Clustering on Demand for Multiple Data Streams

Bi-Ru Dai, Jen-Wei Huang, Mi-Yen Yeh, and Ming-Syan Chen
Department of Electrical Engineering
National Taiwan University
Taipei, Taiwan, ROC
E-mail:mschen@cc.ee.ntu.edu.tw, {brdai, jwhuang, miyen}@arbor.ee.ntu.edu.tw

Abstract

In the data stream environment, the patterns generated by the mining techniques are usually distinct at different time because of the evolution of data. In order to deal with various types of multiple data streams and to support flexible mining requirements, we devise in this paper a Clustering on Demand framework*, abbreviated as* COD framework*, to dynamically cluster multiple data streams. While providing a general framework of clustering on multiple data streams, the COD framework has two major features, namely* one data scan for online statistics collection *and* compact multi-resolution approximations*, which are designed to address, respectively, the time and the space constraints in a data stream environment. Furthermore, with the multi-resolution approximations of data streams, flexible clustering demands can be supported.*

1 Introduction

In recent years, several query problems and mining capabilities have been explored for the data stream environment [2], including those on the statistics [3], the aggregate query [4], association rules [8], frequent patterns [10], data clustering [1][5][9], and data classification [6], to name a few. For data stream applications, the volume of data is usually too huge to be stored on permanent devices or to be scanned thoroughly more than once. It is hence recognized that both approximation and adaptivity are key ingredients for executing queries and performing mining tasks over rapid data streams.

In this paper, the problem of clustering multiple data streams is addressed. It is assumed that at each time stamp, data points from individual streams arrive simultaneously, and the data points are highly correlative to previous ones in the same stream. Unlike that of prior studies, the objective in this work is to partition these data streams, rather than their data points, into clusters. Note that the data streams are not of a fixed length. Instead, they are still evolving when the clustering results are observed at users' requests. The problem studied in this paper is different from the one discussed in [7], which focuses on clustering the windows of a single streaming time series. On the other hand, clustering of evolving streams is discussed in [11]. However, the objective in [11] is to continuously report clusters satisfying the specified distance threshold. To further enhance these techniques, clustering requests of flexible time ranges are supported in our framework.

In the data stream environment, the patterns generated by the mining techniques are usually distinct at different time because of the evolution of data. Depending on different applications, the frequency for patterns in data streams to change varies. For example, the streams gathered from adjacent sensors may always be in the same cluster. However, for stock prices, some companies are probably within the same cluster during several months but in different clusters afterward. The clusters obtained hence change frequently. Therefore, an important question arises: "*Can we design a scheme for modeling both fast and slow evolving patterns adaptively?*" Furthermore, the clustering request is unknown when the data is collected and processed. After the time range of clustering request reveals, recommendations for short-term or long-term investments are desired to be offered precisely. This leads to another important issue: "*Can we provide a system to support various clustering requirements at the same time?*" Consequently, we devise in this paper a framework of *Clustering on Demand*, abbreviated as *COD framework*, to dynamically cluster multiple data streams. While providing a general framework of clustering on multiple data streams, the proposed COD framework has two advantageous features: (1) one data scan for online statistics collection, and (2) compact multi-resolution approximations, that are designed to address, respectively, the time and the space constraints in a data stream environment. Furthermore, with the multi-resolution approximations of data streams, flexible clustering demands can be supported. Note that since the cluster-

Proceedings of the Fourth IEEE International Conference on Data Mining (ICDM'04)
0-7695-2142-8/04 $ 20.00 IEEE

ing algorithms are only applied to the statistics maintained rather than to the original data streams, the COD proposed is very efficient in practice.

The COD framework proposed consists of two phases, namely the *online maintaining phase* and the *offline clustering phase*. The online maintaining phase provides an efficient algorithm to maintain the summary hierarchies of the data streams with multiple resolutions in the time complexity linear to both the number of streams and the number of data points in each stream. On the other hand, an *adaptive clustering algorithm* is devised for the offline phase to retrieve the approximations of the desired sub-streams from the summary hierarchies as precisely as possible according to the clustering queries specified by the users. In general, we keep finer approximations for more recent data and coarser approximations for more obsolete data. The COD framework performs very efficiently in the data stream environment while producing clustering results of very high quality.

The rest of this paper is organized as follows. Preliminaries and advantages of the COD framework are described in Section 2. The online maintaining phase and the offline clustering phase of the COD framework are presented in Section 3. This paper concludes with Section 4.

2 Clustering on Demand Framework

2.1 Framework Definitions

The COD framework has two phases, i.e., the *online maintaining phase* and the *offline clustering phase*. In the online maintaining phase, the arriving data streams are processed and only very brief summaries are maintained. The offline clustering phase deals with the clustering queries. The COD framework supports clustering queries with a flexible window size and a desired number of windows to observe. For example, a clustering query could be 12 windows with the window size of 30 days to observe the clusters of each month during a year. Based on the limited space property in the data stream environment, the raw data streams are parsed only once and then discarded. Therefore, the clustering algorithm in our framework is applied to the statistics maintained by the online phase rather than to the original streams, as illustrated in Figure 1.

At any time stamp, each stream receives a new value simultaneously, and there are totally n data streams. More specifically, we have the n streams $\{S_1, S_2, ..., S_n\}$ at time stamp m where $S_i = \{f_i^1, f_i^2, ..., f_i^m\}$, for $1 \leq i \leq n$, and f_i^j is the value of stream S_i that arrives at time j. First, we introduce the *offline clustering phase*. Let k denote the number of clusters, and let w be the window size of the clustering query submitted at time stamp t_{now}. The algorithm proposed will generate at most p windows of k-clustering results where

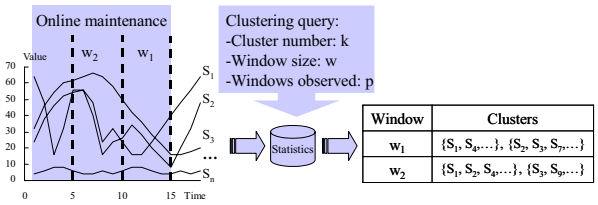

Figure 1. Clustering of multiple data streams by COD at $t_{now} = 15$, for a query of $k = 2$, $w = 5$ and $p = 2$.

$Cl(w_i) = \{C_1(w_i), C_2(w_i), ..., C_k(w_i)\}$, for $1 \leq i \leq p$, which minimizes each clustering cost $Cost(Cl(w_i))$ of the sub-streams in the interval $(t_{now} - w \times i, t_{now} - w \times (i-1)]$. Note that $C_j(w_i)$ is the j^{th} cluster of window w_i with the properties of $\bigcap_{j=1}^{k} C_j(w_i) = \phi$ and $\bigcup_{j=1}^{k} C_j(w_i) = \{S_1(w_i), S_2(w_i), ..., S_n(w_i)\}$, where $S_q(w_i) = \{f_q^{(t_{now}-w\times i)+1}, f_q^{(t_{now}-w\times i)+2}, ..., f_q^{(t_{now}-w\times i)+w}\}$, for $1 \leq q \leq n$.

Example 1: Consider the first three data streams $\{S_1, S_2, S_3\}$ in Figure 1 at time stamp t_{now}=15, where S_1={64,48,16,32,56,56,48,24,32,24,16,16,24,32,40}, S_2={24,38,46,52,54,56,40,16,24,26,34,28,20,14,8}, and S_3={32,46,54,60,62,64,66,64,58,50,42,36,28,22,16}. Assume that the clustering query is $k = 2$, $w = 5$, and $p = 2$. The algorithm will generate at most 2 windows of 2-clustering results, $Cl(w_1) = \{C_1(w_1), C_2(w_1)\}$ and $Cl(w_2) = \{C_1(w_2), C_2(w_2)\}$. Note that the resulting clusters of window w_1 are $C_1(w_1) = \{S_1\}$ and $C_2(w_1) = \{S_2, S_3\}$ since S_2 is more similar to S_3 in the interval $(15 - 5 \times 1, 15 - 5 \times (1-1)] = (10, 15]$. In window w_2, the clusters are $C_1(w_2) = \{S_1, S_2\}$ and $C_2(w_2) = \{S_3\}$ since S_2 is more similar to S_1 in the interval $(15 - 5 \times 2, 15 - 5 \times (2-1)] = (5, 10]$.■

We next describe the *online maintaining phase*. To support various clustering queries in the offline clustering phase, adequate information has to be preserved during the online maintaining phase for discovering fast and slowly evolving patterns. Note that the resolution of statistics maintained could affect the patterns obtained. With a small interval used for summarization, short-term patterns can be observed, but long-term patterns are likely to be neglected. Also, the summaries are possibly affected by noises and oscillations, and the summaries should be updated or reconstructed frequently. On the other hand, if a large interval is used, long-term patterns can be observed while neglecting short-term patterns. Also, the patterns may not catch up with the changes of the data streams. Moreover, the patterns could be very rough because of the generalization/summarization of streams. Therefore, we devise a hierarchical structure to store data summaries at different

resolutions in the online maintaining phase. Whenever a clustering query is submitted, the algorithm of the offline clustering phase will select the approximations from appropriate levels of the summary hierarchies to support the requirements of the clustering query. Details of the online and offline phases will be described in following sections.

2.2 Advantages of the COD Framework

Since the summaries maintained in the summary hierarchies are at multiple resolutions, the COD framework is able to support the *clusters for variable window sizes*. Applying the conventional time series clustering algorithms to the raw streams with the desired window sizes is not practical in the streaming environment because there is not enough space for the storage of continuous streams. On the other hand, the summarization techniques which maintain the streams with a fixed resolution do not perform well for various window sizes. Although the prior work [11] for clustering of evolving streams continuously reports clusters within the given distance threshold, it does not allow the user to specify a desired window size to be observed. In our COD framework, even though the data streams are collected and summarized into the summary hierarchies before the clustering queries are submitted, the clusters with various window sizes can be obtained directly from the existing summary hierarchies without parsing the data streams again to construct the summaries for the desired resolutions. For example, suppose that the summary hierarchies of the stock prices have been kept for the prices in ten years. Then, clusters in one day, one month, or even one year can be observed very efficiently without resorting to the old prices again.

From the definition of clustering query, at most p windows of clustering results can be obtained for a query. It is very efficient to *observe the trends and changes of clusters* at one time. The behaviors of the clusters, such as the moving paths of clusters, the merges and splits of clusters, and the streams jumping between clusters, can be investigated from the results of a clustering query. Consider the example of stock prices again. Assume that the window size is one month and 12 windows are inspected. We might find out that stock A and stock B are in the same cluster for one month and then stock A jumps to another cluster for the following several months. Such trends and changes within one year can be extended from the results of this clustering query.

3 The Framework of COD

3.1 Online Maintaining Phase

The main objective of this online maintaining phase is to provide a one scan algorithm of the incoming multiple data streams for statistics collection. A summary hierarchy is maintained incrementally to provide multi-resolution approximations for a stream. Multiple levels in the hierarchy correspond to various resolutions.

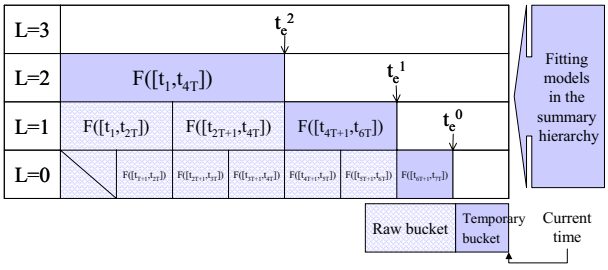

Figure 2. The illustration of summary hierarchy, where $F(h)$ represents the fitting model in the time interval $h = [t_s, t_e]$, and $\alpha = 6$.

A **fitting model** $F(h)$ is defined as an approximation of a raw sub-stream in the interval $h = [t_s, t_e]$ by some summarization techniques. The fitting model on level L is generated by the aggregation of B_h models on level $(L-1)$, and each level keeps the time stamp of the latest data point, which has been summarized to that level, as the end time t_e of the level. The procedure of generating and updating the summary hierarchy of a stream is described as follows.

Procedure of the online maintaining phase

1 For each incoming value, put it in the temporary bucket.

2 If the number of items in the temporary bucket is less than the bucket size B_t, go to Step 1. Else:

2.1 A new fitting model of level 0 is generated according to the values in the temporary bucket.

2.2 Update the end time t_e^0 of level 0.

2.3 Move all the items from the temporary bucket to the raw bucket.

2.4 If the number of items in the raw bucket is more than α, remove the oldest items to keep the number no larger than α.

3 For each level L, if B_h new models are accumulated:

3.1 A new model of level $(L+1)$ will be generated by aggregating the latest B_h models in level L.

3.2 Update the end time t_e^{L+1} of level $(L+1)$.

3.3 If the number of models in level $(L+1)$ is larger than α, remove the oldest model from that level.

As shown in the procedure, to achieve the space limitation in the streaming environment, the number of fitting models maintained at each level is limited to be the maximum number of α. Note the α should be set to a number no smaller than B_h in order to have enough fitting models for the model generation in a higher level. Figure 2 gives an example of the summary hierarchy.

3.2 Offline Clustering Phase

As the clustering query shown in Section 2.1, the users want to inspect clusters with window size w, and at most p windows will be observed. Note that the window size desired could be different from those maintained in the summary hierarchy. In this situation, we have to select the fitting models from appropriate levels of the hierarchy to approximate the desired windows. We have the following theorems for adaptive level selection.

Theorem 1: The highest level to approximate a sub-stream with window size w is $L_{max} = \left\lfloor log_{B_h}(\frac{w}{B_t}) \right\rfloor$.

Theorem 2: The lowest level to approximate a sub-stream with window size w is $L_{min} = \min_{L} \left\{ w \leq (t_{now} - t_e^L) + \alpha_L \times h_L \right\}$, where α_L is the exact number of fitting models in level L, and $h_L = B_t \times (B_h)^L$ is the window size of the fitting models in level L.

From the above theorems, fitting models in the levels between L_{min} and L_{max} are able to approximate the sub-streams in the windows of clustering queries. The fitting models in higher levels span longer intervals and thus provide more generalized fittings to the original streams. In contrast, the fitting models in lower levels possess shorter spans and can thus provide more specific and accurate fittings to the original streams. However, only α models are maintained in each level. Therefore, we devise an *adaptive clustering algorithm* for the offline clustering phase to approximate each window of data streams with the fitting models as accurately as possible. Note that the offline clustering phase is not designed for a specific clustering algorithm. Therefore, users can adopt any traditional clustering algorithm with minor modification if so necessary.

Procedure of adaptive clustering algorithm

1. Calculate L_{min} and L_{max}. For each data stream, do Step 2 and 3.

2. If the end time $t_e^{L_{min}}$ of level L_{min} is not equal to the current time, aggregate the models of lower levels (from $L_{min} - 1$ to 0) and the temporary bucket to generate a temporary model characterizing the interval between $t_e^{L_{min}}$ and the current time. Then, aggregate this temporary model to the latest model in level L_{min}.

3. Encapsulate the fitting models between level L_{min} and L_{max} to generate at most p entries, where each entry represents a window with size w. Set $L = L_{min}$ initially. For the windows from w_1 to w_p, if the range of a desired window is covered by the interval of the fitting models in level L, encapsulate an appropriate number of fitting models into that entry. Else, increase L by one to look for the fitting models with enough coverage. This step stops when p entries have been retrieved or when L exceeds the maximum level L_{max} with p_r entries obtained, where $p_r \leq p$.

4. Run the clustering algorithm to cluster these sub-streams by the retrieved entries for each window.

4 Conclusions

In order to deal with various types of multiple data streams and to support flexible mining requirements, we devised a COD Framework to dynamically cluster multiple data streams. While providing a general framework of clustering on multiple data streams, COD framework had two major advantages, namely one data scan for online statistics collection and compact multi-resolution approximations. The online maintaining phase of COD provided an efficient algorithm to maintain the summaries of the data streams with multiple resolutions. On the other hand, an adaptive clustering algorithm was devised for the offline phase of COD to retrieve the approximations of the desired sub-streams from the summary hierarchies as precisely as possible according to the clustering queries. The COD framework performed very efficiently in the data stream environment while producing clustering results of very high quality.

Acknowledgement

The work was supported in part by the National Science Council of Taiwan, R.O.C., under Contract NSC93-2752-E-002-006-PAE.

References

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

[2] B. Babcock, S. Babu, M. Datar, R. Motwani, and J. Widom. Models and issues in data stream systems. In *Proc. of PODS*, June 2002.

[3] A. Bulut and A. K. Singh. Swat: Hierarchical stream summarization in large networks. In *Proc. of ICDE*, pages 303–314, Mar. 2003.

[4] A. Dobra, M. N. Garofalakis, J. Gehrke, and R. Rastogi. Processing complex aggregate queries over data streams. In *Proc. of ACM SIGMOD*, pages 61–72, June 2002.

[5] S. Guha, N. Mishra, R. Motwani, and L. O'Callaghan. Clustering data streams. In *Proc. of FOCS*, pages 359–366, 2000.

[6] G. Hulten, L. Spencer, and P. Domingos. Mining time-changing data streams. In *Proc. of ACM SIGKDD*, pages 97–106, Aug. 2001.

[7] E. Keogh, J. Lin, and W. Truppel. Clustering of time series subsequences is meaningless: Implications for past and future research. In *Proc. of ICDM*, Nov. 2003.

[8] G. S. Manku and R. Motwani. Approximate frequency counts over streaming data. In *Proc. of VLDB*, pages 346–357, Aug. 2002.

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

[10] W.-G. Teng, M.-S. Chen, and P. S. Yu. A regression-based temporal pattern mining scheme for data streams. In *Proc. of VLDB*, Sep. 2003.

[11] J. Yang. Dynamic clustering of evolving streams with a single pass. In *Proc. of ICDE03*, Mar. 2003.