Expert Systems With Applications 141 (2020) 112947

Contents lists available at ScienceDirect

Expert Systems With Applications

journal homepage: www.elsevier.com/locate/eswa

DGStream: High quality and efficiency stream clustering algorithm

Rowanda Ahmed [a,*], Gökhan Dalkılıç [b], Yusuf Erten [c]

[a] Computer Engineering Department, Izmir Institute of Technology, Urla, Izmir 35433, Turkey
[b] Computer Engineering Department, Dokuz Eylül University, Izmir 35390, Turkey.
[c] Computer Engineering Department, İzmir Institute of Technology and Bakırçay University, Izmir 35433 and 35665, Turkey

ARTICLE INFO

Article history:
Received 6 March 2019
Revised 9 September 2019
Accepted 9 September 2019
Available online 10 September 2019

Keywords:
Data streams architectures
Data stream mining
Grid-based clustering
Density-based clustering
Online clustering

ABSTRACT

Recently as applications produce overwhelming data streams, the need for strategies to analyze and cluster streaming data becomes an urgent and a crucial research area for knowledge discovery. The main objective and the key aim of data stream clustering is to gain insights into incoming data. Recognizing all probable patterns in this boundless data which arrives at varying speeds and structure and evolves over time, is very important in this analysis process. The existing data stream clustering strategies so far, all suffer from different limitations, like the inability to find the arbitrary shaped clusters and handling outliers in addition to requiring some parameter information for data processing. For fast, accurate, efficient and effective handling for all these challenges, we proposed DGStream, a new online-offline grid and density-based stream clustering algorithm. We conducted many experiments and evaluated the performance of DGStream over different simulated databases and for different parameter settings where a wide variety of concept drifts, novelty, evolving data, number and size of clusters and outlier detection are considered. Our algorithm is suitable for applications where the interest lies in the most recent information like stock market, or if the analysis of existing information is required as well as cases where both the old and the recent information are all equally important. The experiments, over the synthetic and real datasets, show that our proposed algorithm outperforms the other algorithms in efficiency.

1. Introduction

Clustering is a very important and crucial process in data mining especially due to the broad applicability of streaming data. Advances in the hardware technology and proliferated deployment of data-gathering devices such as sensors, large amounts of data are collected in a fast-growing rate. Analyzing and clustering these data streams, we need a strategy to constantly and periodically evaluate the data and present updated fresh results and views of the incoming records. Beyond the challenges static data clustering faces, stream data clustering has to cope with additional ones. Some of these challenges are (Ahmed, Dalkılıç, & Erten, 2018): single-pass processing of the endless data streams and, at the same time, extracting necessary information from the data to be used in the clustering process afterward. The limited time constraints should be observed and processing of every record should be able to keep up with the streaming speed. Limited memory is also an important parameter to be considered since the data is unbounded and it is not practical if the stream processing is done with buffering or storing an unbounded stream data. Data evolving continuously over time (Liu, Hou, & Yang, 2016), which is so common in today's applications is another challenge to be tackled. In addition to many other challenges are concept drift, varying time allowances, novelty, scalability, number of clusters, clusters' sizes, and outlier detection, also should all be considered. There are many data clustering methods for static data (Jain, Zhang, & Chang, 2006), and there are various methods for clustering stream data too. We can categorize the stream clustering methods into three types; prototype methods, density-based methods, and model-based methods (Alazeez, Jassim, & Du, 2017). The initial stream data clustering paradigms suffer from several limitations like buffering for later handling or dropping some data which lead to poor-quality clustering results. These approaches deal with the stream data clustering as static clustering but in a continuous version (Guha, Meyerson, Mishra, Motwani, & O'Callaghan, 2003). The evolving data are not taken into consideration in these paradigms and both recent and the outdated data are handled in the same way. Moving window is proposed to solve this problem (Barbará, 2002) to a certain extent. Other more recent stream clustering methods tried to solve some of these limitations and several algorithms are proposed to cluster the stream data, and we shall compare some of the density-based clustering ones such as the

* Corresponding author.
E-mail addresses: rowandaahmed@iyte.edu.tr (R. Ahmed), dalkilic@cs.deu.edu.tr (G. Dalkılıç), muraterten@iyte.edu.tr (Y. Erten).

https://doi.org/10.1016/j.eswa.2019.112947
0957-4174/

ones detailed in Aggarwal, Han, Wang, and Yu (2003); Cao, Estert, Qian, and Zhou (2006); Ruiz, Spiliopoulou, and Menasalvas (2010), and Kranen, Assent, Baldauf, and Seidl (2011) in the related work section.

The remainder of this paper is organized as follows: in Section 2, we shall present a quick overview of the algorithms for clustering static data, then we will go in more detail over some related work in the field of stream clustering. In Section 3, we will go over the most important performance metrics which we used and computed in Section 5 of this paper to compare our proposed method and other clustering methods' qualities and efficiencies. Section 4 describes our new proposed stream clustering method, DGStream, in detail. In Section 5, we will present the results of many experiments carried out on one synthetic and many real-world datasets and show their results along with many performance metrics of our proposed algorithm, and also present the comparison of the results with several outstanding stream clustering algorithms in terms of precision, recall, F1-score measure, clustering purity, and time complexity. Finally, Section 6 will conclude our paper.

2. Related work

There are a huge number of clustering algorithms both for static and stream data. Earlier stream data clustering algorithms are designed as continuous versions of the static ones. We shall, therefore, describe briefly the algorithms developed for clustering static data. We shall then continue with the others developed for clustering the stream data focusing on the density-based ones more than others. That is because density-based clustering, clustering which depends on density-connected points or employing density function, has many merits such as discovering the clusters of arbitrary shapes, handling noise, performing calculations without relying on too many parameters, and they can do it in just one scan. We will describe some of them in the following subsections.

2.1. Static clustering algorithms

Several density-based clustering algorithms are developed to cluster the static data like DBSCAN (Ester, Kriegel, Sander, Xu et al., 1996), DENCLUE (Hinneburg, Keim et al., 1998), OPTICS (Ankerst, Breunig, Kriegel, & Sander, 1999), and CLIQUE (Agrawal, Gehrke, Gunopulos, & Raghavan, 1998). We shall describe them in some detail below.

2.1.1. DBSCAN: Density-based spatial clustering of application with noise

DBSCAN is a density-based algorithm that focuses on clustering large and noisy spatial datasets. It performs a neighborhood density analysis according to two parameters, MinPts, and Eps, so a point which has in its Eps radius at least MinPts points, is classified as the core one. A point which does not qualify as the core point but exists in the neighborhood of one core point can be classified as the border point. Any point that is neither a core point nor a border point is a noise point. The core and border points are assigned to one cluster but the noise points are not. DBSCAN can discover not only the spherical clusters but also the clusters of interwoven arbitrary shapes due to the clusters growing according to a density-based connectivity analysis (Ester et al., 1996). DBSCAN has its limitations; DBSCAN classifies the dataset into two types of regions depending on a predefined threshold, the high density regions that are used in forming the final result of clustering as the cluster sets, and the low density regions that will be considered as noise. However, in some cases, many points identified as noises by DBSCAN may end up being meaningful data but with a low density that is under the threshold set. So, DBSCAN doesn't work well on datasets with varying densities and the high-dimensional ones.

2.1.2. DENCLUE: DENsity-based CLUstering

DENCLUE (Hinneburg et al., 1998) is based on a solid mathematical foundation; it is like DBSCAN as it is also based on neighborhoods analysis. It figures out how one data point in the dataset can affect its neighborhood. The summation of influences of all data points is the overall density of the data space. It computes the local maxima of the density function and identifies it as density attractors that are used to assign data points to the clusters. Objects belong to related or the same cluster depending on whether they are associated with related or the same density attractor. It has been designed for clustering the multimedia in high-dimensional spatial datasets and having large amounts of noise. DENCLUE is significantly faster than DBSCAN but it depends on a large number of parameters.

2.1.3. OPTICS: Ordering points to identify clustering structure

OPTICS (Ankerst et al., 1999) is a phase in the clustering process; it can identify the clustering structure. It orders the points and the reachability distances in a better way to be used by other density-based algorithms afterwards.

2.1.4. CLIQUE

CLIQUE is both a grid and a density-based clustering algorithm. It is designed for clustering high dimensional spatial datasets. It partitions the dimensions into grids, the dense grids that contain at least a threshold number of data points, and the non-dense grids. Then it tries to find the embedded clusters in subspaces of the dataset (Ruiz et al., 2010). However, the above-mentioned algorithms do not work when the data is a stream. They are applicable only to spatial datasets. As mentioned before, earlier data stream clustering algorithms have been developed to be the continuous versions of the static clustering algorithms. These approaches deal with the recent data and the outdated data in the same way. They do not consider the evolving data. Many techniques like moving window are proposed to partially solve this problem (Babcock, Datar, Motwani, & O'Callaghan, 2003; Barbará, 2002; Gama, 2010). Recently, many algorithms have been developed to cluster data streams. We intend to review in the following section some of them focusing on the density-based clustering approaches.

2.2. Stream clustering algorithms

Many clustering algorithms are developed recently to cluster the stream data. Many of these algorithms use two-component technique: online–offline. Generally, in the online phase, the algorithm captures necessary summary statistics of the incoming data records. The output from the online phase is the micro-clusters that will be used in the offline phase to derive the macro-clusters via re-clustering. In the offline phase, the stream algorithm employs one of the algorithms usually used for static data clustering like K-means or DBSCAN. Fig. 1 shows the idea of using the online phase to convert the stream data points to micro-clusters and the offline phase to convert the micro-clusters to macro-clusters

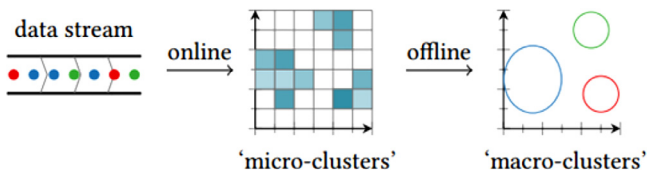


Fig. 1. Online-offline stream clustering process (Carnein et al., 2017).

(Ahmed et al., 2018; Amini, Saboohi, Ying Wah, & Herawan, 2014; Silva et al., 2013).

2.2.1. Denstream

DenStream clustering algorithm (Cao et al., 2006) has an online-offline framework. DenStream does not depend on many user-defined parameters like the number of clusters. Also, it can discover the arbitrary shaped clusters in addition to its capability of handling the outliers. Unlike other DBSCAN-based stream algorithms, DenStream is based on the idea of micro-clusters (Aggarwal et al., 2003) as neighborhood weighing areas instead of the neighborhood number of points in some radius. And, it has three types of micro-clusters; the first one is the dense one, named core-micro-cluster, it summarizes the arbitrary shape clusters. The second type is the potential core-micro-cluster; the weight of this micro-cluster is greater than a threshold value, and its weight decays over time using some decay function to give less importance to the outdated or old data records. The third type is the outlier micro-cluster; its weight is less than the threshold. When the outlier micro-cluster grows and its weight becomes above a predefined threshold, it is upgraded to become a potential one. DenStream applies a variant from the DBSCAN algorithm (Ester et al., 1996) in the offline phase to generate the final macro-clusters. When a new data record arrives, it is added to its closest potential core micro-cluster if its addition does not violate the radius constraint. If it does, it is added to the closest outlier-cluster if this can absorb the new point. Otherwise, a new cluster is initialized and marked as an outlier cluster. The most important advantage of DenStream is saving time since it does not merge data into a micro-cluster. Moreover, it can discover the clusters with arbitrary shapes, yet it can effectively recognize the potential clusters from the real outliers. On the other hand, it handles outliers with low accuracy, so removing them in the pruning phase is a time-consuming process (Thoriya & Shukla, 2015). Moreover, it does not delete or merge micro-clusters to release memory space. rDenStream algorithm is a version of DenStream which is developed to handle outlier's problem in a better way than the DenStream. However, rDenStream suffers from the memory complexity and the time complexity since it processes and saves the historical outlier buffer. Another limitation of DenStream algorithm is that it cannot be applied to applications in which the recent data distributions are the most important because it is not so well in tracking the cluster evolution.

2.2.2. DStream

DStream is a density-based grid structure clustering algorithm (Chen and Tu., 2007). The algorithm in its online phase maps each input point into a grid. And in its offline phase, it clusters the grids according to the density that is computed. DStream maintains the densities of all grid cells so that it can decay these densities overtime to capture the dynamic changes of the stream. Further, DStream removes sporadic grids, which improves the system time efficiency and reduces the memory requirements. When new input data record arrives, DStream maps this input data record into some density grid and updates its characteristic vector accordingly. And then, in every gap time, DStream detects and removes sporadic grids from the grid-list and adjusts the clustering depending on the neighboring dense cells. DStream has many advantages like adjusting the clusters in real-time; learning from data streams that drift over time; finding the interwoven and arbitrarily shaped clusters, detecting and handling the noises and outliers; and employing a decaying technique to deal with the evolving data streams (Thoriya & Shukla, 2015). And DStream has many shortcomings; like considering the minimum time interval gap while practically the algorithm depends on many interval gaps (Thoriya & Shukla, 2015). DStream's another limitation is the existence of those grids at the borders of the clusters, which influences memory especially when the algorithm works with a very high dimensional dataset; these grids need to be removed even if they are non-empty. DD-Stream (Jia, Tan, & Yong, 2008) that is a variation from DStream has been developed tosolve this limitation.

2.2.3. Clustree

ClusTree is a parameter-free algorithm, which handles streams adaptively according to the speed they arrive (Kranen et al., 2011). It proposes new strategies to deal with the clustering to improve its result in slow streams' cases, and it employs aggregation mechanisms to handle the fast streams. Additionally, it is the first algorithm that can present the result to the user at any time through maintaining a current clustering result over time, and it can update the final clustering result for the incoming data. It puts the stream points' ages into account; it gives more importance to more recent data by employing some decay function. When a new data point arrives, it is descended into the closest leaf as a new feature vector if it has empty places for new records. Otherwise, the leaf either splits into two nodes or the new incoming point is merged to the closest feature vector. Except for the leaf nodes, each node has a buffer to store other feature vectors temporarily. This offers a benefit when a new incoming record arrives while the others are descending down the tree, the descending one is stored temporarily in the buffer where it stays until a new data record descends to the same place in the tree, it then completes its descend down to the true leaf. ClusTree uses the leaf nodes to produce the final macro-clusters. ClusTree is the first algorithm proposed for the any-time merit, it is also parameter free, capable of detecting outliers, it adapts concept drift in the stream and it adapts itself to the stream speed automatically. However, in ClusTree, parameter selection is sometimes based on an exhaustive grid search that incorrectly clusters the dataset's samples and ends up with bad performance metric. The reported results always correspond to the best execution obtained in the grid search (Márquez, Otero, Félix, & García, 2018).

3. Performance metrics and basic definitions

We will be exposed to some of the terms over and over again in this paper. So, we shall first explain some theoretical notions by defining the concepts like the SPtree, Density Grids and the Characteristic Vector in the following Section 3.1. And after explaining the methodology of our proposed algorithm in details, to prove how well it performs we compared it with other related stream clustering algorithms. Regarding some performance metrics in the assessment process, so let's move over the used metrics in some brief in Section 3.2.

3.1. Basic definitions

Definition 1. SPtree: is a clustered multidimensional index structure called as the segment-page clustering (SP-clustering) for efficient sequential access. In our proposed algorithm, we used it to improve the query performance by continuous sorting the relevant points in contiguous related grids. Using SPTree in our density-based algorithm is important because dependency on density relies on the neighborhood relationships in growing clusters through the continuity of arriving data points, the connectedness of micro-clusters, and the convergence between them. Topological Spaces allows for the definition of concepts such as continuity, connectedness, and convergence, though accelerating and improving the clustering process afterwards.

Definition 2. Density grids: the grid contains many data records. Each record x inside the grid has its own density coefficient, and

this density coefficient decreases as the data record ages. To illustrate this concept by mathematical equations, we suppose that the data record x arrives at time t_x, so it's timestamp $T(x) = t_x$, and its density coefficient at this time is $D(x,t) = \lambda^{t-T(x)} = \lambda^{t-tc}$, where λ is the decay factor constant, t $\in$ (0, 1). Integrating from this point, so we can define the grid density at some time t as the whole summation of the density coefficients of all the records belonging to that grid. Let R(g,t) be the set of all data records belong to grid g at time t, so the density of g is $D(g,t) = \sum_{x \in R(g,t)} D(x,t)$.

Definition 3. Characteristic vector: it is a tuple with multiple information related to some grid density with the form (t, D, status, label), where t is the arrival time if there is no update since the last arrival time, and if there has been an update afterwards it will be the last updating time for the grid g. D is the last grid density. Status is either sparse grid or normal one, and label is the grid class label.

3.2. Performance metrics

Clustering is unsupervised learning; it is interested in dividing the data into similar groups in the absence of class labels in contrast to supervised learning where you have the data, the class labels, and the algorithm. Supervised learning just learns a function from the input. The absence of class labels in clustering makes the evaluation and the quality assessment more difficult and complicated than supervised classification. So, in clustering, to learn about the data helps to model its distribution and underlying structure. In this sub-section, we summarized the performance metrics we used to evaluate the results of our experiments in Section 5.

Performance metrics determine how good the obtained clustering reflects the actual data. Purity, precision, and Recall are examples of extrinsic methods where the ground truths are available. Precision-recall is a measure of how much the prediction is successful, especially in the case of very imbalanced classes. In information retrieval, precision measures the output relevancy, i.e. the fraction of retrieved relevant documents, but recall deals with the returned results and measures the fraction of the relevant documents that are successfully retrieved. F1-score can be defined as the mean or weighted average of recall and precision to evaluate an algorithm. F1-score provides a single measurement for a system and it reaches the best score at 1 and worst score at 0. Purity measures the extent to which clusters contains a single class (Manning, Raghavan, & Schütze, 2010). To calculate the purity for each cluster, find out the most common class in it and count the number of its data points. After that, compute the average of overall clusters. A perfect purity score of 1 can be reached by mapping each data record to its own class.

4. Our proposed algorithm methodology: DGStream

DGStream algorithm assumes special architecture to cluster such unlimited data records. Like most stream algorithms, DGStream also assumes a model with a discrete-time step model, where every incoming record is labeled by an integer timestamp 0, 1, 2... n. The timestamp indicates the record arrival time. As the online-offline approach has been integrated successfully with many stream clustering algorithms (Cao et al., 2006; Chen & Tu, 2007; Kranen et al., 2011), DGStream has an online-offline processing framework as well. In the online phase, it uses feature vectors represented by a micro-cluster for each grid to dynamically maintain the necessary information about the uninterrupted arriving data records. While in the offline phase, DGStream employs a DBSCAN algorithm to benefit from its speed and to improve the running time. And it depends on grids to reduce the time complexity and accelerates the speed once more (Alhanjouri & Ahmed, 2012; Mekky, 2016). DGStream also employs a decay function mechanism to accurately reflect the stream evolution process. In addition, it uses a mechanism to delete the sparse grids to maintain processing only with a limited number of dense grids, which saves both time and memory of the system. DGStream also employs a mechanism to get rid of the noise and to handle outliers. We will see all the steps that DGStream follows in the following subsections.

4.1. Dataset input and standardization

Standardization of the features of the dataset is a general requirement for many data mining algorithms. It aims to rescale the distribution of data values; i.e. make the data in the dataset dimensionless, though it helps in defining data in some standard indices. So, in our algorithm, it is necessary to standardize the datasets because we are going to calculate the similarity, dissimilarity and a number of associated performance metrics of the resulted clusters after the clustering process. Z-score and minimum-maximum (or normalization, or min-max scaling) are popular examples for standardizations. In implementing DGStream, we used min-max standardization that maps the minimum value to 0, and the maximum value to 1. This type of scaling gets the standard deviations smaller, which can reduce the effect of the outliers.

Since the stream data distribution is almost non-stationary, i.e. it changes over time, which is also known as concept drift; our algorithm detects and considers these changes through the damped window model. To deal with this phenomenon, DGStream assigns the most recent incoming data points to higher weights than the weights of the older points. These weights exponentially decrease as the time goes via an employed decaying function. The density-based algorithms in Cao et al. (2006); Chen and Tu (2007); Isaksson, Dunham, and Hahsler (2012) also adapted this model. Regarding non-stationary stream, DGStream decides to delete or create some grid according to the overall sum of all weights of data points in that grid. So that if one grid keeps receiving new data points the weight of the grid will be high because of the high weight of the new data. In case the grid does not receive any new data and its data points age over the time to become below some threshold, DGStream decides to delete this grid. In this way, DGStream's grids can be adapted to support the nonstationary data stream.

4.2. Divide the multi-dimensional data stream into grids

DGStream divides the multi-dimensional space of the input data into density grids; we used this technique because it is impractical to maintain all the raw data. These small grids each has its density which is associated with its data records counted in it (Alhanjouri & Ahmed, 2012). And after that, the clustering process keeps these density grids and deals with each grid as a local unit to output the final cluster set. Fig. 2 shows how the density grids can be used between the online and offline phases to cluster the data streams.

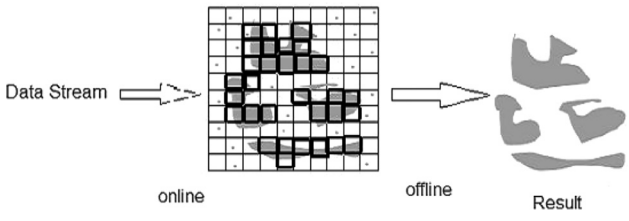


Fig. 2. Explanation art of using the density grids in stream clustering.

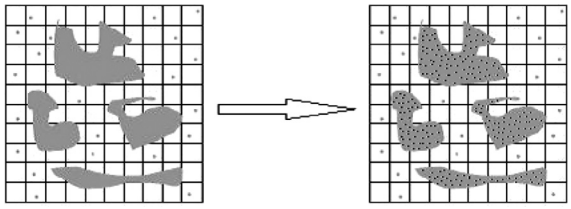


Fig. 3. Black points are the representative points in each cell.

4.3. Choosing representative points from the density grids

In the clustering step, instead of taking all the data points to process together, it is better to choose a set from them to represent the whole data stream we want to process. As in CURE clustering algorithm (Guha, Rastogi, & Shim, 2001), it adapts the idea of choosing points from each cluster which are well scattered and can represent the cluster. After this process, it shrinks them towards the mean of the cluster by some fraction to mitigate the outliers' effects. Using representative points in clustering helps in identifying both spherical and non-spherical clusters and speed up the clustering process. Therefore, DGStream uses the same principle of choosing well-scattered representative points to represent all the read time horizon bunch of objects such that the chosen representative points attempt to capture the physical shape and the geometry of the dataset. Choosing representative points instead of all the data points they represent in DGStream, provides many benefits. It saves execution time because this leads us to deal only with these representative points instead of all the data they represent. For example, in the case of computing the distance between two clusters, the only needed distance to compute is the distance between the closest pair of representative points from each cluster. It also saves the system memory because we need only to store the representative points as input to the clustering algorithm. In this regard, there are other techniques to do this like the constructions in De Silva and Carlsson (2004). For instance, the lazy-witness construction robustly computes topological invariants of geometric objects. It samples the dataset and uses only a comparatively small subset point cloud that can accurately capture the dataset shape. It firstly selects landmark points from the dataset randomly. On the other hand, for achieving more spaced points, it may select this subset by performing a sequential maximum-minimum selection such that selecting the point that maximizes the minimum distance to all the selected points chosen so far.

As we said; it is important for the chosen representative data points to capture the data stream from which they are chosen from, i.e. the original stream and the chosen representative set of points must have the same shape. It is clear from Fig. 3 that the black points that are used in the pre-clustering process are representing the input stream. Moreover, every time we read a number of examples from the incoming stream, according to the time horizon parameter, we choose well-scattered data points from the read data to represent it and continue repeating this process as the stream flow over time. The non-chosen data points from the stream will be labeled to the resulted clusters, as we will see later in this paper. This step benefits in giving our algorithm a good time improvement (Alhanjouri & Ahmed, 2012; Mekky, 2016), as depicted in the experimental results section.

4.4. DGStream clustering process

Fig. 4 outlines the overall DGStream algorithm. First, the algorithm reads a large number of normalized data points, and then it chooses a number of points to represent these. After that, DGStream can build the SP tree of density grids and computes the initial cluster set by clustering the tree leaves. Then, it updates the characteristic vectors of the clustered grids from their initial values. Depending on the values stored in the characteristic vectors, DGStream classifies the grids to dense and sparse ones. The other points which are not chosen are labeled to the output clusters using some strategy for labeling the points to the best clusters they can belong. After that, whenever some data record arrives, the online phase of DGStream reads and maps it to the most suitable density grid in the SP tree, and accordingly updates the grid's characteristic vector values. While in the offline phase, at every pre-specified gap time, DGStream computes the densities of the grids and checks out if there is any sparse grid upgrade to become dense or if there is any grid that must be marked as a sparse grid to be deleted afterwards. The cluster set is checked and corrected dynamically by calling MainClustering method indicated in Fig. 5.

```
1. procedure DGStream
2.    initialize an empty SPTree of grids;
3.    read huge number of records and map them to the SPTree.
4.    do MainClustering(SPTree)
5.     while data stream is active do
6.          read h number of records xi = (xi1, xi2, · · ·, xid); 1<i<h
7.          determine the density grid gi that contains xi; for all xi
8.          if (gi not in SPTree) insert gi to the SPTree
9.          update the characteristic vector of gi
10.         delete sparse grids from SPTree
11.         do MainClustering(SPTree)
12.    end while
13. end procedure
```

Fig. 4. DGStream algorithm pseudocode.

```
1. procedure MainClustering(SPTree)
2.    choose well scattered representative points that capture the
      shape of the original dataset grids;
3.    put the remaining grids in a labeling grid list for post clustering
4.    employ DBSCAN on the chosen SPTree grids to create clusters
5.    map the records in the labeling grid list to the outputted clusters
6.    do post processing
7. end procedure
```

Fig. 5. MainClustering method pseudocode in DGStream algorithm.

To hold the dynamic characteristics of data streams, DGStream algorithm progressively decreases the density for each dense grid over time if it does not receive any data records. This is an important stage since the dense grids may become sparse and vice versa. A sparse grid can be upgraded to become dense if new stream objects are mapped to it. That is why the algorithm must inspect the density for each grid and depending on that, it calls the MainClustering procedure at every gap time to adjust the final cluster set result (line 4 in the code in Fig. 4). The grid density is always changing. DGStream updates the grid's density only in the case the grid receives new input data records instead of updating all data records' weights and therefore the SP tree grids' characteristic vectors as well at each time step. The time of receiving the last data record, which is the time of updating the density of the grid which received that record, should be recorded to be the last update time of that grid which is considered when a new data record is mapped to the grid (Chen & Tu, 2007). Following this step saves $\theta(N)$ to $\theta(1)$ in running time, which means that it improves time efficiency since N, the number of grids, is large. Additionally, this leads to memory saving since there is no need to resave all the densities and all the corresponding timestamps of all records of the updated and not updated grids. What we need to save for each grid is the characteristic vector.

4.5. Removing sparse grids

The very high number of grids is a critical big challenge DGStream algorithm faces especially when the data has high-dimensions. And since most of the grids are either empty i.e. contains very few number of data records or do not receive stream data records for long periods, the number of these sparse grids increases extremely fast as the data stream flows in a high speed, which causes an overall system slowness. So, the solution is to detect the grids whose density become less than some specified threshold due to small data input and remove them afterwards. Only the dense grids taken into consideration in processing and storing. The other sparse grids are neglected and removed afterwards. After that, if one removed grid receives a number of records, it will be added back to the SPTree grids but with a zero density in a hope to be upgraded to a dense one.

4.6. Labeling all points to the resulted cluster set

As it has been mentioned before, the clustering process occurs only on the well-chosen data points from the incoming stream after each time horizon. So, now is the time to do the labeling step which works with the rest of the not-chosen stream data points in placing each to the existing point to the most suitable or similar macro-cluster in the resulted macro-clusters so far. Each data point is assigned to the macro-cluster that contains the closest representative point to this one. After doing the labeling step, all the stream data points will be allocated to macro-clusters. Additionally, there is a post-processing step that specialized in merging and deleting such macro-clusters. That is macro-clusters with the same density and close enough to each other if found, they will be merged in one macro-cluster in the post-processing step. In addition, when there is any macro-cluster whose weight is lower than some specified threshold value, it will be deleted from the cluster set and considered as an outlier.

4.7. Handling outliers

Generally, the datasets have outliers as a result of the problems that may be faced while entering data or errors in the measurement process. The distances between the outlier points and the nearest micro-clusters are high and more than the specified threshold in the DBSCAN offline algorithm. DGStream has its own strategy to deal with the outliers. It detects them and marks these points as outliers. After that, while the algorithm continues reading data points from the stream, and if some outliers near each other form such a dense grid with weight more than the specified threshold value, it is upgraded to become a new micro-cluster. Otherwise, if its weight becomes less and less due to the decay factor aging, and it becomes less than some threshold, DGStream safely deletes it without degrading the algorithm quality. Handling outliers is a very important step to finish the clustering process in the data stream clustering to save both time and space of the system.

4.8. DGStream clustering stability

It is attractive to use stability-based principles when we want to choose our models. Interestingly, it does not require a specific model to be applied to, but it can be applied to any clustering algorithm. One could intuitively assume that clustering stability is very much related to simple solutions that have the most stable parameters, but this is not necessarily true. Many studies show that the more complicated solutions can also be stable by choosing their parameters well, that it is needed to look at the theoretical results when deciding the stability-based model selection.

So, we can claim that algorithm A is stable if it almost surely outputs the same clustering result on a sample whose size approaches to infinity every time we run it. That is $\lim_{m\to\infty} Pr(A(W_m) = C_k)$; m is the sample size, W is the relative frequency, C_k is the k output clustering result. Then, we can measure the instability from $instability(A) := 1 - \lim_{m\to\infty} Pr(A(W_m) = C_k)$, which yields zero if algorithm A is stable. Instability of an algorithm is also obtained by computing the expected distance between two clustering's results on two different datasets of the same size (Von Luxburg et al., 2010) that is $instability(A) := E(distance(C_k(x_n), C_k(\hat{x}_n)))$.

When it comes to our proposed algorithm, stability of DGStream lies in its robustness against independent resampling, random fluctuations in the data, and the replacements of the subsamples. We achieve this by choosing the best combinations with right values for the parameters and so we can get good clustering results with the best stability and avoid wrong ones such as wrong split for at least one true cluster or wrong merge for at least two clusters. In more detail, in DGStream to evaluate the clustering stability, we need to run it several times on slightly different datasets. To achieve this, we need to generate a number of troubled versions of the dataset. These dataset versions are generated by subsampling or adding noise and outliers. In subsampling, we need to work with samples of different sizes. We drew such random noise-inlaid subsamples. In order not to lose the structure we want to discover by clustering with our algorithm, we must not change the samples too often. On the other hand, we might observe no significant stability results if the change in the dataset is not sufficient. So, it is a trade-off which we must cautiously deal with in all cases. Then, as usual, doing the dimensionality reduction to work with a low-dimension dataset is important. In this regard, DGStream doesn't commit any over-sensitive reactions to noise and outliers, which is considered as the most prominent factor in stumbling these bad results of splitting or merging clusters.

Let's compare our algorithm with the stable approach proposed by Carlsson and MÃŠmoli (2010) regarding clustering stability. Carlsson and Memoli's approach constructs a hierarchical relationship among data to do the clustering process. DGStream is a clustering algorithm based on density and grids which detects and handles the dense clusters in the dataset in a different way from Carlsson and Memoli's approach. Carlsson and Memoli's approach obtains an existence and uniqueness theorem instead of a non-existence result obtained by (Kleinberg, 2002) previously which tells that it is impossible for any standard clustering algorithm to simultaneously satisfy scale invariance, richness, and consistency. In Carlsson and Memoli's approach, the stability and convergence are established for a single linkage hierarchical clustering (SLHC) and that relaxes Kleinberg's impossibility result. Carlsson and Memoli's approach allows getting a hierarchical output from clustering methods, and then one can obtain uniqueness and existence. Carlsson and Memoli convergence results also refine the Hartigan's previous observation (Hartigan, 1985) regarding the underlying density. It does single linkage (SL) clustering of an independent, identically distributed (i.i.d.) samples from that density. The convergence results adopt general settings and it neither assumes such a smooth manifold underlying space nor assumes that the underlying probability measure must be with a density related to any reference measure. It does not matter how the points are distributed inside the space grids in the dataset. So, the SLHC is insensitive to variations in the density (Hartigan, 1981).

DGStream does care about how the data is distributed inside the space grids, the order of arrival of the records, time they arrived and enter the clustering process is important in DGStream because DGStream employs a decay factor which ages the points over time. The point may exist and may belong to some cluster in some time, while it does not exist later or may belong to

another cluster. It depends mainly on the timestamp of the point and its assigned weight and what happens to its weight by the decay factor as the time goes. So, DGStream is not an order invariant method; clustering a set of pints randomly in a different order can produce a different cluster set. In topology as well; the location of the point, to which grid it belongs is also of interest in DGStream. Therefore, the order of the records in the space grids also matters and that is what DGStream depends on while capturing the shape of the dataset, and deciding to merge these points together to form a cluster, and separating those points from those to form two or more clusters depending on the distribution of data points in the dataset. Moreover, at any time, DGStream can output a real-time result of the obtained cluster set up to that instant.

DGStream deals with weighted data points and hence weighted grids and these weights controlled by decay factor which ages the points and so the grids over time. This approach makes DGStream strong against random fluctuations in the data. DGStream takes notice of which data is outdated and deletes it. It is also aware of which grid at which time must be upgraded to become dense or downgraded to become a candidate to be deleted later. It is aware of clusters when they must be merged with another clusters or when one becomes necessary to be divided into two clusters. The number of clusters in DGStream is a parameter in a constant change with time in line with the shape of the data which is naturally in a constant change. Grids change due to their ages, which expose deleting some, while emerging others to address the evolving data over time better. DGStream is based on intuitive considerations to achieve good stability, and that is why it can be used in a wide range of practical real-life applications.

5. Experimental results

DGStream algorithm is an algorithm which combines quality and efficiency. We evaluated the quality and the efficiency of our proposed algorithm DGStream and compared it with DenStream (Cao et al., 2006), DStream (Chen & Tu, 2007), and ClusTree (Kranen et al., 2011). We mainly conducted our experiments and demonstrated their results on five datasets. One is a synthetic dataset, which is Chameleon dataset. And the others, KDDCup'99 (Hettich & Bay, 1999), Covertype (Blackard, Dean, & Anderson, 1998), Adult (Kohavi & Becker, 1996), and NSE Stocks (NSE, 2017) are real-world datasets. For both synthetic and real-world datasets, we focus on the numeric variables. So, for all datasets, we first standardize the features by minimum-maximum normalization. This means, the minimum value in one feature is mapped to 0 and the maximum value in it is mapped to 1. Note that this considerably improves the clustering result. The algorithms were implemented in Java programming language and the experiments were conducted on an Intel Core(TM) i7-4510U CPU @ 2.60GHz, 6.00GB RAM machine.

5.1. Chameleon synthetic dataset results

The Chameleon dataset is an important and famous synthetic dataset in data mining and machine learning field, and contains 8000 elements. In this study, we used the first and second numerical attributes of the dataset. The chameleon dataset is a complicated dataset with nested arbitrary shaped clusters, multi-dense clusters with a lot of noise (Alhanjouri & Ahmed, 2012; Mekky, 2016). Our experiments show good results in both the clustering quality and efficiency. For the same dataset, Chameleon, our proposed algorithm DGStream, gives a better result in quality by solving the overlapping problem between clusters and reduces the noise. Also, it catches the outlier points much better, and so, a more accurate shape of clusters appears. Applying DGStream on the synthetic dataset gives very good results that demonstrate how much our proposed algorithm solved the problems that CluStream and all other k-means based algorithms are suffering from. In our experiment with Chameleon dataset evaluation, we set the horizon length value h to 1000. Every time we read h samples from the stream, we update the current result of the cluster set with these new h samples and continue repeating this process. This process improves the quality of clustering over time (Cao et al., 2006; Hahsler & Bolaños, 2016). Fig. 6.a shows the original Chameleon dataset containing all the 8000 data records. Fig. 6b,c,d,e show the results for clustering the dataset with DenStream, DStream, ClusTree, and our proposed algorithm, DGStream respectively. These results show that our algorithm handles the outliers better with high accuracy and within lower time compared to the others. rDenStream (xiong Liu, fei Guo, Kang, & Huang, 2009) is an enhanced version from DenStream algorithm which handles the outliers as well but with high time complexity. So, our algorithm gave better quality results in determining the real clusters in the given dataset with a more appropriate output.

Table 1
Performance matrices for clustering 8000 data records from Chameleon synthetic dataset by using DenStream, DStream, ClusTree, and DGStream stream clustering algorithms.

	DenStream	DStream	ClusTree	DGStream
Time (ms)	8569	9389	6734	3238
F1-score	0.2964	1	0.92	0.9575
Purity	1	1	1	0.918
Precision	1	1	1	0.921
Recall	0.174	1	0.85	0.997

Table 1 shows the Chameleon synthetic dataset clustering performance metrics results of our proposed algorithm DGStream, along with other streaming algorithms. Performance metrics are time, purity, precision, recall, and F1-score. It is clear that DGStream algorithm and all other compared algorithms can perfectly determine the true classes. The purity values of all algorithms are approximately or almost exactly 1. That does not contradict the empirical study in Carnein, Assenmacher, and Trautmann (2017) for comparing the most important stream clustering algorithms which operate on t8.8k dataset, a similar synthetic dataset, to calculate the purity of the algorithms. We notice that the clustering output depends on the insertion order. Regarding recall, DGStream works well in retrieving almost all the relevant records to each cluster without lifting except a little. That is why DGStream's recall is better and almost outperforms all other algorithms. However, neither the precision nor the recall alone can measure the success of the prediction, especially in the case of very imbalanced classes like our dataset examples. Therefore, F1-score is the best to be calculated for the algorithm evaluation according to it is the harmonic mean or weighted average of recall and precision. It is clear that the score for DGStream is a little bit better than ClusTree's F1-score measure, but in DenStream case, it is much better, that is because the DenStream algorithm cannot retrieve all the required records, which resulted in its bad recall measure. Finally in the other important measure, which is the time; DGStream is remarkably faster than all other algorithms as shown in Table 1.

5.2. Real-world datasets results

We tested DGStream on three real-world datasets; KDDCup'99, Covertype, and Adult. Each dataset poses different challenges and different cluster shapes. The details are described in the following sub-subsections. Applying DGStream on the real-world datasets gives very good results, which indicates that our proposed

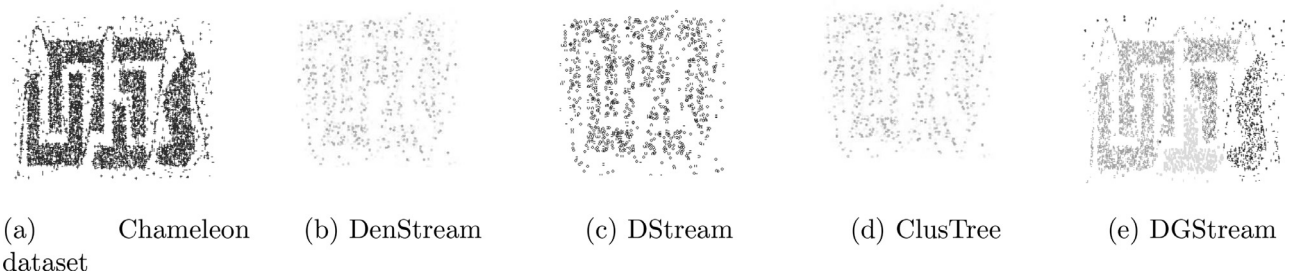


Fig. 6. Results for clustering 8000 points from Chameleon Synthetic dataset by DenStream, DStream, ClusTree, and DGStream stream clustering algorithms.

Fig. 7. Results for clustering 8000 points from KDDCup'99 real-world stream data by DenStream, DStream, ClusTree, and DGStream stream clustering algorithms.

Table 2
Performance matrices for clustering 8000 data records from KDDCup'99 real-world stream data by using DenStream, DStream, ClusTree, and DGStream stream clustering algorithms.

	DenStream	DStream	ClusTree	DGStream
Time (ms)	16513	15295	4458	1737
F1-score	0.969	0.9995	0.979	0.99066
Purity	1	1	1	0.9815
Precision	1	1	1	0.98147
Recall	0.966	0.999	0.96	1

Table 3
Performance matrices for clustering 20,000 data records from KDDCup'99 real-world stream data by using DenStream, DStream, ClusTree, and DGStream stream clustering algorithms.

	DenStream	DStream	ClusTree	DGStream
Time (ms)	9091	10917	6081	4570
F1-score	0.0498	1	0.95	0.9993
Purity	1	1	1	0.9987
Precision	1	1	1	0.9986
Recall	0.981	1	0.905	1

algorithm improves both quality and efficiency compared to all existing density-based stream clustering algorithms so far.

5.2.1. KDDCup'99 real-world dataset results

Among the most popular real-world datasets used for clustering data streams that we utilize is the KDDCup'99 dataset. This dataset contains 4,898,431 network traffic data records. Its attributes describe information about the connection such as the duration of the connection or the protocol type. And it's class label predicts if the connection was normal or attack, and there are 22 different attack types (Hettich & Bay, 1999).

We use the first and second numerical features of the data set, then we standardize the dataset according to the number of points we operate on. Firstly, we consider clustering the first 8000 observations from this dataset with a time horizon of 1000. Fig. 7a shows the first 8000 data records from the original KDDCup'99 real-world dataset. Fig. 7b,c,d,e show the results for clustering the same number of data records from the dataset with DenStream, DStream, ClusTree, and our algorithm, DGStream respectively. Here, too, we observed the same outcomes as in the previous experiment with the synthetic Chameleon dataset, that our algorithm is more successful in handling the outliers and with less time complexity than all other stream algorithms.

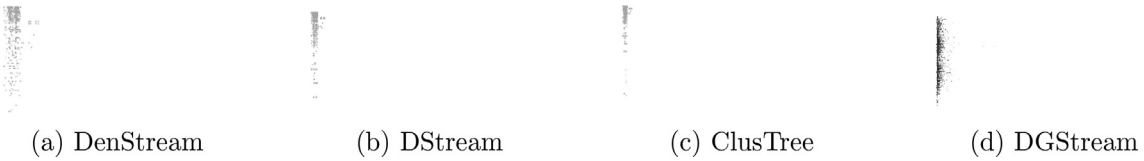
For this dataset, the clustering with DGStream gives good performance metric results as shown in Tables 2 and 3. All stream algorithms along with DGStream give very good purity results. For the F1-score, the same, all algorithms perform very well or near perfect results and that is due to the good measures for both precision and recall for all algorithms. About the time performance, our proposed algorithm, DGStream, is the best with much better than all other compared stream algorithms. The time for clustering 8000 points from KDDCup'99 dataset is 1737 ms. While it is 16513 ms, 15295 ms, and 4458 ms for DenStream, DStream, and ClusTree respectively.

Since this data contains more than just 8000 points, we clustered more than this number of points to test and compare the scalability of the stream clustering algorithms. Again, we repeated the above process with the first 20,000 observations, and the results are in the Fig. 8 and Table 3. All algorithms, in clustering 20,000 points from KDDCup'99, produce high purity clusters. As shown in Fig. 8 it is clear that DGStream is the best in outputting high accurate clustering results.

DGStream's precision and recall values are nearly perfect and so its F1-score. The same applies to the results of DStream algorithm. The average running times, in the case of 20,000 data records with the time horizon of value 1000 are 9091, 10917, 6081, and 4570 ms. for DenStream, DStream, ClusTree, and DGStream respectively are shown in Table 3., DGStream is the fastest algorithm when compared with the other stream clustering algorithms.

5.2.2. Covertype real-world dataset results

Real-world dataset, Covertype, appears to be a challenging one for most of the stream clustering algorithms. It contains about 581,012 data records where each record describes a defined area of forest. The information its attributes use to describe the area are such as the area slope, the area shade or its elevation, and a class label attribute that is a number from one to seven which shows the forest cover type. The US Forest Service (USFS) determined the forest cover types for the observations (Blackard et al., 1998). In this paper, we used the first and the third numerical attributes, then we standardized the dataset according to the number of points we operated on. Firstly, we consider clustering the first



(a) DenStream (b) DStream (c) ClusTree (d) DGStream

Fig. 8. Results for clustering 20,000 points from KDDCup'99 real-world stream data by DenStream, DStream, ClusTree, and DGStream stream clustering algorithms.

(a) Covertype dataset (b) DenStream (c) DStream (d) ClusTree (e) DGStream

Fig. 9. Results for clustering 8000 points from Covertype real-world stream data by using DenStream, DStream, ClusTree, and DGStream stream clustering algorithms.

(a) DenStream (b) DStream (c) ClusTree (d) DGStream

Fig. 10. Results for clustering 30,000 points from Covertype real-world stream data by using DenStream, DStream, ClusTree, and DGStream stream clustering algorithms.

Table 4
Performance matrices for clustering 8000 data records from Covertype real-world stream data by using DenStream, DStream, ClusTree, and DGStream stream clustering algorithms.

	DenStream	DStream	ClusTree	DGStream
Time (ms)	8883	9383	4058	1899
F1-score	0.3051	1	0.882	0.973
Purity	1	1	1	0.9688
Precision	1	1	1	0.9687
Recall	0.18	1	0.79	0.97713

Table 5
Performance matrices for clustering 30,000 from Covertype real-world stream data by DenStream, DStream, ClusTree, and DGStream stream clustering algorithms.

	DenStream	DStream	ClusTree	DGStream
Time (ms)	14423	9589	7974	6178
F1-score	0.6385	1	0.8833	0.9488
Purity	1	1	1	0.91
Precision	1	1	1	0.9098
Recall	0.469	1	0.791	0.9913

8000 observations from this dataset with a time horizon of 1000, in order to make a fair comparison between all stream algorithms on both synthetic and real-world datasets. Tables 4 and 5, along with Figs. 9 and 10 show the results for clustering 8000 and 30,000 data records from the dataset with DenStream, DStream, ClusTree, and our algorithm, DGStream respectively. It is clear that the DGStream clustering result is the highest quality compared with the other algorithms. It handles the outliers accurately and with high efficiency.

Table 4 shows the clustering results based on other performance metrics for Covertype dataset first 8000 observations. Most algorithms yield high purity after slowly increasing in purity to become perfect as the clusters adjust. F1-score in both DGStream and DStream are the highest due to the high value of their recall values, as F1-score depends on both precision and recall. While DenStream's F1-score is low depending on its recall measure, and in ClusTree case, F1-score is quite better also because its recall is better. To test and compare the running time efficiency, we run the experiments many times for each algorithm and then compute the average time consumed for each algorithm. We observed the best performance is for our proposed algorithm, DGStream, and it is much faster than all other stream algorithms. While DStream is the worse one, time performance is 9383 ms and ClusTree is better than DStream and DenStream, its time performance is 4058 ms. But ours is the best, its time performance is 1899 ms, as shown in Table 4.

Again, we repeated the above process with the first 30,000 observations, and the results are in the Fig. 10 and Table 5, DGStream is the most successful algorithm to capture the dataset shape and in handling the outliers. All algorithms produce high purity clusters. DStream's F1-score is perfect due to the perfect values of its precision and recall values. Our algorithm has the second-best F1-score because of its high precision and mostly perfect recall values. The average running times, in this case, are 14423, 9589, 7974 and 6178 ms for DenStream, DStream, ClusTree, and DGStream respectively. Therefore, DGStream is the fastest among all algorithms in clustering 30,000 data records from Covertype real-world dataset.

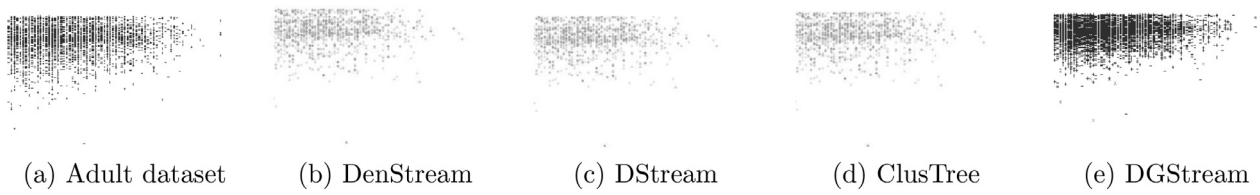


Fig. 11. Results for clustering 8000 points from Adult real-world stream data by using DenStream, DStream, ClusTree, and DGStream stream clustering algorithms.

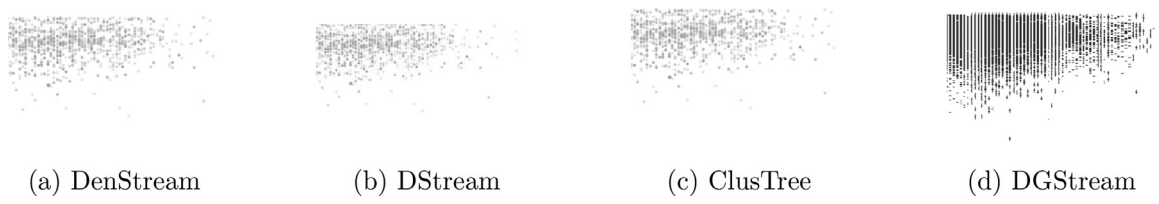


Fig. 12. Results for clustering 32,500 points from Adult real-world stream data by DenStream, DStream, ClusTree, and DGStream stream clustering algorithms.

Table 6
Performance matrices for clustering 8000 data records from Adult real-world stream data by using DenStream, DStream, ClusTree, and DGStream stream clustering algorithms.

	DenStream	DStream	ClusTree	DGStream
Time (ms)	11451	2922	5758	1661
F1-score	0.575	1	0.8538	0.99553
Purity	1	1	1	0.99115
Precision	1	1	1	0.9911
Recall	0.404	1	0.745	1

Table 7
Performance matrices for clustering 32,500 data records from Adult real-world stream data by using DenStream, DStream, ClusTree, and DGStream stream clustering algorithms.

	DenStream	DStream	ClusTree	DGStream
Time (ms)	49880	29723	11958	11615
F1 -score	0.685	1	0.848	0.9995
Purity	1	1	1	0.999
Precision	1	1	1	0.999
Recall	0.521	1	0.736	1

5.2.3. Adult real-world dataset results

The Adult real-world dataset, also known as "Census Income" (Kohavi & Becker, 1996), predicts whether the income exceeds 50*K*/*yr* or not. We use the first and third numerical features of the Adult real-world dataset. We standardize the dataset according to the number of points we operate on. Firstly and as we did with previously datasets all, we considered clustering the first 8000 observations from this dataset with a time horizon of 1000. Tables 6 and 7, along with Figs. 11 and 12 show the results for clustering the 8000 and 32,500 of data records from the dataset with DenStream, DStream, ClusTree, and our algorithm, DGStream respectively. We observed in the previous experiments with the synthetic Chameleon, real-world KDDCup'99 and real-world Covertype datasets, that our algorithm is better in handling the outliers with less time complexity than all other stream algorithms. Clustering with Adult confirms the same result. Therefore, DGStream algorithm is better in both quality and efficiency among the most important algorithms for clustering data streams.

For this data set, the clustering with DGStream gives good performance metric results as shown in Tables 6 and 7. Purity is perfect with all stream algorithms for both 8000 and 32,500 data records. For the F1-score, apart from the DenStream algorithm, all the other algorithms give very good results and that is due to the good outcomes for both precision and recall. DenStream's recall measure is bad and that is why its F1-score is poor. We can notice that our proposed algorithm's time performance is the best among all other compared stream algorithms. The average running time for clustering 8000 data points from Adult dataset is 1661 ms in DGStream algorithm. While it is 11451 ms, 2922 ms, and 5758 ms for DenStream, DStream, and ClusTree respectively as shown in Table 6. The average running time for clustering 32,500 data points from Adult dataset is 4885 ms for DGStream algorithm. While it is 11959 ms, 6237 ms, and 5206 ms for DenStream, DStream, and ClusTree respectively as shown in Table 7.

5.2.4. Stock marketing real-world dataset results

In this experiment, we chose clustering the NSE Stocks real-world dataset. It is the National Stock Exchange of India's stock listings for each trading day of 2016 and 2017. The data is compiled to facilitate machine learning tasks on stocks, without disturbing the Stock APIs. The data has been obtained from the NSE official site (NSE, 2017), *The National Stock Exchange of India Ltd.* Retrieved from https://www.nseindia.com/. In clustering open-ended data streams such as stock market data, it is important to capture temporal dependencies. While Bayesian networks (Buntine, 1991) and dependency networks (Heckerman, Chickering, Meek, Rounthwaite, & Kadie, 2000) model the dependencies of variables, Dynamic Bayesian Networks model discrete time temporal dependencies (Dean & Kanazawa, 1988; Friedman, Murphy, & Russell, 1998). However, in our stream clustering, we want to model the continuous data record timestamps, that is the arrival times of data records. Therefore, sampling is a solution we can apply on continues variable in order to use such a technique. Nevertheless, the sampling rate would have to be determined. Slow sampling ends up with poor data representation, and fast sampling leads to a need for multiple steps of past dependence with costly clustering of the stream (Gunawardana, Meek, & Xu, 2011). Continuous-Time Noisy-Or (Simma et al., 2008), Continuous Time Bayesian Networks (Nodelman, Shelton, & Koller, 2002; 2012), Poisson Networks (Rajaram, Graepel, & Herbrich, 2005; Truccolo, Eden, Fellows, Donoghue, & Brown, 2005), and Poisson Cascades (Simma & Jordan, 2010), are such recent solutions proposed for this problem. In clustering this real-world dataset, NSE Stocks, we used the numerical features; "OPEN" which is the opening market price of the equity symbol on the date, and the "TOTTRDQTY" which is the total traded quantity of the equity symbol. We standardize the dataset according to the number of points we operate on.

In this context, we try to discover the temporal dependencies and relations between intervals in NSE open-ended data stream states. DGStream learns how the recently arrived records affect

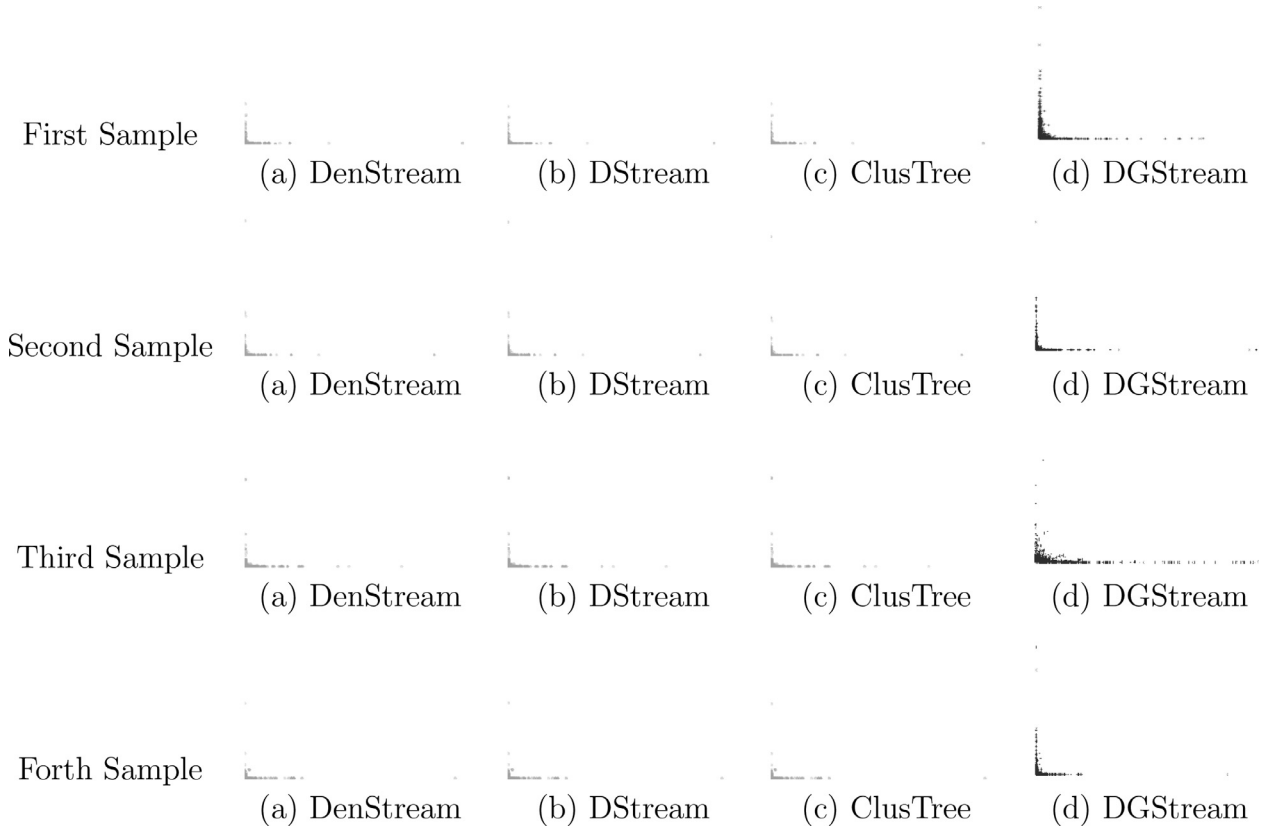


Fig. 13. Results for clustering same size of different samples from NSE real-world stream data without replacement by using DenStream, DStream, ClusTree, and DGStream stream clustering algorithms.

the currently arriving ones and the near future arriving as well. Stock markets' consecutive data is all related and depends on each other. We have conducted experiments with a sampling that can benefit the stability setting. Therefore, we generated such perturbed versions of many samples from this dataset without replacement, and then added noise to these samples. Then by applying DGStream several times on them, we find that the most meaningful one is the seasonal sampling. The depicted clustering results for clustering the different same size samples from this real-world stream data by DGStream, and comparison with DenStream, DStream, and ClusTree clustering algorithms are shown in Fig. 13. The average values of some performance matrices for all these experiments and comparisons are all in Table 8. In every sample, we cluster 10,000 observations from the dataset with a time horizon of 1000. In this sampling, the results show how the stock records in the past affects related future stock records based on their types, daily prices and arrival time, and so they emerged in related micro-clusters and also the same macro-clusters. It is clear from the results in Table 8 that evaluating purity, precision, recall, and F1-score in all experiments, DGStream is still the top first or second most of the time. We again repeated the same sampling steps with replacement and applied our algorithm and the other compared algorithms several times on 40,000 data points from NSE dataset. In this experiment, we measured and compared the time efficiency of our algorithm with the other algorithms. Fig. 14 shows that DGStream is the best followed by ClusTree algorithm. It is clear from Fig. 14 that DGStream can identify the clusters better. It gives the best results in clustering quality and handling outliers compared to other algorithms. In all previous experiments, we observed with both the synthetic and real-world datasets, that our algorithm outperforms all other algorithms in handling the outliers, learning the underlying dependency structure of data records, time performance, and equality. Applying the clustering algorithms to NSE dataset verifies the same claim that our proposed algorithm DGStream is the best in both

Table 8
Performance matrices for clustering same size of different samples from NSE real-world stream data without replacement by DenStream, DStream, ClusTree, and DGStream stream clustering algorithms.

		DenStream	DStream	ClusTree	DGStream
Time (ms)	Sample 1	34,210	33,896	7053	3430
	Sample 2	35,283	36,599	8938	3375
	Sample 3	57,484	9500	5723	3287
	Sample 4	35,574	39,461	6825	3464
F1-score	Sample 1	0.97	1	0.98	0.9993
	Sample 2	0.97	1	0.98	1
	Sample 3	0.98	1	0.98	1
	Sample 4	0.98	1	0.98	1
Purity	Sample 1	1	1	1	0.9989
	Sample 2	1	1	1	1
	Sample 3	1	1	1	1
	Sample 4	1	1	1	0.9996
Precision	Sample 1	1	1	1	0.9994
	Sample 2	1	1	1	1
	Sample 3	1	1	1	0.9996
	Sample 4	1	1	1	1
Recall	Sample 1	0.95	1	0.95	0.9992
	Sample 2	0.95	1	0.96	0.9997
	Sample 3	0.98	1	0.96	1
	Sample 4	0.96	1	0.96	0.9989

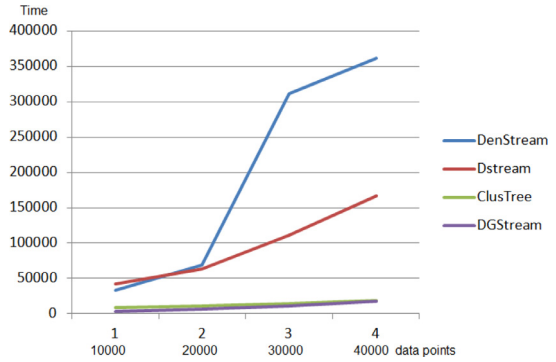


Fig. 14. Time efficiency for clustering different sizes samples from NSE real-world stream data with replacement by using DenStream, DStream, ClusTree, and DGStream stream clustering algorithms.

quality and efficiency among the most important stream clustering algorithms.

6. Conclusions and future work

6.1. Conclusions

In this study, we proposed a new stream clustering algorithm, DGStream. It is a density and grid-based algorithm with insightful implications for clustering stream data. DGStream algorithm has been tested and compared over many datasets, both synthetic and real-world datasets and under different scales and compared with DenStream, DStream, and ClusTree stream algorithms in the same field. So, experimentally under the same conditions and datasets, we have demonstrated that DGStream outperforms several well-known density-based stream clustering algorithms. It can find datasets with clusters of arbitrary shapes, multi-density and without the prior knowledge of parameters like the number of clusters. It is shown by many experiments that our proposed algorithm is significantly fastest among all the compared algorithms; it achieves the best time efficiency along with the best quality. Its recall measurement is always high due to its ability in assigning almost all relevant records to the corresponding correct clusters with, to a large extent, perfect purity, which means that our algorithm can create clusters much close to the true structure of the stream data. DGStream has also many good features; it is a strong and robust algorithm to noise and presence of outliers; needs only one-pass for processing stream data; it considers the evolving data by employing a decaying function that decreases the weights of the outdated data over time. Therefore, it is suitable for real-world applications where the most interest is in the recent information while the old information decreases over time like stock marketing. From all conducted experiments, we can say that our proposed algorithm outperformed all other density-based stream clustering algorithms in both efficiency and accuracy. However, we have to realize that stream clustering algorithms cluster streaming data from different points of view, and choosing between them depends on what we want to achieve from applying them, such as more accuracy is better than more reliability in some instances, and sometimes for particular application low time complexity is the most important property. Therefore, we can say for sure that for some applications or for a particular dataset and under specific conditions there is an algorithm that is much better than one another.

6.2. Future work

As future work, we will look at ways to detect dense grids so then to tune the parameters according to how much the density is and how many grids in the space have this density. In addition, we can go in more deeply work with adapting parameters to be more dynamic. Another improvement may be entering the prediction factor and trying to predict which grids might be useless in the future, depending on how the incoming stream points map to grids and in this aspect, we can focus more at the grids at the borders of the clusters.

Declaration of competing interest

None.

Credit authorship contribution statement

Rowanda Ahmed: Visualization, Methodology, Data curation, Writing - original draft, Software, Validation. **Gökhan Dalkılıç:** Supervision, Project administration, Writing - review & editing, Conceptualization, Investigation. **Yusuf Erten:** Supervision, Project administration, Writing - review & editing.

References

Aggarwal, C. C., Han, J., Wang, J., & Yu, P. S. (2003). A framework for clustering evolving data streams. In *Proceedings of the 29th international conference on very large data bases-volume 29* (pp. 81–92). VLDB Endowment.

Agrawal, R., Gehrke, J., Gunopulos, D., & Raghavan, P. (1998). Automatic subspace clustering of high dimensional data for data mining applications. *ACM, 27*(2), 94–105.

Ahmed, R. D., Dalkılıç, G., & Erten, M. (2018). *Survey: Running and comparing stream clustering algorithms*. CEUR Workshop Proceedings.

Alazeez, A. A. A., Jassim, S., & Du, H. (2017). EINCKM: An enhanced prototype-based method for clustering evolving data streams in big data. In *Proceedings of the ICPRAM* (pp. 173–183).

Alhanjouri, M. A., & Ahmed, R. D. (2012). New Density-Based Clustering Technique: GMDBSCAN-UR. *International Journal of Advanced Research in Computer Science, 3*(1). http://hdl.handle.net/20.500.12358/24451.

Amini, A., Saboohi, H., Ying Wah, T., & Herawan, T. (2014). A fast density-based clustering algorithm for real-time internet of things stream. *The Scientific World Journal, 2014*, 926020.

Ankerst, M., Breunig, M. M., Kriegel, H.-P., & Sander, J. (1999). Optics: ordering points to identify the clustering structure. In *Proceedings of the ACM SIGMOD record: 28* (pp. 49–60). ACM.

Babcock, B., Datar, M., Motwani, R., & O'Callaghan, L. (2003). Maintaining variance and k-medians over data stream windows. In *Proceedings of the twenty-second ACM SIGMOD-SIGACT-SIGART symposium on principles of database systems* (pp. 234–243). ACM.

Barbará, D. (2002). Requirements for clustering data streams. *ACM SIGKDD Explorations Newsletter, 3*(2), 23–27.

Blackard, J. A., Dean, D. J., & Anderson, C. (1998). *The forest covertype dataset* https://archive.ics.uci.edu/ml/datasets/covertype.

Buntine, W. (1991). Theory refinement on Bayesian networks. In *Proceedings of the seventh conference on uncertainty in artificial intelligence* (pp. 52–60). Morgan Kaufmann Publishers Inc.

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

Carlsson, G., & MÃŠmoli, F. (2010). Characterization, stability and convergence of hierarchical clustering methods. *Journal of Machine Learning Research, 11*(Apr), 1425–1470.

Carnein, M., Assenmacher, D., & Trautmann, H. (2017). An empirical comparison of stream clustering algorithms. In *Proceedings of the computing frontiers conference* (pp. 361–366). ACM.

Chen, Y., & Tu, L. (2007). Density-based clustering for real-time stream data. In *Proceedings of the 13th ACM SIGKDD international conference on knowledge discovery and data mining* (pp. 133–142). ACM.

De Silva, V., & Carlsson, G. E. (2004). Topological estimation using witness complexes. *SPBG, 4*, 157–166.

Dean, T. L., & Kanazawa, K. (1988). Probabilistic temporal reasoning. In *Proceedings of the AAAI* (pp. 524–529).

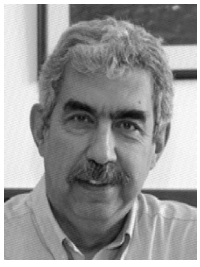
Ester, M., Kriegel, H.-P., Sander, J., Xu, X., et al. (1996). A density-based algorithm for discovering clusters in large spatial databases with noise. In *Proceedings of the KDD: 96* (pp. 226–231).

Friedman, N., Murphy, K., & Russell, S. (1998). Learning the structure of dynamic probabilistic networks. In *Proceedings of the fourteenth conference on uncertainty in artificial intelligence* (pp. 139–147). Morgan Kaufmann Publishers Inc.

Gama, J. (2010). *Knowledge discovery from data streams*. Chapman and Hall/CRC.
Guha, S., Meyerson, A., Mishra, N., Motwani, R., & O'Callaghan, L. (2003). Clustering data streams: Theory and practice. *IEEE Transactions on Knowledge and Data Engineering, 15*(3), 515–528.
Guha, S., Rastogi, R., & Shim, K. (2001). Cure: An efficient clustering algorithm for large databases. *Information Systems, 26*(1), 35–58.
Gunawardana, A., Meek, C., & Xu, P. (2011). A model for temporal dependencies in event streams. In *Proceedings of the advances in neural information processing systems* (pp. 1962–1970).
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.
Hartigan, J. A. (1981). Consistency of single linkage for high-density clusters. *Journal of the American Statistical Association, 76*(374), 388–394.
Hartigan, J. A. (1985). Statistical theory in clustering. *Journal of Classification, 2*(1), 63–76.
Heckerman, D., Chickering, D. M., Meek, C., Rounthwaite, R., & Kadie, C. (2000). Dependency networks for inference, collaborative filtering, and data visualization. *Journal of Machine Learning Research, 1*(Oct), 49–75.
Hettich, S., & Bay, S. (1999). The UCI KDD archive *Department of Information and Computer Science, University of California, CA, 152* [http://kdd.ics.uci.edu]. irvine.
Hinneburg, A., Keim, D. A., et al. (1998). An efficient approach to clustering in large multimedia databases with noise. In *Proceedings of the KDD: 98* (pp. 58–65).
Isaksson, C., Dunham, M. H., & Hahsler, M. (2012). Sostream: Self organizing density-based clustering over data stream. In *Proceedings of the international workshop on machine learning and data mining in pattern recognition* (pp. 264–278). Springer.
Jain, A., Zhang, Z., & Chang, E. Y. (2006). Adaptive non-linear clustering in data streams. In *Proceedings of the 15th ACM international conference on information and knowledge management* (pp. 122–131). ACM.
Jia, C., Tan, C., & Yong, A. (2008). A grid and density-based clustering algorithm for processing data stream. In *Proceedings of the second international conference on genetic and evolutionary computing* (pp. 517–521). IEEE.
Kleinberg, J. M. (2002). An impossibility theorem for clustering. In S. Becker, S. Thrun, & K. Obermayer (Eds.), *NIPS* (pp. 446–453). MIT Press, ISBN 0-262-02550-7.
Kohavi, R., & Becker, B. (1996). Adult dataset[online] available: http://archive.ics.uci.edu/ml/datasets.
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.
Liu, H., Hou, X., & Yang, Z. (2016). Design of intrusion detection system based on improved k-means algorithm. *Computer Technology and Development, 1*, 101–105.
Manning, C., Raghavan, P., & Schütze, H. (2010). Introduction to information retrieval. *Natural Language Engineering, 16*(1), 100–103.
Márquez, D. G., Otero, A., Félix, P., & García, C. A. (2018). A novel and simple strategy for evolving prototype based clustering. *Pattern Recognition, 82*, 16–30.
Mekky, A. R. (2016). Fuzzy neighborhood grid-based DBSCAN using representative points. *Feature Engineering in Hybrid Recommender Systems, Third International Conference on Data Mining, Internet Computing, and Big Data, July 21–23* (pp. 63–73). Konya, Turkey.
Nodelman, U., Shelton, C. R., & Koller, D. (2002). Continuous time Bayesian networks. In *Proceedings of the eighteenth conference on uncertainty in artificial intelligence* (pp. 378–387). Alberta, Canada: Morgan Kaufmann Publishers Inc. August 01–04.
Nodelman, U., Shelton, C. R., & Koller, D. (2012). Expectation maximization and complex duration distributions for continuous time Bayesian networks. arXiv:1207.1402.
Rajaram, S., Graepel, T., & Herbrich, R. (2005). Poisson-networks: A model for structured point processes. In *Proceedings of the 10th international workshop on artificial intelligence and statistics* (pp. 277–284). Citeseer.
Ruiz, C., Spiliopoulou, M., & Menasalvas, E. (2010). Density-based semi-supervised clustering. *Data Mining and Knowledge Discovery, 21*(3), 345–370.
Silva, J. A., Faria, E. R., Barros, R. C., Hruschka, E. R., De Carvalho, A. C., & Gama, J. (2013). Data stream clustering: A survey. *ACM Computing Surveys (CSUR), 46*(1), 13.
Simma, A., Goldszmidt, M., MacCormick, J., Barham, P., Black, R., Isaacs, R., & Mortier, R. (2008). Ct-nor: representing and reasoning about events in continuous time. In *UAI'08 Proceedings of the Twenty-Fourth Conference on Uncertainty in Artificial Intelligence* (pp. 484–493). Helsinki, Finland. July 9–12.
Simma, A., & Jordan, M. I. (2010). Modeling events with cascades of poisson processes. In *UAI'10 Proceedings of the Twenty-Sixth Conference on Uncertainty in Artificial Intelligence* (pp. 546–555). Catalina Island, CA. July 8–11.
Thoriya, D., & Shukla, M. (2015). Study of density based clustering techniques on data streams. *International Journal of Engineering Research and Application (IJERA), 5*(2), 40–47.
Truccolo, W., Eden, U. T., Fellows, M. R., Donoghue, J. P., & Brown, E. N. (2005). A point process framework for relating neural spiking activity to spiking history, neural ensemble, and extrinsic covariate effects. *Journal of Neurophysiology, 93*(2), 1074–1089.
Von Luxburg, U., et al. (2010). Clustering stability: An overview. *Foundations and Trends® in Machine Learning, 2*(3), 235–274.
xiong Liu, L., fei Guo, Y., Kang, J., & Huang, H. (2009). A three-step clustering algorithm over an evolving data stream, *1*, 160–164.

Rowanda Ahmed received B.S. and M.S. degrees both in Computer Engineering from Islamic University, Gaza, Palestine, in 2011. She has worked at various universities as a teacher assistant or as an instructor and in two ministries in Palestine as an engineer. Currently, she is a Ph.D. student at the Izmir Institute of Technology. Her fields of studies are data mining and data stream clustering. She has four papers published. Her research interest areas are machine learning, data science, big data, natural language processing, and deep learning.

Gökhan Dalkılıç received B.S. degree in Computer Engineering from Ege University, Izmir, Turkey, in 1997, M.S. degrees in Computer Science from University of Southern California, Los Angeles, USA, in 1999, and from Ege University International Computing Institute, Izmir, Turkey, in 2001, and Ph.D. degree in Computer Engineering from Dokuz Eylül University, Izmir, Turkey, in 2004. He had been a visiting lecturer in University of Central Florida, Orlando, and the USA from January 2003 to December 2003. He has been an Associate Professor of the Department of Computer Engineering of Dokuz Eylül University, Izmir, Turkey. His research areas are cryptography, statistical language processing, and computer networks. His fields of studies are lightweight authentication, cryptography, and NLP. He has over 50 papers and 4 books to his name.



Yusuf Erten received B.S. degree in Electronics Engineering from University of Birmingham, England, MS degrees in Computer Science from Missouri Institute of Technology, the USA, and the Ph.D. degree in Computer Engineering from Middle East Technical University, Ankara, Turkey. He has worked at various national and international companies in the ICT sector as an engineer, and at different universities as a faculty member for many years. Currently, he is a faculty member at Izmir Bakıçay University. His research interests are computer networks, security, and reliability.