Musa MİLLİ[a] and Hasan BULUT[b]*

[a] *Department of Computer Engineering, Turkish Naval Academy, National Defense University, Tuzla Istanbul, 34942, Turkey;*

[b] *Department. of Computer Engineering, Ege University, Bornova, Izmir, 35100, Turkey*

Corresponding author: Assoc.Prof. Dr. Hasan BULUT
Address:
Department of Computer Engineering
Ege University
Bornova, Izmir, 35100
Turkey

E-mail: hasan.bulut@ege.edu.tr
Mobile Phone: +90 506 603 9654
Work Phone: +90 232 311 2596
Fax Number: +90 232 339 9405

ReHEvo: Recent History Based Evolving Data Clustering Algorithm

Abstract—With the rise of web based applications and interactions, data generated on the internet has increased exponentially. Log files, user transactions, and sensor data generated daily correspond to huge amounts of data. Most of the web data come as streams that need to be processed in an online fashion. The main purpose of stream processing is to adapt to new incoming data without storing old ones and know the next ones. However, it is a hard task to adapt incoming data in a highly driftable environment. This paper highlights the need to develop an efficient real-time clustering algorithm for highly driftable stream data. For this propose, an algorithm called ReHEvo, which uses recent history to forget old data and adapt new data quickly, is developed. To achieve these goals, micro-clusters and macro-clusters keep recent data as an attribute. If the macro-cluster is large enough and is also old enough, it is deleted and a new one created using new data that recently joins the cluster. With this method, the algorithm can easily forget the older data and does not need to learn the pattern again. ReHEvo is a fully online algorithm that does not include an offline macro-clustering stage. In addition, in order to achieve better results, ReHEvo utilizes adaptive parameters. We demonstrate the potentials of ReHEvo in a set of experiments using both synthetic and real data streams and compare it to three state-of-art algorithms namely: CluStream, DenStream, and ClusTree. The obtained results show that ReHEvo is better than others in terms of accuracy.

Keywords: adaptive parameter, big data, data stream, stream clustering, stream mining, time window.

1. Introduction

The opportunities provided by technology and an exponential increase in the amount of data held electronically have caused numerous new problems and have led to the emergence of new research areas in astronomy (Chen et al., 2014; Chen & Zhang, 2014), information technology (Power, 2014), e-commerce (Akter & Wamba, 2016), bioinformatics (Chen et al., 2014; Lesk, 2019), scientific experiment (e.g. big hadron collider, see (Kambatla et al., 2014), social media (Tufekci, 2014; Manovich, 2011), etc. Stream processing is a fundamental approach to cope with the increasing data that has a high velocity.

The main reasons for using streaming are as follows: (i) preference to store small-sized information rather than storing large-scale data, (ii) processing incoming data that does not fit into memory in one time, and (iii) instant need for information and knowledge derived from meaningless data.

One of the widely used data mining techniques for data streams is clustering. For data streams clustering is a way to discover patterns and get knowledge from the stream. A cluster is a collection of elements in which the elements in the same group are similar and in which the elements in the different groups are not (Everitt, 1980). In data mining and machine learning, clustering, which is an NP-Hard problem (Gonzalez, 1985), is extensively studied (Xu & Wunsch, 2005). Compared to offline clustering, stream clustering is a newer research area. It has gained importance after the huge increase in data generation from different sources.

Zhang et al. proposed an algorithm called BIRCH (Zhang et al., 1997) which has a common method to handle stream data effectively, in 1997. After then, many stream clustering algorithms have been studied based on the micro-clustering technique, which is the extended version of cluster feature vectors that are used in BIRCH algorithm. Micro-clusters are inventive data structures that not only allow to create clusters in an online fashion but also allow clusters to stay up to date by capturing data drifts. Data stream is considered to be potentially infinite sequence of data. So, the stream clustering algorithms are designed as a single-pass strategy to overcome the negative effects of large-scale data. This means that stream clustering algorithms cannot access the data randomly. Because of these features, algorithms execute forward only concept and only perform forward operations due to the lack of data.

Dealing with highly driftable and dynamic data, the micro-clustering technique is intensively used(Aggarwal et al., 2003; de Andrade Silva et al., 2017; Laohakiat et al., 2017; Gong et al., 2017; Cao et al., 2006; Ntoutsi et al., 2012; Kranen et al., 2011; Hahsler & Bolaños, 2016; Challa et al., 2021). In fact, the cluster feature vectors developed in the BIRCH algorithm form the basis of stream clustering. The micro-clustering technique allows using time-window based weighting for the purpose of adapting to the new data and forgetting old ones in a convenient way. Stream clustering algorithms calculate cluster attributes in real-time through the flexible structure of micro-clusters.

In addition to the challenges of clustering, stream clustering becomes a much more difficult task due to the difficulties of the dynamic nature of streaming. These challenges can be grouped as follows:

- It is necessary to process the incoming data with high speeds and give real-time responses. To satisfy this constraint, data is usually accessed once (single pass). However, traditional methods can access data several times. That is, while traditional algorithms can make random access, it is not possible for stream algorithms. Because stream data is deleted after it is processed.

- Data stream is continuous and considered to be potentially endless sequence of data instances (Prasad & Agarwal, 2016). It is impossible to store and process such large data because of memory constraints. Therefore, mechanisms that can process large-scale data with less memory should be developed.

- Data stream may evolve over time (concept drift, see Silva et al., 2013; Song et al., 2016). Thus, there may be a mismatch between the first and subsequent data. Due to this situation (assuming $t_1 < t_2$)
    - The number of clusters at time t_1 may be different from time t_2.
    - Since the cluster, seen at time t_1, can be deleted or merged with another cluster, it may not be seen at time t_2.
    - Since new clusters, that is not seen at time t_1, can be created or one cluster can be split into two different clusters, it may be seen at time t_2.
    - The center of a cluster, which is at x_1 at time t_1, may shift to location x_2 at time t_2, where $x_1 \neq x_2$.

- Algorithms that run on data streams may have to learn a repeating pattern continuously (Khalilian & Mustapha, 2010).

- The main difficulty comes from the lack of knowledge of the whole data during clustering.

Clustering on data streams is different from clustering on stationary data. Therefore, algorithms working on data streams should have different capabilities and have to specialize to handle highly driftable data. In this paper, we propose a new stream clustering algorithm, called Recent History Based Evolving Data Clustering Algorithm (ReHEvo), that uses the last arrived data and adapts to new data quickly. In addition, ReHEvo also uses adaptive radius parameter to adapt to the different environments.

The main contributions of this study can be summarized as follows:

- During the clustering process, at any time *t*, ReHEvo can cluster the data stream. So ReHEvo is an anytime clustering algorithm.
- Traditional algorithms in the area of stream clustering, have an offline clustering stage to get ground truth clusters. ReHEvo obtains macro-cluster (estimated ground truth) in an online fashion. So, ReHEvo is a fully online stream clustering algorithm.
- Due to the high potential of the volume and velocity of data, state-of-art algorithms adopt single pass strategy. However, ReHEvo uses recent incoming data to overcome the cold start and rediscovery problem.
- Due to the statistical change of data over time, and using different datasets, ReHEvo utilizes adaptive parameters.

The rest of the paper is structured as follows: Section II provides a brief description of existing stream clustering algorithms. Section III includes a comprehensive explanation of the principles and components of the developed ReHEvo algorithm. The real datasets, scenarios that were created from synthetic datasets and their usage are explained and the evaluation criteria used are discussed in Section IV. In addition, in Section IV, we compared the results obtained from CluStream (Aggarwal et al., 2003), DenStream (Cao et al., 2006), ClusTree (Kranen et al., 2011) and the proposed algorithm ReHEvo and summarized the findings. We concluded the paper in Section V.

2. Related Work

In general, a data stream, which comes at specific times in order or in concurrent, is known as a potentially endless sequence of data instances with finite feature space d.

In stream clustering researches, various algorithms have been proposed to obtain more accurate clustering, including CluStream, DenStream, ClusTree and etc. Most of them use micro-clustering, additivity property and time window strategy (Aggarwal et al., 2003; Cao et al., 2006; Kranen et al., 2011 Ksieniewicz et al., 2019) to keep clusters up to date. Not surprisingly, most of them are based on micro-clusters. Micro-clustering is a basic way to control and manage data streams. To cope with the highly dynamic nature of data streams, the summarization technique was widely used in the past. Points that are close to each other are located in the same micro-cluster and one point in the micro-cluster represents the whole points that belong to this micro-cluster in the macro-clustering stage.

The stream clustering algorithms use time windowing, based on data instance age and assign them natural weights. The main idea of weighting is that the older data has a smaller weight, so they have less importance and hence, less impact on clustering; the recent incoming data has a higher weight value, so they have more importance and more impact on clustering. Therefore, the clustering process is driven by recent incoming data.

One-pass method is adopted in stream clustering algorithms. This means that the incoming data instances affect the clustering process and they are deleted as soon as possible, due to the memory constraints. At any time of clustering process, micro-clusters are discovered using some cluster attributes like linear sum of instances (LS), square sum of instances (SS), last edit time of micro-cluster (t) and etc. When a data instance arrives, it affects LS and SS incrementally with additivity property, then it is deleted.

The general working methods of clustering algorithms are as follows. At the beginning of the stream clustering process, the algorithm waits for the data until it reaches the buffer size. Then, the algorithm executes initial clustering methods on the initial data and get the initial groups. Generally, this step is an offline process and a traditional batch clustering algorithm such as k-means (Lloyd, 1982) and DBScan (Ester, 1996), is used. After the initial clustering, the stream clustering algorithm has two main stages (Dai et al., 2006; Hyde et al., 2017): (i) The micro-clustering stage that works online, and (ii) the macro-clustering stage that works offline. In earlier studies (Aggarwal et al., 2003; Cao et al., 2006), since the initial clustering step is a very small part of the stream clustering algorithm life-cycle, it is not shown as a stage of the clustering process. Recently, some researchers (de Andrade Silva et al., 2017; Bhatnagar et al., 2014) have attempted to perform the initial clustering stage again, due to the statistical changes of data (Figure1).

One of the most widely known algorithms is CluStream (Aggarwal et al., 2003), in which micro-cluster paradigm was proposed. CluStream uses k-means algorithm in the initial clustering step. System information is stored as a snapshot, according to every specific time frame t and these snapshots are utilized like a pyramidal pattern. Recent data is used more frequently than older data. After a while, older data is deleted systematically. CluStream uses additivity property to adapt to new data and also uses subtractive property to forget the old data. Once the micro-clustering stage is completed, macro-clusters are obtained with k-means algorithm in the macro-clustering stage.

The other widely used algorithm in the stream clustering domain is DenStream (Cao et al., 2006). DenStream uses density-based DBScan algorithm for the initial clustering stage and macro-clustering stage. Gradually temporal weighting is used in DenStream algorithm. DenStream attempts to manage outlier data in a separate buffer. If the weight of a micro-cluster is below a certain threshold, this micro-cluster is considered an outlier-micro-cluster and they are stored in the outlier buffer. If the weight of this micro-cluster exceeds a certain threshold over time, it is labeled as a potential-micro-cluster and is used for macro clustering.

Occasionally hierarchical clustering algorithms are proposed for data stream domain. Hierarchical algorithms are generally preferred for scalability and arbitrary shape concern. ClusTree (Kranen et al., 2011) is one of the leading stream clustering algorithms that applies a hierarchical method. Similar to CluStream and DenStream, ClusTree also summarizes the incoming data based on the micro-clustering strategy. Unlike other stream clustering algorithms, ClusTree presents a parameter-free algorithm due to the highly driftable data and the changeable environment in terms of speed and statistical information.

In recent years, evolutionary algorithms have been used to cluster stream data. Silva et. al proposed a genetic algorithm based stream clustering method, called FEAC-Stream (de Andrade Silva, 2017). Different from CluStream and ClusTree algorithms, FEAC-Stream has a component that tries to find the best value of k. To achieve this goal, it utilizes the silhouette coefficient evaluation index as a fitness function. Furthermore, FEAC-Stream has another useful component that detects a change of statistical distribution of incoming data. If is there any statistical change occurs above the predefined threshold

parameter λ1 and λ2, the algorithm triggers warning and alarm state respectively. Then, it is started to store new incoming data into the second buffer to perform reinitialization. In order to adapt new data FEAC-Stream uses time-based decay function like CluStream and DenStream. Therefore, for FEAC-Stream new data is more important than the old one.

3. Recent History Based Evolving Data Clustering Algorithm (ReHEvo)

A recent history-based algorithm was developed to obtain more accurate results on the data stream, called Recent History Based Evolving Data Clustering Algorithm (ReHEvo). Since the data stream concept is different from the stationary data concept, besides some of these issues mentioned below are also present when working traditional batch clustering, but can be more problematic when streams are clustered. Because it is more difficult to handle and maintain stream data than stationary data.

3.1. Problems Encountered

During the implementation of ReHEvo within the scope of this publication, we have encountered some problems that probably other researchers also have faced. In this section, the problems encountered during the implementation of the ReHEvo algorithm and its solutions are discussed.

Cluster Domination: The problem of cluster domination can be defined as the expansion of a single cluster to include all points if some limitations (radius, density, etc.) are not performed. Expert knowledge is required to determine the parameter values of these limitations. However, despite expert knowledge, parameter setting can be difficult, because the data stream is considered to be a potentially infinite data sequence. It is not possible to know the range of data without seeing all the data, so it is not effective to specify a fixed parameter value.

Although a restriction on the size of the cluster may seem problematic in terms of reaching the actual boundaries of the cluster, it is imperative to limit the cluster in some way, in order to avoid domination problem. A cluster view without any restriction is shown in Figure 2-c as a big circle.

Overlapping and nesting problems: If a relationship of a cluster with other clusters is not controlled, as well as its size, we may see overlapping or nested clusters as shown in Figure 3. In fact, a good clustering should not contain overlapping or nested clusters, the clusters should be significantly distinct. However, as can be seen from Figure 3, we can face such situations if the relationship between clusters is not under control.

The cluster centers and radii are dynamic; this dynamism can cause overlapping and nesting problem. Also, the center-surface paradox as shown in Figure 4, exacerbates the problems. That is, a new data point has the problem whether it should be included in any cluster according to the center or the surface of the cluster. The center information of the cluster was used by some algorithms such as k-means. If the cluster, in which the point is included, is determined according to the center information, the nesting and overlapping problems arise. Besides these, the order of the incoming data (Zhang et al., 1997; Aggarwal et al., 2003; de Andrade Silva et al., 2017; Hahsler & Bolaños, 2016; Bhatnagar et al., 2014; Rehman et al., 2014; Hyde & Angelov, 2015) affects clustering (Callister et al., 2017). So, the insertion order problem occurs (Hammouda & Kamel, 2003).

Domination to other features: In many areas, cleaning (of the noisy data), conversion, and normalization are generally performed in order to obtain more accurate results. In a d-dimensional data

space, if normalization is not performed, a d_1 attribute with high variance can dominate the effect of other attributes on the result (Jain et al., 1999; Das et al., 2007; Al-Harbi, & Rayward-Smith, 2006). The most important method to equalize the effect of features is normalization. If the dominating effect of the feature that has a large variance is not reduced, there may be inaccurate clustering as in Figure 5. The reason for the incorrect clustering in Figure 5 is that the attribute with a large variance has no distinctive effect on the data and no normalization is performed. So, this feature leads the results.

Unwieldiness of clusters and cluster memory: The data stream continuously arrives, and the clusters are updated according to the incoming data. Hence, the developed algorithm should keep itself updated according to the incoming data. However, the presence of old data effect, especially if they are many, minimizes the impact of new data in terms of keeping clusters up to date. The fidelity of the cluster to the old makes it difficult to adapt to the new data.

To adapt to the incoming data, researchers use time window, weighting and micro-clustering techniques. Although the data are processed by time-based weighting, the new incoming data is more important. The number of points in the cluster is the natural weight of that cluster. Adaptation to new data is not a one-way process. Besides adapting to the new data, the old data needs to be forgotten (Silva et al., 2013). Therefore, it is not only necessary to adapt to the incoming data in order to make accurate clustering at any time, but also to forget the old data in an appropriate way. Algorithms that do not have a proper forgetting mechanism will cause inaccurate clustering as in Figure 6. According to Figure 6, the black-circled clusters are ground truth and the green clusters are estimated clusters.

Existence and management of outliers: Data that does not actually belong to any cluster is called an outlier. Incoming data may also contain outliers due to different reasons, such as corruption during transmission or experimental error. Managing (detecting, deleting, ignoring, and etc.) outlier data is one of the prerequisites for the correct processing of data. It is important to rapid detection of outliers, whether on stationary data or on streaming data.

One of the most common challenges of working on data streams is the detection and management of outlier data. Outliers can cause more problems when working with data streams (Khalilian & Mustapha, 2010; Elahi et al., 2008; Baruah & Angelov, 2013). Because the data stream is continuous and incomplete data. In stream processing concept, detecting outliers is not enough, it is also needed to manage them in an online fashion (Cao et al., 2006; Prasad & Agarwal, 2016). Since we do not have the complete dataset, the whole picture is not visible. Some data identified as outlier data at any time t may also be outlier data, or the precursor data of a new cluster to be discovered at time t'. For this reason, some researchers have chosen to manage outlier data in different buffers.

According to Hyde et al., cluster elements are actually short-term true positives (2017). In other words, they argued that, at any time t, a cluster element p can be correctly clustered (TP), but if the algorithm does not have an appropriate forgetting strategy, after a while, they can be transformed into incorrectly clustered (FN). On the other hand, it can happen the other way; outliers can be viewed as short-term true negatives (TN). If they are kept in memory for a long time, they become FP after a while. Such a strategy leads to false clustering. For example, as shown in Figure 7, the point that has been identified as outlier data at time $t=8000$, entered into the cluster c_1 at time $t=12000$. Because the algorithm chooses to manage the outlier data. Thus, a wrong clustering occurred.

Due to the difficult management of outliers, keeping outliers in the system caused the loss of accuracy. Hence, deleting outliers was preferred in ReHEvo algorithm.

Forward only problem: The data stream comes in a continuous and rapid manner, so the data stream clustering algorithms have adopted single-pass strategy in order to cope with scalability. If no additional mechanism or data structure is used, previous data cannot be accessed again later. This means that while algorithms are running on the data streams, they cannot provide random access to the data. Random access is a waived property to stream data clustering algorithms to be faster and to save computer memory.

The fact that algorithms running on the data stream cannot access randomly leads to a major problem. Only forward operations can be performed on clusters. Adding and merging operations can be done. However, deleting, separation and re-clustering cannot be performed. Aggarwal and Carnein & Trautmann also mentioned this problem (2007; 2018).

In addition, it is not feasible to keep all incoming data in the memory. ReHEvo algorithm stores recently added k points as a cluster attribute, in order to figure out future trend and direction of evolution of data.

Different space and boundaries problem: Some researchers (Zhang et al., 1997; Aggarwal et al., 2003; Cao et al., 2006; Zhang et al., 2013) have reported that they manually change the parameter values of the algorithms according to the dataset. Although this is suitable for stationary data, it is not practically applicable for data streams. The major reasons for the algorithms to work with different parameter settings for each dataset are the differences in the number of dimensions between the datasets and the statistically significant change in the value of the incoming data.

The different number of dimensions causes some user-defined parameters to be inefficient. Some parameter values may need to be readjusted when the datasets change, especially if dimensionally-dependent distance metrics (Hyde et al., 2017) such as Euclidean and Minkowski are used as similarity criteria. Therefore, if there is a parameter that will be affected by the number of dimensions, it should be adaptive. So, in ReHEvo, we use proactive adaptive parameter (PAP) to adapt to the dimension change.

The fact that the values of the incoming data change statistically over time is another reason for adaptation in order to find a better effective radius parameter. Considering this situation, the researchers updated the radius parameters using the in-cluster standard deviation (Aggarwal et al., 2003; Kranen et al., 2011). So, in ReHEvo there is one more parameter that sets the radius value according to the statistical information of the incoming data which we call reactive adaptive parameter (RAP). As seen in Figure 8, there may be clusters of different scales and densities in the studied space. Using the same radius value for these clusters with different widths will result incorrect clustering. Therefore, the standard deviation value within the cluster is also used to determine the cluster radius value.

Cold start problem: The term cold start problem (Bobadilla et al., 2012; Lika et al., 2014) is actually a problem that is emphasized in the literature of recommender systems, especially in collaborative filtering (Herlocker et al., 2004; Milli & Bulut, 2017). In such recommender systems, when a new product is included in the system, it is difficult to recommend this product to the users. Because there is no one who has evaluated this product before. Similarly, when a new user is registered in the system since the user has no prior assessment, it is difficult to suggest any product to this user. In addition, if the recommender system has just been established, it is difficult to make suggestions in this system; because of the empty user-item rating matrix. For these reasons, cold start problem may be present in recommender systems. A similar problem exists in data stream clustering as well. The initial clustering stage in data stream clustering algorithms is a step to avoid the cold start problem.

Initial clustering is performed offline using one of the conventional clustering algorithms. In fact, the initial clustering process prevents some newborn clusters being stuck and disappear due to the user-defined parameters such as min-points (Hahsler & Bolaños, 2016; Ester, 1996) and gives a chance to newborn clusters for a while. In addition, initial clustering provides a summary of the patterns stored in the data and uses this information during the online clustering phase. Therefore, initial clustering is one of the important stages of data stream clustering algorithms. In order to avoid the cold start problem in ReHEvo, we always keep the recent k points for each cluster.

3.2. Proposed Algorithm (ReHEvo)

The ReHEvo algorithm is a data stream algorithm that also uses a density-based micro-cluster structure for data that can evolve over time. It uses the weights determined over time frames to determine cluster boundaries and radii; however, the time-based weighting is not used to assess the status of clusters in the system. To avoid cluster monopolization and cluster unwieldiness, old clusters are automatically deleted from the system.

3.2.1 Initial Clustering

In ReHEvo, subtractive clustering technique (Chiu, 1994) was used for initial clustering. The subtractive clustering technique is mostly used in fuzzy logic applications, due to the success to find the optimal number of clusters, but it is a technique that makes crisp clustering. In the initial clustering phase, another conventional clustering algorithm may be used.

Figure 9, shows the flowchart of the developed ReHEvo algorithm. The ReHEvo algorithm consists of two main parts: initial clustering and online clustering. Unlike conventional data stream clustering algorithms, there is no offline clustering phase. Based on the overall ReHEvo algorithm, it is completely online and is distinguished from its peers in the literature by this feature.

The subtractive clustering algorithm takes four different parameters as inputs. First, the potential of each point in the dataset is calculated according to their position with respect to each other. If the number of neighbors of a point is high, the potential of that point is high. The point which has a high-potential is the candidate point to become a cluster center. According to the components of the initial clustering step, the incoming data in the buffering step is accumulated until it reaches the user-specified initial data size. Then, if the data has been accumulated sufficiently, preprocessing (cleaning, normalization, etc.) is done if it is required. After preprocessing the data, the algorithm parameters are set to their initial values. For example, the PAP parameter takes values according to the dimension of the incoming data. In the initial clustering phase, the first clusters are obtained on the initial data using the selected clustering algorithm and the initial clustering phase is terminated. If there is a parameter that needs to be updated according to the statistical values of the incoming data, these operations are performed in the adjust parameter component. For example, the RAP parameter is set based on standard deviations within the cluster.

3.2.2 Online Maintenance

After the initial clustering is over, we now have clusters and labels. Since the groups are determined, further stages can be seen as a classification problem (Figure 9). Each incoming stream data is assigned to

the closest cluster within the radius parameter. Otherwise, a new cluster with a single point micro-cluster is generated. At the end of a specified period, overlapping micro-clusters and macro-clusters are merged. A micro-cluster or macro-cluster, stores recently added *k* points in its memory as a cluster attribute. These recent points can be added to the cluster by joining or merging process. Before the cluster is deleted, the successor of that cluster is created with the last *k* points and that cluster is deleted from the system. Newborn cluster only inherits the recent *k* member of parent cluster.

Then the algorithm executes the outlier management component. At this stage, only macro-clusters and outliers remain in the system. The user-defined minPoints parameter allows to easily distinguish outliers from actual clusters. Hyde et al. indicated that cluster members are short-term true positives after a while they may turn into false positives due to the highly dynamic nature of data streams (Hyde et al., 2017). That means, the incoming data changes constantly and the clustered points that are held in the system for a long time will cause incorrect clustering after a while. Likewise, managing outlier data can cause similar problems. This problem is mentioned in Section 3 For this reason, we choose not to manage the outlier data and delete it after it is detected. Only macro-clusters remain in the cluster list after deleting the outlier data. The cluster list is passed onto the cluster evaluation component. If it is needed, the RAP parameter is updated.

Adaptation of new data is not a one-way operation and it is necessary to adapt to new data and forget the old ones appropriately (Silva et al., 2013). For these reasons, in ReHEvo algorithm, a cluster that reaches a certain number of elements (maxPoints) is deleted. This deletion is actually the forgetting mechanism of the system to reduce the effect of the past data. But, if the forgetting process is not completed properly, the algorithm will rediscover the repeating pattern. In this case, the cold start problem in Section 3 arises. A new cluster is created with the last k data added to the cluster to avoid rediscovery. In this way, we begin a new period with a young cluster that has forgotten old data in an appropriate way.

In the reorganize component, if a cluster has grown sufficiently, it is deleted without causing the unwieldiness and cluster monopolization problem. While cluster deletion is an easy and short way to get rid of the negative effect of old data, such a solution causes another side effect called cold start problem. Although the old clusters are deleted to avoid the aforementioned problems, the future potential elements of this cluster will most likely continue to arrive in the space occupied by this cluster before it is deleted. In order to avoid having to rediscover the deleted cluster and to ensure not to delete the new-born cluster as outliers, the last *k* data added to the clusters is stored in cluster objects as a cluster attribute in contrast to mentioned in (Carnein & Trautmann, 2019). If a cluster is deleted, a more recent version of this cluster is re-created with the last *k* point. This is the main idea underlying the ReHEvo algorithm. Thus, after a cluster has been deleted, ReHEvo does not make a cold start, if new data for the deleted cluster continues to come.

If the newborn cluster was not derived from the parent cluster;

(i) The algorithm, would have to rediscover the continuing pattern,

(ii) Because of the parameters given to the algorithm, we would encounter the cold start problem, especially in less dense clusters; perhaps a good cluster would never have arisen.

Adaptive Radius Parameter

In (Carnein et al., 2019) mentioned that the challenging issue for data analysts that choosing the appropriate algorithm with the appropriate parameter value for the current situation. the The ReHEvo

algorithm has an adaptive radius parameter, unlike its peers in the literature. The radius parameter in ReHEvo consists of three components:

- Predefined User Parameter (PUP)
- Proactive Adaptive Parameter (PAP)
- Reactive Adaptive Parameter (RAP)

PUP is the user-specified component; that is, the initialization of the radius parameter provided by the user. Again, another radius parameter component RAP is a component that is calculated based on the statistics of the data, which is also used by other algorithms (Aggarwal et al., 2003; Kranen et al., 2011). If the standard deviation within the cluster is low, it decreases the expected radius value and if the standard deviation within the cluster is high, it increases the expected radius value.

One of the most important reasons for the ineffectiveness of the parameter values, whether it is stationary data or data stream, is the difference in the dimension of the datasets. Working on a d_1 dimensional X dataset is not the same as working on a d_2 dimensional Y dataset, where $d_1 \neq d_2$. Because the change in the number of dimensions changes the boundaries of the space studied. This leads to predetermined parameters become inefficient (such as radius). In this study, an adaptive method is proposed to adjust the distance parameter according to the number of dimensions of the incoming data to minimize this problem caused by the change of the boundaries of space due to the different number of dimensions.

If the distance parameter p_1 is used for an S_1 space with dimensional d_1, it should be updated so that this parameter becomes $k.p_1$ in order to be effective in an S_2 space with dimensional d_2. This k value seems to be the ideal solution to be the ratio of the area bounded by the space S_2 to the area bounded by the space S_1. However, it is impossible to find the ratio of the area bounded by a d_1 dimensional system to the area bounded by a d_2 dimensional system. It is not an acceptable comparison. Therefore, we determined the k value in the experimental studies as follows: Considering the Euclidean distance in 2-dimensional unit space, the farthest distance between two data points is the length of the diagonal and $\sqrt{2}$ Likewise, the farthest distance between two data points in 3-dimensional unit space is the length of the space diagonal and is $\sqrt{3}$. In a d-dimensional space, the farthest distance between the two data points is also $\sqrt{d}$.

Considering the method described above, the distance parameters defined in the algorithm should be adaptive according to the number of dimensions. Therefore, in the proposed method, the square root value of the number of dimensions is determined as the PAP value. Because, according to the changing number of dimensions, the farthest distance between the two points increases in proportion to the square root of the number of dimensions.

According to the Algorithm 1 μ is the PUP parameter, that initializes the radius value and passed the algorithm as an input. $\sqrt{d}$ is actually the parameter we mentioned earlier as PAP and σ, called RAP parameter is calculated online, it is the standard deviation of intra-cluster. The exact micro-cluster radius value is determined with the integration of these three parameters. The data comes point by point continuously and the ReHEvo algorithm checks whether each incoming data fits into any cluster c_i in cluster list C, under the consideration of calculated radius. At the end of the data collection period, the maintenance period starts. While the ReHEvo algorithm performs cluster maintenance, incoming data is stored in a buffer. In the maintenance period, if there is an overlap between the two clusters, the young

cluster is merged with the old one and the recentHistory is updated after these transactions the young cluster is deleted from C. The last transaction of the maintenance period is detecting the outliers. The minPoints threshold (Ester et al., 1996) has been used in order to detect outliers. Outlier data is deleted due to the previously mentioned objectionable situations and the complications they cause while clustering online. After the evaluation period, late maintenance procedures are performed. If the number of elements of any cluster c_i in cluster list C exceeds maxPoints, a new cluster is created via the last added *k* element of c_i in order to avoid getting stuck in the unwieldiness of the cluster. So it can be said that c_{n+1} is a newborn cluster without inheriting any features of its parent. Then the new period is started. This is the process cycle of the ReHEvo algorithm that has been mentioned in Algorithm 1.

Algorithm 1: ReHEvo

Data:	Stream data	*S*
	Initial clusters	*C*
	PUP parameter	*μ*
	Threshold for outliers	*minPoints*
	Threshold for deletion	*maxPoints*
	The number of point keep in memory	*k*

Output: Cluster list $C = \{C_0, C_1, ..., C_n\}$ and each C_i has an attribute vector.

```
//
// Intra-period transactions
foreach p ∈ S do
    c ← min p, c_i in C
    c' ← copy(c)
    c' ← c' ∪ {p}
    if r' ≤ μ × d × σ then
        c ← c ∪ {p}
        update(recentHistory)
    else
        Add p to C as a new cluster c_{n+1}
    end
end
//
// At the end of the period before cluster evaluations
foreach c', c'' ∈ C do
    if c' ∩ c'' /= ∅ then
        c' ← c' ∪ c''
        update(recentHistory)
        Delete c'' from list C
    end
end
foreach c ∈ C do
    if |c_i| < minPoints then
        Delete c_i from list C
    end
end
//
//
// At the beginning of the period after cluster evaluations
foreach c ∈ C do
    if |c_i| > maxPoints then
        Create a new cluster with the last k element of c_i and delete c_i from the
cluster list C
    end
end
```

4. Experimental Results

This section describes the evaluation criteria and the datasets used. The real and synthetic datasets used in the comparison of algorithms will be introduced. The results will be compared with CluStream, DenStream, and ClusTree algorithms.

Experimental results were taken on a computer with a 64-bit Intel i7-3630QM 2.40GHz processor, and a Windows 7 operating system with 8 GB RAM. The Massive Online Analysis (MOA) (Bifet et al., 2010) framework includes the compared algorithms and this framework is used for experimental results. The developed algorithms are also implemented within the MOA environment.

4.1. Datasets and Evaluation Metrics

Cluster evaluation metrics show how well a group of data is divided into pieces, with or without prior knowledge of the data. There are two different cluster evaluation metrics in the literature (Rendón et al., 2011; Kremer et al., 2011): (i) internal validation and (ii) external validation.

Internal validation metrics: These metrics do not have any previous information or label values about the data. The Sum Squared Distance SSQ (Han et al., 2011) and Silhouette Index (Rousseeuw, 1987) are the most well-known and used internal validation metrics. External validation metrics: These metrics already have some information about data and use the label information to evaluate the data. F-measure (van Rijsbergen, 1979), F1-Precision, F1-Recall, Purity (Zhao & Karypis, 2004), Rand-Index (Rand, 1971) are the most well-known and used external validation metrics. Therefore, the results of the ReHEvo algorithm were compared with the results of other algorithms and these validation metrics were used where two of them are internal and four of them are external metrics.

Both synthetic and real datasets were used to compare the algorithms. Synthetic datasets were generated according to different scenarios using the RandomRBFGenerator of the MOA simulation environment. The data generator is based on radial basis functions that randomly generate a fixed number of the cluster center and the data points come around these centers with a standard deviation randomly. So, it generates hyper-spherical clusters with Gaussian distribution.

Synthetic data: Different scenarios are produced as synthetic datasets. Scenarios were obtained by changing the number of clusters, number of clusters interval, noise ratio, number of dimensions, cluster size, the standard deviation of cluster size, merge/separation, and creation/deletion parameters. For example, in the second scenario shown in Table 1, only the ratio of outliers has been changed differently from the default scenario, and the behavior of the algorithms to this change was observed. Similarly, 4 scenarios were created by changing one of the different environment variables and keeping the others constant.

Real datasets: In this section, frequently used datasets, which are mentioned in almost all publications in stream clustering literature, are introduced. Stream data clustering algorithms are usually compared on two real datasets: (i) Forest Cover Type[1] (Blackard & Dean, 1999), (ii) Network Intrusion Detection – KDD CUP'99[2]. These data are stationary data, but simulation environment use them in a streaming fashion.

Forest Cover Type: This dataset is a set of seven different plant types in the Roosevelt National Forest, spread over four different regions, including Rawah, Comanche Peak, Neota, and Cache la Poudre in Northern Colorado. Only seven most dominant plant species in these areas are examined (Blackard & Dean, 1999).

---

[1] https://archive.ics.uci.edu/ml/datasets/covertype

[2] http://kdd.ics.uci.edu/databases/kddcup99/kddcup99.html

The Forest Cover Type dataset consists essentially of 54 attribute spaces, of which 10 are continuous and 44 are binary data. In all studies examined in the literature, only continuous features were used. Therefore, we conducted our experiments on the first 10 features.

Network Intrusion Detection (KDD Cup'99): The KDD Cup'99 dataset contains information about which connections to a server are secure and which are attack attempts. This dataset is one of the most widely used datasets for network intrusion detection, which is derived from the "DARPA 1998" dataset. This dataset contains approximately 4.8 million connection information; however, in the literature, many researchers used only 10% of this dataset. Similar to earlier studies, we use only 494.020 records which are 10% of the entire dataset for experimental studies. Each record corresponds to a normal connection or an intrusion attempt. The dataset includes 23 classes, where 22 of them is a type of attack (neptune, smurf, spy, etc.) and one of them is normal (innocent) connection.

The KDD Cup'99 dataset consists of 42 feature (34 continuous and 8 binary feature) spaces. Researchers (Aggarwal et al., 2013; Ksieniewicz et al., 2019) using this dataset in their publications have worked on 34 continuous data. So, we use only 34 continuous features for experiments.

4.2. Experiments

In this section, the developed ReHEvo algorithm, and stream clustering algorithms in the literature are run on different datasets and compared in different aspects. Experiments were performed on synthetic datasets and real datasets. Patterns in artificially obtained datasets are relatively easier to find than patterns in real datasets. For this reason, the majority of algorithms work with high accuracy on synthetic datasets. However, in order to evaluate the study in more detail, different scenarios were produced while working with synthetic datasets.

CluStream and ClusTree algorithms derive the number of ground truth directly from the simulation when performing offline clustering. The developed ReHEvo algorithm and DenStream try to find the actual number of clusters themselves. In addition, the evaluation criteria such as purity and SSQ are affected by the number of clusters found (Hahsler & Bolaños, 2016). The more clusters an algorithm finds, the better the purity and SSQ values. Therefore, when interpreting purity and SSQ values, it is necessary to look at the number of clusters found by the algorithms.

In order to compare the real performance of ReHEvo, we diversified the synthetic dataset. The first scenario is the default parameter of the MOA tool. Table 2 indicates the result of algorithms that run on scenario 1. According to these results, the ReHEvo algorithm has achieved better values in terms of 5 evaluation metrics. Only CluStream achieved a better value for the SSQ metric. The biggest reason for this is that the SSQ and Purity indexes are affected by the number of clusters found by algorithms, and CluStream has more clusters. And it is because k-means algorithm aims to minimize the within-cluster sum of squares as an objective function and CluStream use k-means in the macro-clustering stage.

The results showed that the best metric of ReHEvo is F1-P. ReHEvo does not store the data detected as an outlier in the buffer, in order to prevent depending on the old data and misleading to new decisions. Because of the policies of algorithms that want to manage outliers, false-positive values increase. This leads to a decrease in F1-precision values. Since incoming data is evaluated on a periodic basis, managing outliers has no direct impact on recall.

Table 3 shows the performance of the developed ReHEvo algorithm on scenario 2. This scenario has noisier data than the previous one. We increase the outlier ratio from 0.1 to 0.25 in order to evaluate the resistance of algorithms to noisy data. Table 3 indicates that ReHEvo gets more accurate results than other algorithms in four evaluation metrics. This shows that ReHEvo is more resistant to noisy data than other algorithms. In an environment with intense outlier data, DenStream finds more clusters than it actually exists, due to the tendency of managing outliers. Outliers placed in the same locations succeed to enter the real cluster list after a while. Therefore, although its clustering quality does not improve, it achieves better results in terms of purity and SSQ.

Table 4 shows the results obtained from scenario 3. In this scenario, the number of dimensions is increased from 2 to 10. With this scenario, the behavior of algorithms in high dimensional environments is measured. Again, it was observed that ReHEvo algorithm obtained good results in all accuracy values.

According to Table 4, a significant increase was observed in the accuracy of all algorithms except DenStream when the number of dimensions is increased. The main reason for this situation is that we are working with artificial data. When working with artificial data, increasing the number of dimensions increases the inter-cluster distances more than intra-cluster distances. In other words, increasing the number of dimensions increases the dissimilarity between the data in different clusters. Therefore, it is easier to distinguish clusters from each other. This is the reason for the increase in accuracy values in Scenario 3.

In Figure 10 there is a comparison between a two-dimensional system and a four-dimensional system. The overlap of clusters makes it difficult for algorithms to find clusters. As seen from Figure 10, the probability of overlapping of two clusters in a two-dimensional system is greater than the probability of overlapping of two clusters in a four-dimensional system. That is, increasing the number of dimensions will reduce the probability of overlapping of two clusters. Likewise, in a two-dimensional system, the stochastic similarity between an outlier and any cluster is greater than in a four-dimensional system. So, algorithms have found clusters easier. In scenario 3, where the dimension is increased, the accuracy values of the compared algorithms and the developed ReHEvo algorithm increase and converge to the ideal. However, one of the factors affecting the complexity of the algorithms is also the number of dimensions of the data. Some algorithms are adversely effected by the increase in the number of dimensions in terms of computational complexity. Therefore, the running times of the experiments performed with scenario 1 and scenario 3 are given in Figure 11. As seen in Figure 11, ClusTree and ReHEvo algorithms are the least effected algorithms with the increase of the dimension. Each algorithm in Figure 11 has been run 50 times for both scenarios and the results are the average of them.

A more dynamic environment was designed with the dataset created in Scenario 4 and the results are shown in Table 5. The purpose of this scenario was to observe the behavior of the algorithms in the dynamic environmental conditions. In addition, it has been tested how quickly algorithms can discover new patterns and how quickly they can forget the old ones. In this scenario, the number of clusters is not constant. One event (create, delete, merge, split) occurs every 30,000 data. With this event, the number of clusters changes to 5 (± 3). In this dynamic scenario, ReHEvo achieved better results than other algorithms in 5 accuracy metrics These results show that the ReHEvo has the ability to focus on new clusters and forget about old clusters.

Next, we used Forest Cover Type dataset which is a well-known and widely used dataset in data stream clustering, to compare algorithms. It is difficult to find clusters for this dataset since the ground truth groups are heavily overlapping with each other (Hahsler & Bolaños, 2016). Therefore, the success rate of all algorithms is low. In this experiment, the runtime parameters of the CluStream, DenStream, and ClusTree algorithms are set to their best value. For ReHEvo, the PUP parameter (predefined user parameter) was kept unchanged. According to Table 6, the developed ReHEvo algorithm yielded better results in the 3 accuracy metrics measured. DenStream, on the other hand, produced better results in SSQ and purity values due to finding higher number of clusters.

Finally, we used the KDD CUP'99, which is another popular dataset, in order to evaluate the data stream clustering algorithm. Again, other algorithms except ReHEvo are set to the ideal parameter values for this dataset. In this experiment, DenStream and ReHEvo algorithms obtained better results than other algorithms. ReHEvo has achieved better results for 5 of the 6 accuracy indexes. The results are shown in Table 7.

5. Conclusion

We developed a new and novel stream clustering algorithm, called ReHEvo. It uses time windows in the online maintenance of clusters, but it does not use time-based weighting when deciding which clusters to continue and which ones to delete. When a cluster grows to a predefined size, it is deleted to eliminate the dependence on the old data and a new cluster is created with k most recent data. Therefore, the adaptive parameters in ReHEvo enable it to easily adapt to different environments.

The experimental results showed that ReHEvo achieved better results than other algorithms in both average and extreme conditions (outlier, dynamism, etc.). This success depends on using the delete and forget mechanisms, recent history information, and using adaptive parameters. It deletes old clusters and forgets them in a proper way and easily adapts to the new data with the recent history knowledge. In addition, with the RAP and developed PAP parameters, it can easily adapt to changing environments.

The offline clustering stage in the ReHEvo algorithm is completely eliminated with the online integration of micro-clusters. Therefore, ReHEvo is a completely online algorithm, which is more suitable for data streams. This makes it faster than many data stream clustering algorithms.

Acknowledgements

This research did not receive any specific grant from funding agencies in the public, commercial, or not-for-profit sectors.

Disclosure statement

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

References

Aggarwal, C. C., Philip, S. Y., Han, J., & Wang, J. (2003). A framework for clustering evolving data streams. In *Proceedings 2003 VLDB conference* (pp. 81-92). Morgan Kaufmann.

Aggarwal, C. C. (2007). *Data streams: models and algorithms* (Vol. 31). Springer Science & Business Media.

Akter, S., & Wamba, S. F. (2016). Big data analytics in E-commerce: a systematic review and agenda for future research. *Electronic Markets*, *26*(2), 173-194.

Al-Harbi, S. H., & Rayward-Smith, V. J. (2006). Adapting k-means for supervised clustering. *Applied Intelligence*, *24*(3), 219-226.

Baruah, R. D., & Angelov, P. (2013). DEC: Dynamically evolving clustering and its application to structure identification of evolving fuzzy models. *IEEE transactions on cybernetics*, *44*(9), 1619-1631.

Bhatnagar, V., Kaur, S., & Chakravarthy, S. (2014). Clustering data streams using grid-based synopsis. *Knowledge and information systems*, *41*(1), 127-152.

Bifet, A., Holmes, G., Kirkby, R., & Pfahringer, B. (2010) . Moa: Massive online analysis. *Journal of Machine Learning Research*, 1601-1604.

Blackard, J. A., & Dean, D. J. (1999). Comparative accuracies of artificial neural networks and discriminant analysis in predicting forest cover types from cartographic variables. *Computers and electronics in agriculture*, *24*(3), 131-151.

Bobadilla, J., Ortega, F., Hernando, A., & Bernal, J. (2012). A collaborative filtering approach to mitigate the new user cold start problem. *Knowledge-based systems*, *26*, 225-238.

Callister, R., Lazarescu, M., & Pham, D. S. (2017, April). Graph-based clustering with DRepStream. In *Proceedings of the Symposium on Applied Computing* (pp. 850-857).

Cao, F., Estert, M., Qian, W., & Zhou, A. (2006). Density-based clustering over an evolving data stream with noise. In *Proceedings of the 2006 SIAM international conference on data mining* (pp. 328-339). Society for industrial and applied mathematics.

Carnein, M., & Trautmann, H. (2018). evostream–evolutionary stream clustering utilizing idle times. *Big data research*, *14*, 101-111.

Carnein, M., & Trautmann, H. (2019). Optimizing data stream representation: An extensive survey on stream clustering algorithms. *Business & Information Systems Engineering*, *61*(3), 277-297.

Carnein, M., Trautmann, H., Bifet, A., & Pfahringer, B. (2019). Towards automated configuration of stream clustering algorithms. In *ECML PKDD 2019* (pp. 137-143). Springer.

Challa, J. S., Goyal, P., Kokandakar, A., Mantri, D., Verma, P., Balasubramaniam, S., & Goyal, N. (2021). Anytime clustering of data streams while handling noise and concept drift. *Journal of Experimental & Theoretical Artificial Intelligence*, 1-31.

Chen, C. P., & Zhang, C. Y. (2014). Data-intensive applications, challenges, techniques and technologies: A survey on Big Data. *Information sciences*, *275*, 314-347.

Chen, M., Mao, S., & Liu, Y. (2014). Big data: A survey. *Mobile networks and applications*, *19*(2), 171-209.

Chiu, S. L. (1994). Fuzzy model identification based on cluster estimation. *Journal of Intelligent*

& fuzzy systems, *2*(3), 267-278.

Dai, B. R., Huang, J. W., Yeh, M. Y., & Chen, M. S. (2006). Adaptive clustering for multiple evolving streams. *IEEE transactions on knowledge and data engineering*, *18*(9), 1166-1180.

Das, S., Abraham, A., & Konar, A. (2007). Automatic clustering using an improved differential evolution algorithm. *IEEE Transactions on systems, man, and cybernetics-Part A: Systems and Humans*, *38*(1), 218-237.

de Andrade Silva, J., Hruschka, E. R., & Gama, J. (2017). An evolutionary algorithm for clustering data streams with a variable number of clusters. *Expert Systems with Applications*, *67*, 228-238.

Elahi, M., Li, K., Nisar, W., Lv, X., & Wang, H. (2008, October). Efficient clustering-based outlier detection algorithm for dynamic data stream. In 2008 Fifth International Conference on Fuzzy Systems and Knowledge Discovery (Vol. 5, pp. 298-304). IEEE.

Ester, M., Kriegel, H. P., Sander, J., & Xu, X. (1996, August). A density-based algorithm for discovering clusters in large spatial databases with noise. In *kdd* (Vol. 96, No. 34, pp. 226-231).

Everitt, B. (1980). *Cluster analysis, 2nd edition* London: Social Science Research Council.

Gong, S., Zhang, Y., & Yu, G. (2017). Clustering stream data by exploring the evolution of density mountain. *Proceedings of the VLDB Endowment*, *11*(4), 393-405.

Gonzalez, T. F. (1985). Clustering to minimize the maximum intercluster distance. *Theoretical computer science*, *38*, 293-306.

Hahsler, M., & Bolaños, M. (2016). Clustering data streams based on shared density between micro-clusters. *IEEE Transactions on Knowledge and Data Engineering*, *28*(6), 1449-1461.

Hammouda, K. M., & Kamel, M. S. (2003, October). Incremental document clustering using cluster similarity histograms. In *Proceedings IEEE/WIC International Conference on Web Intelligence (WI 2003)* (pp. 597-601). IEEE.

Han, J., Pei, J., & Kamber, M. (2011). *Data mining: concepts and techniques*. Elsevier.

Herlocker, J. L., Konstan, J. A., Terveen, L. G., & Riedl, J. T. (2004). Evaluating collaborative filtering recommender systems. *ACM Transactions on Information Systems (TOIS)*, *22*(1), 5-53. L. Herlocker, J. A. Konstan, L. G. Terveen, J. T. Riedl, Evaluating collaborative filtering recommender systems, ACM T. Inform. Syst. 22(1) (2004) 5-53.

Hyde, R., & Angelov, P. (2015). A new online clustering approach for data in arbitrary shaped clusters. In 2015 IEEE 2nd International Conference on Cybernetics (CYBCONF) (pp. 228-233). IEEE.

Hyde, R., Angelov, P., & MacKenzie, A. R. (2017). Fully online clustering of evolving data streams into arbitrarily shaped clusters. *Information Sciences*, *382*, 96-114.

Jain, A. K., Murty, M. N., & Flynn, P. J. (1999). Data clustering: a review. *ACM computing surveys (CSUR)*, *31*(3), 264-323.

Kambatla, K., Kollias, G., Kumar, V., & Grama, A. (2014). Trends in big data analytics. *Journal of parallel and distributed computing*, *74*(7), 2561-2573.

Khalilian, M., & Mustapha, N. (2010). Data stream clustering: Challenges and issues. *arXiv preprint arXiv:1006.5261*.

Kranen, P., Assent, I., Baldauf, C., & Seidl, T. (2011). The clustree: indexing micro-clusters for anytime stream mining. *Knowledge and information systems*, *29*(2), 249-272.

Kremer, H., Kranen, P., Jansen, T., Seidl, T., Bifet, A., Holmes, G., & Pfahringer, B. (2011, August). An effective evaluation measure for clustering on evolving data streams. In *Proceedings of the 17th ACM SIGKDD international conference on Knowledge discovery and data mining* (pp. 868-876).

Ksieniewicz, P., Woźniak, M., Cyganek, B., Kasprzak, A., & Walkowiak, K. (2019). Data stream classification using active learned neural networks. *Neurocomputing*, *353*, 74-82.

Laohakiat, S., Phimoltares, S., & Lursinsap, C. (2017). A clustering algorithm for stream data with LDA-based unsupervised localized dimension reduction. *Information Sciences*, *381*,

104-123.
Lesk, A. (2019). *Introduction to bioinformatics*. Oxford university press.
Lika, B., Kolomvatsos, K., & Hadjiefthymiades, S. (2014). Facing the cold start problem in recommender systems. *Expert Systems with Applications*, *41*(4), 2065-2073.
Lloyd, S. (1982). Least squares quantization in PCM. *IEEE transactions on information theory*, *28*(2), 129-137.
Manovich, L. (2011). Trending: The promises and the challenges of big social data. *Debates in the digital humanities*, *2*(1), 460-475.
Milli, M., & Bulut, H. (2017). The Effect of Neighborhood Selection on Collaborative Filtering and a Novel Hybrid Algorithm. *Intelligent Automation & Soft Computing*, *23*(2), 261-269.
Ntoutsi, I., Zimek, A., Palpanas, T., Kröger, P., & Kriegel, H. P. (2012, April). Density-based projected clustering over high dimensional data streams. In *Proceedings of the 2012 SIAM international conference on data mining* (pp. 987-998). Society for Industrial and Applied Mathematics.
Power, D. J. (2014). Using 'Big Data'for analytics and decision support. *Journal of Decision Systems*, *23*(2), 222-228.
Prasad, B. R., & Agarwal, S. (2016). Stream data mining: platforms, algorithms, performance evaluators and research trends. *International journal of database theory and application*, *9*(9), 201-218.
Rand, W. M. (1971). Objective criteria for the evaluation of clustering methods. *Journal of the American Statistical association*, *66*(336), 846-850.
Rehman, M. Z. U., Li, T., Yang, Y., & Wang, H. (2014). Hyper-ellipsoidal clustering technique for evolving data stream. *Knowledge-Based Systems*, *70*, 3-14.
Rendón, E., Abundez, I., Arizmendi, A., & Quiroz, E. M. (2011). Internal versus external cluster validation indexes. *International Journal of computers and communications*, *5*(1), 27-34.
Rousseeuw, P. J. (1987). Silhouettes: a graphical aid to the interpretation and validation of cluster analysis. *Journal of computational and applied mathematics*, *20*, 53-65.
Silva, J. A., Faria, E. R., Barros, R. C., Hruschka, E. R., Carvalho, A. C. D., & Gama, J. (2013). Data stream clustering: A survey. *ACM Computing Surveys (CSUR)*, *46*(1), 1-31.
Song, G., Ye, Y., Zhang, H., Xu, X., Lau, R. Y., & Liu, F. (2016). Dynamic clustering forest: an ensemble framework to efficiently classify textual data stream with concept drift. *Information Sciences*, *357*, 125-143.
Tufekci, Z. (2014, May). Big questions for social media big data: Representativeness, validity and other methodological pitfalls. In *Eighth international AAAI conference on weblogs and social media*.
van Rijsbergen, C. J. (1979). Information Retrieval, 2nd edButterworths.
Xu, R., & Wunsch, D. (2005). Survey of clustering algorithms. *IEEE Transactions on neural networks*, *16*(3), 645-678.
Zhang, T., Ramakrishnan, R., & Livny, M. (1997). BIRCH: A new data clustering algorithm and its applications. *Data Mining and Knowledge Discovery*, *1*(2), 141-182.
Zhang, X., Furtlehner, C., Germain-Renaud, C., & Sebag, M. (2013). Data stream clustering with affinity propagation. *IEEE Transactions on Knowledge and Data Engineering*, *26*(7), 1644-1656.
Zhao, Y., & Karypis, G. (2004). Empirical and theoretical comparisons of selected criterion functions for document clustering. *Machine learning*, *55*(3), 311-331.

Tables

Table 1. Overview all the used datasets.

Scenario / Parameter	Data Size	Cluster Number	Outlier	Dimension
1 (default)	1m	5	10%	2
2 (noisy)	1m	5	25%	2
3 (high dimension)	1m	5	10%	10
4 (variable number)	1m	5 (+3, -3)	10%	2
forest cover type	581k	7	N/A	10
KDD CUP'99	494k	23	N/A	34

Table 2. The results for scenario 1 (default parameter of MOA RandomRBFGenerator).

Scenario 1	CluStream	DenStream	ClusTree	ReHEvo
F1-P	0.757 (± 0.129)	0.610 (± 0.121)	0.725 (± 0.142)	**0.920 (± 0.056)**
F1-R	0.710 (± 0.078)	0.595 (± 0.068)	0.695 (± 0.086)	**0.750 (± 0.069)**
Purity	0.860 (± 0.069)	0.890 (± 0.077)	0.855 (± 0.069)	**0.922 (± 0.078)**
SSQ	**8.782 (± 2.462)**	10.954 (± 3.437)	9.456 (± 3.297)	9.367 (± 2.461)
Silhouette	0.778 (± 0.110)	0.752 (± 0.138)	0. 759 (± 0.115)	**0.880 (± 0.094)**
Rand Index	0.892 (± 0.056)	0.787 (± 0.044)	0.880 (± 0.064)	**0.934 (± 0.043)**
Finding cluster number	5 (± 0)	5.76 (± 1.573)	5 (± 0)	4.540 (± 0.582)
Exact cluster number	5 (± 0)	5 (± 0)	5 (± 0)	5 (± 0)

Table 3. The results for scenario 2 (noisy data).

Scenario 2	CluStream	DenStream	ClusTree	ReHEvo
F1-P	0.641 (± 0.139)	0.389 (± 0.095)	0.643 (± 0.136)	**0.901 (± 0.076)**
F1-R	0.621 (± 0.101)	0.515 (± 0.066)	0.633 (± 0.096)	**0.734 (± 0.080)**
Purity	0.769 (± 0.067)	**0.906 (± 0.073)**	0.777 (± 0.066)	0.885 (± 0.095)
SSQ	18.205 (± 3.929)	**16.373 (± 4.630)**	17.432 (± 3.607)	22.056 (± 5.415)
Silhouette	0.649 (± 0.081)	0.662 (± 0.139)	0.650 (± 0.083)	**0.830 (± 0.105)**
Rand Index	0.828 (± 0.056)	0.687 (± 0.049)	0.831 (± 0.056)	**0.927 (± 0.040)**
Finding cluster number	5 (± 0)	8.664 (± 2.390)	5 (± 0)	4.425 (± 0.670)
Exact cluster number	5 (± 0)	5 (± 0)	5 (± 0)	5 (± 0)

Table 4. The results for scenario 3 (high dimensional data d=10).

Scenario 3	CluStream	DenStream	ClusTree	ReHEvo
F1-P	0.994 (± 0.029)	0.462 (± 0.101)	0.993 (± 0.038)	**0.981 (± 0.051)**
F1-R	0.814 (± 0.071)	0.533 (± 0.053)	0.772 (± 0.133)	**0.824 (± 0.028)**
Purity	**1 (± 0)**	**1 (± 0)**	**1 (± 0)**	**1 (± 0)**
SSQ	91.604 (± 17.573)	85.333 (± 9.686)	90.540 (± 18.069)	**83.780 (± 9.184)**
Silhouette	0.980 (± 0.001)	0.872 (± 0.066)	**0.981 (± 0.002)**	0.971 (± 0.027)
Rand Index	0.992 (± 0.033)	0.666 (± 0.050)	0.972 (± 0.062)	**0.992 (± 0.022)**
Finding cluster number	5 (± 0)	8.614 (± 1.767)	5 (± 0)	5.117 (± 0.322)
Exact cluster number	5 (± 0)	5 (± 0)	5 (± 0)	5 (± 0)

Table 5. The results for scenario 4 (variable cluster number).

Scenario 4	CluStream	DenStream	ClusTree	ReHEvo
F1-P	0.741 (± 0.143)	0.556 (± 0.136)	0.714 (± 0.153)	**0.885 (± 0.076)**
F1-R	0.700 (± 0.095)	0.583 (± 0.080)	0.685 (± 0.099)	**0.709 (± 0.089)**
Purity	0.857 (± 0.071)	0.903 (± 0.076)	0.850 (± 0.075)	**0.907 (± 0.075)**
SSQ	**8.942 (± 2.546)**	10.176 (± 3.517)	9.557 (± 3.039)	10.886(± 4.109)
Silhouette	0.731 (± 0.117)	0.694 (± 0.152)	0.711 (± 0.127)	**0.831 (± 0.118)**
Rand Index	0.883 (± 0.060)	0.766 (± 0.053)	0.873 (± 0.068)	**0.916 (± 0.047)**
Finding cluster number	5.244 (± 0.809)	6.844 (± 2.364)	5.244 (± 0.809)	4.704 (± 1.357)
Exact cluster number	5.244 (± 0.809)	5.244 (± 0.809)	5.244 (± 0.809)	5.244 (± 0.809)

Table 6. The results for scenario Forest Cover Type.

Forest Cover Type	CluStream	DenStream	ClusTree	ReHEvo
F1-P	0.072 (± 0.053)	0.015 (± 0.012)	0.076 (± 0.051)	**0.397 (± 0.107)**
F1-R	0.048 (± 0.037)	0.018 (± 0.018)	0.050 (± 0.036)	**0.296 (± 0.126)**
Purity	0.856 (± 0.191)	**0.981 (± 0.067)**	0.867 (± 0.160)	0.691 (± 0.134)
SSQ	129.18 (± 55.497)	**108.250 (± 51.117)**	130.823 (± 55.354)	118.592 (± 35.347)
Silhouette	**0.820 (± 0.158)**	0.738 (± 0.093)	0.828 (± 0.150)	0.670(± 0.134)
Rand Index	0.481 (± 0.115)	0.483 (± 0.126)	0.481 (± 0.115)	**0.507 (± 0.045)**
Finding cluster number	4.020 (± 1.727)	15.905 (± 7.664)	4.020 (± 1.727)	2.215 (± 0.717)
Exact cluster number	4.020 (± 1.727)	4.020 (± 1.727)	4.020 (± 1.727)	4.020 (± 1.727)

Table 7. The results for scenario KDD CUP'99.

KDD CUP'99	CluStream	DenStream	ClusTree	ReHEvo
F1-P	0.075 (± 0.233)	0.731 (± 0.389)	0.124 (± 0.304)	**0.804 (± 0.278)**
F1-R	0.070 (± 0.223)	0.802 (± 0.333)	0.119 (± 0.299)	**0.844 (± 0.238)**
Purity	0.251 (± 0.434)	**1 (± 0)**	0.318 (± 0.466)	0.982 (± 0.127)
SSQ	1126 (± 2595)	925 (± 2541)	1139 (± 2606)	**322 (± 779)**
Silhouette	0.613 (± 0.214)	0.962 (± 0.078)	0.648 (± 0.228)	**0.966 (± 0.085)**
Rand Index	0.943 (± 0.139)	0.887 (± 0.162)	0.939 (± 0.143)	**0.794 (± 0.214)**
Finding cluster number	1.289 (± 0.847)	4.113 (± 5.55)	1.289 (± 0.847)	7.268 (± 2.918)
Exact cluster number	1.289 (± 0.847)	1.289 (± 0.847)	1.289 (± 0.847)	1.289 (± 0.847)

Figures

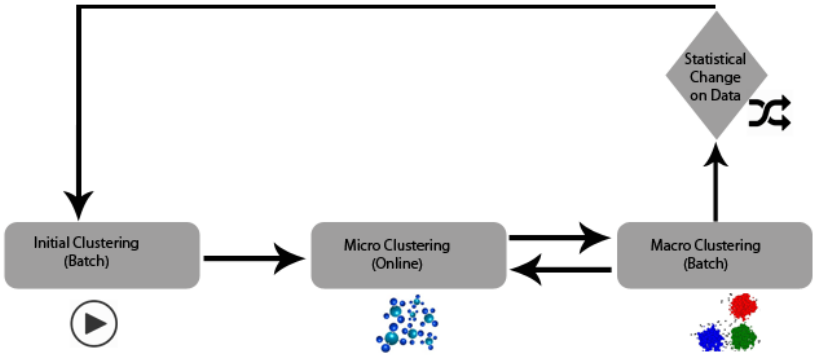

Figure 1. Stream clustering algorithms components.

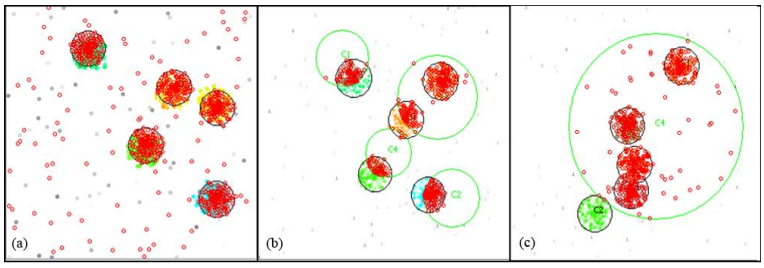

Figure 2. Cluster domination problem, (a), (b), (c) shows snapshots of system for time t1, t2 and t3 respectively where t1 < t2 < t3.

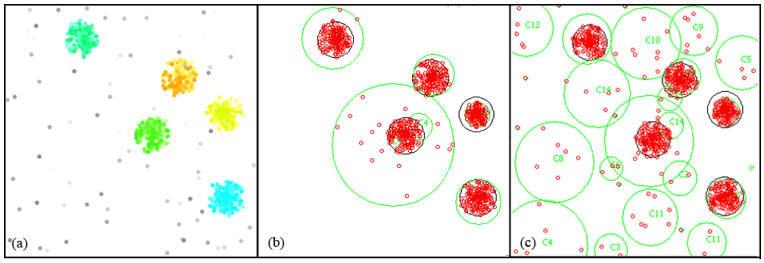

Figure 3. Overlapping and nesting problem, (a), (b), (c) shows snapshots of system for time t1, t2 and t3 respectively and t1 < t2 < t3.

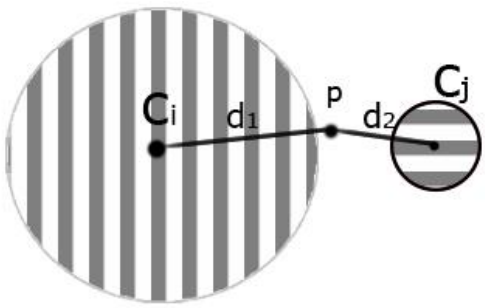

Figure 4. The center-surface paradox. $d_2 < d_1$ but probably p is an element of C_i

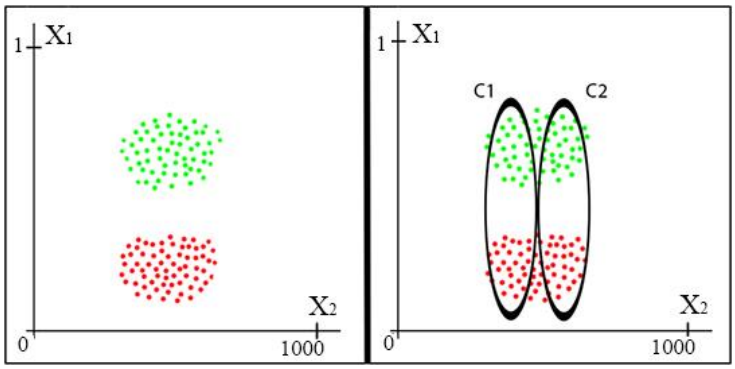

Figure 5. The problem of dominating other features.

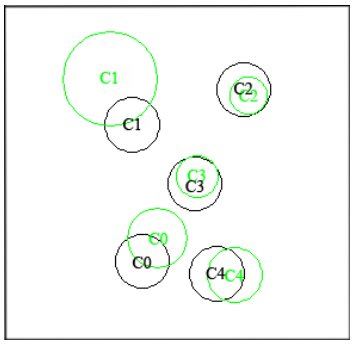

Figure 6. Cluster samples that have not forgotten their old data (especially ground truth C0 and C1).

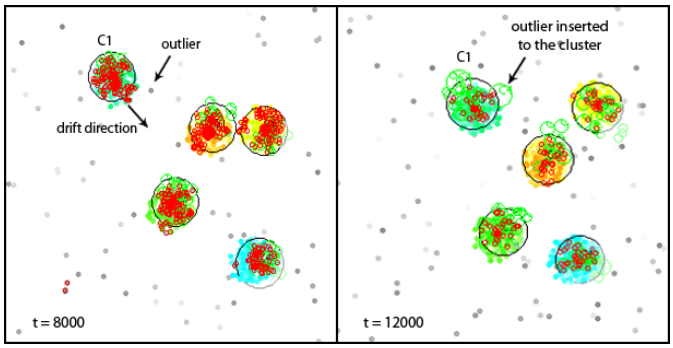

Figure 7. MOA simulation, a screenshot from DenStream algorithm.

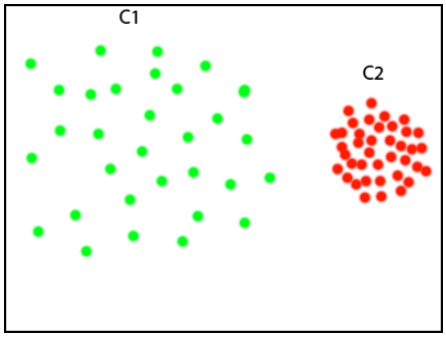

Figure 8. Cluster samples with different scales.

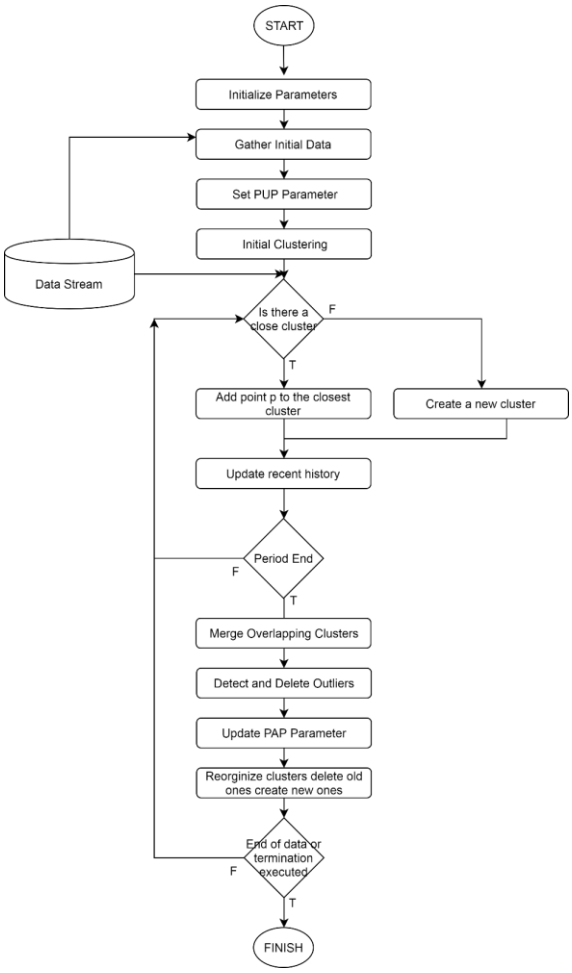

Figure 9. Flow chart of ReHEvo algorithm.

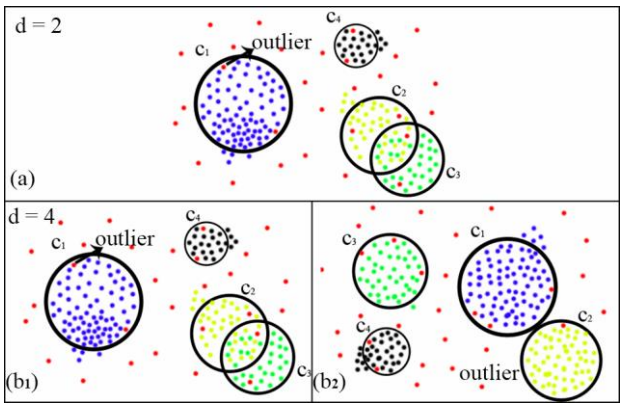

Figure 10. Comparison of 2-dimensional system and 4-dimensional system. (a) indicates the two dimensional data and overlapping some clusters. (b) indicates the four dimensional data (b1) indicates the first two dimensions of four dimensional system, and (b2) indicates the last two dimensions of four dimensional system.

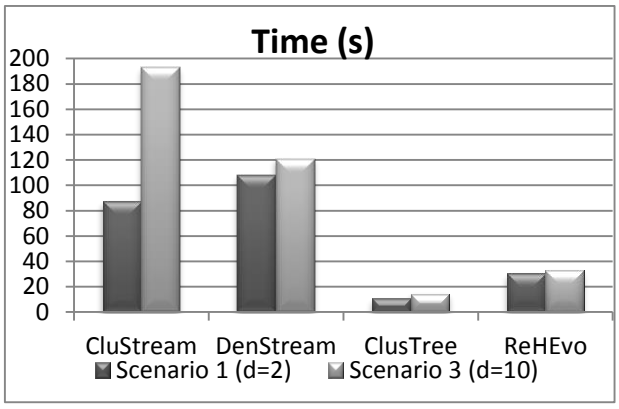

Figure 11. Run times of algorithms for scenario 1 and scenario 3.