Online Distribution Estimation for Streaming Data: Framework and Applications

Themis Palpanas[1], Vana Kalogeraki[2], and Dimitrios Gunopulos[2]

[1] University of Trento
themis@dit.unitn.it
[2] University of California, Riverside
{vana, dg}@cs.ucr.edu

Abstract. In the last few years, we have been witnessing an evergrowing need for continuous observation and monitoring applications. This need is driven by recent technological advances that have made streaming applications possible, and by the fact that analysts in various domains have realized the value that such applications can provide.
In this paper, we propose a general framework for computing efficiently an approximation of multi-dimensional distributions of streaming data. This framework enables the development of a wide variety of complex streaming applications. In addition, we demonstrate how our framework can operate in a distributed fashion, thus, making better use of the available resources.
We motivate our techniques using two concrete problems, both in the challening context of resource-constrained sensor networks. The first problem is outlier detection, while the second is detection and tracking of homogeneous regions. Experiments with synthetic and real data show that our method is efficient and accurate, and compares favorably to other proposed techniques for both the problems that we studied.

1 Introduction

Advances in processor technologies and wireless communications have enabled the deployment of small, low cost and power efficient sensor nodes. Such sensor deployments enable the observation of physical phenomena at a time and space granularity that was never before possible. The same is also true for monitoring the operation of machinery, and the structural integrity of buildings and vehicles.

In all these cases, the applications deal with several data streams and require the ability to efficiently process this type of data in real-time. Applications in several other domains have the same requirements as well. For example, in an e-business scenario, we are interested in continuously monitoring the execution of the various processes and the corresponding services, and in network management, we have to continuously analyze traffic statistics coming from multiple sources.

The way that streaming applications are able to efficiently process continuous data arriving at high rates is by computing succinct summaries of the data, and operating on these summaries [8, 7]. In this paper, we describe a framework for efficient, online approximation of multi-dimensional streaming data distributions, based on *kernel density*

estimators [18]. The proposed framework does not require a priori knowledge about the input distribution, and is adaptive, in the sense that it automatically recognizes changes in the streaming data distributions, and updates the approximations accordingly. Moreover, the framework can operate in a distributed fashion, thus, making use of all the available resources, and reusing any processing that has already taken place.

The framework we propose is general, and enables the development of a wide variety of complex streaming applications. In this study, we demonstrate the usefulness and versatility of the framework using two concrete applications from the area of sensor networks. Sensor networks represent a challenging domain, because they combine the requirements of streaming algorithms (i.e., online processing, and efficient use of main memory) with the restrictions of sensor network deployments (i.e., limited available resources and processing power, and distributed operation).

The first application that we examine is distributed deviation detection in a sensor network. The goal is to identify, among all the sensor readings in a sliding window, those values that have very few near neighbors. Note that this is a challenging problem, even for static datasets [14, 17]. This problem is especially important in the sensor network setting because it can be used to identify faulty sensors, and to filter spurious reports from different sensors.

The second application is identification and tracking of *homogeneous regions* [1, 12], which are defined as spatial divisions of the field under observation that exhibit similar measured values over time. For example, an oil spill detected in the ocean is a homogeneous region (Figure 1). The sensors deployed around the origin of the spill can organize themselves into a network and communicate the measurements, to detect regions of varying oil concentrations. In this work, we address the problems of detecting and tracking such homogeneous regions in *real-time* when the definition of the phenomenon is *not* known in advance.

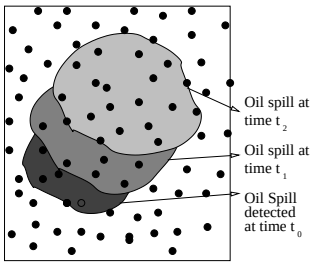

Fig. 1. Spread of an oil spill detected in the ocean over time.

Below we summarize our main contributions.

- We describe a non-parametric, online technique for approximating unknown multi-dimensional streaming data distributions. We then extend this technique in order to efficiently model distributions that evolve over time, and to operate in a distributed fashion.
- We demonstrate the versatility of our approach by describing how it facilitates the implementation of two different applications, namely outlier detection, and detection and tracking of homogeneous regions.

2 Data Distribution Approximation Framework

In this section, we describe a general framework for estimating the underlying distribution of a data stream. We discuss the case where we are interested in the data that falls within a sliding time window W (this is a more general case than unrestricted windows; we omit the presentation of the latter for brevity).

Estimating the Probability Density Function: There are several model estimation techniques that have been proposed in the literature, such as histograms [10], wavelets [8], kernel density estimators [18], and others. In our framework, we choose to estimate the unknown distribution using the kernel density estimators, because of the following desirable properties: (i) they are efficient to compute and maintain in a streaming environment; (ii) they can very accurately approximate an unknown data distribution, with no a priori knowledge and (effectively) no parameters; (iii) they can easily be combined and (iv) they scale well in multiple dimensions. In general, it is computationally more expensive to apply the above operations in histograms or wavelets, and their performance is not better than that of kernels [11].

Kernel Estimators: The *kernel estimator* [18] is a generalized form of sampling, whose basic step is to produce a uniform random sample. In kernel estimation each point distributes its weight in the space around it. A *kernel function* describes the form of this weight distribution, generally distributing most of the weight in the area near the point. Summing up all the kernel functions we obtain a density function for the dataset.

More formally, assume that we have a static relation, T, that stores the d-dimensional values $\mathbf{t}$, $\mathbf{t} = (t_1, \ldots, t_d)$, whose distribution we want to approximate. The recorded values must fall in the interval $[0, 1]^d$. This requirement is not restrictive, since we can map the domain of the input values to the interval $[0, 1]^d$. Let R be a random sample of T, and $k(\mathbf{x})$ a d-dimensional function of $\mathbf{x} = (x_1, \ldots, x_d)$, such that $\int_{[0,1]} k(\mathbf{x}) d\mathbf{x} = 1$, for all tuples in R. We call $k(\mathbf{x})$ the *kernel function*. We can now approximate the underlying distribution $f(\mathbf{x})$, according to which the values in T were generated, using the following function

$$f(\mathbf{x}) = \frac{1}{|T|} \sum_{\mathbf{t_i} \in R} k(x_1 - t_{i_1}, \ldots, x_d - t_{i_d}). \tag{1}$$

The choice of the kernel function is not significant for the results of the approximation [18]. Hence, we choose the Epanechnikov kernel that is easy to integrate:

$$k(\mathbf{x}) = \begin{cases} \left(\frac{3}{4}\right)^d \frac{1}{B_1 \ldots B_d} \prod_{1 \le i \le d} \left(1 - \left(\frac{x_i}{B_i}\right)^2\right) \\ \qquad\qquad \text{, if } \forall i,\, 1 \le i \le d,\, \left|\frac{x_i}{B_i}\right| < 1 \end{cases} \tag{2}$$

where $\mathbf{B} = (B_1, \ldots, B_d)$ is the bandwidth of the kernel function. We use Scott's rule to set $\mathbf{B}$ [18]: $B_i = \sqrt{5}\sigma_i |R|^{-\frac{1}{d+4}}$, where σ_i is the standard deviation of the values in T in dimension i.

Online approximation in a sliding window: For the discussion that follows, and for ease of presentation, we assume the case of a sensor network model (though, our framework is applicable to any environment with multiple streams of data), consisting of a set of sensors, each having a location on a 2-d plane and producing a stream of values. The

sensor network may be organized in a hierarchical way for scalability reasons. The idea is to organize the network using several tiers of overlapping virtual grids with different levels of granularity, ranging from small local areas at the lowest tier, to the entire network area at the highest tier (see Figure 2). We assume that each cell of the grid elects a leader, or *parent* node, that is responsible for processing the measurements of all the sensors in the cell, and for collecting values from the leader nodes of all its sub-cells in the lower level. In such an online setting, we require that each sensor maintains a model

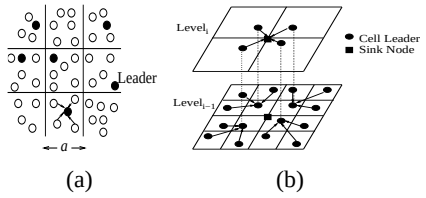

Fig. 2. (a) Sensor field organization (b) *Leader* node

for the distribution of values it generates within a sliding window W of size N (see Figure 3). The first step in creating this model is to maintain online a random sample of size $|R|$ of the set of the values in the most recent window W. The other quantity we need for the kernel estimator is the standard deviation σ of the values in the sliding window W. Both of these operations can be efficiently supported in a data streaming environment [2, 4, 13, 3].

Theorem 1. *The memory requirement for a sensor to maintain an estimate of its distribution is $O(|R|+\frac{1}{\epsilon^2}\log|W|)$, where $|R|$ is the size of the sample, ϵ is the maximum error for the estimation of the standard deviation, and $|W|$ is the size of the sliding window.*

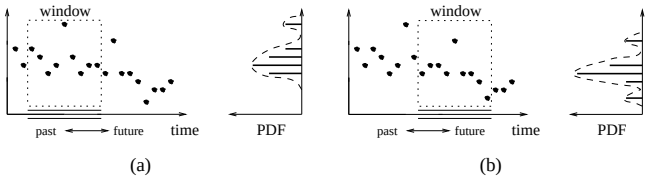

Fig. 3. Estimation of data distribution in sliding window for two time instances (1-d data).

Distributed Computation of Estimators In our framework, we define a mechanism for model composition. This allows us to take the data distribution models of two streams, and construct a single model that describes the behavior of both data streams. Our kernel estimators can be easily combined, and thus are well suited for our framework. There are two quantities that we have to combine, the sample set, R, and the bandwidth of the kernel function, B. We can combine the sample sets just by taking their union. We may then reduce the size of the resulting set by re-sampling, if necessary. In order to

combine the bandwidths of two kernel functions, we only need to combine the two standard deviations upon which the bandwidths depend. This is accomplished using the same techniques as the ones for computing the standard deviation in a sliding window of streaming data [20].

Propagating Estimator Updates in the Network Hierarchy: An interesting question is how often a node should send its model to the leader of the cell it belongs to (assuming a hierarchical organization).

The simplest approach is to have the children transmit updates to the parent as these updates take place in their own estimators. Assume that the parent has l children, each having a kernel estimator of size $|R|$, and that the kernel estimator of the parent has size $|R_p|$. Then, with probability $f = \frac{|R_p|}{l|R|}$, when a child updates its kernel estimator by adding a new kernel, it also propagates this update to its parent (i.e., it transmits the new kernel and the new standard deviation(s)). This simple approach can be used to efficiently maintain a sample with expected size $|R_p|$ at the parent, and has the important advantage that the parent's distribution quickly adjusts to changes in the distribution of the observed data.

Comparing Distributions We now discuss a method for computing the difference between two distributions, which will be useful for the algorithms we describe. Several methods have been proposed to quantify the difference between probability density distributions [15]. One widely used measure is the *Kullback-Liebler divergence* $D(p \parallel q)$ [6], which is defined as

$$D(p \parallel q) = \int_{\mathbf{y}} p(\mathbf{y})(\log p(\mathbf{y}) - \log q(\mathbf{y})), \tag{3}$$

where $p(\mathbf{y})$ and $q(\mathbf{y})$ are probability distribution functions over $\mathbf{y}$, and $\mathbf{y}$ is drawn from a finite set $\mathbf{Y}$. However, the measure is undefined when $p(\mathbf{y}) > 0$ but $q(\mathbf{y}) = 0$ for some $\mathbf{y} \in \mathbf{Y}$. The $KL - divergence$ is therefore not applicable to the density distributions derived by kernel density estimation method, because this method may assign probability of zero for regions in the domain of the values. We use a variation of the KL-divergence, called the *Jensen-Shannon divergence* [16] which is defined as follows

$$JS(p, q) = \frac{1}{2}\left[D(p \parallel avg(p, q)) + D(q \parallel avg(p, q))\right] \tag{4}$$

where $avg(p, q)$ is the average distribution $(p(\mathbf{y}) + q(\mathbf{y}))/2$.

We estimate the JS-distance between two kernel estimator models $p(\mathbf{x})$ and $q(\mathbf{x})$ as follows. We approximate the estimated distribution with the values of the function with a finite set of grid points $\mathbf{b_1}, \mathbf{b_2}, \ldots, \mathbf{b_k}$. Thus, we approximate the term $D(p \parallel avg(p, q))$ in Equation 4 as

$$D(p \parallel avg(p, q)) = \sum_{i=1\ldots k} P_p(\mathbf{b_i}, bs/2) \times \left[\log(P_p(\mathbf{b_i}, bs/2)) - \log(\tfrac{P_p(\mathbf{b_i}, bs/2) + P_q(\mathbf{b_i}, bs/2)}{2})\right] \tag{5}$$

where bs is the grid interval and P_p and P_q are the probabilities estimated with respect to the estimator models $p(\mathbf{x})$ and $q(\mathbf{x})$ respectively. We approximate the term $D(q \parallel avg(p, q))$ in Equation 4 in a similar way, and estimate the JS-divergence between the kernel estimator models. The time complexity for the above procedure is $O(dk|R|)$.

3 Applications of Framework

3.1 Outlier Detection

There exist several formal definitions of an outlier. In our work, we follow two of the commonly-used definitions.

Distance-based outliers [14]: A point $\mathbf{p}$ in a dataset T is a (D, r)-outlier if *at most* D of the points in T lie within distance r from $\mathbf{p}$. The approach to detect such outliers does not require any prior knowledge of the underlying data distribution, but rather uses the intuitive explanation that an outlier is an observation that is sufficiently far from most other observations in the dataset.

Local metrics-based outliers [17]: This method detects outliers based on the metric Multi Granularity Deviation Factor (MDEF). For any given value $\mathbf{p}$, MDEF is a measure of how the neighborhood count of $\mathbf{p}$ (in its *counting* neighborhood) compares with that of the values in its *sampling* neighborhood. A value is flagged as outlier, if its MDEF is (statistically) significantly different from that of the local averages. The parameters r, the *sampling* neighborhood and αr, the *counting* neighborhood, determine the range over which the neighborhood counts are estimated. This method takes into consideration the local density variations in the feature space, and provides an automatic cut-off for the outliers, based on the characteristics of the data.

We make two observations. The first observation is that both definitions of outliers require the estimation of the number of points within specified regions of the data space. As we discussed in Section 2, our framework for estimating the probability density function of streaming data can efficiently provide the answers to such questions.

The second observation (summarized by the following theorem) is specific to the density-based outliers, and leads to an even more efficient implementation.

Theorem 2. *Assume nodes $n_1, \ldots, n_l$ children of node n_p, data streams $S_1, \ldots, S_l$ referring to the l children nodes, and corresponding sliding windows $W_1, \ldots, W_l$. The sliding window of node n_p is defined as $W_p = \bigcup_{i=1}^{l} W_i$. Let, at some point in time, $O_1, \ldots, O_l$ be the sets of distance-based outliers corresponding to the l sliding windows. Then, for the set O_p of outliers in W_p it holds that $O_p \subseteq \bigcup_{i=1}^{l} O_i$.*

This theorem is important for two reasons: (a) it results in significant computation savings for the parent node, because it only needs to examine a very small subset of the streaming values, i.e., the outliers identified by its children; (b) it limits the necessary communication messages from the children nodes to their parents.

The experimental results show that our approach can result in approximately 95% precision and recall for identifying outliers when compared to the offline, centralized approach, while at the same time being orders of magnitude more efficient [20].

3.2 Detection and Tracking of Homogeneous Regions

The problems we are considering in this application are the following.

Problem 1. (Homogeneous Region Discovery) Given L sensors monitoring an area of interest, find a group of sensors (corresponding to a region in space) such that the observations of the sensors belonging to the same group are *similar*, and are different from those of other sensors.

Problem 2. (Homogeneous Region Tracking) For a given region R defined by a set of *similar* sensors, track its movement over time.

In order to solve these problems, we need to address the following issues. First, to detect homogeneous regions, we need an efficient technique to identify sensors with similar readings. Second, we need to provide a formulation that allows us to define and discover homogeneous regions that differ from the surrounding area.

Once again, the basis for our solution is the proposed framework that allows the efficient estimation of data densities in a distributed manner. In our solution we first estimate the data distributions within each cell, and then cluster together cells with similar distributions. During this process, we define the homogeneous regions, and approximate their boundaries, which we subsequently track as they evolve over time.

The experimental evaluation shows that we can effectively detect and track homogeneous regions, with very small processing and communication costs [19].

3.3 Other Applications

An accurate online approximation of the probability density function allows us to solve a number of problems in a sensor network.

Online Query Processing: One category of problems is to provide approximate answers to range queries with both spatial and temporal constraints. These are queries of the following form. "What is the average temperature in region (X, Y) during the time interval $[t_1, t_2]$?". In such cases, the sensors can estimate the density model for the observations during the specified time interval and answer the queries based on the estimated model.

Finding Faulty Sensors: Another important application is online detection of faulty sensors. Examples include queries of the form: "Give a warning when the values of a given sensor are significantly different from the values of its neighbors over the most recent time window W", or queries of the form: "Give a warning if the number of outliers in a given region exceeds a given threshold T over the most recent time window W". With our approach, a parent sensor can compute the difference between the estimator models received from its children, to determine if any of them is faulty.

4 Related Work

A recent study [7] proposes a sensor data acquisition technique, based on models that approximate the data with probabilistic confidences. However, any special characteristics of the data distribution, such as periodic drifts, have to be explicitly encoded in the space of models considered, which requires domain knowledge. In our work, we describe a more general technique, which can efficiently overcome this limitation.

A framework for modeling sensor network data is proposed by Guestrin et al. [9]. The goal is that the nodes in the network collaborate in order to fit a global function to each of their local measurements. Being a parametric approximation technique, it has more parameters to fit than our approach, where we only have to estimate a single parameter. A new study proposes a methodology for approximate data collection that

also exploits spatial correlations [5]. This approach significantly reduces the communication costs, but it is not clear how it can support distributed, in-network processing, or dynamically adapt to changes in the spatial correlations.

Ali et al. [1] propose an interesting approach to detect and track discrete phenomena (PDT) in sensor networks. Detecting phenomena with PDT is a centralized approach, and therefore, has high communication energy cost. In our work, we consider a more general problem and we employ a distributed approach. Hellerstein et al. [12] propose algorithms to partition the sensors into *isobars*, i.e., groups of neighboring sensors with approximately equal values during an epoch. In our case, we are partitioning the sensors according to the summary of their values over a time interval that spans several epochs, and make no a priori decisions as to how to group sensors based on their value ranges.

5 Conclusions

In this paper, we propose a general and flexible framework for approximating the distribution of streaming data. The techniques we describe operate efficiently in an online fashion. Moreover, they distribute the computation effort among the nodes available in the network, thus better exploiting the available resources and cutting back on the processing and communication costs. We evaluated our approaches by applying them to different problems, which demonstrates the versatility of the proposed approach.

References

1. M. H. Ali, M. F. Mokbel, W. G. Aref, and I. Kamel. Detection and tracking of discrete phenomena in sensor-network databases. In *SSDBM*, pages 163–172, Santa Barbara, CA, 2005.
2. B. Babcock, M. Datar, and R. Motwani. Sampling From a Moving Window Over Streaming Data. In *SODA*, 2002.
3. B. Babcock, M. Datar, R. Motwani, and L. O'Callaghan. Maintaining Variance And k-medians Over Data Stream Windows. In *PODS*, pages 234–243, San Diego, CA, USA, 2003.
4. B. Bash, J. Byers, and J. Considine. Approximately uniform random sampling in sensor networks. *DMSN*, 2004.
5. D. Chu, A. Deshpande, J. Hellerstein, and W. Hong. Approximate data collection in sensor networks using probabilistic models. *ICDE*, 2006.
6. T. M. Cover and J. A. Thomas. *Elements of Information Theory*. John Wiley & sons, 1991.
7. A. Deshpande, C. Guestrin, S. R. Madden, J. M. Hellerstein, and W. Hong. Model-Driven Data Acquisition in Sensor Networks. In *VLDB*, Toronto, ON, Canada, 2004.
8. A. C. Gilbert, Y. Kotidis, S. Muthukrishnan, and M. Strauss. Surfing Wavelets on Streams: One-Pass Summaries for Approximate Aggregate Queries. In *VLDB*, Rome, Italy, 2001.
9. C. Guestrin, P. Bodik, R. Thibaux, M. Paskin, and S. Madden. Distributed Regression: an Efficient Framework for Modeling Sensor Network Data. In *IPSN*, Berkeley, CA, 2004.
10. S. Guha and N. Koudas. Approximating a Data Stream for Querying and Estimation: Algorithms and Performance Evaluation. In *ICDE*, pages 567–576, San Jose, CA, 2002.
11. D. Gunopulos, G. Kollios, V. J. Tsotras, and C. Domeniconi. Approximating Multi-Dimensional Aggregate Range Queries over Real Attributes. In *SIGMOD*, Dallas, TX, USA, 2000.

12. J. M. Hellerstein, W. Hong, S. Madden, and K. Stanek. Beyond average: Toward sophisticated sensing with queries. In *IPSN*, pages 63–79, 2003.
13. A. Jain and E. Chang. Adaptive sampling for sensor networks. *DMSN*, 2004.
14. E. M. Knorr and R. T. Ng. Algorithms for Mining Distance-Based Outliers in Large Datasets. In *VLDB*, NY, NY, 1998.
15. L. Lee. On the effectiveness of the skew divergence for statistical language analysis. In *Artificial Intelligence and Statistics 2001*, pages 65–72, 2001.
16. J. Lin. Divergence measures based on the shannon entropy. *IEEE Trans. Infor. Theory*, 37:145–151, 1991.
17. S. Papadimitriou, H. Kitagawa, P. Gibbons, and C. Faloutsos. Loci: Fast outlier detection using the local correlation integral, 2003.
18. D. Scott. *Multivariate Density Estimation: Theory, Practice and Visualization*. Wiley & Sons, 1992.
19. S. Subramaniam, V. Kalogeraki, and T. Palpanas. Distributed Real-Time Detection and Tracking of Homogeneous Regions in Sensor Networks. In *RTSS*, Rio de Janeiro, Brazil, 2006.
20. S. Subramaniam, T. Palpanas, D. Papadopoulos, V. Kalogeraki, and D. Gunopulos. Online Outlier Detection in Sensor Data Using Non-Parametric Models. In *VLDB*, Seoul, Korea, 2006.