Dynamic Clustering Of High Speed Data Streams

J. Chandrika[1], Dr. K.R. Ananda Kumar[2]

[1] Department of CS & E, M C E ,Hassan – 573 201
Karnataka, India

[2] Department of CS & E, SJBIT , Bangalore – 560 060
Karnataka, India

Abstract

We consider the problem of clustering data streams. A data stream can roughly be thought of as a transient, continuously increasing sequence of time-stamped data. In order to maintain an up-to-date clustering structure, it is necessary to analyze the incoming data in an online manner, tolerating but a constant time delay. The purpose of this study is to analyze the working of popular algorithms on clustering data streams and make a comparative analysis**.**

Keywords: *Data streams, Unsupervised learning, Partitional clustering, Hierarchical clustering*

1. Introduction

During the recent years, so-called *data streams* [1] have attracted considerable attention in different fields of computer science, such as e.g. databases or distributed systems. As the notion suggests, a data stream can roughly be thought of as an ordered sequence of data items, where the input arrives more or less continuously as time progresses. There are various applications in which streams of this type are produced, such as network monitoring, telecommunication systems, customer click streams, stock markets, or any type of multi-sensor system. A data stream system may constantly produce huge amounts of data. To illustrate, imagine a multi-sensor system with 10,000 sensors, each of which sends a measurement every second of time. As concerned aspects of data storage, management and processing, the continuous arrival of data items in multiple, rapid, time-varying and potentially unbounded streams raises new challenges and research problems. Indeed, it is usually not feasible to simply store the arriving data in a traditional database management system in order to perform operations on that data later on. Rather, stream data must generally be processed in an online manner in order to guarantee that results are up-to-date and that queries can be answered with a small time delay. In this paper, we consider the problem of clustering data streams. Our focus is on time-series data streams, which means that individual data items are real numbers that can be thought of as a kind of measurement. There are numerous applications for this type of data analysis such as, clustering of stock rates.

Apart from its practical relevance, this problem is also interesting from a methodological point of view. Especially, the aspect of efficiency plays an important role: First, data streams are complex objects making the computation of similarity measures costly. Second, clustering algorithms for data streams should be adaptive in the sense that up-to-date clusters are offered at any time, taking new data items into consideration as soon as they arrive.

The remainder of the paper is organized as follows: Section 2 provides some background information, both on data streams and on clustering. Section 3 is given to the clustering of data streams and introduces an online version of the well-known *K*-means Section 4 gives the details of cure algorithm. Finally a comparative analysis is made in section 5.

2. Background

2.1. The Data Stream Model

The *data stream model* assumes that input data are not available for random access from disk or memory, but rather arrive in the form of one or more continuous data streams. The stream model differs from the standard relational model in the following ways [2]:

- The elements of a stream arrive online (the stream is "active" in the sense that the incoming items trigger operations on the data, rather than being send on request).
- The order in which elements of a stream arrive are not under the control of the system.
- Data streams are potentially of unbounded size.
- Data stream elements that have been processed are either discarded or archived.
- They cannot be retrieved easily unless being stored in memory, which is typically small relative to the size of the stream.
- Due to limited resources (memory) and strict time constraints, the processing of stream data will usually produce *approximate* results.

2.2 **Clustering**

Clustering can be considered the most important *unsupervised learning* problem; so, as every other problem of this kind, it deals with finding a *structure* in a collection of unlabeled data. We can show this with a simple graphical example:

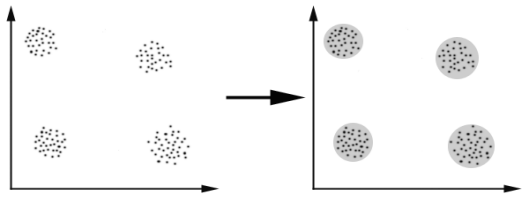


Fig 1. Clustering data in Euclidean space.

Clustering refers to the process of grouping a collection of objects into classes or "clusters" such that objects within the same class are *similar* in a certain sense, and objects from different classes are dissimilar. In addition, the goal is sometimes to arrange the clusters into a natural hierarchy. Also, cluster analysis can be used as a form of descriptive statistics, showing whether or not the data consists of a set of distinct subgroups. Clustering algorithms proceed from given information about the similarity between objects, e.g. in the form of a *proximity matrix*. Usually, objects are described in terms of a set of measurements from which similarity degrees between pairs of objects are derived, using a kind of similarity or distance measure. There are basically three types of clustering algorithms: *Mixture modeling* assumes an underlying probabilistic model, namely that the data were generated by a probability density function, which is a mixture of component density functions. *Combinatorial algorithms* do not assume such a model. Instead, they proceed from an objective function to be maximized and approach the problem of clustering as one of combinatorial optimization. So called *mode-seekers* are somewhat similar to mixture models. However, they take a non-parametric perspective and try to estimate modes of the component density functions directly. Clusters are then formed by looking at the closeness of the objects to these modes which serve as cluster centers. Clustering algorithms can be applied in many fields, for instance:

- *Marketing*: finding groups of customers with similar behavior given a large database of customer data containing their properties and past buying records;
- *Biology*: classification of plants and animals given their features;
- *Libraries*: book ordering;
- *Insurance*: identifying groups of motor insurance policy holders with a high average claim cost; identifying frauds;
- *City-planning*: identifying groups of houses according to their house type, value and geographical location;
- *Earthquake studies*: clustering observed earthquake epicenters to identify dangerous zones;

3.0 Clustering data Streams

The main research area in mining data streams involves developing the techniques that can effectively mine the data streams. The task of mining becomes complex due the certain characteristics of data streams, mentioned in introduction. Clustering data streams is a sub-area of mining data streams. Clustering algorithms arrange a data set into several disjoint groups such that points in the same group are similar to each other and are dissimilar to other groups according to some similarity metrics. In order to use clustering in data streams, the requirements are [3] [4]: generation of overall high-quality clusters without seeing the old data, high quality, efficient incremental clustering algorithms and analysis in multi-dimensional space. A few methods are proposed to summarize the data streams using sketches and then cluster it [5]. There are basically two types of clustering techniques: *Partitional* and *Hierarchical*[7].

Partitional : Given a database of objects, a partitional clustering algorithm constructs k partitions of n the data, where each cluster optimizes a clustering criterion, such as the minimization of the *sum of squared distance from the mean* within each cluster. One of the issues with such algorithms is their high complexity, as some of them exhaustively enumerate all possible groupings and try to find the global optimum. Even for a small number of objects, the number of partitions is huge. That's why, common solutions start with an initial, usually random, partition and proceed with its refinement. A better practice would be to run the Partitional algorithm for different sets of initial _ points (considered as representatives) and investigate whether all solutions lead to the same final partition. Partitional Clustering algorithms try to locally improve a certain criterion. First, they compute the values of the similarity or distance, they order the results, and pick the one that optimizes the criterion. Hence, the majority of them could be considered as greedy-like algorithms.

Hierarchical: Hierarchical algorithms create a hierarchical decomposition of the objects. They are either *agglomerative* (*bottom-up*) or *divisive* (*top-down*). *Agglomerative* algorithms start with each object being a

separate cluster itself, and successively merge groups according to a distance measure. The clustering may stop when all objects are in a single group or at any other point the user wants. These methods generally follow a greedy-like bottom-up merging.

(b) *Divisive* algorithms follow the opposite strategy. They start with one group of all objects and successively split groups into smaller ones, until each object falls in one cluster, or as desired. Divisive approaches divide the data objects in disjoint groups at every step, and follow the same pattern until all objects fall into a separate cluster. This is similar to the approach followed by divide-and-conquer algorithms. We have presented two representative algorithms for clustering data streams in this paper, one is Partitional and the other is hierarchical.

K-means [6][7] is one of the most popular clustering algorithms used. K-Means Technique uses a Partitioning algorithm. Partitioning algorithm constructs various partitions for the data elements and then evaluates them by some criteria. The main reasons for using K-means algorithms are [6] that it is simple to implement, efficient, and the results are easy to interpret and it can work under a variety of conditions. But the disadvantages of using K-means include dependence on initialization, sensitivity to outliers and converging to poor locally optimal solutions. The input for a K-Means clustering algorithm is a data set having n d-dimensional points and k desired number of clusters and the output are three matrices C, R, W containing the centroid, squared distance (variance) and weights for each cluster. K-means is initialized from some random or approximate solution. Every step will assign each point to its nearest cluster and then points belonging to the same cluster are averaged to get new cluster centroids. Every step successively improves cluster centroids until they are stable. This is the standard version of K-Means technique used. It can be summarized with the following steps:

- Clusters are built by assigning each element to the closest cluster center;
- Each cluster center is replaced by the mean of the elements belonging to that cluster.

K means is a prototype based Clustering. It can only be applied to clusters that have the notion of a centre. The algorithm has a space complexity of O (I * K * m * n), where I is the number of iterations, K is the number of clusters, m is the number of dimensions and n is the number of points.

Algorithm K - Means:

Input:

D = $\{t_1,t_2,\ldots t_n\}$ // set of elements
K // Number of desired clusters

Output:

K // set of clusters

K – Means algorithm:

assign intial values for means $m_1,m_2,\ldots m_k$;
repeat:
  assign each item t_i to other cluster which has the closest mean;
  calculate new mean for each cluster;
until convergence criteria is met;

4. CURE (Clustering Using Representatives)

CURE [8] is an efficient algorithm that is more robust to outliers and identifies clusters having non-spherical shapes and wide variances in size. Cure is a hierarchical clustering algorithm. A hierarchical algorithm creates a hierarchical decomposition of the objects. The clustering algorithm starts with each input point as a separate cluster, and at each successive step merges the closest pair of clusters. In order to compute the distance between a pair of clusters, for each cluster, c representative points are stored. These are determined by first choosing c well scattered points within the cluster, and then shrinking them toward the mean of the cluster. The distance between two clusters is then the distance between the closest pair of representative points - one belonging to each of the two clusters. Thus, only the representative points of a cluster are used to compute its distance from other clusters. The c representative points attempt to capture the physical shape and geometry of the cluster. Furthermore, shrinking the scattered points toward the mean by a factor gets rid of surface abnormalities and mitigates the effects of outliers. The reason for this is that outliers typically will be further away from the cluster center, and as a result, the shrinking would cause outliers to move more toward the center while the remaining representative points would experience minimal shifts. The larger movements in the outliers would thus reduce their ability to cause the wrong clusters to be merged.

Input:

D = $\{t_1,t_2,\ldots t_n\}$ // set of elements
K // Desired number of clusters

Output:

Q // Heap containing k-clusters with one entry for each cluster

CURE algorithm:

T = build(D);
Q = heapify(D); // Initially build heap with one entry per item;
repeat
  u = min (Q);
  delete (Q, u.close);
  w = merge(u,v);
  delete (T,u);
  delete (T,v);
  insert (T,w);
  for each x ε Q **do**
    x.close = find closest to x;

if x is closest to w, then
w.close = x;
insert (Q,w);
until number of nodes in Q is k;

We used two data structures namely the KD Tree and Min Heap. Following are the brief description of both of them.

A KD-Tree (short for k-dimensional tree) is a space-partitioning data structure for organizing points in a k-dimensional space. KD-Trees are a useful data structure for several applications, such as searches involving a multidimensional search key In Cure, the KD Tree is initialized during the initial phase of clustering to hold all the points. Later on in the algorithm, we use this tree for nearest neighbor search and finding closest clusters based on representative points of a cluster. When a new cluster is formed, new representative points are added to the KD Trees. The representative points of older clusters are deleted from the tree. KD Tree improves the search of points in k dimensional space from O(n) to O(log n) as it uses binary partitioning across coordinate axes.

Min Heap - A Min Heap is a simple heap data structure created using a binary tree. It can be seen as a binary tree with two additional constraints:

1. The shape property: all levels of the tree, except possibly the last one (deepest) are fully filled, and, if the last level of the tree is not complete, the nodes of that level are filled from left to right.
2. The heap property: each node is lesser than or equal to each of its children.

The Min Heap stores the minimum element at the root of the heap. In Cure, we always merge two clusters at every step. Thus the cluster to be merged would necessary be having the closest distance from another nearby cluster as the heap is created using inter-cluster distance comparisons. We used java.util.PriorityQueue which supports all the Min Heap operations.

5.0 Benefits of CURE over Partitional Algorithms

K-Means (& Centroid based Algorithms) are Unsuitable for non-spherical and size differing clusters The CURE algorithm can handle clusters of arbitrary shapes. An important parameter in the cure algorithm was the Shrink Factor of Representative Points. If we increased it to 1, the algorithm results in clusters of low quality. If the parameter is reduced to 0.1, CURE starts behaving as a centroid based algorithm. Thus for a range of 0.3 to 0.7, CURE identified the right clusters.

The number of Representative Points present in a cluster is also an important parameter. If the cluster is too sparse, it may need more representative points than a compact smaller cluster. We observed that if the number of representative points is increased to 8 or 10, sparse clusters with variable size and density were identified properly. But with increase in representative points, the computation time for clustering increased as for every new cluster formed, new representative points have to be calculated and shrunk.

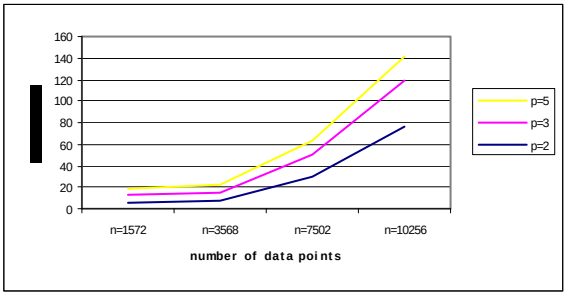
One of the most important observations of our experiments was with respect to partitioning of data sets. As the number of partitions was increased from 2 to 6 or 10, the clustering time dropped significantly. We noticed that if we increased the number of partitions to higher numbers such as 50, the clustering would not give proper results as some of the partitions would not have any data to cluster. Hence, though the time consumed would be lesser, the quality of cluster gets affected and CURE could not identify all the clusters correctly. Some of them got merged to form bigger clusters. Hence, a partitioning of 10 – 20 would result in efficient speed up of algorithm while maintaining the quality of clusters.

Table 1. Partitioning results of CURE Algorithm

No. of Points	1572	3568	7502	10256
Time (in sec)				
Partition P = 2	6.4	7.8	29.4	75.7
Partition P = 3	6.5	7.6	21.6	43.6
Partition P = 5	6.1	7.3	12.2	21.2

The data in the table above is plotted to analyze the clustering behavior of the CURE algorithm. From the graph, we can see that as the partitioning is increased, the time taken to cluster increases very slowly even though the data set size has increased by four times.

From the clusters obtained through various algorithms and the time taken by each algorithm on the datasets, we can say that, K – means is not the best of clustering methods with its high space complexity. For high dimensional data, K – means takes a lot of time and memory. Also it cannot always converge.



Cure could identify all the clusters properly. But CURE depends on some of the user parameters which have to be data specific. The range of such parameters do not vary too much many of them being from 0 – 1. Cure could identify several clusters with high purity which K-means

failed to identify. The comparison between the two algorithms from different perspectives is as shown in the table below:

Table 2. Comparitive Analysis

Algorithm	Input Parameter	Optimized for	Cluster structure	Outlier handling
K means	Number of clusters	Well separated	Spherical Cluster	No
Cure	Number of Clusters & Representatives	Arbitrary Shapes of clusters	Relatively Arbitrary	Yes

6. Conclusion

In this paper we presented major research accomplishments and techniques that have emerged in the field of data stream mining. Data streams have gained ground in the field of research. The research in this field is mainly done in the areas like modeling, query processing, and mining data streams. For instance several papers have been written till now in the field of data stream mining which includes the methods like classification, clustering, and regression analysis. Traditional data stream algorithms are challenged by the feature of data streams. So the conventional techniques for data mining needs to be molded according to he needs of data streams. The properties like infinite data flow and drifting concepts make the life of these researchers tough. To overcome these difficulties we have presented some of the recently developed and experimentally proved approaches for dealing with data streams. The property of data streams that is mainly dealt with these approaches is continuously changing. The application of these approaches, techniques and methods is determined by the problem domain and the properties of problem domain.

References

[1] G Cormode, “Fundamentals of Analyzing and Mining Data Streams”, Workshop On Data Stream Analysis, March, 15-16, 2007

[2] B Babcock, S Babu, M Datar, R Motwani, and J Widom. “Models and issues in data stream systems”, Proceedings of PODS, 2002.

[3] Madjid Khalilian, Norwati Mustapha “Data Stream clustering: Challenges and issues”, Proceedings of the International MultiConference of Engineers and Computer Scientists 2010 Vol I,IMECS 2010, March 17 - 19, 2010, Hong Kong

[4] D. Barbara, "Requirements for clustering data streams," *ACM SIGKDD Explorations Newsletter*, vol. 3, pp. 23-27, 2002.

[5] Charu C. Aggarwal , “A Framework for Clustering Massive-Domain Data Streams”, In Proc. Of IEEE International Conference on Data Engineering DOI 10.1109/ICDE.2009.13

[6] L. O Callaghan, N. Mishra, A. Meyerson, S. Guha, and R. Motwani, "Streaming-data algorithms for high-quality clustering," 2002.

[7] YI-HONG LU1, YAN HUANG, “Mining Data Streams Using Clustering”, Proceedings of the Fourth International Conference on Machine Learning and Cybernetics, Guangzhou, 18-21 August 2005

[8] S. Guha, R. Rastogi, and K. Shim. CURE: An Efficient Clustering Algorithm for Large DataBases. In Proc. of the ACM SIGMOD Intl. Conference on Management of Data, 1998.