Efficient Estimation of Dynamic Density Functions with an Application to Outlier Detection

Abdulhakim Qahtan
King Abdullah University of Science and Technology
Thuwal 23955-6900, SA
abdulhakim.qahtan@kaust.edu.sa

Xiangliang Zhang
King Abdullah University of Science and Technology
Thuwal 23955-6900, SA
xiangliang.zhang@kaust.edu.sa

Suojin Wang
Texas A&M University
College Station, Texas
77843-1372, USA
sjwang@stat.tamu.edu

ABSTRACT

In this paper, we propose a new method to estimate the dynamic density over data streams, named KDE-Track as it is based on a conventional and widely used Kernel Density Estimation (KDE) method. KDE-Track can efficiently estimate the density with linear complexity by using interpolation on a kernel model, which is incrementally updated upon the arrival of streaming data. Both theoretical analysis and experimental validation show that KDE-Track outperforms traditional KDE and a baseline method Cluster-Kernels on estimation accuracy of the complex density structures in data streams, computing time and memory usage. KDE-Track is also demonstrated on timely catching the dynamic density of synthetic and real-world data. In addition, KDE-Track is used to accurately detect outliers in sensor data and compared with two existing methods developed for detecting outliers and cleaning sensor data.

Categories and Subject Descriptors

H.2.8 [**Database Management**]: Database Applications-Data mining

General Terms

Algorithms, Experimentation, Performance

Keywords

Density Estimation, Interpolation, Data Streams, Outlier Detection

1. INTRODUCTION

Data streams can be widely found in applications such as sensor network and internet management. The unbounded, rapid and continuous arrival of data streams disallows the usage of traditional data mining techniques. Therefore, the development of algorithms for processing data streams instantaneously becomes highly important.

CIKM'12, October 29–November 2, 2012, Maui, HI, USA.

This paper targets on estimating the dynamic density that comes with the evolving data streams. The estimated density characterizes the underlying distribution that can be used in many applications such as online outlier detection and online clustering. Comparing with other data stream mining work, e.g., clustering, classification and frequent pattern mining, density estimation of data stream is much less mature. Besides the problem of estimating the density using samples drawn from an unknown distribution that researchers face in case of stationary data, data streams have more challenging properties that complicate the estimation of the Probability Density Function (PDF). In data streams, the theoretical PDF changes dynamically in an unpredictable fashion. Therefore, PDF estimation should rely more on the recent data [8]

Approaches for estimating the density function of samples drawn from a distribution can be categorized into *parametric* and *nonparametric* methods. In this paper, we consider the non-parametric approaches as they make no assumptions on the data distribution. Kernel Density Estimation (KDE) is one of the most popular non-parametric density estimation techniques and has been shown in [13] to converge to the true density function as the number of samples drawn from the distribution goes to infinity. Although KDE is a promising technique, its high computational and space costs hinder its usage on large-size and streaming data.

Zhou et. al. [16] and Heinz et. al. [8] introduced the concepts of M-Kernel and Cluster Kernels (C-Kernels) respectively to minimize the space requirements of KDE. In both methods, each kernel summarizes a cluster of similar samples. When estimating the PDF of big data, they both treat samples one by one in order. Each new sample can either fall into one existing kernel or trigger a new kernel. If the number of kernels exceeds a user defined number, two kernels are merged based on a cost function value. When estimating PDF, only one sample is used to represent all samples in one cluster, which affects the estimation accuracy. C-Kernels introduces data resampling. Regenerating all of the samples might be accurate but very costly in computations.

In this paper, we build our density estimator based on KDE method due to its advantages for estimating the true density [13]. Our proposed method, called KDE-Track, models the data distribution as a set of resampling points with their estimated PDF. In order to overcome the quadratic time complexity and linear space complexity problem of KDE when evaluating the PDF for each new observation, linear interpolation is used with KDE for online density es-

timation. This technique was discussed in [9] and known as Kernel Polygons but for stationary data sets and using fixed number of the resampling points.

Our KDE-Track has three unique properties:
(1) updating the resampling model is done online with a linear time and space complexities w.r.t. the model size;
(2) estimating the PDF of a new arriving data sample is done in a time linearly proportional to the model size;
(3) the adaptive sampling of PDF improves the estimation accuracy. More (less) points are re-sampled in the areas where PDF has a larger curvature (is approximately linear).

Both theoretical analysis and experimental results on synthetic and real-world data show the effectiveness of our approach for dynamic density function estimation. We also apply the density estimation to online outlier detection. Momentary outliers in the evolving data, which are vague in all observations, can be effectively reported through monitoring the dynamics of the estimated density.

The rest of the paper is organized as follows: Section 2 presents our approach KDE-Track with the KDE interpolation and its error bound. Section 3 presents evaluation results. Section 4 demonstrates the application of KDE-Track to outlier detection. Section 5 concludes our work.

2. ONLINE DENSITY ESTIMATION

In this section, we first describe the KDE interpolation method, then analyze its theoretical error bounds that guarantee the accuracy of density estimation, and finally present the KDE-Track method for online density estimation.

Given a set of samples, $\{x_1, x_2, ..., x_m\}$, KDE estimates the density at a point x as $\hat{f}(x) = \left(\frac{1}{mh_m}\right)\sum_{j=1}^{m} K\left(\frac{x-x_j}{h_m}\right)$, where $K(\cdot)$ is a kernel function, h_m is a smoothing parameter called the bandwidth and computed from the m samples. The choice of the kernel function has no significant effects on the estimation provided that the kernel function is continuous with finite support [15]. We choose the Epanechnikov kernel in this paper, $K(x) = \frac{3}{4}(1-x^2)I_{[-1,1]}(x)$. The estimation accuracy of KDE is mainly affected by the bandwidth value [13, 15]. It is crucial to set the bandwidth value by considering the statistic properties of samples. The bandwidth value we used is set according to the *normal rule* that has been widely used and proven to be accurate enough [15]. That is, $h_m = 1.06\hat{\sigma}m^{-\frac{1}{5}}$, where $\hat{\sigma}$ is the estimated standard deviation of the m samples.

KDE stores all the data samples received so far and use them all to estimate the PDF of any given point. Thus, the space requirement for KDE is linearly proportional to the sample size and the time complexity for online density estimation will be quadratic.

2.1 Density Estimation by Interpolation

We model the data stream distribution as a set of sorted resampling points and their corresponding estimated density, $\mathcal{S} = \{\mathcal{S}^{(1)}, \mathcal{S}^{(2)}, ..., \mathcal{S}^{(q)}\}$, where q is the number of the resampling points, and $\mathcal{S}^{(i)}$ is an ordered pair representing a point and its estimated PDF ($\mathcal{S}^{(i)} = (\mathcal{S}^{(i)}_{1,1}, \mathcal{S}^{(i)}_{1,2})$ with $\mathcal{S}^{(i)}_{1,2} = \hat{f}(\mathcal{S}^{(i)}_{1,1})$). The linear interpolation is based on constructing the set of resampling points and the corresponding estimated density values. Estimating the PDF at a point x by linear interpolation of the resampling points has the following 2 steps:
(1) find two adjacent resampling points $\mathcal{S}^{(k)}_{1,1}$ and $\mathcal{S}^{(k+1)}_{1,1}$ such that $\mathcal{S}^{(k)}_{1,1} < x \leq \mathcal{S}^{(k+1)}_{1,1}$, and compute the distance between them $d(\mathcal{S}^{(k)}_{1,1}, \mathcal{S}^{(k+1)}_{1,1}) = |\mathcal{S}^{(k+1)}_{1,1} - \mathcal{S}^{(k)}_{1,1}|$;
(2) estimate the density at x by interpolation

$$\tilde{f}(x) = \frac{d(x, \mathcal{S}^{(k+1)}_{1,1})\mathcal{S}^{(k)}_{1,2} + d(\mathcal{S}^{(k)}_{1,1}, x)\mathcal{S}^{(k+1)}_{1,2}}{d(\mathcal{S}^{(k)}_{1,1}, \mathcal{S}^{(k+1)}_{1,1})}. \quad (1)$$

KDE interpolation is efficient as it stores only the function at the resampling points whose total number is in the constant order and is small compared to the sample size. The running time for estimating the PDF for all n arriving data samples will be in $O(n|\mathcal{S}|)$.

2.2 Error Bound of Linear Interpolation

This subsection discusses the error bound of density estimated by linear interpolation.

PROPOSITION 2.1. *The density estimated by linear interpolation is*

$$\tilde{f}(x) = \hat{f}(x) + \frac{1}{2}\{d(\mathcal{S}^{(k)}_{1,1}, x)d(x, \mathcal{S}^{(k+1)}_{1,1})\}\hat{f}''(x) + O_p(\{d(\mathcal{S}^{(k)}_{1,1}, \mathcal{S}^{(k+1)}_{1,1})\}^3). \quad (2)$$

Compared with the KDE estimation $\hat{f}(x)$, the error incurred in linear interpolation has an upper bound

$$|\tilde{f}(x) - \hat{f}(x)| \leq \frac{\{d(\mathcal{S}^{(k)}_{1,1}, \mathcal{S}^{(k+1)}_{1,1})\}^2}{8}\hat{f}''(x) + O_p(\{d(\mathcal{S}^{(k)}_{1,1}, \mathcal{S}^{(k+1)}_{1,1})\}^3).$$

PROOF. Assume that a point x locates between $\mathcal{S}^{(k)}_{1,1}$ and $\mathcal{S}^{(k+1)}_{1,1}$, where $\mathcal{S}^{(k)}_{1,1} < x \leq \mathcal{S}^{(k+1)}_{1,1}$. The estimated density by KDE at $\mathcal{S}^{(k)}_{1,1}$ and $\mathcal{S}^{(k+1)}_{1,1}$ can be rewritten by Taylor's expansion as:

$$\hat{f}(\mathcal{S}^{(k+1)}_{1,1}) = \sum_{i=0}^{\infty} \frac{\{d(x, \mathcal{S}^{(k+1)}_{1,1})\}^i}{i!}\hat{f}^{(i)}(x), \quad (3)$$

$$\hat{f}(\mathcal{S}^{(k)}_{1,1}) = \sum_{i=0}^{\infty} \frac{\{-d(\mathcal{S}^{(k)}_{1,1}, x)\}^i}{i!}\hat{f}^{(i)}(x), \quad (4)$$

where $\hat{f}^{(i)}(x)$ is the i-th derivative of $\hat{f}(x)$. Substituting Equations (3) and (4) into the interpolation Equation (1), we have Equation (2) that gives the relationship between $\tilde{f}(x)$ and $\hat{f}(x)$.

When x is in the middle of the interval $[\mathcal{S}^{(k)}_{1,1}, \mathcal{S}^{(k+1)}_{1,1}]$, $\tilde{f}(x)$ deviates from $\hat{f}(x)$ most with the maximum error of $\frac{\{d(\mathcal{S}^{(k)}_{1,1}, \mathcal{S}^{(k+1)}_{1,1})\}^2}{8}\hat{f}''(x) + O_p(\{d(\mathcal{S}^{(k)}_{1,1}, \mathcal{S}^{(k+1)}_{1,1})\}^3)$. □

Note that this error bound is calculated w.r.t. the estimated density using KDE instead of the theoretical true density. Hence, the density $\tilde{f}(x)$ will contain an extra propagated error from the KDE. However, as observed in our experimental results, interpolation can improve KDE and result in better accuracy.

2.3 Adaptive Resampling Model

From Proposition 2.1, we know that the accuracy of the linear interpolation depends on 1) the distance between two adjacent resampling points; and 2) the second derivative of the density function. To minimize the error while keeping the number of resampling points within a reasonable margin, we add more resampling points in the regions where the density function has high curvature. By contrast, in the regions where the function is approximately linear, we use less number of resampling points. In this sense, we can

rewrite the resampling set as $\mathcal{S} = \{\mathcal{S}^{(1)}, \mathcal{S}^{(2)}, ..., \mathcal{S}^{(q)}\}$ where $\mathcal{S}^{(i)} \in \mathbb{R}^{c\times 2}$, $c \geq 1$. The two columns of $\mathcal{S}^{(i)}$ present the resampling points and their corresponding estimated PDFs, respectively. The j-th resampling point in the sublist $\mathcal{S}^{(i)}$ will be accessed as $\mathcal{S}^{(i)}_{j,1}$ and its corresponding function value is $\mathcal{S}^{(i)}_{j,2}$. This data structure of resampling model improves the time required for accessing the interval including x. Accessing the target interval can be done by advancing an iterator over the list instead of performing data comparisons.

2.4 KDE-Track Method

To estimate the density for each incoming data sample, KDE-Track only requires to access two resampling points as we use the linear interpolation technique discussed in Subsection 2.1. The key step is thus the maintenance of resampling model (points and their PDF).

The resampling model is initialized by the beginning part of streaming data, e.g., the first 20K points. The resampling points, $\mathcal{S}^{(i)}_{1,1}, i = 1, ..., q$, are defined as a set of equidistant points within the range of initial points received so far. The PDF values $\mathcal{S}^{(i)}_{1,2}$ of these resampling points are computed using a traditional KDE on the batch of initial points.

Once all $\mathcal{S}^{(i)}$ have been initialized, we can estimate the density at each arriving point x. Due to our interpolation method, the density at x can be estimated by 1) calculating the index of one resampling point who and whose successive neighbor will contribute; and 2) computing $\tilde{f}(x)$ by interpolation Equation (1).
The calculation of the index k is done using the equation:

$$k = \text{integer}((x - \mathcal{S}^{(1)}_{1,1})/s), \tag{5}$$

where s is the equidistance between $\mathcal{S}^{(j+1)}_{1,1}$ and $\mathcal{S}^{(j)}_{1,1}$, $\forall j, 1 \leq j \leq q$ positions. If the interval at position k has been further divided, we perform a sequential search within the internal list on c resampling points. The density estimation process of KDE-Track is described in Algorithm 1.

Algorithm 1 KDE-Track: estimate density $\tilde{f}(x)$

Given: x, resampling model $\mathcal{S}$ and interval length s
$k = \text{integer}((x - \mathcal{S}^{(1)}_{1,1})/s)$
if $x = \mathcal{S}^{(k)}_{1,1}$ **then**
  $\tilde{f}(x) = \mathcal{S}^{(k)}_{1,2}$
else if $RowNumber(\mathcal{S}^{(k)}) = 1$ **then**
  $\tilde{f}(x) = \frac{(\mathcal{S}^{(k+1)}_{1,1} - x) * \mathcal{S}^{(k)}_{1,2} + (x - \mathcal{S}^{(k)}_{1,1}) * \mathcal{S}^{(k+1)}_{1,2}}{s}$
else
  search the sublist of $S^{(k)}$ until reaching i such that $\mathcal{S}^{(k)}_{i,1} \leq x < \mathcal{S}^{(k)}_{i+1,1}$ or $i = c$ (end of sublist)
  if $i < c$ **then**
    $\tilde{f}(x) = \frac{(\mathcal{S}^{(k)}_{i+1,1} - x) * \mathcal{S}^{(k)}_{i,2} + (x - \mathcal{S}^{(k)}_{i,1}) * \mathcal{S}^{(k)}_{i+1,2}}{\mathcal{S}^{(k)}_{i+1,1} - \mathcal{S}^{(k)}_{i,1}}$
  else
    $\tilde{f}(x) = \frac{(\mathcal{S}^{(k+1)}_{1,1} - x) * \mathcal{S}^{(k)}_{c,2} + (x - \mathcal{S}^{(k)}_{c,1}) * \mathcal{S}^{(k+1)}_{1,2}}{\mathcal{S}^{(k+1)}_{1,1} - \mathcal{S}^{(k)}_{c,1}}$
  end if
end if

As the resampling model is the basis of our density estimation, the PDF values of the resampling points should be updated after receiving a new data point. Updating the densities of resampling points in the model $\mathcal{S}$ should consider the evolution of the distribution. We use a sliding window strategy to catch the evolution over time. Let w denote the predefined parameter of window size, and n_t denote the number of points we have received until time t. Due to the difference between w and n_t, there are two different scenarios when updating the model $\mathcal{S}$, more specifically, updating the bandwidth and density function values $\mathcal{S}^{(i)}_{j,2} = \hat{f}(\mathcal{S}^{(i)}_{j,1})$.

When $n_t \leq w$. The received points cannot fill the whole window. The bandwidth value at time t is calculated by using all n_t points by the formula $\hat{h}_t = 1.06\hat{\sigma}_t n_t^{-1/5}$, where $\hat{\sigma}_t$ is the sample variance of the received data samples calculated as $\hat{\sigma}_t^2 = \frac{1}{n_t - 1}\left\{\sum_{j=1}^{n_t} x_j^2 - \frac{1}{n_t}\left(\sum_{j=1}^{n_t} x_j\right)^2\right\}$ [5], which can be updated with a constant time at each t.

After receiving a point x_t, the density at a resampling point x at time t is updated using the equation:

$$\hat{f}_t(x) = \frac{n_t - 1}{n_t}\hat{f}_{t-1}(x) + \frac{1}{n_t \hat{h}_t} K\Big(\frac{x - x_t}{\hat{h}_t}\Big). \tag{6}$$

When $n_t > w$. In this case, the bandwidth is calculated on the most recently received w points inside the window as follows: $\hat{h}_t = 1.06\hat{\sigma}_t w^{-1/5}$, where the sample variance $\hat{\sigma}_t^2$ can be easily updated by $\hat{\sigma}_t^2 = \frac{1}{w-1}\left\{\sum_{j=i+1}^{i+w} x_j^2 - \frac{1}{w}\left(\sum_{j=i+1}^{i+w} x_j\right)^2\right\}$.

The density $\hat{f}_t(x)$ is updated by absorbing the new arrived point x_t and deleting the old point that moved out from the window,

$$\hat{f}_t(x) = \frac{1}{w}\Big\{w\hat{f}_{t-1}(x) + \frac{1}{\hat{h}_t} K\Big(\frac{x - x_t}{\hat{h}_t}\Big) - \frac{1}{\hat{h}_{t-w}} K\Big(\frac{x - x_{t-w}}{\hat{h}_{t-w}}\Big)\Big\}.$$

The probabilistic properties of updated density function $\hat{f}_t(x)$ can be proved as:
(i) $\hat{f}_t(x) \geq 0, \forall x$, due to the fact that
$\hat{f}_t(x)$ is a summation of nonnegative terms;
(ii) the integration $\int_{-\infty}^{\infty} \hat{f}_t(x)dx = 1$. This is because the integration of $K(\frac{(x - x_j)}{\hat{h}_j})/\hat{h}_j = 1$, $\forall j$.

Updating the resampling model should consider the changes in the regions of the PDF with high curvature as they change over time. We should also consider the extension of the model to cover the range of the data and shrinking the model when data from some regions are no longer available. Such changes can not be observed frequently. Checking for such changes is done after receiving a predefined number of points that represents a ratio of $5 - 20\%$ of the sliding window size.

2.5 Time and Space Complexity Analysis

Based on the discussion in the Subsection 2.4, the time complexity of estimating the density for a new incoming data point is linearly proportional to the model size $\mathcal{S}$ regardless the number of points that have been received from the data stream. Updating the model when receiving a new point requires computing time linear to the total number of the resampling points $|\mathcal{S}| = \sum_{i=1}^{q} |\mathcal{S}^{(i)}|$, since all the function values at the resampling points are updated. The overall time complexity of processing each arriving point is linear to the model size, which is usually a limited small number. The time required to online density estimation for a data stream with n points is $O(n|\mathcal{S}|)$, which linearly increase with the number of received points.

During the online density estimation process, KDE-Track only keeps the resampling model $\mathcal{S}$ in memory. Therefore, the memory usage is the model size $|\mathcal{S}|$.

3. PERFORMANCE EVALUATION

In order to evaluate our method, we run extensive number of experiments on both synthetic data and real world data from the Intel Berkeley Research Lab [2].

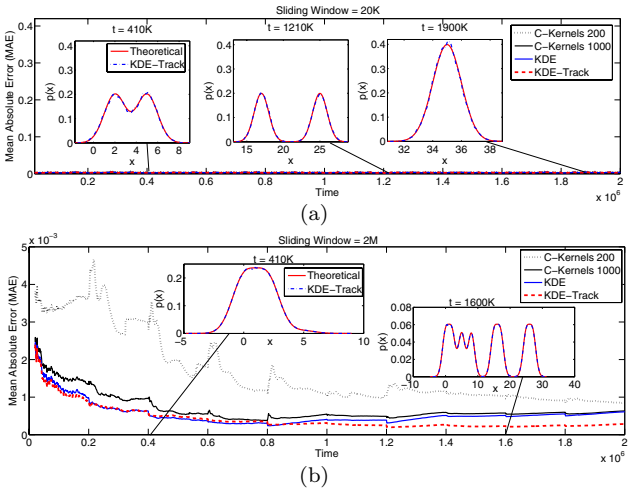

Figure 1: MAE incurred by KDE-Track, KDE and C-Kernels when estimating the density function of the synthetic data set "Dyn10" using window size $w = 20K$ (a) and $w = 2M$ (b).

3.1 Synthetic Data

The synthetic data were generated by varying the mixture of several normal distributions along time. We validated our method on several synthetic data sets with various mixtures, but only present the results on a data stream with 2 million points derived from a dynamic mixture of ten normal distributions (named "Dyn10", each distribution contributes 200K points) due to space limitation.

To evaluate the accuracy of the estimation, we defined evaluation checkpoints at every 1000 samples in the data stream. At each checkpoint, 1000 evaluation points that are uniformly sampled within the range of the received data were generated. The mean absolute error (MAE) is then computed as $MAE = \frac{1}{1000}\sum_{i=1}^{1000}|\tilde{f}(e_i) - f(e_i)|$.

Figure 1 (a) shows the error incurred by the evaluated methods KDE-Track, KDE and C-Kernels with window size $w = 20K$. All methods show very small MAE values around $2.5{*}10^{-3}$, and C-Kernels 200 has a slightly higher MAE than others[1]. The embedded subfigures show both the true dynamic densities and the estimated densities by KDE-Track at different checkpoints. We can see that our method can capture the changes and accurately estimate the dynamic density all the time.

C-Kernels is not designed to capture the dynamic density. At each evaluation checkpoint, we delete the old model created by the previous sliding window and create a new model based on the current window. This adaptation preserves the estimation accuracy of C-Kernels but shows to be impractical for online density estimation due to the high computational cost, as shown in Table 1.

Figure 1 (b) compares the MAE of KDE-Track, KDE and C-Kernels when setting window size to be $2M$. In this case, the underlying distribution is still dynamic but more complex. Our method is shown to perform very well with even smaller errors than KDE, and is more stable. The stability and better accuracy of KDE-Track are due to the incremental update policy, which uses different bandwidth values for arriving samples when updating the model. C-Kernels shows bigger errors when data from a new distribution arrive. Using more clusters could reduce the impact of the dynamic behavior of the data as this example shows, but increase the memory usage and processing time.

[1]MAEs are plotted on the scale of PDF to show that errors are relatively very small compared to the density values.

Table 1: Running time and memory usage of KDE-Track, KDE, and C-Kernerls on "Dyn10" data

	$w = 20K$		$w = 2M$	
Method	Run-Time (sec)	Memory (KB)	Run-Time (sec)	Memory (KB)
KDE-Track	163	164	157	6.9
CK-200	2552	82.4	166	2.4
CK-1000	11487	92	737	12.0
KDE	1265	80	44928	0 to 8000

3.2 Real World Data

We studied the dynamic behavior of real world data sets from the Intel Berkeley Research Lab [2]. Only the results of *temperature* is shown in Figure 2 due to space limitations. The dynamic distribution is demonstrated in 9 subfigures, which are the snapshots of the estimated density functions at random checkpoints. This evaluation used a sliding window with size $w = 20K$ that is approximately a three-hour time window. While we cannot evaluate the estimation accuracy since the theoretical probability density functions are unknown, the figure shows the dynamic changes in the PDF over time. These changes should be considered when making decisions in applications like outlier detection.

We can also notice from the figures that at the end of the data stream, readings becoming inconsistent as we get many temperature values around 122. These inconsistent readings appear because of low battery power in the sensors as reported in [14].

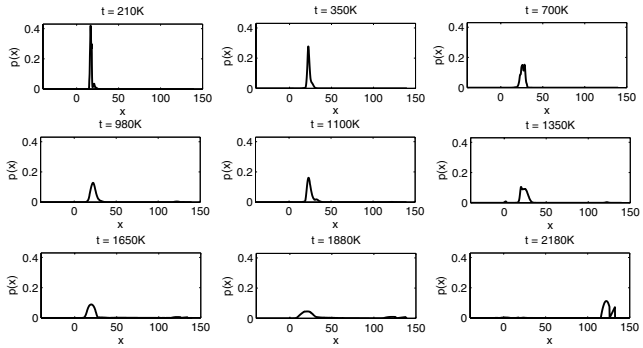

Figure 2: Dynamic density of *temperature* estimated by KDE-Track.

4. DENSITY-BASED OUTLIER DETECTION

There are many approaches for outlier detection that differ in the intuition of outliers and the way reporting outliers. Statistical based approaches identify outliers that do not fit in the statistical model [7]. Distance-based outlier detection method reports p as a $DB(\pi, d_{min})$-outlier if at least percentage π of the objects in a data set lies in distance greater than d_{min} from p [10]. Density-based approaches report a point as an outlier if the density in its neighborhood is too different from the densities around its neighbors

[4, 11]. The main problem in these approaches is that prior knowledge about data application is required for parameter setting. The high computational time is also another issue.

Auto-Regression is widely used for outlier detection as it is timely inexpensive [12]. An auto-regression based outlier detection (AROD) proposed in [6] identifies malicious nodes in wireless sensor network by building an AR-model to predict the value of the new incoming data based on the latest set of observed data samples. If the difference between the predicted and the actual values is greater than a predefined threshold then the point is declared as an outlier.

A median-based outlier detection (MBOD) method detects and removes outliers from sensors data based on comparing the distance between the new arriving data and the median in a window of the most recent received data [3]. A sample is reported as an outlier and removed if its computed distance is greater than a given threshold.

Our approach for outlier detection is based on the intuition that outliers have small PDF values compared to other points. Given a new observation x, the PDF value $\tilde{f}(x)$ is computed and used as an indicator of the density in the neighborhood of x. The average density value of the resampling points, $\bar{f}$, is used to define the threshold. If $\tilde{f}(x)$ is smaller than 5% of $\bar{f}$, x is reported as a suspicious outlier and held in a buffer. A suspicious outlier is released if its density value increases up to a value larger than the threshold, $0.05 * \bar{f}$. Using sliding window in our density estimation provides a temporal locality as x is compared to the average density value estimated using the last w data samples.

We evaluated our approach for outlier detection by comparing with MBOD [3] and AROD [6] on data sets from [1]. Figure 3 (a) shows the outliers detected by the three methods (MBOD, AROD and KDE-Track) when applied on the dew point data set from [1] from January to June, 2011, with more than $260K$ data points. The figure shows that all methods are detecting outliers effectively, but KDE-Track reports more outliers. Zooming in these outliers (the small figure in the left), we find a fact that dew point values were decreasing very fast, which can be considered as an unusual behavior and should be reported.

Figure 3 (b) shows the outliers detected by the three methods when applied on the humidity data from [1] (March-May 2003), with more than $100K$ data points. MBOD and AROD failed to detect outliers correctly, while KDE-Track reports only those points that have either very small or very large values. The small figure on the left shows that the outliers reported by MBOD and AROD are in the middle of dense regions with many normal data samples around. These reported outliers can be considered as false positives.

5. CONCLUSION

In this paper, we studied the problem of estimating the dynamic density that comes with data streams. We proposed a new method KDE-Track that can timely track the evolving distribution and accurately estimate the probability density function of data streams, which is incrementally updated on the arrival of new data samples. The effectiveness and efficiency of KDE-Track have been analytically studied and experimentally demonstrated on synthetic and real data, and on detecting outliers. This work leads to several perspectives on dynamic density estimation, e.g., applications to density-based clustering, window size and bandwidth adaption in streaming data.

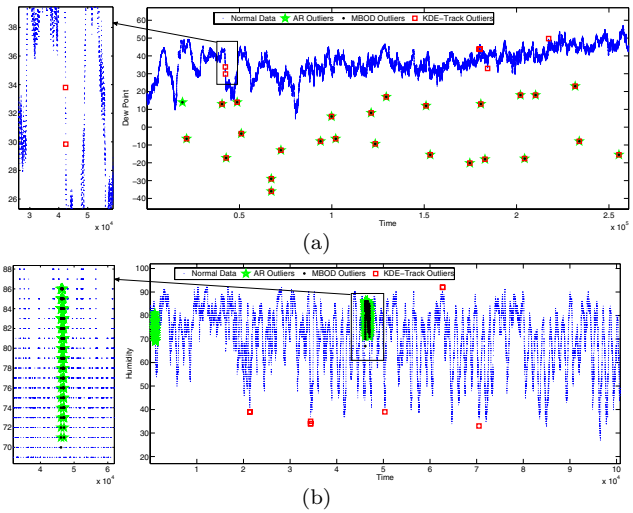

Figure 3: Outliers detected by MBOD, AROD and KDE-Track in the *Dew Point* (a) and *Humidity* (b)

References

[1] Earth climate and weather, http://www.atmos.washington.edu/cgi-bin/list_uw.cgi.

[2] Intel lab data, http://db.csail.mit.edu/labdata/labdata.html.

[3] S. Basu and M. Meckesheimer. Automatic outlier detection for time series: an application to sensor data. *KNOWL INF SYST*, 2007.

[4] M. Breunig, H. Kriegel, R. Ng, and J. Sander. LOF: Identifying density-based local outliers. In *SIGMOD*, 2000.

[5] T. Chan, G. Golub, and R. LeVeque. Algorithms for computing the sample variance: Analysis and recommendations. *The American Statistician*, 1983.

[6] D. Curiac, O. Banias, and O. Dranga. Malicious node detection in wireless sensor networks using an autoregression technique. In *ICNS*, 2007.

[7] D. H. Freedman, R. Pisani, and R. Purves. *Statistics*. W. W. Norton & Company., 1978.

[8] C. Heinz and B. Seeger. Cluster kernels: Resource-aware kernel density estimators over streaming data. *TKDE*, 2008.

[9] M. C. Jones. Discretized and interpolated kernel density estimates. *J. of the American Statistical Association*, 1989.

[10] E. Knorr and R. Ng. Algorithms for mining distance-based outliers in large datasets. In *VLDB'98*.

[11] S. Papadimitriou, H. Kitagawa, P. Gibbons, and C. Faloutsos. LOCI: Fast outlier detection using the local correlation integral. In *ICDE*, 2003.

[12] S. M. Sadik and L. Gruenwald. DBOD-DS: Distance based outlier detection for data streams. In *DEXA'10*.

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

[14] A. Sharma, L. Golubchik, and R. Govindan. On the prevalence of sensor faults in real-world deployments. In *SECON*, 2007.

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

[16] A. Zhou, Z. Cai, L. Wei, and W. Qian. M-kernel merging: Towards density estimation over data streams. In *DASFAA*, 2003.