Clustering Data Streams

Mohamed Elasmar
gtg091e@mail.gatech.edu

Prashant Thiruvengadachari
tprashant@gmail.com

Javier Salinas Martin
javisal1@gatech.edu

Introduction:

Data mining is the science of extracting useful information from large sets of data. A commonly used data mining technique is clustering, the classification of objects into different groups by partitioning sets of data into a series of subsets (clusters). An example of clustering is the tracking of network data used to study changes in traffic patterns and to detect possible intrusions. In this example, additionally, data must be processed as it is produced.

In the last few years, clustering has been extensively studied and a great deal of algorithms has been proposed in order to extract subsets from data sets. These algorithms commonly seek to group data with maximum similarities inside a same cluster and have minimum similarities between different clusters. At the same time, the usage of data streams to transmit relevant information, such as periodic updates of a person's localization, is becoming extremely widespread. Extracting information from data streams is, therefore, extremely useful and applicable to many aspects in the real world.

Motivation and objectives:

As a part of the requirements of the CS4440 course at Georgia Tech, we plan to utilize an efficient clustering algorithm to process incoming streams of data and extract useful patterns that we can use in the near future. Clustering is a very largely employed technique and, as such, our effort in getting better clusters needs to be directed towards testing new methods and processing reactions based on the results. Our proposed clustering algorithm will seek to be efficient in extracting high quality clusters of interest while keeping in mind the importance of data arrival and access. It should overcome most known problems in clusters, such as outliers, different sizes, spherical and non-spherical shapes, densities… Typically, we will use some of the most outstanding ideas from already existing clustering algorithms to process data streams as they arrive and identify clusters of interest, in order to take decisions based on the results.

Clustering data streams is commonly a difficult task. Anyhow, and despite traditional clustering algorithms are extremely inefficient when working with data streams, more adequate one-pass algorithms have been developed in the last few years. Though these algorithms are acceptably efficient, their quality generally diminishes when data evolves greatly over time. If clusters are not properly updated as time passes by, the algorithms might become dominated by the outdated history of the stream. Therefore, better results will be obtained by exploring the stream over different time windows, for the clusters will evolve consequently with the incoming data.

Summing up, this project is based on creating a new clustering algorithm for data streams able to keep up to date with the evolution of the incoming data. While we cannot

guarantee good results at this time, we believe that our clustering mechanism should be able to do well given the conceptual thought gone into its working. After implementing our algorithm, we plan to run it against several test cases proposed for some already existing algorithms and compare the clusters obtained with those returned by them.

Related Work:

A lot of effort has been put into data clustering for quite some time. Clustering has taken its roots from algorithms like K-Means and PAM and has come to an advanced stage with more modern algorithms like BIRCH, CUBE, Chameleon or CURE. While all the former use the general distance measure for computing cluster similarity, Chameleon takes into account intra-cluster similarity, and CURE uses multiple representative points to ensure that complex cluster shapes are also detected.

We can attribute certain general procedures which these clustering algorithms follow:

1. Most clustering algorithms use the distance measure for computing cluster similarity.
2. Clustering algorithms can follow a hierarchical or a partitioning model. Both use some specific points within the cluster to represent it in order to find out similarities.
3. Chameleon takes into account intra-cluster similarity in addition.
4. CURE uses multiple cluster representatives and clusters based on existing cluster shape.

A considerable effort has also been put on computing data streams. Apart from solutions to some basic problems, like frequency estimation or order statistics, several clustering algorithms have been proposed. However, there are some challenges while clustering data streams which do not arise in normal clustering algorithms:

1. The amount of stored data must be reduced.
2. Incoming data must be incrementally clustered.
3. Changes in existing clusters must be rapidly identified. At the same time, new clusters must be formed and some already existing clusters must be deleted.
4. Previous cluster data must be summarized to aid in the clustering mechanism.

Out of the already existing data stream clustering algorithms, the most representative are the following:

- STREAM: this algorithm applies a partitioning approach and then clusters bottom-up, in a fashion similar to CURE. However, whereas CURE seeks robustness and clustering of arbitrary shapes, STREAM simply tries to produce a provably good clustering.
- CluStream: this algorithm is divided into a statistical data collection component and an on-line analytical component based on a pyramidal time window. CluStream claims to achieve a higher accuracy than STREAM thanks to the exploitation of micro-clustering techniques.
- ODAC: this variable clustering algorithm incrementally constructs a tree-like hierarchy of clusters, using a correlation based dissimilarity measure between time series over a data stream.

However, these algorithms do present some issues. STREAM fails to take into account old data points while forming new clusters and, as such, cannot recognize possible changes in cluster formation (since they do not maintain any history). CluStream, on the other hand, does take into account the clusters got at previous instants of time, but provides a very trivial mechanism to cluster data (incremental clustering done by comparing the point location with the previous clusters got and the point is added to the cluster based on a threshold value set). Last of all, ODAC is oriented towards clustering variables, rather than towards clustering data itself. We plan to create a new clustering algorithm that overcomes all these limitations.

Proposed Work:

Initially, we plan to design and implement a new data stream clustering algorithm of our own. We will seek to detect some of the most positive aspects of the preexisting algorithms mentioned above and group them in a single, more efficient, clustering algorithm.

Though we will decide the exact behavior of our algorithm after analyzing preexisting ones more closely, our initial idea will be to base our work on most of CluStream's features. The reason for this is that in [1], experimental results prove that CluStream is more efficient than previous data stream clustering algorithms. However, we believe that some improvements are yet to be done on ClusTream in order to make it truly efficient.

At the same time, we might consider merging Chameleon and CURE. By employing data-shrinking, such as in CURE, and taking the intra-cluster similarity measure, such as in Chamaleon, we would be able to take the goodness of both the algorithms and provide clusters that either of both could obtain individually. However, this is just an idea and there are no guarantees as to whether our proposed method will work better than the existing algorithms for clustering data streams.

Therefore, what we would consider most interesting is to adapt CluStream to the CURE-Chamaleon-based algorithm, moving away from a centroid-based approach and using point compression and the intra-cluster similarity method in order to calculate distances between clusters. That way, we could obtain a more realistic approximation of distances and, at the same time, reduce storage size by simply keeping record of a set of representative points for each cluster.

Existing clustering algorithms for data streams also fail to handle and manage old data efficiently. While we do not want to maintain all the data collected, we would like to keep a summary which would be capable of storing certain information about the data clusters that existed at a particular instant of time Ti. Using this, we would then be able to evaluate clusters in a better fashion at time Tj (j>i).

It has also become necessary for data stream clustering algorithms to work efficiently and fast so that clustering results are readily available when new data arrives. In order to achieve this, it is necessary that the algorithm does not spend time on unwanted processing on going through all the data points to find possible cluster matches.

We are looking at possibilities to optimize this aspect of clustering algorithms by taking into grouping the cluster points into possible ‘herds’. As such, points within a herd are more likely to stay together and hence cluster evaluation need not be done over the entire ‘herd’ at every step. This technique can be applied, for example, in traffic monitoring systems.

Another approach would be to take into account additional information about the domain being clustered. For example, in the traffic domain, by using the ‘final destination’ of the various vehicles, we can group the vehicles together within the same cluster since they would be typically taking the same path until they reach their destination. Further techniques being contemplated are based on looking at previous clusters to compute the formation of new clusters and taking into account the cluster-shape capturing feature of CURE to recognize clusters with complex shapes.

As such, the approaches proposed in the previous paragraph have not been yet implemented altogether (to the best of our knowledge) and will hopefully establish itself as yet another clustering algorithm with some advantages over traditional algorithms and over existing data stream clustering algorithms like BIRCH, Stream, and CluStream..

Plan of Action:

Our algorithm will be implemented in the Linux environment, in a yet to be determined programming language (Java, C, C++…). Compilers for these languages are available in any of the computers in the CoC, being accessible to students.

Week 1	Study of already existing algorithms
Week 2	Design of our own algorithm
Weeks 3-5	Implementation of our algorithm
Week 6	Evaluation of our algorithm
Week 7	Documentation

Evaluation and Testing Method:

Evaluation and testing of clustering algorithms takes a different form in each of the studied papers. Typically, new algorithms try to implement clustering on the same data set used by the algorithms with which it is compared to generate results of the same nature. As such, it would make sense to evaluate our clustering algorithm also on existing data sets. The dataset used by popular data stream clustering algorithms like STREAM and CLUSTREAM use the network intrusion dataset generated during KDD-99 by MIT. This data set is available online.

Additional testing for domains of interest will also be done based on the time available. We will be trying to generate meaningful traffic data sets which can be used by the clustering algorithm. These clusters can then be used to study the traffic generated (which clusters disintegrate at which location, popular points at which clusters merge, etc). The evaluation done here will be purely subjective and the quality of goodness of the cluster will be based on human verification.

More ways to evaluate the clustering algorithm would be contemplated during the course of the project based on its implementation.

References:

[1] C. Charu, H. Jiawei et al: A Framework for Clustering Evolving Data Streams. *Conference* VLDB 2003

[2] G. Sudipto, M.Adam et al: Clustering Data Streams: Theory and Practice. *Conference* IEEE Symposium 2000

[3] Tian Zhang, Raghu Ramakrishnan et al: BIRCH: An Efficient Data Clustering Method for Very Large Databases, Technical Report, Computer Sciences Dept., Univ. of Wisconsin-Madison, 1995.

[4] Keke Chen and Ling Liu: iVIRBRATE: Interactive Visualization Based Framework for Clustering Large Datasets. ACM Transactions on Information Systems.

[5] George Karypis, Eui-Hong Han et al: Chameleon: A hierarchical Clustering Algorithms Using Dynamic Modeling IEEE computer. 1999

[6] S. Guha, R. Rastogi et al: CURE: An efficient clustering algorithm for large databases. *Conference:* International conference on Management of Data. 1998