Online Anomaly Detection with Bandwidth Optimized Hierarchical Kernel Density Estimators

Mine Kerpicci, Huseyin Ozkan, and Suleyman S. Kozat

***Abstract*—We propose a novel unsupervised anomaly detection algorithm that can work for sequential data from any complex distribution in a truly online framework with mathematically proven strong performance guarantees. First, a partitioning tree is constructed to generate a doubly exponentially large hierarchical class of observation space partitions, and every partition-region trains an online kernel density estimator (KDE) with its own unique dynamical bandwidth. At each time, the proposed algorithm optimally combines the class estimators to sequentially produce the final density estimation. We mathematically prove that the proposed algorithm learns the optimal partition with kernel bandwidths that are optimized in both region-specific and time varying manner. The estimated density is then compared with a data adaptive threshold to detect anomalies. Overall, the computational complexity is only linear in both the tree depth and data length. In our experiments, we observe significant improvements in anomaly detection accuracy compared to the state-of-the-art techniques.**

***Index Terms*—Online, anomaly detection, kernel density estimation, bandwidth selection, regret analysis.**

I. Introduction

Anomaly detection, as an unsupervised one-class machine learning problem, finds use in various applications ranging from cyber security [1], surveillance [2], failure detection [3], novelty detection [4] and data imputation [5]. We study this problem for sequential data in a truly online setting; and propose a novel algorithm that observes $\boldsymbol{x}_t \in \mathbb{R}^d$ at each time t, and decides whether $\boldsymbol{x}_t$ is anomalous based on the past observations $\{\boldsymbol{x}_1, \boldsymbol{x}_2, \dots \boldsymbol{x}_{t-1}\}$. In our framework, each instance $\boldsymbol{x}_t$ is processed only once without being stored and no label information is required. Therefore, the proposed algorithm is online and unsupervised, and appropriate for real time anomaly detection applications.

One of the main difficulties in anomaly detection problems is typically the extreme rarity of anomalies and their unpredictable nature [6], which differentiates the anomaly detection from the binary classification. In the binary classification problem, one has observations from both classes. However, in anomaly detection, there are no or extreme rare observations from the anomaly class. This essentially poses a one-class classification problem where it is reasonable to assume uniformity for anomalies [7]. Then, the Neyman-Pearson (NP) test [8] corresponds to comparing the density $f(\boldsymbol{x}_t)$ with an appropriate threshold [9], [10] that maximizes the detection power at a desired false alarm rate. This in turn amounts to density estimation and learning the desired threshold. Therefore, a common approach in the literature [6] consists of two steps, which are assigning a density score to the observed data instance, and then, comparing this score with a threshold [6]. Scores below the threshold indicate anomaly.

Our technique is also two fold. We first estimate the probability density of the data in a nonparametric manner without any model assumption, and then find the anomalies by thresholding the estimated density for each instance.

Our density estimator is regret-optimal [11], where the loss is the accumulated negative log-likelihood, against a large class of estimators. To be more precise, our density estimator asymptotically performs at least as well as the best class estimator in terms of the data likelihood. To obtain an estimator in this class, we use a kernel density estimator (KDE) [12] with a different bandwidth parameter in each region of a given partition of the observation space. Hence, the class of estimators are restricted to KDEs defined on the tree where varying the bandwidth parameter in each region and also varying the partition itself yield the aforementioned class. The proposed algorithm learns the optimal partitioning (that can be defined over the introduced tree) of the observation space and also learns the optimal region-specific bandwidth parameters for the optimal partition even in a time-varying manner. Namely, the optimal bandwidth of the KDE that we use for each region in the optimal partition might well be different as well as nonstationary (the distribution that the data instances are drawn/coming from might be changing in time, i.e., might be time-varying).

The introduced density estimation for anomaly detection is essentially a mixture of experts [13] based approach. These experts are "piecewise" KDEs, "pieces" of which are generated by a hierarchical observation space partitioning via a depth-D binary tree. Pieces correspond to the partition regions; and for each region, the optimal time varying KDE bandwidth from a finite set with cardinality k is separately learned. Our algorithm produces its density estimation by combining these expert estimations. The proposed combination is based on a time-varying weighting that is used to mitigate overfitting/overtraining issues, and to asymptotically pick the optimal partition. The purpose in hierarchically organizing the KDEs is to achieve computational efficiency while still maintaining

This work was supported by the Scientific and Technological Research Council of Turkey under Contract 118E268 as well as Contract 117E153. (*Corresponding author: Mine Kerpicci*)

M. Kerpicci is with the School of Electrical and Computer Engineering, Georgia Institute of Technology, Atlanta, GA 30332, USA (e-mail: mkerpicci3@gatech.edu).

H. Ozkan is with the Faculty of Engineering and Natural Sciences, Sabancı University, Orhanli-Tuzla, 34956, Istanbul, Turkey (e-mail: hozkan@sabanciuniv.edu).

S. S. Kozat is with the Department of Electrical and Electronics Engineering, Bilkent University, Ankara 06800, Turkey (e-mail: kozat@ee.bilkent.edu.tr).

an exponentially large competition class of estimators. Specifically, the introduced tree based hierarchical structure provides us a class of size approximately $(1.5)^{2^D}$ whereas the overall processing complexity in our framework scales linearly only with $O(TDk)$ where T is the number of processed instances.

Our initial findings belonging to the use of KDE with only a single partitioning along with the bandwidth selection have been presented in a national conference [14]. On the other hand, in this paper, we complete our study comprehensively and provide our final greatly detailed experimental results. In particular, in this paper, we generalize to a hierarchically structured competing class of doubly exponentially many partitionings (i.e., density estimators) with our proposed bandwidth selection method while additionally achieving strong mathematical performance guarantees, which therefore yields significantly higher anomaly detection performance as presented in our extensive experiments.

The result is a novel computationally efficient combination of hierarchically organized piecewise online KDEs with bandwidths that are optimally varying in time and space. We mathematically prove and guarantee the convergence of the density estimation performance of our algorithm to the best piecewise constructed class estimator in terms of the data likelihood. Finally, we decide whether the observed data instance is anomalous or not by comparing the estimated probability with a threshold. In our approach, an optimal NP test can be achieved with an appropriate threshold choice for a constant false alarm rate. Alternatively, we provide an algorithm to adaptively learn the optimal threshold in terms of the detection accuracy when the true labels are available. Also, we demonstrate significant performance gains by our algorithm in comparison to the state-of-the-art methods through extensive set of experiments.

A. Related Work

Kernel based techniques are extensively used in the machine learning literature and in particular for density estimation [15] due to their nonparametric high modeling power. Nevertheless, kernel bandwidth selection is one of the most critical issues in these approaches [16] since the bandwidth choice significantly affects the performance of the density estimator. There are two main approaches to handle this issue, which are based on cross validation and plug-in [17]. Cross validation based approaches mostly depend on leave-one-out scheme to find the bandwidth [17]. Plug-in based methods are based on minimization of asymptotic mean integrated squared error (AMISE) [17], where pilot bandwidths are introduced for an iterative solution. Another method is proposed in [18], which pointwise varies the kernel bandwidth based on the minimization of error such as AMISE. Although these techniques and others such as [19], [20] are impressive in their own batch processing setting, they suffer from one or both of the following two issues: i) they use one global time-invariant bandwidth for the complete observation space and time span, and ii) their computational complexity is prohibitively large, preventing their use in our sequential setting. There are also online KDE methods, which use compression techniques to adapt the kernel method to an online framework, with time-varying bandwidth optimization as in [21], [22], [23]. These methods aim to reduce the number of stored data instances to lower the complexity. However, this compression approach requires iterative steps to update the compressed model and bandwidth at each time, which is prohibitively costly in an online framework. An online adapted version of AMISE-optimal bandwidth approach is proposed in [21], which requires iterative plug-in process for each observed data instance. Compressed model of samples is used in [22] to find the AMISE-optimal bandwidth at each time after the model is updated with the observed data instance. However, they are not able to learn the local variations of data due to their bandwidth optimization approaches. Our method, however, rigorously address all of these issues by the introduced highly adaptive hierarchical KDEs and their locally optimized bandwidths with low computational complexity.

There exists numerous methods [24], [25], [9], [26] that solve the anomaly detection problem in the nonsequential batch setting. For example, [25], [9] compute pair-wise cross k^{th} distances between the observed instance and training samples for detecting an anomaly. Another method is proposed in [26], which is less sensitive to the neighborhood parameter k compared to the conventional distance based techniques. Since these methods are all batch processors, repeated use of all of the available data are required that leads to a too large memory and computational complexity. Among the Neyman-Pearson based anomaly detection techniques [10], [7], [27], minimum volume set approach is used in [10] for detecting anomalies. Also, local nearest neighbor distances are used for the anomalies in video observations in [7]. These are again both batch techniques for which an online extension is proposed in [27] that can only work when the data is Markovian. In contrast, our technique is truly online and model free, which is appropriate for real time processing of challenging data sources.

Anomaly detection problem is solved in the online setting in [28], [29], [30], [31], [32]. For example, [29] proposes a k-nearest neighbor technique, with required training steps. Online training is introduced in [32] to adapt the batch algorithms such as support vector machine (SVM) to real time applications. Since these methods [29], [32] still require quadratic optimization, the computational running time significantly increases with the data length. Anomaly detection is considered as a classification problem in [30] and it is solved via a cost-sensitive approach, which requires true label information, whereas our technique is completely unsupervised. [28], [31] estimate the probability density of the observed data, and compare it with a threshold. For this, [28] assumes that the data distribution is from an exponential family. An information theoretic approach is proposed in [31] to increase the detection performance when the number of training data samples is small. These methods [28], [31] assume that the data distribution is Gaussian or a mixture of Gaussians, and solves the anomaly detection problem with a parametric approach. Unlike these methods, we do not assume any parametric distribution shape for the observed sequential data, hence our technique can model any unknown arbitrarily complex distribution in a data driven online manner with random Fourier features [33]. Kernel approximation with

random Fourier features [33] is used for binary classification problem in [34] where the proposed algorithm learns (in online manner) the linear discriminant in the approximated and linear kernel space with online gradient descent based on the true label information. Our method also uses random features for obtaining an online implementation but, on the other hand, solves the online anomaly detection problem by learning both the optimal partition in a hierarchical tree and the optimal bandwidths for all regions in this partition without requiring any true label information.

B. Contributions

Our contributions are as follows:

- As the first time in the literature, we introduce a piecewise nonparametric kernel density estimator with a highly adaptive hierarchical tree structure. This addresses the bandwidth selection problem in a locally adaptive manner. Hence, we achieve a great modeling power for data of any arbitrarily complex distribution while successfully mitigating the overfitting by a carefully designed weighting scheme over the introduced class of experts.
- We efficiently implement our kernel based anomaly detection algorithm in an online setting. To this end, we use a hierarchical approach to represent and manage a very large class of estimators $(1.5)^{2^D}$ to obtain the final estimation with complexity only $O(TDk)$ (T: data size, D: tree depth, k: cardinality of the bandwidth set). This is appropriate for real time processing while achieving a highly powerful estimation power.
- Our algorithm achieves the performance of the best estimator in class of estimators with a regret bound $O(\frac{2^D}{h})$ (D: tree depth, h: learning rate).
- We illustrate significant performance improvements achieved by our algorithm with respect to the state-of-the-art as well as the recently proposed methods through extensive set of experiments over synthetic and real data.

C. Organization

We first define our problem setting in Section II. In Section III, we introduce the hierarchical partitioning of the observation space. Section IV presents our combination of hierarchically organized online KDEs for our density estimation. Section V introduces our optimal time-varying thresholding for the final anomaly detection. In Section VI, we demonstrate our performance improvements on artificial and real datasets. We conclude with final remarks in Section VII.

II. Problem Description

We sequentially observe[1] $\boldsymbol{x}_t \in \mathbb{R}^d$ at each time t. Our aim is to decide whether the observed data $\boldsymbol{x}_t$ is anomalous or not based on the past observations, i.e., $\boldsymbol{x}_1, \boldsymbol{x}_2, \dots \boldsymbol{x}_{t-1}$. For this purpose, we introduce a two step approach for anomaly detection where we first sequentially estimate the probability density of $\boldsymbol{x}_t$ by using previously observed data sequence $\boldsymbol{x}^{t-1} = \{\boldsymbol{x}_i\}_{i=1}^{t-1}$ as $f_t(\boldsymbol{x}_t)$. Then, we compare the estimated probability $f_t(\boldsymbol{x}_t)$ with a threshold to decide whether $\boldsymbol{x}_t$ is anomalous or not. We declare a decision $\hat{d}_t$ at each time t as

$$\hat{d}_t = \begin{cases} +1, & f_t(\boldsymbol{x}_t) < \varsigma \text{ (anomalous)} \\ -1, & f_t(\boldsymbol{x}_t) \geq \varsigma \text{ (normal)} \end{cases}, \tag{1}$$

where ς is the threshold. We also provide an algorithm that updates the threshold ς_t in time to reach the best choice in terms of the detection accuracy, when the true label d_t is available. We emphasize that our framework is completely unsupervised, i.e., no knowledge of the true label is required; but if the true label is available for an instance, then it is incorporated.

We assume that the observation sequence $\{\boldsymbol{x}_t\}_{t\geq 1}$ is generated from an i.i.d. source where S is a bounded compact domain. In order to estimate the probability density of $\boldsymbol{x}_t$, we introduce a novel algorithm based on the context tree weighting method [35], [36] in which we construct a class of density estimators and asymptotically achieve -at least- the performance of the best class estimator.

Our goal is first to sequentially estimate the density of each instance $\boldsymbol{x}_t$ causally in time by combining the class estimators that we construct with hierarchical partitioning of a tree, and asymptotically achieve at least the performance of the best (in terms of the accumulated likelihoods) class estimator. This combination can be seen as a mixture of experts with each expert being a class estimator. Second, we optimally (in terms of the anomaly detection accuracy) threshold our estimated density for anomaly detection.

We use log-loss, i.e., negative log likelihood, as our pointwise loss function: $l(f_t(\boldsymbol{x}_t)) = -\log(f_t(\boldsymbol{x}_t))$ since this loss successfully encodes the data likelihood. Then, our goal of asymptotically performing at least as well as the best density estimator in the introduced class is equivalent to the minimization of the following regret $R(T)$ [11], resulting in a regret-optimal final estimator:

$$R(T) = \sum_{t=1}^{T} \big(-\log(f_t(\boldsymbol{x}_t)) \big) - \min_{\mathcal{P}_i} \left\{ \sum_{t=1}^{T} \Big(-\log(\hat{f}_t^{\mathcal{P}_i}(\boldsymbol{x}_t)) \Big) \right\}, \tag{2}$$

where T is the length of the sequence $\boldsymbol{x}^T = \{\boldsymbol{x}_i\}_{i=1}^T$ and the regret is the performance difference between our proposed estimator f_t and the best class estimator. Here, $\hat{f}^{\mathcal{P}_i}$ is the piecewise constructed KDE for the partition $\mathcal{P}_i$. In Section IV, we achieve the introduced goal by proving that this regret is bounded by a term that is convergent to 0 after normalization with T.

We compare the estimated density $f_t(\boldsymbol{x}_t)$ with a time-varying threshold ς to optimally decide (in terms of the detection accuracy) whether the observed data $\boldsymbol{x}_t$ is anomalous as in (1). For this purpose, by using the logistic function $r_t = \log(1 + \exp(-(\varsigma_t - f_t(\boldsymbol{x}_t))d_t)$, we update the threshold to maximize the detection power, when the true label $d_t \in$

[1] In this paper, all vectors are column vectors, and they are denoted by boldface lower case letters. For a vector $\boldsymbol{w}$, $\boldsymbol{w}'$ is its transpose. Also, for a scalar x, $|x|$ represents its absolute value and for a set $\mathbb{X}$, $|\mathbb{X}|$ represents the cardinality of the set. The time index is given as subscript, and a sequence of $\{\boldsymbol{x}_t\}_{t=1}^N$ is represented as $\boldsymbol{x}^N$. Also, $1_{\{\cdot\}}$ is the indicator function returning 1 if its argument condition holds, and returning 0 otherwise.

$\{-1, +1\}$ is available. If the true label is never available, then one can straightforwardly adjust the threshold to bound the false alarm rate in a similar manner. Recall that the proposed algorithm is unsupervised and can certainly operate without requiring the label information.

III. Tree Construction

In this section, we present the hierarchical partitioning tree to divide the observation space into disjoint regions and construct a class of density estimators.

In the literature, trees are used in various ways to improve the performance of the learning methods. The tree structure is used in a change detection problem in [37]. The use of randomized binary trees in the ensembles for outlier detection is pointed out in [38]. Moreover, binary splitting scheme for change detection problem is proposed in [39]. In our framework, we use a tree structure to construct a class of density estimators localized to various regions and combine their decisions to achieve a better performance.

A hierarchical partitioning tree of depth-D is constructed to partition the observation space S into various regions such that each tree node θ[2] corresponds to a specific region in S. We start splitting from the middle of the first dimension of the observation space. At each depth, we divide the observation space from the middle of the latter dimension. For an illustration, in Fig. 1a, we consider two dimensional bounded $\boldsymbol{x}_t \in \mathbb{R}^2$, i.e., $\boldsymbol{x}_t = [x_{t,1}, x_{t,2}]'$ and $|x_i| < A$, and use a depth-2 tree. There exist $2^{D+1} - 1$ nodes for a depth-D tree in general, and each node region is the union of the regions of its left and right children. For example, the children of the root node θ_o are defined as $\theta_{ol} = [-A, 0] \times [-A, A]$ and $\theta_{or} = [0, A] \times [-A, A]$ in Fig. 1a. The root node θ_o corresponds to the complete observation space such that $\theta_o = \theta_{ol} \cup \theta_{or} = S = [-A, A] \times [-A, A]$.

Remark: Note that there is no limit for the choice of depth D since partitioning is used to construct the disjoint regions, and increasing the number $2^{D+1}-1$ of nodes increases (with sufficient data) the performance of KDE. However, the depth D should be chosen carefully in practice with limited data. In this study, we have conducted separate preliminary experiments in each dataset we used, and observed that small D around $D = 3$ or $D = 4$ generally provides a highly good performance. Hence, we opted not to further optimize D and set it to $D = 3$ across our datasets. On the other hand, we emphasize that the required number of partition regions scale exponentially with the feature dimension d if one desires a specific level granularity in partitioning; and it scales linearly if one desires to split each dimension at least once. This poor scaling with the ambient dimension for a desired degree of fine partitioning appears to be a limitation of our presented technique. To address this limitation and handle high dimension, we apply PCA to the data and reduce its dimensionality to $d = 3$ prior to depth selection with a preliminary experiment. Hence, with PCA, our technique can still be applied successfully in high dimension especially when the covariance structure in the data allows good compression, which we strongly verify in our experiments. Note that PCA is only used for partitioning and constructing the tree, after which the instances traverse over tree with their full dimension. Namely, for an instance, we decide the tree nodes based on the reduced dimension, but the KDE at each node runs on the full dimension.

We combine n_i disjoint regions on the tree to form a partition $\mathcal{P}_i$ such that $\mathcal{P}_i = \{\theta_{i,1}, \theta_{i,2}, \ldots, \theta_{i,n_i}\}$ with $\theta_{i,k} \cap \theta_{i,j} = \varnothing$ if $k \neq j$ and $\bigcup_{j=1}^{n_i} \theta_{i,j} = S$. Examples of such partitions in the case of a depth-2 tree are given in Fig. 1b. In general, we construct $n_p \approx (1.5)^{2^D}$ [35] partitions for depth-D tree. Note that each of these partitions can be obtained by collecting the regions of the leaves after an appropriate pruning. One can obtain 5 different partitions for $D = 2$ as in Fig. 1b. We assign a KDE to each region $\theta_{i,j}$ in each partition $\mathcal{P}_i$ such that the KDE is trained with only the instances falling in the corresponding region. Note that each partition essentially collects n_i many region-specific KDEs yielding a "piecewise" constructed KDE. Consequently, the set of $(1.5)^{2^D}$ partitions from all possible prunings of the tree defines a class of piecewise constructed KDEs.

IV. The Proposed Novel Density Estimator

In this section, we propose a novel density estimator based on hierarchical partitioning of the observation space using a binary tree. Our approach is to sequentially train nonparametric kernel density estimators (KDE) [12] at each node of the introduced tree by instances falling in the corresponding region, while separately optimizing the bandwidth parameter [16] for each region. This process and thus the proposed algorithm is online and sequential with no storing.

A. Bandwidth Optimized Piecewise Online Kernel Density Estimator (KDE)

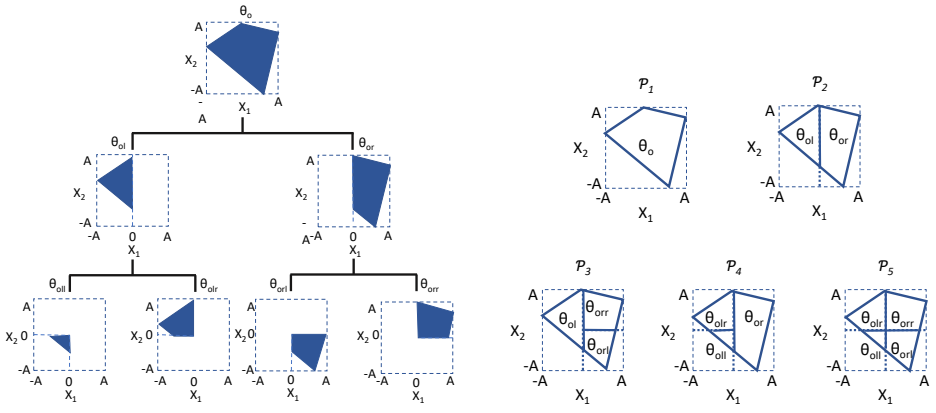
The main building block of our anomaly detection algorithm is the online bandwidth optimized KDE employed in regions of the observation space, i.e., at each node of the defined tree. In this section, our discussion about this building block is confined to a single region from the observation space. In Section IV-B, we introduce the combination of KDEs from all possible regions.

1) Conventional KDE: We use the KDE method at each node θ_k to estimate the local density in the corresponding region of the observation space based on the instances falling in that region. Let $\mathbf{x}$ be any point in θ_k, i.e., $\mathbf{x} \in \theta_k$, at time t before observing $\boldsymbol{x}_t$ along with the past data $\{\boldsymbol{x}_i\}_{i=1}^{t-1} = \boldsymbol{x}^{t-1}$. Then, KDE is obtained as

$$\hat{f}_t^{\theta_k}(\mathbf{x}; \delta) = \frac{1}{\tau_{t-1}^{\theta_k} \delta^d} \sum_{r: \boldsymbol{x}_r \in \theta_k, 1 \leq r \leq t-1} K\Big(\frac{\mathbf{x} - \boldsymbol{x}_r}{\delta}\Big), \qquad (3)$$

where $\tau_{t-1}^{\theta_k} = \sum_{r=1}^{t-1} 1_{\{\boldsymbol{x}_r \in \theta_k\}}$ is the number of instances falling in θ_k until time $t-1$, δ is the bandwidth and $K(\cdot)$ is a kernel function [12]. In this work, we use the Gaussian

[2]Here, θ denotes both the node and the corresponding region for notational simplicity, where the precise meaning can be understood from the text.



(a) (b)

Fig. 1: In this example, a depth-2 binary tree partitions the observation space $S = [-A, A] \times [-A, A]$. (a) Partition regions are illustrated shaded at their corresponding nodes. (b) Every pruning of the introduced partitioning tree defines a unique partition of the observations space.

kernel (or radial basis) function [33] $K(\boldsymbol{x}) \triangleq \frac{1}{(2\pi)^{\frac{d}{2}}} e^{-\frac{\left\|\boldsymbol{x}\right\|^2}{2}}$. After inserting this into (3), we obtain

$$\hat{f}_t^{\theta_k}(\mathbf{x}; g) = \frac{1}{\tau_{t-1}^{\theta_k}} \sum_{r:\boldsymbol{x}_r \in \theta_k, 1 \leq r \leq t-1} k(\mathbf{x}, \boldsymbol{x}_r; g), \tag{4}$$

where $k(\mathbf{x}, \boldsymbol{x}_r; g) \triangleq \frac{1}{N_g} \exp(-g\|\mathbf{x} - \boldsymbol{x}_r\|^2)$ with $g = \frac{1}{2\delta^2}$ being the bandwidth parameter and $N_g = (\frac{\pi}{g})^{\frac{d}{2}}$ is the normalization constant.

2) Online KDE: The summation in (4) requires all of the pairwise kernel evaluations between the test instance $\mathbf{x}$ and all the previous instances, which is computationally prohibitively complex. For an efficient kernel evaluation, we explicitly map the data points $(\mathbf{x}, \boldsymbol{x}_r)$ in the kernel evaluation into a high dimensional kernel feature space with an explicit and randomized "compact" (cf. the remark below for compactness) mapping $\phi : \mathbb{R}^d \to \mathbb{R}^m$ such that the kernel evaluation can be arbitrarily well approximated as an inner product of the mapped points [33] $k(\mathbf{x}, \boldsymbol{x}_r; g) \approx \phi(\mathbf{x}; g)'\phi(\boldsymbol{x}_r; g)$. In order to find such a mapping $\phi(\cdot)$, we exploit the fact that Fourier transform of a symmetric shift invariant and continuous positive definite kernel represents a probability distribution [33], yielding

$$\begin{aligned} k(\mathbf{x}, \boldsymbol{x}_r; g) = \int_{(\boldsymbol{w},b)\in(R^{d\times 1}\times R)} & f_{\boldsymbol{w},b}(\boldsymbol{w}, b; g) \times \\ & \left(\frac{g}{\pi}\right)^{d/2} \sqrt{2}\cos(\boldsymbol{w}'\mathbf{x} + b) \times \\ & \left(\frac{g}{\pi}\right)^{d/2} \sqrt{2}\cos(\boldsymbol{w}'\boldsymbol{x}_r + b) d\boldsymbol{w} db, \end{aligned} \tag{5}$$

$$\begin{aligned} k(\mathbf{x}, \boldsymbol{x}_r; g) = E_{w,b}\Bigg[& \left(\frac{g}{\pi}\right)^{d/2} \sqrt{2}\cos(\boldsymbol{w}'\mathbf{x} + b) \\ & \left(\frac{g}{\pi}\right)^{d/2} \sqrt{2}\cos(\boldsymbol{w}'\boldsymbol{x}_r + b) \Bigg] \end{aligned} \tag{6}$$

where $f_{\boldsymbol{w},b}(\boldsymbol{w}, b; g) = (4g\pi)^{-\frac{d}{2}}(e)^{-\frac{\left\|\boldsymbol{w}\right\|^2}{4g}} \times \frac{1}{2\pi} 1_{\{0 \leq b \leq 2\pi\}}$ is a valid probability density function obtained as a result of the Fourier transform of the kernel being used together with an auxiliary uniform randomization over b [33]. Then, by approximating the expectation result for the kernel evaluation in (6) through a sample mean (due to law of large numbers), we obtain the final approximation with the mapping function

$$\begin{aligned} \phi(\mathbf{x}; g) \triangleq \sqrt{\frac{2}{m}} \left(\frac{g}{\pi}\right)^{d/2} \Bigg[& \cos\left(\sqrt{2g}\boldsymbol{w}_1'\mathbf{x} + b_1\right), \\ & \cos\left(\sqrt{2g}\boldsymbol{w}_2'\mathbf{x} + b_2\right), ..., \cos\left(\sqrt{2g}\boldsymbol{w}_m'\mathbf{x} + b_m\right) \Bigg]', \end{aligned} \tag{7}$$

where $\{(\boldsymbol{w}_i, b_i)\}_{i=1}^m$ is a random sample from $f_{\boldsymbol{w},b}(\boldsymbol{w}, b; \frac{1}{2})$ and $(\boldsymbol{w}_i, b_i)$ are i.i.d. realizations [33]. Note that the algorithm that we introduce can be used with any symmetric shift invariant and positive definite kernel by only using the correct randomization $f_{\boldsymbol{w},b}(\boldsymbol{w}, b; \frac{1}{2})$ (which is the Fourier transform of the kernel) and modifying $\phi(\mathbf{x}; g)$ accordingly [33].

Remark: Note that $\boldsymbol{w}$ and b form $f_{\boldsymbol{w},b}(\boldsymbol{w}, b; g)$, the probability density function in (6), such that the dimensions of $\boldsymbol{w}$ are independent scalar random variables and each has a Gaussian distribution with 0 mean and $2g$ variance, and b is a scalar random variable that is uniformly distributed between 0 and 2π. In our method, we also aim to use adaptive bandwidth by changing g in time to improve the performance of the estimation. Therefore, we define our mapping function in (7) with "$\cos\left(\sqrt{2g}\boldsymbol{w}_i'\mathbf{x} + b_i\right)$" to make the sampling procedure of $\boldsymbol{w}$'s independent of g. This modification corresponds to drawing samples from $f_{\boldsymbol{w},b}(\boldsymbol{w}, b; \frac{1}{2})$ such that the dimensions of $\boldsymbol{w}$ have a Gaussian distribution with 0 mean and 1 variance, and b is uniformly distributed between 0 and 2π. Then, we multiply with $\sqrt{2g}$ in (7) to adjust for the desired covariance.

We obtain the density estimation in (4) with the sample

mean approximation

$$\hat{f}_t^{\theta_k}(\mathbf{x};g) = \frac{1}{\tau_{t-1}^{\theta_k}} \sum_{r:\boldsymbol{x}_r\in\theta_k, 1\le r\le t-1} k(\mathbf{x},\boldsymbol{x}_r;g) \tag{8}$$

$$\simeq \frac{1}{\tau_{t-1}^{\theta_k}} \sum_{r:\boldsymbol{x}_r\in\theta_k, 1\le r\le t-1} \phi(\mathbf{x};g)'\phi(\boldsymbol{x}_r;g) \tag{9}$$

$$= \frac{t-1}{\tau_{t-1}^{\theta_k}} \phi(\mathbf{x};g)'\phi_t^{\theta_k}(g), \tag{10}$$

where

$$\phi_t^{\theta_k}(g) \triangleq \frac{1}{t-1} \sum_{r:\boldsymbol{x}_r\in\theta_k, 1\le r\le t-1} \phi(\boldsymbol{x}_r;g),$$

which can be implemented through sequential updates (after normalization with the total probability mass in θ_k) as in line 11 in Algorithm 1.

Remark: This approximate form (but asymptotically exact form as $m\to\infty$) in (10) for the density estimation in (4) removes the pairwise kernel evaluations, and allows us to evoke only a single dot product for the whole expression. This leads to an efficient online processing at the cost of the explicit and randomized mapping of each instance through $\phi(\cdot)$. We emphasize that this mapping is "compact" and the addition of mapping cost is fairly low: the quality of the approximation of the expectation through the sample mean in (10) improves at an exponential rate in m due to the Hoeffding's inequality. Consequently, we obtain an online kernel density estimation at a negligible transformation cost.

3) Piecewise Online KDE: We point out that every pruning of our tree defines a partition of the observation space with regions attached to the leaves of the pruned tree, and there is an online KDE at each region. Following this observation, for a given partition $\mathcal{P}=\{\theta_1,\theta_2,\cdots,\theta_N\}$, we combine the online KDEs $\hat{f}_t^{\theta_k}$'s at the partition regions θ_k's, i.e., pieces, and form a piecewise online KDE $\hat{f}_t^{\mathcal{P}}$ to model the density for the complete data. Recall that an online KDE is responsible for only those instances falling in its region, hence each online KDE must be normalized with the corresponding probability mass $\frac{\tau_{\theta_k}}{t-1}$ in its region θ_k while combining them. As a result, for each partition $\mathcal{P}$ (there are approximately $(1.5)^{2^D}$ many partitions defined over our tree), we define a piecewise online KDE $\hat{f}_t^{\mathcal{P}}(x;g)$ as

$$\hat{f}_t^{\mathcal{P}}(\mathbf{x};g) = \frac{\tau_t^{\theta_k}}{t-1}\hat{f}_{t-1}^{\theta_k}(\mathbf{x};g) = \phi(\mathbf{x};g)'\phi_t^{\theta_k}(g), \tag{11}$$

which matches the estimation by the local online KDE in region $\theta_k \ni \mathbf{x}$.

The updates for the piecewise online KDE are based on the recursions of the local online KDEs. Note that a local online KDE needs to be updated only when its region observes an instance. Hence, we keep the last update time τ'_{θ_k} and use it for the next coming observation in that region with the recursion $\phi_{t+1}^{\theta_k}(g) = \frac{\tau'_{\theta_k}\phi_t^{\theta_k}(g)+\phi(\mathbf{x};g)}{t}$.

4) Bandwidth optimized piecewise online KDE: In addition, our piecewise online KDE learns the optimal bandwidths separately for its constituent local estimators. For this purpose, we allow the bandwidths for the constituent local estimators to be different from region to region, and also time varying in line with the increasing data length. Hence, our technique is tuned to local variations of the data manifold as well as the rate of increase in the data size of the stream even in a non-stationary environment. In contrast, conventional approaches use one global KDE with a fixed time invariant bandwidth which is certainly a strong limitation that we remove.

Algorithm 1 Hierarchical Kernel Density Estimator (HKDE)

1: Set tree depth D, bandwidth set $\mathbb{G}$ and learning rate h
2: Initialize bandwidth priors $\alpha_0^{\theta_k}(g) = \frac{1}{|\mathbb{G}|}$
3: Draw m i.i.d. samples $(\boldsymbol{w},b)\sim f_{\boldsymbol{w},b}(\boldsymbol{w},b,\frac{1}{2})$: $\boldsymbol{w}\in\mathbb{R}^d$ and $b\in[0,2\pi]$
4: Evaluate $\phi_0^{\theta_k}(g) = \sqrt{\frac{2}{m}}\left(\frac{g}{\pi}\right)^{d/2}[\cos(b_1),\cos(b_2),...,\cos(b_m)]'$
5: **for** $t=1,2,\ldots$ **do**
6: Calculate $\phi(\boldsymbol{x}_t;g)$, $\forall g\in\mathbb{G}$
7: Calculate the loss $l_t^{\theta_k} = -\log\left(\frac{\tau_{\theta_k}}{t-1}\sum_{\forall g\in\mathbb{G}}\alpha_t^{\theta_k}(g)\phi(\boldsymbol{x}_t;g)'\phi_t^{\theta_k}(g)\right), \forall k$
8: Obtain the estimation $f_t(x_t) = \sum_{k=0}^{D} c_t^{\theta_k}\exp(-l_t^{\theta_k})$
9: Update $\tau'_{\theta_k} = t, \forall k$
10: Update $Z_{t+1}^{\theta_k}(g) = Z_t^{\theta_k}(g)\exp(h\log\phi(\boldsymbol{x}_t;g)'\phi_t^{\theta_k}(g))$ for all (k,g)
11: Update $\phi_{t+1}^{\theta_k}(g) = \frac{\tau'_{\theta_k}\phi_t^{\theta_k}(g)+\phi(\boldsymbol{x}_t;g)}{t}$ for all (k,g)
12: Update $\alpha_{t+1}^{\theta_k}(g) = \frac{Z_{t+1}^{\theta_k}(g)}{\sum_{\forall g} Z_{t+1}^{\theta_k}(g)}$ for all (k,g)
13: Update $c_{t+1}^{\theta_k}$ for $k=0,...,D$, i.e., nodes for $\boldsymbol{x}_t$ in each level
14: **end for**

For this purpose, we use a finite set of bandwidths, and run our online local KDEs in parallel with all bandwidth values in the set. The computational complexity of this parallel running in our approach is only $O(D)$, since the local KDEs of only those regions containing $\boldsymbol{x}_t$ must be updated. In time, our approach identifies the optimal bandwidth g^* in node θ_k, which has the highest log likelihood (after scaling with h that is exactly the log-likelihood when $h=1$)

$$Z_t^{\theta_k}(g^*) = \max_g \prod_{j:\boldsymbol{x}_{t_j}\in\theta_k, 1\le t_j<t} \exp(h\log\hat{f}_{t_j}^{\theta_k}(x_{t_j};g)).$$

Note that the introduced optimality can be improved to any desired degree by using a larger finite bandwidth set $\mathbb{G}=\{g_1,g_2,\ldots,g_{|\mathbb{G}|}\}$, from which the optimal bandwidth is inferred. However, the bandwidth set does not have to be fixed in time: it can be dynamic of the same finite cardinality all the time. Namely, as time goes, all bandwidth g values become useless except the one which takes all of the weight. This pattern can be traced and one can continuously inject/replace the obsolete candidate bandwidths with new larger ones such that the cardinality is kept fixed. Hence, our introduced method is certainly still practical under such a continuously improving optimality.

Our convergence result is in line with the weighted majority approach in the mixture of experts [35], [40]. Briefly, we assign time varying probabilities $\alpha_t^{\theta_k}(g)$ to each node on

the tree for all bandwidth values in the set $\mathbb{G}$ such that $\sum_{g=g_1}^{g_{|G|}} \alpha_t^{\theta_k}(g) = 1, \alpha_t^{\theta_k} \geq 0, \forall t$. Initially, all bandwidths at a node θ_k have equal priors, i.e., $\alpha_0^{\theta_k}(g) = \frac{1}{|\mathbb{G}|}, \forall g \in \mathbb{G}$. Then, we update these probabilities as in line 10 and 12 in Algorithm 1 to converge to the best bandwidth in the set in the sense of likelihood maximization by learning the combination parameters $\alpha_t^{\theta_k}(g)$ in time. Namely, the final density estimation of the node θ_k is a combination of density estimations evaluated with different bandwidth values as

$$\hat{f}_t^{\theta_k}(\mathbf{x}) = \sum_{\forall g \in \mathbb{G}} \alpha_t^{\theta_k}(g) \hat{f}_t^{\theta_k}(\mathbf{x}; g), \tag{12}$$

where $\hat{f}_t^{\theta_k}(\mathbf{x}; g)$ is the density estimation evaluated with bandwidth g as in (10) and $\hat{f}_t^{\theta_k}(\mathbf{x})$ is asymptotically tuned to the best g^* since $\alpha_t^{\theta_k}(g^*) \to 1$ as $t \to \infty$. The reason is that we define $\alpha_t^{\theta_k}(g)$ through the likelihoods of the bandwidths based on the sum of their losses. Hence, the weight of the bandwidth value, which provides the best performance among the others, converges to 1 while all the others converge to 0 in time based on the learning parameter h.

Note that this explanation for convergence to 1 is valid if the data statistics is stationary, or asymptotically stationary by allowing nonstationarity in the transient state, which is necessary for the best performing g to consistently accumulate a performance that is increasingly better compared to others so that the best performing g can take all the weight. In case of nonstationarity (the distribution that the data instances are coming from might be changing in time, i.e., might be time-varying), the aforementioned convergence is not true but even in this case, we still can identify the best g by the largest weight.

Similarly, for a given partition $\mathcal{P} = \{\theta_1, \theta_2, \cdots, \theta_N\}$, we obtain bandwidth optimized piecewise online KDE as $\hat{f}_t^{\mathcal{P}}(\mathbf{x}) = \sum_{\forall g \in \mathbb{G}} \alpha_t^{\theta_k}(g) \hat{f}_t^{\mathcal{P}}(\mathbf{x}; g)$.

We emphasize that, since $\alpha_t^{\theta_k}(g^*) \to 1$ as $t \to \infty$, our approach successfully identifies the best bandwidth in region θ_k in a time varying manner. The presented convergence result here indicates that we essentially favor the KDEs with smoother bandwidths in the beginning of the stream and gradually switch to the ones with steeper bandwidths as the data overwhelm, which is specifically to handle the overfitting issue directly from the bandwidth perspective.

B. Efficient Combination of Hierarchically Organized Bandwidth Optimized Piecewise Online KDEs

We point out that one can produce a bandwidth optimized piecewise online KDE for every given partition of the observation space. On the other hand, it is possible to obtain a doubly exponential large number, i.e., $(1.5)^{2^D}$ with D being the depth, of many different partitions by all possible prunings of our partitioning tree. Hence, it is possible for one to approximate any desired partition with a tree-defined partition by using a relatively small D.

It is critical to use the right partition in the case of anomaly detection with density estimation. Specifically, in the case of sequentially observed instances, the optimal partition is also not fixed but time varying. Simpler partitions with less number of regions are advantageous at the beginning of the stream, whereas one needs to use partitions with more complexities as more data become available. Hence, our algorithm produces a weighted combination of the piecewise online KDEs each of which corresponds to a specific partition. The combination weights are obtained based on the performance, i.e., the data likelihood, of each partition. Hence, the weights are time-varying and our algorithm relies mostly on the best-performing partitions due to the weighting where a better performing partition always receives a higher weight. In addition, we mathematically prove that our algorithm asymptotically performs at least as well as the best piecewise online KDE from the introduced large class of tree-defined partitions.

To this end, we introduce *a computationally highly efficient* combination of all of the bandwidth optimized piecewise online KDEs of prunings of our partitioning tree. [3] The resulting final density estimator is given as

$$f_t(\mathbf{x}) = \sum_{k=1}^{n_p} \tilde{c}_t^{\mathcal{P}_k} \hat{f}_t^{\mathcal{P}_k}(\mathbf{x}). \tag{13}$$

Here, $n_p \approx (1.5)^{2^D}$ and $\tilde{c}_t^{\mathcal{P}_k}$ is the weight of the partition $\mathcal{P}_k$ at time t as

$$\tilde{c}_t^{\mathcal{P}_k} = \frac{2^{-\rho(\mathcal{P}_i)} \exp\big(-h \sum_{u:\boldsymbol{x}_u \in \theta_{k_u}, 1 \leq u < t} (l_u^{\theta_{k_u}})\big)}{\tilde{C}}$$

with $\tilde{C} = \sum_{i=1}^{(1.5)^{2^D}} 2^{-\rho(\mathcal{P}_i)} \exp\big(-h \sum_{u:\boldsymbol{x}_u \in \theta_k, 1 \leq u < t} (l_u^{\theta_{i_u}})\big)$ is the normalization constant where $\sum_{u:\boldsymbol{x}_u \in \theta_{k_u}, 1 \leq u < t} (l_u^{\theta_{k_u}})$ is the total loss of the partition $\mathcal{P}_k$ and θ_{k_u} is the corresponding node of $\mathcal{P}_k$ at time u. We define our loss for the node θ_k as

$$l_t^{\theta_k} = -\log(\tilde{f}_t^{\theta_k}(\mathbf{x})), \tag{14}$$

which is the point-wise log-likelihood based on the estimator at that node. Also, $\rho(\mathcal{P}_i) = n_i + l_i - 1$ corresponds to the number of bits required to represent the partition $\mathcal{P}_i$ where n_i is the total number of nodes in partition $\mathcal{P}_i$ and l_i is the number of nodes with depth smaller than D in the partition.

On the contrary, combining $(1.5)^{2^D}$ partitions or running $(1.5)^{2^D}$ many bandwidth optimized piecewise online KDEs in parallel is computationally intractable. However, there exists an efficient solution [36], [35] to combine them with computational complexity only $O(D)$. Instead of combining the partition specific estimations as in (13), one can obtain the same exact final estimation result of (13) by only combining $D + 1$ many node specific estimations as

$$f_t(\mathbf{x}) = \sum_{k=0}^{D} c_t^{\theta_k} \tilde{f}_t^{\theta_k}(\mathbf{x}) \tag{15}$$

where $\tilde{f}_t^{\theta_k}(\mathbf{x})$ is the density estimation of the corresponding node at the k^{th} layer of the tree, and $c_t^{\theta_k}$ is the weight of the node θ_k at time t.

Remark: Recall that the density estimation of a partition is equal to the estimation of the corresponding node in that

[3] For notational simplicity, we represent the normalized bandwidth optimized density estimation of the node θ_k at time t as $\tilde{f}_t^{\theta_k}(\mathbf{x})$ in the rest of the paper by $\tilde{f}_t^{\theta_k}(\mathbf{x}) \leftarrow \frac{\tau_{t-1}^{\theta_k}}{t-1} \hat{f}_t^{\theta_k}(\mathbf{x})$.

partition. Hence, although we have $(1.5)^{2^D}$ many different partitions, we only have $D+1$ unique estimations since the instance travels through only D layers. Therefore, combining the partitions as in (13) exactly matches to combining the corresponding nodes at layers as in (15) with re-collected weights in (16). This allows us to solve the problem in a more efficient way since it requires combining only $D+1$ density estimations.

The weight $c_t^{\theta_k}$ of the node θ_k can be collected as [41]

$$c_t^{\theta_k} = \sum_{\mathcal{P}_k : \hat{f}_t^{\mathcal{P}_k}(\mathbf{x}) = \tilde{f}_t^{\theta_k}(\mathbf{x})} \tilde{c}_t^{\mathcal{P}_k}. \tag{16}$$

The weight $c_t^{\theta_k}$ can also be *recursively* calculated by the evaluated loss or reward of the nodes as explicitly shown in [36]. We directly provide the recursion here while referring the reader to [36] for the details of the proof of this recursion. Then, one can obtain the weight of the node θ_k as

$$c_t^{\theta_k} = \frac{2^{-(D_k+1)} \times \prod_{j=1}^{D_k} L_{\theta_j}(\boldsymbol{x}^{t-1})}{L_{\theta_o}(\boldsymbol{x}^{t-1})} \times \exp\Big(-h \sum_{u:\boldsymbol{x}_u \in \theta_k, 1 \le u < t} (l_u^{\theta_k})\Big), \tag{17}$$

where θ_o is the root node, D_k is the layer of θ_k on the tree, θ_j is the other child of the parent of θ_k at the j^{th} layer of the tree and $L_{\theta_j}(\boldsymbol{x}^{t-1})$ is an auxiliary variable[4] that allows the recursive calculation [36] for any node θ_j using

$$L_{\theta_j}(\boldsymbol{x}^{t-1}) = \begin{cases} \exp(-h \sum_{u:\boldsymbol{x}_u \in \theta_j, 1 \le u < t} l_u^{\theta_j}), & (\theta_j \text{ is leaf}) \\ \frac{1}{2} L_{\theta_{jl}}(\boldsymbol{x}^{t-1}) L_{\theta_{jr}}(\boldsymbol{x}^{t-1}) & \\ + \frac{1}{2} \exp(-h \sum_{u:\boldsymbol{x}_u \in \theta_j, 1 \le u < t} l_u^{\theta_j}), & (\text{otherwise}) \end{cases} \tag{18}$$

with θ_{jl} and θ_{jr} being the left and right child nodes of the node θ_j, respectively. We point out that this recursion is of computational complexity $O(D)$.

Based on this recursion (18) (cf. line 13 in Algorithm 1) and the description in (15) (cf. line 7 and 8 in Algorithm 1), which is based on the bandwidth probability assignments (12) (cf. line 7, 10 and 12 in Algorithm 1), we obtain our computationally efficient algorithm presented in Algorithm 1.

The following theorem provides the bound for the regret of our final density estimator (2), which combines the density estimations of the partitions.

Theorem 1. *When our KDE based hierarchical model is applied with Algorithm 1, the regret defined in* (2) *is upper bounded as*

$$R(T) = \sum_{t=1}^{T} \big(-\log(f_t(\boldsymbol{x}_t))\big) - \min_{\mathcal{P}_i} \Big\{ \sum_{t=1}^{T} \big(-\log(\hat{f}_t^{\mathcal{P}_i}(\boldsymbol{x}_t))\big) \Big\} \le \frac{\rho(\mathcal{P}_i)\log(2)}{h} \le \frac{(2^D-1)\log(2)}{h}, \forall i \tag{19}$$

[4]This variable can be interpreted, cf. [36], as a universal probability assignment based on the losses accumulated over the tree and any legitimate weighting of the prunings depending on the complexities of the resulting pruned trees.

where $\mathcal{P}_i$ is the i^{th} partition in the competition class of estimators, T is the length of sequence $\boldsymbol{x}^T = \{\boldsymbol{x}_i\}_{i=1}^T$, h is the learning rate and $\rho(\mathcal{P}_i) = n_i + l_i - 1$ measures the complexity of each partition by the number of bits required to represent that partition $\mathcal{P}_i$. Here, $\rho(\mathcal{P}_i)$ is defined based on the total number n_i of nodes in the partition and the number l_i of nodes with depth smaller than D in the partition.

Proof of Theorem 1. The proof follows similar lines to the one in [35], with the difference that we use the negative log-loss of the point-wise likelihood instead of the squared error loss. Following the definition in (18), $L_{\theta_o}(\boldsymbol{x}^T)$ for the root node can be written as [36]

$$L_{\theta_o}(\boldsymbol{x}^T) = \sum_{i=1}^{(1.5)^{2^D}} 2^{-\rho(\mathcal{P}_i)} \exp(-h \sum_{t=1}^{T} l_t^{\theta_{t_i}}),$$

where θ_{t_i} is the corresponding node of partition $\mathcal{P}_i$ at time t and $l_t^{\theta_k} = -\log(\tilde{f}_t^{\theta_k}(\mathbf{x}))$ as in (14). Then, we obtain

$$L_{\theta_o}(\boldsymbol{x}^T) \ge 2^{-\rho(\mathcal{P}_i)} \exp(-h \sum_{t=1}^{T} l_t^{\theta_{t_i}}). \tag{20}$$

Also, when we apply (18) recursively to the root node, we observe that

$$\frac{L_{\theta_o}(\boldsymbol{x}^t)}{L_{\theta_o}(\boldsymbol{x}^{t-1})} = \sum_{k=0}^{D} c_t^{\theta_k} \exp(h \log \tilde{f}_t^{\theta_k}(\boldsymbol{x}_t)),$$

where $c_t^{\theta_k}$ is defined as in (17). Since $\sum_{k=0}^{D} c_t^{\theta_k} = 1$ and $\exp(h\log(\cdot))$ is concave for the learning rate $0 \le h \le 1$, we obtain by Jensen's inequality

$$\frac{L_{\theta_o}(\boldsymbol{x}^t)}{L_{\theta_o}(\boldsymbol{x}^{t-1})} \le \exp\left(h \log\Big(\sum_{k=0}^{D} c_t^{\theta_k} \tilde{f}_t^{\theta_k}(\boldsymbol{x}_t)\Big)\right),$$

where $f_t(\boldsymbol{x}_t) = \sum_{k=0}^{D} c_t^{\theta_k} \tilde{f}_t^{\theta_k}(\boldsymbol{x}_t)$ is final density estimation (15). Since we have

$$L_{\theta_o}(\boldsymbol{x}^T) = \prod_{t=1}^{T} \frac{L_{\theta_o}(\boldsymbol{x}^t)}{L_{\theta_o}(\boldsymbol{x}^{t-1})} \le \prod_{t=1}^{T} \exp(h \log(f_t(\boldsymbol{x}_t))) = \exp\big(h \sum_{t=1}^{T} \log(f_t(\boldsymbol{x}_t))\big),$$

where $L_{\theta_o}(\boldsymbol{x}^0) = 1$, we write (20) as

$$\exp\left(h \sum_{t=1}^{T} \log\Big(f_t(\boldsymbol{x}_t)\Big)\right) \ge L_{\theta_o}(\boldsymbol{x}^T) \ge 2^{-\rho(\mathcal{P}_i)} \exp(-h \sum_{t=1}^{T} l_t^{\theta_{t_i}}).$$

$$\sum_{t=1}^{T} \Big(-\log(f_t(\boldsymbol{x}_t))\Big) \le \frac{\log(2)\rho(\mathcal{P}_i)}{h} - \frac{(-h\sum_{t=1}^{T} l_t^{\theta_{t_i}})}{h}.$$

Finally, we obtain

$$\sum_{t=1}^{T} \big(-\log(f_t(\boldsymbol{x}_t))\big) - \sum_{t=1}^{T} \Big(-\log(\hat{f}_t^{\mathcal{P}_i}(\boldsymbol{x}_t))\Big) \le \frac{\rho(\mathcal{P}_i)\log(2)}{h}.$$

Since $\rho(\mathcal{P}_i) \leq 2^D - 1$, we obtain the following regret bound $R(T) \leq \frac{(2^D-1)\log(2)}{h}$ where h must be chosen in $[0, 1]$ and it defines the learning rate. □

Hence, the normalized regret $\frac{R(T)}{T}$ is convergent to 0 at a $O(1/T)$ rate. This shows that our density estimation algorithm is regret optimal, i.e., it asymptotically performs (in terms of the accumulated log-likelihood) as well as the piecewise online KDE that corresponds to the best tree-defined partition.

V. Anomaly Detector

We compare the estimated probability density $f_t(\boldsymbol{x}_t)$ at each time t with a threshold to decide if $\boldsymbol{x}_t$ is anomalous. A data instance x_t having a sufficiently low probability density is considered anomalous with respect to the rule

$$\hat{d}_t = \begin{cases} +1, & f_t(\boldsymbol{x}_t) < \varsigma_t \text{ (anomalous)} \\ -1, & f_t(\boldsymbol{x}_t) \geq \varsigma_t \text{ (normal)} \end{cases}. \tag{21}$$

Based on this rule, we find the optimal threshold that achieves the best anomaly detection performance by updating its value in time. For this, we use the logistic loss function of ς_t that is to be minimized [42]: $r_t = \log(1 + \exp(-(\varsigma_t - f_t(\boldsymbol{x}_t))d_t)$. Note that the loss is small if $d_t = \hat{d}_t$, and it increases if the difference $(\varsigma_t - f_t(\boldsymbol{x}_t))$ is increasing when $d_t \neq \hat{d}_t$. Hence, the model learns the optimal threshold that differentiates the normal and anomalous data in time with loss minimization.

Then, the threshold is updated by using the Online Gradient Descent (OGD) method [11], which learns the optimal parameter minimizing the desired loss, as in Algorithm 2. We calculate the gradient of the loss function with respect to ς_t as $\nabla_{\varsigma_t} r_t = -d_t\Big(1 + \exp\big((\varsigma_t - f_t(\boldsymbol{x}_t))d_t\big)\Big)^{-1}$ and the update follows as $\varsigma_{t+1} = P_{\mathbb{T}}(\varsigma_t - \eta_t \nabla_{\varsigma_t} r_t)$ where η_t is the learning rate, which depends on the number of data instances and the convexity of the defined loss function [11], $P_{\mathbb{T}}(x) = \arg\min_{y \in \mathbb{T}} ||x - y||^2$ is the projection onto convex threshold set $\mathbb{T}$. Since the loss function r_t is convex, our algorithm converges to the optimal threshold in the set by minimizing the cumulative loss.

VI. Experiments

In this section, we extensively evaluate the performance of our anomaly detector on an artificial dataset of a mixture of two Gaussian components, and several other real datasets such as Occupancy [43], Breast Cancer Wisconsin [44], Pen digits [44], Susy [45], Thyroid [44], Pima [44], Mammography and Liver [44]. Comparisons with one-class Support Vector Machine (SVM) [24], incrementally trained SVM (we represent as "I-SVM" throughout this section) [46], K-nearest neighbors (K-NN) [25], K-D tree nearest neighbor search (K-D Tree) [47], K-Localized p-Value Estimation (K-LPE) [9], online oversampling Principal Component Analysis (online osPCA) [48] and Fourier Online Gradient Descent (FOGD) [34] are presented, which are commonly used for anomaly detection. We also include comparisons with anomaly detection method based on KDE with local bandwidths where the bandwidth depends on the distance to the nearest neighbor of the observed point as in [49], and we represent this technique as "K-KDE". Finally, we include the performance of our method without partitioning, which is represented as "Kernel Density Estimation with Adaptive Bandwidth (KDE-AB)". We use the receiver operating characteristics (ROC) and the Area Under Curve (AUC) to compare the anomaly detection performances of the algorithms. The ROC curves are obtained by plotting the rates of true detection versus false alarm while varying the corresponding threshold[5] in each algorithm. For this, we use the scores, i.e., values that are thresholded to detect anomalies, assigned to the observed instances by the algorithms except K-NN and K-D Tree. For them, we consider the score as the proportion of the neighbors with positive class label. The parameters (ν and γ for the SVM, I-SVM and K for the K-NN, K-D Tree and K-LPE) of the compared algorithms[6] are optimized with 10-fold cross validation, where we reserve the 75% of each dataset for training, and the remaining for the test. Since the other methods including our algorithm AD-HKDE does not need cross validation as they are online, we set their parameters, i.e., D, h and $\mathbb{G}$ for AD-HKDE, based on a small preliminary experiment. In Algorithm 1, the tree depth and the learning rate are fixed as $D = 3$ and $h = 0.01$, respectively.

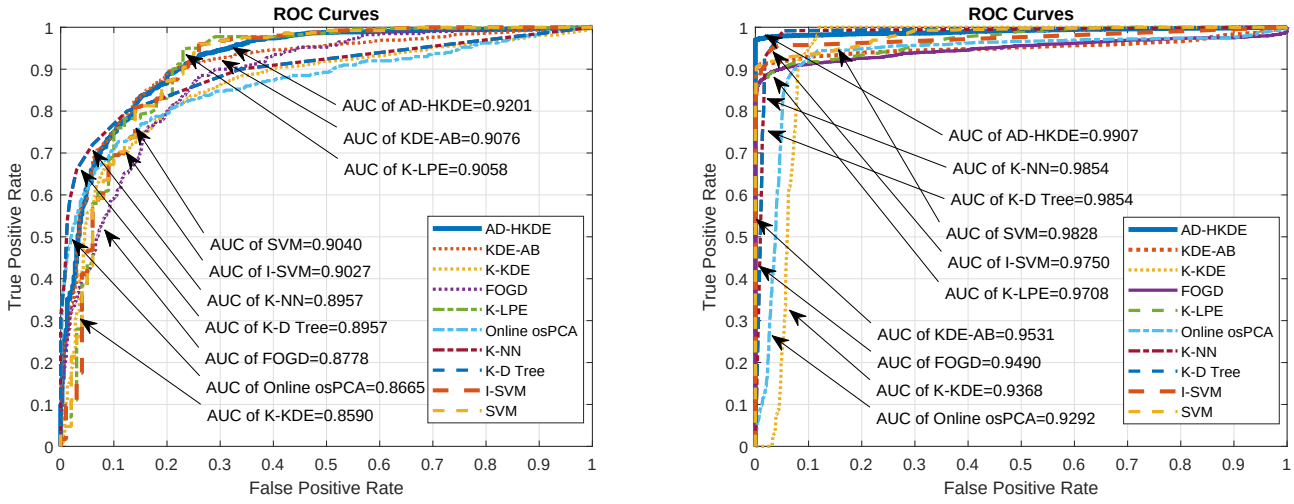
Since in general the higher-layer nodes on our partitioning tree observe exponentially smaller number of data instances compared to the lower-layer ones, we form the bandwidth set $\mathbb{G} = \{g_1, \cdots, g_k\}$ with relatively small g values for the root node, and larger g values for the leaf nodes. We use $|\mathbb{G}| = k = 4$, and form the sets such that $\mathbb{G} = \epsilon \times \{2^0, 2^1, \cdots, 2^{k-1}\}$, where we increase $\epsilon = \mathbb{L}(l)$ (l denotes the layer and $\mathbb{L} = \{0.01, 0.5, 1.5, 2\}$) from the root to the leaves. Hence, the union of all such sets from the layers of the tree yields relatively large actual set (whose size is proportional to the tree depth) of bandwidth parameter values in which our algorithm optimizes for g with low computational and space complexity. Note that, in general, optimal kernel bandwidth δ is proportional to $N^{-\frac{1}{5}}$ (N: number of data points) [12] ($g = \frac{1}{\delta}$ is proportional to $N^{\frac{1}{5}}$). In online framework, the number of data points changes from 1 to T (T: total number of data instances). Therefore, in our experiments, we define bandwidth set such that its maximum is proportional to $T^{\frac{1}{5}}$ based on our largest dataset. Finally, the dimensionality is reduced to $d = 3$ for all datasets (except the artificial one which is already 2-dimensional) by using PCA for only partitioning purposes, otherwise the full dimension is used in the remaining parts

Algorithm 2 Anomaly Detector

Specify $\tilde{\eta}$ and initialize the learning rate $\eta_1 = \frac{1}{\tilde{\eta}}$
Specify the threshold set $\mathbb{T}$ and initialize ς_1
for $t = 1, 2, \ldots$ **do**
  Calculate the learning rate $\eta_t = \frac{1}{\tilde{\eta}t}$
  Compare $f_t(\boldsymbol{x}_t)$ with ς_t and declare the decision $\hat{d}_t$
  Observe the actual d_t and calculate $\nabla_{\varsigma_t} r_t$
  Update the threshold $\varsigma_{t+1} = P_{\mathbb{T}}(\varsigma_t - \eta_t \nabla_{\varsigma_t} r_t)$
end for

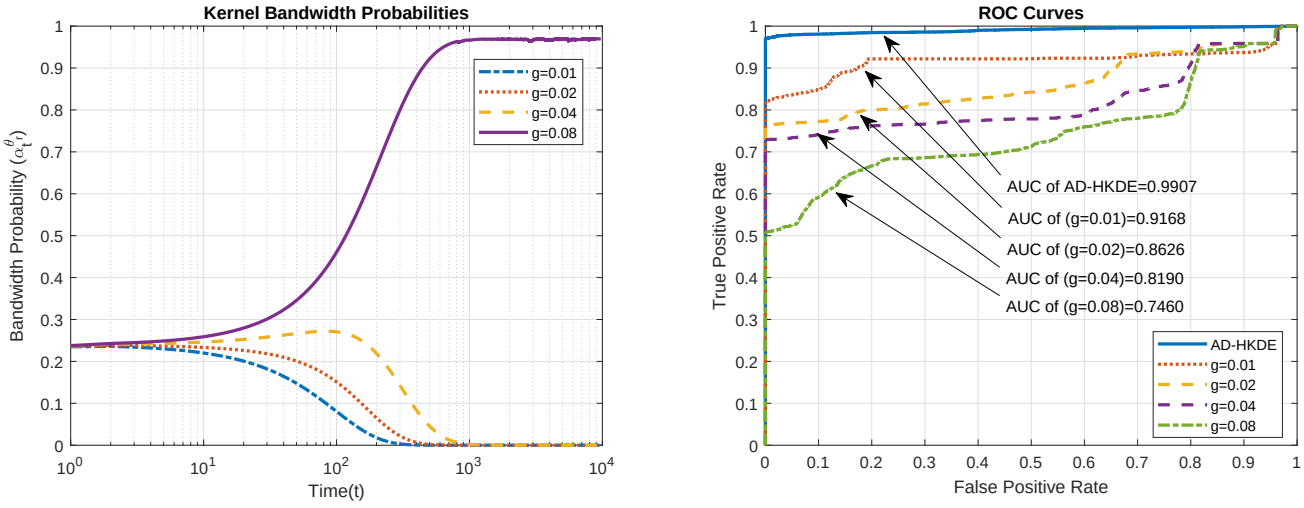
[5]For all experiments, SVM, I-SVM, K-LPE, K-NN and K-D Tree are trained with the training sets, and the ROC/AUC results are obtained based on the test sets. Since the other methods including our method AD-HKDE does not need training, all results for them are based on all data points.

[6]SVM is applied by using the *libsvm* [50] implementation.



(a) (b)

Fig. 2: The ROC performance of the compared algorithms on (a) a mixture of two Gaussian components and (b) the Occupancy dataset.



(a) (b)

Fig. 3: For the occupancy dataset, (a) kernel bandwidth probability variations in time and (b) ROC performances of our algorithm and its static bandwidth version with fixed bandwidths.

such as estimating the density. In Algorithm 2, the learning rate parameter is set as $\tilde{\eta} = 0.001 \in (0, 1)$ and the threshold set is $\mathbb{T} = \{10^{-5}, 2 \times 10^{-5}, \cdots, 1\}$.

In this section, "Anomaly Detector with Hierarchical Kernel Density Estimators (AD-HKDE)" represents our technique.

A. *Artificial Dataset*

In the first part of our experiments, we generate a 2-dimensional dataset of length $T = 2000$, whose probability density is a mixture of two Gaussian components $f \sim \frac{1}{2}N\left(\begin{bmatrix}-1\\-1\end{bmatrix}, \begin{bmatrix}1 & 0\\0 & 4\end{bmatrix}\right) + \frac{1}{2}N\left(\begin{bmatrix}1\\1\end{bmatrix}, \begin{bmatrix}4 & 0\\0 & 8\end{bmatrix}\right)$ as the normal data. Our algorithm does not need anomalous data to work, however, we replace the normal data at 400 randomly picked instances with uniformly distributed anomalous observations to test the anomaly detection performance.

As can be seen in Fig.2a, our algorithm AD-HKDE outperforms all the others in terms of the AUC score, while being largely superior in either relatively smaller or higher false alarm rate regions. We point out to understand the reason that the data consists of two Gaussian components, where the second component has 4 or 2 times larger variance in its dimensions compared to the other component. This specifically detrimentally affects the methods K-NN, K-D Tree and K-LPE. The reason is that, in regions of the higher variance component, one has to base the estimations on wider neighborhoods (compared to the other component) in order to observe the same k number of instances. This certainly degrades their estimation since the underlying density is probably far from being uniform in such wide neighborhoods. K-KDE suffers from the placements of the data points in the observation space due to its point-wise bandwidth. Since its kernel bandwidth

is directly proportional to the distance between the observed instance and its nearest neighbor, it is not able to learn the optimal bandwidth and estimate the density correctly. Online osPCA uses only the first principal direction, and hence it loses the information carried by the second component, which decreases its performance significantly. FOGD, however, suffers from the fixed kernel bandwidth, which does not allow the algorithm to learn the local variations. For the same reason, our algorithm AD-HKDE outperforms SVM and I-SVM since they fit symmetric rbf kernels with equal bandwidths at each instance for each component regardless of the component covariances and the time varying data size. In accordance, our algorithm exploits the local structure of the density and learns the optimal bandwidths for each component even in a time varying manner. Therefore, our algorithm AD-HKDE also outperforms its non-partitioned version, i.e., KDE-AB, which learns the bandwidth based on the whole observation space.

We emphasize that the algorithms SVM, incrementally trained SVM, K-NN, K-D Tree and K-LPE with separate training and test phases are specifically chosen in our experiments due to their competing high performance and comparable model free working environment, but their computational complexity is significantly larger than the one of our online detector. For example, SVM has complexity $O(T_{\text{tr}}^3)$ in training, and $O(T_{\text{tr}}T_{\text{test}})$ in test phase (both are in the worst case). Incrementally trained SVM aims to reduce the computational complexity of SVM. This method proposes to divide the training data into N subsets, which has complexity $O(\frac{T_{\text{tr}}^3}{N^2})$ in training, and $O(\frac{T_{\text{tr}}T_{\text{test}}}{N})$ in test phase. The complexity of K-LPE is approximately $O(T_{\text{tr}}^2)$ in training, and $O(T_{\text{tr}}T_{\text{test}})$ in test phase. The complexity of K-NN is $O(T_{\text{tr}}T_{\text{test}})$, which is prohibitively costly when the number of data points is high. K-D Tree decreases the computational complexity of K-NN by introducing a tree structure. This method has complexity $O(T_{\text{tr}}\log(T_{\text{tr}}))$ in training and $O(T_{\text{test}}\log(T_{\text{tr}}))$ in test phase, which is still high for large number of data points. On the other hand, the complexity of our detector is only $O(DTk)$ (T: data length, D: depth of the partitioning tree, k: number of g values in the bandwidth set). Moreover, our algorithm is truly online without separate training or test phases, where increments are straightforward with new data; however, increments in these other competing ones are computationally demanding.

B. Real Datasets

In this part, we present the ROC performances of the compared algorithms on several real datasets in Table I that are widely used in anomaly detection literature. We first perform an experiment on the Occupancy dataset [43]. This dataset consists of 10808 data points where the labels correspond to occupied (normal) and unoccupied (anomalous) room states.

As seen in Fig. 2b, our technique AD-HKDE achieves the highest AUC score, which indicates that the intrinsic covariance structure within this dataset heavily requires local as well as temporal bandwidth adaptation, therefore, the superiority of our algorithm follows. In all of our experiments including the Occupancy dataset and the others, our algorithm AD-HKDE is generally highly superior in relatively smaller false alarm regions (except a few cases). Small false alarm rate requires the estimation of low density, which is in turn largely susceptible to insufficient data in corresponding low density regions. This necessitates to adjust the bandwidths or the neighborhood sizes intelligently in those regions of the support. Our algorithm elegantly addresses this need which introduces its superiority.

TABLE I: PROPERTIES OF THE REAL DATASETS USED IN THE EXPERIMENTS

Datasets \ Properties	Number of features	Total number of instances	Number of normal instances	Number of anomalous instances
Occupancy	5	10808	8107	2701
Breast Wisconsin	30	569	357	212
Pen digits	16	6870	6714	156
Susy	18	10^6	54×10^4	46×10^4
Thyroid	6	3772	3679	93
Pima	8	768	500	268
Liver	6	345	200	145
Mammography	6	11183	10923	260

In addition, we plot the temporal variations of the bandwidth probabilities/weights of our algorithm in Fig. 3a to show the bandwidth adaptation power of our method. As an example, we specify the bandwidth set as $\mathbb{G} = \{0.01, 0.02, 0.04, 0.08\}$ for the root node θ_r, and apply our algorithm on the Occupancy dataset. Since there are four possible bandwidths, initial probabilities are $\alpha_0^{\theta_k}(g) = \frac{1}{4}, \forall g \in \mathbb{G}$. Then, as seen in Fig. 3a, our algorithm asymptotically chooses $g = 0.08$ (the smallest bandwidth) since its probability converges to one while the others converge to zero[7]. This shows the efficacy of our algorithm since KDEs generally achieve higher performance with small bandwidths when the number of instances increases.

In Fig. 3b, we provide the ROC of our algorithm and its static version with fixed bandwidths, i.e., with singleton $\mathbb{G}$'s at the nodes. The number of data instances is low at the beginning and then increases since the algorithms operate in the online framework. Therefore, KDEs with smaller bandwidths (larger g) achieve higher bandwidth probabilities in Fig. 3a, towards the end of the process. However, KDEs with fixed smaller bandwidths get detrimentally affected by the scarce data in the beginning in terms of the detection performance, which results in an overall lower ROC performance as in Fig. 3b. For this contrast between Fig. 3a and Fig. 3b, we point out that an improvement in the density estimation does not directly translate to an improvement in the ROC performance, as the ranking of point density estimates matters more in our detectors compared to raw density estimates. Also, Fig. 3a shows the dynamic bandwidth probabilities with respect to the changing data size, whereas Fig. 3b demonstrates the overall AUC performance. Namely, a relatively larger bandwidth can still yield a better AUC in overall, although a smaller one is expected to perform better temporally locally toward the end of processing. Finally, our algorithm outperforms the static versions and has the highest AUC since it dynamically adapts its bandwidths in accordance with the available data size.

We also apply the algorithms on Breast Cancer Wisconsin dataset [44]. As in Table II, K-NN and K-D Tree are the best

[7]As we define the bandwidth parameter as $g = \frac{1}{2\delta^2}$, $g = 0.08$ corresponds to the actual bandwidth $\delta = 2.5$.

TABLE II: AREA UNDER CURVE (AUC) SCORES OF THE ALGORITHMS FOR THE OCCUPANCY, BREAST WISCONSIN, PEN DIGITS, SUSY, THYROID, PIMA, BUPA LIVER AND MAMMOGRAPHY DATASETS.

Datasets \ Algorithms	SVM	I-SVM	K-NN	K-D Tree	Online osPCA	K-LPE	FOGD	K-KDE	KDE-AB	AD-HKDE
Occupancy	0.9828	0.9750	0.9854	0.9854	0.9292	0.9708	0.9490	0.9368	0.9531	**0.9907**
Breast Wisconsin	0.9483	0.9480	**0.9778**	**0.9778**	0.7248	0.9408	0.9555	0.8363	0.9083	0.9672
Pen digits	0.9362	0.9334	0.9858	0.9858	0.8951	0.9902	0.9503	0.9145	0.9769	**0.9913**
Susy	0.6821	0.6343	0.8569	0.8569	0.5747	0.7651	0.8474	0.6997	0.6928	**0.9078**
Thyroid	0.8869	0.8747	0.9208	0.9208	0.8772	0.9286	0.8937	0.7431	0.8734	**0.9401**
Pima	0.6694	0.6694	0.7690	0.7690	0.6272	0.7032	0.6309	0.6473	0.6552	**0.7932**
Liver	0.5950	0.5950	0.7062	0.7062	0.5631	0.6923	0.6733	0.7013	0.5756	**0.7436**
Mammography	0.8337	0.8295	0.8676	0.8676	0.7667	0.8709	0.8279	0.7058	0.8568	**0.9115**

performing ones in this case (K-D Tree is K-NN with lower computational complexity), and our method AD-HKDE is the second best (the other methods are strongly outperformed). The reason is that the data size is not large enough to perfectly learn the bandwidths in all regions across time. Namely, in our method, we require to learn more parameters (compared to K-NN and K-D Tree), and hence demand more instances. Even when the data size is not sufficient as in this case, the difference is so small as seen in Table II. Also, note that our method works sequentially in the truly online setting with no separate training or test phases or even cross validation, while K-NN and K-D Tree need training data making them impractical in our processing setting. In our targeted data streaming applications, the data size is typically not an issue.

Moreover, we perform an experiment on pen digits dataset [44] that is composed of handwritten digits from 0 to 9. Here, 0 digits are considered as anomalous, and digits from 1 to 9 with equal number are considered as normal. This dataset consists of 10 digits, which can also be considered as the mixture of 10 distributions. Since our technique AD-HKDE is able to localize these components piecewise in its density estimation with adaptive bandwidths, we obtain a decent performance on this dataset with better AUC than all the others as can be seen in Table II.

We also evaluate the AUC scores of the algorithms on Thyroid [44], Pima [44], Mammography and Liver [44] datasets, whose properties are given in Table I. As seen in Table II, our algorithm AD-HKDE outperforms the other methods in all cases with its high modeling capability. Particularly for Thyroid and Mammography datasets, our method provides significantly higher AUC scores than the compared algorithms. Since the number of instances are greater in these datasets, our algorithm AD-HKDE learns the data structure more, which results in higher detection performance.

Finally, we perform an experiment on Susy dataset [45] with 10^6 instances. As in Table II, our algorithm AD-HKDE has the highest AUC score among all methods. In this case, we clearly observe the advantage of partitioning since our method AD-HKDE also significantly outperforms the other kernel based methods where FOGD uses fixed bandwidth and K-KDE uses point-wise variable bandwidth based on the nearest neighbor distance. The number of data points being large allows our algorithm to perfectly learn the local structure of the dataset and optimal partitioning with corresponding bandwidths.

VII. CONCLUSION

We introduced an online unsupervised anomaly detection algorithm for sequential data. The probability density of the observed instance is first estimated based on the past observations. Our density estimation can be used even if the underlying distribution is of any complex form since we do not have any parametric shape assumption. After the density for a given data instance is estimated, an anomaly is declared if the estimated density is sufficiently low following a comparison to a data adaptive threshold. Our density estimation is a mixture of experts approach based on an ensemble of hierarchically organized kernel density estimators (KDEs). The proposed algorithm guarantees to sequentially achieve the best likelihood performance within the introduced large ensemble represented by a binary partitioning tree. This mathematical performance guarantee implies that our method essentially exploits fitting kernels of bandwidths that are optimally varying both in space and time in the KDE technique. Namely, our method learns the optimal partitioning of the observation space with KDEs trained separately in partition regions. Such partition region KDEs are even fitted together with spatially and temporally optimal kernel bandwidths that can vary across partition regions and time in accordance with the data manifold and the size of the available data. In this work, we use a fixed bandwidth set where the algorithm converges to the optimal one. However, the performance can be improved by using a dynamic set such that one can continuously replace the obsolete candidate bandwidths with new larger ones (Note that our bandwidth parameter g is inversely proportional to the actual kernel bandwidth that should continuously decrease as more data become available) by keeping the cardinality fixed, which we consider as a future work. Consequently, in our extensive experiments, we observe significant performance improvements in anomaly detection by our method over various artificial and real datasets in comparison to the state-of-the-art techniques. The end-to-end processing in our framework is truly online with computational complexity $O(TDk)$ that is only linear in the tree depth D, the length T of the data and the number k of bandwidths in the bandwidth set. Hence, our technique is appropriate for processing at extremely fast rates with decent anomaly detection performance.

REFERENCES

[1] A. Dal Pozzolo, G. Boracchi, O. Caelen, C. Alippi, and G. Bontempi, "Credit card fraud detection: A realistic modeling and a novel learning strategy," *IEEE Transactions on Neural Networks and Learning Systems*, vol. 29, no. 8, pp. 3784–3797, Aug 2018.

[2] L. Maddalena and A. Petrosino, "Stopped object detection by learning foreground model in videos," *IEEE Transactions on Neural Networks and Learning Systems*, vol. 24, no. 5, pp. 723–735, May 2013.

[3] H. Ferdowsi, S. Jagannathan, and M. Zawodniok, "An online outlier identification and removal scheme for improving fault detection performance," *IEEE Transactions on Neural Networks and Learning Systems*, vol. 25, no. 5, pp. 908–919, May 2014.

[4] X. Ding, Y. Li, A. Belatreche, and L. P. Maguire, "Novelty detection using level set methods," *IEEE Transactions on Neural Networks and Learning Systems*, vol. 26, no. 3, pp. 576–588, March 2015.

[5] H. Ozkan, O. S. Pelvan, and S. S. Kozat, "Data imputation through the identification of local anomalies," *IEEE Transactions on Neural Networks and Learning Systems*, vol. 26, no. 10, pp. 2381–2395, Oct 2015.

[6] V. Chandola, A. Banerjee, and V. Kumar, "Anomaly detection: A survey," *ACM Comput. Surv.*, vol. 41, no. 3, pp. 15:1–15:58, Jul. 2009.

[7] V. Saligrama and Z. Chen, "Video anomaly detection based on local statistical aggregates," in *2012 IEEE Conference on Computer Vision and Pattern Recognition*, June 2012, pp. 2112–2119.

[8] H. V. Poor, *An introduction to signal detection and estimation*. Springer Science & Business Media, 2013.

[9] M. Zhao and V. Saligrama, "Anomaly detection with score functions based on nearest neighbor graphs," in *Advances in Neural Information Processing Systems 22*, Y. Bengio, D. Schuurmans, J. D. Lafferty, C. K. I. Williams, and A. Culotta, Eds. Curran Associates, Inc., 2009, pp. 2250–2258.

[10] C. D. Scott and R. D. Nowak, "Learning minimum volume sets," *Journal of Machine Learning Research*, vol. 7, no. Apr, pp. 665–704, 2006.

[11] E. Hazan, A. Agarwal, and S. Kale, "Logarithmic regret algorithms for online convex optimization," *Machine Learning*, vol. 69, no. 2, pp. 169–192, Dec 2007.

[12] B. W. Silverman, *Density estimation for statistics and data analysis*. CRC press, 1986, vol. 26.

[13] S. E. Yuksel, J. N. Wilson, and P. D. Gader, "Twenty years of mixture of experts," *IEEE Transactions on Neural Networks and Learning Systems*, vol. 23, no. 8, Aug 2012.

[14] M. Kerpicci, S. S. Kozat, and H. Ozkan, "Sequential anomaly detection using nonparametric density estimators," in *2019 27th Signal Processing and Communications Applications Conference (SIU)*, 2019, pp. 1–4.

[15] Z. Zivkovic and F. Van Der Heijden, "Efficient adaptive density estimation per image pixel for the task of background subtraction," *Pattern recognition letters*, vol. 27, no. 7, pp. 773–780, 2006.

[16] W. K. Härdle, M. Müller, S. Sperlich, and A. Werwatz, *Nonparametric and semiparametric models*. Springer Science & Business Media, 2012.

[17] M. C. Jones, J. S. Marron, and S. J. Sheather, "A brief survey of bandwidth selection for density estimation," *Journal of the American Statistical Association*, vol. 91, no. 433, pp. 401–407, 1996.

[18] G. R. Terrell and D. W. Scott, "Variable kernel density estimation," *The Annals of Statistics*, pp. 1236–1265, 1992.

[19] S. J. Sheather and M. C. Jones, "A reliable data-based bandwidth selection method for kernel density estimation," *Journal of the Royal Statistical Society. Series B (Methodological)*, vol. 53, no. 3, 1991.

[20] B. U. Park and J. S. Marron, "Comparison of data-driven bandwidth selectors," *Journal of the American Statistical Association*, vol. 85, no. 409, pp. 66–72, 1990.

[21] M. Kristan, D. Skočaj, and A. Leonardis, "Online kernel density estimation for interactive learning," *Image and Vision Computing*, vol. 28, no. 7, pp. 1106–1116, 2010.

[22] M. Kristan, A. Leonardis, and D. Skočaj, "Multivariate online kernel density estimation with gaussian kernels," *Pattern Recognition*, vol. 44, no. 10, pp. 2630 – 2642, 2011.

[23] J. Ferreira, D. M. de Matos, and R. Ribeiro, "Fast and extensible online multivariate kernel density estimation," *CoRR*, vol. abs/1606.02608, 2016.

[24] B. Schölkopf, J. C. Platt, J. Shawe-Taylor, A. J. Smola, and R. C. Williamson, "Estimating the support of a high-dimensional distribution," *Neural computation*, vol. 13, no. 7, pp. 1443–1471, 2001.

[25] B. V. Dasarathy, *Nearest Neighbor (NN) Norms: NN Pattern Classification Techniques*. Los Alamitos, CA: IEEE Computer Society Press, 1991.

[26] K. Zhang, M. Hutter, and H. Jin, "A new local distance-based outlier detection approach for scattered real-world data," in *Pacific-Asia Conference on Knowledge Discovery and Data Mining*. Springer, 2009.

[27] H. Ozkan, F. Ozkan, and S. S. Kozat, "Online anomaly detection under markov statistics with controllable type-i error," *IEEE Transactions on Signal Processing*, vol. 64, no. 6, pp. 1435–1445, March 2016.

[28] M. Raginsky, R. M. Willett, C. Horn, J. Silva, and R. F. Marcia, "Sequential anomaly detection in the presence of noise and limited feedback," *IEEE Transactions on Information Theory*, vol. 58, no. 8, pp. 5544–5562, Aug 2012.

[29] M. Xie, J. Hu, S. Han, and H. H. Chen, "Scalable hypergrid k-nn-based online anomaly detection in wireless sensor networks," *IEEE Transactions on Parallel and Distributed Systems*, vol. 24, no. 8, pp. 1661–1670, Aug 2013.

[30] J. Wang, P. Zhao, and S. C. H. Hoi, "Cost-sensitive online classification," *IEEE Transactions on Knowledge and Data Engineering*, vol. 26, no. 10, Oct 2014.

[31] M. Filippone and G. Sanguinetti, "Information theoretic novelty detection," *Pattern Recognition*, vol. 43, no. 3, pp. 805 – 814, 2010.

[32] F. Camci and R. B. Chinnam, "General support vector representation machine for one-class classification of non-stationary classes," *Pattern Recognition*, vol. 41, no. 10, pp. 3021 – 3034, 2008.

[33] A. Rahimi and B. Recht, "Random features for large-scale kernel machines," in *Advances in neural information processing systems*, 2008, pp. 1177–1184.

[34] J. Lu, S. C. Hoi, J. Wang, P. Zhao, and Z.-Y. Liu, "Large scale online kernel learning," *The Journal of Machine Learning Research*, vol. 17, no. 1, pp. 1613–1655, 2016.

[35] S. S. Kozat, A. C. Singer, and G. C. Zeitler, "Universal piecewise linear prediction via context trees," *IEEE Transactions on Signal Processing*, vol. 55, no. 7, July 2007.

[36] F. M. J. Willems, Y. M. Shtarkov, and T. J. Tjalkens, "The context-tree weighting method: basic properties," *IEEE Transactions on Information Theory*, vol. 41, no. 3, pp. 653–664, May 1995.

[37] T. Dasu, S. Krishnan, S. Venkatasubramanian, and K. Yi, "An information-theoretic approach to detecting changes in multi-dimensional data streams," in *In Proc. Symp. on the Interface of Statistics, Computing Science, and Applications*, 2006.

[38] A. Zimek, R. J. Campello, and J. Sander, "Ensembles for unsupervised outlier detection: Challenges and research questions a position paper," *SIGKDD Explor. Newsl.*, vol. 15, no. 1, pp. 11–22, Mar. 2014.

[39] G. Boracchi, D. Carrera, C. Cervellera, and D. Maccio, "Quanttree: Histograms for change detection in multivariate data streams," in *International Conference on Machine Learning*, 2018, pp. 638–647.

[40] N. Cesa-Bianchi and G. Lugosi, *Prediction, learning, and games*. Cambridge university press, 2006.

[41] H. Ozkan, N. D. Vanli, and S. S. Kozat, "Online classification via self-organizing space partitioning," *IEEE Transactions on Signal Processing*, vol. 64, no. 15, Aug 2016.

[42] K. Gokcesu and S. S. Kozat, "Online anomaly detection with minimax optimal density estimation in nonstationary environments," *IEEE Transactions on Signal Processing*, vol. 66, no. 5, March 2018.

[43] L. M. Candanedo and V. Feldheim, "Accurate occupancy detection of an office room from light, temperature, humidity and co2 measurements using statistical learning models," *Energy and Buildings*, vol. 112, 2016.

[44] M. Lichman, "UCI machine learning repository," 2013.

[45] P. Baldi, P. Sadowski, and D. Whiteson, "Searching for exotic particles in high-energy physics with deep learning," *Nature communications*, vol. 5, p. 4308, 2014.

[46] N. A. Syed, S. Huan, L. Kah, and K. Sung, "Incremental learning with support vector machines," 1999.

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

[48] Y. Lee, Y. Yeh, and Y. F. Wang, "Anomaly detection via online oversampling principal component analysis," *IEEE Transactions on Knowledge and Data Engineering*, vol. 25, no. 7, pp. 1460–1470, July 2013.

[49] L. J. Latecki, A. Lazarevic, and D. Pokrajac, "Outlier detection with kernel density functions," in *Machine Learning and Data Mining in Pattern Recognition*, P. Perner, Ed. Berlin, Heidelberg: Springer Berlin Heidelberg, 2007.

[50] C.-C. Chang and C.-J. Lin, "Libsvm: a library for support vector machines," *ACM transactions on intelligent systems and technology (TIST)*, vol. 2, no. 3, 2011.