is the size of $\{x\}_L$), the algorithm firstly updates the self-adjusting kernel width, $\sigma_G$ using Eqn. (11):

$$\sigma_G^2 = \frac{1}{\sum_{i=1}^{L} K_i} \sum_{i=1}^{L} K_i \sigma_G^2$$

(11)

where $\sigma_G$ is the kernel width calculated from $\{x\}_i$ ($i = 1, 2, \ldots, L$) using Eqn. (5); $K_i$ is the corresponding chunk size. The algorithm then extracts a set of cluster medoids from $\{x\}_L$, denoted by $\{p\}_L = \{p_{L,1}, p_{L,2}, \ldots, p_{L,C_L}\}$ using \textbf{Algorithm 1}. Note that different data chunks do not necessarily need to be of the same size.

After this, the main task is to merge $\{p\}_L$ and $\{p\}_{L-1}^L$ together to produce $\{p\}_L^L$. However, combining them together directly is not the best solution because $\{p\}_L$ may contain a portion of cluster medoids that represent similar patterns to some of cluster medoids identified from the previous chunks. To achieve a more compact and concise data partition, the following two-step approach for cluster medoid selection is proposed:

1. **Step 1.** Extract the most distinctive cluster medoids from both $\{p\}_L$ and $\{p\}_{L-1}^L$ and combine them into $\{p\}_L^L$.

2. **Step 2.** Select out the more representative cluster medoids from the rest of $\{p\}_L$ and $\{p\}_{L-1}^L$ to join $\{p\}_L^L$.

Firstly, Eqn. (12) is employed to split $\{p\}_L^L$ into two subsets, namely, $\{p\}_{L,1}^L$ and $\{p\}_{L,2}^L$:

$$\{p\}_{L,1}^L \leftarrow \{p\}_L \cup \{p_{L,j}\}, \text{ if } \min_{p^* \in \{p\}_{L-1}^L} (||p^* - p_{L,j}||)^2 \geq \sigma_G^2$$

$$\{p\}_{L,2}^L \leftarrow \{p\}_L \cup \{p_{L,j}\}, \text{ else}$$

(12)

where $j = 1, 2, \ldots, C_L$. Similarly, Eqn. (13) is used to split $\{p\}_{L-1}^L$ into $\{p\}_{L-1,1}^L$ and $\{p\}_{L-1,2}^L$, where $i = 1, 2, \ldots, C_{L-1}$:

$$\{p\}_{L-1,1}^L \leftarrow \{p\}_{L-1} \cup \{p_{L-1,j}\}, \text{ if } \min_{p^* \in \{p\}_L} (||p^* - p_{L-1,j}||)^2 \geq \sigma_G^2$$

$$\{p\}_{L-1,2}^L \leftarrow \{p\}_{L-1} \cup \{p_{L-1,j}\}, \text{ else}$$

(13)

Note that $\{p\}_{L-1,1}^L$ and $\{p\}_{L,1}^L$ are the sets of cluster medoids that are spatially distant from each other with no overlap in their areas of influence. Therefore, cluster medoids of the two sets are more discriminative and will be kept as a part of $\{p\}_{L,1}^L$. In contrast, cluster medoids of $\{p\}_{L-1,2}^L$ and $\{p\}_{L,2}^L$ are spatially closer to each other and could represent the same local models of data distribution. Thus, they need to be closely examined to avoid redundancy.

Then, Condition 2 (Eqn. (14)) is employed to identify the more representative cluster medoids from $\{p\}_{L-1,1}^L$ and $\{p\}_{L,1}^L$ to join $\{p\}_L^L$.

Condition 2: If $$(\lambda_L(p_i) > \max_{p_{L,j} \in \{p\}_{L-1}^L} \lambda_L(p_{L,j}))$$

Then $$(\{p\}_L^L \leftarrow \{p\}_L \cup \{p\}_L^L)$$

(14)

where $\{p\}_L^L = \{p_{L,1}^L, p_{L,2}^L, \ldots, p_{L,C_L}^L\}$; $\lambda_L(p_i)$ is the cumulative membership of $p_i$ calculated with $\{x\}_L$ by Eqn. (15):

$$\lambda_L(p_i) = \sum_{j=1}^{K_L} c_{i,j} = \sum_{j=1}^{K_L} \mu(p_i, x_{L,j}, \sigma_G)$$

(15)

where $\mu(p_i, x_{L,j}, \sigma_G) = \sum_{k=1}^{M_L} \mu(p_i, x_{L,j}, \sigma_G)$; $c_{i,j} = \frac{\mu(p_i, x_{L,j}, \sigma_G)}{\sum_{k=1}^{M_L} \mu(p_i, x_{L,j}, \sigma_G)}$; $M_L$ is the cardinality of $\{p\}_{L-1}^L \cup \{p\}_L^L$. After this, the current processing cycle is finished, AFC is ready for the next data chunk ($L \leftarrow L + 1$).

The main aim of Condition 2 is to identify these cluster medoids with higher cumulative membership values than their neighbours locally. According to Eqn. (15), only the cluster medoids that describe the local models of the current data distribution the best can have the highest cumulative membership values. Thus, these cluster medoids satisfying Condition 2 can better represent the patterns of the current data chunk than others in $\{p\}_{L-1}^L$ and $\{p\}_L^L$ and will be kept in $\{p\}_L^L$. Other cluster medoids that represent similar patterns to these selected ones but with lower descriptive ability are removed from the data space to avoid overlapping.

With the proposed extension, AFC can effectively handle both the concept shifts and drifts in the data streams [37]. During each learning cycle, out-of-date cluster medoids will be replaced with the more representative ones to self-adapt to concept drifts, namely, gradual changes of data patterns. At the same time, new cluster medoids that represent emerging data patterns of the data streams will be added to the clustering output when concept shifts, namely, abrupt changes of data patterns are detected. Therefore, AFC is suitable for clustering both stationary and nonstationary data streams.

However, to avoid the loss of valuable knowledge mined from data streams before, AFC will maintain all the cluster medoids identified from historical data chunks as long as they are distinctive and can well represent the local patterns of historical data. Nevertheless, one may choose to remove these cluster medoids that could not represent the latest data patterns of the current chunk from the clustering outputs to further enhance the capability of AFC to handle nonstationary streaming data.

An illustrative example is given in Fig. 4 using S1 dataset, where the dataset is split into two chunks evenly, and the level of granularity is selected as $G = 4$. Fig. 4(a) shows the identified cluster medoids from the first chunk; Fig. 4(b) gives the identified cluster medoids from the second chunk; cluster medoids from the data chunks are plotted together in Fig. 4(c); and the final clustering result after merging the two sets of cluster medoids are given in Fig. 4(d), where all historical data samples are included in the final clustering outcome for better visualization.
Algorithm 2 AFC for streaming data clustering.

**inputs:**
- data chunks, \(\{x\}_t\), \(\{x\}_2\), ..., \(\{x\}_L\);
- level of granularity, \(G\);

**outputs:** cluster medoid matrix, \(P\);

While \((\{x\}_L\) is available) do:
- obtain \(\{p\}_L\) from \(\{x\}_L\) using Algorithm 1;
  - if \((L = 1)\) then
    - \(\{p\}_L\) ← \(\{p\}_L\);
  - else
    - update \(\sigma_G\) by (11);
    - split \(\{p\}_L\) to \(\{p\}_L^1\) and \(\{p\}_L^2\) by (12);
    - split \(\{p\}_L^i\) to \(\{p\}_L^{i-1}\) and \(\{p\}_L^{i+1}\) by (13);
    - \(\{p\}_L\) ← \(\{p\}_L^1\cup\{p\}_L^2\);
    - expand \(\{p\}_L^1\) with \(\{p\}_L^{i-1}\) and \(\{p\}_L^{i+1}\) by (14);
  - end if
- end while
- \(P_L\) ← \(\{p\}_L\);
- return \(P_L\).

B. Summarization

The algorithmic procedure of the proposed AFC algorithm for chunk-by-chunk for streaming data clustering is summarized by **Algorithm 2.**

IV. Computational Complexity Analysis

The computational complexity of the proposed AFC algorithm is analysed in this section.

For static data clustering, AFC firstly treats every sample as a micro-cluster and calculates the membership matrix, \(U_{micro}\). The complexity for this is \(O(K^2)\). Then, AFC identifies \(C\) cluster medoids, \(\{p\}_L^0\), using Condition 1. The computational complexity of calculating cumulative membership and identifying cluster medoids is \(O(K)\). After this, AFC iteratively optimizes these cluster medoids to minimize the loss function, \(J(U, P)\) to produce the ultimate data partition. Assuming that \(J(U, P)\) converges to the local minimum value after \(H\) iterations, the computational complexity for this optimization process is \(O(HCK)\). Therefore, the overall complexity of AFC is \(O(K^2 + HCK)\).

For streaming data clustering, since the computational complexity is dynamically changing, the analysis is assumed to be conducted at the time instance when AFC receives the \(L^{th}\) data chunk, \(\{x\}_L\). The computational complexity for AFC to partition \(\{x\}_L\) and extract \(C_L\) cluster medoids, \(\{p\}_L\) is \(O(K_L^2 + H_L C_L K_L)\), where \(H_L\) stands for the number of iterations before \(J(U, P)\) converges. To merge \(\{p\}_L\) with the identified cluster medoids from historical chunks, \(\{p\}_L^{i-1}\), AFC firstly split \(\{p\}_L\) and \(\{p\}_L^{i-1}\) into \(\{p\}_L^1\), \(\{p\}_L^2\), \(\{p\}_L^{i-1+1}\), and \(\{p\}_L^{i-1}\), respectively, based on their mutual distances, and the computational complexity for this is \(O(C_L C_L^{i-1})\). The complexity for selecting out the more representative cluster medoids from \(\{p\}_L^1 \cup \{p\}_L^{i-1}\) using Condition 2 is \(O(K_L M_L)\). Therefore, the complexity for AFC to process the \(L^{th}\) data chunk is \(O(K_L^2 + H_L C_L K_L + C_L C_L^{i-1} + K_L M_L)\).

Based on the above analysis, the overall computational complexity for AFC to cluster a data stream composed of \(L\) chunks is \(O(\sum_{i=1}^L (K_i^2 + H_i C_i K_i + C_i C_i^{i-1} + K_i M_i))\), and there are \(C_0^0 = 0\) and \(M_1 = 0\).

V. Numerical Examples

In this section, numerical examples are presented for demonstrating the efficacy of the proposed AFC algorithm. Numerical experiments are conducted with Matlab2020b on a laptop with dual core i7 processor with clock frequency 2.6GHz×2 and 16GB RAM.

A. Experimental Setting

For experimental investigation, 16 popular benchmark datasets are used, which include six synthetic problems, eight real-world problems and two image recognition problems. Details of these datasets are listed in Table I, where \(T\), \(K\) and \(N\) represent “number of classes”, “number of samples” and “number of attributes”, respectively. Web links to the 14 datasets are given by Table S1 in the Supplementary Material. For benchmark comparison, a total of 16 state-of-the-art clustering algorithms are involved.

1) Fuzzy c-means (FCM) clustering algorithm [38];
2) Kernel FCM clustering algorithm with membership affinity lasso regularization (MAL) [29];
3) K-means (KM) clustering algorithm [6];
4) DBSCAN (DBS) clustering algorithm [8];
5) Mean shift (MS) clustering algorithm [12];
6) Subtractive (SUB) clustering algorithm [39];
TABLE I: KEY DETAILS OF BENCHMARK DATASETS

<table>
<thead>
<tr>
<th>Dataset</th>
<th>Abbreviation</th>
<th>T</th>
<th>K</th>
<th>N</th>
</tr>
</thead>
<tbody>
<tr>
<td>R15</td>
<td>R15</td>
<td>15</td>
<td>500</td>
<td>2</td>
</tr>
<tr>
<td>Aggregation</td>
<td>AG</td>
<td>7</td>
<td>8</td>
<td>2</td>
</tr>
<tr>
<td>S1</td>
<td>S1</td>
<td>15</td>
<td>2</td>
<td>2</td>
</tr>
<tr>
<td>S2</td>
<td>S2</td>
<td>15</td>
<td>2</td>
<td>2</td>
</tr>
<tr>
<td>S3</td>
<td>S3</td>
<td>15</td>
<td>2</td>
<td>2</td>
</tr>
<tr>
<td>S4</td>
<td>S4</td>
<td>15</td>
<td>2</td>
<td>2</td>
</tr>
<tr>
<td>Abalone</td>
<td>AB</td>
<td>3</td>
<td>417</td>
<td>8</td>
</tr>
<tr>
<td>Spambase</td>
<td>SB</td>
<td>2</td>
<td>460</td>
<td>57</td>
</tr>
<tr>
<td>Cardiotocgraphy</td>
<td>CG</td>
<td>10</td>
<td>212</td>
<td>21</td>
</tr>
<tr>
<td>Steel plate faults</td>
<td>SPF</td>
<td>7</td>
<td>194</td>
<td>2</td>
</tr>
<tr>
<td>Multiple features</td>
<td>MF</td>
<td>10</td>
<td>200</td>
<td>649</td>
</tr>
<tr>
<td>Pen-based handwritten digit recognition</td>
<td>PD</td>
<td>10</td>
<td>10992</td>
<td>16</td>
</tr>
<tr>
<td>Wine quality</td>
<td>WQ</td>
<td>7</td>
<td>649</td>
<td>11</td>
</tr>
<tr>
<td>Occupancy detection</td>
<td>OD</td>
<td>2</td>
<td>20560</td>
<td>5</td>
</tr>
<tr>
<td>MNIST</td>
<td>MNIST</td>
<td>10</td>
<td>70000</td>
<td>784</td>
</tr>
<tr>
<td>Fashion MNIST</td>
<td>FMNIST</td>
<td>10</td>
<td>70000</td>
<td>784</td>
</tr>
</tbody>
</table>

7) Nonparametric mode identification (NMI) algorithm [40];
8) Affinity propagation (AP) algorithm [10];
9) Gaussian density distance (GDD) clustering algorithm [14];
10) Communication with local agents (CLA) clustering algorithm [15];
11) Local gravitation clustering (LGC) algorithm [15];
12) Autonomous data partitioning (ADP) algorithm [16];
13) Evolving clustering (EC) algorithm [45];
14) Online k-means (OKM) algorithm [46];
15) Evolving local means (ELM) algorithm [47];
16) Online clustering and anomaly detection (OCA) algorithm [15].

The respective parameter settings of these comparative clustering algorithms for numerical experiments are listed in Table S2 in the Supplementary Material. Among the 16 algorithms, EC, OKM, ELM and OCA are designed specifically for streaming data clustering. ADP has two different versions, namely, the offline version for static data clustering, and the evolving version for online scenarios. The other 11 algorithms are designed for static data clustering in offline application scenarios. The externally controlled parameter settings for these clustering algorithms are determined based on the recommendations given by the literature. Note that the parameter settings of GDD and ADP are hard coded; thus, there is no need for users to predefine externally controlled parameters.

In order to objectively measure the performance of the clustering algorithms, the following six criteria are considered in this paper.
1) Number of clusters (C);
2) Adjusted Rand index (ARI) [41];
ARI is the corrected-for-chance version of the Rand index for evaluating the accuracy of clustering results. The value range of ARI is $[-1, 1]$ and, generally, the greater ARI is, the better clustering result is.
3) Calinski Harabasz index (CHI) [42];
CHI is used to evaluate the optimal number of clusters. Better clustering results usually have greater CHI values.
4) Davies-Bouldin index (DBI) [43];
DBI is based on a ratio of within-cluster and between-cluster distances. Better clustering results usually have smaller DBI values.
5) Silhouette coefficient (SC) [44];
SC is an indication of how well each sample lies within its cluster. The value range of SC is $[-1, 1]$. SC should also be as high as possible.
6) Execution time in seconds ($t_{exe}$).
$t_{exe}$ is for measuring the computational efficiency and should be as lower as possible.

In this paper, numerical results of fuzzy clustering algorithms including the proposed one are obtained after defuzzification. All the reported numerical results are averaged over five Monte Carlo experiments to allow a certain degree of randomness.

B. Performance Demonstration

In this subsection, the clustering performance of the proposed AFC algorithm is evaluated. By default, the experiments are conducted in offline scenarios unless expressly declared otherwise.

Firstly, the influence of the level of granularity, $G$ on AFC’s clustering performance is investigated. In this example, the following six synthetic benchmark datasets, R15, AG, S1, S2, S3 and S4, are used. The level of granularity, $G$ is set to be 2, 3, 4, 5, 6 and 7, respectively, during this experiment. Clustering results obtained by AFC are presented in Table S3 in the Supplementary Material in terms of the aforementioned six criteria. It can be observed from Table S3 that given a smaller value of $G$, AFC focuses more on the main patterns of data, but it may fail to capture the information of local data patterns if $G$ is too small. On the other hand, AFC has greater ability of disclosing local patterns if a greater value of $G$ is chosen, but the clustering result may contain too many unnecessary details and become uninterpretable for users if the value of $G$ is too large. In addition, its computational efficiency may also decrease due to the higher complexity of the iterative process for optimizing cluster medoids. Based on the clustering results evaluated by the four quality measures, the recommended values of $G$ are 4 and 5. These two values will be used for the remaining experiments presented in this section unless specifically declared otherwise. Nevertheless, it has to be admitted that the most appropriate level of granularity is determined by the nature of data and would vary from problem to problem.

Secondly, the proposed algorithm is compared with 12 state-of-the-art offline clustering algorithms on the six synthetic
### Table II: Static Data Clustering Performance Comparison on Six Benchmark Synthetic Datasets

<table>
<thead>
<tr>
<th>Algorithm</th>
<th>Data set</th>
<th>Measure</th>
<th>$C$</th>
<th>$ARI$</th>
<th>$CHI$</th>
<th>$DBI$</th>
<th>$SC$</th>
<th>$t_{ave}$</th>
</tr>
</thead>
<tbody>
<tr>
<td>AFC ($G = 4$)</td>
<td>15</td>
<td>0.986</td>
<td>4846.116</td>
<td>0.315</td>
<td>0.900</td>
<td>0.013</td>
<td></td>
<td></td>
</tr>
<tr>
<td>AFC ($G = 5$)</td>
<td>15</td>
<td>0.974</td>
<td>4682.205</td>
<td>0.414</td>
<td>0.869</td>
<td>0.014</td>
<td></td>
<td></td>
</tr>
<tr>
<td>FCM</td>
<td>15</td>
<td>0.976</td>
<td>4114.342</td>
<td>0.361</td>
<td>0.873</td>
<td>0.034</td>
<td></td>
<td></td>
</tr>
<tr>
<td>MAL</td>
<td>12</td>
<td>0.727</td>
<td>1125.451</td>
<td>0.830</td>
<td>8.388</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>KM</td>
<td>15</td>
<td>0.908</td>
<td>3481.683</td>
<td>0.470</td>
<td>0.826</td>
<td>0.031</td>
<td></td>
<td></td>
</tr>
<tr>
<td>DBS</td>
<td>15</td>
<td>0.989</td>
<td>4689.330</td>
<td>0.314</td>
<td>0.901</td>
<td>0.029</td>
<td></td>
<td></td>
</tr>
<tr>
<td>MS</td>
<td>R15</td>
<td>1</td>
<td>0.001</td>
<td>NaN</td>
<td>NaN</td>
<td>NaN</td>
<td>0.010</td>
<td></td>
</tr>
<tr>
<td>SUB</td>
<td>8</td>
<td>0.264</td>
<td>760.074</td>
<td>0.349</td>
<td>0.781</td>
<td>0.188</td>
<td></td>
<td></td>
</tr>
<tr>
<td>NMI</td>
<td>1</td>
<td>0.000</td>
<td>NaN</td>
<td>NaN</td>
<td>NaN</td>
<td>NaN</td>
<td>0.388</td>
<td></td>
</tr>
<tr>
<td>AP</td>
<td>15</td>
<td>0.843</td>
<td>4835.193</td>
<td>0.316</td>
<td>0.899</td>
<td>0.664</td>
<td></td>
<td></td>
</tr>
<tr>
<td>GDD</td>
<td>10</td>
<td>0.480</td>
<td>789.314</td>
<td>0.614</td>
<td>0.533</td>
<td>0.105</td>
<td></td>
<td></td>
</tr>
<tr>
<td>CLA</td>
<td>15</td>
<td>0.899</td>
<td>4842.778</td>
<td>0.315</td>
<td>0.899</td>
<td>0.060</td>
<td></td>
<td></td>
</tr>
<tr>
<td>LGC</td>
<td>15</td>
<td>0.989</td>
<td>4862.929</td>
<td>0.315</td>
<td>0.900</td>
<td>0.039</td>
<td></td>
<td></td>
</tr>
<tr>
<td>ADP</td>
<td>15</td>
<td>0.993</td>
<td>4871.983</td>
<td>0.315</td>
<td>0.901</td>
<td>0.033</td>
<td></td>
<td></td>
</tr>
<tr>
<td>AFC ($G = 4$)</td>
<td>15</td>
<td>0.986</td>
<td>22675.165</td>
<td>0.366</td>
<td>0.880</td>
<td>0.058</td>
<td></td>
<td></td>
</tr>
<tr>
<td>AFC ($G = 5$)</td>
<td>20</td>
<td>0.901</td>
<td>18323.773</td>
<td>0.829</td>
<td>0.731</td>
<td>1.023</td>
<td></td>
<td></td>
</tr>
<tr>
<td>FCM</td>
<td>15</td>
<td>0.970</td>
<td>21122.554</td>
<td>0.396</td>
<td>0.868</td>
<td>0.163</td>
<td></td>
<td></td>
</tr>
<tr>
<td>MAL</td>
<td>15</td>
<td>0.884</td>
<td>11891.423</td>
<td>0.740</td>
<td>0.719</td>
<td>298.009</td>
<td></td>
<td></td>
</tr>
<tr>
<td>KM</td>
<td>15</td>
<td>0.903</td>
<td>14190.740</td>
<td>0.510</td>
<td>0.817</td>
<td>0.035</td>
<td></td>
<td></td>
</tr>
<tr>
<td>DBS</td>
<td>32</td>
<td>0.932</td>
<td>12395.415</td>
<td>0.783</td>
<td>0.651</td>
<td>0.818</td>
<td></td>
<td></td>
</tr>
<tr>
<td>MS</td>
<td>S1</td>
<td>4</td>
<td>0.210</td>
<td>2414.677</td>
<td>0.858</td>
<td>0.329</td>
<td>0.511</td>
<td></td>
</tr>
<tr>
<td>SUB</td>
<td>10</td>
<td>0.708</td>
<td>8360.556</td>
<td>0.573</td>
<td>0.722</td>
<td>0.551</td>
<td></td>
<td></td>
</tr>
<tr>
<td>NMI</td>
<td>6</td>
<td>0.479</td>
<td>5922.534</td>
<td>0.731</td>
<td>0.635</td>
<td>7.285</td>
<td></td>
<td></td>
</tr>
<tr>
<td>AP</td>
<td>(1911)</td>
<td>0.336</td>
<td>(207.948)</td>
<td>(0.742)</td>
<td>(0.307)</td>
<td>(97.992)</td>
<td></td>
<td></td>
</tr>
<tr>
<td>GDD</td>
<td>97</td>
<td>0.706</td>
<td>11390.871</td>
<td>0.566</td>
<td>2.557</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>CLA</td>
<td>15</td>
<td>0.987</td>
<td>22591.143</td>
<td>0.367</td>
<td>0.879</td>
<td>1.517</td>
<td></td>
<td></td>
</tr>
<tr>
<td>LGC</td>
<td>16</td>
<td>0.977</td>
<td>21268.683</td>
<td>0.456</td>
<td>0.846</td>
<td>0.747</td>
<td></td>
<td></td>
</tr>
<tr>
<td>ADP</td>
<td>15</td>
<td>0.985</td>
<td>21675.254</td>
<td>0.368</td>
<td>0.880</td>
<td>0.540</td>
<td></td>
<td></td>
</tr>
<tr>
<td>AFC ($G = 4$)</td>
<td>12</td>
<td>0.699</td>
<td>6735.569</td>
<td>0.667</td>
<td>0.650</td>
<td>0.912</td>
<td></td>
<td></td>
</tr>
<tr>
<td>AFC ($G = 5$)</td>
<td>23</td>
<td>0.641</td>
<td>6344.410</td>
<td>0.862</td>
<td>0.600</td>
<td>0.951</td>
<td></td>
<td></td>
</tr>
<tr>
<td>FCM</td>
<td>15</td>
<td>0.718</td>
<td>7004.313</td>
<td>0.686</td>
<td>0.659</td>
<td>0.284</td>
<td></td>
<td></td>
</tr>
<tr>
<td>MAL</td>
<td>15</td>
<td>0.444</td>
<td>2344.551</td>
<td>1.607</td>
<td>0.606</td>
<td>261.302</td>
<td></td>
<td></td>
</tr>
<tr>
<td>KM</td>
<td>15</td>
<td>0.645</td>
<td>6635.234</td>
<td>0.755</td>
<td>0.619</td>
<td>0.042</td>
<td></td>
<td></td>
</tr>
<tr>
<td>DBS</td>
<td>14</td>
<td>0.001</td>
<td>48.748</td>
<td>1.107</td>
<td>-0.638</td>
<td>0.827</td>
<td></td>
<td></td>
</tr>
<tr>
<td>MS</td>
<td>S3</td>
<td>1</td>
<td>0.000</td>
<td>NaN</td>
<td>NaN</td>
<td>NaN</td>
<td>0.029</td>
<td></td>
</tr>
<tr>
<td>SUB</td>
<td>7</td>
<td>0.422</td>
<td>5922.534</td>
<td>0.731</td>
<td>0.635</td>
<td>7.285</td>
<td></td>
<td></td>
</tr>
<tr>
<td>NMI</td>
<td>15</td>
<td>0.727</td>
<td>7673.618</td>
<td>0.654</td>
<td>0.654</td>
<td>12.578</td>
<td></td>
<td></td>
</tr>
<tr>
<td>AP</td>
<td>(2165)</td>
<td>0.226</td>
<td>(74.298)</td>
<td>(0.760)</td>
<td>(0.106)</td>
<td>(95.559)</td>
<td></td>
<td></td>
</tr>
<tr>
<td>GDD</td>
<td>203</td>
<td>0.019</td>
<td>3.730</td>
<td>1.163</td>
<td>-0.793</td>
<td>2.546</td>
<td></td>
<td></td>
</tr>
<tr>
<td>CLA</td>
<td>14</td>
<td>0.700</td>
<td>6815.032</td>
<td>0.652</td>
<td>0.621</td>
<td>1.402</td>
<td></td>
<td></td>
</tr>
<tr>
<td>LGC</td>
<td>23</td>
<td>0.620</td>
<td>5003.105</td>
<td>0.893</td>
<td>0.523</td>
<td>0.771</td>
<td></td>
<td></td>
</tr>
<tr>
<td>ADP</td>
<td>24</td>
<td>0.642</td>
<td>6429.828</td>
<td>0.85</td>
<td>0.599</td>
<td>0.547</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

The table presents the performance comparison of different algorithms on six benchmark synthetic datasets. The performance measures include $C$, $ARI$, $CHI$, $DBI$, and $SC$. The results are normalized to show the performance improvement or degradation compared to the baseline. The $t_{ave}$ column indicates the average performance improvement across the datasets.

Benchmark datasets used before in offline scenarios. Performance comparison is conducted under the six measures and the results are reported in Table II (NaN stands for “not a number”). To better interpret the results, performances of the clustering algorithms on each individual dataset are ranked in terms of the four clustering quality criteria ($ARI$, $CHI$, $DBI$, and $SI$) individually. Ranks of all the algorithms per dataset per criterion are given by Table S5 in the Supplementary Material. Examples of clustering results obtained by AFC are given by Fig. 5 for better illustration, where the level of granularity, $G$, is set as 4.

Furthermore, the following eight real-world datasets, namely, AB, SB, CG, SPF, MF, PD, WQ and OD are used for performance evaluation in offline scenarios. The performance of the proposed AFC algorithm is also compared with the same 12 comparative algorithms used before under the same six measures. The clustering results obtained by AFC and the 12 competitors are presented in Table III ($Inf$ stands for “infinity value”). Similarly, ranks of these algorithms per dataset per criterion are given by Table S6 in the Supplementary Material. For visual clarity, the overall ranks of the offline algorithms over the 14 benchmark datasets are reported in Table IV.

It can be observed from Table IV that the proposed AFC algorithm with $G = 4$ is able to obtain the best overall clustering results over the 14 benchmark datasets. The values of the three quality indices, namely, $CHI$, $DBI$ and $SI$ calculated on its clustering results are ranked the top over all the clustering algorithms involved in the numerical experiments. On the other hand, AFC with $G = 5$ ranks at the sixth place among the 14 algorithms in terms of the overall performance. The reason for this is that the level of granularity controls the degree of fineness of the clustering outcome. With a higher level of granularity, AFC focuses more on the local data patterns and tends to produce more clusters, but this unfavourably decreases the values of clustering quality indices calculated from the partition results.

Next, the synthetic benchmark dataset $S4$ is used for illustrating the online streaming data clustering process of the AFC algorithm. In this example, $S4$ dataset is randomly divided into four chunks evenly, and AFC with $G = 4$ groups the data chunk-by-chunk. Evolution of the clustering outcome over time is visualized in Fig. S2 in the Supplementary Material. Here, the obtained cluster medoids and all processed data chunks at the end of each learning cycle are used for
TABLE III: STATIC DATA CLUSTERING PERFORMANCE COMPARISON ON EIGHT REAL-WORLD DATASETS

<table>
<thead>
<tr>
<th>Algorithm</th>
<th>Data set</th>
<th>Measure</th>
<th>C</th>
<th>ARI</th>
<th>CHI</th>
<th>DBI</th>
<th>SC</th>
<th>tDW</th>
<th>Data set</th>
<th>Measure</th>
<th>C</th>
<th>ARI</th>
<th>CHI</th>
<th>DBI</th>
<th>SC</th>
<th>tDW</th>
</tr>
</thead>
<tbody>
<tr>
<td>AFC (G = 4)</td>
<td>15</td>
<td>0.439</td>
<td>2438.019</td>
<td>1.290</td>
<td>0.250</td>
<td>0.206</td>
<td>32</td>
<td>0.064</td>
<td>5355.400</td>
<td>0.436</td>
<td>0.679</td>
<td>0.251</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>AFC (G = 5)</td>
<td>74</td>
<td>0.303</td>
<td>873.157</td>
<td>1.881</td>
<td>0.174</td>
<td>0.618</td>
<td>154</td>
<td>0.168</td>
<td>6668.156</td>
<td>0.766</td>
<td>0.164</td>
<td>5.124</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>FCM</td>
<td>10</td>
<td>0.140</td>
<td>618.067</td>
<td>1.825</td>
<td>0.231</td>
<td>0.166</td>
<td>7</td>
<td>0.040</td>
<td>4458.406</td>
<td>0.683</td>
<td>0.382</td>
<td>0.114</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>MAL</td>
<td>10</td>
<td>0.102</td>
<td>277.281</td>
<td>2.830</td>
<td>0.082</td>
<td>20.564</td>
<td>7</td>
<td>0.188</td>
<td>11.512</td>
<td>54.231</td>
<td>-0.368</td>
<td>25.827</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>KM</td>
<td>10</td>
<td>0.131</td>
<td>723.773</td>
<td>1.322</td>
<td>0.355</td>
<td>0.038</td>
<td>7</td>
<td>0.044</td>
<td>4477.523</td>
<td>0.598</td>
<td>0.652</td>
<td>0.035</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>DBS</td>
<td>14</td>
<td>0.043</td>
<td>64.498</td>
<td>1.352</td>
<td>-0.228</td>
<td>0.218</td>
<td>18</td>
<td>0.078</td>
<td>288.536</td>
<td>1.098</td>
<td>-0.582</td>
<td>0.186</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>MS</td>
<td>(CG) (509)</td>
<td>(0.053)</td>
<td>(47.591)</td>
<td>(1.002)</td>
<td>(-0.092)</td>
<td>(0.200)</td>
<td>SPF</td>
<td>(907)</td>
<td>(0.073)</td>
<td>(11.800)</td>
<td>(0.725)</td>
<td>(0.249)</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>SUB</td>
<td>165</td>
<td>0.035</td>
<td>80.821</td>
<td>2.152</td>
<td>-0.100</td>
<td>0.576</td>
<td>4</td>
<td>0.034</td>
<td>132.914</td>
<td>3.158</td>
<td>-0.283</td>
<td>0.413</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>NMI</td>
<td>322</td>
<td>0.076</td>
<td>67.695</td>
<td>0.658</td>
<td>0.170</td>
<td>19.668</td>
<td>9</td>
<td>-0.002</td>
<td>690.336</td>
<td>0.303</td>
<td>0.708</td>
<td>39.175</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>AP</td>
<td>44</td>
<td>0.064</td>
<td>370.961</td>
<td>1.301</td>
<td>0.290</td>
<td>6.024</td>
<td>(1438)</td>
<td>(0.035)</td>
<td>(6.113)</td>
<td>(0.625)</td>
<td>(0.807)</td>
<td>(14.545)</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>GID</td>
<td>2</td>
<td>0.029</td>
<td>227.020</td>
<td>1.891</td>
<td>0.471</td>
<td>1.731</td>
<td>380</td>
<td>0.073</td>
<td>25.871</td>
<td>0.816</td>
<td>-0.509</td>
<td>1.797</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>CLA</td>
<td>1</td>
<td>0.000</td>
<td>NaN</td>
<td>NaN</td>
<td>NaN</td>
<td>0.506</td>
<td>7</td>
<td>0.075</td>
<td>3350.024</td>
<td>0.810</td>
<td>0.498</td>
<td>0.537</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>LGC</td>
<td>4</td>
<td>0.031</td>
<td>154.850</td>
<td>1.793</td>
<td>0.458</td>
<td>0.214</td>
<td>6</td>
<td>0.053</td>
<td>871.520</td>
<td>0.972</td>
<td>0.185</td>
<td>0.242</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>ADP</td>
<td>5</td>
<td>0.086</td>
<td>759.834</td>
<td>1.080</td>
<td>0.367</td>
<td>0.173</td>
<td>14</td>
<td>0.061</td>
<td>4509.210</td>
<td>0.717</td>
<td>0.676</td>
<td>0.116</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>AFC (G = 4)</td>
<td>15</td>
<td>0.439</td>
<td>2438.019</td>
<td>1.290</td>
<td>0.250</td>
<td>0.206</td>
<td>32</td>
<td>0.064</td>
<td>5355.400</td>
<td>0.436</td>
<td>0.679</td>
<td>0.251</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>AFC (G = 5)</td>
<td>74</td>
<td>0.303</td>
<td>873.157</td>
<td>1.881</td>
<td>0.174</td>
<td>0.618</td>
<td>154</td>
<td>0.168</td>
<td>6668.156</td>
<td>0.766</td>
<td>0.164</td>
<td>5.124</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>FCM</td>
<td>10</td>
<td>0.140</td>
<td>618.067</td>
<td>1.825</td>
<td>0.231</td>
<td>0.166</td>
<td>7</td>
<td>0.040</td>
<td>4458.406</td>
<td>0.683</td>
<td>0.382</td>
<td>0.114</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>MAL</td>
<td>10</td>
<td>0.102</td>
<td>277.281</td>
<td>2.830</td>
<td>0.082</td>
<td>20.564</td>
<td>7</td>
<td>0.188</td>
<td>11.512</td>
<td>54.231</td>
<td>-0.368</td>
<td>25.827</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>KM</td>
<td>10</td>
<td>0.131</td>
<td>723.773</td>
<td>1.322</td>
<td>0.355</td>
<td>0.038</td>
<td>7</td>
<td>0.044</td>
<td>4477.523</td>
<td>0.598</td>
<td>0.652</td>
<td>0.035</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>DBS</td>
<td>14</td>
<td>0.043</td>
<td>64.498</td>
<td>1.352</td>
<td>-0.228</td>
<td>0.218</td>
<td>18</td>
<td>0.078</td>
<td>288.536</td>
<td>1.098</td>
<td>-0.582</td>
<td>0.186</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>MS</td>
<td>(CG) (509)</td>
<td>(0.053)</td>
<td>(47.591)</td>
<td>(1.002)</td>
<td>(-0.092)</td>
<td>(0.200)</td>
<td>SPF</td>
<td>(907)</td>
<td>(0.073)</td>
<td>(11.800)</td>
<td>(0.725)</td>
<td>(0.249)</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>SUB</td>
<td>165</td>
<td>0.035</td>
<td>80.821</td>
<td>2.152</td>
<td>-0.100</td>
<td>0.576</td>
<td>4</td>
<td>0.034</td>
<td>132.914</td>
<td>3.158</td>
<td>-0.283</td>
<td>0.413</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>NMI</td>
<td>322</td>
<td>0.076</td>
<td>67.695</td>
<td>0.658</td>
<td>0.170</td>
<td>19.668</td>
<td>9</td>
<td>-0.002</td>
<td>690.336</td>
<td>0.303</td>
<td>0.708</td>
<td>39.175</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>AP</td>
<td>44</td>
<td>0.064</td>
<td>370.961</td>
<td>1.301</td>
<td>0.290</td>
<td>6.024</td>
<td>(1438)</td>
<td>(0.035)</td>
<td>(6.113)</td>
<td>(0.625)</td>
<td>(0.807)</td>
<td>(14.545)</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>GID</td>
<td>2</td>
<td>0.029</td>
<td>227.020</td>
<td>1.891</td>
<td>0.471</td>
<td>1.731</td>
<td>380</td>
<td>0.073</td>
<td>25.871</td>
<td>0.816</td>
<td>-0.509</td>
<td>1.797</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>CLA</td>
<td>1</td>
<td>0.000</td>
<td>NaN</td>
<td>NaN</td>
<td>NaN</td>
<td>0.506</td>
<td>7</td>
<td>0.075</td>
<td>3350.024</td>
<td>0.810</td>
<td>0.498</td>
<td>0.537</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>LGC</td>
<td>4</td>
<td>0.031</td>
<td>154.850</td>
<td>1.793</td>
<td>0.458</td>
<td>0.214</td>
<td>6</td>
<td>0.053</td>
<td>871.520</td>
<td>0.972</td>
<td>0.185</td>
<td>0.242</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>ADP</td>
<td>5</td>
<td>0.086</td>
<td>759.834</td>
<td>1.080</td>
<td>0.367</td>
<td>0.173</td>
<td>14</td>
<td>0.061</td>
<td>4509.210</td>
<td>0.717</td>
<td>0.676</td>
<td>0.116</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Fig. 5: Final clustering results obtained by AFC with G = 4 (dots– data samples; diamonds – cluster medoids).
TABLE V: STREAM DATA CLUSTERING PERFORMANCE COMPARISON ON EIGHT REAL-WORLD DATASETS

<table>
<thead>
<tr>
<th>Algorithm</th>
<th>Data set</th>
<th>Measure</th>
<th>Overall</th>
<th>ARI</th>
<th>CHI</th>
<th>DBI</th>
<th>SI</th>
</tr>
</thead>
<tbody>
<tr>
<td>AFC (G=4)</td>
<td>C</td>
<td>5.0</td>
<td></td>
<td>0.035</td>
<td>8947.589</td>
<td>0.713</td>
<td>0.158</td>
</tr>
<tr>
<td>AFC (G=5)</td>
<td>C</td>
<td>107</td>
<td></td>
<td>0.033</td>
<td>12865.95</td>
<td>0.509</td>
<td>0.650</td>
</tr>
<tr>
<td>OKM</td>
<td>AB</td>
<td>3</td>
<td></td>
<td>0.123</td>
<td>1836.795</td>
<td>1.921</td>
<td>0.250</td>
</tr>
<tr>
<td>EC</td>
<td>5</td>
<td>0.114</td>
<td></td>
<td>3737.389</td>
<td>0.872</td>
<td>0.302</td>
<td>0.146</td>
</tr>
<tr>
<td>ELM</td>
<td>3</td>
<td>0.009</td>
<td></td>
<td>164.455</td>
<td>1.352</td>
<td>-0.514</td>
<td>0.277</td>
</tr>
<tr>
<td>OCA</td>
<td>3</td>
<td>0.131</td>
<td></td>
<td>716.284</td>
<td>2.539</td>
<td>0.078</td>
<td>0.404</td>
</tr>
<tr>
<td>ADP</td>
<td>27</td>
<td>0.055</td>
<td></td>
<td>6681.103</td>
<td>0.893</td>
<td>0.538</td>
<td>0.319</td>
</tr>
<tr>
<td>AFC (G=4)</td>
<td>C</td>
<td>69</td>
<td></td>
<td>0.114</td>
<td>165.456</td>
<td>0.975</td>
<td>0.266</td>
</tr>
<tr>
<td>AFC (G=5)</td>
<td>C</td>
<td>180</td>
<td></td>
<td>0.067</td>
<td>134.955</td>
<td>0.889</td>
<td>0.280</td>
</tr>
<tr>
<td>OKM</td>
<td>10</td>
<td>0.119</td>
<td></td>
<td>162.611</td>
<td>4.101</td>
<td>-0.085</td>
<td>1.002</td>
</tr>
<tr>
<td>EC</td>
<td>10</td>
<td>0.119</td>
<td></td>
<td>422.256</td>
<td>1.912</td>
<td>0.072</td>
<td>0.101</td>
</tr>
<tr>
<td>ELM</td>
<td>21</td>
<td>0.060</td>
<td></td>
<td>55.161</td>
<td>2.292</td>
<td>-0.238</td>
<td>0.791</td>
</tr>
<tr>
<td>OCA</td>
<td>4</td>
<td>0.030</td>
<td></td>
<td>132.995</td>
<td>11.643</td>
<td>0.043</td>
<td>2.422</td>
</tr>
<tr>
<td>ADP</td>
<td>54</td>
<td>0.083</td>
<td></td>
<td>306.065</td>
<td>1.223</td>
<td>0.308</td>
<td>0.182</td>
</tr>
<tr>
<td>AFC (G=4)</td>
<td>C</td>
<td>20</td>
<td></td>
<td>0.432</td>
<td>1992.035</td>
<td>1.476</td>
<td>0.412</td>
</tr>
<tr>
<td>AFC (G=5)</td>
<td>C</td>
<td>120</td>
<td></td>
<td>0.375</td>
<td>540.966</td>
<td>1.301</td>
<td>0.302</td>
</tr>
<tr>
<td>OKM</td>
<td>10</td>
<td>0.165</td>
<td></td>
<td>56.099</td>
<td>8.808</td>
<td>-0.281</td>
<td>1.611</td>
</tr>
<tr>
<td>EC</td>
<td>9</td>
<td>0.237</td>
<td></td>
<td>1447.081</td>
<td>1.622</td>
<td>0.268</td>
<td>1.573</td>
</tr>
<tr>
<td>ELM</td>
<td>(1988)</td>
<td>(0.000)</td>
<td></td>
<td>(26.181)</td>
<td>(0.249)</td>
<td>(0.990)</td>
<td>(37.776)</td>
</tr>
<tr>
<td>OCA</td>
<td>1</td>
<td>0.000</td>
<td></td>
<td>NaN</td>
<td>NaN</td>
<td>NaN</td>
<td>0.157</td>
</tr>
<tr>
<td>ADP</td>
<td>12</td>
<td>0.357</td>
<td></td>
<td>2339.414</td>
<td>1.304</td>
<td>0.440</td>
<td>0.268</td>
</tr>
<tr>
<td>AFC (G=4)</td>
<td>C</td>
<td>32</td>
<td></td>
<td>-0.001</td>
<td>4208.644</td>
<td>0.693</td>
<td>0.445</td>
</tr>
<tr>
<td>AFC (G=5)</td>
<td>C</td>
<td>62</td>
<td></td>
<td>0.001</td>
<td>3718.894</td>
<td>0.756</td>
<td>0.386</td>
</tr>
<tr>
<td>OKM</td>
<td>7</td>
<td>0.026</td>
<td></td>
<td>628.519</td>
<td>8.450</td>
<td>-0.192</td>
<td>1.207</td>
</tr>
<tr>
<td>EC</td>
<td>WQ</td>
<td>7</td>
<td></td>
<td>0.009</td>
<td>3977.496</td>
<td>1.391</td>
<td>0.134</td>
</tr>
<tr>
<td>ELM</td>
<td>10</td>
<td>0.003</td>
<td></td>
<td>100.751</td>
<td>3.945</td>
<td>-0.546</td>
<td>0.881</td>
</tr>
<tr>
<td>OCA</td>
<td>2</td>
<td>0.004</td>
<td></td>
<td>842.100</td>
<td>26.696</td>
<td>0.150</td>
<td>0.620</td>
</tr>
<tr>
<td>ADP</td>
<td>39</td>
<td>0.003</td>
<td></td>
<td>5446.205</td>
<td>1.043</td>
<td>0.372</td>
<td>0.483</td>
</tr>
</tbody>
</table>

producing the clustering results. The values of $C$, $ARI$, $CHI$, $DBI$ and $SI$ calculated from these clustering results are reported in Table S8 in the Supplementary Material for better demonstration.

Then, the online chunk-by-chunk learning performance of the proposed algorithm is demonstrated based on the eight real-world benchmark datasets as used for the numerical example given by Table III. In this example, each dataset is randomly divided into 2, 3, 4 and 5 chunks evenly. The obtained clustering results measured by the six criteria are tabulated in Table S4 in the Supplementary Material. Note that $C$ is the number of ultimate cluster medoids obtained at the end of the online chunk-by-chunk clustering process; the reported values of $ARI$, $CHI$, $DBI$ and $SI$ are calculated based on the defuzzified clustering result obtained by using these ultimate cluster medoids to partition all historical data chunks together, namely, the entire dataset. One can see from this table that a smaller chunk size allows AFC to perform clustering more efficiently, which is in coincidence with the computational complexity analysis presented in Section IV. This is because that a smaller chunk size can significantly reduce the computational complexity of cumulative membership calculation as well as cluster medoid optimization. Nevertheless, a smaller chunk size also increases the sensitivity of AFC to the changes of data patterns of successive data chunks. As data patterns may change more rapidly within smaller data chunks, AFC has to identify more clusters from each chunk to follow such changes. This would inevitably result in more clusters in the final clustering outcomes.

For better evaluation, the streaming data clustering performance of the proposed AFC algorithm is compared with the...
and group data into more clusters, this would also increase the computational complexity of cluster medoid optimization. If a lower level of granularity is chosen, AFC tends to focus more on main patterns of data. As a result, data will be partitioned coarsely and the clustering outcomes will have less clusters, but the computational efficiency of the algorithm will be much higher. In practice, users can start with the recommended values given by this paper and adjust the parameter setting based on the specific needs of the problems.

In addition, it has to be admitted that similar to other algorithms that employ PAM or other similar strategies, such as KM, FCM and ADP, the proposed AFC algorithm is less effective in capturing non-convex clusters and low-density clusters. For such types of clusters, AFC usually breaks them into multiple smaller ones, which would inevitably increases the number of clusters in the outcomes. This limitation is caused by the inherent clustering mechanism. Nevertheless, one may partially lift this limitation by adjusting the level of granularity such that data samples of different classes can be well-separated with the minimum number of clusters.

Finally, it is also worth mentioning that during the numerical experiments, all the clustering algorithms involved for benchmark comparison use the same experimental settings as recommended by the literature. Performances of these clustering algorithms may be further improved if their externally controlled parameters are carefully tuned for each individual dataset. The main reason for using the same recommended parameter settings across the experiments is that the majority of existing clustering algorithms require proper experimental settings to achieve meaningful results and such experimental settings can vary a lot from problem to problem. However, prior knowledge in real-world applications is usually very limited. Predefining a set of externally controlled parameters without sufficient prior knowledge is often extremely challenging. In such cases, the recommended generic experimental settings play a very important role in helping users to get the preliminary clustering results. Therefore, the clustering results obtained by a particular algorithm with the generic experimental setting can serve as a good indicator of its efficacy in real-world applications.

### C. Discussions

Numerical examples presented in this section demonstrate that the proposed AFC algorithm is able to produce high-quality clustering results on a wide variety of benchmark problems. The proposed algorithm outperforms its competitors on a number of benchmark problems in both offline and online application scenarios (see Tables IV and VI), and its computational efficiency is also higher than the majority of alternatives. The numerical results presented in this paper demonstrate the efficacy of the proposed algorithm, showing the strong capability of AFC to handle both static and streaming data.

Meanwhile, one may notice that the level of granularity has a direct impact on the fineness of the clustering outcomes, which influence both the number of clusters in the clustering outcome and the computational efficiency of the proposed AFC algorithm. In general, a greater level of granularity enables AFC to give more focuses to the local patterns of data and group data into more clusters, this would also increase the computational complexity of cluster medoid optimization. If a lower level of granularity is chosen, AFC tends to focus more on main patterns of data. As a result, data will be partitioned coarsely and the clustering outcomes will have less clusters, but the computational efficiency of the algorithm will be much higher. In practice, users can start with the recommended values given by this paper and adjust the parameter setting based on the specific needs of the problems.

In addition, it has to be admitted that similar to other algorithms that employ PAM or other similar strategies, such as KM, FCM and ADP, the proposed AFC algorithm is less effective in capturing non-convex clusters and low-density clusters. For such types of clusters, AFC usually breaks them into multiple smaller ones, which would inevitably increases the number of clusters in the outcomes. This limitation is caused by the inherent clustering mechanism. Nevertheless, one may partially lift this limitation by adjusting the level of granularity such that data samples of different classes can be well-separated with the minimum number of clusters.

Finally, it is also worth mentioning that during the numerical experiments, all the clustering algorithms involved for benchmark comparison use the same experimental settings as recommended by the literature. Performances of these clustering algorithms may be further improved if their externally controlled parameters are carefully tuned for each individual dataset. The main reason for using the same recommended parameter settings across the experiments is that the majority of existing clustering algorithms require proper experimental settings to achieve meaningful results and such experimental settings can vary a lot from problem to problem. However, prior knowledge in real-world applications is usually very limited. Predefining a set of externally controlled parameters without sufficient prior knowledge is often extremely challenging. In such cases, the recommended generic experimental settings play a very important role in helping users to get the preliminary clustering results. Therefore, the clustering results obtained by a particular algorithm with the generic experimental setting can serve as a good indicator of its efficacy in real-world applications.

### VI. Conclusion and Future Works

This paper presented a novel data-driven fuzzy clustering algorithm named AFC. It employs a Gaussian-type membership function with the degree of fuzziness controlled by a self-adjusting kernel width, which is derived based on the mutual distances of data and the level of granularity externally controlled by users. The proposed algorithm firstly identifies a small number of highly representative samples in the data.
space as cluster medoids for initial partition, and further utilizes them to achieve the locally optimal partition through iterative optimization. In addition, an extension is introduced to the proposed algorithm for chunk-by-chunk data stream clustering. Numerical examples have demonstrated the efficacy of the proposed algorithm on a wide range of benchmark problems in both offline and online scenarios.

There are several considerations for future works. Firstly, the degree of fineness of the clustering outcomes obtained by the proposed algorithms is determined by the level of granularity, which is externally controlled by users. Although the level of granularity can be determined without prior knowledge of the problems, it may undermine the meaningfulness of the clustering results if not set properly. Thus, developing a fully autonomous approach to self-determine this parameter based on the ensemble properties of data will be very helpful. Secondly, similar to other online clustering algorithms, the proposed algorithm may return different results when clustering streaming data if the order of observed data samples is changed. This may lead to different conclusions in real-world applications. It is possible to address this issue by keeping all the historical data in the system memory and using them to optimize the partition, but a more computational efficient solution will be more helpful. Thirdly, as aforementioned, it would be very useful to develop a more robust version of the proposed algorithm capable to neutralize the negative effects of outliers. Finally, the proposed AFC algorithm is only tested on benchmark problems in this paper, it is worth using AFC in solving real-world problems to further test its efficacy.

REFERENCES
[28] J. Xu, J. Han, X. Kai, and F. Nie, “Robust and sparse fuzzy k-means clustering,” in International Joint Conference on Artificial Intelligence, 2016, pp. 2224–2230.