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 (FCMdd) 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 heterogenous 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

X. Gu is with the Department of Computer Science, Institute of Mathematics, Physics and Computer Science, Aberystwyth University, Aberystwyth, SY23 3DB, UK, email: xig4@aber.ac.uk

Q. Ni is with the School of Computing and Communications, Lancaster University, Lancaster, LA1 4WA, UK, email: q.ni@lancaster.ac.uk

G. Tang is with the School of Management Science and Engineering, Shandong University of Finance and Economics, Jinan, 250014, China, email: guolin_tang@163.com

Corresponding author: G. Tang.

Manuscript received XXXX XX, 2020; revised XXXX XX, 2020.

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 $\{\boldsymbol{x}\} = \{\boldsymbol{x}_1, \boldsymbol{x}_2, ..., \boldsymbol{x}_K\}$ be a static data set in a real N-dimensional data space, $\mathbf{R}^N$, where $\boldsymbol{x}_k = [x_{k,1}, x_{k,2}, ..., x_{k,N}]^T \in \mathbf{R}^N$ ($k = 1, 2, \ldots, K$); the subscript k stands for the time instance at which $\boldsymbol{x}_k$ is observed. In this paper, the commonly used Euclidean distance is employed as the default distance measure, namely, $||\boldsymbol{x}_i - \boldsymbol{x}_j|| = \sqrt{\sum_{n=1}^{N}(x_{i,n} - x_{j,n})^2}$.

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, $\boldsymbol{p}_i$ ($i = 1, 2, \ldots, C$). 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:

$$J(\boldsymbol{U}, \boldsymbol{P}) = \sum_{k=1}^{K}\sum_{i=1}^{C} \bar{\mu}_{i,k} ||\boldsymbol{x}_k - \boldsymbol{p}_i||^2 \tag{1}$$

where $\boldsymbol{U} = [\bar{\mu}_{i,k}]_{i=1:C}^{k=1:K}$ is the membership matrix; $\boldsymbol{P} = [\boldsymbol{p}_i]_{i=1:C}$ is the medoid matrix; $\bar{\mu}_{i,k}$ is the normalized fuzzy membership of $\boldsymbol{x}_k$ in the i^{th} cluster represented by the medoid, $\boldsymbol{p}_i$, subject to:

$$\bar{\mu}_{i,k} > 0 \text{ and } \sum_{i=1}^{C} \bar{\mu}_{i,k} = 1 \tag{2}$$

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

$$\bar{\mu}_{i,k} = \frac{\mu(\boldsymbol{p}_i, \boldsymbol{x}_k, \sigma_G)}{\sum_{j=1}^{C} \mu(\boldsymbol{p}_j, \boldsymbol{x}_k, \sigma_G)} \tag{3}$$

where $\mu(\boldsymbol{p}_i, \boldsymbol{x}_k, \sigma_G)$ is the Gaussian type fuzzy membership defined by Eqn. (4) [34].

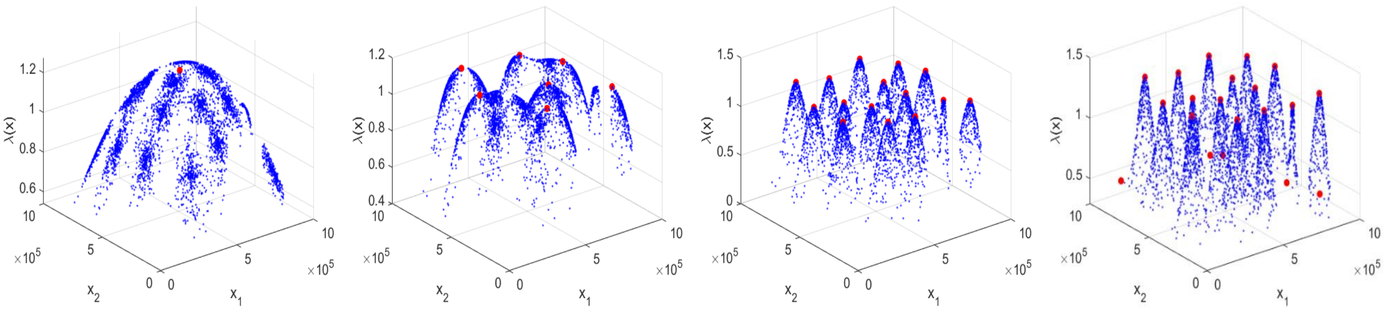
$$\mu(\boldsymbol{p}_i, \boldsymbol{x}_k, \sigma_G) = e^{-\frac{||\boldsymbol{p}_i - \boldsymbol{x}_k||^2}{\sigma_G^2}} \tag{4}$$

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, $\boldsymbol{P}$ 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, σ_G is derived directly from data based on their mutual distances and the level of granularity (G) controlled by users [36]:

$$\sigma_g^2 = \frac{1}{\sum_{i=1}^{K-1}\sum_{j=i+1}^{K} w_{g,i,j}} \sum_{i=1}^{K-1}\sum_{j=i+1}^{K} w_{g,i,j} ||\boldsymbol{x}_i - \boldsymbol{x}_j||^2 \tag{5}$$

where $g = 1, 2, \ldots, G$; G can be any non-negative integer chosen by users and $w_{g,i,j} = \begin{cases} 1, & \text{if } ||\boldsymbol{x}_i - \boldsymbol{x}_j|| \leq \sigma_{g-1} \\ 0, & \text{else} \end{cases}$. There is $\sigma_0^2 = \frac{2}{K(K-1)} \sum_{i=1}^{K-1}\sum_{j=i+1}^{K} ||\boldsymbol{x}_i - \boldsymbol{x}_j||^2$.



(a) $G = 1$ (b) $G = 3$ (c) $G = 4$ (d) $G = 5$

Fig. 1: Illustration of cumulative membership, $\lambda(\boldsymbol{x})$ (blue dots – data samples; red dots – local maxima).

The self-adjusting kernel width, σ_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 $\boldsymbol{U}_{micro}$ is obtained by using each micro-cluster medoid, $\boldsymbol{x}_k$ to assign memberships to all the K data samples in the data space (including itself):

$$\boldsymbol{U}_{micro} = [\bar{\mu}'_{k,j}]_{j=1:K}^{k=1:K} \tag{6}$$

where $\bar{\mu}'_{k,j} = \frac{\mu(\boldsymbol{x}_k,\boldsymbol{x}_j,\sigma_G)}{\sum_{i=1}^{K}\mu(\boldsymbol{x}_i,\boldsymbol{x}_j,\sigma_G)}$; $\mu(\boldsymbol{x}_k,\boldsymbol{x}_j,\sigma_G)$ is calculated by Eqn. (4).

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

$$\lambda(\boldsymbol{x}_k) = \sum_{j=1}^{K}\bar{\mu}'_{k,j} = \sum_{j=1}^{K}\frac{\mu(\boldsymbol{x}_k,\boldsymbol{x}_j,\sigma_G)}{\sum_{i=1}^{K}\mu(\boldsymbol{x}_i,\boldsymbol{x}_j,\sigma_G)} \tag{7}$$

The cumulative membership, $\lambda(\boldsymbol{x})$ 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. It can be observed from Fig. 1 that cumulative membership resembles the unimodal probability mass function if a smaller value of G is selected by users. In such cases, the cumulative membership discloses the main pattern of data. In contrast, if a greater G is chosen, cumulative membership discloses more information about the local models of data distribution. However, it is worth noting that the level of granularity can be determined merely based on users' preference without prior knowledge.

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

$$\begin{aligned} &\textit{Condition 1:}\text{ If } (\lambda(\boldsymbol{x}_k) > \max_{||\boldsymbol{x}_k-\boldsymbol{x}_j||\leq\sigma_G;k\neq j}(\lambda(\boldsymbol{x}_j))) \\ &\qquad\qquad \text{Then } (\boldsymbol{x}_k \in \{\boldsymbol{p}\}_C^0) \end{aligned} \tag{8}$$

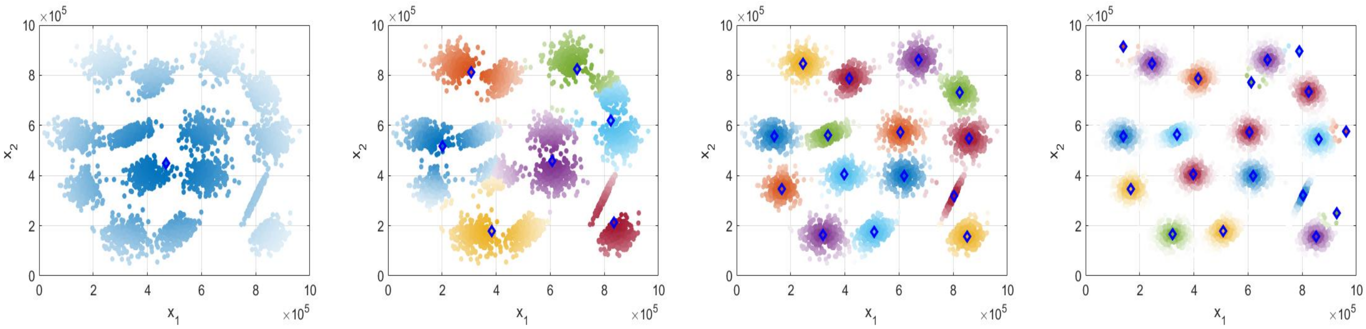
where C is the cardinality of $\{\boldsymbol{p}\}_C^0$. The local maxima of $\lambda(\boldsymbol{x})$ identified by Condition 1 are depicted in Fig. 1 as the red dots.

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

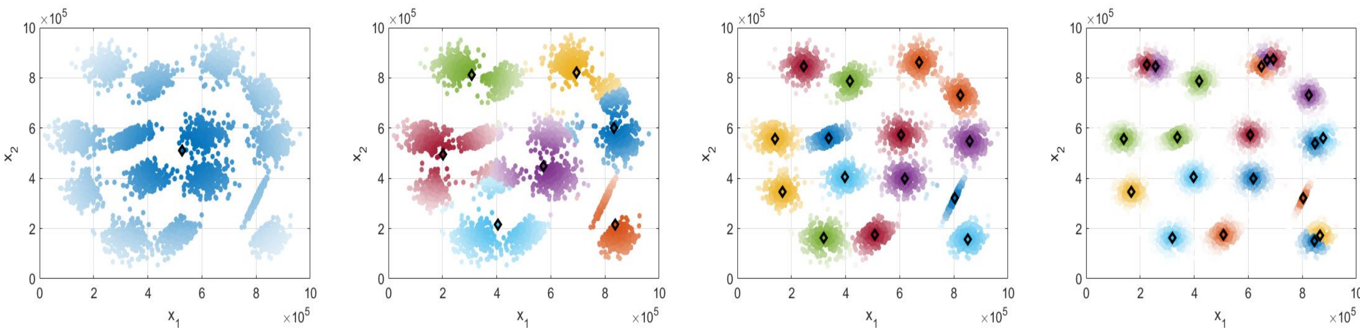
Step 1. Update $\boldsymbol{P}^{t-1}$ to $\boldsymbol{P}^t$ with fixed $\boldsymbol{U}^{t-1}$ by Eqn. (9):

$$\boldsymbol{p}_i^t = \underset{\boldsymbol{z}\in\{\boldsymbol{x}\}}{\arg\min}\left(\sum_{k=1}^{K}\bar{\mu}_{i,k}^{t-1}||\boldsymbol{x}_k-\boldsymbol{z}||^2\right) \tag{9}$$



(a) $G = 1$ (b) $G = 3$ (c) $G = 4$ (d) $G = 5$

Fig. 2: Obtained initial clustering results (dots– data samples; diamonds – cluster medoids).



(a) $G = 1$ (b) $G = 3$ (c) $G = 4$ (d) $G = 5$

Fig. 3: Obtained final clustering results (dots– data samples; diamonds – cluster medoids).

Then, update $\boldsymbol{U}^{t-1}$ to $\boldsymbol{U}^t$ with fixed $\boldsymbol{P}^t$ by Eqn. (10):

$$\bar{\mu}^t_{i,k} = \frac{\mu(\boldsymbol{p}^t_i, \boldsymbol{x}_k, \sigma_G)}{\sum_{j=1}^{C} \mu(\boldsymbol{p}^t_j, \boldsymbol{x}_k, \sigma_G)} \tag{10}$$

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

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

The iteration process continues until $J(\boldsymbol{U}^t, \boldsymbol{P}^t)$ reaches a locally minimum value. Once the iteration process is terminated, the final cluster medoid matrix, $\boldsymbol{P}^t$ and membership matrix, $\boldsymbol{U}^t$ are obtained as the algorithm's output, re-denoted by $\boldsymbol{P}$ and $\boldsymbol{U}$. Following the example given by Figs. 1 and 2, the final cluster medoids obtained after the iterative optimization process from S1 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, $\{\boldsymbol{x}\}$; *ii)* level of granularity, G;
outputs: cluster medoid matrix, $\boldsymbol{P}$;

obtain the self-adjusting kernel width, σ_G by (5);
obtain $\boldsymbol{U}_{micro}$ by (6);
calculate $\lambda(\boldsymbol{x})$ by (7);
obtain $\boldsymbol{P}^0$ by (8) and obtain $\boldsymbol{U}^0$ by (3);
calculate $J(\boldsymbol{U}^0, \boldsymbol{P}^0)$ by (1);
$t \leftarrow 1$;
while true do
  update $\boldsymbol{P}^t$ and $\boldsymbol{U}^t$ by (9) and (10);
  calculate $J(\boldsymbol{U}^t, \boldsymbol{P}^t)$ by (1);
  if ($J(\boldsymbol{U}^t, \boldsymbol{P}^t)$ converges) **then**
    break;
  else
    $t \leftarrow t + 1$;
  end if
end while
$\boldsymbol{P} \leftarrow \boldsymbol{P}^t$;
return $\boldsymbol{P}$.

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} cluster medoids in total, denoted by $\{\boldsymbol{p}\}^*_{L-1} = \{\boldsymbol{p}^*_{L-1,1}, \boldsymbol{p}^*_{L-1,2}, \dots, \boldsymbol{p}^*_{L-1,C^*_{L-1}}\}$, identified from the $L-1$ historical data chunks.

A. Algorithmic Procedure

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

$$\sigma_G^2 = \frac{\sum_{l=1}^{L} K_l \sigma_{G,l}^2}{\sum_{l=1}^{L} K_l} \tag{11}$$

where $\sigma_{G,l}$ is the kernel width calculated from $\{\boldsymbol{x}\}_l$ ($l = 1, 2, \dots, L$) using Eqn. (5); K_l is the corresponding chunk size. The algorithm then extracts a set of cluster medoids from $\{\boldsymbol{x}\}_L$, denoted by $\{\boldsymbol{p}\}_L = \{\boldsymbol{p}_{L,1}, \boldsymbol{p}_{L,2}, \dots, \boldsymbol{p}_{L,C_L}\}$ using ***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 $\{\boldsymbol{p}\}_L$ and $\{\boldsymbol{p}\}^*_{L-1}$ together to produce $\{\boldsymbol{p}\}^*_L$. However, combining them together directly is not the best solution because $\{\boldsymbol{p}\}_L$ may contain a portion of cluster medoids that represent similar patterns to some of cluster medoids identified from the previous chunks before. To achieve a more compact and concise data partition, the following two-step approach for cluster medoid selection is proposed:

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

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

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

$$\begin{cases} \{\boldsymbol{p}\}^1_L \leftarrow \{\boldsymbol{p}\}^1_L \cup \{\boldsymbol{p}_{L,j}\}, \text{if } \min_{\boldsymbol{p}^* \in \{\boldsymbol{p}\}^*_{L-1}} (||\boldsymbol{p}^* - \boldsymbol{p}_{L,j}||^2) \geq \sigma_G^2 \\ \{\boldsymbol{p}\}^2_L \leftarrow \{\boldsymbol{p}\}^2_L \cup \{\boldsymbol{p}_{L,j}\}, \text{else} \end{cases} \tag{12}$$

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

$$\begin{cases} \{\boldsymbol{p}\}^{*1}_{L-1} \leftarrow \{\boldsymbol{p}\}^{*1}_{L-1} \cup \{\boldsymbol{p}^*_{L-1,i}\}, \text{if } \min_{\boldsymbol{p} \in \{\boldsymbol{p}\}_L} (||\boldsymbol{p} - \boldsymbol{p}^*_{L-1,i}||^2) \geq \sigma_G^2 \\ \{\boldsymbol{p}\}^{*2}_{L-1} \leftarrow \{\boldsymbol{p}\}^{*2}_{L-1} \cup \{\boldsymbol{p}^*_{L-1,i}\}, \text{else} \end{cases} \tag{13}$$

Note that $\{\boldsymbol{p}\}^{*1}_{L-1}$ and $\{\boldsymbol{p}\}^1_L$ 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 $\{\boldsymbol{p}\}^*_L$: $\{\boldsymbol{p}\}^*_L \leftarrow \{\boldsymbol{p}\}^{*1}_{L-1} \cup \{\boldsymbol{p}\}^1_L$. In contrast, cluster medoids of $\{\boldsymbol{p}\}^{*2}_{L-1}$ and $\{\boldsymbol{p}\}^2_L$ 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 $\{\boldsymbol{p}\}^{*2}_{L-1}$ and $\{\boldsymbol{p}\}^2_L$ to join $\{\boldsymbol{p}\}^*_L$.

Condition 2: If $\left(\lambda_L(\boldsymbol{p}_i) > \max\limits_{\substack{||\boldsymbol{p}_i - \boldsymbol{p}_j|| \leq \sigma_G; \\ \boldsymbol{p}_j \in \{\boldsymbol{p}\}^{*2}_{L-1} \cup \{\boldsymbol{p}\}^2_L}} (\lambda_L(\boldsymbol{p}_j))\right)$
Then $(\{\boldsymbol{p}\}^*_L \leftarrow \{\boldsymbol{p}\}^*_L \cup \{\boldsymbol{p}_i\})$ (14)

where $\boldsymbol{p}_i \in \{\boldsymbol{p}\}^{*2}_{L-1} \cup \{\boldsymbol{p}\}^2_L$; $\lambda_L(\boldsymbol{p}_i)$ is the cumulative membership of $\boldsymbol{p}_i$ calculated with $\{\boldsymbol{x}\}_L$ by Eqn. (15):

$$\lambda_L(\boldsymbol{p}_i) = \sum_{j=1}^{K_L} \bar{\mu}_{i,j} = \sum_{j=1}^{K_L} \frac{\mu(\boldsymbol{p}_i, \boldsymbol{x}_{L,j}, \sigma_G)}{\sum_{k=1}^{M_L} \mu(\boldsymbol{p}_k, \boldsymbol{x}_{L,j}, \sigma_G)} \tag{15}$$

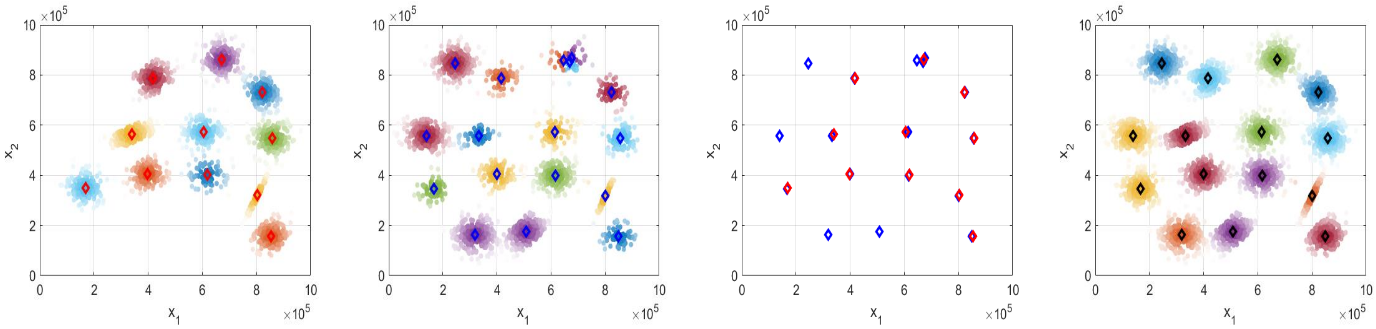
where $\bar{\mu}_{i,j} = \frac{\mu(\boldsymbol{p}_i, \boldsymbol{x}_{L,j}, \sigma_G)}{\sum_{k=1}^{M_L} \mu(\boldsymbol{p}_k, \boldsymbol{x}_{L,j}, \sigma_G)}$; M_L is the cardinality of $\{\boldsymbol{p}\}^{*2}_{L-1} \cup \{\boldsymbol{p}\}^2_L$. After this, the current processing cycle is finished, 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 $\{\boldsymbol{p}\}^{*2}_{L-1}$ and $\{\boldsymbol{p}\}^2_L$ and will be kept in $\{\boldsymbol{p}\}^*_L$. 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.



(a) Clustering result on chunk 1 (b) Clustering result on chunk 2 (c) Identified clustering medoids from two chunks (d) Final clustering results

Fig. 4: Chunk-by-chunk streaming data clustering with $G = 4$ (dots – data samples; diamonds – cluster medoids).

B. Summarization

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

Algorithm 2 AFC for streaming data clustering.

---

inputs: *i)* data chunks, $\{\boldsymbol{x}\}_1, \{\boldsymbol{x}\}_2, ..., \{\boldsymbol{x}\}_L$;
ii) level of granularity, G;
outputs: cluster medoid matrix, $\boldsymbol{P}$;

---

while ($\{\boldsymbol{x}\}_L$ is available) **do**
  obtain $\{\boldsymbol{p}\}_L$ from $\{\boldsymbol{x}\}_L$ using ***Algorithm 1***;
  if ($L = 1$) **then**
    $\{\boldsymbol{p}\}_L^* \leftarrow \{\boldsymbol{p}\}_L$;
  else
    update σ_G by (11);
    split $\{\boldsymbol{p}\}_L$ to $\{\boldsymbol{p}\}_L^1$ and $\{\boldsymbol{p}\}_L^2$ by (12);
    split $\{\boldsymbol{p}\}_{L-1}^*$ to $\{\boldsymbol{p}\}_{L-1}^{*1}$ and $\{\boldsymbol{p}\}_{L-1}^{*2}$ by (13);
    $\{\boldsymbol{p}\}_L^* \leftarrow \{\boldsymbol{p}\}_{L-1}^{*1} \cup \{\boldsymbol{p}\}_L^1$;
    expand $\{\boldsymbol{p}\}_L^*$ with $\{\boldsymbol{p}\}_{L-1}^{*2}$ and $\{\boldsymbol{p}\}_L^2$ by (14);
  end if
end while
$\boldsymbol{P}_L \leftarrow \{\boldsymbol{p}\}_L^*$;
return $\boldsymbol{P}_L$.

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, $\boldsymbol{U}_{micro}$. The complexity for this is $O(K^2)$. Then, AFC identifies C cluster medoids, $\{\boldsymbol{p}\}_C^0$ 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, $J(\boldsymbol{U}, \boldsymbol{P})$ to produce the ultimate data partition. Assuming that $J(\boldsymbol{U}, \boldsymbol{P})$ 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, $\{\boldsymbol{x}\}_L$. The computational complexity for AFC to partition $\{\boldsymbol{x}\}_L$ and extract C_L cluster medoids, $\{\boldsymbol{p}\}_L$ is $O(K_L^2 + H_L C_L K_L)$, where H_L stands for the number of iterations before $J(\boldsymbol{U}, \boldsymbol{P})$ converges. To merge $\{\boldsymbol{p}\}_L$ with the identified cluster medoids from historical chunks, $\{\boldsymbol{p}\}_{L-1}^*$, AFC firstly split $\{\boldsymbol{p}\}_L$ and $\{\boldsymbol{p}\}_{L-1}^*$ into $\{\boldsymbol{p}\}_L^1$, $\{\boldsymbol{p}\}_L^2$, $\{\boldsymbol{p}\}_{L-1}^{*1}$ and $\{\boldsymbol{p}\}_{L-1}^{*2}$, respectively, based on their mutual distances, and the computational complexity for this is $O(C_L C_{L-1}^*)$. The complexity for selecting out the more representative cluster medoids from$\{\boldsymbol{p}\}_L^2 \cup \{\boldsymbol{p}\}_{L-1}^{*2}$ using Condition 2 is $O(K_L M_L)$. Therefore, the complexity for AFC to process the L^{th} data chunk is $O(K_L^2 + H_L C_L K_L + C_L C_{L-1}^* + K_L M_L)$.

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

Dataset	Abbreviation	T	K	N
R15	R15	15	600	2
Aggregation	AG	7	788	2
S1	S1	15	5000	2
S2	S2	15	5000	2
S3	S3	15	5000	2
S4	S4	15	5000	2
Abalone	AB	3	4177	8
Spambase	SB	2	4601	57
Cardiotocography	CG	10	2126	21
Steel plate faults	SPF	7	1941	27
Multiple features	MF	10	2000	649
Pen-based handwritten digit recognition	PD	10	10992	16
Wine quality	WQ	7	6497	11
Occupancy detection	OD	2	20560	5
MNIST	MNIST	10	70000	784
Fashion MNIST	FMNIST	10	70000	784

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.

Detailed expressions of ARI, CHI, DBI and SC are given in the Supplementary Material.

Numerical examples are presented in the following subsection for evaluating the performance of the proposed algorithm. It is worth noting that ARI, CHI, DBI and SC may return abnormal values when the algorithms identify too many clusters. Such results are meaningless and not interpretable for human users. Therefore, the clustering results with $C > 0.25K$ are considered as invalid.

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

Algor ithm	Data set	Measure						Data set	Measure					
		C	ARI	CHI	DBI	SC	t_{exe}		C	ARI	CHI	DBI	SC	t_{exe}
AFC ($G=4$)	R15	15	0.986	4846.116	0.315	0.900	0.013	AG	8	0.736	1365.448	0.701	0.649	0.022
AFC ($G=5$)		16	0.974	4682.205	0.414	0.869	0.014		15	0.412	1658.009	0.774	0.610	0.022
FCM		15	0.976	4114.342	0.361	0.873	0.034		7	0.699	1258.777	0.810	0.620	0.030
MAL		12	0.727	1125.451	0.830	0.560	8.388		7	0.855	1201.975	0.667	0.633	3.035
KM		15	0.908	3481.683	0.470	0.826	0.031		7	0.748	1341.712	0.720	0.666	0.031
DBS		15	0.989	4869.330	0.314	0.901	0.029		6	0.875	730.496	0.592	0.536	0.042
MS		1	0.000	NaN	NaN	NaN	0.010		3	0.528	596.366	0.738	0.512	0.008
SUB		8	0.264	760.074	0.349	0.781	0.188		8	0.806	1262.916	0.694	0.626	0.197
NMI		1	0.000	NaN	NaN	NaN	0.388		8	0.805	1251.002	0.677	0.612	3.804
AP		15	0.986	4835.129	0.316	0.899	0.684		25	0.260	1606.389	0.806	0.552	1.346
GDD		10	0.480	789.314	0.614	0.533	0.105		5	0.809	754.200	0.625	0.514	0.178
CLA		15	0.989	4842.778	0.315	0.899	0.060		7	0.995	1208.391	0.505	0.645	0.091
LGC		15	0.989	4862.929	0.315	0.900	0.039		8	0.951	1131.303	0.602	0.594	0.050
ADP		15	0.993	4871.983	0.315	0.901	0.033		21	0.313	1638.066	0.793	0.579	0.042
AFC ($G=4$)	S1	15	0.986	22675.165	0.366	0.880	0.958	S2	15	0.936	13505.920	0.465	0.801	0.899
AFC ($G=5$)		20	0.901	18323.773	0.829	0.731	1.023		18	0.919	11633.442	0.604	0.759	0.882
FCM		15	0.970	21112.554	0.396	0.868	0.163		15	0.919	12345.288	0.508	0.778	0.278
MAL		15	0.884	11891.423	0.740	0.719	298.009		15	0.682	5459.124	1.037	0.443	287.936
KM		15	0.903	14190.740	0.510	0.817	0.035		15	0.905	12221.218	0.522	0.776	0.038
DBS		32	0.932	12395.415	0.783	0.651	0.818		35	0.736	3750.428	1.062	0.390	0.809
MS		4	0.210	2414.677	0.858	0.329	0.031		4	0.235	3233.804	0.861	0.367	0.026
SUB		10	0.708	8360.556	0.573	0.722	0.551		10	0.618	6258.652	0.663	0.616	0.546
NMI		6	0.479	5922.534	0.731	0.635	7.285		4	0.286	3966.144	0.806	0.501	8.625
AP		*(1911)*	*(0.336)*	*(207.948)*	*(0.742)*	*(0.307)*	*(97.992)*		*(1405)*	*(0.307)*	*(216.999)*	*(0.899)*	*(0.043)*	*(97.539)*
GDD		97	0.796	1139.831	0.566	-0.056	2.557		149	0.101	19.450	0.819	-0.609	2.699
CLA		15	0.987	22591.143	0.367	0.879	1.517		15	0.933	13209.972	0.474	0.793	1.519
LGC		16	0.977	21268.683	0.456	0.846	0.747		19	0.886	11099.194	0.675	0.691	0.801
ADP		15	0.987	22675.254	0.367	0.880	0.540		18	0.910	12109.958	0.615	0.761	0.564
AFC ($G=4$)	S3	12	0.609	6775.569	0.667	0.630	0.912	S4	16	0.588	5463.927	0.731	0.632	0.933
AFC ($G=5$)		23	0.641	6344.410	0.862	0.600	0.951		36	0.446	4648.732	0.934	0.485	1.125
FCM		15	0.718	7704.313	0.654	0.659	0.284		15	0.637	6181.359	0.650	0.642	0.256
MAL		15	0.444	2344.551	1.607	0.066	261.302		15	0.341	732.992	5.036	-0.046	257.821
KM		15	0.645	6635.234	0.755	0.619	0.042		15	0.597	5728.987	0.705	0.629	0.047
DBS		14	0.001	48.748	1.107	-0.638	0.827		10	0.001	73.739	0.899	-0.492	0.860
MS		1	0.000	NaN	NaN	NaN	0.029		2	0.009	341.753	0.829	0.231	0.027
SUB		7	0.422	5347.118	0.738	0.535	0.535		7	0.361	4233.038	0.808	0.525	0.547
NMI		15	0.727	7673.618	0.654	0.654	12.578		15	0.637	5806.551	0.692	0.614	15.908
AP		*(2165)*	*(0.226)*	*(74.298)*	*(0.760)*	*(0.106)*	*(95.359)*		*(2331)*	*(0.192)*	*(33.856)*	*(1.017)*	*(0.100)*	*(94.615)*
GDD		203	0.019	3.730	1.163	-0.793	2.546		123	0.038	14.929	0.619	-0.618	2.557
CLA		14	0.700	6815.032	0.652	0.621	1.402		15	0.572	4057.617	0.811	0.464	1.466
LGC		23	0.620	5003.103	0.893	0.523	0.771		22	0.549	3930.612	0.886	0.469	0.803
ADP		24	0.642	6429.828	0.85	0.599	0.547		28	0.555	4944.119	0.839	0.585	0.562

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.

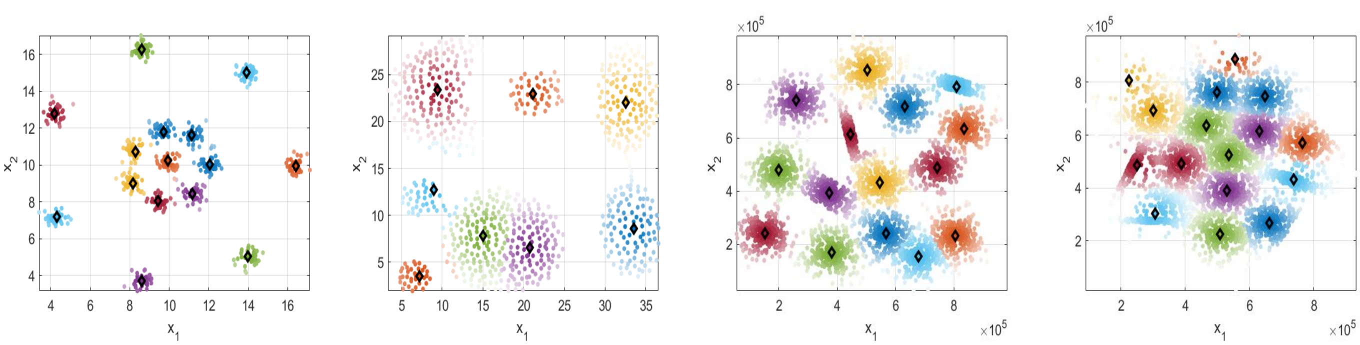
Furthermore, the following eight real-world datasets, namely, AB, SB, CG, SPF, MF, PD, WQ and OD are used for performance evaluation in offline scenarios. The performance of the proposed AFC algorithm is also competed with the same 12 comparative algorithms used before under the same six measures. The 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

Algorithm	Data set	C	ARI	CHI	DBI	SC	t_{exe}	Data set	C	ARI	CHI	DBI	SC	t_{exe}
AFC ($G=4$)		36	0.044	8822.611	0.485	0.709	0.727		184	0.100	22737.403	0.322	0.615	2.795
AFC ($G=5$)		83	0.027	9841.226	0.518	0.671	7.413		287	0.044	30434.437	0.322	0.489	3.950
FCM		3	0.133	7379.656	0.605	0.690	0.040		2	0.051	4426.995	0.615	0.900	0.117
MAL		3	0.158	1199.405	3.013	0.114	61.565		2	0.026	45.855	3.089	-0.373	66.385
KM		3	0.099	6692.543	0.616	0.668	0.031		2	0.024	3534.293	0.531	0.951	0.033
DBS		25	0.037	7993.289	0.901	0.711	0.663		1	0.000	NaN	NaN	NaN	1.238
MS	AB	3	0.000	0.957	0.966	-0.494	0.204	SB	313	-0.007	7.742	3.191	-0.764	1.317
SUB		5	0.099	567.188	3.504	-0.132	0.599		3	0.187	34.160	3.880	-0.758	6.646
NMI		5	0.000	84.941	0.359	0.826	14.05		21	0.032	881.318	0.287	0.842	334.160
AP		*(3948)*	*(0.005)*	*(12.242)*	*(0.192)*	*(0.956)*	*(66.181)*		*(3912)*	*(0.003)*	*(0.728)*	*(0.527)*	*(NaN)*	*(80.781)*
GDD		38	0.042	6860.490	0.576	0.597	3.611		25	0.062	1200.339	0.758	-0.439	16.629
CLA		14	0.039	10369.614	0.694	0.704	1.178		7	0.083	1439.844	1.886	-0.580	1.558
LGC		13	0.039	2668.466	0.895	0.523	0.799		6	0.104	601.683	1.004	-0.543	1.104
ADP		5	0.098	4919.895	0.577	0.639	0.421		12	0.069	3405.017	0.647	0.874	0.538
AFC ($G=4$)		47	0.105	234.801	1.290	0.250	0.206		32	0.064	5355.400	0.436	0.679	0.251
AFC ($G=5$)		116	0.057	179.489	1.091	0.253	0.202		59	0.053	8021.374	0.454	0.675	0.260
FCM		10	0.114	618.067	1.825	0.231	0.166		7	0.040	4458.406	0.683	0.582	0.114
MAL		10	0.102	277.281	2.830	-0.082	20.564		7	0.188	11.512	54.231	-0.368	25.827
KM		10	0.131	723.773	1.322	0.355	0.038		7	0.044	4477.523	0.598	0.652	0.035
DBS		14	0.043	64.498	1.352	-0.228	0.218		18	0.078	288.536	1.098	-0.582	0.186
MS	CG	*(509)*	*(0.053)*	*(47.951)*	*(1.002)*	*(-0.092)*	*(0.200)*	SPF	*(892)*	*(0.073)*	*(11.800)*	*(47.825)*	*(-0.249)*	*(0.246)*
SUB		165	0.036	80.821	2.152	-0.100	0.576		4	0.034	123.914	3.158	-0.283	0.413
NMI		322	0.076	67.695	0.658	0.170	19.688		9	-0.002	690.336	0.303	0.708	39.175
AP		44	0.064	370.961	1.301	0.290	6.024		*(1648)*	*(0.035)*	*(6.113)*	*(0.625)*	*(0.807)*	*(14.545)*
GDD		2	0.029	227.020	1.891	0.474	1.731		380	0.073	28.731	0.816	-0.509	1.797
CLA		1	0.000	NaN	NaN	NaN	0.506		7	0.075	3350.024	0.810	0.498	0.537
LGC		4	0.031	154.850	1.793	0.458	0.214		6	0.053	871.520	0.972	0.185	0.242
ADP		90	0.066	259.834	1.080	0.367	0.173		14	0.061	4509.210	0.717	0.676	0.116
AFC ($G=4$)		15	0.439	2436.069	1.599	0.358	0.458		45	0.361	1297.889	1.720	0.252	4.504
AFC ($G=5$)		74	0.303	873.157	1.881	0.174	0.618		154	0.168	666.815	1.766	0.164	5.124
FCM		10	0.418	3347.672	1.250	0.470	2.962		10	0.391	1800.896	3.493	0.331	0.952
MAL		10	0.140	16.108	8.367	-0.645	12.304		10	0.452	1844.936	2.161	0.254	651.733
KM		10	0.428	3286.262	1.256	0.454	0.126		10	0.556	2678.070	1.342	0.446	0.060
DBS		4	0.042	47.802	2.298	-0.689	0.402		38	0.404	463.082	1.649	0.015	4.490
MS	MF	*(1994)*	*(0.000)*	*(Inf)*	*(0.000)*	*(1.000)*	*(7.228)*	PD	1410	0.382	77.240	0.690	-0.020	9.682
SUB		*(1994)*	*(0.000)*	*(Inf)*	*(0.000)*	*(1.000)*	*(39.286)*		187	0.086	382.627	2.000	0.013	8.979
NMI		*(2000)*	*(0.000)*	*(NaN)*	*(0.000)*	*(NaN)*	*(886.717)*		*(4316)*	*(0.164)*	*(46.619)*	*(0.497)*	*(0.221)*	*(1643.690)*
AP		22	0.461	2098.746	1.470	0.318	7.096		254	0.077	604.713	1.541	0.238	136.345
GDD		*(1994)*	*(0.000)*	*(Inf)*	*(0.000)*	*(1.000)*	*(43.467)*		309	0.001	1.140	1.029	-0.684	39.852
CLA		5	0.317	539.145	2.064	-0.104	3.093		3	0.170	1229.101	2.101	0.087	6.944
LGC		8	0.546	2251.372	1.101	0.394	0.202		18	0.680	1551.212	1.463	0.311	3.584
ADP		54	0.356	1120.389	1.328	0.393	0.812		79	0.347	1057.977	1.326	0.382	2.898
AFC ($G=4$)		22	0.001	6206.402	0.683	0.489	2.541		89	0.126	59096.396	0.536	0.709	35.849
AFC ($G=5$)		42	0.002	4518.470	0.681	0.468	2.113		186	0.109	56560.042	0.616	0.673	44.025
FCM		7	0.002	12659.191	0.914	0.513	0.246		2	0.610	31449.037	0.685	0.778	0.066
MAL		7	0.044	970.235	6.049	-0.350	248.948		2	0.015	3240.895	2.027	0.114	157.313
KM		7	0.003	12741.375	0.886	0.525	0.061		2	0.593	31518.959	0.684	0.781	0.036
DBS		17	-0.003	81.231	1.616	-0.754	1.634		208	0.255	4383.688	1.474	-0.489	12.531
MS	WQ	12	0.000	10.859	2.113	-0.896	0.697	OD	6	-0.045	1164.777	1.909	-0.424	0.195
SUB		7	0.044	1480.964	3.596	-0.046	1.197		9	0.117	7017.468	5.247	-0.207	5.426
NMI		8	0.000	978.797	0.315	0.355	277.747		15	0.710	10922.511	0.331	0.737	224.019
AP		1421	0.001	115.997	0.355	0.309	160.394		*(18598)*	*(0.000)*	*(13.986)*	*(0.525)*	*(NaN)*	*(7084.711)*
GDD		10	0.002	14.945	3.299	-0.651	7.036		32	0.245	1046.534	1.615	-0.656	73.256
CLA		2	0.006	1185.590	0.461	0.390	2.574		26	0.195	6458.906	1.281	-0.377	28.515
LGC		3	-0.001	13.113	2.171	-0.535	1.451		19	0.296	14435.057	1.227	-0.151	14.496
ADP		21	0.000	7406.338	0.915	0.482	0.692		15	0.389	34653.493	0.603	0.761	7.577



(a) R15 (b) AG (c) S2 (d) S4

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

Algorithm	Data set	Measure C	ARI	CHI	DBI	SC	t_{exe}	Data set	Measure C	ARI	CHI	DBI	SC	t_{exe}
AFC ($G=4$)		50	0.035	8947.589	0.475	0.713	0.158		162	0.131	5738.480	1.841	0.614	0.494
AFC ($G=5$)		107	0.033	12856.95	0.509	0.650	0.529		252	0.078	5034.233	0.985	0.539	0.542
OKM		3	0.123	1836.795	1.921	0.250	0.405		2	0.000	0.608	1.114	-0.799	0.406
EC	AB	5	0.114	3737.389	0.872	0.302	0.146	SB	12	0.139	247.234	1.191	0.267	0.481
ELM		3	0.009	164.455	1.352	-0.514	0.277		91	-0.021	30.296	5.522	-0.867	4.691
OCA		3	0.131	716.284	2.593	0.078	0.404		4	0.001	25.101	18.593	-0.295	1.616
ADP		27	0.055	6681.103	0.893	0.538	0.319		34	0.112	2030.182	0.779	0.588	0.390
AFC ($G=4$)		69	0.114	165.456	0.975	0.266	0.087		40	0.057	5238.289	0.529	0.613	0.086
AFC ($G=5$)		180	0.067	134.953	0.889	0.280	0.096		76	0.049	8721.105	0.548	0.573	0.102
OKM		10	0.119	162.611	4.101	-0.085	1.002		7	0.143	24.965	18.629	-0.378	1.002
EC	CG	10	0.119	422.256	1.912	0.072	0.101	SPF	11	0.055	403.337	1.028	0.085	0.109
ELM		21	0.060	55.161	2.292	-0.238	0.791		7	0.011	140.786	5.182	-0.262	0.910
OCA		4	0.030	123.995	11.643	0.043	2.422		4	-0.001	14.595	23.169	-0.124	0.499
ADP		54	0.083	306.065	1.223	0.308	0.182		23	0.068	3482.571	0.668	0.542	0.165
AFC ($G=4$)		20	0.432	1992.035	1.476	0.412	0.207		48	0.536	1044.625	1.529	0.354	0.986
AFC ($G=5$)		120	0.375	540.966	1.301	0.302	0.322		226	0.351	369.892	1.325	0.240	1.196
OKM		10	0.165	56.009	8.808	-0.281	1.611		10	0.292	1087.244	2.260	0.131	0.212
EC	MF	9	0.237	1447.081	1.622	0.268	1.573	PD	10	0.305	1465.554	1.972	0.175	0.707
ELM		*(1988)*	*(0.000)*	*(26.181)*	*(0.249)*	*(0.990)*	*(37.776)*		17	0.056	367.445	1.794	0.023	3.297
OCA		1	0.000	NaN	NaN	NaN	0.157		2	0.095	335.773	5.886	0.054	1.037
ADP		12	0.357	2339.414	1.304	0.440	0.268		102	0.285	878.917	1.397	0.314	0.840
AFC ($G=4$)		32	-0.001	4208.644	0.693	0.445	0.362		131	0.121	50120.910	0.622	0.666	6.778
AFC ($G=5$)		62	0.001	3718.894	0.756	0.386	0.386		209	0.152	27088.020	0.730	0.543	7.785
OKM		7	0.026	628.519	8.450	-0.192	1.207		2	0.158	7700.005	3.937	0.239	0.421
EC	WQ	7	0.009	3977.496	1.391	0.134	0.269	OD	6	0.358	11618.046	1.117	0.376	0.856
ELM		10	0.003	100.751	3.945	-0.546	0.881		4	0.167	5088.854	1.635	-0.134	1.356
OCA		2	0.004	842.100	26.698	0.150	0.620		2	0.063	6642.216	1.257	0.151	1.841
ADP		39	0.003	5446.205	1.043	0.372	0.483		36	0.165	37429.119	0.685	0.679	1.406

TABLE IV: OVERALL STATIC DATA CLUSTERING PERFORMANCE RANKS

Algorithm	Measure ARI	CHI	DBI	SI	Overall
AFC (G=4)	5.3	3.2	3.8	3.5	4.0
AFC (G=5)	8.2	4.8	6.5	6.4	6.5
FCM	4.9	3.6	6.1	3.6	4.6
MAL	6.6	9.0	11.2	8.8	8.9
KM	5.3	4.1	5.5	3.6	4.6
DBS	7.9	9.6	8.7	10.1	9.1
MS	12.1	12.7	10.8	12.5	12.0
SUB	8.5	9.3	9.6	9.2	9.1
NMI	8.7	9.1	5.4	7.1	7.6
AP	11.8	10.8	11.4	11.4	11.4
GDD	9.2	11.0	7.8	11.0	9.8
CLA	5.1	6.2	6.0	6.7	6.0
LGC	5.6	7.4	6.8	6.8	6.7
ADP	5.8	4.3	5.4	4.3	5.0

TABLE VI: OVERALL STREAMING DATA CLUSTERING PERFORMANCE RANKS

Algorithm	Measure ARI	CHI	DBI	SI	Overall
AFC (G=4)	3.5	2.0	2.1	1.5	2.3
AFC (G=5)	4.4	3.1	1.8	2.5	3.0
OKM	3.1	5.0	5.6	5.6	4.8
EC	2.4	3.0	4.0	4.1	3.4
ELM	5.8	6.3	5.3	6.8	6.1
OCA	5.3	6.2	6.7	5.4	5.9
ADP	3.6	2.4	2.5	2.0	2.6

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

TABLE VII: PERFORMANCE DEMONSTRATION ON LARGE-SCALE, HIGH-DIMENSIONAL PROBLEMS

Algorithm	Data set	Measure						Data set	Measure					
		C	ARI	CHI	DBI	SC	t_{exe}		C	ARI	CHI	DBI	SC	t_{exe}
AFC ($G=5$)	MNIST	28	0.156	878.279	3.896	0.022	173.384	FMNIST	50	0.254	2207.523	2.991	0.090	122.551
AFC ($G=6$)		520	0.082	133.916	3.244	0.015	234.361		258	0.199	557.757	2.823	0.062	152.751
ADP		4687	0.025	31.577	2.077	0.065	16234.741		967	0.068	221.144	2.157	0.090	2533.637

aforementioned five well-known online clustering algorithms on the same eight real-world benchmark datasets. In this example, L is set as 5 for AFC. The performance comparison is presented in Table V. For better illustration, the ranks of these online clustering algorithms per dataset per criterion are given by Table S7 in the Supplementary Material, and the overall ranks are tabulated in Table VI. It can be seen from Table VI that AFC with $G = 4$ is able to outrank its competitors in terms of CHI and SI . Meanwhile, AFC with $G = 5$ is ranked the top in terms of DB. Very importantly, by considering all four criteria, AFC is able to rank at the first and third places with the two parameter settings. This shows that AFC is capable of obtaining high-quality clustering results on streaming data in online application scenarios.

In the final numerical example, AFC is tested on MNIST and FMNIST datasets to evaluate its clustering performance on large-scale, high-dimensional problems. Images of both datasets have been converted to vectors prior to the experiments. Due to the very large scale of the two problems, both datasets are randomly divided into seven chunks with 10000 vectors in each chunk, and the proposed algorithm groups the data chunk-by-chunk. During the experiments, two different parameter settings of AFC are considered, namely, $G = 5$ and $G = 6$. The evolving version of the ADP algorithm is used for benchmark comparison thanks to its strong capability of handling large-scale streaming data problems [16]. The numerical results obtained by the two clustering algorithms are reported in Table VII, where it can be observed that the computational efficiency of AFC is much higher than ADP, and the quality of its clustering results is far better as suggested by CHI.

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

[1] A. Saxena et al., "A review of clustering techniques and developments," *Neurocomputing*, vol. 267, pp. 664–681, 2017.

[2] J. Zhou, Z. Lai, D. Miao, C. Gao, and X. Yue, "Multigranulation rough-fuzzy clustering based on shadowed sets," *Inf. Sci. (Ny).*, vol. 507, pp. 553–573, 2020.

[3] P. Maji and S. K. Pal, "Rough-fuzzy c-medoids algorithm and selection of bio-basis for amino acid sequence analysis," *IEEE Trans. Knowl. Data Eng.*, vol. 19, no. 6, pp. 859–872, 2007.

[4] Y. Tang, F. Ren, and W. Pedrycz, "Fuzzy c-means clustering through SSIM and patch for image segmentation," *Appl. Soft Comput.*, vol. 87, p. 105928, 2020.

[5] P. D'Urso and R. Massari, "Fuzzy clustering of mixed data," *Inf. Sci. (Ny).*, vol. 505, pp. 513–534, 2019.

[6] J. B. MacQueen, "Some methods for classification and analysis of multivariate observations," in *Berkeley symposium on mathematical statistics and probability*, pp. 281–297, 1967.

[7] H. S. Park and C. H. Jun, "A simple and fast algorithm for K-medoids clustering," *Expert Syst. Appl.*, vol. 36, no. 2, pp. 3336–3341, 2009.

[8] M. Ester, H. P. Kriegel, J. Sander, and X. Xu, "A density-based algorithm for discovering clusters in large spatial databases with noise," in *International Conference on Knowledge Discovery and Data Mining*, 1996, vol. 96, pp. 226–231.

[9] T. Zhang, R. Ramakrishnan, and M. Livny, "BIRCH: an efficient data clustering method for very large databases," in *ACM SIGMOD International Conference on Management of Data*, 1996, pp. 103–114.

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

[11] A. Corduneanu and C. M. Bishop, "Variational Bayesian model selection for mixture distributions," in *International Conference on Artificial intelligence and Statistics*, pp. 27–34, 2001.

[12] D. Comaniciu and P. Meer, "Mean shift: a robust approach toward feature space analysis," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 24, no. 5, pp. 603–619, 2002.

[13] A. Rodriguez and A. Laio, "Clustering by fast search and find of density peaks," *Science*, vol. 344, no. 6191, pp. 1493–1496, 2014.

[14] E. Gungor and A. Ozmen, "Distance and density based clustering algorithm using Gaussian kernel," *Expert Syst. Appl.*, vol. 69, pp. 10–20, 2017.

[15] Z. Wang et al., "Clustering by local gravitation," *IEEE Trans. Cybern.*, vol. 48, no. 5, pp. 1383–1396, 2018.

[16] X. Gu, P. P. Angelov, and J. C. Principe, "A method for autonomous data partitioning," *Inf. Sci. (Ny).*, vol. 460–461, pp. 65–82, 2018.

[17] Y. Chen et al., "Fast density peak clustering for large scale data based on kNN," *Knowledge-Based Syst.*, vol. 187, p. 104824, 2020.

[18] P. D'Urso, "Fuzzy clustering," in *Handbook of Cluster Analysis*, H. Christian, M. Marina, M. Fionn, and R. Roberto, Eds. Chapman & Hall, 2015, pp. 545–574.

[19] Y. Lin and S. Chen, "A centroid auto-fused hierarchical fuzzy c-means clustering," *IEEE Trans. Fuzzy Syst.*, DOI: 10.1109/TFUZZ.2020.2991306, 2020.

[20] J. C. Dunn, "A fuzzy relative of the ISODATA process and its use in detecting compact well-separated clusters," *J. Cybern.*, vol. 3, no. 3, 1973.

[21] J. C. Bezdek, "Fuzzy mathematics in pattern classification," *Cornell University*, 1973.

[22] E. H. Ruspini, J. C. Bezdek, and J. M. Keller, "Fuzzy clustering: a historical perspective," *IEEE Comput. Intell. Mag.*, vol. 14, no. 1, pp. 45–55, 2019.

[23] R. Krishnapuram, A. Joshi, O. Nasraoui, and L. Yi, "Low-complexity fuzzy relational clustering algorithms for Web mining," *IEEE Trans. Fuzzy Syst.*, vol. 9, no. 4, pp. 595–607, 2001.

[24] D. Q. Zhang and S. C. Chen, "Clustering incomplete data using kernel-based fuzzy C-means algorithm," *Neural Process. Lett.*, vol. 18, no. 3, pp. 155–162, 2003.

[25] D. Q. Zhang, S. C. Chen, Z. S. Pan, and K. R. Tan, "Kernel-based fuzzy clustering incorporating spatial constraints for image segmentation," in *International Conference on Machine Learning and Cybernetics*, 2003, pp. 2189–2192.

[26] L. Chen, C. L. P. Chen, and M. Lu, "A multiple-kernel fuzzy c-means algorithm for image segmentation," *IEEE Trans. Syst. Man, Cybern. Part B*, vol. 41, no. 5, pp. 1263–1274, 2011.

[27] L. Chen, L. Guo, X. Lu, and C. L. P. Chen, "Fuzzy clustering method with graph-based regularization," in *International Conference on Fuzzy Theory and Its Applications*, pp. 1-6, 2017.

[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.

[29] L. Guo, L. Chen, X. Lu, and C. L. P. Chen, "Membership affinity lasso for fuzzy clustering," *IEEE Trans. Fuzzy Syst.*, vol. 28, no. 2, pp. 294–307, 2020.

[30] M.S. Yang and Y. Nataliani, "Robust-learning fuzzy c-means clustering algorithm with unknown number of clusters," *Pattern Recognit.*, vol. 71, pp. 45–59, 2017.

[31] M. J. Li, M. K. Ng, Y. M. Cheung, and J. Z. Huang, "Agglomerative fuzzy K-Means clustering algorithm with selection of number of clusters," *IEEE Trans. Knowl. Data Eng.*, vol. 20, no. 11, pp. 1519–1534, 2008.

[32] U. Kaymak and M. Setnes, "Fuzzy clustering with volume prototypes and adaptive cluster merging," *IEEE Trans. Fuzzy Syst.*, vol. 10, no. 6, pp. 705–712, 2002.

[33] P. D'Urso and J. M. Leski, "Fuzzy clustering of fuzzy data based on robust loss functions and ordered weighted averaging," *Fuzzy Sets Syst.*, vol. 389, pp. 1–28, 2020.

[34] X. Gu, Q. Shen, and P. P. Angelov, "Particle swarm optimized autonomous learning fuzzy system," *IEEE Trans. Cybern.*, DOI: 10.1109/TCYB.2020.2967462, 2020.

[35] T. C. Havens, J. C. Bezdek, C. Leckie, L. O. Hall, and M. Palaniswami, "Fuzzy c-means algorithms for very large data," *IEEE Trans. Fuzzy Syst.*, vol. 20, no. 6, pp. 1130–1146, 2012.

[36] X. Gu and P. P. Angelov, "Self-organising fuzzy logic classifier," *Inf. Sci. (Ny).*, vol. 447, pp. 36–51, 2018.

[37] E. Lughofer and P. Angelov, "Handling drifts and shifts in on-line data streams with evolving fuzzy systems," *Appl. Soft Comput.*, vol. 11, no. 2, pp. 2057–2068, 2011.

[38] J. C. Bezdek, R. Ehrlich, and W. Full, "FCM: the fuzzy c-means clustering algorithm," *Comput. Geosci.*, vol. 10, no. 2–3, pp. 191–203, 1984.

[39] S. L. Chiu, "Fuzzy model identification based on cluster estimation," *J. Intell. Fuzzy Syst.*, vol. 2, no. 3, pp. 267–278, 1994.

[40] J. Li, S. Ray, and B. G. Lindsay, "A nonparametric statistical approach to clustering via mode identification," *J. Mach. Learn. Res.*, vol. 8, no. 8, pp. 1687–1723, 2007.

[41] L. Hubert and P. Arabie, "Comparing partitions," *J. Classif.*, vol. 2, no. 1, pp. 193–218, 1985.

[42] T. Calinski and J. Harabasz, "A dendrite method for cluster analysis," *Commun. Stat. Methods*, vol. 3, no. 1, pp. 1–27, 1974.

[43] D. L. Davies and D. W. Bouldin, "A cluster separation measure," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 2, pp. 224–227, 1979.

[44] P. J. Rousseeuw, "Silhouettes: a graphical aid to the interpretation and validation of cluster analysis," *J. Comput. Appl. Math.*, vol. 20, pp. 53–65, 1987.

[45] P. Angelov, "An approach for fuzzy rule-base adaptation using on-line clustering," *Int. J. Approx. Reason.*, vol. 35, no. 3, pp. 275–289, 2004.

[46] S. Zhong, "Efficient online spherical k-means clustering," in *International Joint Conference on Neural Networks*, 2005, pp. 3180–3185.

[47] R. D. Baruah and P. Angelov, "Evolving local means method for clustering of streaming data," in *IEEE International Conference on Fuzzy Systems*, 2012, pp. 10–15.

[48] M. Chenaghlou, M. Moshtaghi, C. Leckie, and M. Salehi, "Online clustering for evolving data streams with online anomaly detection," in *Pacific-Asia Conference on Knowledge Discovery and Data Mining*, 2018, pp. 508–521.