Applied Intelligence (2019) 49:1228–1244
https://doi.org/10.1007/s10489-018-1324-x

FGCH: a fast and grid based clustering algorithm for hybrid data stream

Jinyin Chen[1] · Xiang Lin[1] · Qi Xuan[1] · Yun Xiang[1]

Published online: 30 October 2018

Abstract

Streaming large volumes of data has a wide range of real-world applications, e.g., video flows, internet calls, and online games etc. Thus, fast and real-time data stream processing is important. Traditionally, data clustering algorithms are efficient and effective to mine information from large data. However, they are mostly not suitable for online data stream clustering. Therefore, in this work, we propose a novel fast and grid based clustering algorithm for hybrid data stream (FGCH). Specifically, we have made the following main contributions: 1), we develop a non-uniform attenuation model to enhance the resistance to noise; 2), we propose a similarity calculation method for hybrid data, which can calculate the similarity more efficiently and accurately; and 3), we present a novel clustering center fast determination algorithm (CCFD), which can automatically determine the number, center, and radius of clusters. Our technique is compared with several state-of-art clustering algorithms. The experimental results show that our technique can achieve more than better clustering accuracy on average. Meanwhile, the running time is shorter compared with the closest algorithm.

Keywords Data stream · Clustering analysis · Non-uniform attenuation · Grid clustering

1 Introduction

With the explosive development of next generation of ethernet, such as internet of things and 5G network, unprecedent amount of data are generated online. Most of the data are in the form of continuous flow, i.e., data stream. Data stream is common in many applications, e.g., online music, surveillance camera, and video phone etc [1, 2]. Therefore, data stream clustering, which aims to partition desired knowledge from huge data streams, is becoming active and popular [3, 4]. However, existing data stream clustering algorithms still face numerous challenges, such as noisy points, parameter sensitivities, and hybrid data integrations. To address those problems, we propose FGCH, a fast and grid based clustering algorithm to process hybrid data streams.

Stream clustering is an effective way to mine hidden information inside tremendous data flows [5, 6]. It can deal with evolving distributions and find clusters dynamically. Therefore, an efficient and effective data stream clustering algorithm can have many real-world application scenarios.

✉ Yun Xiang
xiangyun@zjut.edu.cn

1 The College of Information Engineering, Zhejiang University of Technology, Hangzhou, China

For example, stream clustering algorithms can dynamically analyze the product supply and demand relationships in marketing; they can effectively filter out noisy points and harmful information through information filtering; and they can also provide personalized recommendation for individual user in social networks [7–9]. In general, data stream clustering algorithms are useful, effective, and widely popular. However, clustering GBS, or even TBs, of data in a very short and rigid period of time is non-trivial task.

There are many challenges and tradeoffs for existing data stream clustering algorithms. For example, although processed with density attenuation models, existing algorithms still suffer from significant noise problems [10, 11]. Besides, current clustering algorithms can not divide arbitrarily shaped clusters and determine cluster centers automatically [12]. Moreover, data streams often contain hybrid data, which includes both numeric and categorical values. Thus, the current data stream clustering algorithms, which are tuned for single dimensional data, do not handle hybrid data well. They typically employ Euclidean distance, Hamming distance, or other matching calculation methods to evaluate the similarity between data points [13]. However, their poor migration capabilities often cause inferior results.

In general, to address these problems, we propose FGCH, a fast and grid based clustering algorithm for data stream with hybrid attributes. Our technique gets improvements based upon previous algorithms from three aspects, which are similarity calculation method, online decay model, and off-line clustering algorithm, respectively.

Specifically, the main contributions of this work can be generalized as follows.

1) We propose the FGCH algorithm to process the hybrid data. It employs a novel non-uniform attenuation model to increase the algorithm resilience to noise.
2) We develop a similarity calculation technique based on frequency distribution and dimension correlation analysis. Our method can significantly improve the distance estimation accuracy for hybrid data.
3) We design the CCFD algorithm, which is based on density-distance clustering. It can automatically determine the clustering centers and the density radius.

The experimental results on simulated and real-world datasets demonstrate that our technique has the best performance compared with state-of-art data stream clustering algorithms with significantly less running time.

The rest of this paper is organized as follows. Section 2 introduces related works. Section 3 discusses the basic theories and techniques. Section 4 describes the specific method of FGCH and, presents the experimental data analysis results compared with other state-of-art data stream clustering algorithms. Section 6 concludes the paper.

1.1 System flow

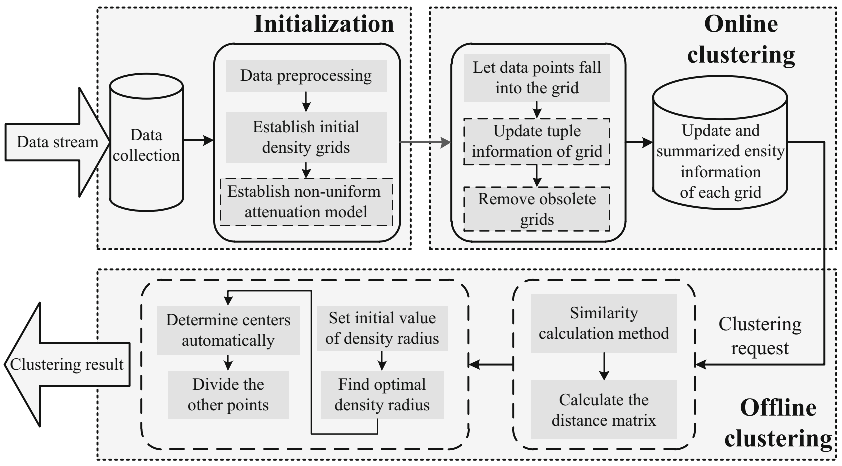
Figure 1 shows the system framework of the FGCH algorithm. The algorithm includes three main parts, which are data initialization, online clustering, and offline clustering, respectively. During the initialization stage, the data stream is first pre-processed. Then we perform dimensional analysis to determine the grid granularity and the initial grid location of each data dimension. Then we process the grid based data using our non-uniform attenuation model. After the initialization, the data then flows into the designated grid in the online clustering stage. Whenever a new data point arrives, the number of grids is updated and the grid density information is re-calculated. After the data stream is processed by the online clustering stage, the data is further processed in offline stage based on user request. During offline stage, the grid is treated as a virtual data point located at the grid geometric center. The similarity calculation algorithm is employed to calculate the distance matrix of all the virtual points. Then we automatically determine the density radius and cluster centers using our improved climbing and CCFD algorithms. Thus, we can derive the final clustering result with optimal density radius.

2 Related work

The related work can be generalized into two categories, hybrid data processing and stream clustering techniques.

2.1 Hybrid data processing

In order to process hybrid data, Huang et al. present K-prototypes algorithm [14], which uses a cost function for hybrid datasets along with definitions of cluster centers and distances. Ji et al. propose WFK-prototypes [15], which combines the mean values of fuzzy centroid to represent the prototypes of the cluster and extend K-prototypes with significance concepts. Gath et al. develop FCM algorithm [16], which is based on the implicit assumption that the clusters can be modeled using Gaussian distributions. The clustering distance is defined as the negative log likelihood of the Gaussian distribution. KL-FCM-GM [17] is an extension of FCM, which is based on the Gauss-Multinomial data distributed. Li et al. design similarity-based agglomerative clustering (SBAC) algorithm [18] based on biometric classification, where



Initialization
Data stream
Data collection
Data preprocessing
Establish initial density grids
Establish non-uniform attenuation model
Online clustering
Let data points fall into the grid
Update tuple information of grid
Remove obsolete grids
Update and summarized ensity information of each grid
Clustering request
Similarity calculation method
Calculate the distance matrix
Set initial value of density radius
Find optimal density radius
Determine centers automatically
Divide the other points
Clustering result
Offline clustering

Fig. 1 The FGCH system framework

greater weights are assigned to uncommon features in the population. However, the computational cost is expensive. Based on the variance and entropy, Hsu et al. propose a distance hierarchy based clustering algorithm [19]. However, this algorithm requires the domain expertise and hence, is infeasible to be implemented in real-world applications. Ryu et al. defines similarity as a combination of qualitative and quantitative features and thus, solve the multi-valued data clustering problem [20]. Marie et al. combines the Hausdorff distance and a dynamic clustering algorithm [21]. Chen et al. developed the CLUB algorithm [22], which identifies the cluster densities based on k nearest neighbours. Silva et al. designed a Fast Evolutionary Algorithm that allows estimating k automatically [23]. This method uses the PagëHinkley Test to detect eventual degradation and trigger an evolutionary algorithm that re-estimates k accordingly. Rodriguez et al. propose a clustering algorithm combining partitioning around medoids methods and density based methods [24], where cluster centers have higher density and larger distances. However, their approach can not find cluster centers automatically, and is sensitive to density radius.

2.2 Stream clustering techniques

Based on static clustering algorithms, researchers have proposed improved algorithms for data streams. Mishra et al. propose the STREAM algorithm [25]. The algorithm first sets m buckets, then divides the points into k clusters. Aggarwal et al. develop Clustream [26], which divides the whole framework into online and offline stages. During online stage, micro-clusters are updated and stored using pyramid time frame structure. In offline stage, the micro-clusters are processed using static clustering algorithm. Based on Clustream, Aggarwal et al. propose HPStream [27], which has better performance for high-dimensional data stream using online attenuation model. Based on density clustering, Cao et al. propose DenStream [28]. It extends the DBSCAN algorithm in offline stage, and use potential cluster to improve the processing power of outliers. Kremer et al. proposed Cluster Mapping Measure (CMM) [29]. CMM can effectively indicate different types of errors by considering evolving data streams. Hyde et al. present a fully online technique to cluster evolving data streams into arbitrary shaped clusters [30]. This algorithm is accurate, robust, and efficient. Based on conventional Kohonen neurons, Bodyanskiy et al. propose a new evolving cascade connection system [31], where a clustering process is implemented with different fuzzifier values in every system node. Based on Dirichlet process mixture (DPM) [32], Huynh et al. developed a class of variational methods for posterior inference [33]. Their methods perform better on streaming learning with text corpora. EXCC [34] is an exclusive and comprehensive clustering algorithm for heterogeneous data stream. Techniques such as dynamic thresholds, wait and watch policy etc., are implemented to detect noises at the cost of memory and processing time. Barddal et al. present the social network clustering stream (SNCStream) algorithm [35], which requires no offline processing. Based on SNCStream, they further proposed SNCStream+ [36], which uses network and evolution techniques to process the data stream. Xu et al. designed a fat node leading tree based stream clustering method [37]. Han et al. uses Extreme Learning Machine (ELM) and MapReduce to handle uncertain data streams [38], which could classify uncertain data streams and handle concept drift effectively. Sang et al. proposes multiple data streams clustering [39], which consider the geometric structure of both the data and feature manifold instead of clustering multiple data streams periodically. Wang et al. proposes a semi-supervised ensemble of classifiers approach [40], for learning in time-varying data streams. In addition, the ADaptive WINdowing (ADWIN2) is introduced to deal with concept drift, which makes it adapt to the varying environment.

3 Preliminaries

In this section, we introduce the fundamental theories and algorithms.

3.1 Uniform attenuation model

Currently, most stream clustering algorithms employ the uniform attenuation model [18]. In the problem definition, for any data points x, if it is inside a grid t_c, then we have the data arriving time $T(x) = t_c$, and the density at time t can be calculated as follows.

$$D(x, t) = 2^{-\lambda(t-T_c)} \tag{1}$$

where λ is the system attenuation factor. Then for any grid den, assuming that C is the set of data points inside den before time t, the density of den can be calculated using the following equation.

$$D(den, t) = \sum_{x \in C} D(x, t) \tag{2}$$

Thus, if grid den receives a new data point at t_p, and the previous density update time is t_l, the grid density can be updated using the following equation.

$$D(den, t_p) = 2^{-\lambda(t_p-t_l)} D(den, t_l) + 1 \tag{3}$$

The uniform attenuation model can effectively solve the noise accumulation problem and improve the processing speed. However, in uniform attenuation model, the attenuation coefficient λ is manually selected and fixed. Therefore, it is vulnerable to the upcoming noise points with abnormal distributions. It is desirable to develop a new non-uniform attenuation model, where attenuation coefficients are changeable according to the dispersion degree of the current data stream.

3.2 Similarity calculation algorithm

The most widely used similarity measurement method is the SBAC algorithm [18]. It calculates the distance based on the eigenvalue frequency for each data dimension. It also employs different calculation methods for numerical and categorical data, respectively. For the detailed algorithm, assuming in the kth classification dimension, there are four data objects, which are denoted as i, j, l, and m, respectively. Thus, the similarity scores can be calculated using the following equation.

$$\left.\begin{array}{l}((V_i)_k = (V_j)_k) \wedge ((V_l)_k = (V_m)_k) \\ ((p_i)_k = (p_j)_k) > ((p_l)_k = (p_m)_k)\end{array}\right\} \rightarrow similar(i,j)_k < similar(l,m)_k$$

where for dimension k, $(V_i)_k$ is the value, $(p_i)_k$ is the frequency, and $similar(i,j)_k$ is the similarity score between corresponding nodes.

Similarly, for numerical and ordinal dimensions, the similarity score can be calculated as

$$\left.\begin{array}{l}((V_i)_k - (V_j)_k) = ((V_l)_k - (V_m)_k) \\ \sum\limits_{t=(V_i)_k}^{(V_j)_k} p_t > \sum\limits_{t=(V_l)_k}^{(V_m)_k} p_t\end{array}\right\} \rightarrow similar(i,j)_k < similar(l,m)_k$$

where $\sum_{t=(V_i)_k}^{(V_j)_k} p_t$ is the sum of data point frequencies contained in $[(V_i)_k, (V_j)_k]$.

3.3 RLM algorithm

RLM algorithm [24] is the state-of-art clustering center determination algorithm. It assumes that the clustering centers are surrounded with low density neighbors and high density data points are relatively far away from each other. Specifically, for any virtual data point i, its local density ρ is defined as

$$\rho_i = \sum f(d_{ij} - d_c) \tag{4}$$

where d_{ij} is the distance and function $f(d_{ij} - d_c)$ can be calculated using the following equation.

$$f(d_{ij} - d_c) = \begin{cases} w_j & d_{ij} \le d_c \\ 0 & d_{ij} > d_c \end{cases} \tag{5}$$

where w_j is the density of grid j. Moreover, for any virtual data point i, its distance to another object j with larger local density is defined as

$$\delta_i = min(d_{ij}), \quad \rho_j \ge \rho_i \tag{6}$$

where δ is the corresponding distance.

Therefore, we can derive a $\rho - \delta$ decision graph as shown in Fig. 2. In the figure, A1, A2, and A3 are clustering centers, which have both high ρ and high δ. B1, B2, and B3 are the noise points, which have low ρ and high δ. Other points are boundary points which belong to a certain cluster. They typically have low δ.

4 The FGCH algorithm

In this section, we describe in detail our proposed FGCH algorithm. We first present the system flow of the algorithm. Then we discuss our non-uniform attenuation model, similarity calculation method, and CCFD algorithm, respectively.

4.1 Data initialization

The data initialization stage consists of three parts, which are data preprocessing, initial density grids creation and non-uniform attenuation model.

4.1.1 Data preprocessing

For a hybrid data set, dimensions can be divided into three categories, which are numerical, categorical, and ordinal. The numerical dimension reflects the physical measurement; the categorical dimension reflects the object semantics; while the ordinal dimension reflects the data order based on certain rules. If each data point contains d dimensions, it can be described using a d-dimensional vector, which contains p numerical attributes, q categorical

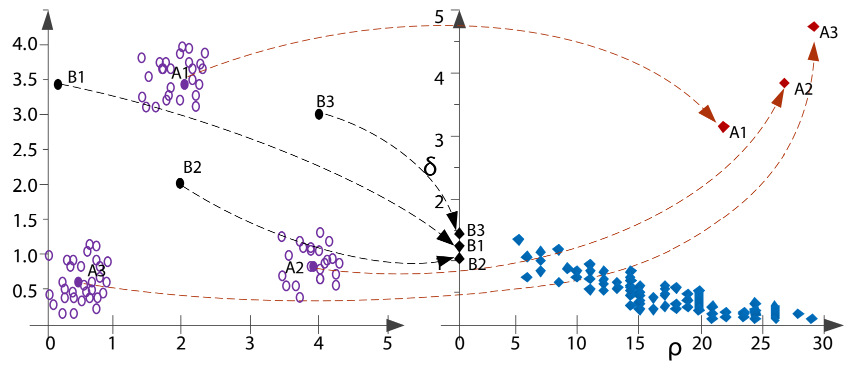


Fig. 2 The $\rho - \delta$ decision graph

attributes, and r ordinal attributes. Obviously, we have $p + q + r = d$. The based on the numerical and ordinal dimension information, we can derive the minimal grid division length and the total number of grids for the corresponding dimension.

4.1.2 Create initial density grids

After data preprocessing, the original d-dimensional space is divided into a set of sub-spaces with grid units. To create the initial density grids, we randomly select a small portion of data points from D and create a new set M. Then we use set M as the sampling data stream and feed it into the grids with fixed speed. The density of the girds in updated in real-time. In the end, grids with non-zero density are designated as the initial grids and recorded in the initial grid list.

4.1.3 Non-uniform attenuation model

In this section, we introduce our non-uniform attenuation model. First, assuming at time t_p, there are n grids for incoming data points and a total of N non-zero density grids in the list. Then we define the dispersion degree as follows.

$$\omega = \frac{n}{N} \tag{7}$$

where ω represents the current inflow data concentration. A smaller ω value means a higher inflow data concentration. Depending on the ω value, we assign the following piecewise attenuation parameters.

$$\lambda_{t_p} = \begin{cases} \lambda_{high} & \omega \geq 0.7 \\ \lambda_{med} & 0.3 < \omega < 0.7 \\ \lambda_{low} & \omega \leq 0.3 \end{cases} \tag{8}$$

where λ is the parameters for non-uniform attenuation, where the attenuation coefficients vary with the dispersion degree of the inflow data.

Thus, for any grid den, assuming the current update time is t_p and the previous one is t_l, we update the grid density using the following equation.

$$D(den, t_p) = D_{high}(den, t_p) + D_{med}(den, t_p) + D_{low}(den, t_p) \tag{9}$$

Taking $D_{high}(den, t_p)$ as an example, the density update method is described in the following equation.

$$D_{high}(den, t_p) = \begin{cases} 2^{-\lambda_{high}} D_{high}(den, t_l) + v & \lambda_t = \lambda_{high} \\ 2^{-\lambda_{high}} D_{high}(den, t_l) & else \end{cases} \tag{10}$$

where v is the grid data inflow speed and we have $D_{high}(den, 0) = 0$. $D_{med}(den, t_p)$ and $D_{low}(den, t_p)$ are updated similarly. Thus, at any given time t, we can calculate the average grid density as follows.

$$D_{ave}(t) = \frac{\sum_{i \in G}(D_{high}(i, t) + D_{med}(i, t) + D_{low}(i, t))}{N} \tag{11}$$

where G is the set of all grids and N is the total number of non-zero density grids.

Therefore, for any given grid den, its grid density can be classified using the following equation.

$$density = \begin{cases} S_m & D(den, t) \geq \mu D_{ave} \\ S_p & else \end{cases} \tag{12}$$

where S_m represents the dense grids and S_p represents the sparse grids. μ is a manually selected parameter and is typically greater than 1.

Therefore, for each grid, its information can be stored in a tuple, which is defined as follows.

$$T = (t_b, t_p, D, D_v, status, position) \tag{13}$$

where t_b is the initial time that data points arrived, t_p is the last time grid density is updated, D is the sum of all densities, D_v is a three dimensional vector to hold the information of $D_{high}(t_p)$, $D_{med}(t_p)$, and $D_{low}(t_p)$, $status$ indicates the dense or sparse states of the grid, and $position$ stores the location of the grid in each dimension. Tuple is the basic processing elements, which is later used in the online and offline clustering stages.

4.2 Online clustering stage

4.2.1 Online data update

The densities of grids change over time. Without incoming data, a dense grid can decay to a sparse one, and vice versa. Thus, to maintain the online clustering performance, the grid densities should be checked and updated regularly. However, checking the densities too frequently can occupy significant calculation resources and hurt performance. Therefore, it is important to find the appropriate update interval. We determine the update time using the minimal transforming time from sparse to dense as shown below.

Lemma 1 *For any grid den, the minimal time it takes to transform it from sparse to dense is*

$$\Delta T = \frac{1}{\lambda_{high}} log_2\left(\frac{D_{bd}}{D_{bd}-1}\right) \tag{14}$$

where ΔT is the maximum attenuation coefficient. D_{bd} is equal to μD_{ave} and represents the sparse and dense threshold.

4.2.2 Grid deletion

Maintaining a large non-zero grid list increases the requirement of storage and calculation resources. Moreover, many grids contain only a few noise points and tend to have zero density over time. Obviously, these grids need to be removed. To address this problem, we develop a minimal density detection method as shown in the following equation.

$$\varsigma(t_c, t_0) = (1 - 2^{-\lambda}) D_{bd} \tag{15}$$

where t_0 is the grid initializing time, t_c is the current time, ς is the threshold function, and $\lambda = \{\lambda_{high}, \lambda_{med}, \lambda_{low}\}$ is the attenuation coefficient. The threshold curve varies according to λ. At any given time t, if the density of a sparse grid is less than $\varsigma(t, t_0)$, this grid is deleted.

4.3 Offline clustering stage

After user issues a clustering request, the online process sends the tuple information and perform offline clustering. Offline clustering treats all grids as virtual points with its density as weights. Firstly, a similarity calculation method based on frequency distribution and dimension correlation analysis is developed to process the hybrid data. Then we use the CCFD algorithm to derive the clustering results.

4.3.1 Similarity calculation method

First, we define the distance of any two points, denoted as x_i and x_j, as follows.

$$Dis(x_i, x_j) = d(x_i, x_j)_q + d(x_i, x_j)_p + d(x_i, x_j)_r \tag{16}$$

where $d(x_i, x_j)_q$ is the distance of the categorical attribute, $d(x_i, x_j)_p$ is the distance of the numerical attribute, and $d(x_i, x_j)_r$ is the distance of the ordinal attribute. To derive the distance of a specific attribute, the existing methods use the following frequency based equation.

$$d(x_{in}, x_{jn}) = \begin{cases} e^{p_{in}-1} & x_{in} = x_{jn} \\ 1 & else \end{cases} \tag{17}$$

where for any data points i, j, and classification dimension n, x_{in} and x_{jn} represent the data point weights and p_{in} represents the frequency of x_{in}. The similarity between i, j increases when p_{in} becomes larger.

However, (17) assumes that the data points of different dimensions are independent, which is not true. Therefore, for the categorical data, we define the distance between dimension n, k as $d_{nk}(x_{in}, x_{in})$ and calculate it using the following equation.

$$d_{nk}(x_{in}, x_{jn}) = max(p_n(S|x_{in}) + p_n(S^C|x_{jn})) \tag{18}$$

where for any data points i, j, and classification dimension n, k, x_{in} and x_{jn} are the data point weights, S is a subset of dimension n, S^C is the complement set of S, $p_n(S|x_{in})$ is the probability that a data point in n dimension with a value of x_{in} belongs to set S in k dimension, and $p_n(S^C|x_{jn})$ is the probability that a data point in n dimension with a value of x_{jn} belongs to set S^C in k dimension.

Therefore, we can derive the distance calculation method considering both frequency and dimension correlations as follows.

$$d(x_{in}, x_{jn}) = \begin{cases} e^{p_{in}-1} & x_{in} = x_{jn} \\ \frac{\sum_{k=1,k\neq n}^{d} d_{nk}(x_{in}, x_{jn})}{d-1} & else \end{cases} \tag{19}$$

and the categorical distance can be calculates using

$$d(x_i, x_j)_q = \sum_{n=1+p}^{p+q} d(x_{in}, x_{jn}). \tag{20}$$

For numerical and ordinal dimensions, the distance can be calculated using the following equation.

$$d(x_{in}, x_{jn}) = |x_{in} - x_{jn}| e^{\chi(\sum_{t=x_{in}}^{x_{jn}} P_t - 1)}, \tag{21}$$

where χ is the constraint parameter and P_t is the probability that t is in $[x_{in}, x_{jn}]$. Therefore, we can derive the numerical distance as follows.

$$d(x_i, x_j) = \sqrt{\sum_{n=1}^{p}\left(\frac{d(x_{in}, x_{jn})}{X_n}\right)}, \tag{22}$$

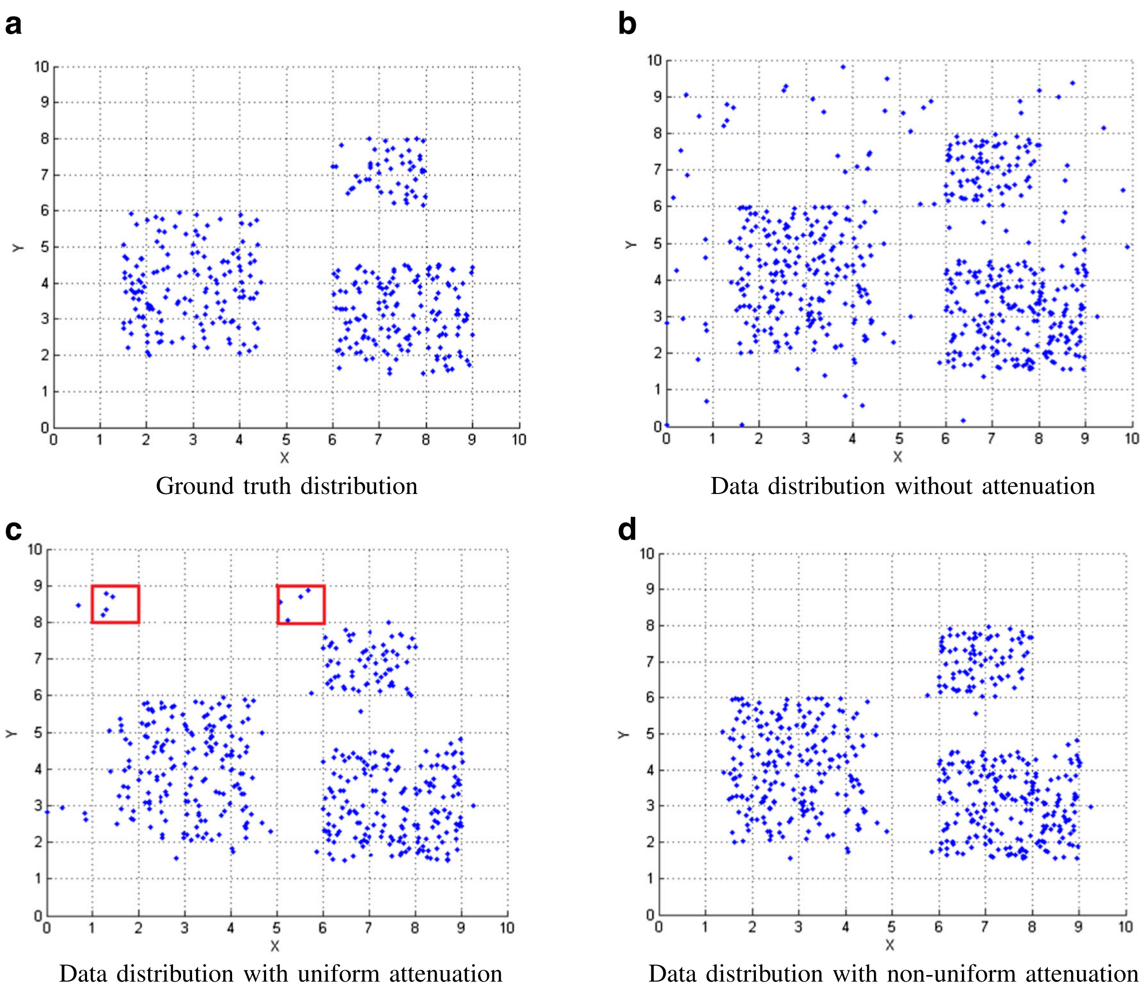


Fig. 3 Comparison of uniform and non-uniform attenuation models

where $X_n = (x_n)_{max} - (x_n)_{min}$. Similarly, we can derive $d(x_i, x_j)_r$, which is the distance for the ordinal dimension.

4.3.2 The CCFD algorithm

The performance of the RLM algorithm, as introduced in Section 3.3, is highly sensitive to the density radius and clustering center number, which, however, cannot be determined automatically. To address those problems, we propose the CCFD algorithm, which employs the normal distribution technique and improved hill climbing algorithm to automatically determine the clustering centers and dynamically adapt the density radius.

First, we defined coefficient γ as follows.

$$\gamma_i = \rho_i \times \delta_i. \tag{23}$$

It is discovered that γ follows Gaussian distribution with mean μ and standard deviation σ. We set the clustering threshold to be 5σ. In other words, any singular point i with γ_i greater than 5σ is considered as the clustering center candidate.

To dynamically adjust the density radius d_c, we first define the following fitness functions.

$$F1 = \frac{\sum_{j=1}^{m}[\sum_{x_i \in C_j} d(x_i, C_j)/|C_j|]}{m}, \tag{24}$$

$$F2 = \frac{\sum_{j=1}^{m}[\sum_{i=1, i \neq j}^{m} d(C_i, C_j)/(m-1)]}{m}, \tag{25}$$

where for clusters i and j, m is the total number of clusters, C_i and C_j are the corresponding cluster centers, and $|C_j|$ is the number of points. In the equations, $F1$ represents the global average intra-cluster distance and $F2$ represents the global average inter-cluster distance. Therefore, the fitness can be defined as

$$Fit = \frac{F2}{F1}. \tag{26}$$

For any given d_c, greater Fit represents better clustering results.

Algorithm 1 shows the flow of our proposed improved climbing algorithm. It sets 10% of the maximal distance as the starting point and extends to the two sides with a iteration limit r. The clustering accuracy is determined based on the Fit value. The iteration limit gradually reduces to zero, where the program quits. Our improved climbing algorithm can automatically find a near optimal radius d_c within limited time.

Algorithm 1 The improved climbing algorithm

```
Input: $S$ // Similarity distance matrix
Input: $P_o$ // The initial position
Input: $r$ // The iteration limit
Output: $d_c$
  set $P_{tbest} = P_o$
  while $r > 0$ do
      calculate the Fitness value of $Fit(P_{tbest})$, $Fit(P_{tbest} + r)$, and $Fit(P_{tbest} - r)$
      $P_{tbest} \leftarrow \arg\max_P Fit(P)$
      $r = r - 0.5$
  end while
  $P_{best} = P_{tbest}$
  calculate the corresponding $d_c$
```

5 Experimental results

5.1 Experiment setup

The experiments are performed on a Radeon R5 Graphics machine with 4GB memory and Matlab2013a installed. To compare the performance of various algorithms, we use the following criteria.

1) Clustering accuracy. It is defined as

$$r = \frac{\sum_{i=1}^{k} \alpha_i}{N}, \tag{27}$$

where α_i is the number of correctly classified samples, k is the number of clusters, N is the total number of samples. r is in range [0, 1] and a higher r represents better performance.

2) Average Purity. It is defined as

$$Pur = \frac{1}{k} \sum_{i=1}^{k} \frac{|C_i^d|}{|C_i|}, \tag{28}$$

where k is the number of clusters, $|C_i^d|$ is the number of data points in d, which is the most important class label in cluster i, $|C_i|$ is the total number of data points in cluster i. Pur is in range [0, 1] and a higher Pur represents better clustering quality.

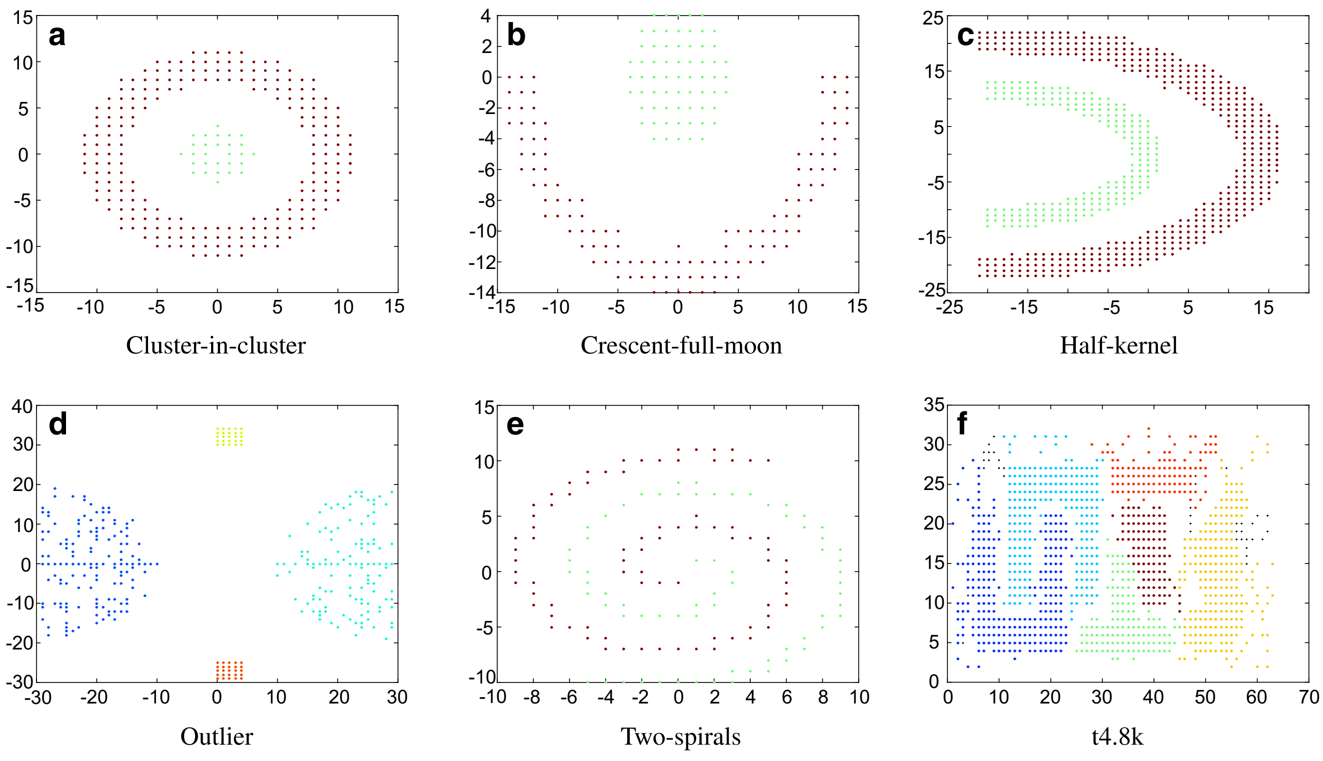


Fig. 4 CCFD clustering results on simulated data

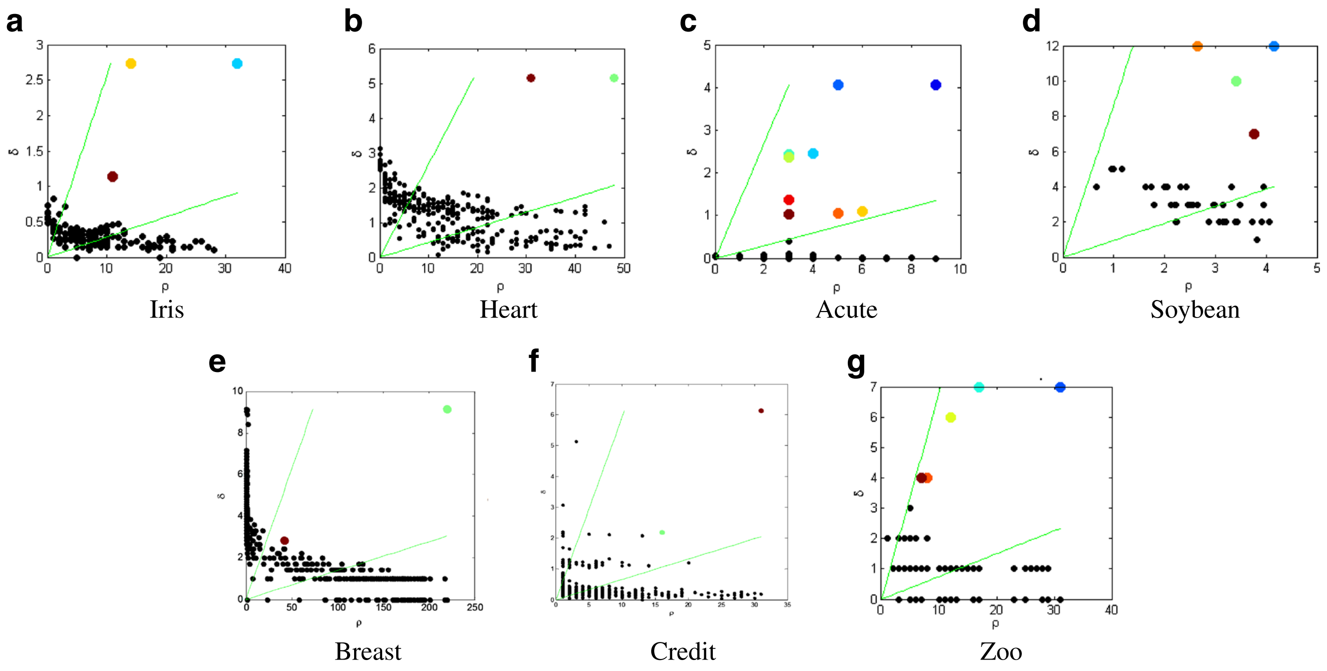


Fig. 5 The ρ-σ graph for real-world datasets

3) Sum of squared distance (SSQ). It is defined as

$$SSQ(\Delta) = \sum d^2(x_i, C_{x_i}), \tag{29}$$

where $d^2(x_i, C_{x_i})$ is the square distances from data points to the corresponding cluster center. A smaller SSQ represents better clustering result.

5.2 Non-uniform attenuation model

First we evaluate the performance of our Non-uniform attenuation model. Figure 3 shows the comparing clustering results of various models. Figure 3a shows the ground truth input testing data. Figure 3b is the data distribution after the stream is processed by the grid based online clustering

Table 1 The clustering results in CCFD and other classical algorithms on some datasets

Algorithm	Iris		Breast		Soybean		Zoo		Acute		Heart		Credit	
	r	*Pur*	*r*	*Pur*	*r*	*Pur*	*r*	*Pur*	*r*	*Pur*	*r*	*Pur*	*r*	*Pur*
K-prototypes	0.819	0.842	0.840	0.862	0.856	0.877	0.806	0.830	0.610	0.720	0.557	0.644	0.562	0.624
SBAC	0.426	0.460	0.450	0.530	0.617	0.642	0.426	0.485	0.508	0.556	0.752	0.778	0.555	0.627
KL-FCM-GM	0.335	0.382	0.673	0.724	0.903	0.895	0.864	0.843	0.682	0.749	0.758	0.802	0.574	0.632
EKP	0.432	0.450	0.530	0.610	0.831	0.780	0.629	0.694	0.508	0.586	0.545	0.589	0.682	0.749
WFK-prototypes	0.871	0.820	0.761	0.780	0.810	0.820	0.908	0.876	0.710	0.765	0.835	0.826	0.838	0.826
IWKM	0.822	0.840	0.882	0.893	0.908	0.922	0.510	0.560	0.620	0.540	0.550	0.620	0.779	0.806
SpectralCAT	0.960	**0.980**	0.921	0.939	0.894	0.895	0.881	0.910	0.867	0.824	0.820	0.824	0.770	0.794
k-modes	0.840	0.831	0.840	0.863	0.915	0.900	0.878	0.867	0.780	0.767	0.783	0.840	**0.870**	0.723
RLM	0.960	0.964	0.923	0.932	1.000	1.000	**0.921**	**0.880**	0.900	0.918	0.832	**0.842**	0.853	**0.869**
DC-MDACC	0.960	0.964	0.938	0.945	0.957	0.985	0.892	0.849	0.917	0.918	**0.848**	0.833	0.796	0.833
ACC-FSFDP	0.960	0.958	0.938	**0.947**	0.957	0.985	0.874	0.862	0.900	0.921	0.832	**0.842**	0.784	0.814
CCFD	**0.968**	0.960	**0.948**	0.935	**1.000**	**1.000**	**0.921**	**0.880**	**0.921**	**0.980**	**0.848**	0.820	0.853	**0.869**

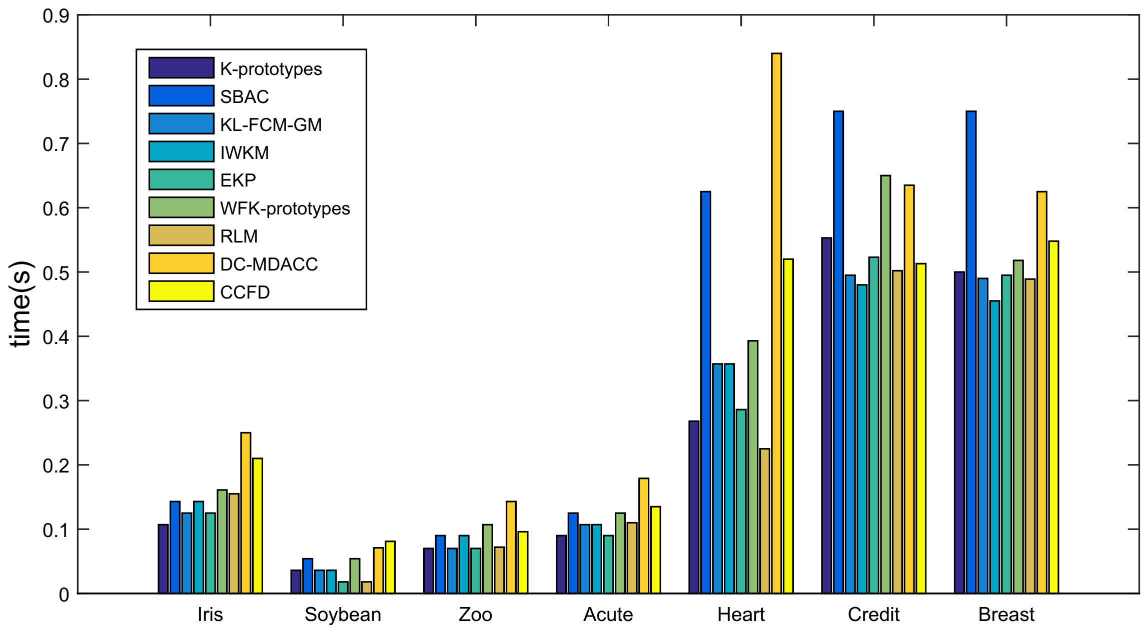


Fig. 6 Execution time comparison

algorithm without attenuation. Figure 3c shows the clustering results using uniform attenuation model. Figure 3d presents the clustering results using our non-uniform attenuation model. It is demonstrated that attenuation models can effectively remove noise points. Comparing Fig. 3c and ds, it shows that the non-uniform attenuation model can remove more residual noise point and generate compact cluster structure, i.e., clusters with higher in-cluster densities.

5.3 CCFD algorithm evaluation

5.3.1 Performance analysis

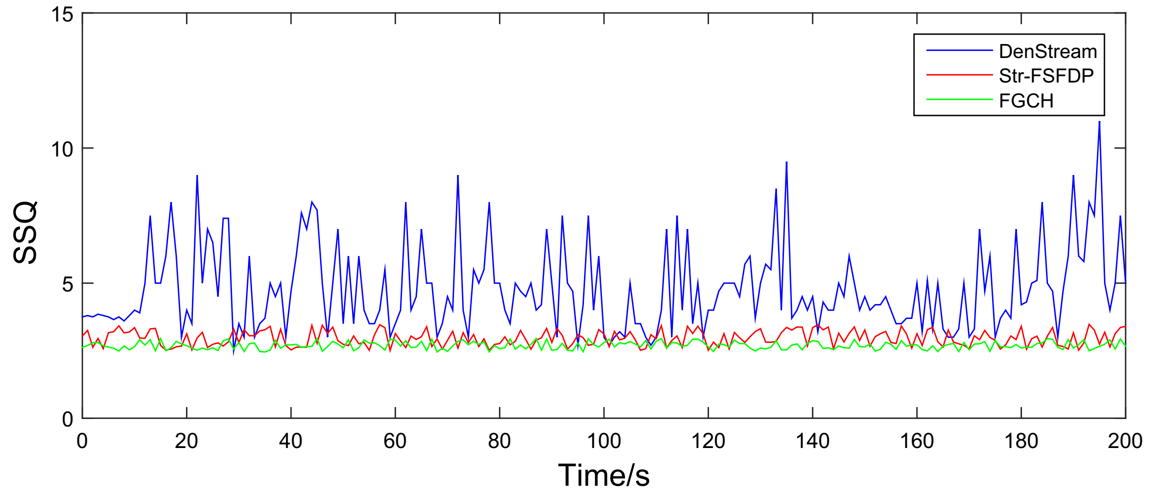
To evaluate the performance of the CCFD algorithm, we employ 6 simulated two-dimensional datasets, which are cluster-in-cluster, crescent-full-moon, half-kernel, outlier, two-spirals, and t4.8k [1], respectively. Figure 4 shows the corresponding clustering results. It is demonstrated that CCFD can cluster arbitrary shape datasets accurately. Furthermore, we evaluate our algorithm on real-world datasets and the results are shown in Fig. 5. The datasets are derived from the UCI database. Among the data sets, aggregation, spiral, jain, flame, and iris are numeric datasets; soybean, zoo, and acute are categorical datasets; and heart, breast, and Credit are hybrid datasets. Figure 5 shows the ρ-δ graph when d_c is optimal. The colored points represent clustering centers. In the experiment, our algorithm can always derive exactly the same number of clusters as the ground truth.

In this work, we compare the performance of following techniques.

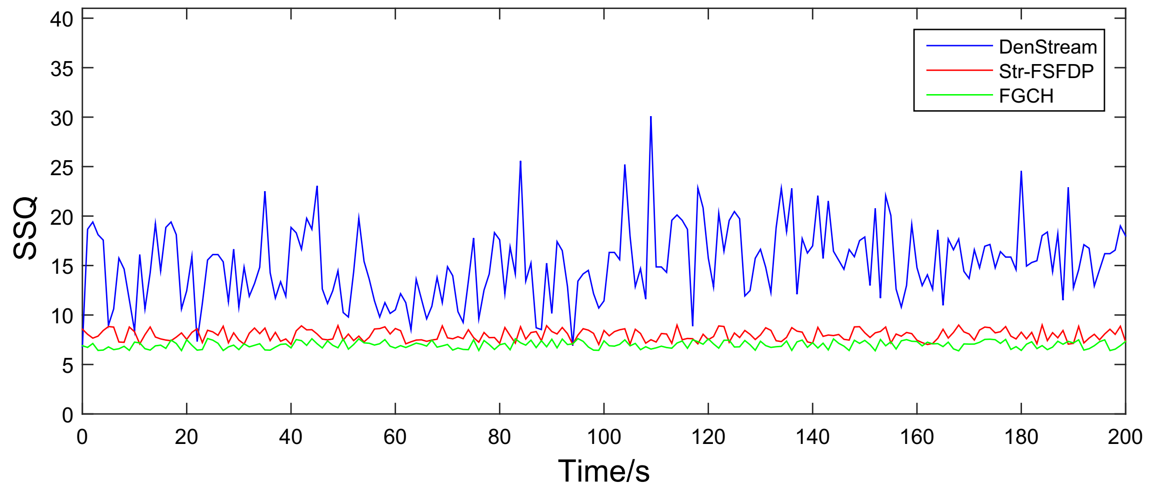
(1) K-prototypes [14]: this method combines k-means and k-mode algorithms, which use a cost function for hybrid datasets of distance calculation.
(2) SBAC [18]: a technique based on biometric classification, which assign greater weights to uncommon features.
(3) KL-FCM-GM [17]: a technique based on Gauss-Multinomial distribution model, which employ a fully probabilistic dissimilarity functional for handling hybrid data.
(4) EKP [41]: a novel unsupervised evolutionary clustering algorithm based on K-prototypes.
(5) WFK-prototypes [15]: this method combines the mean values of fuzzy centroid and extend K-prototypes with significance concepts.

Table 2 Experimental datasets information

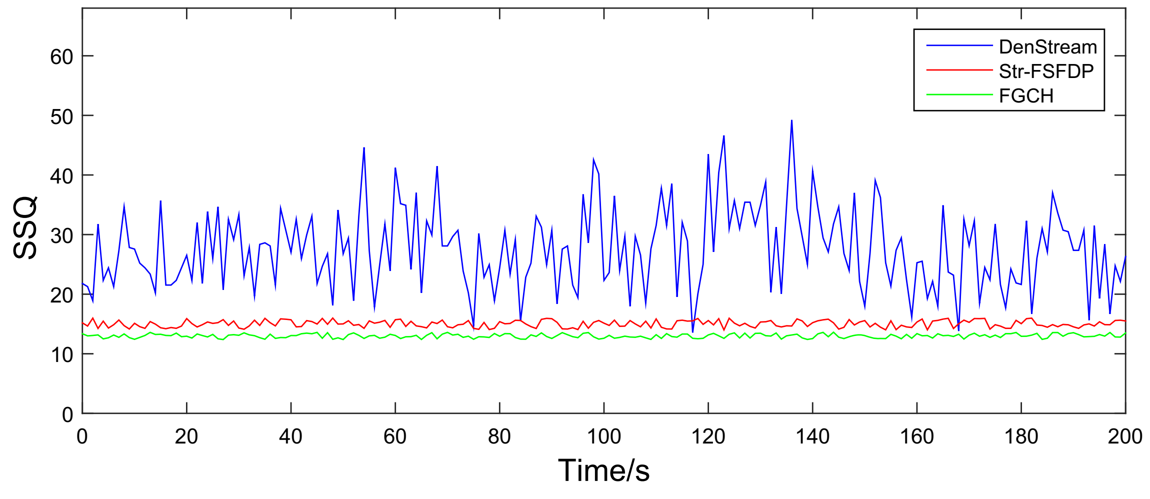
Data set	Object num	Dimension num	Cluster num
SIGMA_DATA	500000	3	4
EDS	100000	2	15
RBF5	300000	5	5
RBF10	300000	10	5
RBF15	300000	15	5
KDD CUP 99	494031	41	24
Forest CoverType	581012	54	7



RBF5



RBF10



RBF15

Fig. 7 Different SSQ of three algorithms on RBF datasets

Table 3 Clustering accuracy and purity of RBF datasets on three algorithms

Data sets	DenStream		Str-FSFDP		FGCH	
	Acc	*Pur*	*Acc*	*Pur*	*Acc*	*Pur*
RBF5	0.91	0.89	0.98	0.97	**1.00**	**1.00**
RBF10	0.90	0.88	0.96	0.97	**1.00**	**1.00**
RBF15	0.86	0.86	0.95	0.95	**1.00**	**1.00**

(6) IWKM [42]: it combines both mean with distribution centroid to represent the prototype of the cluster, which takes into account the significance of different attributes towards the clustering process.
(7) SpectralCAT [43]: this algorithm automatically transform the high-dimensional input data into categorical values for hybrid data clustering.
(8) k-modes [44]: a technique extend the k-means paradigm to categorical domains.
(9) RLM [24]: it is the state-of-art clustering center determination algorithm, which based on the idea that cluster centers are characterized by a higher density than their neighbors and by a relatively large distance from points with higher densities.
(10) DC-MDACC [45]: a technique which use dominant analysis to calculate dissimilarity for different types of hybrid data.
(11) ACC-FSFDP [46]: a algorithm based on RLM which use linear regression model and residuals analysis to find cluster centers automatically.

In other to more clearly show the advantages of different algorithms, we bolded the optimal results obtained by different algorithms under each data set in Table 1. And the same operation is performed in Tables 3 and 4.

As shown in Table 1, CCFD has the best performance on Iris. On soybean, zoo, and acute datasets, CCFD outperforms the state-of-art algorithm in both accuracy and purity by 1-6%. It is demonstrated that our algorithm can generate good clustering results for both numeric and categorical datasets. On heart, breast, and credit datasets, our algorithm achieves best accuracy on two and best purity on one. Our CCFD algorithm achieves an excellent balance between clustering accuracy and purity. In general, there are two major advantages of our technique.

1) CCFD calculates the distance matrix based on frequency distribution and conditional probability analysis for hybrid data. It considers the relevance of various dimensions and can effectively process hybrid datasets.
2) CCFD can automatically determine the number of clustering centers. Moreover, it can effectively process non-spherical clusters and is more resilient to noise points.

5.3.2 Time analysis

The algorithm running time mainly consists of four parts, i.e., the number of iterations, calculating ρ and σ, locating the clustering centers, and calculating Fitness. According to Algorithm 1, the total number of iteration is $r + 1$. The time required for ρ and σ calculation is $O(n^2)$. The time complexity of the clustering center location algorithm is $O(n)$. For the Fitness calculation, the running time is $O(mn)$, where m is the number of clustering centers and is typically much smaller than n. Therefore, the running time complexity of the CCFD algorithm is $O(rn^2)$.

Figure 6 shows the execution time comparison of different algorithms. As expected, the running time of CCFD is longer than the linear time complexity algorithms such as K-means and EKP. However, it takes less running time than other state-of-art algorithms such as SBAC and DC-MDACC. Overall, the algorithm running time is adequate for data stream clustering applications.

5.4 FGCH algorithm evaluation

To validate our technique, we evaluate the algorithm performances on 5 simulated datasets and 1 real-world dataset. Table 2 shows the properties of the datasets.

In this work, we compare the performance of following techniques.

(1) DenStream [28]: this method extends the DBSCAN algorithm in offline stage, and use potential cluster to improve the processing power of outliers.

Table 4 Clustering accuracy and purity of SIGMA_DATA and EDS on several algorithms

Data sets	Speed(v)	FGCH		Str-FSFDP		StrDenAP		StrAP		DenStream		HCluStream	
		Acc	*Pur*	*Acc*	*Pur*	*Acc*	*Pur*	*Acc*	*Pur*	*Acc*	*Pur*	*Acc*	*Pur*
SIGMA_DATA	*1000*	**95.32**	**94.80**	92.36	91.60	93.56	92.78	83.06	82.30	79.63	76.50	81.03	79.23
	500	**96.58**	**95.12**	91.23	91.10	93.80	91.65	84.52	81.07	81.25	80.25	82.06	80.36
EDS	*1000*	**97.13**	**97.63**	94.33	96.48	89.32	91.49	85.43	87.67	84.30	84.60	82.57	84.42
	500	**99.13**	**98.82**	96.58	97.32	89.90	92.33	86.32	88.12	82.64	84.98	85.32	87.06

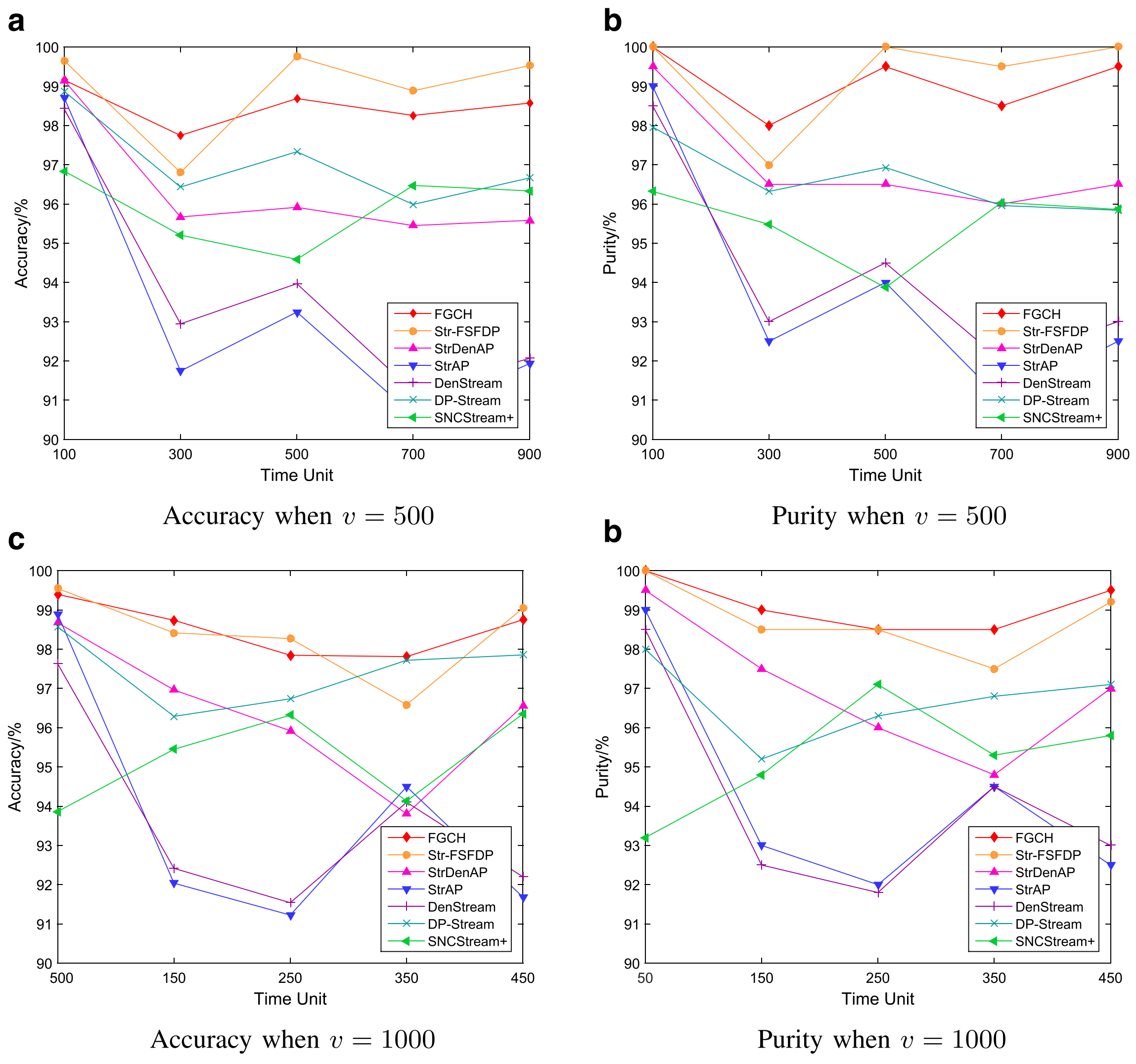


Fig. 8 Accuracy and purity of clustering algorithms in KDD CUP 99 dataset

(2) Str-FSFDP: [46]: a algorithm which use micro clusters decay function and deletion mechanism of micro clusters for hybrid data stream.
(3) StrAP [47]: a algorithm which extending Affinity Propagation (AP) to data steaming.
(4) StrDenAP [48]: this method based on StrAP, which adopts micro cluster decay density and micro cluster deletion mechanism to guarantee better clustering results.
(5) ExCC [34]: it is an exclusive and complete clustering algorithm for heterogeneous data stream which use dynamic thresholds, wait and watch policy are adopted to detect noises.
(6) DP-Stream [37]: it employ the density peaks based clustering algorithm to construct a fat node leading tree for data stream clustering.
(7) SNCStream+ [36]: a technique which uses network and evolution techniques to process the data stream.

5.4.1 Simulated datasets comparison

In Table 2, SIGMA_DATA, EDS, RBF5, RBF10, and RBF15 are simulated data sets. SIGMA_DATA is a three-dimensional dataset, which contains 500,000 data points. It has 4 categories, which include three striped clusters and one annular cluster. EDS is a two-dimensional dataset with 100,000 data points. It has 15 categories with varying cluster shapes. Some of its clusters overlap with each other spatially. RBF5, RBF10, and RBF15 are generated by WEKA. They represent high-dimensional data with 5-dimensional categorical attributes, 10-dimensional numeric attributes, and 15-dimensional ordinal attributes. Each dataset contains 300,000 data points. During experiment, we first select 10% of the dataset to initialize the grids. We use two different flow speed v to process the data.

Figure 7 shows the SSQ comparison results of our technique and two state-of-art clustering algorithms. Compared

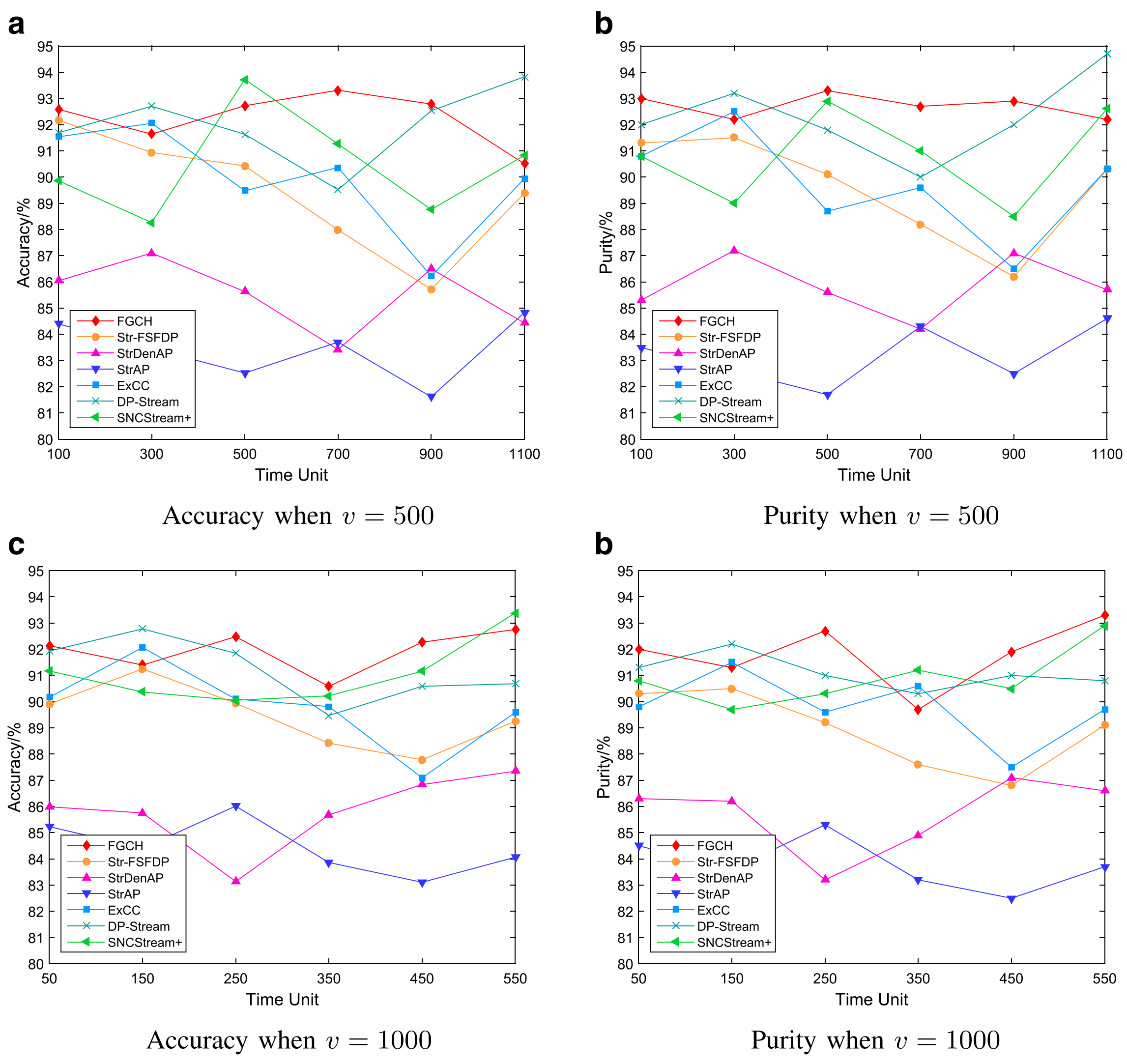


Fig. 9 Accuracy and purity of clustering algorithms in Forest CoverType dataset

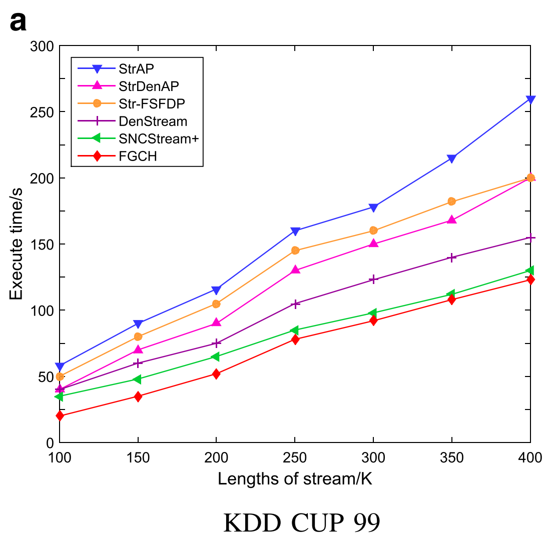
with Denstream and Str-FSFDP, the average SSQ obtained by our technique is reduced about 50% and 10% respectively. On the other hand, the SSQ values of DenStream are more volatile. This implies that the clustering effect of DenStream is greatly affected by offline request timestamps. Meanwhile, our technique can judge clusters number correctly and obtain stable and efficient results at any timestamp. In addition, compared with Str-FSFDP, our technique is more resilient to noise points.

Table 3 shows the average accuracy and purity in RBF datasets for these three algorithms. The experiment results show that our technique can derive the optimal results, for both accuracy and purity. Thus, we do not include more algorithms during the comparison. The reason for the performance of our algorithm can be mostly attributed to the more accurate similarity calculation method. Table 4 shows the average clustering accuracy and purity results for the SIGMA_DATA and EDS datasets. We have implemented 5 algorithms from existing literate and compare them with our technique. As a result, in SIGMA_DATA, our FGCH algorithm outperforms the state of the art in both accuracy and purity by 3-12%. In EDS, FGCH outperforms the state-of-art algorithm in both accuracy and purity by 2-10%. Meanwhile, the accuracy and purity of FGCH keep stable at different speed, which shows that our algorithm is robust at online stage.

5.4.2 Real-world datasets comparison

To evaluate our technique on real-world hybrid datasets, we implement the algorithms on the widely used KDD CUP 99 dataset and Forest CoverType dataset. The following paragraphs briefly describe these datasets.

KDD CUP 99 is used in the third international knowledge discovery and data mining tools competition. It contains 42 dimensions, including 34 numerical ones, 7 categorical ones



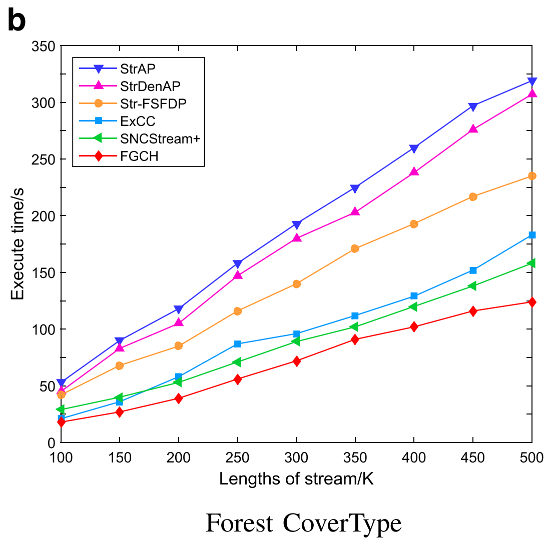


Fig. 10 The running time of various algorithms on KDD CUP 99 and Forest CoverType

and 1 labeled dimension.The experiment takes 10% of the sample data for testing, which includes 494031 records. as shown in Table 2. In addition, this dataset is known that its distribution changes over time, representing previously unknown types of cyber-attacks.

Forest CoverType is determined from US Forest Service Region 2 Resource Information System, which contains 581012 records. Each record contains 54 dimensions, including 10 numerical dimensions and 44 classification dimensions, a total of 7 categories. Numerical attributes mainly describe the numerical information such as elevation, slope and the distance from surface water. The first 4 classification dimensions indicate the geographical features of the area, and the last 40 classification dimensions indicate soil information of the forest environment.

The clustering performance is evaluated using the average accuracy and purity of clusters. Figures 8 and 9 shows the comparing results of these two datasets. In KDD CUP 99, the clustering accuracy and purity of our FCGH algorithm maintains top two and above 98%. The closest technique is Str-FSFDP. In Forest CoverType, FGCH also can also achieve the best accuracy and purity stably, which are around 92%. The closest technique are DP-Stream and SNCStream+. However, as shown in Fig. 10, Str-FSFDP, DP-Stream and SNCStream+ require significantly larger running time, which is important for online data stream clustering. Therefore, the overall performance of the FGCH algorithm surpasses the comparing state-of-art algorithms.

The performance of the FGCH algorithm can be explained by the following observations.

1) FGCH implements a non-uniform attenuation model, which can slow the decaying rate of the core data points. The clusters generated by the non-uniform attenuation model typically have more compact structures.
2) The dense grids containing noise points or outdated elements are deleted online, which can improves the clustering efficiency and quality.
3) FGCH utilizes the CCFD algorithm to automatically determine the clustering center, which improves the parameter sensitivity.

The running time is an important criterion for online algorithms. Thus, we compare the running time of our algorithm and other algorithms in two real datasets. Figure 10 shows the comparison results. In general, the running time increases linearly with the data volume. In KDD CUP 99, the running time of FGCH is only about half of the Str-FSFDP algorithm, which also has excellent performance. In Forest CoverType, compared with SNCStream+ and ExCC, FGCH also runs faster slightly. The reason is that our algorithm uses the grid mapping method and deletes the outdated and slowly growing grids online.

6 Conclusion

In this work, we present FGCH, a fast data stream clustering algorithm based on grid. Our algorithm first utilizes a novel non-uniform attenuation model to initialize the grids. During online clustering stage, the grid tuple information is updated according to the corresponding attenuation coefficients. During offline clustering stage, the distance matrix between data points is calculated using the proposed similarity calculation technique based on frequency and inter-dimensional correlations. Finally, the CCFD algorithm is used to automatically determine the cluster centers and radius. As a result, our technique can achieve good clustering accuracy and purity with better running time requirements compared with the second best algorithm.

References

1. Karypis G, Han EH, Kumar V (1999) Chameleon: hierarchical clustering using dynamic modeling. IEEE Computer Society Press
2. Wu Z, Xu Q, Li J, Fu C, Qi X, Xiang Y (2018) Passive indoor localization based on CSI and naive Bayes classification. IEEE Trans Syst Man Cybern Syst 48(9):1566–1577
3. Silva JA, Faria ER, Barros RC, Hruschka ER (2013) Data stream clustering: a survey. Acm Comput Surv 46(1):13
4. Chen JY, He HH (2016) A fast density-based data stream clustering algorithm with cluster centers self-determined for mixed data. Elsevier Science Inc.
5. Goodall DW (1966) A new similarity index based on probability. Biometrics 22(4):882–907
6. Fu C, Zhao M, Lu F, Chen X, Chen J, Wu Z, Xia Y, Xuan Q (2018) Link weight prediction using supervised learning methods and its application to yelp layered network. IEEE Trans Knowl Data Eng 30(8):1507–1518
7. Qi X, Fang B, Yi L, Wang J, Zhang J, Zheng Y, Bao G (2018) Automatic pearl classification machine based on a multistream convolutional neural network. IEEE Trans Ind Electron 65(8):6538–6547
8. Dawar S, Sharma V, Goyal V (2017) Mining top-k high-utility itemsets from a data stream under sliding window model. Appl Intell 47(4):1240–1255
9. Xuan Q, Zhang ZY, Fu C, Hu HX, Filkov V (2018) Social synchrony on complex networks. IEEE Trans Cybern 48(5):1420–1431
10. Hassanien AE, Azar AT, Snasael V, Kacprzyk J, Abawajy JH (2015) Big data in complex systems. Springer International Publishing
11. Xiang Y, Tang Y, Zhu W (2016) Mobile sensor network noise reduction and re-calibration using Bayesian network. Atmosp. Measur. Techn. 9(9):347–357
12. Chen JY, He HH (2016) A fast density-based data stream clustering algorithm with cluster centers self-determined for mixed data. Elsevier Science Inc.
13. Wang S, Fan Y, Zhang C, Xu H, Hao X, Hu Y (2008) Entropy based clustering of data streams with mixed numeric and categorical values
14. Huang Z (1997) Clustering large data sets with mixed numeric and categorical values, pp 21–34
15. Ji J, Pang W, Zhou C, Han X, Wang Z (2013) Corrigendum: corrigendum to ’a fuzzy k-prototype clustering algorithm for mixed numeric and categorical data’ [knowledge-based systems, 30 (2012) 129-135]. Neurocomputing 120(10):590–596
16. Gath I, Geva A (1989) Unsupervised optimal fuzzy clustering. IEEE Trans Pattern Anal Machine Intell 11(7):773–780
17. Chatzis SP (2011) A fuzzy c-means-type algorithm for clustering of data with mixed numeric and categorical attributes employing a probabilistic dissimilarity functional. Pergamon Press, Inc.
18. Li C, Biswas G (2002) Unsupervised learning with mixed numeric and nominal data. IEEE Trans Knowl Data Eng 14(4):673–690
19. Hsu CC, Yu CC (2007) Mining of mixed data with application to catalog marketing. Expert Syst Appl 32(1):12–23
20. Ryu TW, Eick F (1998) Similarity measures for multi-valued attributes for database clustering. In: Conf on smart engineering system design: neural networks, fuzzy logic, evolutionary programming, data mining & rough sets, pp 1–4
21. Chavent M, De Carvalho F, Lechevallier Y, Verde R (2006) New clustering methods for interval data. Comput Stat 21(2):211–229
22. Chen M, Li L, Bo W, Cheng J, Pan L, Chen X (2016) Effectively clustering by finding density backbone based-on k nn. Pattern Recogn 60:486–498
23. De Andrade Silva J, Hruschka ER, Gama J (2017) An evolutionary algorithm for clustering data streams with a variable number of clusters. Expert Syst Appl 67:228–238
24. Rodriguez A, Laio A (2014) Machine learning. Clustering by fast search and find of density peaks. Science 344(6191):1492
25. Liadan O, Meyerson A, Motwani R, Mishra N, Guha S (2002) Streaming-data algorithms for high-quality clustering. In: International conference on data engineering, 2002. Proceedings, pp 685–694
26. Aggarwal CC, Yu PS, Han J, Wang J (2003) A framework for clustering evolving data streams. In: Proceedings 2003 VLDB conference, pp 81–92. Elsevier
27. Aggarwal CC, Han J, Wang J, Philip S (2004) A framework for projected clustering of high dimensional data streams. In: Thirtieth International conference on very large data bases, pp 852–863
28. Cao F, Ester M, Qian W, Zhou A (2006) Density-based clustering over an evolving data stream with noise. In: Siam International conference on data mining, April 20-22, 2006, Bethesda, MD, USA, pp 328–339
29. Kremer H, Kranen P, Jansen T, Seidl T, Bifet A, Holmes G, Pfahringer B (2011) An effective evaluation measure for clustering on evolving data streams. In: ACM SIGKDD International conference on knowledge discovery and data mining, pp 868–876
30. Hyde R, Angelov P, Mackenzie AR (2017) Fully online clustering of evolving data streams into arbitrarily shaped clusters. Inf Sci 382C383:96–114
31. Bodyanskiy YV, Tyshchenko OK, Kopaliani DS (2017) An evolving connectionist system for data stream fuzzy clustering and its online learning. Neurocomputing
32. Blei D (2006) Variational inference for Dirichlet process mixtures. J Bayesian Anal 1(1):121–143
33. Huynh V, Phung D (2017) Streaming clustering with Bayesian nonparametric models. Neurocomputing, 258
34. Bhatnagar V, Kaur S, Chakravarthy S (2014) Clustering data streams using grid-based synopsis. Knowl Inf Syst 41(1):127–152
35. Gomes HM, Gomes HM (2015) Sncstream: a social network-based data stream clustering algorithm. In: ACM Symposium on applied computing, pp 935–940
36. Barddal JP, Gomes HM, Enembreck F, Barthს JP (2016) Sncstream +: extending a high quality true anytime data stream clustering algorithm. Inf Syst 62:60–73
37. Xu J, Wang G, Li T, Deng W, Gou G (2016) Fat node leading tree for data stream clustering with density peaks. Knowl-Based Syst 120:99–117
38. Han D, Giraud-Carrier C, Li S (2015) Efficient mining of high-speed uncertain data streams. Kluwer Academic Publishers
39. Sang CY, Di HS (2014) Co-clustering over multiple dynamic data streams based on non-negative matrix factorization. Appl Intell 41(2):487–502
40. Yi W, Li T (2018) Improving semi-supervised co-forest algorithm in evolving data streams. Appl Intell 4:1–15
41. Zheng Z, Gong M, Ma J, Jiao L (2010) Unsupervised evolutionary clustering algorithm for mixed type data. In: Evolutionary computation, pp 1–8
42. Ji J, Bai T, Zhou C, Ma C, Wang Z (2013) An improved k-prototypes clustering algorithm for mixed numeric and categorical data. Neurocomputing 120:590–596

43. David G, Averbuch A (2012) Spectralcat: categorical spectral clustering of numerical and nominal data. Pattern Recogn 45(1):416–433
44. Huang Z (1997) A fast clustering algorithm to cluster very large categorical data sets in data mining. Research Issues Data Mining Knowl Discov, 1–8
45. Chen JY, He HH (2015) Research on density-based clustering algorithm for mixed data with determine cluster centers automatically. Acta Automatica Sinica
46. Chen JY, He HH (2016) A fast density-based data stream clustering algorithm with cluster centers self-determined for mixed data. Inform Sci 345(C):271–293
47. Zhang X, Furtlehner C, Sebag M (2008) Data streaming with affinity propagation. Lect Notes Comput Sci 5212:628–643
48. Zhang JP, Chen FC, Li SM, Liu LX (2011) Data stream clustering algorithm based on density and affinity propagation techniques. Zidonghua Xuebao/acta Automatica Sinica 40(2):277–288

Jinyin Chen received BS and PhD degrees from Zhejiang University of Technology, Hangzhou, China, in 2004 and 2009, respectively. She studied evolutionary computing in Ashikaga Institute of Technology, Japan in 2005 and 2006. She is currently an associate professor in college of information engineering, Zhejiang University of Technology. Her research interests include evolutionary computing, data mining and deep learning algorithm.

Xiang Lin is a Master student at the college of Information engineering, Zhejiang University of Technology. His research interests include data mining and applications, and artificial intelligence.

Qi Xuan received the BS and PhD degrees in control theory and engineering from Zhejiang University, Hangzhou, China, in 2003 and 2008, respectively. He was a Post-Doctoral Researcher with the Department of Information Science and Electronic Engineering, Zhejiang University, from 2008 to 2010, and a Research Assistant with the Department of Electronic Engineering, City University of Hong Kong, Hong Kong, China, in 2010 and 2017. From 2012 to 2014, he was a Postdoctoral Fellow with the Department of Computer Science, University of California at Davis, Davis, CA, USA. He is currently a Professor with the College of Information Engineering, Zhejiang University of Technology, Hangzhou. His current research interests include network-based algorithm design, social network data mining, social synchronization and consensus, reactiondiffusion network dynamics, machine learning and computer vision.

Yun Xiang received B.S in Information and Electrical Engineering from Zhejiang University in 2006, Ma. Sc in Electrical Engineering from University of Massachusetts in 2008, and Ph.D. in Electrical Engineering from University of Michigan in 2014, respectively. Currently, he is hired assistant professor of Zhejiang University of Technology with the College of Information Engineering. His research interests including wireless sensor network, data fusion, machine learning, and network algorithms.