International Journal of Knowledge-based and Intelligent Engineering Systems 13 (2009) 39–44
DOI 10.3233/KES-2009-0168
IOS Press

Clustering data stream: A survey of algorithms

Alireza Rezaei Mahdiraji
Multimedia University, Cyberjaya, Malaysia
E-mail: alireza.rezaei.mah07@mmu.edu.my

Abstract. A data stream is a massive, continuous and rapid sequence of data elements. The data stream model requires algorithms to make a single pass over the data, with bounded memory and limited processing time, whereas the stream may be highly dynamic and evolving over time. Mining data streams is a real time process of extracting interesting patterns from high-speed data streams. Mining data streams raises new problems for the data mining community in terms of how to mine continuous high-speed data items that you can only have one look at. Clustering, useful tool in data mining, is the process of finding groups of similar data elements which are defined by a given similarity measure. Over the past few years, a number of clustering algorithms for data stream have been put forth. In this paper, we survey five different algorithms for clustering data stream. These algorithms consist divide and conquer, doubling, statistical grid-based, STREAM and CluStream. We compare these algorithms based on several different characters.

Keywords: Data stream, mining data stream, clustering, K-center, K-median, statistical grid-based

1. Introduction

Research on data stream is motivated by emerging applications involving massive data sets such as customer click streams, telephone records, multimedia data, and sensor data. A data stream is defined as a massive unbounded sequence of data elements continuously generated at a rapid rate. Due to this reason, it is impossible to maintain all elements of a data stream. Consequently, on-line data stream processing should satisfy the following requirements. First, each data element should be examined at most once to analyse a data stream. Second, memory usage for data stream processing should be confined finitely although new data elements are continuously generated in a data stream. Third, newly generated data elements should be processed as fast as possible to produce the up-to-date analysis result of a data stream, so that the result can be instantly utilized upon request [8].

Traditional data mining methods cannot be easily applied to the data stream domain due to characteristics of data streams. Mining data streams is a real time process of extracting interesting patterns from high-speed data streams. Mining data streams raises new problems for the data mining community in terms of how to mine continuous high-speed data items that you can only have one look at. A general framework for mining data stream requires small constant time per data point, must use only fixed amount of memory, using at most one scan of data, the model should be up-to-date and include information from the past that has not become outdated.

In this review paper, we discuss five data stream clustering algorithms and compare and contrast these algorithms. The rest of this paper is organized as follows. In Section 2 we survey five clustering algorithms. We compare and contrast these algorithms in Section 3. Summary and discussions are presented in Section 4.

2. Data stream clustering algorithms

Clustering is the process of finding groups of similar data elements which are defined by a given similarity measure. Most conventional clustering algorithms assume that a data set is fixed and focus on how to

ISSN 1327-2314/09/$17.00

minimize processing time or memory usage. While in data stream circumstance, the one pass constraint and the limitation of storage resource challenge traditional clustering algorithms, designed for static database mining. The one pass constraint means that the elements in data stream can be accessed only once except explicitly stored. The limitation of storage resource means that not all the elements in the data streams can be stored even if some of them can be cached. Clustering algorithms designed for static database mining task have to be modified to accommodate these constraints under data stream environment. Many approaches have been proposed for data stream clustering. In this section, we present five algorithms for clustering data streams.

2.1. Divide and conquer algorithm

Divide and conquer algorithm [11] is a data stream clustering algorithm using K-median technique.K-median selects k cluster center from n data points, and then assigns each new point to selected center that is closest to it. Formally, the k-median problem is to find k clusters such that the sum squared error (SSQ) between the points and the cluster center to which they are assigned is minimized. After assigning each point to its nearest cluster, points belonging to the same cluster are averaged to produce new cluster centroids. K-median is a NP-hard problem and there is no tractable optimal solution for it.

Divide and Conquer algorithm makes a single pass over the data stream and uses small space. The final solution of algorithm is a constant-factor approximation of optimal solution (i.e. approximation ratio is bounded by a constant).

Initially, data stream is partitioned into disjoint pieces. Then each piece is clustered to k centers and each point in the piece is assigned to its closest center. The centers obtained from clustering all pieces with their weights, the number of points assigned to them, are saved in the main memory. In the next step, centers obtained from step 1 are clustered again.

The algorithm calculates size of pieces so that each piece can be fitted in the available main memory. When the size of data stream is very large, it may be impossible to find such size. To solve this problem, algorithm recursively calls itself. Recursive call may increase the number of levels used by algorithm and as a result approximation factor will increase.

2.2. Doubling algorithm

Agglomerative hierarchical clustering is a bottom-up strategy starts by placing each object in its own cluster and then repeatedly merges the closest pair of clusters into larger and larger clusters, until all of the objects are in a single cluster or until certain termination conditions are satisfied. Many hierarchical agglomerative clustering (HAC) heuristics exist.

Doubling [5] is an incremental hierarchical agglomerative clustering algorithm based on K-center technique for data stream. In K-center, the goal is to minimize inter-cluster distance. In other words, the problem in K-center is to find k cluster centroids that minimize the distances between any data point and chosen centers.

At the start of i-th phase of this algorithm there are $k+1$ clusters $C_1, C_2, \ldots, C_{k+1}$ and a lower bound d_i is the optimal clustering's diameter (denoted by OPT). Each cluster C_i has a center c_i which is one of the points of the cluster. In addition the following invariants are assumed at the start of i-the phase: (**a**) for each cluster C_i, the radius of C_i, defined as the maximum distance of its points to its center, is at most $2d_i$. (**b**) for each pair of clusters inter-center distance of them are greater than or equal to d_i. (**c**) d_i is less than or equal to OPT.

Each phase of algorithm consists of two stages: (**1**) **merging stage**: in this stage all centers that their distance is less than $2d_i$ will be merged. Due to this fact the radiuses of clusters will increase. When there are no more centers for merging, the merge stage will be terminated. (**2**) **update stage:** in this stage the algorithmtries to maintain at most k clusters without increasing radius of the clusters or violating the invariants. For each new point, if the distance of this new point from one of the centers be less than $2d_i$ then this new point will be assigned to that cluster otherwise a new cluster will be created for this new point.

This algorithm overcomes the problem of increasing approximation factors in divide and conquer algorithm with the increase in the number of levels used to result final solution.

2.3. Grid-based data stream clustering algorithm

Several algorithms were proposed for grid based clustering such as STING [13], Wave Cluster [10] and Statistical Grid based approach [7]. The algorithm was proposed in [7] is a grid based clustering approach for data stream. In order to finding data stream clusters, this algorithm maintains the distribution statistics of

data stream elements. Keeping distribution statistics of data stream elements in a dynamic partitioned grid-cell enables the algorithm to find data stream clusters without maintaining the data stream elements physically.

Some important concepts in this algorithm include unit cell and support. Unit cell is a cell whose length in each dimension is less than a predefined distance parameter. Support is the ratio of number of data elements inside the cell to total number of data elements from data stream seen so far.

Initially, the multidimensional data space of a data stream is partitioned into a set of disjoint and equal-size initial cells. When a new data element arrives, each initial cell monitors the distribution statistics of data elements within the range. When the support of a cell becomes high enough, the cell is dynamically divided into disjoint intermediate cells based on its distribution statistics. Similarly, when support of an intermediate cell becomes high enough, it is partitioned in the same way. The partitioning process is recursively continued until the smallest cell called a unit cell is produced. A cluster of a data stream is a group of adjacent dense unit cells.

Grid based algorithm automatically removes outliers and its time complexity is usually $O(N)$. These algorithms are useful for large data set with high density and they are inappropriate for low density data set.

New version of grid-based clustering algorithm was proposed in [8]. The new algorithm, grid-based subspace clustering algorithm, is proposed to effectively extract the on-going change of a data stream with respect to all the subsets of the dimensions of the data stream.

2.4. STREAM algorithm

STREAM [12] is a single-pass, constant factor approximation algorithm that was developed based on the k-median problem. As we discussed for Divide and Conquer algorithm, the k-median tries to find k clusters with minimum SSQ between the points and the cluster center to which they are assigned.

STREAM algorithm divides data stream into buckets of m data points, with each bucket fitting in main memory and then processes result buckets. For each bucket, b_i, STREAM clusters the bucket's points into k clusters by using LOCALSEARCH algorithm. LOCALSEARCH is linear in the number of data points. It summarizes information of each bucket by maintaining only the information regarding the bucket centers and their weights and then discards the points. Weight of each cluster center is the number of points assigned to the cluster. Once m centers have been collected, these centers are again clustered to produce another set of $O(k)$ cluster centers, e.g. $2k$ center. These intermediate centers belong to second level of algorithm.

Different steps of STREAM algorithm are as below:

1. Input the first $m$points to main memory. Cluster these points by using LOCALSEARCH to obtain $2k$ intermediate centers. Assign to each intermediate center its appropriate weight.
2. Repeat the step 1 till we have seen $m^2/(2k)$ of the original data points. At this point we have m intermediate medians.
3. Cluster these m first-level centers into $2k$ second-level centers.
4. Repeat above steps to see the entire data stream.

After seeing the entire data original data stream, cluster all the intermediate centers into k final centers.

STREAM tries to optimize SSQ criteria, so this algorithm tends to create spheral cluster and cannot recognize clusters with different shapes. Due to this fact, STREAM is not appropriate for data sets that have clusters with different size and shape.

Cluster representation in STREAM is concise and only maintains cluster centroids and their weights. Although, LOCALSEARCH is a linear algorithm, but experiments in [12] show that its time complexity for real data set is more than *k-means* algorithm.

In STREAM there is no solution for detecting outliers. So, if there is enough number of outliers in a piece of data stream, the result centers will be different from other centers of other pieces. When STREAM clusters these centers with other centers by using LOCALSEARCH, it may hide the change in trend of stream, so the output clusters do not show the evolution of stream data.

Moreover, STREAM does not consider time granularity. The clustering can become dominated by the older, outdated data of the stream. In real life, the nature of the clusters may vary with both the moment at which they are computed, as well as the time horizon over which they are measured. For example, a user may wish to examine clusters occurring last week, last month, or last year. Therefore, a data stream clustering algorithm should also can to compute clusters over user-defined time periods in an interactive manner [4]. The CluStream algorithm in next section, addresses these concerns.

2.5. CluStream algorithm

CluStream [1] is a framework for clustering of evolving data streams based on user-specified, on-line clustering queries. This framework can solve some problems in previous algorithms. The most important problem is that computed clusters over the entire data stream have less quality and cannot reflect change of data stream over time.

The main idea in CluStream is to divide the clustering process into on-line and offline components. The on-line component computes and stores summary statistics about the rapid data stream using micro-clusters, and performs incremental on-line computation and maintenance of the micro-clusters. The off-line component does macro-clustering and answers various user questions using the stored summary statistics, which are based on the tilted time frame model.

To cluster evolving data streams based on both historical and current stream data information, the tilted time frame model (such as a progressive logarithmic model) is adopted, which stores the snapshots of a set of micro-clusters at different levels of granularity depending on recency. Snapshots are classified into different orders which can vary from 1 to log(T), where T is the clock time elapsed since the beginning of the stream. The main idea here is that more information will be needed for more recent events in compare to older events. The stored information can be used for processing history-related, user-specific clustering queries.

A micro-cluster in CluStream is represented as a clustering feature. A micro-cluster for a set of d-dimensional points $X_1, \ldots, X_n$ with timestamps $T_1, \ldots, T_n$ is defined as the $(2d+3)$ tuple ($CF2^x$, $CF1^x$, $CF2^t$, $CF1^t$, n), wherein $CF2^x$ and $CF1^x$ are vectors of d entry and $CF2^t$, $CF1^t$, and n are scalars. For each dimension, $CF2^x$ maintains the sum of the squares of the data values. Thus, $CF2^x$ contains d values. Similarly, for each dimension, the sum of the data values is maintained in $CF1^x$. The sum of squares of the time stamps is maintained in $CF2^t$. The sum of the time stamps is maintained in $CF1^t$. Finally, the number of data points in the micro-cluster is maintained in n.

Clustering features have additive and subtractive properties that make them very useful for data stream cluster analysis. For example, two micro-clusters can be merged by adding their respective clustering features. Furthermore, a large number of micro-clusters can be maintained without using a great deal of memory. Snapshots of these micro-clusters are stored away at key points in time based on the tilted time frame.

The on-line micro-cluster processing is divided into two phases: (1) statistical data collection and (2) updating of micro-clusters. In the first phase, a total of q micro-clusters, $M_1, \ldots, M_q$, are maintained, where q is usually significantly larger than the number of natural clusters and is determined by the amount of available memory. In the second phase, micro-clusters are updated. Each new data point is added to either an existing cluster or a new one. To decide whether a new cluster is required, a maximum boundary for each cluster is defined. If the new data point falls within the boundary, it is added to the cluster; otherwise, it is the first data point in a new cluster. When a data point is added to an existing cluster, it is "absorbed" because of the additive property of the micro-clusters. When a data point is added to a new cluster, the least recently used existing cluster has to be removed or two existing clusters have to be merged, depending on certain criteria, in order to create memory space for the new cluster.

The off-line component can perform user-directed macro-clustering or cluster evolution analysis. Macro-clustering allows a user to explore the stream clusters over different time horizons. A time horizon, h, is a history of length h of the stream. Given a user-specified time horizon, h, and the number of desired macro-clusters, k, macro-clustering finds k high-level clusters over h.

Cluster evolution analysis looks at how clusters change over time. Given a user-specified time horizon, h, and two clock times, t_1 and t_2 (where $t_1 < t_2$), cluster evolution analysis examines the evolving nature of the data arriving between $(t_2 - h, t_2)$ and that arriving between $(t_1 - h, t_1)$. This involves answering questions like whether new clusters in the data at time t_1 were not present at time t_2, or whether some of the original clusters were lost. This also involves analyzing whether some of the original clusters at time t_1 shifted in position and nature. Such evolution analysis of the data over time can be used for network intrusion detection to identify new types of attacks within the network.

CluStream was shown to derive high-quality clusters, especially when the changes are dramatic. Moreover, it offers rich functionality to the user because it registers the essential historical information with respect to cluster evolution. The tilted time frame along with the micro-clustering structure allow for better accuracy and efficiency on real data. Finally, it maintains scalability in terms of stream size, dimensionality, and the number of clusters.

A number of experiments on real datasets have been conducted to prove the accuracy and efficiency of the

Table 1
Comparison of Five Stream Clustering Algorithms

Algorithm Property	Divide and Conquer	Doubling	Statistical Grid-based	STREAM	CluStream
Approsimation of Optinal Solution	Yes	Yes	No	Yes	No
Analysis of Stream Evolution	No	No	No	No	Yes
Outlier Detection	No	No	Yes	No	Yes
Discovering Clusters with any Shapt	No	No	Yes	No	No
Pre-determined Numver of Clusters	Yes	Yes	Yes	Yes	No
Support for High Dimensional Data	No	No	No	Yes	Yes
Memory Requirements	High	Moderate	Low	Low	Low

CluStream algorithm [1]. They have proposed HPStream; a projected clustering for high dimensional data streams. HPStream has outperformed CluStream in recent results [2].

HClustream is a novel approach based on CluStream algorithm for clustering data stream with heterogeneous features. Many real-world data streams have both continuous attributes and categorical attributes, which are usually called heterogeneous attributes. However, most of the existing stream mining algorithms can manipulate only continuous attributes or categorical attributes. Simply omitting categorical or continuous attributes may lose important information about the data stream and decrease the mining quality. HClustream, is a novel approach for clustering data stream with different attributes, continuous attributes or categorical attributes [3].

3. Comparison of five clustering algorithms

We discussed about the five algorithms for clustering data stream in previous sections. All of these algorithms suffer from at least one of the below shortcomings:

1. Most of these algorithms are unable to track the change in trend of data stream (except that CluStream).
2. These algorithms cannot manipulate categorical data.
3. Most of them work with a pre-determined number of clusters. In large data streams that have significant changes over times, it is impossible for user to determine this parameter.
4. Most of them tend to create spheral clusters (except that Grid-based algorithm).
5. Most of them are designed for low-dimensional data and cannot adapt themselves with growth of dimensionality.

In this section we compare these algorithms based on seven different characters, namely approximation of optimal solution, analysis of stream evolution, outlier detection, discovering clusters with any shape, predetermined numbers of clusters, support for high dimensional data and memory requirements. Some of these characters are clear and we just give more definition about two of them, namely, outlier detection and support for high dimensional data.

A dataset may contain data objects that do not comply with the general behavior or model of the data. These data objects are outliers. Most data mining methods discard outliers as noise or exceptions. However, in some applications such as fraud detection, the rare events can be more interesting than the more regularly occurring ones. The analysis of outlier data is referred to as outlier mining. Outliers may be detected using statistical tests that assume a distribution or probability model for the data, or using distance measures where objects that are a substantial distance from any other cluster are considered outliers. Rather than using statistical or distance measures, deviation-based methods identify outliers by examining differences in the main characteristics of objects in a group [4].

Most clustering methods are designed for clustering low-dimensional data and encounter challenges when the dimensionality of the data grows really high (say, over 10 dimensions, or even over thousands of dimensions for some tasks). This is because when the dimensionality increases, usually only a small number of dimensions are relevant to certain clusters, but data in the irrelevant dimensions may produce much noise and mask the real clusters to be discovered. Moreover, when dimensionality increases, data usually become increasingly sparse because the data points are likely located in different dimensional subspaces. When the data become really sparse, data points located at different dimensions can be considered as all equally distanced, and the distance measure, which is essential for cluster analysis, becomes meaningless. To overcome this difficulty, we may consider using feature (or at-

tribute) transformation and feature (or attribute) selection techniques. The high-dimensional case presents a special challenge to clustering algorithms even in the traditional domain of static data sets [4].

Table 1 presents theses characters and comparison between these algorithms based on these characters.

4. Conclusions

Traditional data mining are challenged by the features of data streams. So the conventional techniques for data mining need to adapt to the requirements of data stream. The several research papers in the field of mining data stream have been written till now. Data stream clustering is an important issue in data stream mining. In this paper, we presented the important algorithms for clustering data stream. We discussed about five different algorithms and compared different characters of these algorithms. Most of discussed algorithms for clustering data stream need to adapt themselves with features such as change detection in data stream, manipulating heterogeneous data (numerical and categorical data), automatic detection of number of clusters, high-dimensional data and detecting clusters with different shapes.

Data stream mining is a fast-evolving research field that has raised challenges and research issues for database and data mining communities. With the massive amount of data streams populating many applications, it is expected that many new stream data mining methods will be developed, especially for data streams containing additional semantic information, such as time-series streams, spatiotemporal data streams, and video and audio data streams. Mining data streams in sensor network and distributed environment are other open research issues.

References

[1] C.C. Aggrawal, J. Han, J. Wang and P.S. Yu, *A Framework for Clustering Evolving Data Streams*, Proceedings of the 29th VLDB Conference, 2003.

[2] C.C. Aggrawal, J. Han, J. Wang and P.S. Yu, *A Framework for Projected Clustering of High Dimensional Data Streams*, Proceeding of the 30th VLDB Conference, 2004.

[3] C. Yang and J. Zhou, *HCluStream: A Novel Approach for Clustering Evolving Heterogeneous Data Stream*, Sixth IEEE International Conference on Data Mining, 2006.

[4] J. Han and M. Kamber, *Data Mining: Concepts and Techniques*, The Morgan Kaufmann Publishers, Series in Data Management Systems, San Francisco, 2006.

[5] M. Charikar, C. Chekur, T. Feder and R. Motwani, *Incremental Clustering and Dynamic Information Retrieval*, Proc. STOC, 1997.

[6] M. Medhat Gabet, A. Zaslavsky and S. Krishnaswamy, Mining Data Streams: A Review. In: ACM SIGMOD Record, 2005.

[7] Nam Hun Park, *Won Suk Lee.: Statistical Grid based Clustering over Data Streams*, In: ACM SIGMOD Record, 2004.

[8] Nam Hun Park, *Won Suk Lee.: Grid-based Subspace Clustering over Data Streams*, In: Proceedings of the sixteenth ACM conference on Conference on information and knowledge management, 2007.

[9] Pavel Berkhin, *Survey of Clustering Data Mining Techniques*, In: Accrue Software, 2003.

[10] G. Seikholeslami, S. Chatterjee and A. Zhang, *WaveCluster: A Multi-Resolution Clustering for Large Spatial Databases*, In: Proc. Of the 24th VLDB, 1998.

[11] Sudipto Guha, *Adam Meyerson, Nina Mishra, Rajeev Motwani*, Liadan O'Callaghan.: Clustering Data Streams. In: FOCS, 2000.

[12] Sudipto Guha, Adam Meyerson, Nina Mishra, Rajeev Motwani and Liadan O'Callaghan, *Clustering Data Streams: Theory and Practice*, In: IEEE Transactions on Knowledge and Data Engineering, 2003.

[13] W. Wang, J. Yang and R. Muntz, *STING: A Statistical Information Grid Approach to Spatial Data Mining*, In: VLDB'97, 1997.

[14] Yi-Hong Lu and Yan Huang, *Mining Data Stream Using Clustering*, In: Proceedings of Fourth International Conference on Machine Learning and Cybernetics, 2005.