Online and Unsupervised Anomaly Detection for Streaming Data Using an Array of Sliding Windows and PDDs

Lingyu Zhang, Jiabao Zhao, and Wei Li, *Senior Member, IEEE*

***Abstract*—In this article, we propose an online and unsupervised anomaly detection algorithm for streaming data using an array of sliding windows and the probability density-based descriptors (PDDs) (based on these windows). This algorithm mainly consists of three steps: 1) we use a main sliding window over streaming data and segment this window into an array of nonoverlapping subwindows; 2) we propose the PDDs with dimension reduction, based on the kernel density estimation, to estimate the probability density of data in each subwindow; and 3) we design the distance-based anomaly detection rule to determine whether the current observation is anomalous. The experimental results and performances are presented based on the Numenta anomaly benchmark. Compared with the anomaly detection algorithm using the hierarchical temporal memory proposed by Numenta (which outperforms a wide range of other anomaly detection algorithms), our algorithm can perform better in many cases, that is, with higher detection rates and earlier detection for contextual anomalies and concept drifts.**

***Index Terms*—Anomaly detection, concept drift, contextual anomaly, data sequence, kernel density estimation (KDE), sliding window, streaming data.**

I. Introduction

Anomaly detection for streaming data is the process of identifying unexpected items that deviate from previous normal behaviors and has significant applications in many disciplines [4]–[9], for example, cloud applications [4]; environment monitoring [5]; network anomaly detection [6], [7]; etc. Anomalies in streaming data can be *spatial*, *contextual*, or *concept drifts* [1], [2], [9].

Many anomaly detection techniques for streaming data have been developed over the years [2], [10]–[26]. Some techniques are supervised. For example, an anomaly detection approach based on the supervised machine learning is proposed in [11]. Ensemble classifiers are utilized by training the models with labels for anomalous data, for network intrusion detection in [12]. These approaches rely on labeled training data, which are typically expensive to produce [13].

In many cases, the problem of anomaly detection is typically unsupervised (e.g., [13] and [15]), implicitly assuming that the dataset contains only few anomalies [1]. Anomaly detection for streaming data is performed traditionally based on statistical analysis [14].

In most real-world applications, usually, we have no prior knowledge about streaming data. Hence, the nonparametric density estimation approaches are very appealing, as they make no assumptions regarding the underlying distribution [27]. Some density estimation-based online approaches have been proposed in recent years. For example, a fully unsupervised approach is proposed in [15] based on the recursive density estimation, which is sensitive to *abrupt changes*, however, regular abrupt changes will also be detected as anomalies by [15], which ignores *temporal context* within streaming data. The approaches [16], [17], using typicality and eccentricity data analytics, are proposed based on the spatial proximity in data space. Some approaches based on the kernel density estimation (KDE) are also provided (e.g., [18] and [19]), estimating the data distribution and determining if the current observation is the realization of such probability distribution. These approaches [16]–[19] may perform well in detecting *spatial* anomalies, while it is hard for them to detect subtle *contextual* anomalies or *concept drifts*. In addition, the *early* anomaly detection is valuable [2], and algorithms are expected to detect anomalies as early as possible.

Recently, a novel anomaly detection algorithm is proposed by Numenta [2], which performs well in detecting the contextual anomalies and concept drifts; this algorithm is based on hierarchical temporal memory (HTM) [28], which is an online sequence memory method. Numenta [2] also builds the Numenta anomaly benchmark (NAB) [3], which is a benchmark containing real-world streaming data with labeled anomalies. For the following real-time anomaly detection algorithms: the HTM-based algorithm [2], EXPoSE [22], Twitter's anomaly detection [23], Etsy's Skyline [24], multinomial relative entropy [25], and Bayesian online changepoint detection [26]; comparative results in [2] show that the HTM-based algorithm [2] performs best based on the NAB. In the following text, we will show that our algorithm can still perform better than the HTM-based algorithm [2] in many cases based on the NAB.

To process streaming data, there are some kinds of window models. For example, the landmark window model is to analyze the whole streaming data to mine frequent patterns [29]; the damped window model is to differentiate recently generated data from historical data [30], [31]; and the sliding window model is to highlight recently generated data [32].

In this article, we propose an online and unsupervised anomaly detection algorithm, which mainly consists of three steps. First, we use a main sliding window with a constant size over streaming data, containing the most recent observations, and segment this window into an array of nonoverlapping and equal-length subwindows (this setting of subwindows is to granulate the streaming data, similar to the granularity in [33]–[36]). Then, we propose our probability density-based descriptors (PDDs) with dimension reduction, to estimate the probability density of data in each subwindow. The PDD is based on KDE [37], [38], which is implemented by using the K-dimension tree data structure [39], to further speed up the estimation and reduce the computational complexity of our algorithm. Finally, we design the distance-based anomaly detection rule to determine whether the current observation is anomalous.

Our algorithm is unsupervised (i.e., it requires no labeled data to train the model) and robust to anomalies (i.e., it requires no specific distributional assumptions). Our algorithm is for real-time anomaly detection, that is, in an online manner, with neither offline learning nor relying on look-ahead to label previously seen anomalous

Manuscript received January 24, 2019; revised May 24, 2019; accepted July 28, 2019. This article was recommended by Associate Editor P. P. Angelov. *(Corresponding author: Wei Li.)*

The authors are with the Department of Control and Systems Engineering, Nanjing University, Nanjing 210093, China (e-mail: wei.utdallas@live.com).

This article has supplementary downloadable material available at http://ieeexplore.ieee.org, provided by the author.

Color versions of one or more of the figures in this article are available online at http://ieeexplore.ieee.org.

Digital Object Identifier 10.1109/TCYB.2019.2935066

TABLE I
NOTATIONS IN THIS ARTICLE

Notations	Meaning
$\mathbb{N}^+$	the positive integer set $\{x \in \mathbb{N} \mid x > 0\}$
$\mathbb{R}^+$	the positive number set $\{x \in \mathbb{R} \mid x > 0\}$
$x_t \in \mathbb{R}$	the observation at time t
W	the main sliding window, refer to (1)
$N \in \mathbb{N}^+$	the size of W, refer to (1)
$w_k \subset W$	the k-th sub-window of W, refer to (2)
$n \in \mathbb{N}^+$	the size of each sub-window, refer to (2)
$m \in \mathbb{N}^+$	the number of the sub-windows, refer to (2)
$\lfloor\cdot\rfloor$	the floor function, $\lfloor x\rfloor = \max\{m \in \mathbb{Z} \mid m \le x\}$ [42]
T	the target set, refer to (3)
$p \in \mathbb{N}^+$	the size of T, refer to (3)
$W_{\text{min}} \in \mathbb{R}$	the minimum value of W, refer to (3)
$W_{\text{max}} \in \mathbb{R}$	the maximum value of W, refer to (3)
$y_j \in \mathbb{R}$	the j-th value of T, refer to (3)
$h \in \mathbb{R}^+$	the bandwidth parameter, refer to (4)
$\sigma \in \mathbb{R}^+$	the standard deviation of W, refer to (4)
$f_w(\cdot\|k)$	the PDF of w_k, refer to (5)
$f_{w\|k} \in \mathbb{R}^p$	the PDD of w_k, refer to (6)
$l_t \in \{0,1\}$	the indicator variable of x_t, Definition 8
$d(\cdot,\cdot) \in [0,+\infty)$	the distance between vector parameters, refer to (7)
D	the distance set of size $m-2$, Definition 9
C	the absolute difference set for D, refer to (8)
$\widetilde{D}_{\text{max}} \in \mathbb{R}^+$	the expanded maximum distance, refer to (9)
$D_{\text{avg}} \in \mathbb{R}^+$	the average distance of D, refer to (10)
$s_t \in \{0,1\}$	another indicator variable of x_t, refer to (12)

data. Our algorithm features a high detection rate, as it cannot only effectively detect spatial anomalies but also detect subtle contextual anomalies and concept drifts as early as possible. Compared with the HTM-based algorithm [2] (which outperforms a wide range of other anomaly detection algorithms), our algorithm can perform better in many cases based on the NAB [3].

The remainder of this article is structured as follows. Section II describes the problem. Section III proposes our algorithm. Section IV presents the experimental results and performances. Section V is the conclusion. Table I lists the main notations. Tables II–IV and Figs. 4–10 are provided in the supplementary material of this article, due to page limitations.

II. Problem Description

A. Definitions

Definition 1: An *anomaly* in streaming data is defined as an observation that deviates from the previous normal behaviors [1].

Definition 2: Spatial anomalies are defined as the obvious anomalies (usually single anomalies), which may be too small or too large, exceeding the normal range of streaming data [1].

Definition 3: Contextual anomalies are defined as the subtle anomalies, which may not be regarded as anomalies independently, but occur only in a specific temporal context [1], [2].

Definition 4: Concept drifts [2], [41] are defined as the subtle anomalies, where the statistical characteristics of data change over time.

B. Problem Description

Given streaming data, $x_1, x_2, \ldots, x_t$, where $x_t \in \mathbb{R}$ denotes the observation at time t, we wish to determine if the current observation x_t is anomalous in an online and unsupervised manner. In addition to the spatial anomalies, we also wish to detect subtle contextual anomalies and concept drifts as early as possible.

III. Our Algorithm

Our algorithm contains three parts: 1) the main sliding window and the subwindows; 2) the PDDs; and 3) the distance-based anomaly detection rule. We briefly introduce each part here and later explain them in the following sections.

1) The first part, corresponding to Fig. 1(a), is the main sliding window W and the subwindows. At each time, current observation x_t is added to W and the oldest observation is removed from W. Refer to Section III-A.
2) The second part, corresponding to Fig. 1(b), is the PDDs for the subwindows, using KDE to estimate the underlying distribution of each subwindow and converting the distribution to a PDD (a vector) using targets selection. Refer to Section III-B.
3) The third part, corresponding to Fig. 1(c), is the distance-based anomaly detection rule, calculating the distance between each pair of adjacent PDDs, and then creating a rule to determine whether an anomaly is detected. Refer to Section III-C.

A. Main Sliding Window and the Subwindows

We use a main sliding window with a constant size over streaming data and segment this window into an array of nonoverlapping and equal-length subwindows.

Definition 5: The main sliding window W over streaming data is defined as a set of observations with size $N \in \mathbb{N}^+$

$$W := \{x_i\}_{i=t-N+1}^{t} \tag{1}$$

which includes current observation x_t and its previous data of size $N-1$. The term *sliding* means at each time, the current observation is added to W and the oldest observation is removed from W.

Definition 6: Define an array of nonoverlapping subwindows $\{w_k\}_{k=1}^{m}$, where

$$m = \left\lfloor \frac{N}{n} \right\rfloor$$

and

$$w_k := \{x_i\}_{i=t-kn+1}^{t-(k-1)n}. \tag{2}$$

These subwindows are segmented from W and each subwindow has the same size $n \in \mathbb{N}^+$.

The array of the m subwindows $w_1, w_2, \ldots, w_m$ are ordered in the direction from the current observation x_t to the oldest observation [refer to Fig. 1(a)], that is, w_1 is the latest subwindow including the current observation, which will be the focus of anomaly detection.

Remark 1: All data in W will be used to calculate the bandwidth in Section III-B. Integer N is *not required* to be divisible by integer n. The remaining observations $\{x_i\}_{i=t-N+1}^{t-mn}$ (the oldest) will be aborted in this step.

B. Probability Density-Based Descriptors

In this section, we propose an array of PDDs, with the kth PDD taking the kth subwindow w_k as its object.

Definition 7: Let $f_w(\cdot|k)$ be the probability density function (PDF) of w_k. We define the PDD of w_k as vector $f_{w|k} \in \mathbb{R}^p$ (with $p \in \mathbb{N}^+$ targets), whose elements are the samples on the PDF of w_k.

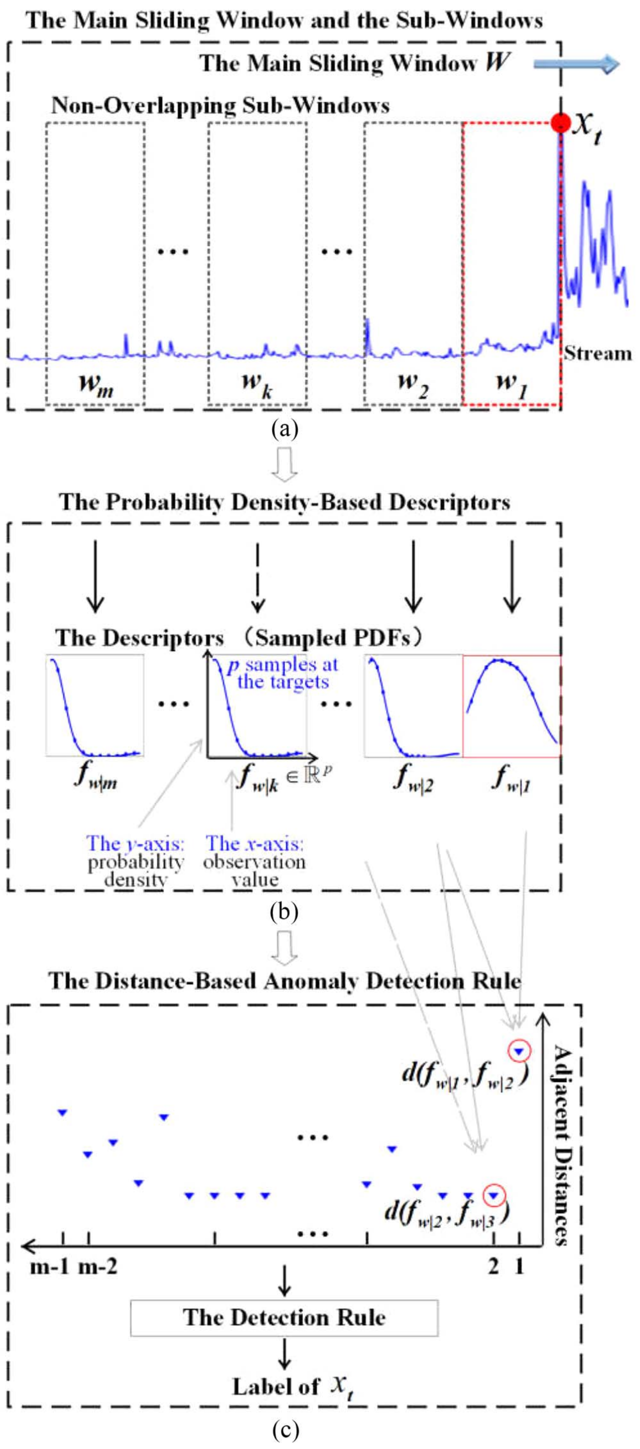

The Main Sliding Window and the Sub-Windows
The Main Sliding Window W
Non-Overlapping Sub-Windows
x_t
Stream
w_m
w_k
w_2
w_1
(a)
The Probability Density-Based Descriptors
The Descriptors (Sampled PDFs)
p samples at the targets
$f_{w|m}$
$f_{w|k} \in \mathbb{R}^p$
$f_{w|2}$
$f_{w|1}$
The y-axis: probability density
The x-axis: observation value
(b)
The Distance-Based Anomaly Detection Rule
$d(f_{w|1}, f_{w|2})$
$d(f_{w|2}, f_{w|3})$
Adjacent Distances
m-1 m-2
2 1
The Detection Rule
Label of x_t
(c)

Fig. 1. Our anomaly detection procedure. (a) Current observation x_t is fed to the main sliding window, which is segmented into an array of nonoverlapping subwindows. (b) KDE with targets selection is applied to obtain the corresponding PDD for each subwindow. (c) Distances (illustrated as blue triangles) between two adjacent PDDs are computed, and the label indicating whether x_t is anomalous is given.

Remark 2: In Fig. 1(b), taking the subfigure below the dotted arrow as an example, the blue curve represents the PDF of w_k and the x-coordinates of the blue dots represent the targets.

The estimation procedure of PDD $f_{w|k}$ is as follows.

1) *Sources:* w_k (2) is the PDF estimation set.
2) *Targets:* Define the target set $T := \{y_j \in \mathbb{R}\}_{j=1}^{p}$, where p is the number of the targets. Let $W_{\min}$ and $W_{\max}$ be the minimum and maximum values of W, respectively. Then, we select the targets by partitioning the range $[W_{\min}, W_{\max}]$ into p equally spaced intervals and taking the midpoints of each interval, that is

$$y_j = W_{\min} + \frac{1}{p}(W_{\max} - W_{\min})\left(j - \frac{1}{2}\right) \in \mathbb{R} \tag{3}$$

where $j = 1, 2, \ldots, p$. For the selection of p, refer to Section IV-E.

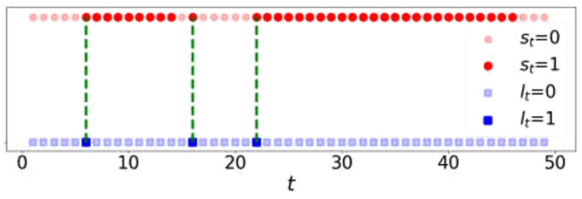


Fig. 2. Example for s_t and l_t.

3) *The PDD Calculation:* The PDD $f_{w|k}$ is a p-dimensional vector (Definition 7) defined as

$$f_{w|k} := \left[f_w(y_1|k), \ldots, f_w(y_p|k)\right] \in \mathbb{R}^p, k = 1, 2, \ldots, m \tag{4}$$

where

$$f_w(y_j|k) := \frac{1}{\sqrt{2\pi}nh} \sum_{i=t-kn+1}^{t-kn+n} \exp\left(-\frac{1}{2}\left(\frac{y_j - x_i}{2}\right)^2\right) \tag{5}$$

where $j = 1, 2, \ldots, p$. In (5), $h \in \mathbb{R}^+$ is a bandwidth (which will be determined in the next item), and we use a Gaussian kernel for KDE, as the shape of the kernel does not critically affect the statistical accuracy [27], [42]. To speed up the computation of KDE, we utilize the K-dimension tree data structure [39], which builds space-partitioning trees for source set w_k and target set.
4) *The Bandwidth:* Bandwidth h is used to model the degree of uncertainty about sources (observations) and control the smoothing behavior of KDE [27]. As a Gaussian kernel is used for KDE in (5), the corresponding optimal bandwidth h can be determined by the rule-of-thumb selector [27] as

$$h = \left(\frac{4}{3n}\right)^{\frac{1}{5}} \sigma \in \mathbb{R}^+ \tag{6}$$

where $\sigma \in \mathbb{R}^+$ is the standard deviation of current W.

C. Distance-Based Anomaly Detection Rule

From (4), we have m PDDs, then we wish to determine if current observation x_t is anomalous, that is, if the distribution of the latest subwindow w_1, represented by PDD $f_{w|1}$, quite deviates from the previous usual distributions.

Definition 8: Let $l_t \in \{0, 1\}$ be an indicator variable (label), value $l_t = 1$ indicating current observation x_t is an anomaly and value $l_t = 0$ indicating x_t is normal.

We employ the *distance* to evaluate the similarity between the underlying distributions of two adjacent subwindows w_k and w_{k+1} [refer to Fig. 1(c)].

Definition 9: The distance between two subwindows w_k and w_{k+1} is defined as the distance between their respective PDDs. Here, we employ the distance given by

$$d\left(f_{w|k}, f_{w|k+1}\right) := \sum_{j=1}^{p} \left|f_w\left(y_j|k\right) - f_w\left(y_j|k+1\right)\right| \geq 0. \tag{7}$$

Define $D := \{d(f_{w|k}, f_{w|k+1})\}_{k=2}^{m-1}$.

The greater the distance is, the more distinct the corresponding two subwindows are.

Definition 10: Define the absolute difference set for D as

$$C := \left\{\left|d\left(f_{w|i+1}, f_{w|i+2}\right) - d\left(f_{w|i}, f_{w|i+1}\right)\right|\right\}_{i=2}^{m-2} \quad (8)$$

where $m \geq 4$. Define the expanded maximum distance as

$$\tilde{D}_{\max} := \max D + \min C \in \mathbb{R}^{+}. \quad (9)$$

Let D_{avg} be the average distance of D, given by

$$D_{\text{avg}} := \frac{1}{m-2} \sum_{k=2}^{m-1} d\left(f_{w|k}, f_{w|k+1}\right) \in \mathbb{R}^{+}. \quad (10)$$

The anomaly detection rule is designed as follows:

$$l_t = \begin{cases} 1, & \text{if } s_{t-1} = 0 \text{ and } s_t = 0 \\ 0, & \text{otherwise} \end{cases} \quad (11)$$

where $s_t \in \{0, 1\}$ is an indicator variable and updated as

$$s_t = \begin{cases} 1, & \text{if } d\left(f_{w|1}, f_{w|2}\right) > \tilde{D}_{\max} \text{ and } d\left(f_{w|2}, f_{w|3}\right) \leq D_{\text{avg}} \\ 0, & \text{otherwise.} \end{cases} \quad (12)$$

Initially, $s_{N-1} = 0$.

Condition $s_{t-1} = 0$ in (11) is to avoid frequent continuous alarms, as shown in Fig. 2. If the latest distance $d(f_{w|1}, f_{w|2}) > \tilde{D}_{\max}$, it means that the distribution of subwindow w_1 is quite different from that of subwindow w_2. However, it is still likely that the distribution of w_2 is anomalous. Hence, we add an additional condition in (12), that is, whether the previous distance $d(f_{w|2}, f_{w|3}) \leq D_{\text{avg}}$. This additional condition can greatly reduce the detection possibility that the distribution of w_2 is considered abnormal.

Remark 3: This anomaly detection rule is dependent on the statistical characteristics of the data itself.

D. Our Algorithm

Algorithm 1 presents our *anomaly detection on the current observation*, and Algorithm 2 is the main algorithm that presents our overall *anomaly detection over streaming data*, using Algorithm 1 as the subfunction.

The computational complexity of Algorithm 1 is provided in Section IV-D. The effects of the system parameters of Algorithm 2 are provided in Section IV-E.

Algorithm 1 Anomaly Detection on Current Observation

```
Inputs: the main sliding window $w$,
        the indicator variable $s_{t-1}$,
        the parameters $N$, $n$, $p$
Outputs: the label $l_t$ for the current observation $x_t$,
         the updated indicator $s_t$
 1: Initialize: $D \leftarrow [\cdot]$
 2: $m \leftarrow \lfloor \frac{N}{n} \rfloor$
 3: get the m sub-windows $\{w_k\}_{k=1}^{m}$ from $W$ using (2)
 4: // calculate the PDDs:
 5: get the range $[W_{\min}, W_{\max}]$ of $W$
 6: get the target set $T = \{y_j \in \mathbb{R}\}_{j=1}^{p}$ using (3)
 7: update $\sigma$ and thus bandwidth $h$ using (6)
 8: for $k \leftarrow 1$ to $m$ do
 9:     for $j \leftarrow 1$ to $p$ do
10:         calculate $f_w(y|k)$ using (5)
11:     end for
12:     calculate PDD $f_{w|k}$ using (4)
13: end for
14: // calculate distances:
15: for $k \leftarrow 2$ to $m-1$ do
16:     calculate $d(f_{w|k}, f_{w|k+1})$ using (7)
17:     add $d(f_{w|k}, f_{w|k+1})$ to $D$
18: end for
19: calculate $\tilde{D}_{\max}$ using (9)
20: calculate $D_{\text{avg}}$ using (10)
21: // determine if $x_t$ is an anomaly using (11):
22: get the updated $s_t$ using (12)
23: if $s_{t-1} = 0$ and $s_t = 1$ then
24:     $l_t \leftarrow 1$ // $x_t$ is anomalous
25: else
26:     $l_t \leftarrow 0$ // $x_t$ is normal
27: end if
28: return $l_t$, $s_t$
```

Algorithm 2 Anomaly Detection Over Streaming Data

```
Inputs: the streaming data $x_1, x_2, \ldots, x_N, x_{N+1}, \ldots$,
        the parameters $N$, $n$, $p$
Outputs: an array $\ell$ of anomaly labels for streaming data
 1: Initialize: $t \leftarrow N$, $s_{t-1} \leftarrow 0$, $W \leftarrow \{x_1, x_2, \ldots, x_N\}$, $\ell \leftarrow$
    $[0, 0, \ldots, 0]$ of size $N-1$
 2: // the anomaly detection process over streaming data:
 3: repeat
 4:     // detecting:
 5:     $(l_t, s_t) \leftarrow$ Algorithm 1 (inputs: $W$, $s_{t-1}$, $N$, $n$, $p$)
 6:     append $l_t$ to the label array: $\ell \leftarrow [\ell, l_t]$
 7:     // update the main sliding window W:
 8:     remove $x_{t-N+1}$ from $W$
 9:     $t \leftarrow t+1$
10:     add $x_t$ to $W$
11: until streaming data is terminated
12: return the label array $\ell$
13: // The value of the i-th entry in the label array $\ell$ indicates whether
    $x_i$ in the streaming data is anomalous. (The first $N-1$ observations
    are assumed normal.)
```

IV. Experimental Results and Performances

A. True Positives, False Positives, and the NAB Score

The scoring mechanism of the NAB [3] is as follows. Let A be the data file under consideration, with A_{TP}, A_{FP}, A_{TN}, $A_{\text{FN}} \in [0, 1]$, the corresponding weights for true positives (TPs), false positives (FPs), true negatives (TNs), and false negatives (FNs), respectively. (Consistent with [2], we take $A_{\text{TP}} = 1$, $A_{\text{TN}} = 1$, $A_{\text{FP}} = 0.11$, and $A_{\text{FN}} = 1$.) Let Y_{TP} and Y_{FP} be the set of data instances of TPs and FPs, respectively. The number of FNs is represented by f. The raw score S^A on data file A is

$$S^A = A_{\text{TP}} \sum_{y \in Y_{\text{TP}}} \beta(y) + A_{\text{FP}} \sum_{y \in Y_{\text{FP}}} \beta(y) - A_{\text{FN}} f$$

where

$$\beta(y) = \frac{2}{1 + e^{5y}} - 1$$

which is designed for rewarding early detection [3], and

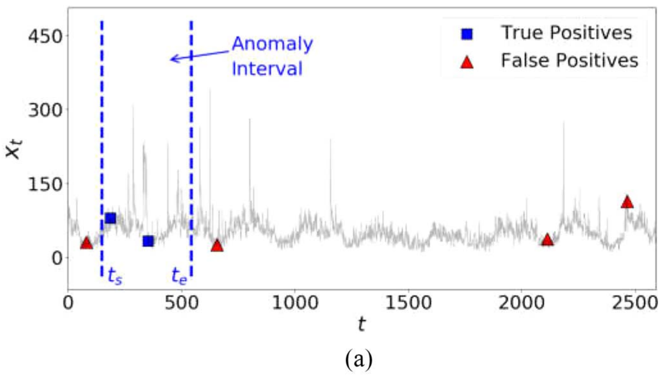
$$y = \frac{t_e - t + 1}{t_e - t_s + 1} \geq -1$$

which is the relative position to a given *anomaly interval* $[t_s, t_e]$ (the anomaly interval is a range of data points centered around a ground-truth anomaly label).

When multiple TPs occur within an anomaly interval, only the *earliest* (most valuable) one is calculated for the score contribution, while the others are ignored [3].

Definition 11: The *scored-TPs* are defined as the TPs that are only used for the NAB scoring.

A scoring example is illustrated in Fig. 3.



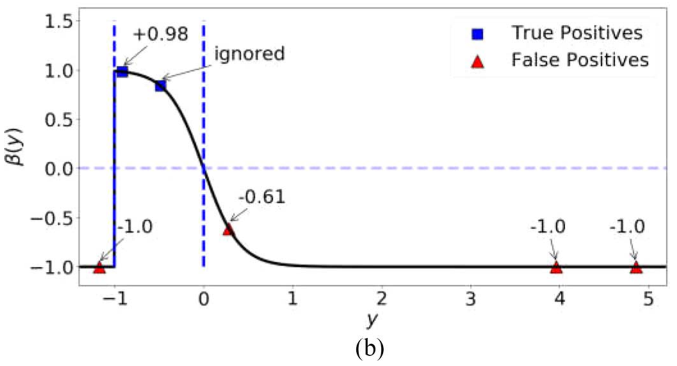


Fig. 3. Scoring example for a sample anomaly interval $[t_s, t_e]$. (a) TPs and FPs. (b) Corresponding $\beta(y)$ values. When multiple TPs occur within such an interval, only the earliest (most valuable) one is calculated for the score contribution. Each FP has a negative score, which is dependent on how far it is from the corresponding previous interval. The NAB score for this interval is calculated as: $0.98A_{\text{TP}} - 0.61A_{\text{FP}} - 1.0A_{\text{FP}} - 1.0A_{\text{FP}}$.

The final NAB score is

$$S^A_{NAB} = 100 \cdot \frac{S_A}{rA_{\text{TP}}} \in [-\infty, 100]$$

where r is the number of anomaly intervals in the data file A.

Remark 4: There is a tradeoff between early anomaly detection and FPs when scoring [2].

B. Evaluation Results and Comparisons

Here, we list the performances obtained by the implementation[1] of the HTM-based algorithm [2] and the implementation[2] of our algorithm, respectively, across data from different kinds of sources provided by the benchmark [3], including realTweets (Table II in the supplementary material), realKnownCause (Table III in the supplementary material), and realAWSCloudwatch and artificialWith-Anomaly (Table IV in the supplementary material), which contain 31 data files and 284 842 observations.

In this article, we take $p = 16$. All of the experiments are run in the same environment (CPU: Intel Core i5-7400 CPU 3.00GHz, RAM: 16.0GB, Windows 10).

From Tables II–IV, in the supplementary material, our algorithm outperforms the HTM-based algorithm [2] in terms of both scored-TPs and score on most of the benchmark datasets.

Remark 5: Based on the NAB [3], the following real-time anomaly detection algorithms are evaluated in [2]: the HTM-based algorithm [2], EXPoSE [22], Twitter's anomaly detection [23], Etsy's Skyline [24], multinomial relative entropy [25], and Bayesian online changepoint detection [26]. The results in [2] show that the HTM-based algorithm [2] performs best based on the NAB.

Remark 6: Comparison results with other algorithms, including the density-based algorithms [15]–[17], multinomial relative entropy [25], and Bayesian online changepoint detection [26], are also available online,[3] which demonstrate that our algorithm can perform better in many cases.

C. Examples of Detected Anomalies

As shown in Fig. 4(a), in the supplementary material, our algorithm can effectively detect spatial anomalies. In addition, Fig. 4(b) and (c), in the supplementary material, demonstrates the ability of our algorithm to detect subtle contextual anomalies and concept drifts effectively and early.

D. Computational Complexity

The computational complexity of KDE using (5) directly is $O(pn)$ [43], where n is the number of sources and p is the size of target set T. This $O(pn)$-complexity is impractical for online applications.

In this article, we implement KDE by using the K-dimension tree data structure [39] for efficient queries,[4] reducing the complexity down to $O(p\log(n))$ or better [44]. As in our algorithm, m subwindows need to be estimated for each observation, hence, the computational complexity of our algorithm is $O(mp\log(n))$.

E. Effects of the System Parameters

1) Effects of the Parameters (N, n): Figs. 5–8 in the supplementary material demonstrate the relation between parameters (N, n) and the performance of our algorithm. We select four typical data files from the NAB [3] with different numbers of observations from different data sources. We can find the following.

1) From Figs. 5(a), 6(a), 7(a), and 8(a), in the supplementary material, both parameters N and n affect the score performance of our algorithm. For various streaming data with different data characteristics, the corresponding N (for better score performance) has different appropriate ranges.
2) From Figs. 5(b), 6(b), 7(b), and 8(b), in the supplementary material, as N increases, both scored-TPs and FPs decrease. Hence, it is a tradeoff between scored-TPs and FPs when choosing the value of N.
3) From Figs. 5(c), 6(c), 7(c), and 8(c), in the supplementary material, as n increases, scored-TPs decrease and FPs tend to increase, that is, n can take a smaller value for *effectiveness* of our algorithm. In addition, n needs to take a relatively larger value for *efficiency* of our algorithm, as shown in Fig. 9, in the supplementary material. Hence, it is a tradeoff between the effectiveness and efficiency when choosing the value of n.

2) Effect of the Number p of Targets: Fig. 10, in the supplementary material, demonstrates the relation between the number p of targets and the performance of our algorithm. From Fig. 10 in the supplementary material, we can find the following.

1) The number p of targets has less of an effect on the score performance of our algorithm as p increases to a relative larger value, as shown in Fig. 10(a) in the supplementary material.
2) The number p of targets has less effect on the scored-TPs and FPs of our algorithm as p increases to a relative larger value, as shown in Fig. 10(b) in the supplementary material.

[1]The implementation of the HTM-based algorithm [2] is available at https://github.com/numenta/NAB.

[2]The implementation of our algorithm is available at https://github.com/lyzhang0614/windowKDEdetector.

[3]The comparison results with other algorithms are available at https://github.com/lyzhang0614/windowKDEdetector/tree/master/comparisons.

[4]We implement KDE by importing the *sklearn.neighbors.KernelDensity* estimator in Python, and the corresponding function is *KernelDensity* (·).

3) Usually, we select $p < n$ to make dimension reduction and, thus, to speed up the computation of our algorithm.

V. Conclusion

In this article, we proposed an online and unsupervised anomaly detection algorithm for streaming data, which mainly consists of three parts: 1) the main sliding window and the subwindows; 2) the PDDs; and 3) the distance-based anomaly detection rule. Our algorithm features a high detection rate as it cannot only effectively detect the spatial anomalies but also detect the subtle contextual anomalies and concept drifts as early as possible. Our algorithm can perform better in many cases than the HTM-based algorithm [2] (which outperforms a wide range of other algorithms) based on the NAB [3].

The main drawback of our algorithm is that parameters *N and n*, especially *N*, of our algorithm need to be tuned according to different streaming data. In practical applications, we can determine the appropriate sizes using Section IV-E1 as a guide and then perform online anomaly detection.

There are several future considerations. The current NAB benchmark is limited to streaming data containing a single metric (one dimension). An algorithm using multivariate data would be a valuable addition, and KDE can be easily extended to multivariate cases. Moreover, the relationship between the anomalous observations and anomalous events is another problem, which will be helpful in the positioning of the anomalous events in operation and maintenance.

References

[1] V. Chandola, A. Banerjee, and V. Kumar, "Anomaly detection: A survey," *ACM Comput. Surveys*, vol. 41, no. 3, pp. 1–58, 2009.

[2] S. Ahmad, A. Lavin, S. Purdy, and Z. Agha, "Unsupervised real-time anomaly detection for streaming data," *Neurocomputing*, vol. 262, pp. 134–147, Nov. 2017.

[3] A. Lavin and S. Ahmad, "Evaluating real-time anomaly detection algorithms—The Numenta anomaly benchmark," in *Proc. IEEE Int. Conf. Mach. Learn. Appl.*, Miami, FL, USA, 2015, pp. 38–44.

[4] D. Sun, M. Fu, L. Zhu, G. Li, and Q. Lu, "Non-intrusive anomaly detection with streaming performance metrics and logs for DevOps in public clouds: A case study in AWS," *IEEE Trans. Emerg. Topics Comput.*, vol. 4, no. 2, pp. 278–289, Apr./Jun. 2016.

[5] D. J. Hill and B. S. Minsker, "Anomaly detection in streaming environmental sensor data: A data-driven modeling approach," *Environ. Modell. Softw.*, vol. 25, no. 9, pp. 1014–1022, 2010.

[6] V. L. Cao, M. Nicolau, and J. McDermott, "Learning neural representations for network anomaly detection," *IEEE Trans. Cybern.*, vol. 49, no. 8, pp. 3074–3087, Aug. 2019.

[7] X. Miao, Y. Liu, H. Zhao, and C. Li, "Distributed online one-class support vector machine for anomaly detection over networks," *IEEE Trans. Cybern.*, vol. 49, no. 4, pp. 1475–1488, Apr. 2019.

[8] Y. Yuan, J. Fang, and Q. Wang, "Online anomaly detection in crowd scenes via structure analysis," *IEEE Trans. Cybern.*, vol. 45, no. 3, pp. 548–561, Mar. 2015.

[9] A. Mahimkar *et al.*, "Rapid detection of maintenance induced changes in service performance," in *Proc. Conf. Emerg. Netw. Exp. Technol.*, Tokyo, Japan, 2011, pp. 1–12.

[10] F. Rasheed and R. Alhajj, "A framework for periodic outlier pattern detection in time-series sequences," *IEEE Trans. Cybern.*, vol. 44, no. 5, pp. 569–582, May 2014.

[11] D. Liu *et al.*, "Opprentice: Towards practical and automatic anomaly detection through machine learning," in *Proc. Int. Meas. Conf.*, Tokyo, Japan, 2015, pp. 211–224.

[12] V. Timčenko and S. Gajin, "Ensemble classifiers for supervised anomaly based network intrusion detection," in *Proc. IEEE Int. Conf. Intell. Comput. Commun. Process.*, 2017, pp. 13–19.

[13] J. Dromard, G. Roudière, and P. Owezarski, "Online and scalable unsupervised network anomaly detection method," *IEEE Trans. Netw. Service Manag.*, vol. 14, no. 1, pp. 34–47, Mar. 2017.

[14] A. Bernieri, G. Betta, and C. Liguori, "On-line fault detection and diagnosis obtained by implementing neural algorithms on a digital signal processor," *IEEE Trans. Instrum. Meas.*, vol. 45, no. 5, pp. 894–899, Oct. 1996.

[15] B. S. J. Costa, P. P. Angelov, and L. A. Guedes, "Fully unsupervised fault detection and identification based on recursive density estimation and self-evolving cloud-based classifier," *Neurocomputing*, vol. 150, pp. 289–303, Feb. 2015.

[16] P. Angelov, "Anomaly detection based on eccentricity analysis," in *Proc. IEEE Symp. Evol. Auton. Learn. Syst.*, Orlando, FL, USA, 2014, pp. 1–8.

[17] B. S. J. Costa, C. G. Bezerra, L. A. Guedes, and P. P. Angelov, "Online fault detection based on typicality and eccentricity data analytics," in *Proc. Int. Joint Conf. Neural Netw.*, 2015, pp. 1–6.

[18] T. Ahmed, "Online anomaly detection using KDE," in *Proc. IEEE Glob. Telecommun. Conf.*, Honolulu, HI, USA, 2009, pp. 1009–1016.

[19] K. Wu, K. Zhang, W. Fan, A. Edwards, and P. S. Yu, "RS-forest: A rapid density estimator for streaming anomaly detection," in *Proc. IEEE Int. Conf. Data Min.*, Shenzhen, China, 2014, pp. 600–609.

[20] R. Laxhammar and G. Falkman, "Online learning and sequential anomaly detection in trajectories," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 36, no. 6, pp. 1158–1173, Jun. 2014.

[21] D. Zambon, C. Alippi, and L. Livi, "Concept drift and anomaly detection in graph streams," *IEEE Trans. Neural Netw. Learn. Syst.*, vol. 29, no. 11, pp. 5592–5605, Nov. 2018.

[22] M. Schneider, W. Ertel, and F. T. Ramos, "Expected similarity estimation for large-scale batch and streaming anomaly detection," *Mach. Learn.*, vol. 105, no. 3, pp. 305–333, 2016.

[23] A. Kejariwal, *Introducing Practical and Robust Anomaly Detection in a Time Series*, Twitter Blog, San Francisco, CA, USA, 2015.

[24] A. Stanway. (2013). *Esty Skyline*. [Online]. Available: https://github.com/etsy/skyline

[25] C. Wang, K. Viswanathan, L. Choudur, V. Talwar, W. Satterfield, and K. Schwan, "Statistical techniques for online anomaly detection in data centers," in *Proc. Int. Symp. Integr. Netw. Manag.*, Dublin, Ireland, 2011, pp. 385–392.

[26] R. P. Adams and D. J. MacKay, "Bayesian online changepoint detection," *arXiv*, 2007.

[27] B. W. Silverman, *Density Estimation for Statistics and Data Analysis*. London, U.K.: Chapman & Hall, 1986.

[28] Y. Cui, S. Ahmad, and J. Hawkins, "Continuous online sequence learning with an unsupervised neural network model," *Neural Comput.*, vol. 28, no. 11, pp. 2474–2504, 2016.

[29] G. S. Manku and R. Motwani, "Approximate frequency counts over data streams," in *Proc. Int. Conf. Very Large Data Bases*, 2002, pp. 346–357.

[30] S. Lin, J. Qiao, and Y. Wang, "Frequent episode mining within the latest time windows over event streams," *Appl. Intell.*, vol. 40, no. 1, pp. 13–28, 2014.

[31] Y. Guo, G. Wang, F. Hou, and Q. Mei, "Recent frequent item mining algorithm in a data stream based on flexible counter windows," *J. Softw.*, vol. 9, no. 1, pp. 258–263, 2014.

[32] B. Chen, C. Y. Duan, and X.-E. Gao, "A frequent pattern parallel mining algorithm based on distributed sliding window," *Comput. Syst. Sci. Eng.*, vol. 31, no. 2, pp. 101–107, 2016.

[33] W. Pedrycz, W. Lu, X. Liu, W. Wang, and L. Wang, "Human-centric analysis and interpretation of time series: A perspective of granular computing," *Soft Comput.*, vol. 18, no. 12, pp. 2397–2411, 2014.

[34] D. Leite, P. Costa, and F. Gomide, "Interval approach for evolving granular system modeling," in *Learning Non-Stationary Environments*. New York, NY, USA: Springer, 2012, pp. 271–300.

[35] A. Gacek, "Signal processing and time series description: A perspective of computational intelligence and granular computing," *Appl. Soft Comput.*, vol. 27, pp. 590–601, Feb. 2015.

[36] H. Guo, W. Pedrycz, and X. Liu, "Hidden Markov models based approaches to long-term prediction for granular time series," *IEEE Trans. Fuzzy Syst.*, vol. 26, no. 5, pp. 2807–2817, Oct. 2018.

[37] M. Rosenblatt, "Remarks on some nonparametric estimates of a density function," *Ann. Math. Stat.*, vol. 27, no. 3, pp. 832–837, 1956.

[38] E. Parzen, "On estimation of a probability density function and mode," *Ann. Math. Stat.*, vol. 33, no. 3, pp. 1065–1076, 1962.

[39] J. L. Bentley, "Multidimensional binary search trees used for associative searching," *Commun. ACM*, vol. 18, no. 9, pp. 509–517, 1975.

[40] J. W. S. Cassels, *An Introduction to Diophantine Approximation*. Cambridge, U.K.: Cambridge Univ. Press, 1957.

[41] C. O'Reilly, A. Gluhak, and M. A. Imran, "Adaptive anomaly detection with kernel eigenspace splitting and merging," *IEEE Trans. Knowl. Data Eng.*, vol. 27, no. 1, pp. 3–16, Jan. 2015.

[42] D. Kazakos, "Choice of kernel function for density estimation," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. PAMI-2, no. 3, pp. 255–258, May 1980.

[43] C. B. Akgül, B. Sankur, Y. Yemez, and F. Schmitt, "3D model retrieval using probability density-based shape descriptors," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 31, no. 6, pp. 1117–1133, Jun. 2009.

[44] F. Pedregosa *et al.*, "Scikit-learn: Machine learning in Python," *J. Mach. Learn. Res.*, vol. 12, no. 10, pp. 2825–2830, 2011.