Can Exclusive Clustering on Streaming Data be Achieved?

Maria E. Orlowska
School of ITEE
The University of Queensland
Brisbane, Australia
maria@itee.uq.edu.au

Xingzhi Sun
School of ITEE
The University of Queensland
Brisbane, Australia
sun@itee.uq.edu.au

Xue Li
School of ITEE
The University of Queensland
Brisbane, Australia
xueli@itee.uq.edu.au

ABSTRACT

Clustering on streaming data aims at partitioning a list of data points into k groups of "similar" objects by scanning the data once. Most current one-scan clustering algorithms do not keep original data in the resulting clusters. The output of the algorithms is therefore not the clustered data points but the approximations of data properties according to the predefined similarity function, such that k centers and radiuses reflect the up-to-date data grouping. In this paper, we raise a critical question: can the partition-based clustering, or exclusive clustering, be achieved on streaming data by those currently available algorithms? After identifying the differences between traditional clustering and clustering on data streams, we discuss the basic requirements for the clusters that can be discovered from streaming data. We evaluate the recent work that is based on a subcluster maintenance approach. By using a few straightforward examples we illustrate that the subcluster maintenance approach may fail to resolve the exclusive clustering on data streams. Based on our observations, we also present the challenges on any heuristic method that claims solving the clustering problem on data streams in general.

1. INTRODUCTION

Data streams [3; 7] are generated by many electronic devices; some with a very high volume of complex data types, and others with high volume but mainly numeric data. Typically, analytical applications executed on such data sets are mainly concerned with derived data dependencies among the data entries or rather simple statistical measures characterizing such data flows. For many applications, there is considerable merit with processing such requirements only in the main memory without the necessity (or need) to store the complete data sets in secondary memory devices. This last requirement makes some computations of data streams very challenging.

Without any specific selection of a computationally complex problem, let us reflect on the intrinsic difficulties imposed on the computational task when assumption of a single data scan is imposed.

A brief departure to the past to examine computational complexity analysis of typical problems such as clustering, classification, sorting, some types of data mining etc., indicates that the main (biggest) component of corresponding computational complexity expressions is generated by the fact that one must unavoidably execute several scans through the entire data sets to deliver an exact outcome. There is no known way to remove this drawback and deliver the expected output stated in those problem definitions. Hence, if the complexity of the considered problem is too high, we normally invent poorer or better heuristic methods to deliver acceptable quality solutions. In general for a given problem, the question whether a heuristic method is adequate or not is difficult to evaluate and often inadequately addressed in many research papers. If there is only one heuristic known (an uncommon case) one could accept less accurate results but there is a limit on what one could regard as a result and what is too 'far' from the expected qualities. In the case of many known heuristic methods one could order them (normally only partial order is possible) and select the best (with some measure of fitness) or reject all and search further for a more accurate approximation of an exact solution.

Data stream analysis, such as clustering, clearly belongs to the class of hard computational problems where several heuristics exist [13; 8; 11; 5; 12; 1; 2]. To establish a suitable context for this paper, let us begin with a few comments on clustering problems in general.

Clustering can be considered the most important unsupervised learning problem; and as with every other problem of this kind, it deals with finding a structure in a collection of unlabelled data. An informal definition of clustering could be "the process of organizing objects into groups whose members are similar in some way".

A cluster is therefore a collection of objects which are "similar", but are "dissimilar" to the objects belonging to other clusters. Often, the similarity criterion is distance: two or more objects belong to the same cluster if they are "close" according to a given distance measure. This is called distance-based clustering. Another kind of clustering is conceptual clustering where two or more objects belong to the same cluster if this one defines a concept common to all those objects. In other words, objects are grouped according to their fit to descriptive concepts, not according to simple similarity measures. Clearly, in our context (of data streams) we are mainly concerned with distance-based clustering.

So, the goal of clustering is to determine the intrinsic grouping in a set of unlabelled data. But how to decide what constitutes a good clustering? It can be shown that there is no absolute "best" criterion which would be independent of the final aim of the clustering. Consequently, it is the user that must supply this criterion, in such a way that the result of the clustering will suit their needs. Thus in a static environment when we can scan the data several times as well in a dynamic data environment such as streaming data, the main requirements that a clustering algorithm should satisfy are: insensitivity to order of input records, deliverability of the expected relationship between clusters (over-

lapping or exclusive), scalability, dealing with different types of attributes, discovering clusters with arbitrary shapes, effective dealing with high dimensionality of data and more user related issues, and interpretability. For the completeness of this brief summary we add that there are no known algorithms that satisfy all these constraints effectively and precisely.

In this paper, we discuss exclusive clustering, or partition-based clustering, on data streams. First, considering in a static dataset, we refer to exclusive clustering as the following problem.

PROBLEM 1. *Dividing N data points into k groups (clusters) such that 1) each data point belongs to exactly one cluster and 2) the objective function, which reflects the cohesion of the points in clusters (such as the sum of the square distances from the cluster centers), is minimized.*

In general, the problem stated above belongs to the class of NP-hard problems (refer to [9]). Many heuristic methods [10] have been invented to generate approximations, being near optimum solutions, but all with multiple scans on the data set.

Now, let us consider the exclusive clustering on data streams (i.e., *Problem 1* on the context of data streams). Since it is already well understood that random access of streaming data is not possible, the problem of exclusive clustering on data streams can be regarded as the *Problem 1* with two constraints on the computation: the data points can only be processed in the order they arrive and can be scanned only once during developed procedures. Only those two constraints suggest that any traditional heuristic clustering methods will have serious difficulty to deliver.

Besides these two constraints, a significant difference between the traditional clustering approach and streaming clustering approach lies in their results. Because data points on the stream cannot be stored due to limited size of main memory, unlike traditional clustering problem which could output k set of points, each outputted cluster on data streams can only be represented by the compact features rather than all points in it. For example, possible features to represent a cluster are the center and radius of all points in the cluster. Since the goal of exclusive clustering is to partition data and the clustering results on stream are only features of clusters, intuitively, these features need to reflect the partition of the data. That is, if any features are adopted to describe the k clusters, all points in a cluster must be represented by the features, and there is no overlap between clusters in terms of their features.

Based on the above discussion, we re-evaluate the problem of exclusive clustering on on data streams as follows

PROBLEM 2. *Given a stream S of N points and an integer k, by scanning S once, find k clusters (each of which is represented by the compact features of the points in the cluster) such that*

1. *for any point p in the dataset, by referring to cluster features, we can identify exact one cluster that contains p.*

2. *the value of objective function is close to optimum.*

Explicitly, we call the exclusive clustering on data streams can be achieved if the clustering result satisfies the above two **requirements**.

Once the problem is defined, we discuss whether the *Problem 2* can be achieved by evaluating some recent studies [13; 1; 2], which address the clustering task of partitioning N points into k groups.

All reported approaches follow two phases of computations. The principle idea is that the summary of historical data is incrementally maintained in main memory, and the clustering result is derived from the summary. Particularly, in phase one, a fixed number of subclusters are maintained as the summary of the data stream. Each subcluster is represented by its cluster feature [13], which gives the cardinality, center, and radius of the cluster. Based on some heuristics, each new arriving point is assigned to a subcluster, which leads to the update of its cluster feature. When the final cluster needs to be generated, the subclusters are treated as weighted data points and various traditional clustering algorithms can be applied in the phase two computation without any I/O cost. In this paper, we denote this type of approach as **subcluster maintenance approach**.

After careful analysis of these solutions and development of illustrative simple examples, we developed some interesting observations that form the main objective of this paper. The goal of this paper is to clarify some fundamental questions in clustering problem on data streams. Especially, we evaluate whether the subcluster maintenance approach [13; 1; 2; 4] can lead to exclusive and "good" clustering on data streams. Using simple but powerful examples, we illustrate that the incrementally maintained subclusters, as a clustering method, may generate results violating the main assumptions/expectation of expected set of clusters (two requirements in the *Problem 2*): firstly, being a partition of the data set and secondly, the value of the objective function is close to the minimum. Consequently, we claim that, in general, the subcluster maintenance approach cannot achieve exclusive clustering on data streams. This statement also can be extended to any other type of clustering on a stream that depends on the prior selection of subclusters, e.g., density-based clustering [4].

We conclude this paper by summarizing the challenge of clustering on data streams and arguing that in general, any heuristic method face the same difficulty to generate good clustering result.

The rest of this paper is organized as follows. Section 2 gives a background of traditional portioning-based clustering problem on data streams. In Section 3, we illustrate the exclusive clustering on data streams cannot be achieved by the existing subcluster maintenance algorithms. Finally, we provide concluding remarks in Section 4.

2. BACKGROUND

In this section, we establish the background of our discussion in this paper. First, we introduce the most frequently used exclusive clustering algorithm, k-means [9], to show how the exclusive clustering are computed on static datasets. Then, we review the clustering techniques on data streams which are based on the maintenance of subclusters.

2.1 k-Means Clustering

For clarity, we give the formal definition of k-means problem as below.

DEFINITION 1. *Let an integer k and a set S of points in Euclidean space be given. For any set of k points $P = \{c_1, c_2, \ldots, c_k\}$ in the space, S can be partitioned into k corresponding clusters $C_1, C_2, \ldots, C_k$ by assigning each point in S to the closest center c_i. The problem of k-means is to find the set P such that the sum of squared distance (SSQ) measure, defined as $\sum_{i=1}^{k} \sum_{p \in C_i} |p - c_i|^2$, is minimized.*

In general, the k-means problem belongs to the class of NP-hard problems [9]. Many heuristic methods have been invented to provide near optimum solutions. Now, we review the process of famous k-means algorithm.

The procedure of k-means algorithm follows a simple way to partition a given data set into k subsets. The main idea is to define k centers, one for each cluster. The challenge is to place these centers in a clever way because each different location of centers results in different clustering. Hence, a reasonable starting choice is to locate them far away from each other as much as possible. The subsequent step is to allocate each point from a given data set to the nearest center. We stop the first phase when all points are considered. At this stage we must to re-calculate k new centers resulting from the previous two steps. After we identify these k new centers, a new allocation has to be done between the same data set points and the nearest new center. Clearly, a loop has been generated, and as usual in such cases, we stop the process when location of centers is stable; in other words centers do not move any more. Finally, this algorithm aims at minimizing an objective function, which reflects the cohesion of the points in clusters.

Even though it is easy to prove that the procedure always terminates, the k-means algorithm does not necessarily find the most optimal configuration, corresponding to the global minimum of the objective function. Also, it is easy to demonstrate that this algorithm is significantly sensitive to the initial randomly selected centers. To remove this effect the k-means algorithm can be executed multiple times in order to identify a better global solution. This takes us back to previous comments about the fitness of a heuristic method. Despite different heuristics are applied, the result of k-means clustering **always** satisfy the following **condition**:

Condition 1. For any point p in S, there exists exact one cluster C_i containing p. Importantly, for any $p \in C_i$, there does not exist another cluster C_j satisfying $|p - c_i| > |p - c_j|$, where c_i and c_j are the center of C_i and C_j respectively.

The *Condition 1* is determined by the definition of k-means clustering. Importantly, when considering the features for representing k clusters, once the clustering result satisfies the *Condition 1*, the most succinct features that meet the *requirement 1* in the *Problem 2* are k cluster centers. Visually, given k centers in the metric space, a Voronoi diagram can be created by uniquely partitioning the space into k regions, such that every point of the space is assigned to the closet point among the k specified centers. Obviously, in the k-means clustering result, each cluster corresponds to a region in the Voronoi diagram determined by k centers. Figure 1 gives a Voronoi diagram which could be the representation of the k-means clustering result for a set of points in 2-D space.

2.2 Representation of cluster on data streams

Since the data stream is continuous and unbounded, the representation of clusters should be compact and could be maintained incrementally. Based on the above consideration, most recent studies [13; 1; 2; 4] adopt the cluster features that was originally defined in [13] as the representation of clusters. Due to importance of this key concept, we give the formal definition as below.

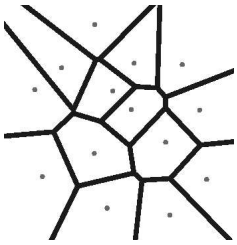

Figure 1: Voronoi diagram and clustering result

DEFINITION 2. *Let a point p_i in d-dimensional space be denoted as $(p_i^1, p_i^2, \ldots, p_i^d)$. Given a cluster C with points $\{p_1, p_2, \ldots, p_n\}$, the cluster feature CF of C is a $(d+2)$-tuple defined as $\{n, ls^1, \ldots, ls^d, ss\}$, where n is the cardinality of C, $ls^j = \sum_{i=1}^{n} p_i^j$ $(j : 1 \ldots d)$, and $ss = \sum_{i=1}^{n} \sum_{j=1}^{d} \left(p_i^j\right)^2$.*

The cluster feature is regarded as an ideal representation of clusters on streaming data in many literatures [13; 1; 2; 4] due to the following properties.

1. In d-dimensional space, given n points in a cluster $C = \{p_1, p_2, \ldots, p_n\}$, the center c and radius r of the cluster C are defined as:

$$c = \left(\frac{\sum_{i=1}^{n} p_i^1}{n}, \frac{\sum_{i=1}^{n} p_i^2}{n}, \ldots, \frac{\sum_{i=1}^{n} p_i^d}{n} \right)$$

$$r = \left(\frac{\sum_{i=1}^{n} |p_i - c|^2}{n} \right)^{\frac{1}{2}}$$

   Given the cluster feature $CF = \{n, ls^1, \ldots, ls^d, ss\}$ of C, the center and radius can be computed as follows:

$$c = \left(\frac{ls^1}{n}, \frac{ls^2}{n}, \ldots, \frac{ls^d}{n} \right) \quad (1)$$

$$r = \left(\frac{ss}{n} - \frac{\sum_{j=1}^{d} \left(ls^j\right)^2}{n^2} \right)^{1/2} \quad (2)$$

   This property shows that cluster feature can be a compact summary for sphere-shaped clusters. However, due to the definition of radius r, points belonging to the cluster C can be outside the region determined by equation $|p - c| = r$. Thus, only the center and radius cannot represent every point in the cluster.

2. Given the cluster feature $\{n_1, ls_1^1, \ldots, ls_1^d, ss_1\}$ and $\{n_2, ls_2^1, \ldots, ls_2^d, ss_2\}$ for two clusters C_1 and C_2, the cluster feature of $C_1 \cup C_2$ is $\{n_1 + n_2, ls_1^1 + ls_2^1, \ldots, ls_1^d + ls_2^d, ss_1 + ss_2\}$. Thanks to this additive property, cluster features can be incrementally maintained. Considering when a new point $p = (p^1, p^2, \ldots, p^d)$ is assigned into a cluster C_1 (which is the case that $C_2 = \{p\}$), the cluster feature of $C_1' = C_1 \cup \{p\}$ can be

incrementally updated as $\{n_1 + 1, ls_1^1 + p^1, \ldots, ls_1^d + p^d, ss_1 + \sum_{j=1}^{d} (p^j)^2\}$. In addition, when two clusters are merged, which happens regularly in the hierarchical clustering methods, the cluster feature of the new cluster can be computed accurately.

After knowing the representation of clusters on data streams, in the next section, we discuss how previous studies address the exclusive clustering problem on data streams. Particularly, we focus on the subcluster maintenance approach, which has been reported in recent literatures [13; 1; 2; 4].

2.3 Process of subcluster maintenance algorithms

The biggest challenge of finding k clusters on data streams is that there is no global information of data. Points can only be accessed based on the order of timestamps and past data are no longer available. To address these issues, existing work [13; 1; 2; 4] uses a hierarchical clustering technique, which consists of two stages.

1. Incrementally maintain m ($m > k$) subclusters (each of which is represented by its cluster feature) as the summary of the historical data. The number m is determined by the size of main memory and the size of cluster feature for each subcluster. For example, given the memory available for storing subclusters M and the size of cluster feature M_{cf}, m is computed as $\lfloor \frac{M}{M_{cf}} \rfloor$. In previous work, m is assumed much larger than the number of final cluster k and much less than the total number of data points N, i.e., $k \ll m \ll N$.

2. Generate final k clusters based on m subclusters: In this stage, each subcluster is treated as a weighted point and the final k clusters are computed by any traditional clustering algorithm on these weighted points.

Intuitively, the idea of hierarchical clustering technics is suitable for data streams because it fully utilizes the main memory to reduce the impact of non-random access. The first stage is the most important one. The soundness of stream clustering algorithm depends on whether stage 1 can build m subclusters which well summarize the N points in the data stream.

For the second stage, since the subclusters can fit in the memory, random access is allowed when generating the final clusters. Actually, the cluster analysis at stage 2 is not confined to the partitioning-based clustering problem. In [4], density-based clusters are generated based on subclusters. In their algorithm, DBSCAN [6] is applied at stage 2 and each final cluster is represented by a set of subclusters.

Due to the significance of stage 1, various approaches have been proposed to maintain m subclusters, while all of these studies follow the structure in Figure 2.[1]

At step 1, the initial m subclusters can either be formed by using the first m points or be computed from the first $IniNumber$ points of the data stream. For a given point p, once the closest subcluster C_i is found in step 3, Step 4 decides whether the new cluster needs to be created. Normally, two types of thresholds are applied. One is to evaluate the radius of $C_i \cup \{p\}$ against the radius threshold, such as in [13; 4], another is to compare the distance between p and the center of C_i with a distance threshold, like in [1; 2]. Once the point can be absorbed by the existing cluster C_i, the cluster features of C_i is updated in step 5. Otherwise, a new subcluster with single element p needs to be created. Before that, to maintain the number m remains, two closest clusters needs to be merged. Note that a subcluster can also be deleted if it is believed as an outlier. For simplicity, we do not consider the outlier in the following discussion.

[1]Particularly, in [13], a tree structure is applied to organize the subclusters efficiently.

Input: An integer m and a data stream S
Output: A set of m subclusters represented by cluster features
Method:
1 Initialize cluster features for m subclusters;
2 **For each** point p in S
3 Find the subcluster C_i whose center is closest to p;
4 **If** C_i can absorb p **then**
5 Update the cluster feature of C_i;
6 **Else**
7 Merge two clusters whose centers are closest;
8 Form a new subcluster $\{p\}$ with cluster feature;

Figure 2: High-level structure for subcluster maintenance

3. EVALUATION OF SUBCLUSTER MAINTENANCE APPROACH

In this section, we evaluate whether subcluster maintenance approach can achieve exclusive clustering on data streams (i.e., *Problem 2*). First, let us be aware that for each subcluster, there is no information about individual points in it. So, we refer to a subcluster C as a triple (n, c, r) where n, c, r are the cardinality, center, and radius of C. In the remaining part, we will first discuss that by following the maintenance process at stage 1, what are the possible changes to the center and radius of a single subcluster. Further, we investigate how these changes will impact on the clustering result.

Let us start with the following simple observation.

OBSERVATION 1. *Let a point $p = (p^1, \ldots, p^d)$ and a subcluster C with center c be given. In addition, let c' be the center of $C \cup \{p\}$. c' is equal to c iff $p = c$.*

We consider the condition $c = c'$. Given the cluster feature $\{n, ls^1, \ldots, ls^d, ss\}$ of C, according to Equation 1, we get:

$$\left(\frac{ls^1}{n}, \frac{ls^2}{n}, \ldots, \frac{ls^d}{n}\right) = \left(\frac{ls^1 + p^1}{n+1}, \frac{ls^2 + p^2}{n+1}, \ldots, \frac{ls^d + p^d}{n+1}\right)$$
$$\Longrightarrow p^j = \frac{ls^j}{n} (j = 1 \cdots d)$$
$$\Longrightarrow p = c$$

Observation 1 indicates that after inserting a point in a subcluster, the new center c' will move unless the point p falls on the old center c.

Since in the maintenance algorithm, the new point is assigned to the subcluster whose center is closest, we now investigate how the center of a subcluster changes when continually adding points that are always close to the center.

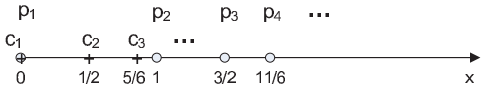

Figure 3: Center movement

Example 1. Let us consider 1-D data points on a line. Suppose that the process of adding new points into a subcluster C is as follows. At moment t_1, a subcluster C is $\{p_1\}$ where $p_1 = 0$ and cluster center $c_1 = 0$. For any the new point p_i arriving at t_i ($i = 2$ to n), let p_i always be $c_{i-1} + \Delta$, where Δ is a real constant and c_{i-1} is the center of C at moment t_{i-1}. As a result, at moment t_n, C is $\{p_1, \ldots, p_n\}$ and c_n is $\sum_{i=1}^{n-1} \frac{\Delta}{i}$ (as shown in Figure 3, where $\Delta = 1$). Clearly, when $n \to \infty$, no matter how small Δ is, we have $\lim_{n\to\infty} c_n = \Delta \lim_{n\to\infty} \sum_{i=1}^{n-1} \frac{1}{i} \to \infty$, indicating that the center of C moves from 0 to infinity.

The above example shows that although at any step, the point that is added in the subcluster is close to the center, the position of center can change significantly.

Observation 2. *During the subcluster maintenance process, center movement may cause subclusters to violate Condition 1. That is, there exist two subclusters C_1 and C_2, such that $\exists\ p_0 \in C_1$, $|p_0 - c_1| > |p_0 - c_2|$, where c_1 and c_2 are the centers of C_1 and C_2 respectively.*

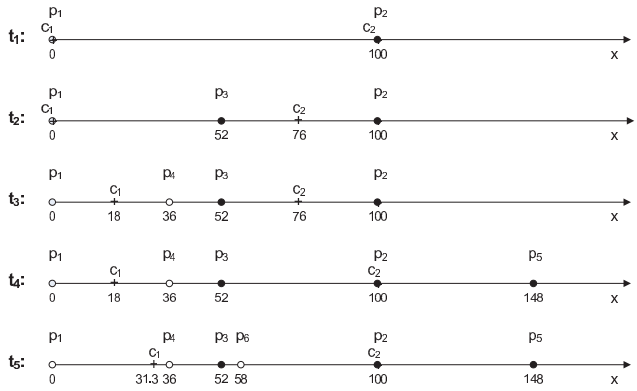

Figure 4: Overlap of clusters

Example 2. Suppose that we maintain two subclusters ($m = 2$). Figure 4 shows the process of how the subclusters evolve. In the figure, the points belonging to two subclusters are denoted as '$\circ$' and '$\bullet$' respectively, and centers are labelled as '+'. At moment t_1, two points p_1 and p_2 are observed and each of them is the center of a subcluster. When a point p_3 arrives at t_2, p_3 is assigned to C_2 (because c_2 is the closest center) and c_2 is updated. With such a process continues, we can see that at moment t_4, p_3 is closer to the center of C_1, but it belongs to subcluster C_2. Note that in such situation, the violation of *Condition 1* could be rectified later if there arrives other points which drive the center c_1 close to the origin. However, after the arrival of points p_6 at t_5, the points in subcluster C_1 and C_2 are interleaved. That is, there does not exist a point p_0 such that $\forall p \in C_1, p < p_0$ and $\forall p \in C_2, p > p_0$. Generally, we say that two sets of points in a metric space are interleaved if there does not exist a hyperplane which can sperate them. According to this definition, we can see that once the points in the subclusters are interleaved, it is not possible to get the subclusters satisfy the *Condition 1* unless two subclusters are merged later.

Example 2 shows the fundamental weakness of the subcluster maintenance approach. That is, the membership of the point in a subcluster depends on the order of observation. This weakness causes that points in a subcluster may not be closely "around" the center. Obviously, because the maintenance result can violate the *Condition 1*, given a point p in S, the membership of p in the subcluster cannot be determined by m cluster centers found in stage 1.

After introducing the weakness of center movement, we discuss the possible change on the radius of a subcluster during the maintenance process.

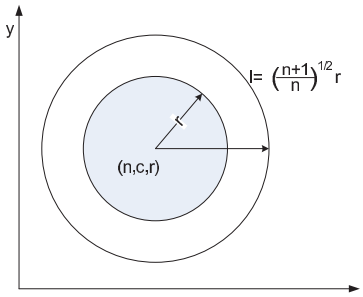

Figure 5: Change of radius

Observation 3. *Let a point p and a subcluster C with the feature $\{n, c, r\}$ be given. The radius r' of $C' = C \cup \{p\}$ is greater than r if the distance between p and the cluster center c is greater than $\left(\frac{n+1}{n}\right)^{\frac{1}{2}} r$.*

Given the cluster feature $\left\{n, ls^1, \ldots, ls^d, ss\right\}$ of C, for brevity of presentation, we assume that the center c is the origin of d-dimensional space, i.e., $ls^j = 0$ for $j = 1, \ldots, d$. According to Equation 2, the radius $r = \left(\frac{ss}{n}\right)^{\frac{1}{2}}$. Given the point $p = \{p^1, \ldots, p^d\}$, the cluster feature of $C \cup \{p\}$ is $\left\{n+1, p^1, \ldots, p^d, ss + \sum_{j=1}^{d} \left(p^j\right)^2\right\}$ and the new radius $r' = \left(\frac{ss+\sum_{j=1}^{d}\left(p^j\right)^2}{n+1} - \frac{\sum_{j=1}^{d}\left(p^j\right)^2}{(n+1)^2}\right)^{\frac{1}{2}}$. Let l be the distance between c and p, i.e., $l = \left(\sum_{j=1}^{d}\left(p^j\right)^2\right)^{\frac{1}{2}}$. Then, we get:

$$r' = \left(\frac{nr^2 + l^2}{n+1} - \frac{l^2}{(n+1)^2}\right)^{\frac{1}{2}} = \left(\frac{nr^2}{n+1} + \frac{nl^2}{(n+1)^2}\right)^{\frac{1}{2}} \tag{3}$$

Considering the inequality $r' > r$, we have:

$$l > \left(\frac{n+1}{n}\right)^{\frac{1}{2}} r$$

Intuitively, Observation 3 indicates that the radius of cluster with $\{n, c, r\}$ will increase if the point is outside the sphere determined by equation $|p - c| = \left(\frac{n+1}{n}\right)^{\frac{1}{2}} r$, as is shown in Figure 5.

Since the radius of a subcluster is likely to increase, some heuristics are applied in the subcluster maintenance approach to control the size of the subclusters. In [13; 4], the radius threshold is applied as follows. For a new arriving point p, when finding its closest cluster C_i, if the radius of $C_i \cup \{p\}$ is greater than a threshold r_0, instead of putting the point into C_i, two closest clusters are merged and a new cluster $\{p\}$ is created. However, this heuristic may have difficulty to control the radii of subclusters because the threshold

r_0 itself has to increase if the radius of the merged subcluster exceeds the current threshold. Intuitively, the less the radius threshold is, the more subclusters needs to be maintained. Given a fixed number of subclusters, with data points continuously arrive, the radius threshold may keep increasing. In [1; 2], merging old clusters and creating a new cluster are required when the distance l between p and the center of C_i is greater than a factor T ($T > 1$) of the RMS deviation of the data points in C_i (i.e., $l > T * r_i$). T is set as 2 in [1; 2]. The above heuristic is based on the distance between the point and its closest cluster center. Although this heuristic does not need to update threshold, it may fail to control the size of subclusters, as shown below.

OBSERVATION 4. *Let a cluster C with $\{n_0, c_0, r_0\}$ ($n_0 \geq 2, r_0 > 0$) be given. In addition, let the n-th point p_n ($n = n_0+1, \ldots$) satisfy the condition $|p_n - c_{n-1}| = T*r_{n-1}$, where T is a real parameter larger than 1, c_{n-1} and r_{n-1} are the center and radius of cluster $\{p_1, \ldots, p_{n-1}\}$. If $T \geq \sqrt{2}$, the radius r_n does not converge when $n \to \infty$.*

Let l_n be the distance between p_n and the center c_{n-1}, i.e, $l_n = T * r_{n-1}$. According to Equation 3, we have

$$r_{n+1} = \left(\frac{nr^2}{n+1} + \frac{nl_n^2}{(n+1)^2}\right)^{\frac{1}{2}} = \left(\frac{n\left(n+1+T^2\right)}{(n+1)^2}\right)^{\frac{1}{2}} r_n$$

For the simplicity of presentation, suppose that $n_0 = 2$ and $r_2 = r_0$, then we have:

$$r_3 = \left(\frac{2\left(3+T^2\right)}{3^2}\right)^{\frac{1}{2}} r_0$$

$$r_4 = \left(\frac{3\left(4+T^2\right)}{4^2}\right)^{\frac{1}{2}} r_3$$

$$\vdots$$

$$r_n = \left(\frac{(n-1)\left(n+T^2\right)}{n^2}\right)^{\frac{1}{2}} r_{n-1}$$

By combining the above equations: we get

$$r_n = \left(\frac{4\,(n-1)!\prod_{i=3}^{n}\left(i+T^2\right)}{(n!)^2}\right)^{\frac{1}{2}} r_0$$

Given $T^2 \geq 2$, we can see

$$\begin{aligned}\lim_{n\to\infty} r_n &\geq \lim_{n\to\infty} \frac{4r_0\,(n-1)!\prod_{i=3}^{n}(i+2)}{(n!)^2} \\ &= \lim_{n\to\infty} \frac{r_0\,(n-1)!\,(n+2)!}{6\,(n!)^2} \\ &= \lim_{n\to\infty} \frac{r_0\,(n+1)\,(n+2)}{6n} \to \infty\end{aligned}$$

Observation 4 indicates that by strictly following the maintenance approach proposed in [1; 2], the radius of a subcluster could increase to infinity. Even when $1 < T < \sqrt{2}$, if T is close to 1, the subclusters need to be merged very frequently, which also causes the uncontrollable increase of radius; if T is close to $\sqrt{2}$, the radius r_n may also significantly increase and converge to a large value.

Until now we have known that when inserting points into a subcluster, the cluster center can shift for a long distance and its radius may increase dramatically. Let us imagine the situation in incremental update process: when a list of points continuously arrive, m subcluster are maintained simultaneously and each subcluster is evolved with such potential changes. A natural concern is: will this lead to the result that subclusters are significantly overlapped, or even subsume others?

OBSERVATION 5. *Following the subcluster maintenance approach (shown in Figure 2), subclusters are likely to be overlapped or to subsume others in terms of the center and radius.*

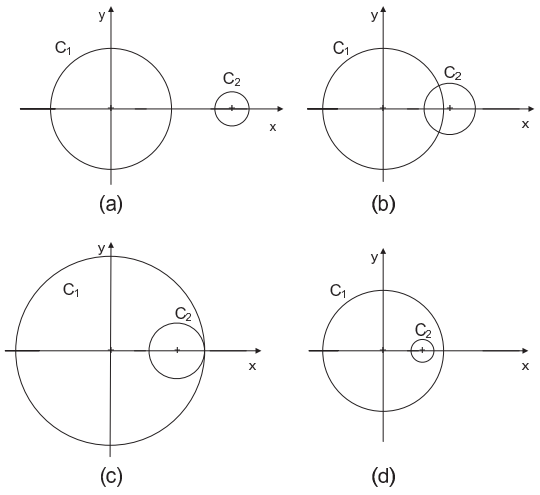

Figure 6: Overlap and subsume between clusters

Example 3. Let us consider two subclusters shown in Figure 6(a). Suppose that subcluster C_1 has a large radius (refer to the Observation 4). According to Example 1, we know that subcluster C_2 can keep moving to the center of C_1 when the significant amount of new arriving points are closer to the center of C_2 than the center of C_1. This will lead to the significant overlap between two clusters, as shown in Figure 6(b). Suppose after the overlap between two clusters, by adding pairs of points on y axis such as $(0, y_i), (0, -y_i)$ into C_1, the radius of C_1 could increase further. As a result, C_1 subsumes C_2 (Figure 6(c)). Note that according to the definition, the radius of clusters may shrink as well. Therefore, it is also possible that C_1 subsumes C_2 due to the center movement and the decrease of radius for C_2, which is shown in Figure 6(d).

To sum up, the clustering result of stage 1, which are m subclusters computed by maintenance, does not satisfy the *requirement 1*, i.e., data points are not partitioned in terms of cluster features. According to Observation 2, the points of subclusters cannot be represented by a region in the Voronoi diagram determined by the centers. Now we consider center and radius together, by which each subcluster can be regarded as a sphere. According to the definition of radius, there could be significant amount of points outside the subcluster in terms of the sphere. Also, subclusters are likely to be overlapped (see Observation 5). As a result, for any point

that falls outside of spheres or in the overlapped regions, its membership to the subcluster is not certain.

After knowing the weakness of subclusters computed in stage 1, we naturally think the following question: Can stage 2 generate the final k clusters, which meet both requirements stated in *Problem 2*? First, we notice that hierarchically grouping subclusters does not guarantee the final k centers to satisfy the *requirement 1*. Actually, in the grouping process, even though the subclusters satisfying the *Condition 1*, the final k clusters can still violate the same condition (this is straightforward when considering the aggregation on the regions of Voronoi diagram). Considering the center and radius of k clusters, it is clear that hierarchical clustering tends to group the subclusters which are overlapped. When the optimal k clusters are *significantly separated* (i.e., the radius of each cluster is much less than the distance between any two centers), the overlapped subclusters found in stage 1 are likely within one of optimal clusters and could be grouped in stage 2. Therefore, in such a case, the subcluster maintenance approach may work effectively. However, in general case, stage 2 cannot guarantee that 1) all overlapped subclusters are merged and 2) any merged subclusters are within one optimal cluster. Therefore, the clustering result cannot satisfy the *requirement 1*. Also, due to the overlap between the cluster features (which means fail to partition data into distinct groups), the *requirement 2* can not be satisfied as well. Based on the above discussion, we can see that although the subcluster maintenance approach may working on specific datasets, in general, it can not resolve the exclusive clustering problem on data streams.

4. CONCLUSION AND DISCUSSION

Clustering on data streams has attracted many attentions from database research community. The work in this area has mainly focused on the exclusive clustering, i.e., partitioning a stream of points into k groups of similar objects, where the similarity is mostly defined by a distance function. In this paper, we have clarified two fundamental questions on this problem. Firstly, we pointed out that as a basic requirement, the clustering result, which is a set of cluster features, must be a true partition of data points. Secondly, we defined the problem of exclusive clustering on data streams and evaluated the existing approaches that are based on the subcluster maintenance.

By using a few straightforward yet powerful examples, we have illustrated that the subcluster maintenance approach may fail to resolve the exclusive clustering on data streams. Particularly, because of a progressive maintenance approach, the membership of a point in a cluster shall depend upon the current subcluster centers and the location of these centers may change significantly later. Therefore along with the time, the subcluster maintenance approach may unavoidably group data points from different optimal clusters. This will result in that 1) representation of clusters may overlap and 2) the value of objective function could be far from the optimum.

Regarding the exclusive clustering problem on data streams, it is obvious that clustering analysis, which task is to group data objects based on their similarity, requires a global picture of data. However, on data streams, one constraint is that data can only be processed in the order as they arrive and are seen only once. This means that any grouping decision made during the clustering on data streams is only based on the data observed so far. Thus, the clustering result is inevitably dependant on the arriving order of data. Due to this contradiction between the nature of clustering problem and the constraint of data stream management, we believe that in general, any heuristic method will face the difficulty to generate good quality clusters.

Finally, during the evaluation of the subcluster maintenance approach, we are aware that the discussed approaches can work effectively when the optimal clusters, which are k intrinsic groups of data points, are *significantly separated*. So, in the existing work, great efforts have been made to address the clustering problem in the context of data streams. A sensible solution has been proposed for resolving the clustering problem for certain types of streaming data. However, due to complexity of the clustering problem and the computation constraints on data streams, it is also clear that the discussed existing methods may not resolve the problem in general.

5. ACKNOWLEDGEMENTS

This work is funded by the Australian ARC Large Grant DP0558879, Mining Distributed, High-Speed, Time-Variant Data Streams, 2005-2007.

6. REFERENCES

[1] C. C. Aggarwal, J. Han, J. Wang, and P. S. Yu. A framework for clustering evolving data streams. In *VLDB*, pages 81–92, 2003.

[2] C. C. Aggarwal, J. Han, J. Wang, and P. S. Yu. A framework for projected clustering of high dimensional data streams. In *VLDB*, pages 852–863, 2004.

[3] B. Babcock, S. Babu, M. Datar, R. Motwani, and J. Widom. Models and issues in data stream systems. In *PODS '02*, pages 1–16, 2002.

[4] F. Cao, M. Ester, W. Qian, and A. Zhou. Density-based clustering over an evolving data stream with noise. In *SDM*, 2006.

[5] M. Charikar, L. O'Callaghan, and R. Panigrahy. Better streaming algorithms for clustering problems. In *STOC '03: Proceedings of the thirty-fifth annual ACM symposium on Theory of computing*, pages 30–39, 2003.

[6] M. Ester, H. Kriegel, J. Sander, and X. Xu. A density-based algorithm for discovering clusters in large spatial databases with noise. In *KDD*, pages 226–231, 1996.

[7] L. Golab and M. T. Ozsu. Issues in data stream management. *SIGMOD Rec.*, 32(2):5–14, 2003.

[8] S. Guha, N. Mishra, R. Motwani, and L. O'Callaghan. Clustering data streams. In *IEEE Symposium on Foundations of Computer Science*, pages 359–366, 2000.

[9] J. MacQueen. Some methods for classification and analysis of multivariate observations. In *Fifth Berkeley Symposium on Mathematical Statistics and Probability*, pages 281–297, 1967.

[10] B. Mirkin. *Clustering For Data Mining A Data Recovery Approach*. Chapman & Hall/CRC, 2005.

[11] L. O'Callaghan, A. Meyerson, R. Motwani, N. Mishra, and S. Guha. Streaming-data algorithms for high-quality clustering. In *ICDE*, pages 685–, 2002.

[12] C. Ordonez. Clustering binary data streams with k-means. In *DMKD '03: Proceedings of the 8th ACM SIGMOD workshop on Research issues in data mining and knowledge discovery*, pages 12–19, 2003.

[13] T. Zhang, R. Ramakrishnan, and M. Livny. Birch: An efficient data clustering method for very large databases. In *SIGMOD Conference*, pages 103–114, 1996.