INTERNATIONAL JOURNAL OF MERGING TECHNOLOGY AND ADVANCED RESEARCH IN COMPUTING

IJMTARC – VOLUME – V – ISSUE – 22, APR - JUNE, 2018

ISSN: 2320-1363

CLUSTERING DATA STREAMS USING SHARED DENSITY BETWEEN MICRO-CLUSTERS

SRINIVASARAO PAMIDI[1], N.NAVEEN KUMAR[2]

[1]M.Tech Student, Dept of CSE, NALANDA INSTITUTE OF ENGINEERING TECHNOLOGY, AP

[2]Associate professor, Dept of CSE, NALANDA INSTITUTE OF ENGINEERING TECHNOLOGY, AP

Abstract-

As we taken micro cluster in streaming data, cluster data streaminghas been become an important technique, which used for data and knowledge technology. A generally we approach is to describing the data stream in Problem solving-time with an onlineprocess into a huge number of so called micro clusters. Micro clusters represent provincial density estimates by aggregating the information of every data points in a defined particular area. On Requirement, a converted using conventional clustering algorithm is used at in a second offline step by step to re-cluster theMicroclusters into huge final clusters. For re-clustering, the midpoint of the micro clusterissuing as pseudo points are with the density assessment used as their converted weights. After all, contain information Regarding density in the area between microclusters is not conserve in the online process and using re-clustering is established on possibly wrong assumptions about the sharing of data within process between microclusters. This paper describes DATABSAESTREAM, the first step micro clusterbased on online clustering basic that especially captures the density between this microclusters along a shared density graph. The density information using in this graph is then shows for re-clustering based on which actual density between two adjacent micro clusters. We present that the space and time complexity of maintains using the shared density graph. Experiment on a broad range of fabricatedand data sets highlight particular that using shared density improves is clustering quality up to other popular data stream clustering many methods which require the formation of a huger number of mini micro-clusters to produce comparable results over here.

Index Terms— DATABASESTREAM, Data mining,density-based clustering, data stream clustering.

I. INTRODUCTION

A RE-CLUSTERING data stream [2], [3], [5], [7] has become auseful technique for data and knowledge technology information. A data stream is an arranged and potentiallyunbounded arrangement of data points using. Such Data streams ofarriving data are developed for every type of techniqueand adding GPS data from latest smart phones, webstream data, computer network operating data, telecommunication connection between data, and reading from wireless sensor networks.Data stream clustering is normally done as a two-step process withonline step which summary the data into every micro clusters and grid cells then, other step in an offline process,these all micro-clusterare re-clustered or merged into a less number of final clusters. Thereclustering herefor an offline process and which thus not time critical, it is generally not presented in detail in paper about new data stream clustering algorithms. Most papers suggest touse anexisting conventionalis clustering algorithm, wherethemicroclustersareusedaspseudopoints in cluster.Other approach used in Den Stream [4] is to use reachablewhere all micro clusters areless than a provided distance from each other, linked balanced to form particular clusters. Grid-based algorithms merge two adjacent dense grid cells to form many clustersCurrent re-clustering approaches completely ignore everydata density in the area between the micro-clusters and thus might join micro clusters which are closetogetherbutatthesametimeseparatedbyasmallarea of very low density. To locate this problem, introduced an increase to the gridbased DBStreamalgorithm based on the concept of appealbetween two adjacent grids cell and showed itseffective process.We present this paper, we developing and checking a new process tolocate

IJMTARC – VOLUME – V – ISSUE – 22, APR - JUNE, 2018

ISSN: 2320-1363

this problem for the micro cluster-based algorithm. We present the technique of a shared density graph that we explicitly capture thedensityin main data between micro clusters during clustering and then show how the graph can be used as for re-clustering micro-clusters. Usingofourknowledge, this paper is present the first to propose and investigate using a shared-density-based re-clustering this approach for data stream clustering process.The paper is balancing as follows. After a discussion of the background process in Section 2, we discuss in Section 3 the shared density graph for and the algorithm used tocapturethedensitybetweenmany microclustersintheonline process. here In Section 4 we explain the re-clustering approach which is based on clustering or finding connected in the shared density the graph. In Section 5 we present the computational complexity of the maintaining the shared density graph. In Section 6 contains detailed experiments are with synthetic and data sets. We concluding the paper with Section7 and etc.

II. BACKGROUND

Density-based clustering is a well-researched area and we can only give a very brief overview here. DBSCAN [10] and several of its improvements can be seen as the proto typical density-based clustering approach. DBSCAN estimates the density around each data point by counting the number of points in a user-specified eps-neighborhood and applies user-specified thresholds to identify core, border and noise points. In a second step, core points are joined into a cluster if they are density-reachable (i.e., there is a chain of core points where one falls inside the eps-neighborhood of the next). Finally, border points are assigned to clusters. Other approaches are based on kernel density estimation (e.g., DENCLUE [11]) or use shared nearest neighbors (e.g., SNN [12], CHAMELEON [13]).

However, these algorithms were not developed with data streams in mind. A data stream is an ordered and potentially unbounded sequence of data points X = hx1; x2; x3; : : :i. It is not possible to permanently store all the data in the stream which implies that repeated random access to the data is infeasible. Also, data streams exhibit concept drift over time where the position and/or shape of clusters changes, and new clusters may appear or existing clusters disappear. This makes the application of existing clustering algorithms difficult. Data stream clustering algorithms limit data access to a single pass over the data and adapt to concept drift. Over the last 10 years, many algorithms for clustering data streams have been proposed [5], [6], [8], [9], [14], [15], [16], [17], [18], [19], [20]. Most data stream clustering algorithms use a two-stage online/offline approach [4]:

1) Online: Summarize the data using a set of k0 microclusters organized in a space-efficient data structure, which also enables fast lookup. Micro-clusters are representatives for sets of similar data points and are created using a single pass over the data (typically in real time when the data stream arrives). Micro-clusters are typically represented by cluster centers and additional statistics as weight (density) and dispersion (variance). Each new data point is assigned to its closest (in terms of a similarity function) micro-cluster. Some algorithms use a grid instead and non-empty grid cells represent microclusters (e.g., [8], [9]). If a new data point cannot be assigned to an existing micro-cluster, a new microcluster is created. The algorithm might also perform some housekeeping (merging or deleting microclusters) to keep the number of micro-clusters at a manageable size or to remove noise or information outdated due to concept drift.

2) **Offline:** When the user or the application requires a clustering, the k0 micro-clusters are reclustered into k (k _ k0) final clusters sometimes referred to as macro-clusters. Since the offline part is usually not regarded time critical, most researchers only state that they use a conventional clustering algorithm (typically k-means or a variation of DBSCAN [10]) by regarding the micro-cluster center positions as pseudo-points. The algorithms are often modified to take also the weight of micro-clusters into account.

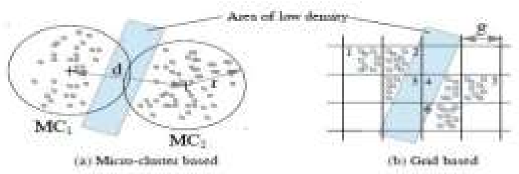

Fig. 1. Problem with reclustering when dense areas are separated bysmall areas of low density with (a) micro clusters and (b) grid cells.

Reclustering methods based solely on micro-clusters only, take closeness of the micro-clusters into account. This makes it likely that two micro-clusters which are close to each other, but separated by an area of low density still will be merged into a cluster. Information about the density between micro-clusters is not available since the information does not get recorded in the online step and the original data points are no longer available. Figure 1(a) illustrates the problem where the micro-clusters MC1 and MC2 will be merged as long as their distance d is low. This is even true when density-based clustering methods (e.g., DBSCAN) are used in the offline reclustering step, since the reclustering is still exclusively based on the micro-cluster centers and weights.

Several density-based approaches have been proposed for data-stream clustering. Density-based data stream clustering algorithms like D-Stream [7] and MR-Stream [8] use the idea of density estimation in grid cells in the online step. In the reclustering step these algorithms group adjacent dense grid cells into clusters. However, Tu and Chen [9] show that this leads to a problem when the data points within each cell are not uniformly distributed and two dense cells are separated by a small area of low density. Figure 1(b) illustrates this problem where the grid cells 1 through 6 are merged because 3 and 4 are adjacent ignoring the area of low density separating them. However, this comes at high computational cost. MR-Stream [8] approaches this problem by dynamically creating grids at multiple resolutions using a quad tree. LeaDen-Stream [20] addresses the same problem by introducing the concept of representing a MC by multiple mini-micro leaders and uses this finer representation for clustering.

For non-streaming clustering, CHAMELEON [13] proposes a solution to the problem by using both closeness and interconnectivity for clustering. An extension to DStream [9] implements this concept for data stream clustering in the form of defining attraction between grid cells as a measure of interconnectivity. Attraction information is collected during the online clustering step. For each data point that is added to a grid cell a hypercube of a userspecified size is created and for each adjacent grid the fraction of the hypercube's volume intersecting with that grid cell is recorded as the attraction between the point and that grid cell. The attraction between a grid cell and one of its neighbors is defined as the sum of the attractions of all its assigned points with the neighboring cell. For reclustering, adjacent dense grid cells are only merged if the attraction between the cells is high enough. Attraction measures the closeness of data points in one cell to neighboring cells and not density. It is also not directly applicable to general microclusters. In the following we will develop a technique to obtain density-based connectivity estimated between microclusters directly from the data.

III. THE DBSTREAM ONLINE COMPONENT

Typical micro-cluster-based data stream clustering algorithms retain the density within each micro-cluster (MC) as some form of weight (e.g., the number of points assigned to the MC). Some algorithms also capture the dispersion of the points by recording variance. For reclustering, however, only the distances between the MCs and their weights are used. In this setting, MCs which are closer to each other are more likely to end up in the same cluster. This is even true if a density-based algorithm like DBSCAN [10] is used for reclustering since here only the position of the MC centers and their weights are used. The density in the area between MCs is not available since it is not retained during the online stage.

The basic idea of this work is that if we can capture not only the distance between two adjacent MCs but also the connectivity using the density of the original data in the area between the MCs, then the reclustering results may be improved. In the following we develop DBSTREAM which stands for density-based stream clustering.

3.1 Leader-based Clustering

Leader-based clustering was introduced by Hardigan [21] as a conventional clustering algorithm. It is straight-forward to apply the idea to data streams (see, e.g., [20]).

DBSTREAM represents each MC by a leader (a data point defining the MC's center) and the density in an area of a user-specified radius r (threshold) around the center. This is similar to DBSCAN's concept of counting the points is an eps-neighborhood, however, here the density is not estimated for each point, but only for each MC which can easily be achieved for streaming data. A new data point is assigned to an existing MC (leader) if it is within a fixed radius of its center. The assigned point increases the density estimate of the chosen cluster and the MC's center is updated to move towards the new data point. If the data point falls in the assignment area of several MCs then all of them are updated. If a data point cannot be assigned to any existing MC, a new MC (leader) is created for the point. Finding the potential clusters for a new data point is a fixed-radius nearest-neighbor problem [22] which can be efficiently dealt with for data of moderate dimensionality using spatial indexing data structures like a k-d tree [23]. Variations of this simple algorithm were suggested in [24] for outlier detection and in [25] for sequence modeling. The base algorithm stores for each MC a weight which is the number of data points assigned to the MC (see w1 to w4 in Figure 2). The density can be approximated by this weight over the size of the MC's assignment area. Note that we use for simplicity the area here, however, the approach is not restricted to two-dimensional data. For higher-dimensional data volume is substituted for area.

Definition 3.1. The **density** of MC iis estimated by,

$$p_i = \frac{w_i}{A_i}$$

wherewi is the weight and Ai, the area of the circle with radius r around the center of MC i.

3.2 Capturing Shared Density

Capturing shared density directly in the online component is a new concept introduced in this paper. The fact, that in dense areas MCs will have an overlapping assignment area, can be used to measure density between MCs by counting the points which are assigned to two or more MCs. The idea is that high density in the intersection area relative to the rest of the MCs' area means that the two MCs share an area of high density and should be part of the same macrocluster. In the example in Figure 2 we see that MC2 and MC3 are close to each other and overlap. However, the shared weight s2;3 is small compared to the weight of each of the two involved MCs indicating that the two MCs do not form a single area of high density. On the other hand, MC3 andMC4 are more distant, but their shared weight s3;4 is large indicating that both MCs form an area of high density and thus should form a single macro-cluster. The shared density between two MCs can be estimate by:

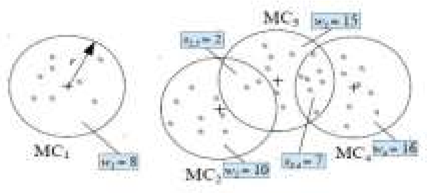

Fig. 2. MC1 is a single MC. MC2 and MC3 are close to each other butthe density between them is low relative to the two MCs densities whileMC3 and MC4 are connected by a high density area.

Definition 3.2. The **shared density** between two MCs, i and j, is estimated by

$$p_{i,j} = \frac{s_{i,j}}{A^{i,j}}$$

wheresij is the shared weight and Aij is the size of the overlapping area between the MCs.

Based on shared densities we can define a shared density graph.

Definition 3.3. A **shared density graph** Gsd = (V;E) isan undirected weighted graph, where the set of verticesis the set of all MCs, i.e., V (Gsd) = MC, and the setof edges

$$E(G_{sd}) = \{(vi, vj) | vi, vi \in V(G_{sd}) \nabla \, \mathrm{p}_{\mathrm{i,j}} > 0\}$$

represents all the pairs of MCs for which we havepairwise density estimates. Each edge is labeled with thepairwise density estimate ^_ij .

Note that most MCs will not share density with each other in a typical clustering. This leads to a very sparse shared density graph. This fact can be exploited for more efficient storage and manipulation of the graph. We represent the sparse graph by a weighted adjacency list S. Furthermore, during clustering we already find all fixed-radius nearest-neighbors. Therefore, obtaining shared weights does not incur any additional increase in search time.

3.4 Fading and Forgetting Data

To adapt to evolving data streams we use the exponential fading strategy introduced in DenStream [6] and used in many other algorithms. Cluster weights are faded in $2^{-\tau}$ every time step by a factor of where $\tau > 0$ is a user-specified fading factor. We implement fading in a similar way as in D-Stream [9], where fading is only applied when a value changes (e.g., the weight of a MC is updated). For example, if the current time-step is t = 10 and the weight w was last updated at tw = 5 then we apply for fading the factor $2^{-\tau}(t - t_w)$resulting in the correct fading for five time steps. In order for this approach to work we have to keep a timestamp with the time when fading was applied last for each value that is subject to fading.

The leader-based clustering algorithm only creates new and updates existing MCs. Over time, noise will cause the creation of low-weight MCs and concept shift will make some MCs obsolete. Fading will reduce the weight ofthese MCs over time and the reclustering has a mechanism to exclude these MCs. However, these MCs will still be stored in memory and make finding the fixed-radius nearest neighbors during the online clustering process slower. This problem can be addressed by removing weak MCs and weak entries in the shared density graph. In the following we define weak MCs and weak shared densities.

Definition 3.4. We define MC mci as a **weak MC** if its weight wi increases on average by less than one new data point in a user-specified time interval tgap.

Definition 3.5. We define a **weak entry in the shared density graph** as an entry between two MCs, i and j, which on average increases its weight sij by less then σ from new points in the time interval tgap. $\sigma$is the intersection factor related to the area of the overlap of the MCs relative to the area covered by MCs.

The rational of using σ is that the overlap areas are smaller than the assignment areas of MCs and thus are likely to receive less weight. σwill be discussed in detail in the reclustering phase.

Let us assume that we check every tgap time step and remove weak MCs and weak entries in the shared density graph to recover memory and improve the clustering algorithm's processing speed. To ensure that we only remove weak entries, we can use the weight At any $W_{weak} = 2^{-\tau} t_{gap}$

time, all entries that have a faded weight of less than wweak are guaranteed to be weak. This is easy to see since any entry that gets on average an additional weight of $W^1 \geq 1$during each tgap interval will have a weight of at least $W_{weak} = 2^{-\tau} t_{gap}$which is greater or equal to wweak. Noise entries (MCs and entries in the shared density graph) often receive only a single data point and will reach wweak after tgap time steps.

Obsolete MCs or entries in the shared density graph stop to receive data points and thus their weight will be faded till it falls below wweak and then they are removed. It is easy to show that for an entry with a weight w it will take $t = \frac{log_2(w)}{\lambda} + t_{gap}$time steps to reach wweak.

Forexample, at = 0:01 and tgap = 1000 it will take 1333 time steps for an obsolete MC with a weight of w = 10 to fall below wweak. The same logic applies to shared density entries using _wweak. Note that the definition of weak entries and wweak is only used for memory management purpose. Reclustering uses the definition of strong entries (see Section 4). Therefore, the quality of the final clustering is not affected by the choice of tgap as long as it is not set to a time interval which is too short for actual MCs and entries in the shared density graph to receive at least one data point. This clearly depends on the expected number of MCs and therefore depends on the chosen clustering radius r and the structure of the data stream to be clustered. A low multiple of the number of expected MCs is typically sufficient. The parameter tgap can also be dynamically adapted during running the clustering algorithm. For example tgap can be reduced to mark more entries as weak and remove them more often if memory or processing speed gets low. On the other hand, tgap can be increased during clustering if not enough structure of the data stream is retained.

3.5 The Complete Online Algorithm

Algorithm 1 shows our approach and the used clustering data structures and user-specified parameters in detail.

Micro-clusters are stored as a set MC. Each micro-cluster is represented by the tuple (c;w; t) representing the clustercenter, the cluster weight and the last time it was updated, respectively. The weighted adjacency list S represents thesparse shared density graph which captures the weight of the data points shared by MCs. Since shared density estimates are also subject to fading, we also store a timestamp with each entry. Fading also shared density estimates is important since MCs are allowed to move which over time would lead to estimates of intersection areas the MC is not covering anymore.

The user-specified parameters r (the radius around the center of a MC within which data points will be assigned to the cluster) and τ (the fading rate) are part of the base algorithm. α, t_{gap}andwmin are parameters for reclustering and memory anagement and will be discussed later.

Updating the clustering by adding a new data point x to the clustering is defined by Algorithm 1. First, we find all MCs for which x falls within their radius. This is the same as asking which MCs are within r from x, which is the fixedradius nearest neighbor problem which can be efficiently solved for data of low to moderate dimensionality [22]. If no neighbor is found then a new MC with a weight of 1 is created for x (line 4 in Algorithm 1). If one or more neighbors are found then we update the MCs by applying the appropriate fading, increasing their weight and then we try to move them closer to x using the Gaussian neighborhood function h() (lines 7–9).

Next, we update the shared density graph (lines 10–13). To prevent collapsing MCs, we restrict the movement for MCs in case they come closer than r to each other (lines 15–19). Finally, we update the time step.

The cleanup process is shown in Algorithm 2. It is executed every tgap time steps and removes weak MCs andweak entries in the shared density graph to recover memory and improve the clustering algorithm's processing speed.

4SHAREDDENSITY-BASED RECLUSTERING

Reclustering represents the algorithm's offline component which uses the data captured by the online component. For simplicity we discuss two-dimensional data first and later discuss implications for higher-dimensional data. For reclustering, we want to join MCs which are connected by areas of high density. This will allow us to form macroclusters of arbitrary shape, similar to hierarchical clustering with single-linkage or ←DBSCAN's reachability, while avoiding joining MCs which are close to each other but are separated by an area of low density

4.1 Micro-Cluster Connectivity

In two dimensions, the assignment area of a MC is given by$A = \pi r^2$. It is easy to show that the intersecting area between two circles with equal radius r and the centers exactly r apart from each other is $A *= \frac{4\pi - \sqrt[3]{3}}{\beta} r^2$ By normalizing the area of the circle to A = 1 (i.e., setting the radius to) $r = \sqrt{1/_{\pi}}$ we get an intersection area A_ = 0:391.or 39.1% of the circle's area. Since we expect adjacent MCs i and j which form a single macro-cluster in a dense area to eventually be packed together till the center of one MC is very close to the r boundary of the other, 39.1% is the upper bound of the intersecting area.

Algorithm 1 Update DBSTREAM clustering.

Require: Clustering data structures initially empty or 0

*d*set of MCs *mc* has elements *mc* = (**c**, *w*, *t*) *d* center, weight, last updatetime
S *d*weighted adjacency list for shared densit∈ygraph s_{ij}**S** has an additional field *t* *d* time of last update *t* ,*d*currenttimestep

Require: User-specified parameters
r d clusteringthreshold
λ d fadingfactor
t_{gap}*d* cleanupinterval
w_{min}*d* minimumweight
α d intersectionfactor

```
1: function UPDATE(x) d new data point x
2:     find FixedRadiusNN(x, ,r)
3:         if |N| <1 then d create new MC
4:             add (c = x, t = t, w = 1) to MC
5:         else                                   d update
existingMCs
6:             for each ∈ N do
7:                 $mc_i[w] \leftarrow mc_i[w] 2^{-\lambda(t - mc_i[t])} + 1$
8:                 $mc_i[c]$   $mc_i[c]$   +   $h(x,$
$mc_i[c])(x mc_i[c])$
9:                 $mc_i[t] \leftarrow t$
d update shared density
10:                foreach j ∈ N where $j > i$ do
11:                    $s_{ij} \leftarrow s_{ij} 2^{-\lambda(t - s_{ij}[t])} + 1$
12:                    $s_{ij}[t] \leftarrow t$
13:                endfor
14:            endfor
d prevent collapsing clusters
15:            for each (i,j) ∈ N × N and $j > i$ do
16:                if dist($mc_i[c]$, $mc_j[c]$) < r then
```

```
17:              revert  $mc_i$[c],  $mc_j$[c]  to previouspositions
18:                  endif
19:              endfor
20:          endif
21:          $t$    $t + 1$
22:  endfunction
```

$(\hat{\rho}+\hat{\rho})/2$

Algorithm 2 Cleanup process to remove inactive micro- clusters and shared density entries from memory.

```
Require: $\lambda$, $\alpha$, $t$gap, $t$, MC and S from the clustering.
1: function CLEANUP( )
2:      $w$weak $=2^{-\lambda t\text{gap}}$
3:      foreach$mc$            do
4:          if $mc[w]$ $2^{-\lambda(t-mc[t])}$ < $w$weak then
5:              remove weak $mc$from
6:          endif
7:      endfor
8:      foreach$s_{ij}$      Sdo
9:          if$s_{ij}2^{-\lambda(t-s_{ij}[t])}<\alpha w$weakthen
10:             remove weak shared density $s_{ij}$from S
11:         endif
12:     endfor
13: end function
```

Less dense clusters will also have a lower shared density. To detect clusters of different density correctly, we need to define connectivity relative to the densities (weights) of the participating clusters. That is, for two MCs, i and j, which are next to each other in the same macro-cluster we expect that $p_{ij} \approx (p_i + p_j)/2$i.e., the density between the MCs is similar to the average density inside the MCs. To formalize this idea we introduce the connectivity graph.

Definition 4.1. The **connectivity graph** Gc = (V;E) is an undirected weighted graph with the micro clusters as vertices, i.e., V (Gc) = MC. The set of edges is defined by $E(G_c) = \{(vi, vj) | vi, vi \in V(G_c) \nabla\, c_{i,j} > 0\}$ with$c_{ij} = s_{ij}(w_i + w_j)/2$sij is the weight in the intersecting area of MCs i and j and wi and wj are the MCs' weights. The edges are labeled with weights given by cij .

$\leftarrow$

Note that the connectivity is not calculated as $\frac{p_{i,j}}{(p_i+p_j)/2}$ and thus has to be corrected for the difference in the size of the area of the MCs and the intersecting area. This can be easily done by introducing an intersection factor _ij = Aij=Ai which results in $\frac{p_{i,j}}{(p_i+p_j)/2} = \alpha_{i,j} c_{i,j}$$\alpha_{ij}$ depends on the distance between the participating MCs i and j. Similar to the non-streaming algorithm CHAMELEON [13], we want to combine MCs which are close together and have high interconnectivity. This objective can be achieved by simply choosing a single global intersection factor α. This leads to the concept of α -connectedness.

Definition 4.2. Two MCs, i and j, are α **-connected** iffcij≥α, where α is the user-defined intersection factor.

For two-dimensional data the intersection factor α has a theoretical maximum of 0.391 for an area of uniform density when the two MCs are optimally packed (the centers are exactly r apart). However, in dynamic clustering situations MCs may not be perfectly packed all the time and minor variations in the observed density in the data are expected. Therefore, a smaller value than the theoretically obtained maximum of 0.391 will be used in practice. It is important to notice that a threshold on α is a single decision criterion which combines the fact that two MCs are very close to each other and that the density between them is sufficiently high. Two MCs have to be close together or the intersecting area and thus the expected weight in the intersection will be small and the density between the MCs has to be high relative to the density of the two MCs. This makes using the
concept of α -connectedness very convenient.

For 2-dimensional data we suggest α = :3 which is a less stringent cut-off than the theoretical maximum. Doing this will also connect MCs, even if they have not (yet) moved into a perfect packing arrangement. Note also that the definitions of α -connectedness uses the connectivity graph which depends on the density of the participating MCs and thus it can automatically handle clusters of vastly different density within a single clustering.

4.2 Noise Clusters

To remove noisy MCs from the final clustering, we have to detect these MCs. Noisy clusters are typically characterized as having low density represented by a small weight. Since the weight is related to the number of points covered by the MC, we use a user-set minimum weight threshold to identify noisy MCs. This is related to minPoints in DBSCAN or Cm used by D-Stream.

Definition 4.3. The set of **noisy MCs**, MCnoisy, is the subset of MC containing the MCs with less than a userspecified minimum weight wmin. That is, MCnoisy = $\{ MC_i | MC_i \in MC \wedge w_i < w_{\min} \}$.

Given the definition of noisy and weak clusters, we can define strong MCs which should be used in the clustering.

Definition 4.4. A **strong MC** is a MC that is not noisy or weak, i.e., $MC_{strong} = MC \setminus (MC_{noisy} \cup MC_{weak})$.

Note that tgap is typically chosen such that $MC_{weak} \subseteq MC_{noisy}$ and therefore the exact choice of tgap has no influence on the resulting clustering, only influencing runtime performance and memory requirements.

Algorithm 3 Reclustering using shared densitygraph.

```
Require: $\lambda$, $\alpha$, $w_{\min}$, $t$, and S from theclustering.
1: function RECLUSTER( )
2:     weighted adjacency list C ← $\emptyset d$ connectivitygraph
3:     for each $s_{ij} \in$ S do d construct connectivitygraph
4:         if $MC_i[w] \geq w_{\min} \wedge MC_j[w] \geq w_{\min}$ then
5:             $c_{ij} \leftarrow$
6:         endif
7.     end for
8:     return findConnectedComponents(C $\alpha$)
9: END FUNCTION
```

4.3 The Offline Algorithm

The parameters are the intersection. Calculate a Density between A3 *(A1,A2,A3,A4,A5,A6)

Table 1 Calculate a Density between A3*(A1-A6)

1	Attribute	A1*A3	A2*A3	A3*A3	A4*A3	A5*A3	A6*A3
2	Density value	2.5457	3.7088	3.2087	3.7778	3.33149	4.0746

TABLE 2. Relavance data

sno	Feature	Entropy	C_entropy	gain	relavance
1	A1	-118.5444	0.008399	-118.5528	2.00014
2	A2	-22.05218	0.044302	-22.09658	2.00040
3	A3	-31.72286	0.03666	-31.75390	2.00195
4	A4	-25.57785	0.033219	-25.61789	2.00299
5	A5	-28.89565	0.034678	-28.09978	2.00249

The connectivity graph C is constructed using only shared density entries between strong MCs. Finally, the edges in the connectivity graph with a connectivity value greater than the intersection threshold are used to find connected components representing the final clusters.

4.4 Relationship to Other Algorithms

DBSTREAM is closely related to DBSCAN [10] with two important differences. Similar to DenStream [6], density estimates are calculated for micro-clusters rather than the epsilon neighborhood around each point. This reduces computational complexity significantly. The second change is that DBSCAN's concept of reachability is replaced by α - connectivity. Reachability only reflects closeness of data points, while α -connectivity also incorporates interconnectivity introduced by CHAMELEON [13].

In general, the connectivity graph developed in this paper can be seen as a special case of a shared nearest neighbor graph where the neighbors shared by two MCs are given by the points in the shared area. As such it does not represent k shared nearest neighbors but the set of neighbors given by a fixed radius. DBSTREAM uses the most simple approach to partition the connectivity graph by using α as a global threshold and then finding connected components. However, any graph partitioning scheme, e.g., the ones used for CHAMELEON or spectral clustering, can be used to detect clusters.

Compared to D-Stream's concept of attraction which is used between grid cells, DBSTREAM's concept of α -connectivity is also applicable to micro-clusters. DBSTREAM's update strategy for micro cluster centers based on ideas from competitive learning allows the centers to move towards areas of maximal local density, while

DStream's grid is fixed. This makes DBSTREAM more flexible which will be illustrated in the experiments by the fact that DBSTREAM typically needs fewer MCs to model the same data stream.

6 CONCLUSION

In this paper, we have developed the first data stream clustering algorithm which explicitly records the density in the area shared by micro-clusters and uses this information for reclustering. We have introduced the shared density graph together with the algorithms needed to maintain the graph in the online component of a data stream mining algorithm. Although, we showed that the worst-case memory requirements of the shared density graph grow extremely fast with data dimensionality, complexity analysis and experiments reveal that the procedure can be effectively applied to data sets of moderate dimensionality. Experiments also show that shared-density reclustering already performs extremely well when the online data stream clustering component is set to produce a small number of large MCs. Other popular reclustering strategies can only slightly improve over the results of shared density reclustering and need significantly more MCs to achieve comparable results. This is an important advantage since it implies that we can tune the online component to produce less micro-clusters for shared-density reclustering. This improves performance and, in many cases, the saved memory more than offset the memory requirement for the shared density graph.

REFERENCES

[1] S. Guha, N. Mishra, R. Motwani, and L. O'Callaghan, "Clustering data streams," in Proceedings of the ACM Symposium on Foundations of Computer Science, 12-14 Nov. 2000, pp. 359–366.

[2] C. Aggarwal, Data Streams: Models and Algorithms, ser. Advances in Database Systems, Springer, Ed., 2007.

[3] J. Gama, Knowledge Discovery from Data Streams, 1st ed. Chapman & Hall/CRC, 2010.

[4] J. A. Silva, E. R. Faria, R. C. Barros, E. R. Hruschka, A. C. P. L. F. d. Carvalho, and J. a. Gama, "Data stream clustering: A survey," ACM Computing Surveys, vol. 46, no. 1, pp. 13:1–13:31, Jul. 2013.

[5] C. C. Aggarwal, J. Han, J. Wang, and P. S. Yu, "A framework for clustering evolving data streams," in Proceedings of the International Conference on Very Large Data Bases (VLDB '03), 2003, pp. 81–92.

[6] F. Cao, M. Ester, W. Qian, and A. Zhou, "Density-based clustering over an evolving data stream with noise," in Proceedings of the 2006 SIAM International Conference on Data Mining. SIAM, 2006, pp. 328–339.

[7] Y. Chen and L. Tu, "Density-based clustering for real-time stream data," in Proceedings of the 13th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining. New York, NY, USA: ACM, 2007, pp. 133–142.

[8] L. Wan, W. K. Ng, X. H. Dang, P. S. Yu, and K. Zhang, "Densitybased clustering of data streams at multiple resolutions," ACM Transactions on Knowledge Discovery from Data, vol. 3, no. 3, pp. 1–28, 2009.

[9] L. Tu and Y. Chen, "Stream data clustering based on grid density and attraction," ACM Transactions on Knowledge Discovery from Data, vol. 3, no. 3, pp. 1–27, 2009.

[10] M. Ester, H.-P. Kriegel, J. Sander, and X. Xu, "A density-based algorithm for discovering clusters in large spatial databases with noise," in Proceedings of the ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (KDD'1996), 1996, pp. 226– 231.

[11] A. Hinneburg, E. Hinneburg, and D. A. Keim, "An efficient approach to clustering in large multimedia databases with noise," in Proceedings of the Fourth International Conference on Knowledge Discovery and Data Mining (KDD-98). AAAI Press, 1998, pp. 58– 65.

[12] L. Ertoz, M. Steinbach, and V. Kumar, "A new shared nearest neighbor clustering algorithm and its applications," in Workshop on Clustering High Dimensional Data and its Applications at 2nd SIAM
International Conference on Data Mining, 2002.

[13] G. Karypis, E.-H. S. Han, and V. Kumar, "Chameleon: Hierarchical clustering using dynamic modeling," Computer, vol. 32, no. 8, pp. 68–75, Aug. 1999. [Online]. Available:http://dx.doi.org/10.1109/2.781637

[14] S. Guha, A. Meyerson, N. Mishra, R. Motwani, and L. O'Callaghan, "Clustering data streams: Theory and practice," IEEE Transactions on Knowledge and Data Engineering, vol. 15, no. 3, pp. 515–528, 2003.

[15] C. C. Aggarwal, J. Han, J. Wang, and P. S. Yu, "A framework for projected clustering of high dimensional data streams," in Proceedings of the International Conference on Very Large Data Bases
(VLDB '04), 2004, pp. 852–863.

[16] D. Tasoulis, N. Adams, and D. Hand, "Unsupervised clustering in streaming data," in IEEE International Workshop on Mining Evolving and Streaming Data. Sixth IEEE International Conference on Data Mining (ICDM 2006), Dec. 2006, pp. 638–642.
[17] D. K. Tasoulis, G. Ross, and N. M. Adams, "Visualising the cluster structure of data streams," in Advances in Intelligent Data Analysis VII, ser. Lecture Notes in Computer Science. Springer, 2007, pp. 81–92.
[18] K. Udommanetanakit, T. Rakthanmanon, and K. Waiyamai, "Estream: Evolution-based technique for stream clustering," in ADMA '07: Proceedings of the 3rd international conference on Advanced Data Mining and Applications. Berlin, Heidelberg: Springer-Verlag, 2007, pp. 605–615.
[19] P. Kranen, I. Assent, C. Baldauf, and T. Seidl, "The clustree:indexing micro-clusters for anytime stream mining," Knowledge and Information Systems, vol. 29, no. 2, pp. 249–272, 2011.
[20] A. Amini and T. Y. Wah, "Leaden-stream: A leader densitybased clustering algorithm over evolving data stream," Journal of Computer and Communications, vol. 1, no. 5, pp. 26–31, 2013.