Online Anomaly Detection With Nested Trees

Ibrahim Delibalta, Kaan Gokcesu, Mustafa Simsek, Lemi Baruh, and Suleyman S. Kozat, *Senior Member, IEEE*

***Abstract*—We introduce an online anomaly detection algorithm that processes data in a sequential manner. At each time, the algorithm makes a new observation, produces a decision, and then adaptively updates all its parameters to enhance its performance. The algorithm mainly works in an unsupervised manner since in most real-life applications labeling the data is costly. Even so, whenever there is a feedback, the algorithm uses it for better adaptation. The algorithm has two stages. In the first stage, it constructs a score function similar to a probability density function to model the underlying nominal distribution (if there is one) or to fit to the observed data. In the second state, this score function is used to evaluate the newly observed data to provide the final decision. The decision is given after the well-known thresholding. We construct the score using a highly versatile and completely adaptive nested decision tree. Nested soft decision trees are used to partition the observation space in a hierarchical manner. We adaptively optimize every component of the tree, i.e., decision regions and probabilistic models at each node as well as the overall structure, based on the sequential performance. This extensive in-time adaptation provides strong modeling capabilities; however, it may cause overfitting. To mitigate the overfitting issues, we first use the intermediate nodes of the tree to produce several subtrees, which constitute all the models from coarser to full extend, and then adaptively combine them. By using a real-life dataset, we show that our algorithm significantly outperforms the state of the art.**

***Index Terms*—Intrusion detection, semisupervised learning, statistical learning, tree data structures.**

I. Introduction

We INTRODUCE an online algorithm for anomaly detection [1]–[3] that works on sequentially observed data. At each time, the algorithm decides whether the newly observed data are anomalous or not, and then updates all its internal parameters. We mainly work in an unsupervised manner since in most real-life applications labeling the data is usually impractical [2]. Nevertheless, if such labeling is present, we use this information to improve adaptation. The algorithm has two stages. In the first stage, we sequentially assign a score (probability) to the observed data based on previous observations. Based on this score, we decide whether the newly observed data are anomalous or not. This decision is formed by comparing this score with a threshold [1], [2]. To sequentially assign these scores, we use highly versatile nested soft decision trees [4]–[6], where we adaptively optimize every component of the tree including the decision regions, probabilistic models at each node as well as the overall structure based on the sequential performance [4].

Two-stage anomaly detection methods, especially in unsupervised and/or adversarial settings, are extensively studied in the literature [2], [7], [8]. Although there exist several nonparametric approaches to model the nominal distribution, especially in adversarial settings [1], [9], [10], parametric models offer significant advantages such as quick convergence and high accuracy [9]. However, the parametric models suffer enormously if the assumed model does not match to the underlying true model (if such a true model exists) or if it is not rich enough to accurately capture the salient nature of the data [2]. Even if the assumed model correctly fits to a certain extent, we may still face underfitting or overfitting issues since real-life environments are usually highly nonstationary.

To this end, we first introduce a highly adaptive and efficient decision tree, which softly partitions the observation space. To boost modeling capabilities, we assign to each terminal leaf node a probability density function (pdf) from an exponential family of distributions, where parameters of these pdfs are sequentially learned. The boundaries of the regions assigned to each leaf are soft such that they are also updated based on the performance. In this form, the tree structure is similar to self-organizing maps (SOM)s or Gaussian mixture models (GMMs) [11], [12], where learning the partitions (or boundaries) corresponds to learning the *a priori* weights of the Gaussian pdfs in the GMMs (or SOMs). It is well known that the mixture models provide high modeling power [1], [2], [13]; however, they may overfit due to excessive number of leaves, i.e., Gaussians, in the mixture. Hence, to avoid overfitting or committing to a fixed decision tree, we go one step further and use all the nodes of the tree in addition to the leaf nodes such that each node is assigned to a particular region with its own pdf. This structure effectively constructs several subtrees with different depths on the original tree, which are then adaptively combined to maximize the overall performance. Since we adaptively merge both coarser and finer models, our algorithm avoids overfitting issues while preserving the modeling power [5].

II. Anomaly Detection Framework

Here[1], we sequentially receive $\{x_t\}_{t\geq 1}$, where $x_t \in \mathbb{R}^m$, and seek to find whether the received data are anomalous or not at each time t. To produce the decision, we sequentially construct a pdf $p_t(\cdot)$ (or a scoring function to be rigorous) using $\{x_1, \ldots, x_{t-1}\}$ to model the underlying nominal distribution (or to fit to the observed data if no such nominal distribution exists). Then, at each time t, based on the constructed distribution $p_t(\cdot)$, we score x_t as $p_t(x_t)$ and produce our decision $\hat{d}_t$. We produce the final decision using thresholding [1] (where such

Manuscript received July 1, 2016; revised September 11, 2016 and October 17, 2016; accepted October 19, 2016. Date of publication November 1, 2016; date of current version November 28, 2016. This work was supported in part by the Turkish Academy of Sciences Outstanding Researcher Programme. The associate editor coordinating the review of this manuscript and approving it for publication was Dr. Antonio Paiva.

I. Delibalta and L. Baruh are with the Design, Technology and Society Program, Koc University, Istanbul 34450, Turkey (e-mail: idelibalta@ku.edu.tr; lbaruh@ku.edu.tr).

K. Gokcesu, M. Simsek, and S. S. Kozat are with the Electrical and Electronics Engineering Department, Bilkent University, Ankara 06800, Turkey (e-mail: gokcesu@ee.bilkent.edu.tr; mustafa.simsek@ee.bilkent.edu.tr; kozat@ee.bilkent.edu.tr).

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

Digital Object Identifier 10.1109/LSP.2016.2623773

[1]We represent vectors (matrices) by bold lower (upper) case letters. For a matrix A (or a vector a), A^T is the transpose and $\|a\|$ is the Euclidean norm. For notational simplicity, we work with real valued data. All vectors are column vectors.

1070-9908 © 2016 IEEE. Personal use is permitted, but republication/redistribution requires IEEE permission.
See http://www.ieee.org/publications_standards/publications/rights/index.html for more information.

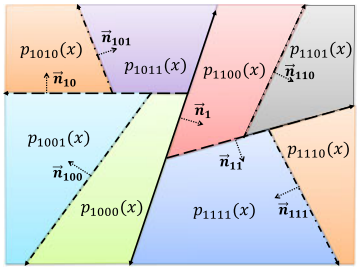


Fig. 1. Hard decision boundaries for a depth-3 decision tree.

an approach is optimal minimizing the type-1 error in certain settings [14]), i.e., if

$$p_t(x_t) \geq \tau_t \tag{1}$$

then $\hat{d}_t = 0$ (not anomalous), otherwise $\hat{d}_t = 1$ (anomalous) for some time-varying threshold τ_t. Then, this decision is compared with the correct result d_t if available, otherwise we work in an unsupervised manner [7].

To construct the sequential distribution, we use the decision trees. A decision tree is a hierarchical structure composed of both internal and terminal nodes, i.e., the leaf nodes. Unlike [5], [15], and [4], we do not require the tree to be complete. After we observe x_t, each node η produces its score as

$$f_\eta(x_t) = \begin{cases} p_\eta(x_t), & \text{if } \eta \text{ is a leaf} \\ f_{\eta r}(x_t), & \text{if } \sigma_\eta(x_t) \geq 0 \text{ (go to right child)} \\ f_{\eta l}(x_t), & \text{if } \sigma_\eta(x_t) < 0 \text{ (go to left child)} \end{cases} \tag{2}$$

where $\sigma_\eta(\cdot)$ is the function that determines which side of the hard decision boundary of the node η an observation belongs to, as shown in Fig. 1. In this letter, we use linear separating hyper planes for decision boundaries such that $\sigma_\eta(\cdot)$ is given as $\sigma_\eta(x_t) = n_\eta^T [x_t; 1]$, where n_η is the normal vector of the separating hyper plane and we extend x_t as $[x_t; 1]$ to include the bias term for a compact notation. Our approach is generic such that one can also use nonlinear separation boundaries; however, we use linear boundaries to avoid overfitting. Here, $f_{\eta l}(x_t)$ (or $f_{\eta r}(x_t)$) is the score of the left hand (or the right hand) child node. Each leaf node η is assigned a pdf from an exponential family of distributions as

$$p_\eta(x_t) = \exp(\theta_\eta^T x_t - G(\theta_\eta)) P_o(x_t)$$

where θ_η is from some convex set, $G(\theta_\eta)$ is sufficient statistics, and $P_o(x_t)$ is for normalization [11]. For each x_t, the final probability is given by

$$p_t(x_t) = f_1(x_t)$$

that is the score of the root node. Starting from the root node, we recursively move down the tree until we reach to one of the leaves to find this probability.

As the first extension to the hard decision tree, we use soft partitioning [4] similar to the SOM models and set

$$\sigma_\eta(x_t) = 1/\left[1 + \exp\left(n_\eta^T x_t\right)\right] \tag{3}$$

where we denote $[x_t; 1]$ as x_t with an abuse of notation. Then, for each node, we obtain

$$f_\eta(x_t) = \begin{cases} p_\eta(x_t), & \text{if } \eta \text{ is a leaf} \\ \sigma_\eta(x_t) f_{\eta l}(x_t) + (1 - \sigma_\eta(x_t)) f_{\eta r}(x_t), & \text{otherwise.} \end{cases}$$

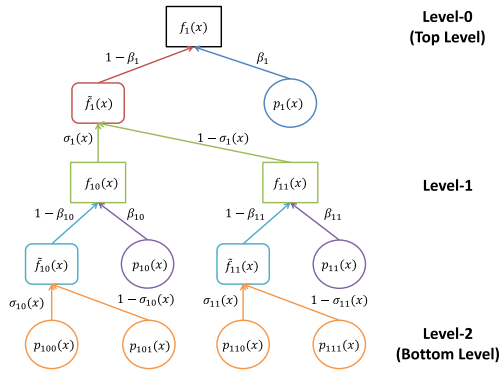


Fig. 2. Nested combination structure for a depth-2 decision tree.

For the soft decision tree, the calculation starts from the leaf nodes such that all the leaf nodes contribute to the final pdf, unlike a hard decision tree as shown in (2). To obtain the final score, we start from bottom of the tree and proceed to the top node, i.e., to the root node, as shown in Fig. 2.

As the second and final extension, we assign pdfs from exponential family distributions to all nodes of tree, including both the terminal and internal nodes. In the previous cases, either hard or soft, only the leaves of the tree, i.e., the finest and the most detailed structure of the tree, were used to partition the space of observations or assign scores. Here, by assigning pdf to the internal nodes, we also represent much coarser models or partitions of the observation space. After this assignment of pdfs, we define for each node

$$f_\eta(x_t) = \begin{cases} p_\eta(x_t), & \text{if } \eta \text{ is a leaf} \\ \beta_\eta p_\eta(x_t) + (1 - \beta_\eta) \times & \\ \left[\sigma_\eta(x_t) f_{\eta l}(x_t) + (1 - \sigma_\eta(x_t)) f_{\eta r}(x_t)\right], & \text{otherwise} \end{cases} \tag{4}$$

where $0 \leq \beta_\eta \leq 1$ for all η. Since, we use the stochastic gradient descent (SGD) for optimization [16], we reparametrize the mixture weight β_η as

$$\beta_\eta = 1/\left[1 + \exp\left(-\alpha_\eta\right)\right] \tag{5}$$

where $\alpha_\eta \in \mathbb{R}$, to satisfy $0 \leq \beta_\eta \leq 1$. The final probability is given by $p(x_t) = f_1(x_t)$. Here, after we observe x_t, the calculation of the final probability starts from the bottom of the tree, where not only each leaf but also all the internal nodes contribute to the final probability.

In this form, for a decision tree of depth d, we have $\eta = 2^{d+1} - 1$ nodes including both internal and terminal nodes. For each internal node, we have a tuple $\{\beta_\eta, \sigma_\eta, \theta_\eta\}$ (or equivalently $\{\alpha_\eta, n_\eta, \theta_\eta\}$), the mixture coefficient that merges a node's score with its children's scores, the soft partition parameter that is similar to the *a priori* weights assigned to each child and pdf parameters assigned to the node. For the terminal nodes, we only have one parameter $\{\theta_\eta\}$.

Remark 1: In [6] and [5], the combination weights are fixed in time and equal to $\beta_\eta = 1/2$ for all nodes η. In [15], these weights are again fixed in time; however, they are set to desired *a priori* values based on the user preference. In [4], in a regression framework, these weights are unconstrained, i.e., $\beta_\eta \in \mathrm{R}$, can even take nonpositive values and adapt in time to minimize the final regression error. Here, inspired from [12], and since we work with probabilities, we constrained these weights to the unit simplex, i.e., $\beta_\eta \in [0, 1]$.

Remark 2: Although the soft decision tree that only uses the leaf nodes, i.e., the finest models, has the highest modeling power, there is no guarantee that it would provide the best performance in applications involving online or sequential data. The modeling power comes with increase in the number of parameters that must be sequentially learned. Hence, when there are limited data or the data are highly nonstationary, coarser models, i.e., subtrees of the full tree, may perform better. By adaptively combining both the coarser and finer models, we retain the modeling power of the finest model while avoiding slow convergence and overfitting problems.

III. Online Algorithm

In this section, we sequentially train our algorithm. We have two cases. In the first case, the true label is not present and we decide the label of the data based on (1). If $\hat{d}_t = 0$, then the observation x_t can be used to update $p_t(\cdot)$. If $\hat{d}_t = 1$, then we discard it. In the second case, we have the true label d_t. If $d_t = 0$, then we naturally update $p_t(\cdot)$. If $d_t = 1$, we do not update $p_t(\cdot)$. When we have d_t, we also update the threshold.

When we update $p_t(\cdot)$, we first measure the performance of our sequential probability assignment using the most obvious loss measure [11] that is the negative log probability

$$l_t(x_t) = -\ln p_t(x_t). \tag{6}$$

To optimize and learn the system parameters, we use the stochastic gradient descent algorithm [16]. The SGD recursion provides deterministic performance bounds in sequential convex optimization problems [18]. The pdf estimation problem is convex under the loss in (6) when we have only one exponential distribution. However, due to the sigmoid nonlinearities in (3) and (5), the problem is not convex.

To use the SGD, we need to calculate the gradient of the final loss with respect to all parameters. We observe that the soft decision structure shown in Fig. 2 is similar to a neural network architecture, where the bottom of the tree corresponds to the input layer with $p_\eta(x_t)$ s as inputs, i.e., the input layer has 2^d neurons, and the output layer corresponds to the root of the tree, where the final output of the system is given by $p_t(x_t) = f_1(x_t)$. In this sense, β_η s correspond to gating functions and σ_η s correspond to combination weights on each layer [11]. Hence, to calculate the gradients at each level, we can use the well-known back-propagation algorithm [11], which is basically the chain rule. The back-propagation algorithm proceeds as follows. When x_t arrives, we start from the leaf nodes, i.e., from the input layer, and calculate all the terms, $\sigma_{\eta,t}$, $\beta_{\eta,t}$, $p_{\eta,t}(x_t)$, and $f_{\eta,t}(x_t)$. This is the "forward-propagation" [11]. In the back-propagation step, we start from the top (the root node) and calculate step by step the gradient until we reach to the bottom nodes (leaves). For any internal η including the root node, using the chain rule, we have

$$\nabla_{\theta_\eta} l_t(x_t) = \delta_{\eta,t}\beta_{\eta,t}p_\eta(x_t)\left(x_t - \nabla_{\theta_\eta} G(\theta_{\eta,t})\right)$$

$$\partial l_t(x_t)/\partial\alpha_\eta = \delta_{\eta,t}(1-\beta_{\eta,t})\beta_{\eta,t}\Big(p_\eta(x_t) - \tag{7}$$

$$[\sigma_\eta(x_t)f_{\eta l}(x_t) + (1-\sigma_\eta(x_t))f_{\eta r}(x_t)]\Big)$$

$$\nabla_{n_\eta} l_t(x_t) = \delta_{\eta,t}(1-\beta_{\eta,t})\times \tag{8}$$

$$(1-\sigma_{\eta,t}(x_t))\sigma_{\eta,t}(x_t)\left(f_{\eta l,t}(x_t) - f_{\eta r,t}(x_t)\right)x_t \tag{9}$$

where $\delta_{\eta,t} \triangleq \partial l_t(x_t)/\partial f_\eta(x_t)$. We calculate $\delta_{\eta,t}$ using the backpropagation. For the root node, by using (6), we obtain

$$\delta_{\eta,t} = -1/p_t(x_t). \tag{10}$$

After that starting from the top, we backpropagate to the lower nodes. For any internal node, we distinguish the left and right children. For node η that is the left child of $\tilde{\eta}$, we have

$$\delta_{\eta,t} = \partial l_t(x_t)/\partial f_{\tilde{\eta}}(x_t)\,(1-\beta_{\tilde{\eta},t})\sigma_{\tilde{\eta},t}(x_t). \tag{11}$$

Similarly for node η that is the right child of $\tilde{\eta}$, we have

$$\delta_{\eta,t} = \partial l_t(x_t)/\partial f_{\tilde{\eta}}(x_t)\,(1-\beta_{\tilde{\eta},t})(1-\sigma_{\tilde{\eta},t}(x_t)). \tag{12}$$

The recursion stops at the terminal leaf nodes. Then, we update the corresponding parameters using the SGD as

$$\theta_{\eta,t+1} = \theta_{\eta,t} - \mu_t\nabla_{\theta_\eta} l_t(x_t) \tag{13}$$

$$\alpha_{\eta,t+1} = \alpha_{\eta,t} - \mu_t\partial l_t(x_t)/\partial\alpha_\eta \tag{14}$$

$$n_{\eta,t+1} = n_{\eta,t} - \mu_t\nabla_{n_\eta} l_t(x_t) \tag{15}$$

for some learning rate μ_t.

When we have the feedback, we train the threshold using SGD. For loss, we use the square error, $(d_t - \hat{d}_t)^2$, and obtain

$$\tau_{t+1} = \tau_t - \mu_t(d_t - \hat{d}_t)\partial\hat{d}_t/\partial\tau. \tag{16}$$

Since $\hat{d}_t$ given in (1) is not differentiable, as widely performed in the signal processing literature [16], we use

$$\tilde{d}_t = 1/\left[1+\exp\left(-(\tau_t - p_t(x_t))\right)\right].$$

Hence, (16) yields

$$\tau_{t+1} = \tau_t - \mu_t(d_t - \tilde{d}_t)\tilde{d}_t(1-\tilde{d}_t). \tag{17}$$

This completes the full set of equations.

The complete algorithm is given in Algorithm 1.

IV. Experiments

We use the Istanbul stock exchange (ISE) [17] dataset for real data benchmark purposes, which is a time series of 536 samples with nine features. We normalize every dimension into $[-1, 1]$. Then, in random indexes, we artificially add 64 anomalous samples generated from a multivariate Gaussian process with batch mean and 16 times the batch covariance of the dataset [19]. We fed this time series to all the algorithms one at a time in various settings to obtain a performance comparison. We run online anomaly detection algorithms using hard, soft, nested decision trees, and the state-of-the-art methods, such as support vector data description (SVDD) [20], nearest neighbor (NN) data description [21], maximum likelihood (ML) [14] based and kernel density (KD) estimation [22] based anomaly detector. The algorithms are implemented with *libsvm* [23], *prtools* [24], *ddtools* [25].

Our algorithm, nested decision tree, combines the beliefs of internal and terminal nodes in the tree (both coarser and finer models). Soft decision trees only update its boundaries and terminal nodes (no internal node). Hard decision trees only update the terminal nodes. We set the learning rate $\mu = 1/\sqrt{t}$ to compensate the nonstationarity of the data [7]. We run a multivariate Gaussian density estimator in each node. Initial self-combination weights are $\beta_{\eta,1} = 0.5$ for the nested trees and the threshold is $\tau_1 = 1$. Initial boundaries are selected such that, the split at the first layer splits according to the first feature

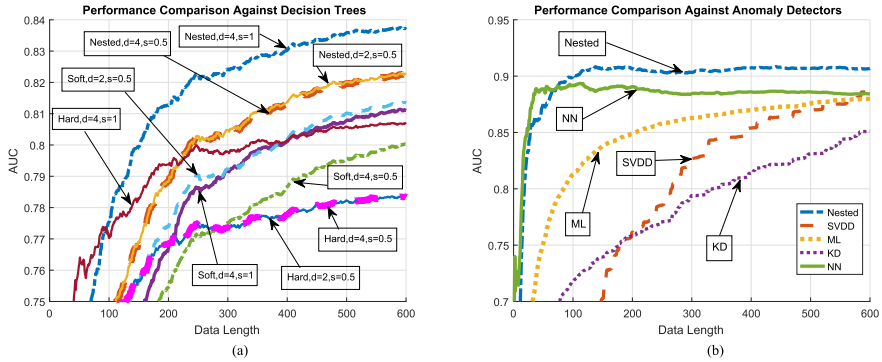


Fig. 3. (a) AUC over time performance of hard, soft, and nested decision trees at tree-depth and feedback-probability pairs $d = 2, \ \ s = 0.5$, $d = 4, \ \ s = 0.5$ and $d = 4, \ \ s = 1$ for ISE dataset [17] averaged over 100 trials. (b) AUC over time performance of nested decision tree with depth $d = 1$, SVDD, ML, KD estimation and NN data description with window size 100 for ISE dataset [17] averaged over 100 trials.

Algorithm 1: Online Anomaly Detection Algorithm.

Initialize tree and all parameters $\alpha_{\eta,1}$, $n_{\eta,1}$, τ_1
for $t = 1$ **to** ... **do**
  Receive observation, x_t
  for all nodes **do**
    Calculate $\sigma_{\eta,t}$, $\beta_{\eta,t}$, $f_\eta(x_t)$ according to (3), (5), (4)
  end for
  $p_t(x_t) = f_1(x_t)$
  $\hat{d}_t = \max(0, \text{sgn}\,(\tau_t - p_t(x_t)))$
  if ($d_t = 0$) **or** (d_t is **not** avaliable **and** $\hat{d}_t = 0$) **then**
    for all nodes **do**
      Calculate $\delta_{\eta,t}$ according to (10), (11), (12)
      Calculate $\nabla_{\theta_\eta} l_t(x_t)$, $\partial l_t(x_t)/\partial\alpha_\eta$, $\nabla_{n_\eta} l_t(x_t)$ according to (7), (8), (9)
      Update parameters $\theta_{\eta,t+1}$, $\alpha_{\eta,t+1}$, $n_{\eta,t+1}$ according to (13), (14), (15)
    end for
  end if
  if d_t is available **then**
    Update τ_t according to (17)
  end if
end for

(greater or less than zero), the second layer splits the second feature etc. All Gaussian density estimators are started from zero mean and identity covariance. SVDD, NN, ML, and KD use a sliding window of length 100. This length gives algorithms a window big enough to train fairly and small enough for computational feasibility. We have optimized the parameters of the algorithms ML, KD, and SVDD for each sliding window. ML algorithm inherently optimizes its parameters. We have optimized the bandwidth of the KD estimation with Silverman's rule [26]. For SVDD, optimizing the bandwidth of radial basis function (RBF) in each sliding window is computationally infeasible. Hence, we find the optimum parameter at the beginning, where, at each trial, we exponentially search the values in $[2^{-10}, 2^{10}]$ and select the parameter with the best performance. The selected bandwidth differs for each trial (different datasets) but has a mean of 0.2676 over 100 trials.

We use area under receiving operating characteristic (ROC) curve metric [27], [28] to compare the performances. We have sampled the ROC curve at multiple points by varying each methods' discrimination threshold [29]. This sampling of the ROC curve provides true positive rate (TPR)/false positive rate (FPR) pairs, to which we fit a piecewise linear function, which approximates the ROC curve. We use the area under this curve, i.e., area under curve (AUC) [30] to evaluate the performances . We sample the ROC curve by varying a threshold of the anomaly detector [29]. To sample the ROC curve in our dynamic threshold algorithm, we change the update rule and add a cost metric. We vary parameter α that is the ratio of the cost of false positive to false negative. The error (cost) is given by $\sqrt{\alpha}(d_t - \hat{d}_t)$ and $(d_t - \hat{d}_t)/\sqrt{\alpha}$ for mislabeled normal and anomalous data, respectively. Hence, for $\alpha > 1$ and $\alpha < 1$, the threshold will tend to decrease and increase, respectively. This behaviour samples the ROC curve at different points for different α.

We have illustrated the AUC [27], [28], [31] performances of the algorithms averaged over 100 independent trials in Fig. 3(a) and (b) (for each trial, the dataset with anomalies has been created independently). In Fig. 3(a), we have illustrated the AUC performances of nested, soft, and hard decision trees with varying tree depth ($d = 2$ and $d = 4$) and feedback probability environments ($s = 0.5$ and $s = 1$). As shown in Fig. 3(a), both hard and soft decision trees perform better with $d = 2$ trees, since they have less parameters to learn; hence, they converge faster. However, the depth selection is not an issue for nested decision trees, since they also use the internal nodes to improve performance. In Fig. 3(a), we also illustrate that higher feedback provides higher performance and feedback dependence changes across algorithms. Soft trees with $s = 0.5$ show comparable performance to hard trees with $s = 1$. Nested trees with $s = 0.5$ outperform soft trees with $s = 1$. Using nested trees mitigates overfitting and undertraining; hence, they outperform other combination structures. In Fig. 3(b), we have compared our algorithm against SVDD, ML, KD, and NN. We illustrate that KD has the slowest convergence since it is a nonparametric algorithm. SVDD has a rather slow start but quickly catches up with ML and NN. Nevertheless, nested decision trees outperform all of the algorithms significantly.

V. Conclusion

We introduced a highly versatile and effective online anomaly detection algorithm based on nested trees. Based on the sequential performance, we learn every component of the tree including decision regions, probabilistic models at each node as well as the overall structure. We mitigate overfitting issues by using all nodes of the tree to produce several subtrees from coarser models to the full extend.

REFERENCES

[1] H. Ozkan, F. Ozkan, and S. S. Kozat, "Online anomaly detection under Markov statistics with controllable type-I error," *IEEE Trans. Signal Process.*, vol. 64, no. 6, pp. 1435–1445, Mar. 2016.

[2] V. Chandola, A. Banerjee, and V. Kumar, "Anomaly detection: A survey," *ACM Comput. Surv.*, vol. 11, pp. 1–72, 2009.

[3] M. Kloft and P. Laskov, "Online anomaly detection under adversarial impact," in *Proc. Thirteenth Int. Conf. Artif. Intell. Statist.*, 2010, pp. 405–412.

[4] N. D. Vanli and S. S. Kozat, "A comprehensive approach to universal piecewise nonlinear regression based on trees," *IEEE Trans. Signal Process.*, vol. 62, no. 20, pp. 5471–5486, Oct. 2014.

[5] F. M. J. Willems, Y. M. Shtarkov, and T. J. Tjalkens, "The context-tree weighting method: Basic properties," *IEEE Trans. Inf. Theory*, vol. 41, no. 3, pp. 653–664, May 1995.

[6] S. S. Kozat, A. C. Singer, and G. C. Zeitler, "Universal piecewise linear prediction via context trees," *IEEE Trans. Signal Process.*, vol. 55, no. 7, pp. 3730–3745, Jul. 2007.

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

[8] C. Horn and R. M. Willett, "Online anomaly detection with expert system feedback in social networks," in *Proc. IEEE Int. Conf. Acoust., Speech, Signal Process.*, 2011, pp. 1936–1939.

[9] H. Ozkan, F. Ozkan, I. Delibalta, and S. S. Kozat, "Efficient NP tests for anomaly detection over birth-death type DTMCs," *J. Signal Process. Syst.*, vol. 1, no. 1, pp. 1–10, Jun. 2016.

[10] S. Mukherjee and V. Vapnik, "Support vector method for multivariate density estimation."

[11] R. O. Duda, P. E. Hart, and D. G. Stork, *Pattern Classification*. Hoboken, NJ, USA: Wiley, 2000.

[12] J. Arenas-Garcia, A. R. Figueiras-Vidal, and A. H. Sayed, "Mean-square performance of a convex combination of two adaptive filters," *IEEE Trans. Signal Process.*, vol. 54, no. 3, pp. 1078–1090, Mar. 2006.

[13] O. J. J. Michel, A. O. Hero, and A.-E. Badel, "Tree-structured nonlinear signal modeling and prediction," *IEEE Trans. Signal Process.*, vol. 47, no. 11, pp. 3027–3041, Nov. 1999.

[14] H. V. Poor, *An Introduction to Signal Detection and Estimation*. Berlin, Germany: Springer, 1994.

[15] J. Suzuki, "A CTW scheme for some FSM models," in *Proc. IEEE Int. Symp. Inf. Theory*, Whistler, BC, Canada, 1995, p. 389.

[16] A. H. Sayed, *Fundamentals of Adaptive Filtering*. Hoboken, NJ, USA: Wiley, 2003.

[17] O. Akbilgic, H. Bozdogan, and M. E. Balaban, "A novel hybrid RBF neural networks model as a forecaster," *Statist. Comput.*, vol. 24, no. 3, pp. 365–375, 2014. [Online]. Available: http://dx.doi.org/10.1007/s11222-013-9375-7

[18] K. Z. Azoury and M. K. Warmuth, "Relative loss bounds for on-line density estimation with the exponential family of distributions," *Mach. Learn.*, vol. 43, pp. 211–246, 2001.

[19] R. G. Steel *et al.*, *Principles and Procedures of Statistics.* New York, NY, USA: McGraw-Hill, 1960.

[20] D. M. Tax and R. P. Duin, "Support vector data description," *Mach. Learn.*, vol. 54, no. 1, pp. 45–66, 2004.

[21] G. G. Cabral, A. L. Oliveira, and C. B. Cahú, "Combining nearest neighbor data description and structural risk minimization for one-class classification," *Neural Comput. Appl.*, vol. 18, no. 2, pp. 175–183, 2009.

[22] J. S. Simonoff, *Smoothing Methods in Statistics*. Berlin, Germany: Springer, 2012.

[23] C.-C. Chang and C.-J. Lin, "LIBSVM: A library for support vector machines," *ACM Trans. Intell. Syst. Technol.*, vol. 2, pp. 27–1–27–27, 2011. [Online]. Available: http://www.csie.ntu.edu.tw/cjlin/libsvm

[24] R. Duin *et al.*, "PR-Tools4.1, a MATLAB toolbox for pattern recognition," 2007. [Online]. Available: http://prtools.org

[25] D. Tax, "Ddtools, the data description toolbox for MATLAB," Version 2.1.2, Jun. 2015.

[26] B. W. Silverman, *Density Estimation for Statistics and Data Analysis*. Boca Raton, FL, USA: CRC press, 1986.

[27] A. P. Bradley, "The use of the area under the ROC curve in the evaluation of machine learning algorithms," *Pattern Recognit.*, vol. 30, no. 7, pp. 1145–1159, 1997.

[28] T. Fawcett, "An introduction to ROC analysis," *Pattern Recognit. Lett.*, vol. 27, no. 8, pp. 861–874, 2006.

[29] X. Ding, Y. Li, A. Belatreche, and L. P. Maguire, "An experimental evaluation of novelty detection methods," *Neurocomputing*, vol. 135, pp. 313–327, 2014.

[30] D. M. W. Powers, "Evaluation: From precision, recall and F-measure to ROC, informedness, markedness & correlation," *J. Mach. Learn. Technol.*, vol. 2, no. 1, pp. 37–63, 2011.

[31] M. Goldstein and S. Uchida, "A comparative evaluation of unsupervised anomaly detection algorithms for multivariate data," *PloS One*, vol. 11, no. 4, , 2016, Art. no. e0152173.