Resource-Aware Kernel Density Estimators over Streaming Data

Christoph Heinz
Department of Mathematics and Computer Science
University of Marburg, Germany
heinzch@mathematik.uni-marburg.de

Bernhard Seeger
Department of Mathematics and Computer Science
University of Marburg, Germany
seeger@mathematik.uni-marburg.de

ABSTRACT

A fundamental building block of many data mining and analysis approaches is density estimation as it provides a comprehensive statistical model of a data distribution. For that reason, its application to transient data streams is highly desirable. A convenient, nonparametric method for density estimation utilizes kernels. However, its computational complexity collides with the rigid processing requirements of data streams. In this work, we present a new approach to this problem that combines linear processing cost with a constant amount of allocated memory. Our approach also supports a dynamic memory adaptation to changing system resources.

Categories and Subject Descriptors

G.3 [**Probability and Statistics**]: Nonparametric statistics

General Terms

Algorithms, Performance

Keywords

Data Streams, Kernel Density Estimation

1. INTRODUCTION

The mining and analysis of transient data streams has come to the fore in recent years [3]. To be applicable to data streams, a mining technique has to meet stringent processing requirements [2].

Taking those requirements into account, we address in this work the adaptation of a fundamental building block of many analysis techniques, namely density estimation, to the data stream scenario. Density estimation reveals the unknown probability density function of a distribution, given solely a representative sample of observations. With a well-defined density estimate at hand, a variety of mining and analysis issues can be addressed [5], [4].

In most real-world applications over streams, we have no a priori knowledge about the stream. Hence, the class of *nonparametric* density estimation approaches is very appealing as they make no assumptions on the unknown density function; "*the data speak strictly for themselves*" [5]. A theoretically well-founded and also practically approved approach for nonparametric density estimation utilizes kernels. Kernel-based density estimators can approximate *any* distribution arbitrarily good (in probabilistic terms), provided that the associated sample is sufficiently large [5]. Hence, an adaptation of kernel density estimation to data streams seems to be a highly promising approach. However, the high computational cost of kernel density estimators is a severe obstacle; their memory allocation is linear in the sample size, accompanied by linear evaluation cost. Since these facts violate the given processing requirements, we can not directly build kernel density estimators over data streams.

In this paper, we present resource-aware kernel density estimators over streaming data that fully comply with these processing requirements. We build on the general idea of M-Kernels [1], a previous approach for kernel density estimation over data streams. Specifically, we solve crucial deficiencies of M-Kernels concerning their parameter settings and essential processing steps. For the core operation of the algorithm, the merge of two adjacent kernels, we introduce a new optimal method based on a cost measure which is inexpensive to compute. An experimental study confirmed that our estimators perform well for synthetic and real-world data streams; they outperformed M-Kernels with respect to accuracy and maintenance cost.

CIKM'06, November 5–11, 2006, Arlington, Virginia, USA.
ACM 1-59593-433-2/06/0011.

2. PRELIMINARIES

2.1 Data Stream Model

A one-dimensional data stream consists of an unbounded sequence $X_1, X_2, ...$ of numbers with $X_i \in \mathbb{R}$ for $i \in \mathbb{N}$. Except where otherwise stated, we assume that the stream represents at each time instant a sample with independent and identically distributed (iid) observations of an unknown continuous random variable X. The premise of independence of two arbitrary stream elements is reasonable for most applications as data sources typically send their elements autonomously, e.g. traffic sensors. The premise of an identical distribution is weakened in Section 3.

2.2 Kernel Density Estimation

One of the core concepts in mathematical statistics is the **probability density function** (pdf). Essentially, each continuous random variable X has a unique pdf f. As f provides a comprehensive summary of X, its knowledge is crucial to the analysis [4].

In real-world scenarios, however, neither X nor its pdf are known. Typically, we only have observations of X in form of a sample $X_1, ..., X_n$. Density estimation provides suitable methods to estimate a pdf using a sample as major ingredient. We particularly focus on kernel density estimation [5], a nonparametric approach that *solely* bases on the sample. A **kernel density estimator** (KDE) with **kernel function** K and **bandwidth** $h^{(n)}$ is defined as

$$\hat{f}^{(n)}(x) = \frac{1}{n}\sum_{i=1}^{n} \frac{1}{h^{(n)}} K\left(\frac{x - X_i}{h^{(n)}}\right), x \in \mathbb{R} \qquad (1)$$

for iid observations $X_1, ..., X_n$ drawn from a continuous random

variable X, whose pdf f is unknown. Generally, the setting of the bandwidth is crucial to the quality of a KDE. To guarantee probabilistic convergence, the bandwidth has to decrease with the sample size [5]. Contrary to the bandwidth, the setting of the kernel function is minor.

Since we assume a data stream to be an iid sample of an unknown random variable X, the adaptation of kernel density estimation seems straightforward. However, the computational cost of KDEs collides with the processing requirements of streams: they request memory linear in the sample size, i.e., in the size of the stream. Furthermore, each processed stream element must be accessible anytime because common bandwidth strategies require access to the whole sample.

3. OUR APPROACH

In the following, we present our approach to KDEs over streaming data, starting from (1) as point of origin. Let n be the current number of processed stream elements.

3.1 Parameter Settings

As underlying kernel function, we use the Epanechnikow kernel [5]. Not only is this kernel asymptotically optimal among all kernels, it is also simple and has a bounded support.

Concerning the bandwidth, the vital parameter of a KDE, we use the **normal scale rule** [5]. It defines the bandwidth as $h^{(n)} = 1.06 \cdot \sigma^{(n)} \cdot n^{-\frac{1}{5}}$ for a sample with n elements and standard deviation $\sigma^{(n)}$. Since $\sigma^{(n)}$ can be estimated in a single pass, we can apply this rule to streaming data.

3.2 Kernel Entries and their Processing

Kernel entries are the main building blocks of our KDEs. A **kernel entry** $\langle X_i^{(n)}, c_i^{(n)}, L2costs_i^{(n)} \rangle$ consists of a mean $X_i^{(n)}$, a weight $c_i^{(n)}$, and $L2costs_i^{(n)}$, the costs of a merge with its "neighbor". As we are only allowed to allocate a constant amount of memory [2], we confine the maximum number of kernel entries to a constant m. We utilize the entirety of kernel entries to build a KDE:

$$\hat{f}^{(n)}(x) = \frac{1}{n} \sum_{i=1}^{m} \frac{c_i^{(n)}}{h^{(n)}} K\left(\frac{x - X_i^{(n)}}{h^{(n)}}\right). \tag{2}$$

Note that each kernel entry has weight $1/n$ in (2). By using exponentially decreasing weights for older entries instead, we can emphasize recent data. Thus, we can also cope with evolving streams.

Let us now discuss the processing of kernel entries during runtime. For a new stream element X_{n+1}, we build a new kernel entry $\langle X_{n+1}, 1, L2costs^{(n+1)} \rangle$, provided X_{n+1} is not equal to the mean of an existing kernel entry. If that is the case, we only increment the weight of this entry.

If the total number of kernel entries exceeds m, we determine and merge the adjacent kernel entries $\langle X_i^{(n)}, c_i^{(n)}, L2costs_i^{(n)} \rangle$ and $\langle X_j^{(n)}, c_j^{(n)}, L2costs_j^{(n)} \rangle$ closest to each other. The resulting **merge kernel** is defined as $\langle X^*, c_i^{(n)} + c_j^{(n)}, L2costs^{(n)} \rangle$. Its mean X^* is the minimum of

$$L2costs(X) = \int_{-\infty}^{\infty} \Big(\frac{c_i^{(n)}}{h^{(n)}} K\Big(\frac{x - X_i^{(n)}}{h^{(n)}}\Big) + \frac{c_j^{(n)}}{h^{(n)}} K\Big(\frac{x - X_j^{(n)}}{h^{(n)}}\Big) - \frac{c_i^{(n)} + c_j^{(n)}}{h^{(n)}} K\Big(\frac{x - X}{h^{(n)}}\Big) \Big)^2 dx. \tag{3}$$

Due to the simple form of the Epanechnikow kernel, (3) can be converted into a closed formula. We determine for each pair of adjacent kernel entries their merge kernel and set their merge costs as $L2costs(X^*)$. If a merge is required, we choose the pair with *overall* minimum merge costs.

Generally, our KDEs can flexibly adapt to a changing m. If m is increased, we build a separate kernel entry for each new element. If m is decreased, we merge kernel entries sufficiently often.

3.3 Comparison to M-Kernels

The processing of M-Kernels [1] is similar to that sketched above. However, some of their settings severely limit their applicability. First, their use of the Gaussian kernel as underlying kernel function exacerbates an efficient evaluation due to its unbounded support. Second, the proposed bandwidth strategy does not ensure bandwidths decreasing with the sample size; but this is a fundamental prerequisite for the probabilistic convergence of KDEs. Third, the mean of the merge kernel is numerically approximated which introduces additional computational effort and less exact values.

4. EXPERIMENTAL EVALUATION

We scrutinized our approach in an experimental study with synthetic as well as real-world data. Concerning the estimation quality, our KDEs have proved to be very robust. For all examined streams, their estimation error with respect to the best "offline" KDE rapidly decreased for an increasing number of processed elements. In comparison to, M-Kernels were clearly inferior; their estimation error mostly remained constant in the average. M-Kernels were also inferior in terms of processing time. In another experiment, we examined the resource-awareness of our KDEs by abruptly varying their maximum number of kernel entries during runtime. The experimental results indicate that our KDEs reacted very flexible and recovered fast, even after significant reductions of m.

5. CONCLUSIONS

In this work, we tackled kernel density estimation over streaming data. Our approach utilizes simple building blocks, namely kernel entries, to build KDEs anytime while processing the stream. In comparison to M-Kernels, a previous kernel method for data streams, our KDEs were clearly superior as our experimental study revealed. In future work, we aim at generalizing our approach to multidimensional data streams. We will also couple our KDEs with change point detection methods from the area of mathematical statistics in order to locate and react to concept drifts in the stream.

Acknowledgments

This work has been supported by the German Research Society (DFG) under grant no. SE 553/4-3.

6. REFERENCES

[1] Z. Cai, W. Qian, L. Wei, and A. Zhou. M-Kernel Merging: Towards Density Estimation over Data Streams. In *Proc. of DASFAA*, 2003.

[2] P. Domingos and G. Hulten. A general framework for mining massive data streams. *Journal of Computational and Graphical Statistics*, 2003.

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