Online Neyman-Pearson Classification with Hierarchically Represented Models

Basarbatu Can, S. Ozgun Pelvan, Huseyin Ozkan, *Member, IEEE*

***Abstract*—We consider the statistical anomaly detection problem with regard to false alarm rate (or false positive rate, FPR) controllability, nonlinear modeling and computational efficiency for real-time processing. A decision theoretical solution can be formulated as Neyman-Pearson (NP) hypothesis testing (binary classification: anomaly/nominal). In this framework, we propose an ensemble NP classifier (Tree OLNP) that is based on a binary partitioning tree. Tree OLNP generates an ensemble of sample space partitions. Each partition corresponds to an online piecewise linear (hence nonlinear) expert classifier as a union of online linear NP classifiers (union of OLNPs). While maintaining a precise control over the FPR, Tree OLNP generates its overall prediction as a performance driven and time varying weighted combination of the experts. This provides a dynamical nonlinear modeling power in the sense that simpler (more powerful) experts receive larger weights early (late) in the data stream, which manages the bias-variance trade-off and mitigates overfitting/underfitting issues. We mathematically prove that, for any stream, Tree OLNP asymptotically performs at least as well as of the best expert in terms of the NP performance with a regret diminishing in the order $O(1/\sqrt{t})$ (t : data size). Our algorithm is computationally highly efficient since it is online and its complexity scales linearly with respect to both the data size and tree depth, and scales twice-logarithmic with respect to the number of experts. We experimentally show that Tree OLNP strongly outperforms the state-of-the-art alternative techniques. The code is available for reproducibility at https://github.com/basarbatucan/Tree-OLNP.**

***Index Terms*—Anomaly detection, Neyman Pearson, classification, false alarm rate, online, decision tree, partitioning.**

I. Introduction

NEYMAN-PEARSON (NP) classification framework [1] is widely used in areas such as medical diagnosis [2]–[4], change point detection [5] and video anomaly detection [6], [7], where false alarm rate controllability is essential and requires asymmetrical cost assignment to the type I (false alarm) and type II (miss) errors. In cyber fraud detection [8], [9] for instance, missing a suspicious activity of a client (miss) might have severe security consequences whereas predicting a regular activity as suspicious (false alarm) leads to a tiring unnecessary measure. The CFAR (constant false alarm rate) detection in radar applications [10], [11] is another example among numerous others developed in the past decades [1]. The NP framework appears as a viable approach in such situations as it minimizes one of the error rates (typically the miss rate) under a user-specified upperbound constraint on the other (typically the false alarm rate). The classification solution in this framework is then the minimization of the Bayes risk (cf. the NP lemma [12]) with unknown asymmetrical costs satisfying the error constraint.

In recent years, NP frameworks have gained significant attention for their application in Large Language Models (LLMs), particularly in distinguishing between text generated by LLMs and text produced by humans. This capability is crucial for tasks such as detecting misinformation or fabricated content generated by LLMs, as well as for managing copyright issues. One notable contribution in this area is the robust watermarking scheme introduced by [13], which leverages the NP framework to control error rates effectively. Similarly, [14] demonstrates that the NP framework can be used to establish a lower bound on the number of samples required to reliably differentiate between AI-generated and human-generated text. Furthermore, the NP framework has been applied to enable LLMs to forget specific training samples by eliminating their influence on the model, as explored in [15].

In this paper, we introduce a novel online NP classifier for false alarm controllability matching to the user-specified level, while addressing the two important practical needs of fast stream data applications: Dynamical modeling power that is tuned to the learnibility that the available data allows at a time and computational scalability for real time processing. In particular, our method constructs a rich hierarchical ensemble of expert algorithms via a binary tree partitioning of the observation space, and obtains a randomized mixture of experts [16], [17] as the proposed NP classification algorithm. We mathematically show that the proposed algorithm asymptotically achieves the NP performance of the best expert for any stream.

We continue with the related work in Section II below and then introduce the problem description and our novel contributions in Section III which is followed by our solution in Section IV. The performance evaluations are presented in Section V. We conclude in Section VI.

II. Related Work

Neyman-Pearson (NP) classification is a statistical framework that studies problems with asymmetrical type I (predicting non-target as target, false positive / alarm) and type II (predicting target as non-target, false negative or miss) error costs [1], with the main purpose of controlling the false alarm rate by guaranteeing that it does not exceed a certain level

Basarbatu Can (corresponding author) works as a Sr. Data Scientist in ABN Amro Bank N.V., Amsterdam, Netherlands (e-mail: basarbatu.can@nl.abnamro.com). Soner O. Pelvan and Huseyin Ozkan are with Electronics Engineering, Faculty of Engineering and Natural Sciences at Sabanci University, Istanbul, Turkey (e-mail: [sozgun, huseyin.ozkan]@sabanciuniv.edu).

This work was supported by The Scientific and Technological Research Council of Turkey (TUBITAK) under Contract 118E268.

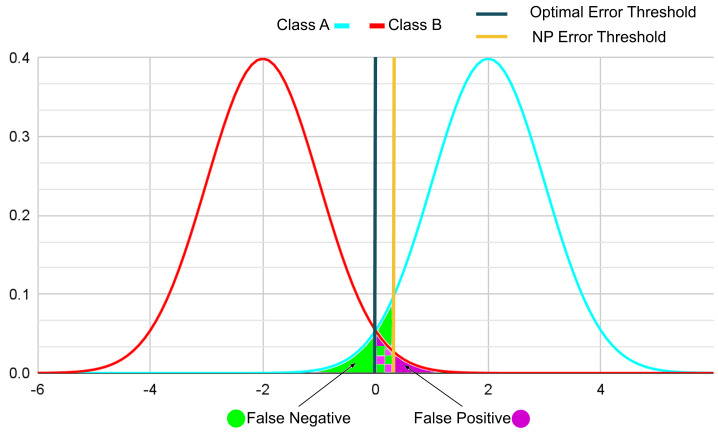

Figure 1. In this figure, we illustrate the difference in threshold selection between a conventional error-minimizing classifier and an NP classifier. The figure depicts two classes, Class A and Class B, as Gaussian distributions with differing means. While the optimal error threshold designed to minimize both false negatives and positives may not achieve the required false positive rate (FPR), attaining a specific FPR necessitates that the user iteratively search for the appropriate threshold. The NP methodology distinctively addresses this challenge by integrating the desired false alarm rate into the design process, thereby ensuring that the classifier complies with the FPR constraint while optimizing overall performance.

provided by the user. In the data driven cost sensitive learning, the asymmetry is directly incorporated into the empirical error minimization [18], [19], where the drawback is that the correspondence between the desired false alarm rate and the right cost structure is typically unknown. Another approach is to use plug-in methods to estimate the class specific densities and apply the log likelihood ratio test with the right threshold that matches the desired false alarm rate [20], [21]. We have demonstrated this approach using a theoretical scenario shown in Figure 1, where two classes are each represented by Gaussian distributions with different means. The figure highlights the distinction between threshold selection in a typical error-minimizing classifier and an NP classifier. The same drawback is still in effect since the issue of unknown cost structure in the cost sensitive learning translates to the issue of unknown threshold in the plug-in methods. Likewise, the neural network in [20] estimates the densities and leaves the threshold as a parameter to manual tuning. In [21], authors introduce an umbrella algorithm that can handle any given batch classifier in the NP framework. Their method succeeds to remove the additional work required for selecting the suitable threshold. However, this umbrella algorithm [21] as well as the plug-in studies [20], [21] are computationally prohibitive and thus experience scalability issues when the data is voluminous or streaming fast due to the multiple passes in their batch processing.

Similar to these prominent examples, the vast majority of the existing NP methods in the literature either fail at addressing the false alarm rate controllability without manual tuning or do not scale to large amount fast streaming data. Nevertheless, the online classifier in [22] addresses the both through two stochastic gradient descent (SGD) updates for a Lagrangian minimax resulting from the constrained optimization of the NP classification. The first update is for the classifier parameters, and the second is for estimating the asymmetrical costs. The outcome is an online linear NP classifier (OLNP) that is scalable with constant processing complexity per instance, but it is incapable of modeling nonlinear class separations. The authors [22] has left the handling of nonlinear separations to a future study. As a follow up to that within the formulation of OLNP, we recently introduced an NP classifier (NP-NN) in [23] based on a single hidden layer feed forward neural network for extending to the nonlinear settings as an initial solution which utilizes the random Fourier features [24] expansion of the radial basis function (RBF) kernel. Our NP-NN is online (data are received sequentially, and each instance is discarded after it is looked only once), scalable allowing real time processing, capable of nonlinear modeling and successfully controls the false alarm rate without manual tuning. Updates of NP-NN are for the Fourier feature projection (first layer) and the classifier (second layer) parameters as well as the asymmetrical costs. However, NP-NN has two disadvantages. (i) It requires extensive cross validations for the RBF bandwidth, making it prone to overfitting (underfitting) if the bandwidth is chosen too small (large), and (ii) it has fixed modeling power since the bandwidth is kept constant, causing a model mismatch, i.e., overfitting (underfitting) in the earlier (later) phases of a data stream. In fact, the modeling power (i.e. the degree of nonlinearity) should be gradually increased in online applications as the learnability gradually improves with increasing data. The presented study in this paper removes these two disadvantages.

To be more precise, our focus here is the nonlinear data modeling in the context of online (i.e. scalable to large fast stream data) NP classification. We linearly combine an ensemble of online piecewise linear (so nonlinear) NP classifiers (i.e. experts as piecewise OLNPs) of varying modeling powers (i.e. varying number of pieces). The combination weights are time varying and proportional to the accumulated NP performances of the experts. Therefore, simpler models take larger weights early in the stream and more complex models start dominating as time progresses. This implements a desired gradual increase in the modeling power that is dynamically adaptive to the learnibility implied by the size of the available data at a time.

We mathematically show that the NP performance difference between our mixture of experts algorithm and that of the best expert vanishes asymptotically in the order $O(\frac{1}{\sqrt{T}})$ (T is the total number of processed instances), where the vanishing difference characterizes the difficulty of the NP classification (the NP problem becomes more difficult as $(\tau, \pi^-) \to (0, 0)$, where τ is the desired user-specified false alarm rate, π^- is the prior probability of the negative class). Our proposed algorithm successfully controls the false alarm rate at any specified level, it is dynamical in its modeling power and computationally highly efficient with online processing of complexity $O(T \log\log N_q)$, (N_q is the number of experts). In order to design the expert ensemble (as piecewise OLNPs; piecewise low complexity NP-NNs are also a possible design) and cutting down the processing complexity from $O(TN_q)$ to $O(T \log\log N_q)$, we use the well-known context tree weighting (CTW) technique [25] as a binary tree partitioning of the observation space. We call our algorithm (proposed in this manner) as Tree OLNP, which is falling into the category of cost sensitive learning with the stochastic updates described above. To our best knowledge, NP classification acquires these properties and performance guarantees as the first time in the literature which draws the main contribution of the presented study.

The celebrated CTW technique was originally introduced in [25] for sequential universal data compression and coding, which has been later adapted to various applications such as Bayesian inference in Markov chain for modeling discrete time sequences [26], recommendation systems [27], kernel design for string classification [28] and reinforcement learning [29]. Examples that are most relevant to our study are in [30]–[34]. Our work presented in this current manuscript is a fully developed version of our previous short conference paper (Turkish) [33]. Particularly, here in this manuscript, we additionally provide mathematically proven strong performance guarantees from the Neyman-Pearson perspective with far more comprehensive experimental analyses. The short conference version lacks these. The CTW partitions the observations space in [30] to obtain a piecewise linear regression that is twice universal in the space of all linear regressors as well as the ensemble of piecewise linear regressors over the tree. The binary context tree is generalized to nonbinary trees in [32] for multiarmed bandit problems. On the other hand, the CTW is used in [31] for regular binary classification while also optimizing the tree split parameters (hence, self organizing; we use iterative PCA in this study for a sound splitting) whereas [34] studies CTW based kernel density estimation for anomaly detection. We strongly emphasize that the problem considered in these studies and the problem we consider in the presented study are completely different. None of them considers the estimation of asymmetrical costs for false alarm rate controllability, and thus, we are the first in the literature to generalize the classification with the CTW into the Neyman-Pearson framework. Nevertheless, one might consider to directly combine the partitioning in [31] and OLNP [22] to achieve our goals in this study, and in fact, our method is similar to that. However, we also stress that if OLNP works naively in the set of partition region observations with no change, as it would be in the direct combination, then the overall false alarm rate can match the global target rate as desired but the true detection rate would substantially suffer from being suboptimal. This is because a global false alarm rate is to be satisfied globally only, and so does not have to be uniform across local regions. To address this in the algorithm design, we keep the OLNP parameter update local, but we conduct the class cost updates globally and share the globally updated class costs with all the local OLNPs. Also, [31] utilizes a self-organizing tree with learnable split parameters at nodes, here we opt for a PCA-based iterative approach for a more robust structure. Lastly, as a major distinction, we also analyze the behaviour of the partitioning tree from the perspective of NP loss, providing valuable insights into the scenarios with class imbalance.

III. Problem Description, Highlights and Novel Contributions

We introduce a randomized binary classification algorithm g to classify an observation $x \in \mathbb{R}^d$ as $\hat{y} \triangleq g(x) \in \{1, -1\}$ (predicted label), where $p_{X,Y}$ is the joint distribution that an i.i.d. (independent and identically distributed) labeled data stream $(x \in \mathbb{R}^d, y \in \{-1, 1\}) \sim p_{X,Y}$ follows. The proposed algorithm is desired to have the following properties.

P1 *False alarm (or false positive) controllability:* The detection power is maximized at a user-specified false alarm rate (τ, type I error). We address this by using the Neyman-Pearson (NP) formulation from the detection estimation theory [12]. Our algorithm solves the optimization problem

$$\max_{g} \mathrm{P}_d(g) \text{ subject to } \mathrm{P}_{fa}(g) \leq \tau, \tag{1}$$

where $\mathrm{P}_d(g) = \mathrm{E}[\mathrm{sgn}(g(X))|Y = 1]$[1] is the detection power, $\mathrm{P}_{fa}(g) = \mathrm{E}[\mathrm{sgn}(g(X))|Y = -1]$ is the false alarm rate, τ is the desired user-specified false alarm rate (TFPR: target false positive rate), and $\mathrm{E}[\cdot]$ is the expectation operator with respect to the data distribution as well as the randomization in the algorithm. The solution can be obtained through the NP-lemma [12] that also draws an *NP loss* (explained in the next section) measuring the level of optimality in (1). We call a classifier having this property **P1** as an *NP classifier* [1].

P2 *Dynamic data modeling power:* Data modeling power refers here to the classifier's capacity to capture the inherent complexity in the data distribution, or equivalently the degree of nonlinearity in the boundary between normal and abnormal classes. On the other hand, nonlinear decision boundaries can be effectively approximated by piecewise linear functions. Consequently, a tree-based classifier (as in the current work) possessing high power can be considered to have high number of pieces available in modeling the class separations through piecewise linear functions. In this regard, we simply consider the number of pieces as the indicator of degree of nonlinearity, and the depth of the tree as the data modeling power as it directly determines the number of pieces and available piecewise linear functions (size of expert ensemble). The

[1]Let $\mathrm{sgn}(\cdot)$ be the sign function returning 1 if its argument is nonnegative and 0 otherwise.

proposed algorithm operates at the right data modeling power (degree of nonlinearity in separation) by dynamically adapting to the amount of available data in an NP-optimal manner (based on the NP loss). We address this with a randomized mixture of experts [16]: for $1 \leq i \leq N_q$,

$$g(x) = 2 \times \text{sgn}(f_i(x)) - 1 \text{ with probability } q_i, \qquad (2)$$

where f_i's are the nonrandom and continuous valued (akin discriminants), i.e., $f_i(x) \in \mathbb{R}^d$, experts solving the same classification problem at varying modeling powers (at varying nonlinearities) as $\hat{y}_i = 2 \times \text{sgn}(f(x)) - 1 \in \{-1, 1\}$, and q_i's are the expert randomization weights such that $\sum_{i=1}^{N_q} q_i = 1$ and $q_i \geq 0 \ \forall i$. The proposed algorithm adjusts the weights in a way that the powerful experts achieve larger weights when enough data is available. Otherwise, simpler (less powerful) experts are favored. This property (**P2**) helps us mitigate the issues of overfitting and underfitting.

P3 *Computational scalability:* The proposed algorithm is designed computationally highly efficient for ensuring scalability to voluminous data. In particular, we use online processing together with a tree-based hierarchical representation of the experts. Given an i.i.d. fast stream data $\{(x_t, y_t)\}$, the instance x_t is received at time t, the prediction $\hat{y}_t = g_{t-1}(x_t)$ is made, g_{t-1} is updated based on the error $y_t - \hat{y}_t$ and (x_t, y_t) is discarded, yielding g_t in an online manner with linear $O(T)$ computational complexity as desired (T is the total number processed instances). Here, scalability to infinite amount of data is achieved with finite and fixed processing power since the complexity of our method per unit time / instance is constant. On the other hand, the number of experts N_q is typically chosen large to guarantee sufficient data modeling power, but this might be computationally prohibitive since the proposed algorithm requires (when naively implemented) complexity $O(N_q)$. To greatly cut down the computational complexity to $\sim O(\log \log N_q)$ in a twice-logarithmic manner, we use the binary partitioning tree based data processing framework (context tree weighting as mentioned in Section II) of [30], [31] which designs an ensemble of experts (f_i, such that $1 \leq i \leq N_q$, piecewise linear classifiers) of varying modeling powers (varying pieces). Our proposed method is especially applicable and useful when (1) adaptation to data in the run time is required (training and testing / inference / use of the method are intermingled) for handling the cold start problem (data inadequacy [38]) or for handling the nonstationarities (such as the presence of an adversary [39], concept change / drift [40], abrupt changes in observations due to an anomaly [41] or due to the emergency / investigation caused by an anomaly) in the data statistics and when (2) instant feedbacks are available in case of errors.

Our main contribution in this paper is to propose, as the first time in the literature, an algorithm that satisfies all these properties (P1, P2 and P3) at once. We use (1) the template of online linear NP classifier (OLNP) of [22] for satisfying the property P1, (2) mixture of piecewise OLNPs across which our algorithm provides a dynamical weighting in the competitive sense in terms of the NP performance for satisfying the property P2 (and also provides nonlinear classification as piecewise linear approximations), and (3) the binary partitioning tree of [31] to design a hierarchically represented ensemble for satisfying the property P3. We emphasize that the tree based classification in [31] does not have the NP property (P1) whereas we specifically consider here the false alarm rate controllability which is a fundamentally important difference (between [31] and the presented study). We also emphasize that the studies [22] and [23] have the NP property (P1) but they are not dynamic (i.e. static) in terms of the nonlinear modeling power (lacking the property P2), which is another fundamentally important difference (between [22], [23] and the presented study). While satisfying these properties in our methodology, we mathematically prove that our algorithm is competitive against the expert ensemble in terms of the NP loss. Namely, our algorithm approaches the NP performance of the best expert as more data become available while matching to the desired false alarm rate which has been achieved as the first time in the literature, to our best knowledge.

In this study, we use the Neyman-Pearson (NP) methodology primarily to control the false positive rate, not to enhance classification performance. In this case, classification improvements come from training an ensemble of expert models and their combined decisions in our final model. For controlling the false positive rate, one could alternatively adjust the model post-epoch by estimating the FPR and tuning parameters to meet a specified target rate τ. However, we view this alternative as impractical for two main reasons. (1) Since we study here a continuous learning of models in online manner, the reorganization in the alternative approach (FAR estimation and parameter tuning) can become quickly outdated in between epochs as the models keep changing while we continuously learn from the received data samples. (2) Even when enough data samples have been collected and the models have matured, sudden or gradual shifts in the observed data statistics, particularly in prior class probabilities, can render the models from the last update obsolete. In both scenarios, the false positive rate constraint is breached, causing the alternative method to fail. In the second case, a change detection process is needed to check if reorganization is required due to potential nonstationarity. Conversely, we introduce a versatile method that consistently maintains accurate false positive rate control using the NP formulation, effectively handling nonstationary prior probabilities without the need for parameter adjustments or change detection.

IV. Method

In this section, we construct our proposed algorithm by satisfying these desired properties.

Let us first clarify our notation. In the following Section IV-A, the general structure of our NP classifiers is denoted by f which is a basic building block in our complete classification scheme and it is linear. On the other hand, each f_i represents an expert in Section IV-B, which is a nonlinear NP classifier as a union of linear basic NP classifiers f_ν's. Here, ν represents a node of our binary partitioning tree and it indicates

that the corresponding linear NP classifier is trained in that node's observation space. Note that each expert f_i acquires a probability q_i that is proportional to the performance of the expert f_i. In this setup, the notation g denotes our final nonlinear and online NP classifier which picks the decision from the expert ensemble with respect to those probabilities q_i's. Note that since our method is dynamic, these classifiers and quantities are actually time varying, so we also use a subscript t in our notations when needed. Section IV-C details the binary partitioning tree-based combinations for obtaining the experts f_i's as union of local f_ν's in a computationally efficient and online manner. Here, we rely on the distinction between the subscripts i and ν to indicate whether it is an expert or a local linear NP classifier, and in general whether it is an expert related or a local node related quantity.

IV-A. *False Alarm Controllability: Neyman-Pearson (NP) Classification*

We start with explaining the false alarm controllability with the notation f below to denote an NP classifier, which serves as a basic building block, i.e., a linear NP template classifier, in constructing our final randomized binary classifier g.

It is known by the NP lemma that the solution f^* to the optimization problem in (1) is given by the likelihood ratio test (LRT) when the threshold is chosen appropriately, whereas LRT is known to minimize the Bayes risk [12]. Hence, $f^* = \arg\max_f \mathrm{P}_d(f)$ subject to $\mathrm{P}_{fa}(f) \leq \tau$ yields

$$2 \times \operatorname{sgn}(f^*(x)) - 1 = \hat{y} = 1 \Leftrightarrow \frac{p_{X|Y}(x|1)}{p_{X|Y}(x|-1)} \geq \hat{\mu}\frac{\pi^-}{\pi^+}$$

$$\text{subject to } \mathrm{P}_{fa}(f^*) = \tau \text{ (likelihood ratio test)} \tag{3}$$

$$\Leftrightarrow f^* = \arg\min_f \hat{\mu}_\tau \pi^- \mathrm{P}_{fa}(f) + \pi^+ \mathrm{P}_{nd}(f)$$

$$\text{(Bayes risk min. with } \hat{\mu}_\tau \geq 0 \text{ s.t. } \mathrm{P}_{fa}(f^*) = \tau) \tag{4}$$

$$\Leftrightarrow f^* = \arg\min_f \hat{\mu}_\tau \frac{n_T^-}{T}\frac{1}{n_T^-}\sum_{t=1}^{T} \operatorname{sgn}(f(x_t))\operatorname{sgn}(-y_t)$$

$$+ \frac{n_T^+}{T}\frac{1}{n_T^+}\sum_{t=1}^{T} \operatorname{sgn}(-f(x_t))\operatorname{sgn}(y_t) \text{ (empirical risk)} \tag{5}$$

$$\Leftrightarrow f^* = \arg\min_f \frac{1}{T}\sum_{t=1}^{T} \mu_\tau \operatorname{sgn}(-y_t f(x_t))$$

$$\text{(with } \mu_\tau = \hat{\mu}_\tau \text{ if } y_t = -1 \text{ and } \mu_\tau = 1 \text{ if } y_t = 1), \tag{6}$$

where $\hat{\mu}$ is the unknown cost of deciding 1 when in fact $y = -1$ (false alarm or false positive or type-I error cost) and it is to be estimated such that $\mathrm{P}_{fa}(f^*) = \tau$. Thus, $\hat{\mu}$ is replaced with $\hat{\mu}_\tau$. In (4), $\mathrm{P}_{nd}(f) = 1 - \mathrm{P}_d(f)$ is the no detection rate (the type-II error or miss rate). False alarm $\mathrm{P}_{fa}(f)$ and miss $\mathrm{P}_{nd}(f)$ rates as well as the prior probabilities of the negative (normal) and the positive (abnormal) classes (π^-, π^+) are replaced in (5) by the corresponding empirical estimates

$$\hat{\mathrm{P}}_{fa}(f) = \frac{1}{n_T^-}\sum_{t=1}^{T} \operatorname{sgn}(f(x_t))\operatorname{sgn}(-y_t)$$

$$\hat{\mathrm{P}}_{nd}(f) = \frac{1}{n_T^+}\sum_{t=1}^{T} \operatorname{sgn}(-f(x_t))\operatorname{sgn}(y_t)$$

and $\hat{\pi}^- = \frac{n_T^-}{T}$, $\hat{\pi}^+ = \frac{n_T^+}{T}$, where n_T^- (and n_T^+) is the number of data instances with the label -1 (and 1) out of T observations. To obtain a differential loss function for the optimization in (6), we replace the binary valued $\operatorname{sgn}(-y_t f(x_t)) \in \{0, 1\}$ with the continuous valued sigmoid

$$[0, 1] \ni l(-y_t f(x_t)) = \frac{1}{1 + e^{y_t f(x_t)}},$$

and arrive at the optimization

$$\begin{aligned} f^* &= \arg\min_f \frac{1}{T}\sum_{t=1}^{T} \mu_\tau l(-y_t f(x_t)) \\ &= \arg\min_f \frac{1}{T}\sum_{t=1}^{T} s_{t+1} \\ &= \arg\min_f L(f). \end{aligned} \tag{7}$$

This optimization is solved by the stochastic gradient descent based steps:

$$f^* \leftarrow f_t = f_{t-1} - \beta_f \nabla_{f_{t-1}} \mu_{\tau,t-1} l(-y_t f_{t-1}(x_t)), \tag{8}$$

with $\beta_f > 0$ being the step size. As for the update for μ_τ, since $\hat{\mathrm{P}}_{fa} \simeq l(f(x_t))$ can be seen as an estimate for $\mathrm{P}_{fa}(f)$ based on a single instance. To enhance the robustness of this estimation, it is recommended to use multiple sample instances. In our implementation, we maintain a buffer of 200 samples to estimate the false alarm rate. Since the false alarm rate cost $\hat{\mu}_\tau$ should be increased if $l(f(x_t))$ exceeds the user-specified desired false alarm rate τ (and decreased otherwise), we can estimate $\hat{\mu}_\tau$ through stochastic updates as in [22] by

$$\begin{aligned} \hat{\mu}^* \leftarrow \hat{\mu}_{\tau,t} &= \hat{\mu}_{\tau,t-1} + \beta_\mu (l(f_{t-1}(x_t)) - \tau) \\ &\text{only when } y_t = -1 \end{aligned} \tag{9}$$

(with $\beta_\mu > 0$ being the step size), which also sets $\mu_{\tau,t}$. It is essential to clarify that the variable μ_τ is initialized at 1. As indicated in (9), when the estimated FPR falls below τ, μ_τ decreases. To ensure convergence, β_f is set significantly larger than β_μ. Here we designate

$$s_t = \mu_{\tau,t-1} l(-y_t f_{t-1}(x_t)) \tag{10}$$

as the instantaneous or stochastic NP loss measuring the level of optimality when accumulated in time. Analogous to the NP classifier f_t, s_t also exhibits similar notational features. As detailed in the appendices, the instantaneous (accumulated) NP loss for an expert trained within a specific partition is represented by s_i (S_i for the accumulated version). This is further defined as the aggregate of instantaneous (accumulated) NP losses s_ν (S_ν for the accumulated version) for the nodes associated with the corresponding partition regions.

To obtain the *online linear NP classifier (OLNP)* of [22], one can simply use $f(x) = w^T x$ with $w = [\tilde{w}, b]^T$, where $\tilde{w}$ denotes the linear projection parameters and b is the bias. Then, based on a data stream $\{(x_t, y_t)\}$ and the updates in (8) and (9) for estimating the parameters w, OLNP can be sequentially trained as $f_t(\cdot)$ in an online manner. Note that OLNP has the property of false alarm controllability, and employs a typical regularization by also minimizing $\lambda\frac{||f||^2}{2}$, i.e., penalizing the squared magnitude of the classifier parameters

where λ is the regularization constant. OLNP is used as a template in our design of the expert ensemble ($\{f_i\}_{i=1}^{N_q}$ in (2)). In this section, we obtained the derivations of the NP classifiers based on the NP lemma in the detection/estimation theoretical framework, which allowed us to use the original inequality constrained in (1) as an equality constraint and we were able to parameterize $\hat{\mu}$ as $\hat{\mu}_\tau$ as a function τ. Similar end results are obtained in [22] through the Lagrange optimization by directly considering the original inequality constraint. We used their $\hat{\mu}$ updates here. We stress that our particular purpose in these derivations was to obtain the stochastic NP-loss which we use in the following section in our performance analysis.

IV-B. Dynamic Data Modeling Power: Ensemble of Experts

The algorithms OLNP (linear) of [22] and NP-NN (nonlinear) of [23] are both designed to operate at a fixed modeling power, which gives rise to problems if we consider the well-known central concept of bias-variance trade-off, or simply the issue of underfitting/overfitting. OLNP (NP-NN) is eventually too simple (too powerful) for data with nonlinear (linear) separation. Moreover, when given some data at time T from a data stream $\{(x_t, y_t)\}$, a linear classifier (e.g. OLNP) suffers from high bias (or high approximation error or simply underfitting) when the true separation is nonlinear. One can think of using a nonlinear classifier (e.g. NP-NN) in this case, but then at an earlier time when given less data, the nonlinear model potentially suffers from high variance (or high estimation or simply overfitting), as illustrated in Fig. 2a. Consequently, one must consider not only the underlying true separation (linear or nonlinear) but also what the amount of data allows to learn. We might be required to settle for a linear classifier when the available data is small even if the underlying true separation is nonlinear; and as more data become available, a good strategy would be not to attempt to learn the most powerful available nonlinear classifier but would be instead to gradually increase the degree of nonlinearity. Hence, the degree of nonlinearity we employ must be time-varying as opposed to being fixed.

To that end, we introduce a dynamic (varying) data modeling power through a finite ensemble of piecewise linear classifiers, i.e., ensemble of experts. An η-piece piecewise linear classifier $f_{\mathcal{P}}$ utilizes a partition $\mathcal{P} = \{R_1, R_2, \cdots, R_\eta\}$ of the observation space $\mathbb{R}^d$, where we have $R_k \cap R_{k'} = \emptyset$ for $k \neq k'$ and $\bigcup_{k=1}^{\eta} R_k = \mathbb{R}^d$ for the partition regions, and keeps a separate OLNP f_R for each partition region R. The update of the false alarm cost $\hat{\mu}$ in (9) is conducted based on the complete set of observations in order for all partition region OLNPs to have access to the same cost structure, whereas each partition region OLNP (say f_R) conducts the parameter update in (8) with only those instances falling in the respective region (with only the instances $x_t \in R$). Overall, the entire set of parameters $\bigcup_{k=1}^{\eta} w_{R_k}$ are learned. The resulting $f_{\mathcal{P}}$ is an online η-piece piecewise linear (so nonlinear) OLNP, whose degree of nonlinearity is η (the number of pieces or partition regions it employs). Thus, $f_{\mathcal{P}}(x) = f_R(x)$ if $x \in R$. Note that if $\eta = 1$, then $f_{\mathcal{P}}$ is just a regular OLNP, whereas as η increases it acquires the power of modeling any smooth nonlinearity (cf. Fig. 2a) as an η-piece piecewise linear (overall nonlinear).

Given an ensemble of partitions $\{\mathcal{P}_i\}_{i=1}^{N_q}$ with various corresponding pieces $\{\eta_i\}_{i=1}^{N_q}$, an ensemble of piecewise OLNPs $\{f_{\mathcal{P}_i}\}_{i=1}^{N_q}$ with various degrees of nonlinearities (with varying modeling powers) follows. We call $f_{\mathcal{P}_i}$ as an expert and denote it by f_i (reducing the subscript for simplicity). We leave the question of how to design the ensemble of partitions to the next section, and continue with our proposed randomized NP classifier (previously defined in (2)) as a mixture of experts in the online setting here with the subscript t: for $1 \leq i \leq N_q$,

$$g_t(x) = 2 \times \text{sgn}(f_{i,t}(x)) - 1 \text{ with probability } q_{i,t}. \quad (11)$$

At any time $t \geq 1$, the accumulated NP loss $S_{i,t}$ as well as the NP performance (i.e. quality) of the expert f_i can be measured by

$$\begin{aligned} S_{i,t} &= \sum_{t'=1}^{t} s_{i,t'} \\ &= S_{i,t-1} + s_{i,t} \end{aligned}$$

with the initial $S_{i,0} = 0$ and

$$\begin{aligned} U_{i,t} &= q_{i,0} \exp(-hS_{i,t}) \\ &= U_{i,t-1} \times \exp(-hs_{i,t}), \end{aligned}$$

respectively, where $s_{i,t}$ is the instantaneous or stochastic NP loss, $q_{i,0}$'s are initial probability assignments (i.e. priors with $\sum_{i=1}^{N_q} q_{i,0} = 1$) which should be chosen inversely proportional to the expert powers, and $h > 0$ is a learning rate parameter. Using the accumulated performance, we define the dynamic probability assignments as

$$q_{i,t} \triangleq \frac{U_{i,t-1}}{\sum_{j=1}^{N_q} U_{j,t-1}}, \quad (12)$$

where $\sum_{i=1}^{N_q} q_{i,t} = 1$ and $q_{i,t} \geq 0$, $\forall t \geq 1$. The critical observation regarding this probability assignment is that, intuitively, simpler experts outweigh in the beginning of the stream and more powerful experts gradually dominate as time progresses, implementing the idea in Fig. 2a just as desired. This provides a naive direct implementation of our proposed online randomized NP classification g_t given in (11), which is asymptotically guaranteed to outperform (or perform at least as well as) the best expert on average in terms of the NP loss and is tuned to the best observation space partition. Namely, we mathematically prove that for all i and for any data stream $\{(x_t, y_t)\}$,

$$\begin{aligned} \frac{1}{t}\left(\overline{S}_t - S_{i,t}\right) \leq & \frac{1}{8}hC^2 - \frac{1}{ht}\log q_{i,0} = \frac{1}{\sqrt{t}}\left(\frac{C^2}{8} - \log q_{i,0}\right) \\ & \to 0 \text{ as } t \to \infty \text{ with } h = \frac{1}{\sqrt{t}} \end{aligned} \quad (13)$$

holds, where the accumulated expected (with respect to the randomization in the algorithm) NP loss is given by $\overline{S}_t = \sum_{t'=1}^{t} \sum_{i=1}^{N_q} q_{i,t'} s_{i,t'}$, and C is a constant upper bounding $C \geq |s_{i,t}|$ for all i and t. The derivations of the proof are given in Appendices, which follow the lines of the derivations of the mathematical analysis in [31], but also deviate in the sense that we adapted here to the Neyman-Pearson formulation by specifically considering the NP performance. To be more

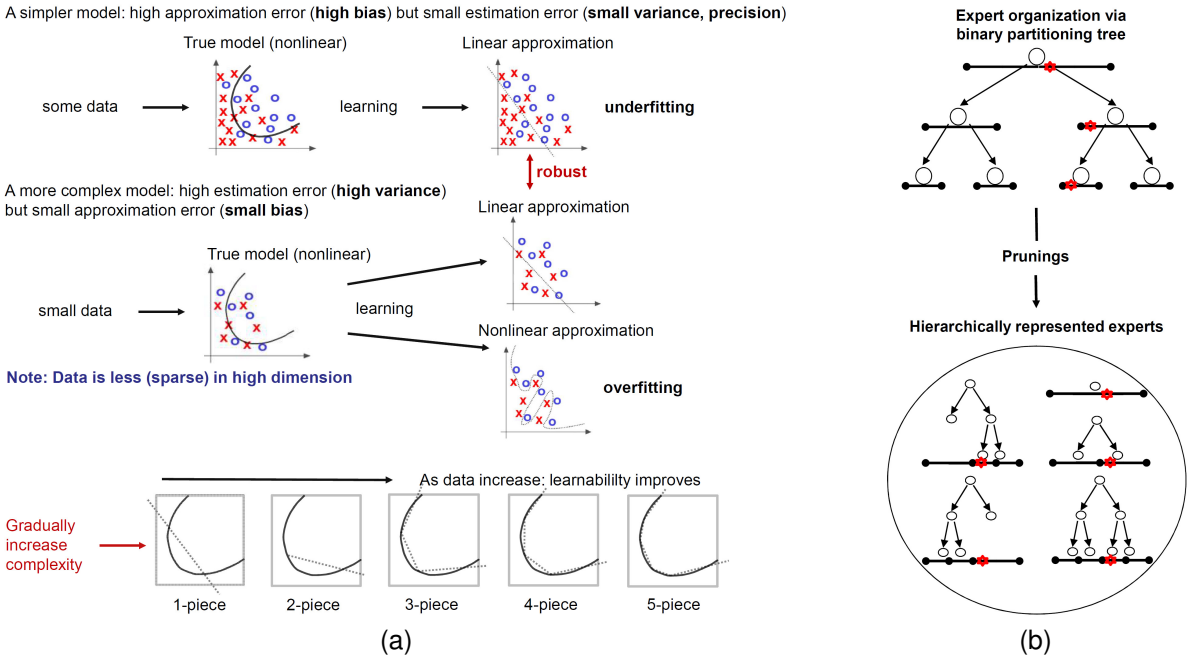

Figure 2. A general illustration of the presented method. (a) Bias-variance trade-off, and dynamic data modeling power: Ensemble of experts (b) Hierarchical expert ensemble

precise, the loss structure in [31] and our NP loss here are different.

In this sense, the introduced algorithm weights over the ensemble dynamically optimally to operate with the right modeling power and mitigate the issue of overfitting/underfitting. Here, we set for the constant $C \geq \max(\hat{\mu}_{\tau,t-1}, 1)\ \forall i,t \Rightarrow C \geq |\mu_{\tau,t-1}|\ \forall i,t \Rightarrow C \geq |\mu_{\tau,t-1} l(-y_t f_{i,t-1}(x_t))|\ \forall i,t \Leftrightarrow C \geq |s_{i,t}|\ \forall i,t$, where $\hat{\mu}_{\tau,t-1} \to \hat{\mu}_\tau \to \infty$ as $(\tau,t) \to (0,\infty)$ and $\hat{\mu}_{\tau,t-1} \to \hat{\mu}_\tau = 0$ as $(\tau,t) \to (1,\infty)$ with an appropriately chosen β_μ. Inspecting (13), outperforming (or achieving the performance of) the best expert takes longer time as τ gets smaller. On the other hand, $\hat{\mu}_{\tau,t}$ takes updates only when $y_t = -1$ (as shown in (9)) which essentially multiplies the required number of i.i.d. samples from the stream with $\frac{1}{\pi^-}$ for convergence to the desired false alarm rate τ, for both our algorithm as well as the ensemble experts. Thus, as another important contribution of the presented study, here we also characterize that the NP problem becomes harder as $(\tau, \pi^-) \to (0,0)$.

IV-C. Computational Scalability: Online Processing and Binary Partitioning Tree

The data modeling power (nonlinear learning capability) of our algorithm can be improved arbitrarily to any desired degree by increasing the size as well as the variety of the employed expert ensemble. As this size increases, the best expert improves, so does the proposed algorithm as guaranteed in a competitive manner, cf. (13). On the contrary, a naive implementation of the proposed algorithm requires to run N_q experts in parallel to pick a decision randomly at each time. The resulting computational complexity $O(N_q)$ that is linear with respect to the ensemble size is impractical if one uses a necessarily large ensemble in an attempt to improve the modeling, which potentially hinders the real-time processing. However, the complexity can be immensely reduced to $O(\log\log N_q)$ with a versatile implementation, if the experts are designed carefully and represented hierarchically. For this, we use the general information processing framework of the binary partitioning context tree that has been previously successfully applied for various purposes such as density estimation in [34] and classification (this is not Neyman-Pearson, unlike the presented study) in [31], dating back to coding in [25]. A depth-D binary tree is constructed on which a region from the observation space is kept and split into two, and the resulting two sub-regions are assigned to the corresponding child nodes. This process is applied to all the nodes recursively starting from the root node until the leaves are reached at the depth D, with the root node keeping the complete observation space. As an example, in the case of a one dimensional observation space $[0,4]$ being partitioned by a depth-2 tree, the observation $x_t = 2.1$ in Fig. 2b visits the root node $[0,4]$ and its right $(2,4]$ and the right-left $(2,3)$ child nodes, respectively. The set of the prunings of this tree here provides an ensemble of partitions, $\mathcal{P}_1 = \{[0,4]\}$ (single piece), $\mathcal{P}_2 = \{[0,2],(2,4]\}$ (two pieces), $\mathcal{P}_3 = \{[0,2],(2,3],(3,4]\}$ (three pieces), $\mathcal{P}_4 = \{[0,1],(1,2],(2,4]\}$ (three pieces), $\mathcal{P}_5 = \{[0,1],(1,2],(2,3],(3,4]\}$ (four pieces), where an expert $f_{\mathcal{P}_i,t}$ is obtained when a separate OLNP is trained for each region of the partition $\mathcal{P}_i$, e.g., $f_{\mathcal{P}_3,t-1}(2.1) = f_{(2,3],t-1}(2.1)$, and the ensemble size is $N_q = 5$. In general, $f_{\mathcal{P}_i,t-1}(x_t) = f_{R_{i,j},t-1}(x_t)$ if $x_t \in R_{i,j}$, $\mathcal{P}_i = \{R_{i,1}, R_{i,2}, \cdots, R_{i,\eta_i}\}$ and $f_{R_{i,j},t-1}$ is the OLNP trained with only those observations falling in $R_{i,j} \in \mathcal{P}_i$. As for the tree in this example, the node regions are given by $O \equiv [0,4], O_l \equiv [0,2], O_r \equiv$

$(2,4], O_{l,l} \equiv [0,1], O_{l,r} \equiv (1,2], O_{r,l} \equiv (2,3]$, and $O_{r,r} \equiv (3,4]$, where O is the root node and O_l (O_r) are the left (right) child of the root node, and similarly for the others. Hence, at each node ν, there is a region and an OLNP $f_{\nu,t-1}$ producing the corresponding node decision. Let us use for simplicity $f_{i,t-1} \triangleq f_{\mathcal{P}_i,t-1}$.

Since $\{f_{i,t-1}(2.1)\}_{i=1}^{N_q=5} = \{f_{O,t-1}(2.1), f_{O_r,t-1}(2.1), f_{O_{rl},t-1}(2.1)\}$, obtaining the node decisions only is sufficient to obtain the final decision of our algorithm $g_{t-1}(x_t)$ for any observation x_t in general for a depth-D tree visiting the nodes $\{\nu_0 = O, \nu_1, \cdots, \nu_D\}$ as

$$g_{t-1}(x_t) = 2 \times \text{sgn}(f_{\nu_i,t-1}(x_t)) - 1 \tag{14}$$

with probability $\tilde{q}_{i,t-1}$ for $0 \leq i \leq D$, where the expert probabilities q are accumulated in $\tilde{q}$ by $\tilde{q}_{i,t-1} = \sum_{j=1}^{N_q} q_{j,t-1} 1_{\{f_{j,t-1}(x_t) = f_{\nu_i,t-1}(x_t)\}}, 0 \leq i \leq D$. Here, $1_{\{\cdot\}}$ is the indicator function returning 1 if its argument holds, and 0 otherwise.

Therefore, the binary partitioning tree-based implementation of our algorithm (described in Section IV-A and Section IV-B) has computational complexity $O(D)$ that is linear in the tree depth. Comparing with the complexity $O(N_q)$ of the naive implementation, immense computational gains are achieved in twice logarithmic manner by reducing from $O(N_q)$ to $O(D)$ since the number of prunings of a binary tree is twice exponential in the depth, i.e., $N_q \simeq (1.5)^{2^D}$ [25]. We call this efficient implementation Tree OLNP.

Two main aspects of the computational efficiency of our algorithm is first the online processing with linear complexity in the number of the processed instances ($O(t)$), as already explained in Section IV-A and Section IV-B. The second aspect is obtaining a complexity that is linear in the tree depth and twice logarithmic in the number of the experts in the ensemble ($O(D) \sim O(\log \log N_q)$, overall: $O(Dt)$), as explained in this section -however- by introducing the lumped probabilities $\hat{q}_{i,t}$'s which must also be obtained sequentially in the $O(D)$ (linear in depth) manner. The $O(D)$ computation of the lumped probabilities is shown in Appendices together with a table of our algorithm as pseudo-code. We stress that our main contribution in this section is to incorporate the Neyman-Pearson formulation of classification with false alarm rate controllability into the general information processing framework of the context trees [25], [31], [34].

Our algorithm Tree OLNP with only a root node (single depth) is a regular OLNP running for the complete observation space, and it collects and organizes OLNPs in a piecewise manner with higher depths. Tree OLNP models nonlinear separations more powerfully as the tree depth increases whereas a regular OLNP is not capable of learning nonlinearity. A limitation for Tree OLNP emerges as the observation ($x \in \mathbb{R}^d$) dimensionality d grows since finding the right way of splittings (partitionings) at the nodes is challenging. This is not an issue when d is small, e.g., $d = 1, 2$ or $d = 3$, and one can simply use even splits (mid-point between min/max values) feature by feature at each depth; but that would require an extremely large tree depth in high dimension. As a solution for high dimension, we use PCA transformation at the nodes for Tree OLNP and then evenly split after projecting to the vector of the largest variance. We also provide detailed computational complexity comparisons in terms of the required number of dot products in Table I along with the actual realizations in Table III (in Appendices) for various benchmark datasets. Generally, OLNP [22] has the fastest processing and Tree OLNP is the second fastest. In contrast, NP-NN [23] is computationally heavy. This is for the regular phase of running the algorithms. All three algorithms are online (OLNP, NP-NN and Tree OLNP) and able to process data in real time sequentially without a separate training. However, NP-NN (unlike OLNP and Tree OLNP) additionally requires a separate phase of extensive cross validations for its size M of Fourier features as well as the bandwidth parameter Ω of the RBF kernel for every τ. Namely, our algorithm Tree OLNP dynamically determines the right modeling power on the fly (with regards to the amount of available data at each time) during the data stream without having to know the horizon (the total number of instances to be processed), whereas NP-NN achieves the required degree of modeling complexity via extensive cross validations (before the processing starts, and it is not dynamic as it is fixed for all times in the stream once it is set) which not only hinders a fluent online processing but also requires the knowledge of the horizon. Finally, OLNP and NP-NN neither have a dynamical control over the modeling power nor provide competitive performance guarantees; on the contrary, our Tree OLNP has those valuable properties. This is reflected in our performance results which we present next.

V. Performance Evaluations and Experimental Results

We present performance evaluations of Tree OLNP as well as comparisons with OLNP [22] and NP-NN [23], based on 8 different real and synthetic datasets [35], [36] from various fields after applying z-score (zero mean unit variance) normalization. For each dataset, class with the largest size is assigned to label -1, and the remainings are considered as label 1. We split each dataset to training (70%), validation (15%) and test (15%). Validation set is used for parameter optimization via cross-validation and algorithms are trained in an online manner on the training set. The performance is probed on the test set at logarithmically spaced 100 time points. Cross validation is only applied to NP-NN whereas OLNP and Tree OLNP do not require extensive parameter tuning. We run each experiment for every target false alarm rate in $\{0.005, 0.01, 0.05, 0.1, 0.2\}$ for 32 times and calculate NP-score $= \kappa \max(\hat{P}_{\text{fa}}(f) - \tau, 0) + \hat{P}_{\text{nd}}(f)$ of [37] which measures the maximization of realized TPR together with the trackability of the target FPR (TFPR), i.e., TFPR shall match realized FPR. Here, lower NP-score is better. We opted for the default choice $\kappa = 1/\tau$ in our evaluations. To improve the convergence, we augment the training set by concatenating random shuffles of the same set to achieve a sample size of 150×10^3. This augmentation is only for the training set and does not apply to fixed validation or test sets. We select 15 runs out of 32 and filter out the cases of convergence issues. This filtering is applied for all algorithms to all datasets.

Table I
COMPUTATIONAL COMPLEXITY COMPARISONS AMONG OLNP, TREE OLNP AND NP-NN ALGORITHMS. T IS THE TOTAL NUMBER OF PROCESSED OBSERVATIONS AND M (FOURIER PROJECTION SIZE IN NP-NN) IS TYPICALLY FAR LARGER THAN THE TREE DEPTH D IN TREE OLNP, I.E., $D >> M$. TYPICALLY, $2 \leq M \leq 100$ DEPENDING ON THE CROSS VALIDATION WHEREAS $2 \leq D \leq 6$.

	Partition	Transformation	Classification	Total Dot Products Per Sample	Total Dot Products	Complexity
OLNP	0	0	1	1	T	$O(T)$
Tree OLNP	$D-1$	0	D	$2D-1$	$2TD-T$	$O(TD)$
NP-NN	0	Md	$2M$	$Md+2M$	$TMd+2TM$	$O(TMd)$

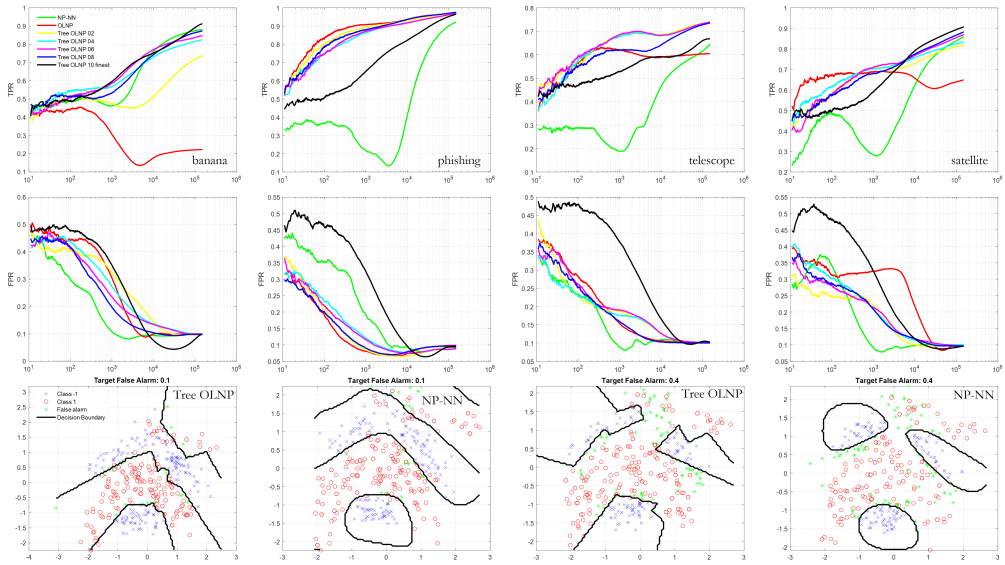

Figure 3. The transient behavior (TPR and FPR) of the algorithms are shown for TFPR $\tau = 0.1$ with varying depths on the Banana, Phishing, Telescope and Satellite datasets. We also present decision boundaries learned by NP-NN and Tree OLNP with depth 6 on the visually representable 2 dimensional Banana dataset for TFPR $\tau = 0.1$ and TFPR $\tau = 0.4$.

Transient graphs given in Fig. 3 are mean values collected from the training. Similarly, results represented in Table II are calculated on test sets (more specifically, we use the latest data point in logarithmically spaced performance evaluation sequence) on the same run population.

The classifier parameters $w = [\tilde{w}, b]^T$ are both initialized around 0. For NP-NN, the RBF kernel bandwidth Ω and the projection size M are cross-validated with $\Omega \in \{0.01, 0.02, 0.05, 0.1, 0.2, 0.5, 1, 2, 5, 10\}/d$ and $M \in \{2, 5, 10, 20, 40, 80, 100\}$, where d is the observation dimension. We initially set $\hat{\mu}_\tau = 1$. The step sizes β_f and β_μ are cross-validated with $\beta_f \in \{0.1, 0.01\}$ and $\beta_\mu \in \{2.5\times10^2, 5\times10^2, 1\times10^3, 5\times10^3, 1\times10^4, 1\times10^5\}/n^-$. The split probability is set as $q_s = \frac{1}{2}$. Performance metrics are (1) visual inspection of the receiver operating characteristics curve (ROC): realized true positive rate (TPR) vs realized false positive (alarm) rate (FPR), (2) area under the ROC curve (AUC), and (3) the NP-score.

Fig. 3 reports the transient behavior (realized TPR vs realized FPR) of OLNP, NP-NN and our proposed Tree OLNP with different depths on Banana, Phishing, Telescope and Satellite datasets for TFPR $\tau = 0.1$. We also include the finest partition of Tree OLNP with depth of 10 (Tree OLNP 10 finest is asymptotically the best expert at depth 10) to our comparisons. We first observe that for all datasets, all models successfully control the target false alarm rate. On Phishing, Telescope and Satellite datasets, the Tree OLNP models strongly outperform the finest partition (the best expert), until the end of the stream due to the overfitting of the finest partition, in terms of TPR compared to the Tree OLNP models. At the end of the stream, the performance difference between our proposed Tree OLNP and the finest partition vanishes, which experimentally reinforces/demonstrates our mathematically proven performance guarantees. The finest partition does not suffer much on the Banana dataset since it is of low dimension and overfitting is not much issue. As clearly observed, our proposed algorithm Tree OLNP mitigates overfitting by favoring the less complicated models when the number of data processed by the ensemble model is low. As more samples are observed, weights of complex experts increase and underfitting is successfully prevented. Hence, a superior performance is obtained by our proposed Tree OLNP generally for all times. As expected, Tree OLNP (piecewise linear, hence a nonlinear model) strongly outperforms OLNP (linear model) when the dataset is nonlinear, and perform comparably when the dataset is linear, e.g., Phishing, which shows robustness of Tree OLNP. Similar to the finest partition of Tree OLNP 10, NP-NN is also a complex algorithm that - in addition- requires extensive parameter tuning. As seen from Fig. 3, Tree OLNP strongly outperforms NP-NN as well in all cases uniformly at all times. Also, lower (high) depth trees generally tend to perform slightly better early (late). We also present decision boundaries for Tree OLNP and NP-NN using 2 dimensional banana dataset for $\tau \in \{0.1, 0.4\}$, where Tree

This article has been accepted for publication in IEEE Journal of Selected Topics in Signal Processing. This is the author's version which has not been fully edited and content may change prior to final publication. Citation information: DOI 10.1109/JSTSP.2025.3544024

Table II
PERFORMANCE COMPARISONS OF OLNP, NP-NN AND TREE OLNP[2]

Dataset (n_-, n_+, d)	OLNP			NP-NN			Tree OLNP		
	TFPR(τ)$\rightarrow$ 0.01	0.05	0.1	0.01	0.05	0.1	0.01	0.05	0.1
avila (12295, 8572, 10)	TPR$\rightarrow$ 0.040$\pm$0.005	0.231$\pm$0.026	0.408$\pm$0.020	0.220$\pm$0.040	0.502$\pm$0.022	0.603$\pm$0.023	0.170$\pm$0.014	0.422$\pm$0.016	0.580$\pm$0.018
	NP-score$\rightarrow$ 1.696$\pm$0.213	0.965$\pm$0.082	0.721$\pm$0.069	1.328$\pm$0.831	0.691$\pm$0.099	0.551$\pm$0.047	1.221$\pm$0.103	0.852$\pm$0.120	0.604$\pm$0.076
	AUC(until 0.2 TFPR)$\rightarrow$ 0.076$\pm$0.007			0.119$\pm$0.008			0.107$\pm$0.009		
banana (2924, 2376, 2)	0.108$\pm$0.015	0.173$\pm$0.016	0.227$\pm$0.019	0.703$\pm$0.080	0.844$\pm$0.020	0.889$\pm$0.016	0.715$\pm$0.029	0.838$\pm$0.018	0.886$\pm$0.015
	1.518$\pm$0.254	1.090$\pm$0.169	0.875$\pm$0.084	0.829$\pm$0.324	0.534$\pm$0.174	0.291$\pm$0.081	0.697$\pm$0.235	0.536$\pm$0.201	0.369$\pm$0.090
	0.045$\pm$0.005			0.176$\pm$0.011			0.179$\pm$0.012		
covertype (297711, 283301, 54)	0.156$\pm$0.010	0.379$\pm$0.013	0.550$\pm$0.006	0.325$\pm$0.022	0.603$\pm$0.009	0.744$\pm$0.006	0.138$\pm$0.016	0.363$\pm$0.015	0.590$\pm$0.009
	1.067$\pm$0.101	0.689$\pm$0.034	0.481$\pm$0.017	0.862$\pm$0.159	0.421$\pm$0.028	0.282$\pm$0.009	1.041$\pm$0.123	0.709$\pm$0.046	0.442$\pm$0.020
	0.101$\pm$0.004			0.136$\pm$0.003			0.102$\pm$0.002		
fourclass (555, 307, 2)	0.434$\pm$0.073	0.504$\pm$0.078	0.567$\pm$0.060	1.000$\pm$0.000	1.000$\pm$0.000	1.000$\pm$0.000	0.959$\pm$0.033	0.981$\pm$0.021	0.987$\pm$0.021
	2.860$\pm$1.009	0.892$\pm$0.353	1.033$\pm$0.245	0.182$\pm$0.362	0.320$\pm$0.415	0.246$\pm$0.261	1.552$\pm$1.164	0.857$\pm$0.317	0.479$\pm$0.143
	0.118$\pm$0.022			0.231$\pm$0.033			0.222$\pm$0.028		
miniboone_pid (93565, 36499, 50)	0.275$\pm$0.020	0.591$\pm$0.013	0.726$\pm$0.012	0.377$\pm$0.011	0.693$\pm$0.012	0.817$\pm$0.007	0.373$\pm$0.026	0.717$\pm$0.013	0.840$\pm$0.010
	0.915$\pm$0.089	0.493$\pm$0.045	0.319$\pm$0.039	0.811$\pm$0.094	0.418$\pm$0.043	0.225$\pm$0.028	0.928$\pm$0.118	0.416$\pm$0.059	0.252$\pm$0.061
	0.132$\pm$0.007			0.150$\pm$0.005			0.159$\pm$0.011		
phishing (6157, 4898, 68)	0.855$\pm$0.010	0.934$\pm$0.008	0.968$\pm$0.007	0.876$\pm$0.013	0.947$\pm$0.009	0.968$\pm$0.006	0.918$\pm$0.011	0.959$\pm$0.006	0.979$\pm$0.006
	0.597$\pm$0.176	0.243$\pm$0.095	0.125$\pm$0.069	0.956$\pm$0.371	0.343$\pm$0.107	0.121$\pm$0.094	1.241$\pm$0.291	0.331$\pm$0.113	0.222$\pm$0.042
	0.196$\pm$0.009			0.187$\pm$0.007			0.197$\pm$0.011		
satellite (4399, 2036, 36)	0.624$\pm$0.027	0.642$\pm$0.040	0.655$\pm$0.036	0.727$\pm$0.036	0.869$\pm$0.019	0.899$\pm$0.021	0.744$\pm$0.022	0.852$\pm$0.020	0.890$\pm$0.020
	0.919$\pm$0.378	0.524$\pm$0.141	0.495$\pm$0.099	1.098$\pm$0.407	0.572$\pm$0.181	0.259$\pm$0.063	2.176$\pm$0.308	0.701$\pm$0.147	0.437$\pm$0.075
	0.124$\pm$0.007			0.183$\pm$0.017			0.173$\pm$0.013		
telescope (12332, 6688, 10)	0.350$\pm$0.018	0.518$\pm$0.020	0.627$\pm$0.017	0.520$\pm$0.049	0.647$\pm$0.021	0.728$\pm$0.021	0.481$\pm$0.032	0.675$\pm$0.017	0.764$\pm$0.018
	0.877$\pm$0.179	0.641$\pm$0.157	0.456$\pm$0.054	0.817$\pm$0.301	0.503$\pm$0.082	0.364$\pm$0.066	1.079$\pm$0.361	0.490$\pm$0.118	0.445$\pm$0.157
	0.121$\pm$0.004			0.146$\pm$0.012			0.163$\pm$0.016		

[2]We present performance evaluations of the proposed algorithm Tree OLNP, as well as the state-of-the-art algorithms OLNP and NP-NN. The evaluation metrics are calculated for each target false positive rate (TFPR $\in \{0.005, 0.01, 0.05, 0.1, 0.2\}$) on 8 different datasets. First column shows the number of features for each dataset with the number of positive (target) and negative (non-target) samples. For each dataset, first row is the achieved TPR, second row is the achieved NP-score and the third row is the area under curve (AUC) of the receiver operating characteristics (ROC) curve of TPR vs TFPR. AUC of ROC curves are calculated up to 0.2 target false alarm rate and do not include operating points with false positive rate higher than 0.2. We emphasize that in most anomaly detection applications, typically, only the low TFPR values are considered.

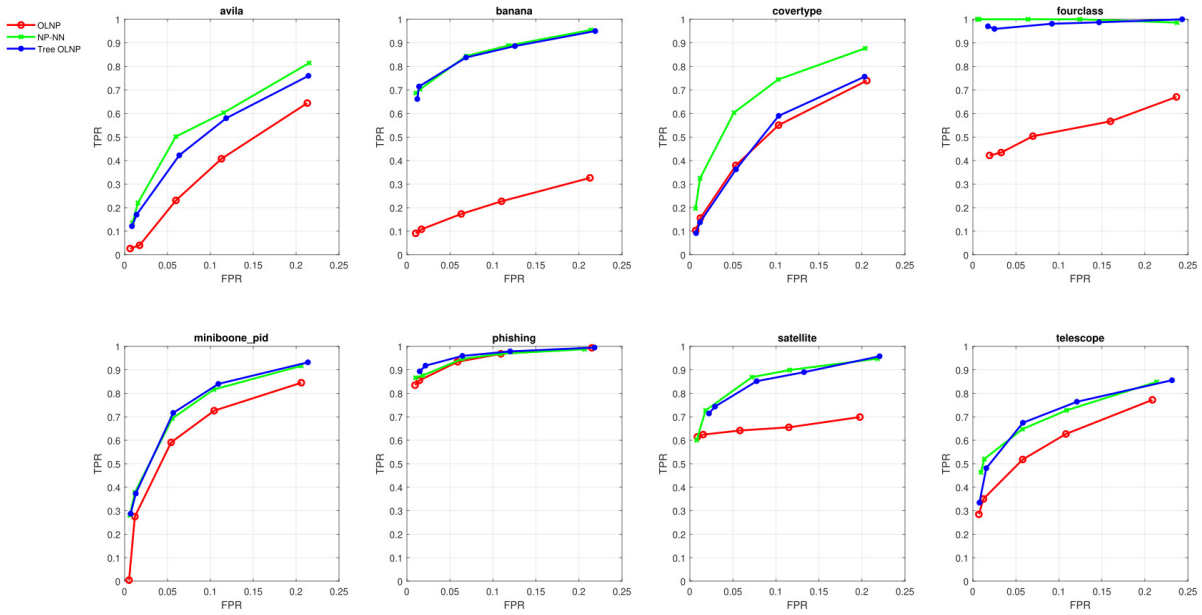

Figure 4. Receiver operator characteristics (ROC: realized FPR vs realized TPR) curves are shown for all the datasets. This figure visualizes (and should be inspected together with) Table II for better readability, as each data point on these curves corresponds to the mean values provided in Table II.

OLNP shows nonlinear decision boundaries in piecewise linear manner.

Our detailed asymptotical results (end behavior at the point of convergence when all the data instances are processed) are in Table II along with the corresponding ROC curves in Fig. 4. We observe that our proposed Tree OLNP (depth 8) and NP-NN outperform OLNP in 6 out of 8 datasets in terms of AUC (spanning various multiple TFPRs). On phishing and covertype, Tree OLNP performs similarly with OLNP, and outperforms (underperforms) NP-NN on phishing (covertype). Tree OLNP and NP-NN perform comparably on 5 datasets (AUC difference is equal or less than 0.1) except that Tree OLNP outperforms (underperforms) on 1 (2) dataset(s). Note that NP-NN requires extensive hyperparameter tuning in order to perform well; and it severely suffers from overfitting during the transient phase (in fact until the very end, cf. Fig. 3). Also, these findings are only for the end point in time. The similar performance at the end of the processing between Tree OLNP and NP-NN indicates that Tree OLNP asymptotically performs at least as well as NP-NN. This again experimentally demonstrates our performance guarantees since NP-NN is powerful like the best expert of finest partition of Tree OLNP at depth 10. As a result, thanks to the introduced dynamical data modeling power, Tree OLNP manages to successfully obtain a powerful nonlinear data modeling at the end while delivering a superior performance during the transient phase.

VI. Conclusion

We proposed a classification framework for online Neyman-Pearson (NP) classifiers which maximizes detection power subject to a given false alarm rate constraint. Based on a binary partitioning tree in the observation space, the proposed framework generates an hierarchical ensemble of online NP classifiers (experts) and makes a decision based on a performance driven nonstationary expert mixture. Since our ensemble spans various degrees of nonlinearities as piecewise linear models, we achieve a dynamic modeling power (hence mitigating the under/over fitting issues) in a computationally highly efficient manner. While successfully controlling the false alarm rate, we showed that, both mathematically and experimentally, our algorithm asymptotically achieves the NP performance of the best expert as well as the most prominent competing techniques, whereas our algorithm is significantly superior during learning in the transient phase as a result of the introduced dynamical nonlinear modeling.

Appendices: Derivations of our performance analysis in Section IV-B

The expected (with respect to the randomization in the algorithm) instantaneous NP loss is given by $\overline{s}_t = \sum_{i=1}^{N_q} q_{i,t} s_{i,t}$, and similarly for the expected accumulated loss $\overline{S}_t = \sum_{t'=1}^{t} \overline{s}_{t'}$ yielding the inequality for the expected instantaneous NP performance $\sum_{i=1}^{N_q} q_{i,t} \exp(-h s_{i,t})$ as $\exp(-h\overline{s}_t + h^2C^2/8) \geq \sum_{i=1}^{N_q} q_{i,t} \exp(-h s_{i,t}) = \sum_{i=1}^{N_q} \frac{U_{i,t-1}}{\sum_{j=1}^{N_q} U_{j,t-1}} \exp(-h s_{i,t}) = \frac{\sum_{i=1}^{N_q} U_{i,t}}{\sum_{j=1}^{N_q} U_{j,t-1}}$ due to the Hoeffding's inequality, where C is a constant upper bounding $C \geq |s_{i,t}|$ for all i and t. Then, we obtain $\exp(-h\overline{S}_t + th^2C^2/8) = \prod_{t'=1}^{t} \exp(-h\overline{s}_{t'} + h^2C^2/8) \geq \prod_{t'=1}^{t} \frac{\sum_{i=1}^{N_q} U_{i,t'}}{\sum_{j=1}^{N_q} U_{j,t'-1}} = \sum_{i=1}^{N_q} U_{i,t}$ to conclude that for all i and

Table III
ACTUAL COMPUTATIONS REALIZED CORRESPONDING TO TABLE 1 IN SECTION IV-C[3]

Dataset (n^-, n^+, d)	OLNP	Tree OLNP	NP-NN				
	all τ	all τ	0.005	0.01	0.05	0.1	0.2
avila (12295, 8572, 10)	20867	104335	500808	1252020	20032320	25040400	2504040
banana (2924, 2376, 2)	5300	26500	2120000	106000	212000	848000	212000
covertype (297711, 283301, 54)	581012	3486069	650733440	65073344	3253667200	1301466880	325366720
fourclass (555, 307, 2)	862	4310	6896	17240	6896	6896	6896
miniboone_pid (93565, 36499, 50)	130064	650320	135266560	135266560	676332800	33816640	33816640
phishing (6157, 4898, 68)	11055	55275	77385000	61908000	77385000	7738500	77385000
satellite (4399, 2036, 36)	6435	32175	489060	2445300	19562400	2445300	489060
telescope (12332, 6688, 10)	19020	95100	456480	18259200	9129600	2282400	456480

Algorithm The pseudo-code of Tree OLNP

1: Set the tree depth D, target false alarm rate τ, regularization λ, and the parameters β_f, β_μ and h.
2: Construct the tree and initialize.
3: **for** $t = 1, 2, \dots$ **do**
4: Receive the observation x_t and find the visited nodes ν_i's.
5: Calculate the lumped probabilities (top-down) $\hat{q}_{i,t-1}$'s. (12)
6: Calculate the decision using $g_{t-1}(x_t)$ and observe the feedback y_t. (11)
7: Update μ_τ and obtain $\mu_{\tau,t}$. (9)
8: Calculate the instantaneous (top-down) NP loss $s_{\nu,t}$ for each visited node. (10)
9: Obtain the update node classifiers (top-down) $f_{\nu,t}$'s. (8)
10: Obtain the updated combined performance $\mathcal{U}_{\nu,t}$'s for each visited node (bottom-up).(15, 16)
11: **end for**

for any stream $\{(x_t, y_t)\}$,

$$\begin{aligned}
&\exp(-h\overline{S}_t + th^2C^2/8) \geq U_{i,t} = q_{i,0}\exp(-hS_{i,t}) \\
&\Leftrightarrow -h\overline{S}_t + th^2C^2/8 \geq \log q_{i,0} - hS_{i,t} \\
&\Leftrightarrow \frac{1}{t}\left(\overline{S}_t - S_{i,t}\right) \leq \frac{1}{8}hC^2 - \frac{1}{ht}\log q_{i,0} \\
&= \frac{1}{\sqrt{t}}\left(\frac{C^2}{8} - \log q_{i,0}\right) \to 0 \text{ as } t \to \infty \text{ with } h = \frac{1}{\sqrt{t}}.
\end{aligned}$$

These derivations follow the lines of the derivations of the mathematical analysis in [31], but also deviate in the sense that we adapted here to the Neyman-Pearson formulation by specifically considering the NP performance. To be more precise, the loss structure in [31] and our NP loss here are different.

[3]Number of dot products for various benchmark datasets with $\tau \in \{0.005, 0.01, 0.05, 0.1, 0.2\}$ for 5 experts is given. In NP-NN, due to the required cross validation for Ω and M, we have different number of dot products whereas this number is same for all τ in Tree OLNP.

APPENDICES: $O(D)$ COMPUTATION OF THE $\tilde{q}_{i,t}$ RANDOMIZATION PROBABILITIES IN SECTION IV-C

Let us introduce the combined performance variable $\mathcal{U}_{\nu,t}$ that is defined for any node over the tree but updated in time for only those nodes $\nu \in \{\nu_0 = O, \nu_2, \cdots, \nu_D = leaf\}$ the observation x_t visits as

$$\mathcal{U}_{\nu,t} \triangleq \exp(-hS_{\nu,t}) = \exp(-hS_{\nu,t-1}) \times \exp(-hs_{\nu,t}) \quad (15)$$

if ν is a leaf node, and

$$\mathcal{U}_{\nu,t} \triangleq q_s\mathcal{U}_{\nu_l,t}\mathcal{U}_{\nu_r,t} + (1-q_s)\exp(-hS_{\nu,t}), \quad (16)$$

if ν is not a leaf node,

where $s_{\nu,t}$ and $S_{\nu,t}$ are the stochastic and the accumulated NP losses of the OLNP running at the node ν and $0 \leq q_s \leq 1$ is the node split probability of the tree that is related to the expert probabilities $q_{i,t} = \frac{U_{i,t-1}}{\sum_{j=1}^{N_q} U_{j,t-1}}$ with $1-q_s$ being the probability of choosing the expert of the complete observation space (single piece) at time $t = 1$, as further explained below. Importantly, at the root node $O = \nu_1$, we have $\mathcal{U}_{O,t} = \sum_{i=1}^{N_q} U_{i,t}$ (so the name "combined performance"), and note that these two variables $S_{\nu,t}, \mathcal{U}_{\nu,t}$ can be updated bottom up from the leaf ν_D to the root ν_0 at each time for the observation x_t with complexity in the order $O(D)$.

Now, let us consider the node choosing probability $\tilde{q}_{i,t}$, which is verbally the sum of the probabilities of the experts (consider the corresponding partitions over the tree) containing the node ν_i as the leaf in its corresponding pruned tree. For the root node, it is simply $\tilde{q}_{0,t} = \frac{(1-q_s)\exp(-hS_{O,t-1})}{\mathcal{U}_{O,t}}$ and for the other nodes,

$$\tilde{q}_{i,t} = \frac{q_s}{1-q_s}\tilde{q}_{i-1,t}\mathcal{U}_{\nu_{i,s},t}\mathcal{U}_{\nu_i,t}\left((1-q_s)1_{\{\nu_i=\nu_D\}} + 1_{\{\nu_i\neq\nu_D\}}\right),$$

where $\nu_{i,s}$ is the sibling node of ν_i. As a result, the computation of the lumped probabilities $\tilde{q}_{i,t}$'s can be completed at each time for the observation x_t in two passes of complexity $O(D)$: (1) first, bottom up computation from the leaf ν_D to the root ν_0 for the variables $S_{\nu,t}, \mathcal{U}_{\nu,t}$ as described above, (2) and then, top down computation from the root ν_0 to the leaf ν_D to finally get $\tilde{q}_{i,t}$'s along the way. We point out that split probability q_s draws the initial expert probabilities $q_{j,1}$'s as the following. Considering back our depth-2 example,

$\mathcal{P}_1 = \{[0,4]\} \rightarrow q_{1,1} = 1 - q_s$ (single piece), $\mathcal{P}_2 = \{[0,2],(2,4]\} \rightarrow q_{2,1} = q_s(1-q_s)^2$ (two pieces), $\mathcal{P}_3 = \{[0,2],(2,3],(3,4]\} \rightarrow q_{3,1} = q_s^2(1-q_s)$ (three pieces), $\mathcal{P}_4 = \{[0,1],(1,2],(2,4]\} \rightarrow q_{4,1} = q_s^2(1-q_s)$ (three pieces), $\mathcal{P}_5 = \{[0,1],(1,2],(2,3],(3,4]\} \rightarrow q_{4,1} = q_s^3$ (four pieces), where the experts with more pieces obtain smaller probabilities, intuitively and as desired. Typically $q_s \leq \frac{1}{2}$. This finalizes the description of the versatile implementation of our proposed randomized algorithm g_t for which a pseudo-code table is also given below. Note that this $O(D)$ computationally efficient computation is a direct outcome of the context tree weighting method that we use in this study and briefly summarize here, all further details can be followed from [25], [31].

REFERENCES

[1] Xin Tong, Yang Feng, and Anqi Zhao. A survey on neyman-pearson classification and suggestions for future research. *Wiley Interdisciplinary Reviews: Computational Statistics*, 8(2):64–81, 2016.

[2] Shahin, Mostafa, Usman Zafar, and Beena Ahmed. The automatic detection of speech disorders in children: Challenges, opportunities, and preliminary results. *IEEE Journal of Selected Topics in Signal Processing*, 14(2):400–412, 2019.

[3] Wang, Lijia, YX Rachel Wang, Jingyi Jessica Li, and Xin Tong. Hierarchical Neyman-Pearson Classification for Prioritizing Severe Disease Categories in COVID-19 Patient Data. *Journal of the American Statistical Association*, 119(545):39–51, 2024.

[4] Su, Jianpo and Shen, Hui and Peng, Limin and Hu, Dewen. Few-shot domain-adaptive anomaly detection for cross-site brain images. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 46(3):1819–1835, 2021.

[5] Bai, Yue and Safikhani, Abolfazl and Michailidis, George. Hybrid modeling of regional COVID-19 transmission dynamics in the US. *IEEE Journal of Selected Topics in Signal Processing*, 16(2):261–275, 2022.

[6] Cao, Congqi and Zhang, Hanwen and Lu, Yue and Wang, Peng and Zhang, Yanning. Scene-dependent prediction in latent space for video anomaly detection and anticipation. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 2024.

[7] Luo, Weixin and Liu, Wen and Lian, Dongze and Gao, Shenghua. Future frame prediction network for video anomaly detection. *IEEE transactions on pattern analysis and machine intelligence*, 44(11):7505–7520, 2021.

[8] Mishra, Sandhya and Soni, Devpriya. Dsmishsms-a system to detect smishing sms. *Neural Computing and Applications*, 35(7):4975–4992, 2023.

[9] Han, Xizewen and Zheng, Huangjie and Zhou, Mingyuan. Card: Classification and regression diffusion models. *Advances in Neural Information Processing Systems*, 35:18100–18115, 2022.

[10] Xue, Jian and Li, HongEn and Pan, Meiyan and Liu, Jun. Adaptive persymmetric detection for radar targets in correlated CG-LN sea clutter. *IEEE Transactions on Geoscience and Remote Sensing*, 2023.

[11] Nerea del-Rey-Maestre, María-Pilar Jarabo-Amores, David Mata-Moya, José-Luis Bárcena-Humanes and Pedro Gómez del Hoyo. Machine learning techniques for coherent CFAR detection based on statistical modeling of UHF passive ground clutter. *IEEE Journal of Selected Topics in Signal Processing*, 12(1):104–118, 2017.

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

[13] Zhao, Xuandong and Ananth, Prabhanjan and Li, Lei and Wang, Yu-Xiang. Provable robust watermarking for ai-generated text. *arXiv preprint arXiv:2306.17439*, 2023.

[14] Chakraborty, Souradip and Bedi, Amrit Singh and Zhu, Sicheng and An, Bang and Manocha, Dinesh and Huang, Furong. On the possibilities of ai-generated text detection. *arXiv preprint arXiv:2304.04736*, 2023.

[15] Pawelczyk, Martin and Neel, Seth and Lakkaraju, Himabindu. In-context unlearning: Language models as few shot unlearners. *arXiv preprint arXiv:2310.07579*, 2023.

[16] Seniha Esen Yuksel, Joseph N Wilson, and Paul D Gader. Twenty years of mixture of experts. *IEEE transactions on neural networks and learning systems*, 23(8):1177–1193, 2012.

[17] Necati Demir and Gökhan Dalkiliç. Modified stacking ensemble approach to detect network intrusion. *Turkish Journal of Electrical Engineering & Computer Sciences*, 26(1):418–433, 2018.

[18] Mark A Davenport, Richard G Baraniuk, and Clayton D Scott. Tuning support vector machines for minimax and neyman-pearson classification. *IEEE transactions on pattern analysis and machine intelligence*, 32(10):1888–1898, 2010.

[19] Bin Gu, Victor S Sheng, Keng Yeow Tay, Walter Romano, and Shuo Li. Cross validation through two-dimensional solution surface for cost-sensitive svm. *IEEE transactions on pattern analysis and machine intelligence*, 39(6):1103–1121, 2016.

[20] David Casasent and Xue-wen Chen. Radial basis function neural networks for nonlinear fisher discrimination and neyman–pearson classification. *Neural networks*, 16(5-6):529–535, 2003.

[21] Xin Tong, Yang Feng, and Jingyi Jessica Li. Neyman-pearson classification algorithms and np receiver operating characteristics. *Science advances*, 4(2):eaao1659, 2018.

[22] Gilles Gasso, Aristidis Pappaioannou, Marina Spivak, and Léon Bottou. Batch and online learning algorithms for nonconvex neyman-pearson classification. *ACM transactions on intelligent systems and technology (TIST)*, 2(3):1–19, 2011.

[23] Basarbatu Can and Huseyin Ozkan. A neural network approach for online nonlinear neyman-pearson classification. *IEEE Access*, 8:210234–210250, 2020.

[24] Ali Rahimi, Benjamin Recht, et al. Random features for large-scale kernel machines. In *NIPS*, volume 3, page 5. Citeseer, 2007.

[25] Frans MJ Willems, Yuri M Shtarkov, and Tjalling J Tjalkens. The context-tree weighting method: Basic properties. *IEEE transactions on information theory*, 41(3):653–664, 1995.

[26] I Papageorgiou, Ioannis Kontoyiannis, L Mertzanis, A Panotopoulou, and M Skoularidou. Revisiting context-tree weighting for bayesian inference. In *2021 IEEE International Symposium on Information Theory (ISIT)*, pages 2906–2911. IEEE, 2021.

[27] Fei Mi and Boi Faltings. Adaptive sequential recommendation using context trees. In *IJCAI*, pages 4018–4019, 2016.

[28] Marco Cuturi and Jean-Philippe Vert. The context-tree kernel for strings. *Neural Networks*, 18(8):1111–1123, 2005.

[29] Nikolaos Tziortziotis, Christos Dimitrakakis, and Konstantinos Blekas. Cover tree bayesian reinforcement learning. *Journal of Machine Learning Research*, 15:2313–2335, 2014.

[30] Suleyman S Kozat, Andrew C Singer, and Georg Christoph Zeitler. Universal piecewise linear prediction via context trees. *IEEE Transactions on Signal Processing*, 55(7):3730–3745, 2007.

[31] Huseyin Ozkan, N Denizcan Vanli, and Suleyman S Kozat. Online classification via self-organizing space partitioning. *IEEE Transactions on Signal Processing*, 64(15):3895–3908, 2016.

[32] Mohammadreza Mohaghegh Neyshabouri, Kaan Gokcesu, Hakan Gokcesu, Huseyin Ozkan, and Suleyman Serdar Kozat. Asymptotically optimal contextual bandit algorithm using hierarchical structures. *IEEE transactions on neural networks and learning systems*, 30(3):923–937, 2018.

[33] Basarbatu Can and Huseyin Ozkan. Neyman–pearson classification via context trees. *2020 28th Signal Processing and Communications Applications Conference (SIU)*, pages 1–4, 2020.

[34] Mine Kerpicci, Huseyin Ozkan, and Suleyman Serdar Kozat. Online anomaly detection with bandwidth optimized hierarchical kernel density estimators. *IEEE Transactions on Neural Networks and Learning Systems*, 2020.

[35] Chih-Chung Chang and Chih-Jen Lin. Libsvm: A library for support vector machines. *ACM Transactions on Intelligent Systems and Technology*, 2(3):1–27, 2011.

[36] Dheeru Dua and Casey Graff. Uci machine learning repository (2017). *URL http://archive. ics. uci. edu/ml*, 37, 2017.

[37] Clayton Scott. Performance measures for neyman–pearson classification. *IEEE Transactions on Information Theory*, 53(8):2852–2863, 2007.

[38] Gope, Jyotirmoy and Jain, Sanjay Kumar. A survey on solving cold start problem in recommender systems. *2017 International Conference on Computing, Communication and Automation (ICCCA)*, 133–138, 2017.

[39] Cesa-Bianchi, Nicolo and Lugosi, Gábor. Prediction, learning, and games. *Cambridge university press*, 2006.

[40] Lu, Jie and Liu, Anjin and Dong, Fan and Gu, Feng and Gama, Joao and Zhang, Guangquan. Learning under concept drift: A review. *IEEE transactions on knowledge and data engineering*, 31(12):2346–2363, 2018.

[41] Ozkan, Huseyin and Ozkan, Fatih and Kozat, Suleyman S. Online anomaly detection under markov statistics with controllable type-i error. *IEEE Transactions on Signal Processing*, 64(6):1435–1445, 2015.