# Granularity Adaptive Density Estimation and on Demand Clustering of Concept-Drifting Data Streams*

Weiheng Zhu[1], Jian Pei[2], Jian Yin[1], and Yihuang Xie[1]

[1] Zhongshan University, China
gz_zwh@263.net, issjyin@mail.sysu.edu.cn, lion21@163.com
[2] Simon Fraser University, Canada
jpei@cs.sfu.ca

**Abstract.** Clustering data streams has found a few important applications. While many previous studies focus on clustering objects arriving in a data stream, in this paper, we consider the novel problem of *on demand clustering concept drifting data streams*. In order to characterize concept drifting data streams, we propose an effective method to estimate densities of data streams. One unique feature of our new method is that its granularity of estimation is adaptive to the available computation resource, which is critical for processing data streams of unpredictable input rates. Moreover, we can apply any clustering method to on demand cluster data streams using their density estimations. A performance study on synthetic data sets is reported to verify our design, which clearly shows that our method obtains results comparable to CluStream [3] on clustering single stream, and much better results than COD [8] when clustering multiple streams.

## 1 Introduction

Recently, clustering data streams has found a few important applications, such as stock market and financial data analysis, sensor networks, wireless communication, and network traffic management. A data stream can be regarded as a (potentially endless) list of data entries. Typically, data streams are assumed arriving continuously and rapidly.

In this paper, we study the problem of *on demand clustering concept drifting data streams*, which is illustrated in the following example.

*Example 1 (Motivation).* In a coal mine, thousands of sensors are deployed in pits to monitor the temperature, the humidity, and the concentrations of oxygen and gas. Each sensor keeps reporting the observed data, and thus generates a data stream. Typically,

* This work was supported by the National Natural Science Foundation of China (60573097) , Natural Science Foundation of Guangdong Province (05200302, 04300462), Research Foundation of National Science and Technology Plan Project (2004BA721A02), Research Foundation of Science and Technology Plan Project in Guangdong Province (2005B10101032), Research Foundation of Disciplines Leading to Doctorate degree of Chinese Universities(20050558017), the NSERC Grants 312194-05 and 614067, and the NSF Grant IIS-0308001. All opinions, findings, conclusions and recommendations in this paper are those of the authors and do not necessarily reflect the views of the funding agencies.

A Min Tjoa and J. Trujillo (Eds.): DaWaK 2006, LNCS 4081, pp. 322–331, 2006.

the sensors are not synchronized. That is, the rate that a sensor reports data is independent of the others. Generally, such surveillance data streams are concept drifting. That is, the distribution of a stream may evolve over time.

Clustering the surveillance data streams is important to monitor the working conditions in pits. However, due to the nature of sensors and the detection environment, the data collected is often noisy. It is not surprising at all that about $30\%$ of data in a large sensor network is noise. Data records may be lost on their way to the servers. Apparently, clustering the objects (i.e., records of (temperature, humidity, oxygen concentration, gas concentration)) in a data stream does not make good sense. Instead, if we can characterize the distributions of temperature, humidity, oxygen concentration, and gas concentration of sensors, we should cluster the sensors according to their distributions. ∎

To tackle the problem of on demand clustering concept drifting data streams motivated in Example 1, we need to address some challenges.

**Challenge 1:** *How we can characterize the distributions of noisy data streams?*

**Our contribution.** We propose to characterize a data stream using its *kernel density estimation* [15]. Accordingly, data streams should be clustered by their densities. Although kernel density estimation is also used in [2] to estimate the changes in a data stream, to the best of our knowledge, this is the first paper to apply kernel density estimation to cluster multiple data streams.

**Challenge 2:** *When the number of fast data streams is large, how can we develop an efficient and scalable method to maintain the density estimations adaptively*?

**Our contribution.** Many previous methods employ load-shedding, i.e., dropping some data, to handle workload heavier than their capabilities. However, load-shedding may lose some important information. Here, we propose a load-skimming approach, which lowers down the granularity of the kernel density estimation adaptively but still captures every incoming data entry. Our load-skimming approach can provide a solid bound on the quality of density estimation.

**Challenge 3:** Concept drifting may happen in a data stream. *How can we capture the concept drifting effectively?*

**Our contribution.** We develop an effective approach to detect significant changes of the density of a data stream. The general idea is to monitor the top-$k$ densest regions in a data stream, and catch the changes. Comparing to the existing methods of change detection, our method is simple and efficient, and thus can be used to handle a large number of fast data streams.

The rest of the paper is organized as follows. In Section 2, we formulate the problem and review the related work. The granularity adaptive density estimation method is proposed in Section 3. We develop the notion of top-$k$ synopsis and the concept drifting detection method in Section 4. The experimental results are reported in Section 5. Section 6 concludes the paper.

## 2 Problem Definition and Related Work

In this section, we first present the model of on demand clustering of data streams according to their kernel densities. Then, we review related work.

### 2.1 The Model

Without loss of generality, we consider *data space* $\mathcal{D}^n$, where $\mathcal{D} = [\alpha, \beta] \subset \mathcal{R}$ is a range of real numbers. A data stream $S$ is a sequence of vectors $v_j$ $(j = 1, 2, \ldots)$ such that each vector $v_j \in \mathcal{D}^n$. A vector is also called an *entry*.

To characterize a data stream $S$, the distribution of the vectors in the data space can be used. Let $f$ be the density function of data stream $S$. Conceptually, stream $S$ is generated by sampling an infinite data set generated by $f$.

A data stream $S$ is call *concept drifting* if the density function $f$ of $S$ evolves over time. In this paper, we consider a set of concept drifting data streams $S_1, \ldots, S_m$. We assume that the data streams in question are not synchronized. In other words, the $j$-th vector $v_j^1$ in stream $S_1$ and the $j$-th vector $v_j^2$ in $S_2$ may not arrive at the same time or at the same time slot. This assumption reflects the application scenarios where the input rates of data streams may vary from stream to stream, and from time to time.

Essentially, the similarity between two streams $S_1$ and $S_2$ can be measured by the similarity between their density functions $f_1$ and $f_2$. Therefore, we can cluster the streams according to their densities.

**Problem definition.** Given a set of concept drifting data streams $S_1, \ldots, S_m$ in the data space $\mathcal{D}^n$, the problem of *on demand clustering* the data streams is to continuously maintain the density functions $f_1, \ldots, f_m$ for the streams, and cluster the streams on demand according to the similarity among their density functions. ∎

The on demand clustering of data streams consists of two steps: continuous density estimation and on demand clustering. In the continuous density estimation step, data streams are summarized using density functions. In the clustering step, any clustering method can be used, such as k-means employed in our experimental study. In the rest of the paper, we shall focus on the continuous density estimation step.

### 2.2 Related Work

Clustering has been studied extensively in both statistics and computer science literature. Jain et al. [10] provides a nice survey.

Clustering data streams has been studied in depth recently (e.g., [1,3,2,5,9,14]). In real applications, clustering analysis may be conducted under different models. For example, many of the previous studies (e.g., [1,3,5,9,14]) focus on clustering data objects in one stream. That is, an entry in the data stream in question represents a data object. The task is to cluster the objects, probably with some constraints such as considering only objects arriving in a sliding window, or finding clusters in subspaces. As another example, some studies (e.g., [12,17]) maintain clusters of moving objects over time.

Some methods have also been developed to detect changes of clusters. For example, Aggarwal [2] uses velocity density estimation to capture and visualize changes in an evolving data stream. Bursts are often considered as an important type of changes. Burst-detection in data streams has been studied in [11,13,18].

When the input rates of streams exceed the capacity, load shedding techniques can be used. Essentially, a load shedder samples the input data streams, and the stream

processing methods are applied on the samples only instead of the raw streams. Therefore, load shedding can improve the latency of the data analysis result by trading off the answer quality. Various load shedding strategies for data stream processing have been studied, such as [4,7,16].

The critical difference between load shedding and load skimming developed in this paper is that load skimming still processes every observation instead of sampling the incoming streams. Load skimming adapts to changing input rates of streams by adjusting the granularity of the data analysis.

Recently, clustering data streams on demand has been studied in [3,8]. In [3], Aggarwal et al. use micro-clusters to summarize objects in a data stream. Then, clustering on demand is conducted using the micro-clusters instead of the original data. CluStream [3] clusters objects in a single data stream, and does not address the problem of clustering multiple data streams and clustering using densities. In [8], Dai et al. assume that each data stream is an infinite time series and the data streams are synchronized in entry arrival. COD [8] approximates the time series and use the approximation to conduct clustering on demand. In our model, a stream is not a time series and streams are not synchronized in arrival. In Section 5, we shall experimentally compare our approach and CluStream [3] and COD [8].

## 3 Granularity Adaptive Density Estimation

In this section, we discuss how to estimate the kernel density function for a data stream. We assume that concept drifting does not happen. Section 4 will address how to handle concept drifting.

### 3.1 Kernel Density Estimation

To estimate the density function $f$ from a data stream, we adopt the *kernel density estimation* method [15].

Consider estimating the density function $f$ for a data stream $S$ that has no concept drifting. At a point $p$ in the data space, $f(p)$ is the density at the point. Suppose the density function $f$ is in Gaussian distribution. Then, $f(p)$ can be estimated by

$$f(p) = \frac{1}{k} \sum_{j=1}^{k} \left( \frac{1}{\sqrt{2\pi}} \cdot e^{-\frac{dist(p,v)^2}{2h^2}} \right), \tag{1}$$

where $k$ is the number of entries in the data stream arrived so far, $dist(p, v)$ is the Euclidian distance between $p$ and $v$, and $h$ is a parameter to describe the influence of a data point in its neighborhood. The intuition is that an entry can be regarded as being generated by different points in the data space according to the density function, and the density function at a point can thus be estimated by the sum of the contributions of all the entries.

Generally, we can use any kernel functions for density estimation. In this paper, we use the Gaussian kernel due to its simplicity and its universal applicability in applications. Our approach can still be applied using other kernel functions.

### 3.2 Grid-Based Estimation

To estimate a continuous density function for a data stream is difficult. In practice, we can approximate a continuous density function by a discrete representation. Intuitively, by estimating the densities of a large number of probe points evenly distributed in the data space $\mathcal{D}^n$, we can achieve a good estimation of the continuous density function. For any point $q$ in the data space, $f(q)$ can be estimated by $f(p)$ where $p$ is the probe point closest to $q$.

Technically, we can organize the probe points as a grid in the data space $\mathcal{D}^n$. Let $l$ be a *granularity parameter* specified by the user. Recall that $\mathcal{D} = [\alpha, \beta]$ is the range in each dimension. Let $\omega = \frac{\beta-\alpha}{l}$. Then, we deploy $(l+1)^n$ probe points $(\alpha+i_1\omega, \ldots, \alpha+i_n\omega)$ in the data space, where $0 \leq i_1, \ldots, i_n \leq l$. That is, on each dimension we have $(l+1)$ probe point coordinate values evenly distributed. We call $(l+1)^{-n}$ the *granularity* of the probe grid. The smaller the granularity, the better the estimation quality.

When a new entry $v$ in the data stream comes, for each probe point $p$, we shall update the density function $f(p)$ according to Equation 1.

### 3.3 Granularity Adaptive Estimation

Clearly, the complexity of updating the density estimation for each new entry in a stream is $O(l^n)$. To control the cost in practice, we propose two methods: granularity adapting discussed here and localized estimation introduced in Section 3.4.

The input rate of a data stream may change over time. When the input rate of a data stream increases and becomes so fast that the density estimation cost exceeds the capacity of the stream mining system, updating every probe point in the grid may become infeasible or unacceptable.

In order to still process every new observation point in the upcoming stream, we can lower down the granularity of the probe grid. In other words, we can make the granularity adaptive to the input rate of the data stream.

Technically, we can adjust the granularity parameter $l$. By reducing $l$, we can reduce the number of probe point coordinates in each dimension, and thus reduce the granularity. Suppose the new granularity parameter is $l'$. then, the reduction on the granularity is $((l+1)^n - (l'+1)^n)$. The following result help to set the granularity parameter $l$ properly.

**Theorem 1.** *Let update time of the density estimation for a probe point be $t$, the input rate of a stream be $\nu$, and the fraction of runtime allocated to processing the stream be $a$. Then, the granularity is minimized when the granularity parameter $l = \lfloor (\frac{a}{\nu t})^{\frac{1}{n}} \rfloor - 1$.*

**Proof.** The maximum number of probe points that can be processed in a unit time is $\frac{a}{\nu t}$. $l$ must be an integer and satisfy inequality $\frac{a}{\nu t} \geq (l+1)^n$. The theorem follows. ∎

In implementation, adjusting granularity parameters frequently can be very costly. Instead, we use a granularity upgrade rate $\psi > 1$. Let the current granularity parameter be $l$. When the input rate of a data stream exceeds the capacity of the system, the granularity parameter is reduced to $\frac{l}{\psi}$. On the other hand, when the available resource is sufficient to support a finer granularity of parameter at least $l \cdot \psi$, then the granularity is lowered down.

When the granularity parameter is changed, a new probe point may not be a probe point in the previous probe grid. To avoid loss of historical data, a new probe point that is not in the previous probe grid inherits the density function estimation from the closest probe point that is in the previous probe grid.

### 3.4 Localized Estimation

As analyzed before, when the granularity is small, updating the density estimation at each probe point for a new entry in a data stream can be costly. From Equation 1, we can observe that the effect of a new entry on a remote probe point can be very small. The effect decreases exponentially with respect to the distance between the entry and the probe point: the larger the distance, the smaller the effect.

This observation motivates the localized estimation method as follows. We can ignore the effect on the probe points that are far away from the new entry. Technically, we can neglect probe points that are of distance at least $\delta h$ from the observation point. We can prove the following.

**Theorem 2 (Error bound).** *For a probe point $p$, if all entries $q$ in the data stream such that $dist(p,q) > \delta h$ are ignored, the error on the estimation of $f(p)$ is bound by*

$$E = \frac{1}{k} \sum_{q, dist(p,q) > \delta h} e^{-\frac{dist(p,q)^2}{2h^2}} \leq \frac{|\{q | dist(p,q) \geq \delta h\}|}{k} \cdot e^{-\frac{\delta^2}{2}},$$

*where $k$ is the total number of entries arrived so far in the data stream.* ∎

By Theorem 2, we can update only the probe points around a newly arrived entry in a data stream. In practice, the error can be much smaller than the upper bound when the granularity is not too rough.

### 3.5 Summary

We can use a probe grid to estimate a discrete representation of the density function of a data stream. By localized estimation, when a new entry arrives, we only need to update the density estimation of the probe points around the entry, and the accumulated error is bound by Theorem 2. To address the change of input rate of a data stream, we can adjust the granularity parameter adaptively.

## 4 Tracing Concept Drifting and on Demand Clustering

Section 3 presents an adaptive method to estimate densities of data streams. When concept drifting happens, a data stream evolves and its density function changes over time. How can we handle concept drifting data streams effectively and timely?

### 4.1 Tracing Drifting by Decaying

When concept drifting happens in a data stream, the density of the stream in a recent window is different from the density in the long term history, as elaborated in Figure 1.

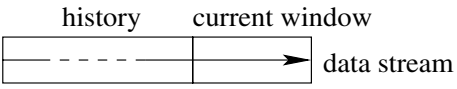


**Fig. 1.** Concept drifting

Therefore, to handle concept drifting, we shall trace the drifting concepts by finding the current density. Here, we propose two methods.

Critically, to trace the current density, we need a mechanism to eliminate stale data gradually. Decaying is a typical and effective strategy, which is also used in some previous studies on data streams, such as [6]. Essentially, each data entry carries a weight which decays over time. More recent data has higher weights than older data.

In the context of this study, we use a decay factor $\rho$, which is a value between 0 and 1. Periodically, the density estimation of each probe point is decayed by factor $\rho$, i.e., $f(p) = \rho f(p)$. In implementation, decaying can be conducted lazily. That is, each probe point $p$ carries a time stamp $\tau_p$. When the probe point $p$ is updated due to a new entry in the stream, $\tau_p$ is compared with the system current time stamp $\tau$. The density estimation $f(p)$ should be decayed to $f(p) \cdot \rho^{\tau - \tau_p}$ before the effect of the new entry is added in.

Alternatively, to capture the current density distribution, we can use a window $W$ of size $\tau$, where $\tau$ is a user-specified parameter. We estimate the density distribution in $W$. When the window is full (i.e., $\tau$ new entries arrive), the density distribution in $W$ is compared with the historical density distribution. If the difference between the two distributions is minor, then no concept drifting happens and the density distribution in $W$ can be added to the historical density distribution, since the density distribution is addable. On the other hand, if the difference is substantial, then the concept drifting happens. We should use the distribution in $W$ to replace the historical distribution.

### 4.2 On Demand Clustering

We can compare two data streams using their densities. Let $f_1$ and $f_2$ be the estimated density functions of data streams $S_1$ and $S_2$, respectively. Then, the similarity between $S_1$ and $S_2$ can be measured as

$$sim(S_1, S_2) = \sqrt{\int_{\mathcal{D}^n} (f_1(p) - f_2(p))^2 dp} \tag{2}$$

With the probe grids, Equation 2 can be approximated by

$$sim(S_1, S_2) = \sqrt{\sum_{\text{probe point } p} (f_1(p) - f_2(p))^2} \tag{3}$$

We can apply any clustering algorithms on demand to find clusters of the data streams.

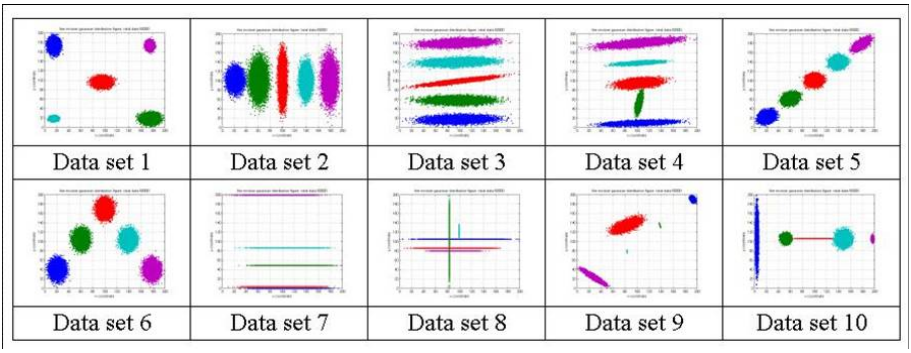


**Fig. 2.** The 10 data sets used in Section 5.1

**Table 1.** The sum of squares of distances to means on the 10 synthetic data sets

| Method | $D_1$ | $D_2$ | $D_3$ | $D_4$ | $D_5$ | $D_6$ | $D_7$ | $D_8$ | $D_9$ | $D_{10}$ |
|---|---|---|---|---|---|---|---|---|---|---|
| CluStream | **32394.9** | 69811.6 | 73561.5 | **49371.7** | 42980.7 | 53675.6 | **21952.9** | **29070.7** | 43029.2 | **34900.2** |
| Our method | 33887.7 | **66779.9** | **68583.9** | 58816.3 | **37240.7** | **47000.1** | 51351.3 | 49511.2 | **32093.4** | 68476.9 |

## 5 Experimental Results

To verify the clustering method proposed in this paper, we conduct an extensive performance study using synthetic data sets. Limited by space, we can only report some results on selected aspects here. More experimental results can be found in the full version of the paper.

### 5.1 Clustering One Data Stream

First, we examine whether clustering using density estimation can be competent with other clustering methods on mining one data stream. We use 10 synthetic data sets as shown in Figure 2. Each data set contains $50,000$ entries of 5 clusters in a 2-dimensional data space.

On each data set, we run CluStream [3] our density estimation algorithms, respectively. CluStream uses $10,000$ entries to initialize $5,000$ micro-clusters. Our density estimation algorithm uses a $200 \times 200$ probe gride (i.e., the granularity parameter is set to 19), and $h = 10$ in localized estimation (Section 3.4). Both methods scan the data set only once. Then, micro-clusters and probe points are clustered by k-means, where the number of clusters in clustering is set to 5. To compare the quality of clusters found, we compute the sum of squares (SSQ) of distances between entries in the clusters and the corresponding means. The results are shown in Table 1.

We observe that each method obtains the better clustering results on 5 data sets, respectively. Interestingly, on data sets $D_7$, $D_8$ and $D_{10}$ where some clusters distributed in a long stripe, CluStream performs substantially better. In such cases, micro-clusters are capable to capture the stripe structures and thus the situations are to the advantage of CluStream. In the other data sets, our method achieves better or comparable results.

The runtime of computing micro-clusters in CluStream and the density estimation time in our method are also comparable. Limited by space, we omit the details.

### 5.2 Clustering Multiple Streams

We generate synthetic data streams in Gaussian distribution. Each stream contains $10,000$ data entries. The experimental results on $100$ streams are shown in Figure 3. In this test, $10$ clusters are found, which contain $2$, $3$, $8$, $2$, $7$, $11$, $14$, $3$, $43$, and $7$ streams, respectively. The distribution of the streams are listed in the order of clusters. As can be seen, streams in each cluster are of similar distribution. To the best of our knowledge, there exists no other methods that can also cluster multiple streams according to their density distributions. Please note that COD [8] treats each stream as a time series, and thus cannot identify clusters properly in this test. In fact, COD outputs a cluster of $91$ streams and $9$ clusters each of $1$ stream in this experiment.

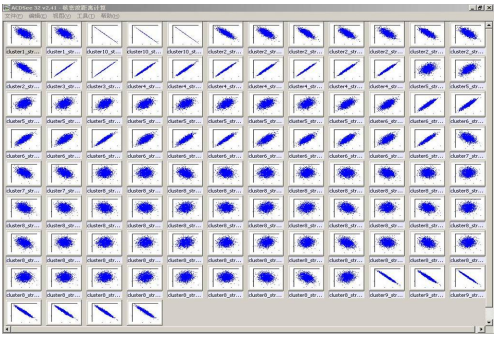

**Fig. 3.** Clustering multiple streams

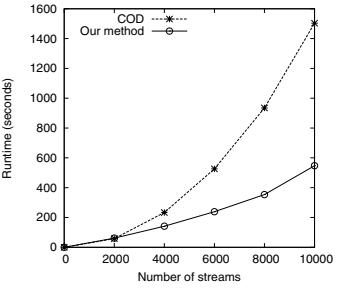


**Fig. 4.** Scalability of clustering multiple streams

Since COD also cluster multiple streams, we compare the runtime of our method and that of COD. The result is shown in Figure 4. As can be seen, our method is scalable with respect to the number of data streams. When there are many streams, our method is clearly more efficient than COD.

## 6 Conclusions

In this paper, we propose a simple yet effective method for on demand clustering multiple concept drifting data streams. The central idea is to characterize concept drifting data streams by estimating densities. Our new method is adaptive to the available computation resource, which is critical for processing data streams of unpredictable input rates. Moreover, we can apply any clustering method to on demand cluster data streams using their density estimations. We report a performance study on synthetic data sets to verify our design. The experimental results clearly show that our method can obtain clusters of good quality and is scalable in mining a large number of data streams.

## References


1. C. Aggarwal, et al. A framework for projected clustering of high dimensional data streams. In *Proceedings of the 30th International Conference on Very Large Data Bases (VLDB'04)*, Toronto, ON, Canada, August 2004.
2. C. C. Aggarwal. A framework for diagnosing changes in evolving data streams. In *Proceedings of the 2003 ACM SIGMOD international conference on Management of data*, pages 575–586. ACM Press, 2003.
3. C. C. Aggarwal, et al. A framework for clustering evolving data streams. In *Proc.the 19th Int. Conf. on Very Large Data Bases (VLDB'03)*, Berlin, Germany, September 2003.
4. B. Babcock, et al. Load shedding techniques for data stream systems. In *Proceedings of the 2003 Workshop on Management and Processing of Data Streams (MPDS 2003)*, San Diego, California, June 2003.
5. B. Babcock, et al. Maintaining variance and k-medians over data stream windows. In *Proceedings of the twenty-second ACM SIGMOD-SIGACT-SIGART symposium on Principles of database systems (PODS'03)*, pages 234–243, New York, NY, USA, 2003. ACM Press.
6. J. H. Chang and W. S. Lee. Finding recent frequent itemsets adaptively over online data streams. In *KDD '03: Proceedings of the ninth ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 487–492. ACM Press, 2003.
7. Y. Chi, et al. Loadstar: A load shedding scheme for classifying data streams. In *Proc. 2005 SIAM Int. Conf. Data Mining*, New Port Beach, CA, April 2005.
8. B-R Dai, et al. Clustering on demand for multiple data streams. In *Proceedings of the Fourth IEEE International Conference on Data Mining (ICDM'04)*, pages 367–370. IEEE, November 2004.
9. S. Guha, et al. Clustering data streams. In *Proc. IEEE Symposium on Foundations of Computer Science (FOCS'00)*, pages 359–366, Redondo Beach, CA, 2000.
10. A. K. Jain, et al. Data clustering: A survey. *ACM Comput. Surv.*, 31:264–323, 1999.
11. J. Kleinberg. Bursty and hierarchical structure in streams. In *KDD '02: Proceedings of the eighth ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 91–101, New York, NY, USA, 2002. ACM Press.
12. Y. Li, et al. Clustering moving objects. In *KDD '04: Proceedings of the tenth ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 617–622, New York, NY, USA, 2004. ACM Press.
13. J. Ma and S. Perkins. Online novelty detection on temporal sequences. In *KDD '03: Proceedings of the ninth ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 613–618, New York, NY, USA, 2003. ACM Press.
14. L. O'Callaghan, et al. High-performance clustering of streams and large data sets. In *Proc. 2002 Int. Conf. Data Engineering (ICDE'02)*, San Fransisco, CA, April 2002.
15. B. W. Silverman. *Density Estimation for Statistics and Data Analysis*. Chapman and Hall, 1986.
16. N. Tatbul, et al. Load shedding in a data stream manager. In *VLDB*, pages 309–320, 2003.
17. Q. Zhang and X. Lin. Clustering moving objects for spatio-temporal selectivity estimation. In *Proceedings of the fifteenth conference on Australasian database (CRPIT'04)*, pages 123–130, Darlinghurst, Australia, Australia, 2004. Australian Computer Society, Inc.
18. Y. Zhu and D. Shasha. Efficient elastic burst detection in data streams. In *Proceedings of the ninth ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 336–345. ACM Press, 2003.