Stream Mining via Density Estimators: A concrete Application

Christoph Heinz, Bernhard Seeger

Department of Mathematics and Computer Science
University of Marburg
Germany
{heinzch, seeger}@mathematik.uni-marburg.de

Abstract

Many real-world applications share the property that the data they process arrives in streams. The transient and volatile nature of these streams renders the application of common processing and analysis techniques difficult. In particular, the mining of streams has proved to be a difficult task due to the rigid processing requirements that must be met within the data stream scenario. We propose to exploit kernel density estimation for stream mining. Kernel density estimation as a technique from the area of mathematical statistics found its way into many mining related topics and applications. However, its heavy computational cost makes the direct application to streams impossible. On account of this, we developed in a previous work a sophisticated adaptation that continuously computes kernel density estimators over streams. By means of these estimators, we can tackle a variety of mining tasks over streaming data. We illustrate some of them against the background of a concrete application in a medical environment. More precisely, we will demonstrate a prototypical implementation of a medical monitor that visualizes an online analysis of the vital signs of patients. Besides demonstrating the monitor, we will also present the concepts underlying its implementation.

1 Introduction

In recent years, a plethora of real-world applications based on continuously arriving data has emerged, e.g. facility monitoring, traffic management. The sheer volume of the data combined with the limited computational resources makes an adequate processing and analysis of the streams difficult. Even though the storage facilities are continuously increasing, which offers to store large parts of a stream, the handling of data streams still poses a lot of open questions. Typically, the data often accumulates faster than it can be processed or analyzed. For that reason, research academia has been developing appropriate techniques [1, 3] to circumvent these limitations, typically at the expense of an approximate instead of an exact solution.

In our work, we concentrate on a basic building block of many data mining and analysis techniques, namely density estimation. The essence of density estimation is to reveal the unknown density of a distribution by exploiting a representative set of observations. By means of a suitable density estimate, we can explore the main features and characteristics of the data. In mathematical statistics, density estimation is among the core topics. A theoretically and also practically approved approach for density estimation uses so-called kernels as main ingredients [10]. Kernel density estimation is a convenient approach as it combines simplicity with guaranteed convergence for arbitrary distributions. Therefore, it has been successfully applied in diverse application scenarios. However, computing a kernel density estimator comes along with a heavy computational burden. In fact, kernel density estimators can not directly be applied in the data stream scenario since their computational complexity collides with the processing requirements of streams. In [7], we tackled this problem and developed a suitable adaptation that meets the rigid processing requirements of streams. Generally, our online kernel density estimators can prepare the ground for diverse mining techniques over data streams.

The objective of our demonstration is to convey an idea of stream mining via density estimators by means of a concrete application. Specifically, we examine the analysis of vital signs of sleeping patients in a sleep laboratory and demonstrate a tailor-suit monitor that continuously visualizes the analytical results. Besides the demonstration of the monitor and its features, we will also address its implementation within PIPES [9], our infrastructure for processing and ex-

International Conference on Management of Data
COMAD 2006, Delhi, India, December 14–16, 2006
©*Computer Society of India, 2006*

ploring streams.

Before we go into the details of the demonstration, let us first briefly introduce kernel density estimators over data streams and discuss how to use them for stream mining.

2 Kernel Density Estimators over Data Streams

Point of origin is a given set of observations of an unknown random variable X. To understand X and its characteristics, the knowledge of its probability density function f is indispensable. With f, we have a comprehensive description of the distribution of X, i.e., we know the probabilities of all possible outcomes of X. To determine the probability of X being in an interval $[a, b]$, we simply have to integrate f with respect to $[a, b]$. However, in real-world scenarios neither X nor its associated density are known. Typically, the only information we have is a sample of representative values.

Kernel density estimation provides a convenient solution to this problem in form of a well-defined estimate of f. For a sample of independent and identically distributed observations $X_1, ..., X_n$, a kernel density estimator (KDE) is defined as

$$\hat{f}(x) := \frac{1}{n} \sum_{i=1}^{n} \frac{1}{h} K\left(\frac{x - X_i}{h}\right) \tag{1}$$

with bandwidth h and kernel function K. Note that a KDE essentially spreads the weight of each observation; h controls the width of this spread. It is worth mentioning that an appropriate setting of the bandwidth is crucial for the quality of a KDE while the setting of the kernel function is minor. For a detailed discussion of the principles of kernel density estimation, we refer to [10, 11].

In our context, we assume the stream to be a continuously increasing sequence of observations $X_1, X_2, ...$ of an unknown random variable X. Since the stream processing requirements dictate single-pass algorithms and restrict the available memory to a constant amount, KDEs can not directly be applied to data streams. The approximate solution we present in [7, 5] continuously provides KDEs over data streams and inherently ensures that all stream processing requirements are met. The basic idea is to represent with one kernel not only one, but multiple observations. The overall number of these kernels is confined to a constant due to the limited amount of available memory. A necessary prerequisite to keep this number constant is the merge of kernels. Therefore, we developed a merge scheme that guarantees a minimum loss of accuracy. Besides the merge scheme, we also addressed suitable strategies for bandwidth and kernel function setting as well as for the underlying implementation.

3 Stream Mining via Density Estimators

By means of our online KDEs, a plethora of mining and analysis tasks for data streams can be tackled. The corresponding analysis methods use online KDEs as essential building blocks. As online KDEs are continuously computed, the methods can inherently keep pace with the stream. In the following, we briefly list a few of those methods. For the sake of simplicity, we concentrate on more basic ones, which already deliver valuable insights into the stream characteristics.

In order to get a first impression of the stream, we can plot the current online KDE, which gives a compact visualization of the already processed elements. By visually exploring the KDE, we can get a good grasp of the stream's basic characteristics, e.g., dense or less dense regions, local extrema. Moreover, a continuous observation of the plots allows us to capture current stream trends.

To get deeper insights into the stream, we can examine statistically important characteristics like mean, variance, value range, or quantiles. Those summary statistics deliver reasonable measures of the variability of the stream as well as of its general tendencies. We can efficiently determine them by evaluating the closed formulas online KDEs provide for their computation.

Besides plotting the density or computing summary statistics, we can also exploit online KDEs to draw more complex conclusions from the stream. Let us illustrate this for two practically relevant tasks: detecting outliers of a stream and estimating the selectivity of range queries.

In [6], we presented an initial approach to detect outliers while processing a data stream. The basic idea is to define an outlier in probabilistic terms. Intuitively, an outlier occurs seldom. Thus, if we take the current online KDE into account, the region the outlier fell into must have a low probability. More formally, we label an incoming element as an outlier if the probability for its local neighborhood falls below a preset threshold. Apparently, this definition has two main parameters: the width of the neighborhood and the threshold. These parameters basically control the sensitivity of the outlier detection. Hence, they must be appropriately chosen.

Another practically relevant task is to estimate the selectivity of range queries. Given a range query $[a, b]$, we want to determine the percentage of already processed stream elements that fell into $[a, b]$. An exact computation of this selectivity is unfeasible due to its storage requirements. Therefore, we propose to approximate the selectivity with the probability for a stream element to fall into $[a, b]$. This probability in turn can be computed with online KDEs; we only have to integrate them with respect to the interval $[a, b]$.

4 Implementation in PIPES

To implement stream mining tasks based on online KDEs, we make use of PIPES, our public infrastructure for processing and exploring streams. We already demonstrated the rich functionality of PIPES as well as its implementation concepts in [9, 2, 4]. Therefore, we only give a brief summary of the implementation of online KDEs and stream mining tasks in PIPES.

On the one side, PIPES offers the essential functionality to express, implement, and run continuous queries over data streams. On the other side, PIPES offers a set of sophisticated techniques, including online KDEs, to estimate the distribution of a data stream and exploit it for a continuous stream mining.

In order to implement continuous queries or analysis techniques over data streams, we have to construct directed acyclic operator graphs. For the case of continuous queries, the nodes correspond to the physical operators implementing this query. For the case of analysis techniques, a node corresponds to a specific operator that implements the analysis functionality. The advantages of implementing an analysis technique as an operator are twofold: First, we can unify diverse analysis tasks within a single operator graph. Second, one node, i.e. one analysis technique, can serve multiple analysis techniques simultaneously. For example, we want to plot the density of a data stream and compute its summary statistics. Instead of implementing these tasks separately, we build an operator graph with the following nodes: The first node corresponds to the data source. The second node is connected to the first one and computes online KDEs. The third and fourth node are connected to the second node; one computes the summary statistics and the other one plots the current KDE.

5 A concrete Application

In the demonstration, we will investigate the exploitation of online KDEs against the background of a real-world scenario. As data stream within this scenario serves a time series from the Times Series Data Mining Archive of UC Riverside [8]. This archive comprises a variety of heterogeneous time series originating from diverse applications. The time series we chose describes the monitoring of vital signs of sleeping patients in a sleep laboratory. In the demonstration, we will concentrate on the monitoring of the heart rate.

The typical method to gain insight into the sleeping disorders the patients exhibit is to analyze the recorded measurements manually in a post-processing phase. On the contrary, we propose to analyze the streams online to get insights while the patient is sleeping. As discussed above, we can use online KDEs for this purpose. Specifically, we can use them to describe the long-term sleeping behavior of the patient since they rely on all processed elements. To describe the short-term sleeping behavior, we additionally maintain a KDE over a sliding time window.

In the demonstration, we will present how to exploit both techniques for analyzing the heart rate of a sleeping patient.

6 Demonstration Content

The objective of our demonstration is to convey an impression of how to use KDEs for stream mining. With respect to the sleep laboratory scenario, we will demonstrate a prototypical heart rate monitor that visualizes the online analysis of the heart rate of a sleeping patient. Besides demonstrating the monitor, we will also present the implementation of its functionality in PIPES.

6.1 Heart Rate Monitor

Let us briefly describe the components of the heart rate monitor with the screenshot given in Figure 1. To refer to the components, we marked them with numbers.

The component marked with 1 displays the incoming heart rate measurements. Based on these measurements, we continuously compute two KDEs: one to analyze the short-term sleeping behavior and the other one to analyze the long-term sleeping behavior.

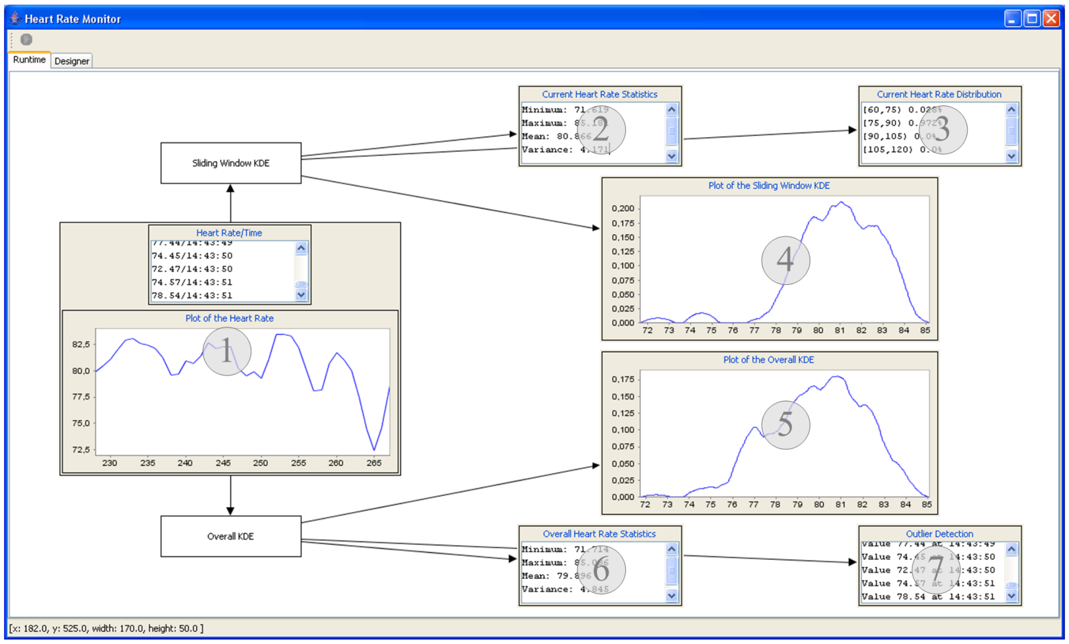
The KDEs for the short-term behavior base on a sliding time window over the last 10 minutes. By means of these KDEs, we compute summary statistics (3) for the current window. Additionally, we plot the current KDE (4). In another component (5), we observe how deep the patient sleeps (5): We partition the value range equidistantly and associate each partition with a range query. By estimating the selectivity of these range queries, we can determine whether the heart rate was high or low in the last 10 minutes.

To analyze the long-term behavior, we use online KDEs. Again, we plot the current KDE (5) and compute the according summary statistics (6). In addition to, we display the measurements labeled as outliers (7) with respect to the current KDE.

Overall, we see that we can unify a set of analysis tasks within a single monitor. By means of this monitor, a medical examiner can get a rough picture of a patient's sleep as well as more detailed insights.

6.2 Implementation Details

Besides demonstrating the monitor and its features, we will also go into the details of its implementation. As Figure 1 already suggests, we implemented the functionality underlying the monitor as an operator graph in PIPES. In the demonstration, we will discuss the construction of this graph as well as the functionality implemented within the different nodes. Besides this concrete example, we will also show how the existing graph can be extended to integrate further analysis functionality.



```
Minimum: 71.619
Maximum: 85.181
Mean: 80.866
Variance: 4.171
```

```
[60,75) 0.028%
[75,90) 0.972%
[90,105) 0.0%
[105,120) 0.0%
```

```
77.44/14:43:49
74.45/14:43:50
72.47/14:43:50
74.57/14:43:51
78.54/14:43:51
```

```
Minimum: 71.714
Maximum: 85.086
Mean: 79.896
Variance: 4.845
```

```
Value 77.44 at 14:43:49
Value 74.45 at 14:43:50
Value 72.47 at 14:43:50
Value 74.57 at 14:43:51
Value 78.54 at 14:43:51
```

```
[x: 182.0, y: 525.0, width: 170.0, height: 50.0 ]
```

Figure 1: Heart Rate Monitor

7 Our Contributions

Finally, let us recapitulate the main contributions of this demonstration:

- For a concrete medical application, we discussed how the estimated density of a data stream can be exploited to tackle diverse mining and analysis tasks.
- We developed a prototypical medical monitor that continuously visualizes the results of the analysis in an intuitive way.
- We discussed how the complex functionality underlying the monitor can be efficiently implemented as an operator graph in our infrastructure PIPES.

Acknowledgements

We thank all members of the PIPES team, in particular Maxim Schwarzkopf, for their valuable contributions. This work has been supported by the German Research Society (DFG) under grant no. SE 553/4-3.

References

[1] B. Babcock, S. Babu, M. Datar, R. Motwani, and J. Widom. Models and Issues in Data Stream Systems. In *PODS*, 2002.

[2] M. Cammert, C. Heinz, J. Krämer, T. Riemenschneider, M. Schwarzkopf, B. Seeger, and A. Zeiss. Stream Processing in Production-to-Business Software. In *Proc. of ICDE*, 2006.

[3] M. Gaber, A. Zaslavsky, and S. Krishnaswamy. Mining data streams: a review. *SIGMOD Record*, 34(2), 2005.

[4] C. Heinz and B. Seeger. Exploring Data Streams with Nonparametric Estimators. In *Proc. of SSDBM*, 2006.

[5] C. Heinz and B. Seeger. Resoure-Aware Kernel Density Estimators over Streaming Data. In *Proc. of CIKM (to appear)*, 2006.

[6] C. Heinz and B. Seeger. Statistical Modeling of Sensor Data and its Application to Outlier Detection. Technical report, University of Stuttgart, Germany, 2006.

[7] C. Heinz and B. Seeger. Towards Kernel Density Estimation over Streaming Data. In *Proc. of COMAD (to appear)*, 2006.

[8] E. Keogh and T. Folias. The UCR Time Series Data Mining Archive. `www.cs.ucr.edu/~eamonn/TSDMA`, 2002.

[9] J. Krämer and B. Seeger. PIPES - A Public Infrastructure for Processing and Exploring Data Streams. In *Proc. of ACM SIGMOD*, 2004.

[10] D. W. Scott. *Multivariate Density Estimation : Theory, Practice, and Visualization*. John Wiley & Sons, 1992.

[11] B. Silverman. *Density Estimation for Statistics and Data Analysis*. Chapman and Hall, 1986.