Int. J. Nonlinear Anal. Appl. In Press, 1–14
ISSN: 2008-6822 (electronic)
http://dx.doi.org/10.22075/ijnaa.2025.35682.5305

A new variant of online binary passive-aggressive algorithm for interval data

Marziyeh Talebi[a], Mojtaba Baymani[a,*], Nima Salehi-M.[b]

[a]Department of Mathematics, Faculty of Engineering Science, Quchan University of Technology, Quchan, Iran

[b]Department of Computer Engineering, Faculty of Electrical and Computer Engineering, Quchan University of Technology, Quchan, Iran

(Communicated by Mouquan Shen)

Abstract

Online algorithms process data sequentially, updating their parameters as new data points arrive to minimize prediction errors. However, traditional online algorithms are not inherently designed to handle interval data, which poses a significant challenge in many real-world applications. In this paper, we propose a novel online binary classification algorithm specifically tailored for interval data. We introduce a new loss function that extends the online passive-aggressive (PA) framework, enabling it to effectively handle both ordinal and interval data. The proposed loss function is computationally efficient and provides a robust solution for online learning in the presence of interval data. We provide theoretical guarantees for the algorithm, including cumulative squared loss bounds and relative loss bounds, which demonstrate its effectiveness. Empirical evaluations on multiple datasets show that our method achieves competitive performance compared to existing online algorithms while uniquely addressing the challenges posed by interval data.

Keywords: Online Learning, Binary Classification, Interval Data, Optimization Problem, Loss Function, Passive-Aggressive Algorithms
2020 MSC: 68Q32, 68T05, 65K10

1 Introduction

Online learning [19, 22, 23] is a class of efficient and effective algorithms designed for solving large-scale problems. Unlike traditional batch learning methods [34], which require all training samples to be available at once, online learning algorithms process data sequentially, updating their prediction models incrementally as new data arrives. The sequential nature of online learning provides several key advantages, including computational efficiency, scalability, and adaptability to dynamic data distributions. These properties make online learning particularly well-suited for large-scale and real-time applications, where data streams continuously and the model must adapt to evolving patterns. Despite these advantages, most existing online learning algorithms are designed under the assumption that input data is precise, represented as single points in a high-dimensional space (i.e., $\boldsymbol{x} \in \mathbb{R}^d$). However, in many real-world scenarios, data is inherently imprecise or uncertain, and it is more appropriate to represent inputs as intervals. Interval

*Corresponding author
Email addresses: m.talebi59@chmail.ir (Marziyeh Talebi), m_baymani@qiet.ac.ir (Mojtaba Baymani), n.salehi@qiet.ac.ir (Nima Salehi-M.)

Received: October 2024 *Accepted:* January 2025

data captures the uncertainty or variability in measurements, which is common in applications such as sensor networks, financial forecasting, and medical diagnostics. Formally, interval data can be defined as:

$$\hat{\boldsymbol{x}} \triangleq [\boldsymbol{x} - \Delta_1, \boldsymbol{x} + \Delta_2], \tag{1.1}$$

where $\Delta_i \in \mathbb{R}^d$ for $i \in \{1, 2\}$ are non-negative vectors representing the bounds of uncertainty or imprecision. While several batch learning methods have been proposed to handle interval data [2, 4, 1, 31, 36], these approaches are not directly applicable to the online learning setting. The sequential and incremental nature of online learning requires specialized algorithms that can update models efficiently without revisiting past data.

In this paper, we address this critical gap by proposing a novel online binary classification algorithm specifically designed to handle interval data. Our approach extends the well-established online passive-aggressive (PA) framework [11] to accommodate the interval nature of input data. By introducing a new loss function that accounts for the uncertainty in interval data, our algorithm provides a robust and computationally efficient solution for online learning in the presence of imprecise inputs. The core idea of our method is illustrated in Figure 1, where the hyperplane obtained until step t is updated in step $t+1$ based on the arrival of new interval data $\hat{\boldsymbol{x}}$. This update aims to minimize the classification error by satisfying the most constraints imposed by the interval data. We provide theoretical guarantees for the proposed algorithm, including cumulative squared loss bounds and relative loss bounds, which demonstrate its effectiveness in minimizing prediction errors over time.

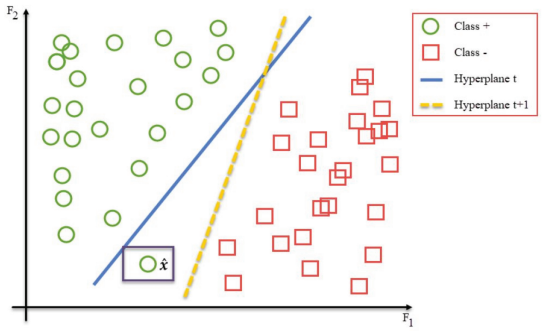

Figure 1: The idea of online updating of hyperplane based on a new interval data arriving $\hat{\boldsymbol{x}}$.

2 Related Works

Batch kernel methods, such as Support Vector Machines (SVMs), have been extensively utilized in machine learning due to their robustness and effectiveness in handling complex datasets [10, 33, 35]. However, in real-world scenarios, training data often contains inherent uncertainties and measurement errors, making it challenging to determine exact values for variables. In such cases, interval-based representations are employed to model incomplete or imprecise information, where intervals are treated as disjunctive sets [21]. To address the challenges posed by imprecise and uncertain data, various machine learning strategies have been proposed. These strategies aim to extend traditional learning algorithms to handle interval data effectively [6, 9, 17]. Specifically, several algorithms have been developed to process interval data, which is a specialized form of uncertain data [1, 36, 14, 13].

Recently, an "Interval Support Vector Machine" (ISVM) tailored for interval data was introduced [2]. This method was designed for batch classification tasks, but it is not directly applicable to online learning scenarios. Nevertheless, integrating batch kernel methods with online learning algorithms has garnered significant attention, especially in the context of large-scale problems where scalability and real-time adaptation are critical. These integrated approaches, known as "Online Kernel Learning" (OKL), have demonstrated remarkable effectiveness [44].

Several variants of online kernel learning algorithms exist, including the Perceptron, Voted Perceptron [18, 32], Online Stochastic Gradient Descent (OGD) [23], Passive-Aggressive (PA) [11], and Truncated OGD [24]. The Passive-Aggressive (PA) framework has been enhanced with a fuzzy adaptation, enabling more effective modeling of uncertainty in online classification tasks [37]. This adaptation plays a key role in improving the robustness of the algorithm, particularly in scenarios where the input data is noisy or unreliable. Recent advancements also indicate promising methodologies in active learning, particularly the development of a new algorithm tailored for trapezoidal data streams, which, similar to interval data, requires robust query strategies and can significantly improve classification performance [26]. Asynchronous parallel methods in fuzzy stochastic gradient descent can further optimize high-dimensional and

incomplete data representations, establishing a promising approach for enhancing overall model performance [30]. Moreover, several studies have addressed multi-label classification in online learning settings, prominently featuring advancements in the passive-aggressive framework. A newly proposed algorithm effectively learns label correlations while performing multi-label classification online [26, 41]. Furthermore, learning label correlations dynamically is crucial for adapting multi-label classifiers efficiently as highlighted in [42]. This synergy of learning correlations and adapting to new data instances also extends to a specialized approach that maintains computational efficiency through budget adherence [40].

Despite their advantages, these algorithms face a critical limitation known as the "*curse of kernelization.*" This phenomenon refers to the unbounded growth of the model size as the training data size increases, leading to significant challenges in computational complexity, memory usage, and training time. The primary challenges associated with these methods include scalability, overfitting, support vector (SV) maintenance in memory, SV size reduction, sparsity, and the reduction of both test and training time. Additionally, achieving an optimal convergence rate (whether fixed or variable) remains a critical concern. To mitigate these challenges, several well-known algorithms have been developed, focusing primarily on reducing the number of support vectors. Three main strategies are employed to achieve this: removal, projection, and merging. Notable algorithms that utilize these strategies include Budget Perceptron [12], Budget Online Kernel Gradient Descent (BOGD) [43], Random Budget Perceptron (RBP) [5], Space Implicit Online Learning with Kernel (SILK) [8], Forgetron [15], and Budget PA Simple (BPA-Simple) [39]. These algorithms primarily rely on the removal strategy. Other approaches, such as Projectron, Projectron++ [28, 29], BPA-S, BPA-P, BPA-NN [39], and BSGD-R, BSGD-P, BSGD-M [38], employ projection and merging strategies. Collectively, these algorithms are referred to as "Budget Online Kernel Learning."

Additionally, the exploration of random scaling in stochastic gradient descent has yielded insights into maintaining low computational overhead while processing massive data streams, which can be highly beneficial for interval data applications [25]. To address these issues, various online algorithms have been proposed, focusing on both the traditional removal strategy and more innovative methods. Recent findings on adversarial sparse online learning lend further insight into enhancing the robustness of algorithms in uncertain environments [7].

It is important to note that none of the aforementioned online algorithms are capable of handling interval data directly. However, our proposed method, which extends these approaches, will be compared with these algorithms. In our proposed method, we have not incorporated strategies for reducing the number of support vectors, as this lies beyond the scope of the current study. Nevertheless, such strategies can be integrated into our method in future work. Furthermore, our proposed approach can be adapted to address interval data in other online learning algorithms, paving the way for more robust and scalable solutions in the future.

3 Proposed Method

In this section, we discuss the problem of online binary classification learning for interval data and provide a detailed analysis. Online binary classification occurs in a sequence of rounds, with one iteration for each incoming data. The algorithm observes the new input data in each round and predicts its label, which is either $+1$ or -1. We denote the input for the algorithm in round t by $\hat{\boldsymbol{x}}_t$, which we consider as interval data, defined as follows.

Definition: A sample interval value is represented by $\hat{\boldsymbol{x}}_t$ and is defined as $\hat{\boldsymbol{x}}_t \in \left[\boldsymbol{x}_t^l, \boldsymbol{x}_t^u\right]$, where $\boldsymbol{x}_t^l, \boldsymbol{x}_t^u \in \mathbb{R}^n$ and $(\boldsymbol{x}_t^l)_i \leq (\hat{\boldsymbol{x}}_t)_i \leq (\boldsymbol{x}_t^u)_i$.

Geometrically, this interval sample forms a hypercube or hyper-rectangle in the input space. Assume $\hat{\boldsymbol{x}}_t$ corresponds to a unique label $y_t \in \{+1, -1\}$. In round t, we express the input pair with its corresponding label as $(\hat{\boldsymbol{x}}_t, y_t)$ and refer to it as an "*interval value training sample*". In online binary classification problems, the traditional Passive-Aggressive algorithm [11] is extended to deal with the interval data. The proposed algorithm utilizes the newly arrived interval input pair to enhance its prediction rule. We restrict our discussion to binary classification based on a weight vector $\boldsymbol{w} \in \mathbb{R}^n$, taking the form $sign(\boldsymbol{w}.\boldsymbol{x})$. The algorithm maintains the weight vector in its internal memory and updates it in each round. The goal is to learn the weight vector $\boldsymbol{w}_t$ based on the arrival of new interval input data, where $\boldsymbol{w}_t$ denotes the weight vector in round t. We initially consider the weight vector $\boldsymbol{w}$ as zero, i.e., $\boldsymbol{w}_1 = (0, \cdots, 0)^T$. In round t, the updated weight vector $\boldsymbol{w}_{t+1}$ is the solution of the following optimization problem:

$$\boldsymbol{w}_{t+1} = \arg\min_{\boldsymbol{w}} \|\boldsymbol{w} - \boldsymbol{w}_t\|^2 \quad \text{subject to} \quad l_{M_t}(\boldsymbol{w}; (\hat{\boldsymbol{x}}_t, y_t)) = 0, \quad \boldsymbol{w} \in \mathbb{R}^n \tag{3.1}$$

where $\|.\|$ denotes the 2-norm, and l_{M_t} is a loss function for round t, defined as:

$$l_{M_t} \triangleq \max\left\{0, 1 - y_t < \boldsymbol{w}, \boldsymbol{x}_t^l >, 1 - y_t < \boldsymbol{w}, \boldsymbol{x}_t^u >\right\} \tag{3.2}$$

where $< .,. >$ represents the inner-product of two vectors. This modified loss function is based on the hinge loss function for binary classification and is essential for determining the maximum relative deviation in the neighborhood of sample $\hat{\boldsymbol{x}}_t$, where

$$\hat{\boldsymbol{x}}_t \in [\boldsymbol{x}_t - \Delta_1, \boldsymbol{x}_t + \Delta_2] \in \left[\boldsymbol{x}_t^l, \boldsymbol{x}_t^u\right]$$

and Δ_1, Δ_2 are non-negative vectors with the same dimension as $\boldsymbol{x}_t$. The weight vector $\boldsymbol{w}_t$ is updated based on this loss function. If the loss function value, l_{M_t}, equals zero, the weights remain unchanged: $\boldsymbol{w} = \boldsymbol{w}_t$. If $l_{M_t} > 0$, the weights require updating. Now assume the loss function is positive, i.e., $l_{M_t} > 0$. In this case, the Lagrange function of the problem is formulated as follows:

$$L(\boldsymbol{w}, \eta) = \frac{1}{2}\|\boldsymbol{w} - \boldsymbol{w}_t\|^2 + \eta l_{M_t}(\boldsymbol{w}; (\hat{\boldsymbol{x}}_t, y_t)) \tag{3.3}$$

According to the KKT conditions and by setting the partial derivatives of L with respect to the elements of $\boldsymbol{w}$ to zero, it can be written as:

$$\frac{\partial L(\boldsymbol{w}, \eta)}{\partial \boldsymbol{w}} = 0 \Rightarrow \boldsymbol{w} = \boldsymbol{w}_t + \eta y_t \boldsymbol{x}_t^k. \tag{3.4}$$

where

$$\boldsymbol{x}_t^k = \begin{cases} \boldsymbol{x}_t^l & \text{if} \quad l_{M_t} = 1 - y_t < \boldsymbol{w}, \boldsymbol{x}_t^l > \\ \boldsymbol{x}_t^u & \text{if} \quad l_{M_t} = 1 - y_t < \boldsymbol{w}, \boldsymbol{x}_t^u > \end{cases}$$

By substituting $\boldsymbol{w}$ in equation (3.3), we obtain:

$$L(\eta) = -\frac{1}{2}\eta^2 \|\boldsymbol{x}_t^k\|^2 + \eta(1 - y_t < \boldsymbol{w}_t, \boldsymbol{x}_t^k >)$$

Taking the derivative of the expression with respect to η and setting it to zero gives:

$$\frac{\partial L(\eta)}{\partial \eta} = 0 \Rightarrow -\eta\|\boldsymbol{x}_t^k\|^2 + (1 - y_t < \boldsymbol{w}_t, \boldsymbol{x}_t^k >) = 0 \quad \Rightarrow \quad \eta = \frac{1 - y_t < \boldsymbol{w}_t, \boldsymbol{x}_t^k >}{\|\boldsymbol{x}_t^k\|^2}.$$

Since η is a variable step-size with respect to time t, we denote it as η_t. The optimization problem (3.1) has the following solution:

$$\boldsymbol{w}_{t+1} = \boldsymbol{w}_t + \eta_t y_t \boldsymbol{x}_t^k \quad \text{where} \quad \eta_t = \frac{l_{M_t}}{\|\boldsymbol{x}_t^k\|^2}.$$

In the proposed method, the variable step-size η_t is computed at time step t as follows. Based on the defined loss function in equation (3.2), if $l_{M_t} = 1 - y_t < \boldsymbol{w}_t, \boldsymbol{x}_t^l >$, we use $\eta_t = \dfrac{l_{M_t}}{\|\boldsymbol{x}_t^l\|^2}$ and otherwise if $l_{M_t} = 1 - y_t < \boldsymbol{w}_t, \boldsymbol{x}_t^u >$, we use $\eta_t = \dfrac{l_{M_t}}{\|\boldsymbol{x}_t^u\|^2}$. The pseudo-code for the above algorithm is summarized in Algorithm 1.

The proposed method is extended to a nonlinear prediction model. The decision function for the nonlinear problem is defined as follows:

$$g_t(\boldsymbol{x}_t) = sign\big(< \boldsymbol{w}, \phi(\boldsymbol{x}_t) > \big) = sign\big(f_t(\boldsymbol{x}_t)\big) \tag{3.5}$$

where $f_t(\boldsymbol{x}_t) \triangleq \langle \boldsymbol{w}, \phi(\boldsymbol{x}_t) \rangle$, and $\phi(\cdot)$ is an explicit nonlinear feature map function that transforms data from the *input space* to the *feature space*. Given that the dimensions of the feature space can be very high or even infinite, explicit feature maps are impractical. Kernel methods address this limitation by performing computations directly in the input space, utilizing the implicit feature map. This technique is commonly referred to as the "kernel trick" (i.e., $k(\boldsymbol{x}, \boldsymbol{x}') = \langle \phi(\boldsymbol{x}), \phi(\boldsymbol{x}') \rangle$). Consequently, the loss function in equation (3.2) is modified as follows:

$$\begin{aligned} l_{M_t} &\triangleq \max\{0, 1 - y_t < \boldsymbol{w}, \phi(\boldsymbol{x}_t^l) >, 1 - y_t < \boldsymbol{w}, \phi(\boldsymbol{x}_t^u) >\} \\ &= \max\{0, 1 - y_t f_t(\boldsymbol{x}_t^l), 1 - y_t f_t(\boldsymbol{x}_t^u)\} \end{aligned} \tag{3.6}$$

and the Lagrangian problem, as described in Equation (3.3), can be reformulated as follows:

$$L(\boldsymbol{w}, \eta) = \frac{1}{2}\|\boldsymbol{w} - \boldsymbol{w}_t\|^2 + \eta l_{M_t} \tag{3.7}$$

Algorithm 1 Online linear binary classification for interval data

Initialize $\boldsymbol{w}_0 = (0, \cdots, 0)$;
for $t = 1, 2, \ldots$ **do**
  Receive instance $\boldsymbol{x}_t \in R^n$;
  Predict $\hat{y}_t = sign(< \boldsymbol{w}_t, \boldsymbol{x}_t >)$;
  Receive correct label: $y_t \in \{-1, +1\}$;
  Suffer loss: $l_{M_t} = \mathbf{max}\{0, 1 - y_t < \boldsymbol{w}_t, \boldsymbol{x}_t^l >, 1 - y_t < \boldsymbol{w}_t, \boldsymbol{x}_t^u >\}$;
  if $l_t > 0$ **then**
    update:
    if $l_{M_t} = 1 - y_t < \boldsymbol{w}_t, \boldsymbol{x}_t^l >$ **then**
      set $\eta_t = \dfrac{l_{M_t}}{\|\boldsymbol{x}_t^l\|^2}$;
      $k = 'l'$; \\ ▶ k is superscript of $\boldsymbol{x}_t^k$
    else
      set $\eta_t = \dfrac{l_{M_t}}{\|\boldsymbol{x}_t^u\|^2}$;
      $k = 'u'$; \\ ▶ k is superscript of $\boldsymbol{x}_t^k$
    end if
  end if
  update: $\boldsymbol{w}_{t+1} = \boldsymbol{w}_t + \eta_t y_t \boldsymbol{x}_t^k$;
end for

Based on the KKT conditions and $\dfrac{\partial L(\boldsymbol{w}, \eta)}{\partial \boldsymbol{w}} = 0$, we obtain:

$$\boldsymbol{w} = \boldsymbol{w}_t + \eta_t y_t \phi(\boldsymbol{x}_t^k), \tag{3.8}$$

$$\eta_t = \frac{l_{M_t}}{\|\phi(\boldsymbol{x}_t^k)\|^2} \tag{3.9}$$

where

$$\boldsymbol{x}_t^k = \begin{cases} \boldsymbol{x}_t^l & if \quad l_{M_t} = 1 - y_t < \boldsymbol{w}, \phi(\boldsymbol{x}_t^l) > \\ \boldsymbol{x}_t^u & if \quad l_{M_t} = 1 - y_t < \boldsymbol{w}, \phi(\boldsymbol{x}_t^u) > \end{cases}$$

According to the loss function specified in equation (3.6), if $l_{M_t} = 1 - y_t f_t(\boldsymbol{x}_t^l)$, we set $\eta_t = l_{M_t} / \|\phi(\boldsymbol{x}_t^l)\|^2$. Otherwise if $l_{M_t} = 1 - y_t f_t(\boldsymbol{x}_t^u)$, we set $\eta_t = l_{M_t} / \|\phi(\boldsymbol{x}_t^u)\|^2$. Considering that the initial weights are zero, the weight vector at time t can be expressed as:

$$\boldsymbol{w}_t = \sum_{i=1}^{t-1} \eta_i y_i \phi(\boldsymbol{x}_i^k). \tag{3.10}$$

Given that $\phi(.)$ is unknown and potentially high-dimensional, direct computation of the weight vector is intractable. To address this challenge, we multiply both sides of equation (3.10) by $\phi(\boldsymbol{x}_t')$ and utilize the kernel trick $k(\boldsymbol{x}, \boldsymbol{x}') =< \phi(\boldsymbol{x}), \phi(\boldsymbol{x}') >$, obtaining:

$$f_t(\boldsymbol{x}_t) = \sum_{i=1}^{t-1} \eta_i y_i k(\boldsymbol{x}_i^k, \boldsymbol{x}_t^{k'}), \tag{3.11}$$

$$\eta_t = \frac{1 - y_t f_t(\boldsymbol{x}_t^k)}{k(\boldsymbol{x}_t^k, \boldsymbol{x}_t^k)}. \tag{3.12}$$

where $\|\phi(\boldsymbol{x}_t)\|^2 =< \phi(\boldsymbol{x}_t), \phi(\boldsymbol{x}_t) >= k(\boldsymbol{x}_t, \boldsymbol{x}_t)$. Algorithm 2 summarizes the pseudo-code for the proposed online nonlinear classification learning method, specifically designed to process interval input data.

4 Analysis

In this section, we obtain the relative loss bounds for the proposed algorithm. Specifically, we compare the cumulative squared loss attained by the algorithm on the same sequence of samples with the loss attained by an

Algorithm 2 Online nonlinear binary classification for interval data

for $t = 1, 2, \ldots$ **do**
  Receive instance $\boldsymbol{x}_t \in R^n$;
  Predict $g_t(\boldsymbol{x}_t)$ according to (3.5);
  Receive correct label: $y_t \in \{-1, +1\}$;
  Suffer loss: $l_{M_t} = \mathbf{max}\{0, 1 - y_t f_t(\boldsymbol{x}_t^l), 1 - y_t f_t(\boldsymbol{x}_t^u)\}$;
  if $l_{M_t} > 0$ **then**
    update:
    if $l_{M_t} = 1 - y_t f_t(\boldsymbol{x}_t^l)$ **then**
      set $\eta_t = \dfrac{l_{M_t}}{k(\boldsymbol{x}_t^l, \boldsymbol{x}_t^l)}$;
      $k = 'l'$; \\ ▶ k is superscript of $\boldsymbol{x}_t^k$
    else
      set $\eta_t = \dfrac{l_{M_t}}{k(\boldsymbol{x}_t^u, \boldsymbol{x}_t^u)}$;
      $k = 'u'$; \\ ▶ k is superscript of $\boldsymbol{x}_t^k$
    end if
  end if
  update: $f_t(\boldsymbol{x}_t) = \sum_{i=1}^{t-1} \eta_i y_i k(\boldsymbol{x}_i^k, \boldsymbol{x}_t^{k'}), \quad k, k' = 'l', 'u'$;
end for

arbitrary fixed classification function of the form $sign(\boldsymbol{u}.\boldsymbol{x})$, where $\boldsymbol{u}$ is an arbitrary vector in $\mathbb{R}^n$. For simplicity, we introduce the following notation: 1) l_{M_t} denotes the loss suffered by our algorithm at step t and 2) l_t^* denotes the loss incurred by the arbitrarily fixed predictor, which is used for performance comparison. This quantity is defined as follows:

$$l_t^* = l(\boldsymbol{u}; (\boldsymbol{x}_t, y_t)) = \max\{0, 1 - y_t < \boldsymbol{u}, \boldsymbol{x}_t >\} \tag{4.1}$$

We begin this section by generalizing Lemma 4.1, which serves as the foundation for deriving the loss bounds for the algorithm. Subsequently, we apply this generalized lemma to establish the desired bounds, thereby providing a theoretical guarantee for the algorithm's performance.

Lemma 4.1. Let $(\hat{\boldsymbol{x}}_1, y_1), \ldots, (\hat{\boldsymbol{x}}_T, y_T)$ be a sequence of samples where $\hat{\boldsymbol{x}}_t \in \mathbb{R}^n$ and $y_t \in \{+1, -1\}$ for all t. Let η_t denote the step size defined in the algorithm. Then, using the notation defined in (4.1), the following bound holds for every $\boldsymbol{u} \in \mathbb{R}^n$:

$$\sum_{t=1}^{T} \eta_t \left(2 l_{M_t} - \eta_t \|\boldsymbol{x}_t^k\|^2 - 2 l_t^*\right) \leq \|\boldsymbol{u}\|^2$$

Proof . Define $\triangle_t$ as $\triangle_t = \|\boldsymbol{w}_t - \boldsymbol{u}\|^2 - \|\boldsymbol{w}_{t+1} - \boldsymbol{u}\|^2$. We prove the lemma by summing $\triangle_t$ over $t = 1, \ldots, T$ and bounding the sum from above and below. First, note that $\sum_t \triangle_t$ is a telescoping sum, which collapses as follows:

$$\sum_{t=1}^{T} \triangle_t = \sum_{t=1}^{T} \left(\|\boldsymbol{w}_t - \boldsymbol{u}\|^2 - \|\boldsymbol{w}_{t+1} - \boldsymbol{u}\|^2\right) = \|\boldsymbol{w}_1 - \boldsymbol{u}\|^2 - \|\boldsymbol{w}_{T+1} - \boldsymbol{u}\|^2.$$

Since $\boldsymbol{w}_1$ is defined as the zero vector, and $\|\boldsymbol{w}_{T+1} - \boldsymbol{u}\|^2 \geq 0$, we obtain the following inequality:

$$\sum_{t=1}^{T} \triangle_t \leq \|\boldsymbol{u}\|^2. \tag{4.2}$$

Next, we bound $\triangle_t$ from below. According to the definition of the loss function:

$$l_{M_t} = \max\left\{0, 1 - y_t \langle \boldsymbol{w}_t, \boldsymbol{x}_t^l \rangle, 1 - y_t \langle \boldsymbol{w}_t, \boldsymbol{x}_t^u \rangle\right\},$$

if $l_{M_t} = 0$, then $\eta_t = 0$, and therefore $\triangle_t = 0$. Thus, we focus on the rounds where $l_{M_t} > 0$. Using the update rule:

$\boldsymbol{w}_{t+1} = \boldsymbol{w}_t + \eta_t y_t \boldsymbol{x}_t^k$, we can express $\triangle_t$ as:

$$\begin{aligned}\triangle_t &= \|\boldsymbol{w}_t - \boldsymbol{u}\|^2 - \|\boldsymbol{w}_{t+1} - \boldsymbol{u}\|^2 \\ &= \|\boldsymbol{w}_t - \boldsymbol{u}\|^2 - \|\boldsymbol{w}_t - \boldsymbol{u} + \eta_t y_t \boldsymbol{x}_t^k\|^2 \\ &= \|\boldsymbol{w}_t - \boldsymbol{u}\|^2 - \left(\|\boldsymbol{w}_t - \boldsymbol{u}\|^2 + 2\eta_t y_t \langle \boldsymbol{w}_t - \boldsymbol{u}, \boldsymbol{x}_t^k\rangle + \eta_t^2 \|\boldsymbol{x}_t^k\|^2\right) \\ &= -2\eta_t y_t \langle \boldsymbol{w}_t - \boldsymbol{u}, \boldsymbol{x}_t^k\rangle - \eta_t^2 \|\boldsymbol{x}_t^k\|^2. \end{aligned} \tag{4.3}$$

Since we assume $l_{M_t} > 0$, it follows that $l_{M_t} = 1 - y_t \langle \boldsymbol{w}_t, \boldsymbol{x}_t^k\rangle$. Then $y_t \langle \boldsymbol{w}_t, \boldsymbol{x}_t^k\rangle = 1 - l_{M_t}$. Moreover, the definition of the loss function gives $l_t^* \geq 1 - y_t \langle \boldsymbol{u}, \boldsymbol{x}_t^k\rangle$. Then $y_t \langle \boldsymbol{w}_t, \boldsymbol{x}_t^k\rangle \geq 1 - l_t^*$.

Substituting these relations into Equation (4.3), we obtain:

$$\begin{aligned}\triangle_t &\geq 2\eta_t \left((1 - l_t^*) - (1 - l_{M_t})\right) - \eta_t^2 \|\boldsymbol{x}_t^k\|^2 \\ &= \eta_t \left(2l_{M_t} - \eta_t \|\boldsymbol{x}_t^k\|^2 - 2l_t^*\right). \end{aligned} \tag{4.4}$$

Finally, summing $\triangle_t$ over $t = 1, \ldots, T$, and comparing the lower bound in Equation (4.4) with the upper bound in Equation (4.2), the lemma is proved. We now proceed to prove a loss bound for our algorithm. □

Having established Lemma 4.1, we can confidently proceed with the analysis of our algorithm's performance. The lemma provides a critical inequality that sets the groundwork for bounding the cumulative squared loss. This bound will enable us to demonstrate that our algorithm performs competitively in comparison to a fixed, arbitrary classification rule. The next step is to utilize this lemma to establish an upper bound on the algorithm's loss, thereby affording us a theoretical performance guarantee.

Theorem 4.2. Let $(\hat{\boldsymbol{x}}_1, y_1), \ldots, (\hat{\boldsymbol{x}}_T, y_T)$ be a sequence of samples where $\hat{\boldsymbol{x}}_t \in \mathbb{R}^n$ and $y_t \in \{+1, -1\}$ for all t, and $\|\hat{\boldsymbol{x}}_t\| \leq L$ for all t and for some $L > 0$. Then, for any vector $\boldsymbol{u} \in \mathbb{R}^n$, the cumulative squared loss of our algorithm on this sequence of samples is bounded from above by,

$$\sum_{t=1}^{T} l_{M_t}^2 \leq \left(\|\boldsymbol{u}\| L + 2\sqrt{\sum_{t=1}^{T} (l_t^*)^2}\right)^2$$

Proof . In lemma 4.1, we have:

$$\sum_{t=1}^{T} \eta_t (2l_{M_t} - \eta_t \|\boldsymbol{x}_t^k\|^2 - 2l_t^*) \leq \|\boldsymbol{u}\|^2 \tag{4.5}$$

We know $\eta_t = \dfrac{l_{M_t}}{\|\boldsymbol{x}_t^k\|^2}$. By substituting in equation (4.5), we have:

$$\begin{aligned}\sum_{t=1}^{T} \frac{l_{M_t}}{\|\boldsymbol{x}_t^k\|^2}(2l_{M_t} - \frac{l_{M_t}}{\|\boldsymbol{x}_t^k\|^2}\|\boldsymbol{x}_t^k\|^2 - 2l_t^*) &= \sum_{t=1}^{T} \frac{l_{M_t}^2}{\|\boldsymbol{x}_t^k\|^2} - 2\sum_{t=1}^{T} \frac{l_{M_t} \cdot l_t^*}{\|\boldsymbol{x}_t^k\|^2} \\ &\leq \|\boldsymbol{u}\|^2\end{aligned}$$

Now, using the fact that $\|\boldsymbol{x}_t^k\|^2 \leq L^2$ for all t, we get:

$$\sum_{t=1}^{T} \frac{l_{M_t}^2}{L^2} \leq 2\sum_{t=1}^{T} \frac{l_{M_t} \cdot l_t^*}{L^2} + \|\boldsymbol{u}\|^2$$

Multiplying both sides of this inequality by L^2 implies that:

$$\sum_{t=1}^{T} l_{M_t}^2 \leq 2\sum_{t=1}^{T} l_{M_t} \cdot l_t^* + (\|\boldsymbol{u}\|^2 \cdot L^2)$$

Using the Cauchy-Schwarz inequality to upper bound the right-hand side of the above inequality, and denoting:

$$L_T = \sqrt{\sum_{t=1}^{T} l_{M_t}^2} \quad \text{and} \quad U_T = \sqrt{\sum_{t=1}^{T} (l_t^*)^2} \tag{4.6}$$

we get:

$$L_T^2 \leq (\|\boldsymbol{u}\|^2 \cdot L^2) + 2L_T U_T \tag{4.7}$$

The largest value of L_T for which the inequality (4.6) is satisfied is equal to the largest of two values for which this inequality holds with equality. Thus, in order to obtain the upper bound on L_T, we must obtain the largest root of the quadratic equation $L_T^2 - \|\boldsymbol{u}\|^2 \cdot L^2 - 2L_T U_T = 0$. By solving this equation, the largest root is as follows:

$$U_T + \sqrt{U_T^2 + \|\boldsymbol{u}\|^2 \cdot L^2}$$

Using the fact that $\sqrt{a+b} \leq \sqrt{a} + \sqrt{b}$, we conclude that:

$$\begin{aligned} &\sqrt{U_T^2 + \|\boldsymbol{u}\|^2 \cdot L^2} \leq \sqrt{U_T^2} + \sqrt{\|\boldsymbol{u}\|^2 \cdot L^2} \\ &\Rightarrow U_T + \sqrt{U_T^2 + \|\boldsymbol{u}\|^2 \cdot L^2} \leq U_T + U_T + \|\boldsymbol{u}\| L \\ &\Rightarrow L_T \leq \|\boldsymbol{u}\| L + 2U_T \end{aligned} \tag{4.8}$$

To obtain the desired bound, we can take the square of both sides of this inequality and substitute the definitions of L_T and U_T from equation (4.6). □

In this section, we derived key theoretical results that give insights into our algorithm's effectiveness. By first proving Lemma 4.1, we established a foundational result that allowed us to derive an upper bound on the cumulative squared loss of the algorithm. This bound illustrates that the algorithm can achieve a loss comparable to an optimal fixed classifier within a certain range. The theoretical analysis presented here serves as a testament to the robustness and reliability of the proposed algorithm, underlining its potential efficacy in a wide array of practical scenarios.

5 Experimental Results

In this section, we compare the proposed binary classification algorithm with various online kernel learning algorithms. Since our proposed method extends the Passive-Aggressive algorithm for interval data and no other online algorithm is specifically designed for classifying interval data, we compare it with extended versions of the Passive-Aggressive algorithm. These algorithms, while not designed for interval data, have been adapted to handle nonlinear cases and reduce the number of support vectors. We evaluate these algorithms on standard UCI datasets and real-world datasets of varying sizes and complexities. The algorithms selected for comparison include:

- The kernelized Perceptron without a budget [16]
- The online kernelized gradient descent without a budget, "OGD" [23]
- The random budget perceptron using a random removal strategy, "RBP" [5]
- The Forgetron, which removes the oldest support vectors (SVs) [15]
- The Projectron, which employs a projection strategy [28]
- The Projectron++, an improved version of Projectron with a projection strategy [29]
- Budget Passive-Aggressive with a simple removal strategy, "BPA-S" [39]
- Budget Online Gradient Descent with a removal strategy, "BOGD" [43]
- Fourier Online Gradient Descent, "FoGD" [20, 27]

- Nyström Online Gradient Descent, "NysGD" [20, 27]

Among the above algorithms, only FoGD and NysGD combine online kernel learning methods with kernel approximation techniques. Cumulative error and test error are used to measure the efficiency of the algorithms, along with runtime. The cumulative error is defined as: $\sum_{t=1}^{T} l_t^2$, which quantifies the accuracy of the learning model over a sequence of training data points over time. Since the initial prediction of weights in the early stages often leads to misguided predictions, the cumulative error is typically greater than the errors observed during training and testing. For a fair comparison between the methods, the parameters are set as follows:

- The budget size for all datasets is set to 200.
- A Gaussian kernel is used for nonlinear algorithms, and hyperparameter tuning [3] is applied to achieve optimal performance on a given dataset. This includes tuning hyperparameters such as σ in the Gaussian kernel and the learning rate η.
- The search intervals for model hyperparameters are as follows:
  - $\sigma \in [0.01, 15]$
  - The learning rate $\eta \in [0.01, 0.2]$
  - The number of Random Fourier Features (RFFs) for the FoGD algorithm is $RFFs \in [50, 200]$
  - For the BPA-S algorithm, $C_{BPAs} \in [0.5, 2]$
  - For the NysGD algorithm, the 20 to 50 largest magnitude eigenvalues are considered.

For the proposed method, the data is converted into interval data as follows:

$$\boldsymbol{x}_t^l = \boldsymbol{x}_t - 0.02 \times \boldsymbol{x}_t \quad \text{(lower bound)}, \qquad \boldsymbol{x}_t^u = \boldsymbol{x}_t + 0.03 \times \boldsymbol{x}_t \quad \text{(upper bound)},$$

where $\Delta_1 = 0.02$ and $\Delta_2 = 0.03$ represent the lower and upper bounds, respectively. In contrast, the other methods use the original data $\boldsymbol{x}_t$ without any modifications. Additionally, the Hinge loss function is employed, which is a widely used loss function in classification problems. These methods are evaluated on binary classification tasks across a variety of datasets. The 10-fold cross-validation method is selected to evaluate the algorithms. Without loss of generality, the data is assumed to be irregularly entered into the algorithms. We evaluate the performance of the presented algorithm for binary classification on datasets of different sizes and applications selected from the UCI database[1] and the LIBSVM database[2]. The general properties of these datasets are summarized in Table 1.

Dataset	Samples	Features
ionosphere	351	33
wdbc	569	30
australian	690	14
chess	3196	36
twonorm	7400	20
coil2000	9822	85
magic	19020	10

Table 1: Description of binary classification datasets.

The results of the 10-fold cross-validation include the mean and standard deviation (mean ± std) of cumulative and test errors, which are detailed in Tables 2, 3, and 4. Additionally, the average runtime is reported in these tables as "runtime." The comparative results demonstrates that the proposed algorithm achieves the best or near-best results across all datasets. Notably, the proposed algorithm exhibits superior performance in the nonlinear case compared to the linear case, where other comparison algorithms are also implemented in the nonlinear case. Compared to competing methods, the proposed approach avoids employing a budget maintenance strategy, as the reduction of support vectors is not within the scope of our current objectives. As a consequence, this results in slower runtime performance in nonlinear scenarios. Conversely, in the linear case, where maintaining support vectors is unnecessary, the proposed

[1] https://archive.ics.uci.edu/datasets/
[2] https://www.csie.ntu.edu.tw/~cjlin/libsvmtools/datasets/

method demonstrates superior runtime performance compared to alternative approaches. For the Australian, Chess, and Coil2000 datasets, the distinction between the proposed method and other algorithms is pronounced, with the former outperforming the latter. In specific datasets, such as Magic, while the proposed method incurs marginally higher training errors compared to some algorithms, the cumulative error remains lower than that of all competing algorithms.

To facilitate a deeper understanding of the algorithm implementation results, the learning curve for all algorithms is shown in Figure 2. The horizontal axis represents the number of received training data along with the scale factor, which divides the axis into 50 equal parts. The vertical axis represents the average cumulative error on the training data sequences. Perceptron, RBP, Forgetron, Projectron, and Projectron++ algorithms have similar update rules but use different strategies for maintaining the budget. As shown in Figure 2, the learning curve of these algorithms is very close to each other in some datasets, resulting in overlapping curves at specific points.

	ionosphere			wdbc			australian		
Algorithm's Name	Test Error	Cumulative Error	Runtime	Test Error	Cumulative Error	Runtime	Test Error	Cumulative Error	Runtime
Proposed (Linear)	21.94% ± 14.07	23.05% ± 1.27	0.0062	21.82% ± 8.23	34.52% ± 1.11	0.0092	19.42% ± 9.52	20.37% ± 0.85	0.0224
Proposed (Kernel)	19.33% ± 6.68	23.71% ± 2.82	0.1226	23.02% ± 16.00	35.85% ± 2.03	0.1542	17.25% ± 3.89	17.95% ± 1.00	1.1565
FoGD	50.13% ± 6.68	51.82% ± 2.44	0.01288	49.40% ± 7.31	47.74% ± 2.32	0.0255	49.71% ± 5.84	50.97% ± 1.20	0.1106
Perceptron	27.65% ± 14.41	27.95% ± 1.93	0.0230	19.51% ± 16.18	35.52% ± 1.52	0.0472	20.43% ± 8.92	20.84% ± 1.21	0.2170
ogd	19.96% ± 11.61	23.49% ± 1.39	0.0279	20.04% ± 16.19	35.66% ± 1.64	0.0596	16.38% ± 3.99	18.53% ± 1.16	0.6588
rbp	27.65% ± 14.41	27.95% ± 1.93	0.0246	33.05% ± 18.68	38.92% ± 2.47	0.0651	21.74% ± 8.20	22.62% ± 1.50	0.2063
Forgetron	26.51% ± 14.65	27.83% ± 2.01	0.0239	19.51% ± 16.18	35.52% ± 1.52	0.0611	18.55% ± 5.58	22.90% ± 1.21	0.1462
Projectron	27.65% ± 14.41	27.95% ± 1.93	0.0333	29.36% ± 19.53	41.52% ± 1.73	0.0351	19.13% ± 8.53	20.84% ± 1.13	0.1757
Projectron++	24.80% ± 12.52	28.11% ± 1.49	0.0494	29.36% ± 19.53	41.52% ± 1.73	0.0364	17.83% ± 6.70	20.76% ± 1.04	0.2742

Table 2: Part I: Comparison of cumulative error, test error and runtime of binary classification datasets

	chess			twonorm		
Algorithm's Name	Test Error	Cumulative Error	Runtime	Test Error	Cumulative Error	Runtime
Proposed (Linear)	13.24% ± 7.87	13.08% ± 7.31	0.0536	3.58% ± 1.10	3.57% ± 0.15	0.0946
Proposed (Kernel)	10.75% ± 5.89	11.23% ± 5.56	4.6392	2.08% ± 0.71	2.27% ± 0.08	7.9806
FoGD	37.11% ± 0.63	47.49% ± 3.14	0.3251	50.12% ± 1.72	50.12% ± 0.52	0.6994
Perceptron	18.27% ± 12.97	19.05% ± 13.33	1.2860	3.39% ± 1.33	3.71% ± 0.20	0.7646
ogd	11.39% ± 9.17	11.24% ± 8.38	1.4548	2.08% ± 0.72	2.26% ± 0.10	3.4382
rbp	26.36% ± 12.78	26.29% ± 13.10	1.3012	3.72% ± 1.42	3.85% ± 0.26	0.7949
Forgetron	28.39% ± 17.10	29.17% ± 17.52	1.3282	3.57% ± 1.08	4.02% ± 0.17	0.6897
Projectron	21.42% ± 16.99	20.16% ± 15.54	1.4048	3.34% ± 1.36	3.71% ± 0.20	1.5331
Projectron++	26.38% ± 15.94	25.04% ± 14.85	1.3495	3.45% ± 0.87	3.66% ± 0.21	1.2582

Table 3: Part II: Comparison of cumulative error, test error and runtime of binary classification datasets

	coil2000			magic		
Algorithm's Name	Test Error	Cumulative Error	Runtime	Test Error	Cumulative Error	Runtime
Proposed (Linear)	10.65% ± 14.12	10.72% ± 0.25	0.2282	30.40% ± 3.26	38.28% ± 0.36	0.2524
Proposed (Kernel)	5.97% ± 0.05	6.61% ± 0.34	25.2084	32.61% ± 5.88	36.40% ± 0.32	34.7537
FoGD	48.98% ± 1.60	48.58% ± 0.49	0.8844	50.05% ± 1.20	49.86% ± 0.27	1.8027
Perceptron	7.12% ± 1.38	10.80% ± 0.16	12.3949	34.58% ± 10.56	36.48% ± 0.28	16.9545
ogd	5.97% ± 0.05	6.45% ± 0.07	13.3811	30.97% ± 3.78	36.39% ± 0.24	16.2899
rbp	6.73% ± 2.39	11.10% ± 0.33	2.2514	36.89% ± 11.68	38.36% ± 0.45	4.7109
Forgetron	6.66% ± 1.96	10.87% ± 0.19	7.5202	35.28% ± 11.07	37.49% ± 0.32	6.1794
Projectron	12.33% ± 4.02	12.99% ± 2.15	4.0715	32.56% ± 5.05	36.77% ± 1.10	6.6049
Projectron++	12.33% ± 4.02	12.99% ± 2.15	4.4198	32.56% ± 5.05	36.77% ± 1.10	7.1136

Table 4: Part III: Comparison of cumulative error, test error and runtime of binary classification datasets

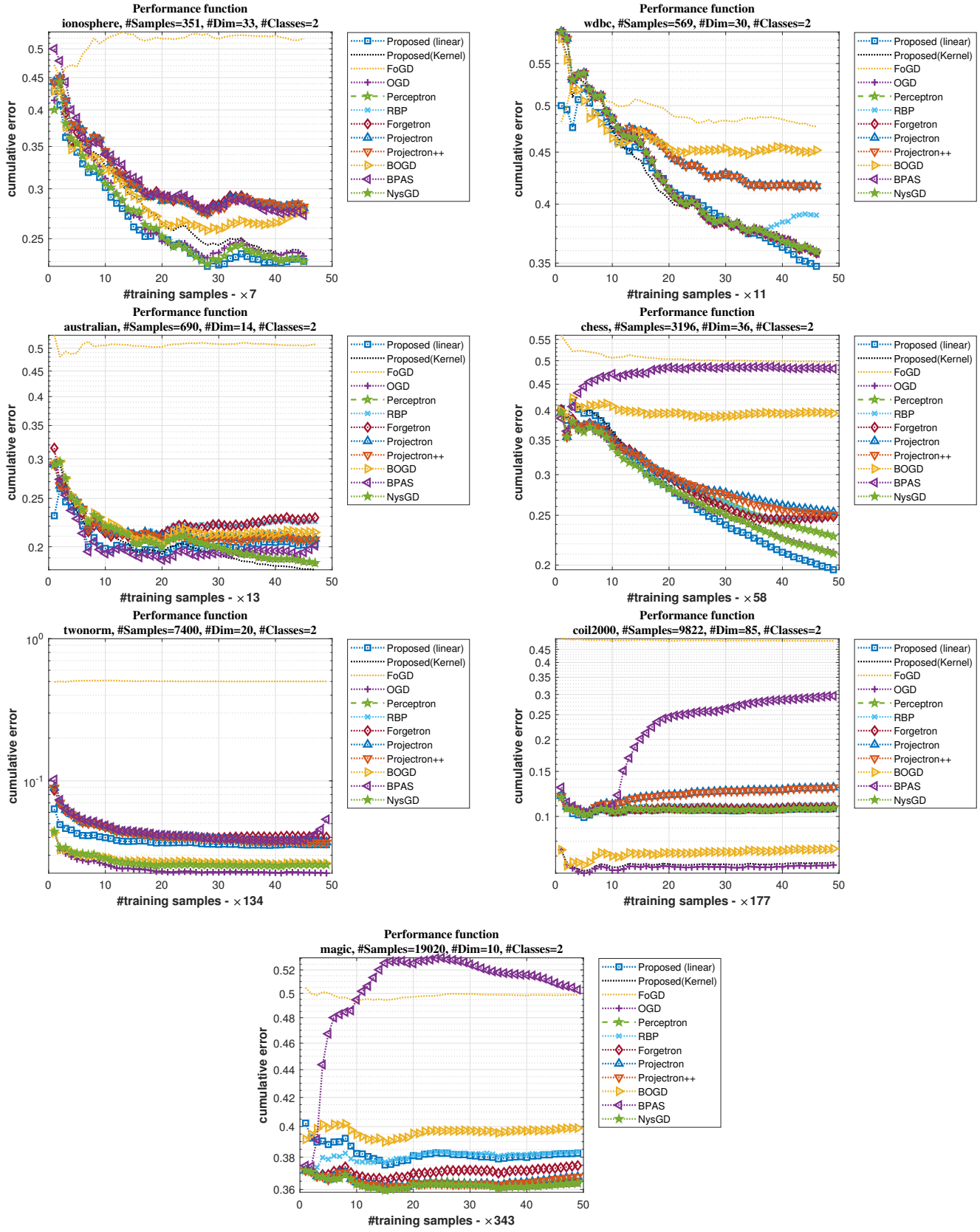

Figure 2: Learning curve of all algorithms for binary datasets

In summary, the proposed algorithm demonstrates robust performance across various datasets, particularly excelling in nonlinear scenarios. While it incurs marginally higher training errors in some cases, its cumulative error remains consistently lower than competing methods. The learning curves in Figure 2 highlight the close performance of budget-maintaining algorithms, while the proposed method avoids such strategies, focusing instead on accuracy. These results underscore the effectiveness of the proposed approach for binary classification tasks, especially when dealing with interval data. Future work could explore integrating budget maintenance techniques to further enhance runtime efficiency without compromising accuracy.

Conclusion

This paper introduced a novel online binary classification algorithm designed to handle interval data, extending the Passive-Aggressive (PA) framework with a new loss function. The proposed method effectively addresses the limitations of traditional online learning algorithms when dealing with imprecise data represented as intervals. Theoretical guarantees, including cumulative squared loss bounds and relative loss bounds, demonstrate the algorithm's effectiveness in minimizing prediction errors. Empirical evaluations on multiple datasets confirm the competitive performance of the proposed method compared to existing online algorithms. In future work, the method can be extended to multi-class and multi-label classification problems, and techniques for reducing the number of support vectors (SVs) can also be investigated to enhance computational efficiency and scalability.

References

[1] C. Angulo, D. Anguita, L. Gonzalez-Abril, and J. A. Ortega, *Support vector machines for interval discriminant analysis*, Neurocomputing **71** (2008), no. 7, 1220–1229.

[2] M. Baymani, N. Salehi-M., and A. Mansoori, *Applying norm concepts for solving interval support vector machine*, Neurocomputing **311** (2018), 41–50.

[3] J. Bergstra, R. Bardenet, Y. Bengio, and B. Kégl, *Algorithms for hyper-parameter optimization*, Advances in neural information processing systems, vol. 24, 2011.

[4] A. Blanco-Fernandez, A. Colubi, and M. García-Bárzana, *A set arithmetic-based linear regression model for modelling interval-valued responses through real-valued variables*, Inf. Sci. **247** (2013), 109–122.

[5] G. Cavallanti, N. Cesa-Bianchi, and C. Gentile, *Tracking the best hyper-plane with a simple budget perceptron*, Machine Learn. **69** (2007), no. 2, 143–167.

[6] Y. Chen and J. Z. Wang, *Support vector learning for fuzzy rule-based classification systems*, IEEE Trans. Fuzzy Syst. **11** (2003), no. 6, 716–728.

[7] Z. Chen, *Robust sparse online learning through adversarial sparsity constraints*, 2024 9th IEEE Int. Conf. Smart Cloud (SmartCloud), IEEE, 2024, pp. 42–47.

[8] L. Cheng, D. Schuurmans, S. Wang, T. Caelli, and S. Vishwanathan, *Implicit online learning with kernels*, Advances in neural information processing systems, MIT Press, 2007, pp. 249–256.

[9] W.-Y. Cheng and C.-F. Juang, *An incremental support vector machine-trained TS-type fuzzy system for online classification problems*, Fuzzy Sets and Systems **163** (2011), no. 1, 24–44.

[10] C. Cortes and V. Vapnik, *Support-vector networks*, Machine Learn. **20** (1995), no. 3, 273–297.

[11] K. Crammer, O. Dekel, J. Keshet, S. Shalev-Shwartz, and Y. Singer, *Online passive-aggressive algorithms*, J. Machine Learn. Res. **7** (2006), 551–585.

[12] K. Crammer, J. Kandola, and Y. Singer, *Online classification on a budget*, Advances in Neural Information Processing Systems, vol. 16, MIT Press, Cambridge, MA, 2004, pp. 225–232.

[13] F.D.A. de Carvalho, P. Brito, and H.-H. Bock, *Dynamic clustering for interval data based on l2 distance*, Comput. Statist. **21** (2006), no. 2, 231–250.

[14] R.M. de Souza and F.D. A. De Carvalho, *Clustering of interval data based on city–block distances*, Pattern Recog. Lett. **25** (2004), no. 3, 353–365.

[15] O. Dekel, S. Shalev-Shwartz, and Y. Singer, *The forgetron: A kernel-based perceptron on a fixed budget*, Advances in Neural Information Processing Systems, vol. 18, 2005, pp. 259–266.

[16] H. Edelstein, *Introduction to data mining and knowledge discovery*, 3rd ed., Two Crows Corporation, Potomac, MD, USA, 1999.

[17] Y. Forghani and H. S. Yazdi, *Robust support vector machine-trained fuzzy system*, Neural Networks **50** (2014), 154–165.

[18] Y. Freund and R. E. Schapire, *Large margin classification using the perceptron algorithm*, Machine Learn. **37** (1999), no. 3, 277–296.

[19] S.C. Hoi, D. Sahoo, J. Lu, and P. Zhao, *Online learning: A comprehensive survey*, arXiv preprint arXiv:1802.02871 (2018).

[20] S. C. H. Hoi, J. Wang, P. Zhao, J. Zhuang, and Z. Liu, *Large scale online kernel classification*, Int. Joint Conf. Artific. Intell., IJCAI/AAAI, 2013, pp. 1750–1756.

[21] E. Hüllermeier, *Learning from imprecise and fuzzy observations: Data disambiguation through generalized loss minimization*, Int. J. Approx. Reason. **55** (2014), no. 7, 1519–1534.

[22] J. Kivinen, A.J. Smola, and R.C. Williamson, *Online learning with kernels*, Advances in neural information processing systems, 2002, pp. 785–792.

[23] J. Kivinen, A.J. Smola, and R.C. Williamson, *Online learning with kernels*, IEEE Trans. Signal Process. **52** (2004), no. 8, 2165–2176.

[24] J. Langford, L. Li, and T. Zhang, *Sparse online learning via truncated gradient*, J. Machine Learn. Res. **10** (2009), 777–801.

[25] S. Lee, Y. Liao, M.H. Seo, and Y. Shin, *Fast and robust online inference with stochastic gradient descent via random scaling*, Proc. AAAI Conf. Artific. Intell., vol. 36, 2022, pp. 7381–7389.

[26] Y. Liu, X. Fan, W. Li, and Y. Gao, *Online passive-aggressive active learning for trapezoidal data streams*, IEEE Trans. Neural Networks Learn. Syst. **34** (2022), no. 10, 6725–6739.

[27] J. Lu, S.C. Hoi, J. Wang, P. Zhao, and Z.-Y. Li, *Large scale online kernel learning*, J. Machine Learn. Res. **17** (2016), 1–43.

[28] F. Orabona, J. Keshet, and B. Caputo, *The projectron: A bounded kernel-based perceptron*, Proc. 25th Int. Conf. Machine learn., ACM, 2008, pp. 720–727.

[29] F. Orabona, J. Keshet, and B. Caputo, *Bounded kernel-based online learning*, J. Machine Learn. Res. **10** (2009), 2643–2666.

[30] W. Qin and X. Luo, *Asynchronous parallel fuzzy stochastic gradient descent for high-dimensional incomplete data representation*, IEEE Trans. Fuzzy Syst. **32** (2023), no. 2, 445–459.

[31] A.B. Ramos-Guajardo and P. Grzegorzewski, *Distance-based linear discriminant analysis for interval-valued data*, Inf. Sci. **372** (2016), 591–607.

[32] F. Rosenblatt, *The perceptron: A probabilistic model for information storage and organization in the brain*, Psychol. Rev. **65** (1958), no. 6, 386–408.

[33] B. Schölkopf and A.J. Smola, *Learning with Kernels: Support Vector Machines, Regularization, Optimization, and Beyond*, MIT Press, Cambridge, MA, USA, 2001.

[34] J. Shawe-Taylor and N. Cristianini, *Kernel Methods for Pattern Analysis*, Cambridge University Press, New York, NY, USA, 2004.

[35] V. Sindhwani, P. Niyogi, and M. Belkin, *Beyond the point cloud: From transductive to semi-supervised learning*, Proc. 22nd Int. Conf. Machine Learn., ACM, 2005, pp. 824–831.

[36] L. V. Utkin, A. I. Chekh, and Y. A. Zhuk, *Binary classification SVM-based algorithms with interval-valued training data using triangular and Epanechnikov kernels*, Neural Networks **80** (2016), 53–66.

[37] L. Wang, H.-B. Ji, and Y. Jin, *Fuzzy passive–aggressive classification: A robust and efficient algorithm for online*

classification problems, Inf. Sci. **220** (2013), 46–63.

[38] Z. Wang, K. Crammer, and S. Vucetic, *Breaking the curse of kernelization: Budgeted stochastic gradient descent for large-scale SVM training*, J. Machine Learn. Res. **13** (2012), no. 1, 3103–3131.

[39] Z. Wang and S. Vucetic, *Online passive-aggressive algorithms on a budget*, Proc. Thirteenth Int. Conf. Artific. Intell. Statist., vol. 9, 2010, pp. 908–915.

[40] C.-H. Wu, H. Horng-Shing Lu, and H.-M. Hang, *Budgeted passive-aggressive learning for online multiclass classification*, IEEE Access **8** (2020), 227420–227437.

[41] T. Zhai and H. Wang, *Online passive-aggressive multilabel classification algorithms*, IEEE Trans. Neural Networks Learn. Syst. **34** (2022), no. 12, 10116–10129.

[42] Y, Zhang, *Learning label correlations for multi-label online passive aggressive classification algorithm*, Wuhan Univ. J. Natur. Sci. **29** (2024), no. 1, 51–58.

[43] P. Zhao, J. Wang, P. Wu, R. Jin, and S.C. Hoi, *Fast bounded online gradient descent algorithms for scalable kernel-based online learning*, Proc. 29th Int. Conf. Machine Learn., Omnipress, New York, NY, USA, 2012, pp. 169–176.

[44] Z. Zhou, W.-S. Zheng, J.-F. Hu, Y. Xu, and J. You, *One-pass online learning: A local approach*, Pattern Recog. **51** (2016), 346–357.